Associação Propagadora Esdeva
Centro de Ensino Superior de Juiz de Fora – CES/JF
Curso de Engenharia de Software
Projeto de Iniciação Científica

APLICAÇÕES MULTIPLATAFORMA COM REACT

Desenvolvimento de uma Aplicação Educacional

Rogério Nogueira Tostes[1]
Centro de Ensino Superior de Juiz de Fora, Juiz de Fora, MG
Romualdo Monteiro de Resende Costa[2]
Centro de Ensino Superior de Juiz de Fora, Juiz de Fora, MG

RESUMO

Este trabalho apresenta uma proposta de desenvolvimento de um aplicativo que oferece informações educacionais através de diferentes plataformas, com um único código. A aplicação pode ser executada através do navegador, na Web, bem como em dispositivos móveis, nos sistemas Android e iOS. Para o desenvolvimento, foi utilizado o React, uma biblioteca Javascript que favorece o desenvolvimento multiplataforma e que, no caso dos dispositivos móveis, permite que as aplicações sejam executadas de forma nativa, graças a uma outra biblioteca, conhecida como React Native. Dessa forma, com alterações apenas na representação visual que são tratadas neste trabalho, uma mesma aplicação pode ser executada em diferentes ambientes.

Palavras-chave: Aplicações móveis. Multiplataforma. React. React Native. Javascript.

1 INTRODUÇÃO

A computação móvel é suportada pelo uso de dispositivos móveis, que se tornam cada vez mais comuns e com diversas funcionalidades. Esses dispositivos, atualmente, dispõem de uma capacidade considerável de armazenamento e processamento, com recursos para comunicação sem fio, localização, câmeras e diversas outras funcionalidades cuja operação é suportada por um sistema operacional. Todos esses recursos, quando explorados adequadamente pelas

[1] Discente do Curso de Engenharia de Software, do Centro de Ensino Superior de Juiz de Fora.
[2] Docente do Curso de Engenharia de Software do Centro de Ensino Superior de Juiz de Fora. Orientador.

aplicações, facilitam a realização das mais diversas tarefas pelos usuários, tornando a computação móvel uma tecnologia integrada a vida cotidiana e, até mesmo, indistinguível dela (WEISER, 1991).

Parte da revolução tecnológica que torna a computação móvel tão importante inclui o desenvolvimento de aplicações para essa plataforma. Em alguns casos, essas aplicações podem já existir e estar integradas a serviços oferecidos na Web. Nesses casos, por questões de usabilidade, pode ser mais interessante permanecer utilizando essas aplicações através de computadores, onde acessórios, como telas maiores, teclados e mouses podem favorecer o uso dos serviços. Em outros casos, por fatores diversos, como a mobilidade e a facilidade de acesso a recursos, pode ser necessário que as aplicações estejam disponíveis em dispositivos como celulares e tablets. Pode ser, ainda, que o ambiente ideal seja uma combinação dos casos citados, sendo parte dos recursos das aplicações acessado através de computadores e, outra parte, através de tablets e celulares, aproveitando o melhor que cada tecnologia pode oferecer.

No ambiente educacional, por exemplo, pode ser necessário que diferentes aplicações sejam utilizadas, de acordo com o objetivo a ser alcançado. Para o acesso ao conteúdo, por exemplo, os alunos poderiam utilizar dispositivos móveis como celulares ou tablets, que permitiriam o acesso em diferentes ambientes e a qualquer tempo. Nesse caso, o acesso a recursos disponíveis nesses equipamentos, como o contexto da localização e o acesso à câmera poderiam, inclusive, ser utilizados nas tarefas de aprendizado. Por outro lado, para o uso dos professores, pode ser mais adequado, a fim de facilitar a formatação e a inclusão de conteúdos, o uso de dispositivos como computadores, com recursos de usabilidade que favoreçam a tarefa da produção e divulgação do conteúdo.

Devido a importância do processo de aprendizagem, o emprego de aplicativos educacionais para auxiliar esse processo tem se tornado uma estratégia comum. Esses aplicativos podem auxiliar a tarefa dos estudantes e, principalmente, favorecer a assimilação dos conteúdos ministrados em sala, mas podem também facilitar a tarefa dos educadores na divulgação do conhecimento. Quando adequadamente projetados, devem estimular o raciocínio e o pensamento crítico, apresentar conteúdo relevante ao usuário e possibilitar a exploração de conteúdos adicionais (VIEIRA, 2012). Além disso, podem favorecer a autonomia dos estudantes, suportando diversos perfis de aprendizagem dos alunos (BARBOSA, 2017). No

entanto, para alcançar esses objetivos, além do viés pedagógico, é necessário que as aplicações contemplem aspectos de usabilidade e ergonomia (NUNES, 2018) que podem ser favorecidos quando múltiplas opções de acesso são oferecidas.

Assim, para que os benefícios da computação móvel possam ser obtidos em ambientes complexos, como é o caso do ambiente educacional, pode ser necessário o desenvolvimento de aplicações que funcionem tanto em computadores tradicionais quanto em dispositivos móveis, dos mais diferentes modelos e sistemas operacionais. Nesse cenário, pode ser interessante, quando for possível, reutilizar as aplicações nos diferentes ambientes, evitando o retrabalho no desenvolvimento e, principalmente, na manutenção dessas aplicações.

Diversas estratégias podem ser adotadas para o desenvolvimento de aplicações para dispositivos móveis. Uma dessas estratégias consiste em utilizar linguagens e ferramentas de desenvolvimento específicas para uma determinada plataforma. Esse é o caso, por exemplo, da linguagem Java[3] com um conjunto de bibliotecas desenvolvidas para o sistema operacional Android[4] que, mais recentemente, adotou também o desenvolvimento através da linguagem Kotlin[5]. No caso do sistema iOS[6], podem ser utilizadas, de forma similar, as linguagens Objective-C[7] e Swift[8]. Esse modelo de desenvolvimento, em ambos os sistemas, é conhecido como nativo, uma vez que as especificações dos desenvolvedores são compiladas e interpretadas diretamente em instruções existentes nos sistemas operacionais.

O desenvolvimento nativo para os dispositivos móveis tem como vantagem a utilização plena dos recursos dos dispositivos uma vez que as funcionalidades oferecidas pelos sistemas operacionais são diretamente acessadas. Esse acesso direto também é responsável por uma melhor performance das aplicações, uma vez que não há a necessidade de elementos intermediários entre as instruções e o próprio sistema operacional. No entanto, é necessário o desenvolvimento de uma aplicação para cada sistema operacional existente nos dispositivos móveis, além do desenvolvimento de aplicações para o uso em computadores, se for o caso.

---

[3] https://www.java.com
[4] https://www.android.com
[5] https://kotlinlang.org
[6] https://developer.apple.com/ios
[7] https://developer.apple.com/documentation/objectivec
[8] https://developer.apple.com/documentation/swift

Como alternativa, tecnologias para o desenvolvimento Web podem ser empregadas no desenvolvimento de aplicações que são interpretadas nos dispositivos móveis. Nesse caso, uma única aplicação é desenvolvida e pode ser interpretada, através de componentes nativos, como o WebView (LECHETA, 2018), em diferentes sistemas operacionais. O problema é que, nesse caso, a utilização dessa camada intermediária pode ocasionar dificuldade para acesso aos recursos nativos dos dispositivos, além da queda na performance da execução das aplicações.

A fim de avaliar uma proposta alternativa de desenvolvimento, este trabalho propõe o desenvolvimento de uma aplicação educacional utilizando React[9] que é uma biblioteca Javascript (FLANAGAN, 2011) para a construção de interfaces que podem ser adicionadas a uma página Web. Além de ser executada nos navegadores, aplicações React também podem ser executadas localmente, inclusive em dispositivos móveis, onde são conhecidas como React Native[10]. Nesse caso, os componentes são traduzidos diretamente para os componentes oferecidos pelo sistema operacional local. Além disso, as aplicações React também podem ser executadas localmente, através de técnicas conhecidas como PWA (*Progressive Web Apps*)[11]. Nesse caso, as aplicações não são totalmente locais, mas representam uma independência de conectividade, pois são aprimoradas para trabalhar desconectadas (*off-line*), através, por exemplo, da utilização de um sistema de armazenamento local.

Assim, o estudo das aplicações desenvolvidas com React permite comparar diferentes estratégias para o desenvolvimento, sobretudo para dispositivos móveis, principalmente em ambientes complexos, que podem exigir múltiplas funcionalidades, como é o caso do ambiente educacional que será abordado neste trabalho. Nesta proposta, uma aplicação educacional é desenvolvida, permitindo explorar as funcionalidades da biblioteca React. A próxima seção discute os trabalhos relacionados em relação às tecnologias que oferecem suporte à proposta apresentada. A seguir, na terceira seção é apresentada a proposta de construção de uma aplicação para conteúdo educacional. A quarta seção apresenta os resultados

[9] https://reactjs.org
[10] https://facebook.github.io/react-native
[11] https://developers.google.com/web/progressive-web-apps

alcançados com discussões correlatas e é seguida pela última seção, onde são apresentadas as conclusões finais.

2 REFERENCIAL TEÓRICO

2.1 React

React é uma biblioteca para construção de interfaces que teve sua primeira versão publicada em 2013[12]. Projetada para a construção de interfaces, essa biblioteca está intimamente ligada ao desenvolvimento da parte visual das aplicações, também conhecida como "*front-end*". Assim, é usual que essa biblioteca seja empregada em conjunto com outras tecnologias, dependendo do escopo da aplicação.

Para a especificação dos seus componentes é empregada a linguagem JavaScript e uma linguagem declarativa baseada em XML[13] conhecida como JSX[14]. Dessa forma, o React pode ser classificado como uma biblioteca declarativa, uma vez que, principalmente na especificação do design e na apresentação dos seus componentes, a especificação do código segue esse paradigma, onde o programador especifica o que o código deve fazer, ao invés de detalhar as ações para a sua execução. Assim, ao invés de utilizar estruturas de dados, variáveis e estruturas condicionais e de repetição, o código contém elementos e propriedades XML que, de acordo com a sua organização, definem a semântica da aplicação. A Figura 1 apresenta um exemplo de código JSX.

FIGURA 1. Uso do JSX para representar um Componente da Interface.

```
08. ...
09.     var dropdown =
10.         <Dropdown>
11.           Uma lista dropdown
12.           <Menu>
13.             <MenuItem text="Fazer alguma coisa"/>
14.             <MenuItem text="Fazer alguma coisa"/>
15.             <MenuItem text="Fazer alguma coisa"/>
16.           </Menu>
17.         </Dropdown>
18.
19.     render(dropdown);
21. ...
```

Fonte: do autor.

[12] https://reactjs.org/blog/2013/06/05/why-react.html
[13] https://www.w3.org/XML
[14] https://facebook.github.io/jsx

Na Figura 1, a variável *dropdown* é especificada com um conteúdo JSX formado pelos elementos *Dropdown*, *Menu* e *MenuItem*, esses elementos, por sua vez, precisam ser definidos através de elementos da plataforma destino para serem apresentados. No caso do React, a plataforma destino é a Web e, portanto, esses componentes precisam ser especificados como elementos HTML[15].

Elementos no React, como os apresentados na Figura 1 são conhecidos como componentes. Eles gerenciam seu próprio estado e, portanto, são unidades de representação individual da interface do usuário. Para a especificação do funcionamento desses componentes a linguagem JavaScript é empregada. Dessa forma, o programador consegue utilizar toda a representatividade dessa linguagem, de forma complementar às especificações declarativas do JSX.

Em relação ao uso do JavaScript, o React emprega as padronizações especificadas pelo consórcio ECMA[16] (*European Computer Manufactures Association*), conhecidas como ECMAScript 5 e 6[17] ou, mais resumidamente, como ES5 e, principalmente, o ES6. A Figura 2, como exemplo, apresenta a especificação de parte do componente *MenuItem* apresentado, inicialmente, na Figura 1. Esse componente é especificado como uma *Arrow Function*[18], uma forma simplificada de definir uma função sem usar, por exemplo, a palavra *function*. Essa é uma construção representativa da sintaxe definida no ES6.

FIGURA 2: Especificação do Componente MenuItem no React.

```
28. ...
29.     const Menu = (props) => {
30.         const {text} = props;
31.
32.         return (
33.           <h1>{this.text}</h1>
34.         )
35.     };
36. ...
```

Fonte: do autor.

Apesar de especificada em 2015, o ES6 ainda não é suportado por muitos navegadores. Para resolver essa questão, tradutores podem ser empregados para

[15] https://www.w3.org/html
[16] https://www.ecma-international.org
[17] https://es6-features.org
[18] https://es6-features.org

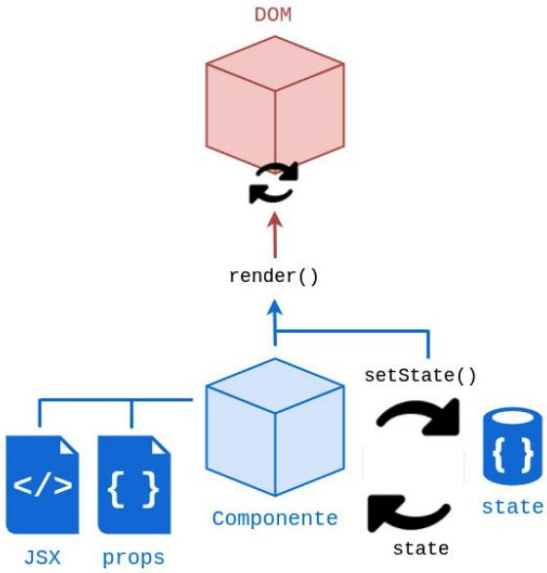
converter o código E6 para a versão 5 antes da publicação da aplicação[19]. Ainda que a versão seis fosse suportada por todos os navegadores, a conversão do código antes da publicação ainda seria necessária, uma vez que a especificação JSX não é padronizada pelo W3C[20] e, portanto, não é suportada pelos navegadores. Assim, para distribuição final das aplicações, o código JSX deve ser convertido para JavaScript, em uma versão compatível com a existente nos principais navegadores.

JavaScript, particularmente na versão ES6, é considerada uma linguagem de programação funcional. Nessa linguagem funções são elementos de primeira ordem. Isso significa que funções tem a mesma importância que variáveis, por exemplo. Não obstante, o componente apresentado na Figura 2 foi definido como uma função. A programação funcional é parte do paradigma de programação declarativa. Assim, tanto na especificação JSX, quanto na especificação ES6, aplicações React especificam o que o programa deve fazer ao invés de especificar como ele deve fazer.

Além de uma biblioteca para especificação da interface do usuário, o React também pode ser considerado um framework da camada de apresentação ao definir componentes que contém, previamente, propriedades, estado e um ciclo de vida previamente determinado. Esse framework define as propriedades como os parâmetros recebidos para customizar a apresentação do componente. Essas propriedades são passadas como atributos dos elementos especificados no código JSX. A Figura 3 apresenta a representação básica desse framework, com as propriedades (props), ao lado da representação do código JSX.

FIGURA 3: Elementos de um Componente React definidos pelo Framework

[19] https://babeljs.io
[20] https://www.w3.org



Fonte: PONTES, 2018.

Na Figura 3, o método *render* aparece em destaque. Esse é o único método obrigatório em todo componente. Quando o componente é construído, esse método é chamado e o seu conteúdo define a forma de apresentação desse componente. Além de ser executado no processo de construção do componente, esse método também deve ser executado por ocasião de modificações sobre o seu estado. O estado, definido também em destaque na Figura 3, representa um conjunto de valores que, toda vez que são alterados, provocam novamente a execução do método *render*. Para destacar as alterações no estado, um método especial, conhecido como *setState*, normalmente é empregado.

Além do *render*, o ciclo de vida dos componentes do React, especificados segundo o ES6, é completado pelos métodos apresentados na Tabela 1.

TABELA 1 - Métodos do Ciclo de Vida da Construção de um Componente React

Método	Contexto
constructor()	*Método invocado quando o componente é montado*
getDerivedStateFromProps()	*Método invocado antes da renderização*
render()	*Método invocado toda vez que o estado muda*
componentDidMount()	*Método invocado depois da renderização*
componentWillUnmount()	*Método invocado quando o componente é removido*

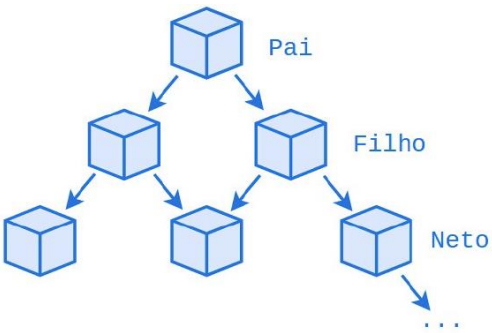
Fonte: do autor.

O método construtor (*constructor*) é usado para a inicialização do componente. É nesse método que o estado é inicializado. Esse método tem acesso as propriedades do componente e, se necessário, pode utilizar esses valores para definir seu estado inicial. Logo após a obtenção das propriedades e inicialização do

estado, o método *getDerivedStateFromProps* é executado. Esse método é chamado antes da apresentação do componente e pode ser utilizado, por exemplo, para a requisição de conteúdos e configurações adicionais que precisem ser realizadas antes da apresentação. Também é possível, ainda nesse método, realizar modificações sobre o estado. Esse método recebe o estado como um argumento e retorna um objeto com as mudanças no estado.

Após a apresentação do componente, o método *componenteDidMount* é chamado. Esse método pode ser utilizado para fazer requisições de conteúdos, porém, toda vez que o estado for modificado nesse método o ciclo de vida será recomeçado e o componente novamente apresentado. Finalmente, o método *componentWillUnmount* é chamado antes do componente ser removido. Essa remoção pode acontecer quando o componente que chamou a apresentação do componente considerado remove esse componente, ou explicitamente, através de métodos definidos no React. A Figura 4 apresenta uma típica hierarquia de componentes no React. Um componente que chama outro componente é chamado de componente pai e, consequentemente, o componente chamado é conhecido como componente filho. Quando um componente pai é removido, todos os componentes filhos são removidos primeiro.

FIGURA 4. Hierarquia de Componentes no React.



Fonte: PONTES, 2017.

Como pode ser observado na Figura 4, o fluxo de dados do React é unidirecional. Através das propriedades, um componente pai pode passar informações para os componentes filhos que, assim por diante, podem repassar essa informação. De maneira similar, considerando que a linguagem segue um paradigma funcional, os componentes filhos podem receber funções dos componentes pais como propriedades e, assim, ao executar essas funções, reportar informações para os seus pais.

Voltando à Tabela 1, os métodos nela apresentados são chamados em decorrência de ações no ciclo de vida de um componente durante a sua construção. Atualizações nesse componente, por outro lado, decorrentes de mudanças no seu estado ou devido ao recebimento de novas propriedades do componente pai definem a chamada dos métodos de atualização, que são apresentados na Tabela 2.

TABELA 2. Métodos do Ciclo de Vida da Atualização de um Componente React

Método	Contexto
getDerivedStateFromProps()	*Método inicial invocado quando o componente é atualizado*
sholdComponentUpdate()	*Método invocado antes da renderização*
render()	*Método invocado toda vez que o estado muda*
getSnapshotBeforeUpdate()	*Método invocado depois da renderização*
componentDidUpdate()	*Método invocado quando o componente é removido*

Fonte: do autor.

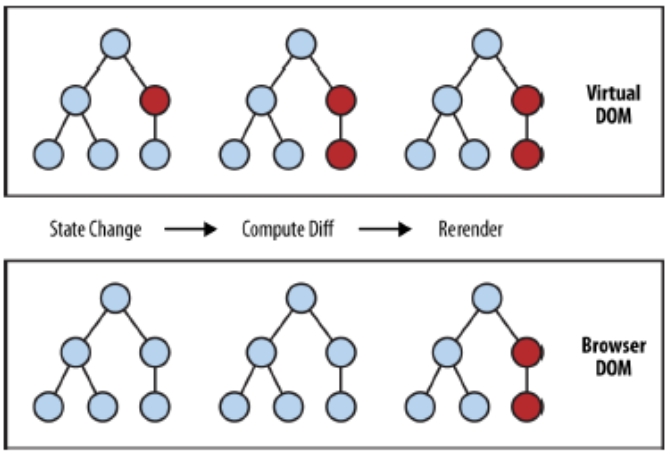
Quando um componente é atualizado, o primeiro método a ser chamado é o *getDerivedStateFromProps*. Esse método recebe o estado e as propriedades de um componente e retorna as modificações que forem necessárias. Nesse método é possível realizar modificações necessárias para preservar valores no estado de um componente, apesar das modificações realizadas. Após a atualização, mas ainda antes da apresentação, o método *sholdComponentUpdate* é chamado. Esse método, quando usado, pode retornar um valor booleano que especifica se o componente deve ser atualizado, sendo o valor padrão verdadeiro. O método *getSnapshotBeforeUpdate* é utilizado nos casos em que é necessário acessar os valores das propriedades e do estado de um componente antes da atualização. Assim, através desse método, mesmo após uma atualização, é possível verificar os valores anteriores do estado do componente. Finalmente, o método *componentDidUpdate* é chamado após a apresentação do componente, isto é, depois que o componente foi novamente apresentado a partir de modificações no seu estado.

O método *render*, apresentado nas Tabelas 1 e 2, é o único, entre todos esses métodos, obrigatório na concepção do componente. Esse método, como já mencionado, realiza a apresentação visual do componente, através da construção do código HTML e de modificações no DOM[21] (*Document Object Model*), que é a interface de acesso utilizada para modificar, individualmente, os elementos que

[21] https://www.w3.org/DOM

constituem a representação visual do navegador Web. Na verdade, ao invés de modificar diretamente a apresentação visual através de modificações no DOM, o React trabalha com um DOM virtual, que atua como uma camada entre as especificações do desenvolvedor e o resultado final que será apresentado aos usuários. O DOM é representado como uma estrutura de dados em forma de árvore. A Figura 5 ilustra uma modificação em um nó dessa árvore e suas consequências.

FIGURA 5. Exemplo de Modificações no DOM em um Programa React.



Fonte: EISENMAN, 2018.

Documentos HTML seguem uma estrutura hierárquica, onde modificações na organização desses elementos modificam a apresentação nos navegadores. Nessa estrutura, modificações em um elemento poderão provocar alterações na apresentação dos elementos relacionados. Na Figura 5, por exemplo, a parte superior da figura representa a estrutura definida pelo React como DOM Virtual. Nessa estrutura, quando um elemento é modificado através, por exemplo, de modificações no seu estado, os elementos relacionados também são alterados. Uma alteração em um elemento, e as modificações nos elementos relacionados, é representada na Figura 5 em vermelho. O DOM Virtual evita que as alterações referentes ao relacionamento entre os elementos sejam calculadas pelo programa responsável pela apresentação, como é o caso do navegador nas aplicações na Web. Ao contrário, antes da apresentação ser atualizada, todas as modificações são

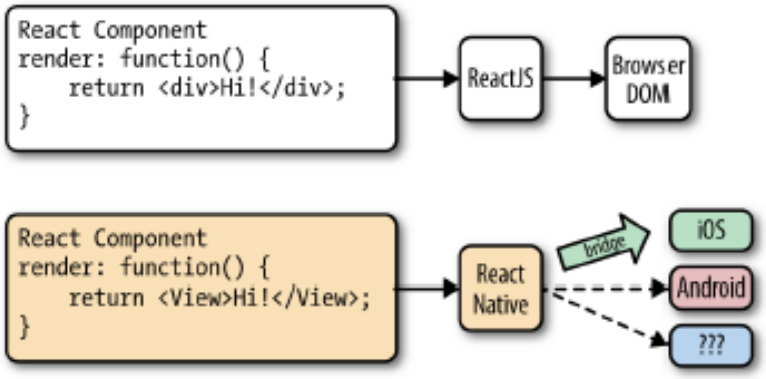
calculadas. Na Figura 5, por exemplo, no DOM do navegador, representado na parte inferior dessa figura, apenas ao final, quando todas as modificações nos elementos relacionados, a partir da modificação inicial, já foram realizadas, é que a apresentação é atualizada. Dessa forma, o React provê um maior controle nas alterações visuais e, eventualmente, pode melhorar o tempo de renderização das aplicações (BANKS, 2017).

2.2 React Native

O desenvolvimento através do React Native além de permitir que as aplicações sejam multiplataforma, uma vez que tecnologias Web, como o Javascript, são empregadas no desenvolvimento, permite também que as aplicações possam ter acesso diretamente aos recursos nativos dos dispositivos, uma vez que não é necessário um elemento intermediário para o acesso a esses componentes, principalmente aqueles relacionados à interface com o usuário que, usualmente, são responsáveis por grande parte dos recursos necessários às aplicações.

As especificações do React Native são as mesmas definidas no React, isto é, as aplicações são escritas de forma declarativa, seja usando as especificações funcionais do Javascript, ou usando as especificações declarativas do JSX. De forma similar, o ciclo de vida das aplicações é o mesmo, isto é, o framework para a construção das aplicações permanece inalterado. Assim, a principal diferença consiste na apresentação visual dos componentes, conforme apresentado na Figura 6. Na parte superior dessa figura é representada a apresentação tradicional do React, onde os componentes são apresentados no navegador Web. Na parte inferior, por outro lado, é representada a apresentação dos componentes em diferentes plataformas, como acontece no React Native.

FIGURA 6. Comportamento da Apresentação da Aplicação React Native em relação ao React.



Fonte: EISENMAN, 2018.

Modificações nos componentes do React Native, de forma semelhante ao React, provocam modificações no DOM virtual mas que, ao contrário do React, não atualizam o DOM da máquina de apresentação. No React Native, o DOM virtual é representado diretamente nos componentes de apresentação nativos do sistema operacional onde a aplicação está sendo executada. Como apresentado na Figura 6, as modificações nos componentes da aplicação React Native, através de uma ponte (*bridge*, na Figura 6), são traduzidas em modificações para a apresentação da plataforma alvo. Dessa forma, aplicações do React Native podem ser apresentadas em qualquer plataforma, desde que seja construída uma interface intermediária que realize a tradução das modificações do componente considerado para as representações na plataforma destino.

Uma vez que as especificações dos componentes no React Native podem ser convertidas em apresentações diversas dependendo da plataforma de destino, ao invés de especificar elementos do HTML para apresentação, como é o caso do React, nessa outra biblioteca são definidos elementos com semântica de apresentação própria (EISENMAN, 2018). A Tabela 3 apresenta, como exemplo, alguns desses elementos.

TABELA 3 - Alguns Elementos definidos no React Native e sua equivalência com HTML.

Componente	HTML	Objetivo
<View>	* *	*Organizar a disposição dos elementos*
<Text>	**	*Especificar informações textuais*
<FlatList>	* *	*Criar uma lista com informações*
<Image>	*<img>*	*Apresentar uma imagem*
<TextInput>	*<input>*	*Receber entrada de dados dos usuários*

Fonte: do autor.

Para organizar a apresentação dos componentes, o React Native define o elemento <*View*>. Esse elemento funciona de maneira semelhante ao <*div*> do HTML permitindo organizar a representação visual dos seus elementos relacionados. Durante a apresentação da aplicação esse elemento deverá ser traduzido para um elemento nativo da plataforma. No caso do sistema operacional Android, por exemplo, esse elemento será traduzido para o elemento homônimo <*View*>. Já no caso do sistema iOS, por outro lado, esse elemento será traduzido para o componente *UIView*.

A Tabela 3 apresenta, como exemplo, alguns outros elementos do React Native, como o elemento <*Text*> que permite a apresentação de conteúdos textuais,

o elemento <*FlatList*> para a construção e apresentação de listas. O elemento <*Image*>, para a apresentação de imagens e o elemento <*TextInput*> que permite ao usuário preencher um campo com informações no formato texto. Diversos outros elementos são oferecidos por essa biblioteca (EISENMAN, 2018).

Construídos para serem apresentados na Web, componentes React são formatados para apresentação usando CSS[22]. No React Native, por outro lado, por não fazer parte, necessariamente, da Web, outras formas de definição do estilo de apresentação podem ser empregadas. Por padrão, a formatação da apresentação no React Native segue uma forma similar a definida pelo CSS, usando um subconjunto dessa tecnologia especificada através de objetos JavaScript.

A forma padronizada para a especificação dos estilos usados na apresentação dos componentes, além de definir apenas um subconjunto das possibilidades do CSS, o que torna mais difícil essa especificação também exige que o programador aprenda uma nova sintaxe, ainda que próxima da sintaxe do CSS. Dessa forma, neste trabalho, as especificações dos estilos foram realizadas usando a própria representação CSS, através da instalação de um módulo conhecido como *styled componentes*[23].

3 METODOLOGIA

Buscando explorar os conceitos do React para o desenvolvimento multiplataforma foi desenvolvida uma aplicação com enfoque educacional. O objetivo dessa aplicação é, por um lado, permitir que os professores possam construir cursos. Esses cursos são formados por seções que, por sua vez, são formados por várias atividades (conteúdos). Por outro lado, os alunos, associados a um ou mais cursos, tem seu desempenho avaliado à medida que realizam as atividades. Quando completam todas as atividades o curso é concluído.

Como o conteúdo é compartilhado entre diferentes usuários (professores e alunos) nessa aplicação é usado um banco de dados distribuído, sendo escolhido o Firebase[24]. Esse banco, além oferecer o armazenamento gratuito para o volume de informações necessário à realização deste trabalho também oferece funcionalidades

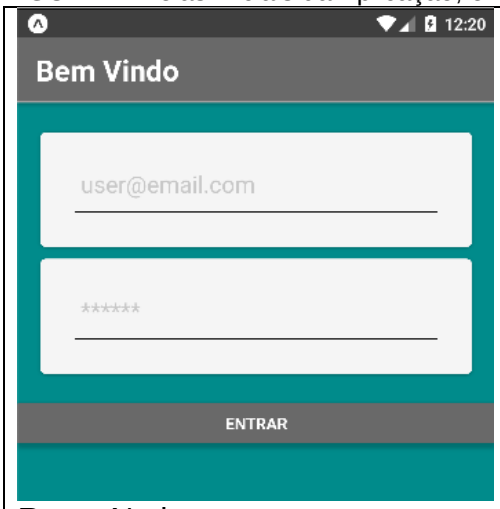
[22] https://www.w3.org/Style/CSS/specs.en.html
[23] https://www.styled-components.com
[24] https://firebase.google.com

adicionais ao armazenamento de dados que são úteis a realização deste trabalho, como o armazenamento de mídias e a autenticação do usuário.

Sobre o armazenamento de dados, o Firebase suporta dois modelos de bancos de dados não relacionais: o *Realtime Database* e o *Cloud Firestore*. No primeiro os dados são armazenados em uma estrutura hierárquica de uma árvore, em um formato muito próximo a um documento JSON[25], sendo o formato original do Firebase. No segundo, os dados são armazenados em dois níveis, sendo o primeiro o conteúdo e o segundo os documentos. Um conteúdo pode ser formado por diversos documentos, cada um, por sua vez, sendo formado por campos. Os campos definem um tipo e um valor. Documentos que formam um conteúdo podem ser formados por campos diferentes. Ambos os formatos suportam atualizações em tempo real e, portanto, atendem aos requisitos do projeto deste trabalho.

Outra funcionalidade que a aplicação utiliza do Firebase consiste no processo de autenticação dos usuários. Na primeira tela da aplicação, representada na Figura 7, o usuário deve fornecer suas credenciais, formadas pelo usuário e senha. Essas informações são necessárias para que, no caso dos alunos, o seu perfil seja identificado e, assim, possam ser selecionados os seus cursos. Através do Firebase o administrador pode controlar os usuários cadastrados. O Firebase oferece ainda facilidades para a administração dos usuários, como a possibilidade de recuperação de senhas.

FIGURA 7. Telas iniciais da Aplicação, em React Native e em React.



React Native

React

Fonte: do autor.

[25] http://www.json.org

Na Figura 7 são apresentadas as telas de autenticação no React Native (a esquerda) funcionando no sistema operacional Android e a tela de autenticação do React (a direita) em execução no navegador Web. Ambas as telas são apresentadas através de um componente definido como *LoginScreen*. Apesar desse componente estar presente em duas aplicações, a estrutura do seu código é a mesma. Esse componente contém os métodos *onChangeMail* e *onChangePassword*, com o mesmo código em ambas as aplicações, que alteram o estado do componente a partir das informações do email e da senha do usuário. Esse componente contém também o método *tryLogin* que é chamado quando o usuário seleciona o botão "Entrar" de qualquer uma das interfaces. Esse método consulta o banco de dados com as informações do estado do componente a respeito do email e da senha do usuário, para validar o seu acesso à aplicação e, para os usuários autenticados, retornar informações do seu contexto. Em relação ao ciclo de vida do componente, ele contém os métodos *constructor*, onde as propriedades e o seu estado são inicializados, o método *componentDidMount*, com as configurações de acesso ao banco de dados e o método *render*, com as especificações de apresentação do componente.

O método *render* apresentado nas Figuras 8 e 9, é o único método do componente *LoginScreen* que possui duas implementações, uma para apresentação no dispositivo móvel e outra para apresentação no navegador. O código para exibição no navegador é apresentado na Figura 8. Nesse caso, a exibição é realizada através de elementos do HTML, para apresentação nos dispositivos móveis, por outro lado, são usados elementos do React Native, conforme apresentado na Figura 9.

FIGURA 8. Método render do Componente LoginPage para o React.

```
01. render() {
02.     return (
03.       <div className="pure-form">
04.         <h1 className="pure-button-primary">Bem Vindo</h1>
05.         <fieldset className="pure-group">
06.           <Input type="email"
07.           className="pure-input-1-1"
08.           placeholder="Email"
09.           onChange={this.onChangeMail.bind(this)}/>
10.         </fieldset>
11.         <fieldset className="pure-group">
12.           <Input type="text"
13.             className="pure-input-1-1"
```

```
14.                 placeholder="Password"
15.                 onChange={(e)=>this.onChangePassword(e)}/>
16.             </fieldset>
17.             <fieldset className="pure-group">
18.             <Button type="submit"
19.                 className="pure-button pure-button-primary"
20.                 onClick={()=>this.tryLogin()}>Entrar</Button>
21.             </fieldset>
22.         </div>
23.     )
24. }
```

Fonte: do autor.

FIGURA 9. Método render do Componente LoginPage para o React Native.

```
01. render() {
02.   return (
03.     <View marginTop="20px">
04.       <TextInput
05.             placeholder="user@email.com"
06.             onChangeText={(value)=>this.onChangeMail(value)}/>
07.       <TextInput
08.             placeholder="******"
09.             secureTextEntry
10.             onChangeText={(value)=>this.onChangePass(value)}/>
11.       <View marginTop="20px" paddingLeft="5px" paddingTop="10px"
12.             paddingRight="5px" >
13.       <Button
14.         title="Entrar"
15.         onPress={()=> this.tryLogin()}
16.       </View>
17.     </View>
18.   );
19. }
```

Fonte: do autor.

Na Figura 8, a formatação da apresentação foi realizada através da biblioteca CSS Pure[26]. Na Figura 9, por outro lado, os elementos do React Native que compõem a apresentação nos dispositivos móveis foram formatados com especificações CSS utilizando a biblioteca *styled components*[27], que permite a especificação direta de código CSS.

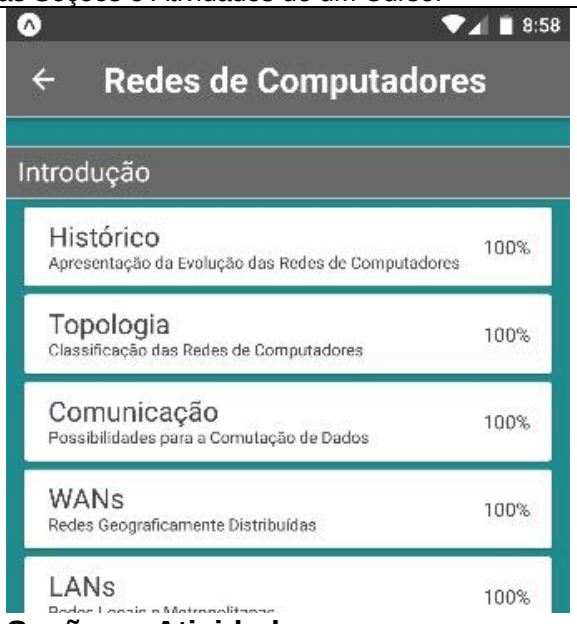
[26] https://purecss.io
[27] https://www.styled-components.com

Apesar do código do método de apresentação dos componentes ser diferente, considerando que a formatação é realizada através do CSS seria possível que a tela das aplicações fosse exatamente a mesma. Essa opção, no entanto, pode não ser desejada, uma vez que os recursos visuais, incluindo o tamanho das telas é diferente podendo ser, portanto, mais interessante, duas apresentações diferentes, cada uma adequada a sua plataforma de exibição.

Ao longo da aplicação, o padrão de desenvolvimento esperado e que foi apresentado na tela inicial se repete. São construídos componentes com o mesmo código, com exceção do método para apresentação do componente, que possui representações visuais para a Web e para dispositivos móveis. Como exemplo, a Figura 10 apresenta as telas dos cursos (a esquerda) e, escolhido um curso, suas seções e atividades (a direita).

FIGURA 10. Telas de Seleção dos Cursos e Tela das Seções e Atividades de um Curso.

Cursos



Seções e Atividades

Fonte: do autor.

4 RESULTADOS E DISCUSSÃO

A construção de aplicações móveis representa uma importante questão nos dias atuais, uma vez que, atualmente, essas aplicações fazem parte das tarefas diárias dos usuários. Assim, é importante desenvolver técnicas e mecanismos que facilitem esse desenvolvimento. Uma possibilidade de aprimorar esse desenvolvimento foi explorada neste trabalho, através do desenvolvimento utilizando a biblioteca React. Através dessa biblioteca, que define também um ciclo de vida

para as aplicações, é possível desenvolver uma mesma aplicação capaz de ser executada em diferentes sistemas operacionais, muitas vezes sem a necessidade de camadas intermediárias para interpretar essas aplicações, o que permite aprimorar a experiência dos usuários.

Apesar da possibilidade de executar as aplicações em diferentes plataformas, neste trabalho, através do desenvolvimento de uma aplicação educacional, foi possível observar que as aplicações necessitam, ao menos em parte, de especificações particulares. Esse é o caso da especificação da apresentação visual das aplicações que, no caso do desenvolvimento utilizando as bibliotecas React e React Native, precisam ser individualmente especificadas. Essas especificações particulares permitem adaptar as aplicações às características de cada plataforma. Assim, no caso dos aspectos de apresentação, pode ser interessante realizar especificações em cada plataforma a fim de explorar melhor as diferentes características de cada uma.

Aplicações em dispositivos móveis usualmente necessitam ser executadas em dispositivos com interfaces limitadas, uma vez que esses dispositivos não dispõem de mouses funcionais, apesar da possibilidade do toque na tela. Além disso, a funcionalidade do teclado é limitada, além do tamanho da tela ser pequeno quando comparado a tela de computadores tradicionais. Além disso, pela própria característica da mobilidade, o usuário, ao utilizar os dispositivos móveis, muitas vezes não dispõe de um ambiente adequado de acesso, podendo realizar esse acesso, por exemplo, enquanto realiza outras tarefas. Assim, as aplicações nesse ambiente não podem, na maioria das vezes, ser as mesmas aplicações do ambiente computacional tradicional. É necessário adaptar essas aplicações e, portanto, modificações nas apresentações dessas aplicações tornam-se, muitas vezes, obrigatórias.

Como trabalho futuro pretende-se aprimorar o desenvolvimento da aplicação apresentada neste trabalho, incluindo novas funcionalidades, uma vez que a biblioteca React apresentou características interessantes para o desenvolvimento dessa aplicação, incluindo a possibilidade do desenvolvimento em múltiplas plataformas com adaptações particulares na apresentação do conteúdo, necessárias para tornar as aplicações adequadas para cada uma das plataformas.

5 CONSIDERAÇÕES FINAIS

Neste trabalho foi avaliada a possibilidade de construir uma aplicação complexa com a possibilidade de execução em diferentes plataformas, incluindo o ambiente móvel. Foi avaliada a utilização de uma biblioteca que, apesar de precisar de modificações na especificação para apresentação das aplicações nessas diferentes plataformas, mostrou-se adequada a esse desenvolvimento. As modificações necessárias se mostraram pontuais e, de certa forma, desejáveis, uma vez que permitem que as características particulares dos ambientes e, em particular, do ambiente móvel, sejam adequadamente tratadas.

ABSCTRACT

This paper presents a proposal for developing an application that provides educational information across different platforms with a single code. The application can be run through the browser, the web, as well as mobile devices on Android and iOS systems. For development, React was used, a Javascript library that favors multiplatform development and, in the case of mobile devices, allows applications to run natively, thanks to another library, known as React Native. Thus, with only changes in visual representation that are dealt with in this paper, the same application can be executed in different environments.

Keywords: Mobile Applications. Multiplataform. React. React Native. Javascript.

REFERÊNCIAS

BANKS, A.; PORCELLO, E. **Learning React: Functional Web Development With React and Redux**. ISBN 978-1-491-95462-1. O'Reilly Media. 2017.

BARBOSA, G.; OLIVEIRA, E. **Usabilidade em aplicativos móveis educacionais: Um conjunto de heurísticas para avaliação**. XXVII Simpósio Brasileiro de Informática na Educação, 2017.

EISENMAN, B. **Learning React Native**. ISBN 978-1-491-98914-2. O'Reilly Media. 2018.

FLANAGAN, D. **JavaScript: The Definitive Guide**. ISBN 978-05-96805-52-4. O'Reilly Media. 2011.

LECHETA, R. R. **Android Essencial com Kotlin**. ISBN 978-85-75226-89-6. Novatec. 2018.

NUNES R. P., SANTOS, I. M. **A importância da avaliação ergonômico-pedagógica de aplicativos educacionais e os desafios encontrados no campo da aprendizagem móvel.** VII Congresso Brasileiro de Informática na Educação, 2018.

PONTES, G. **Progressive Web Apps: Construa Aplicações Progressivas com React**. ISBN 978-85-94188-56-4. Casa do Código. 2018.

VIEIRA, M.; SIMÕES, L; BARRETO, A. **Avaliação de Software Educativo: Aspectos Pedagógicos e Técnicos.** Revista Faculdade Cearence, v5, n1, 2012.

WEISER, M. **The Computer for the 21st Century**. Scientific American, pp. 94-104. Setembro, 1991.