Associação Propagadora Esdeva
Centro Universitário Academia – UniAcademia
Curso de Engenharia de Software
Projeto de Extensão

MANUAL DE REFERÊNCIA ELETRÔNICO PARA REACT NATIVE

Natan José Ribeiro Fernandes[1]
Andrei de Almeida Cardozo[1]
Romualdo Monteiro de Resende Costa[2]
Centro Universitário Academia, Juiz de Fora, MG

RESUMO

Este trabalho apresenta o desenvolvimento de um manual de referência eletrônico, sob a forma de um site da Web, com informações sobre a biblioteca *React Native*. Este manual, inicialmente, possui duas seções, uma para descrever os componentes dessa biblioteca e suas propriedades e outra para apresentar os elementos da API, incluindo seus métodos. Em cada uma dessas seções são apresentados exemplos que podem ser executados diretamente na plataforma. Este manual, proposto para facilitar o acesso dos programadores às especificações necessárias ao desenvolvimento de aplicações móveis, usa a mesma estrutura do site oficial da linguagem (*reactnative.dev*)

Palavras-chave: Aplicações móveis. Componentes. React Native.

1 INTRODUÇÃO

Os desenvolvedores, para construir as mais diversas aplicações computacionais, utilizam ferramentas específicas para este fim. Desde a invenção da máquina diferencial por Charles Babbage em 1822[3], é necessário fornecer aos dispositivos instruções de como executar uma tarefa específica. No começo, a execução dessas tarefas era modificada por engrenagens e, assim, a noção da programação era guiada por movimentos físicos. Mais tarde, com a invenção da eletricidade, esses dispositivos passaram a ser programados por circuitos. Em ambos os casos, as

[1] Discente do Curso de Engenharia de Software, do Centro Universitário Academia.
[2] Docente do Curso de Engenharia de Software, do Centro Universitário Academia. Orientador.
[3] https://pt.wikipedia.org/wiki/M%C3%A1quina_diferencial

modificações na programação exigiam uma grande reconstrução de parte dos equipamentos, o que tornava o processo bastante complexo e trabalhoso.

Somente a partir do desenvolvimento das linguagens de programação, sobretudo a partir da década de 1950, com a invenção da linguagem FORTRAN[4], o desenvolvimento das aplicações para computadores se tornou mais produtivo. A ideia era permitir aos desenvolvedores especificar o comportamento e execução das tarefas através de instruções que, posteriormente, eram compiladas para programas executados diretamente pelo computador (SILBERSCHATZ, 1998).

As instruções que formavam inicialmente as linguagens de programação eram baseadas em variáveis, com tipos que permitiam especificar operações matemáticas, sejam elas no domínio dos valores inteiros ou reais. Também existiam operações lógicas, baseadas na lógica booleana, com tipos verdadeiros e falsos. Estruturas complementares permitiam testes de valores e ciclos, oferecendo repetições de ações que podiam ser especificadas de forma relativamente simples.

Ao longo dos anos, as linguagens de programação, bem como sua aplicação, evoluíram, juntamente com as novas tecnologias. Recentemente, com a popularização da computação móvel, caracterizada pelo uso de dispositivos móveis, principalmente celulares, onde os usuários, através das aplicações, podem se comunicar, sobretudo, através da Internet (WEISER, 1991), o modelo de desenvolvimento baseado em componentes, tem-se tornado bastante popular. Nesse modelo, as especificações do programador, através de uma linguagem, são encapsuladas em unidades que podem ser utilizadas individualmente, mas, sobretudo, podem ser combinadas na construção das aplicações. Esse modelo tende a permitir a divisão da complexidade e favorecer o reuso, já que cada componente pode ter sido desenvolvido por programadores diferentes.

Para alcançar os objetivos esperados do desenvolvimento baseado em componentes, é necessário que a especificação desses componentes seja realizada através de uma linguagem adequada e, principalmente, é necessário que exista uma ampla documentação a respeito dos componentes existentes, a fim de permitir que os programadores conheçam suas funcionalidades e possam utilizá-las adequadamente. Nesse contexto, este trabalho tem por objetivo oferecer à comunidade de desenvolvedores de aplicações móveis um manual de referência

[4] https://en.wikipedia.org/wiki/Fortran

para a biblioteca *React Native* (EISENMAN, 2018), que oferece um conjunto de componentes *Javascript* (FLANAGAN, 2011). A próxima seção apresenta algumas características dessa biblioteca, importantes para o desenvolvimento deste trabalho e que, assim, justificam sua escolha.

A seguir, a terceira seção apresenta o manual de referência eletrônico desenvolvido com destaque para a sua seção de componentes. A quarta seção apresenta a parte do manual de referência desenvolvido para documentar a API da biblioteca, formada por elementos que complementam a utilização dos seus componentes. Finalmente, os resultados alcançados com discussões correlatas são encontrados nas últimas seções, onde são apresentadas as conclusões, bem como os possíveis trabalhos futuros.

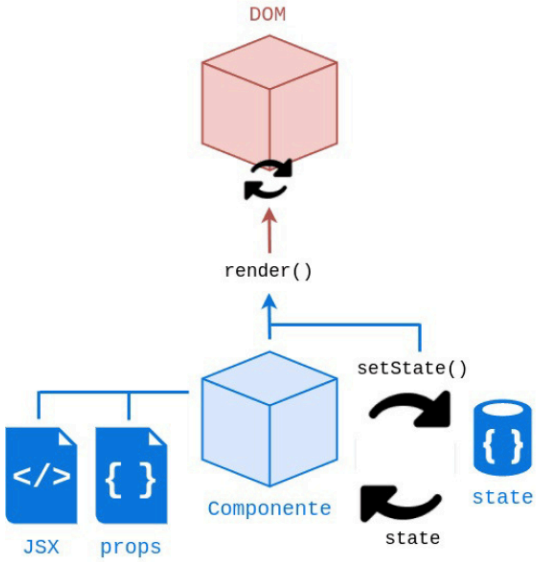
2 REFERENCIAL TEÓRICO

2.1 React Native

React Native é uma biblioteca para construção de aplicativos móveis baseada no *framework* definido originalmente pela biblioteca *React* (BANKS, 2017). Assim, as especificações básicas 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*[5]. De forma similar, o ciclo de vida das aplicações é o mesmo, isto é, o *framework* para a construção das aplicações permanece inalterado. A Figura 1 apresenta a representação básica desse framework, com destaque para seus elementos. Cabe destacar que a atualização dos componentes, no caso do *React Native*, é realizada diretamente na plataforma de execução, como será visto adiante.

[5] https://facebook.github.io/jsx

FIGURA 1: Elementos de um Componente *React Native* definidos pelo Framework



Fonte: PONTES, 2018.

Na Figura 1, 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 1 (*state*), 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 Native* é completado pelos métodos apresentados na Tabela 1.

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

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.

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 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 Native*

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*

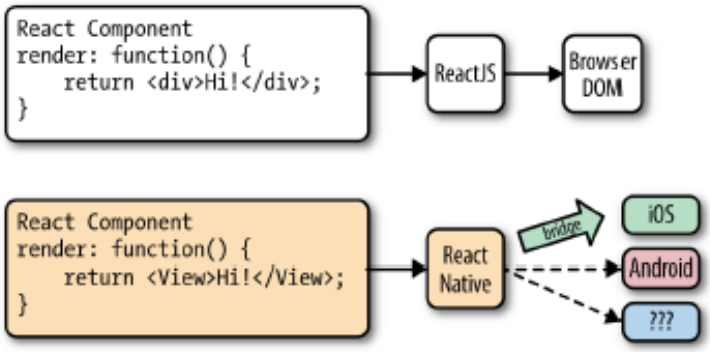
Fonte: do autor.

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. Particularmente, no caso do *React Native*, a apresentação visual é realizada por componentes próprios dessa biblioteca, como representado na Figura 2. 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 2. Comportamento da Apresentação da Aplicação React Native em relação ao React.



Fonte: EISENMAN, 2018.

Como apresentado na Figura 2, as modificações nos componentes da aplicação *React Native*, através de uma ponte (*bridge*, na Figura 2), 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* (LECHETA, 2015), por exemplo, esse elemento será traduzido para o elemento homônimo <*View*>. Já no caso do sistema *iOS* (iOS, 2021), 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).

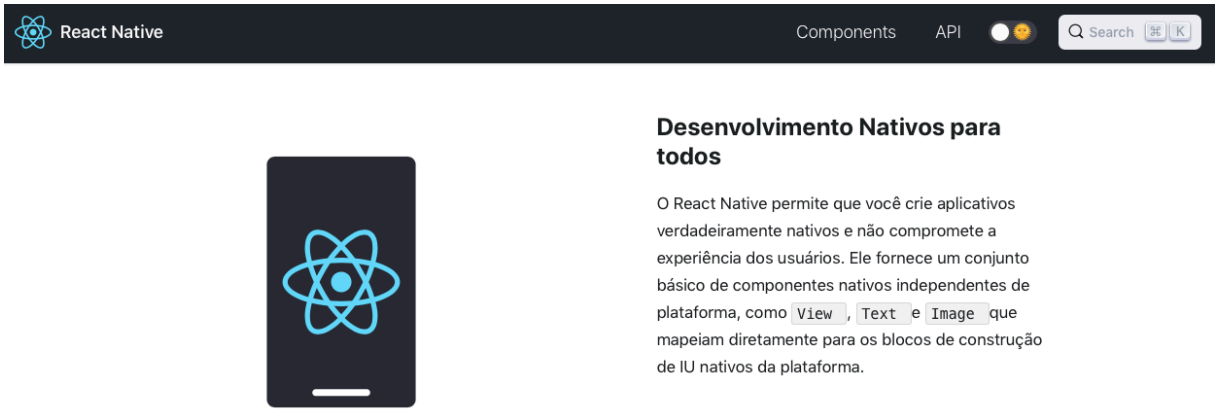
Cada um dos elementos apresentados, bem como os demais elementos existentes na biblioteca *React Native* são implementados como componentes. Essa biblioteca, portanto, é baseada em componentes visuais, alguns dos quais serão vistos nas próximas seções.

3 COMPONENTES REACT NATIVE

A Figura 3 apresenta a tela inicial do site que contém o manual de referência eletrônico para o *React Native*. A partir desta tela, o desenvolvedor pode escolher navegar entre os componentes da biblioteca, que serão descritos nesta seção e os elementos da API, que serão tratados na próxima seção. Também existe a opção de

acessar um item diretamente, através da realização de uma pesquisa utilizando a ferramenta de busca.

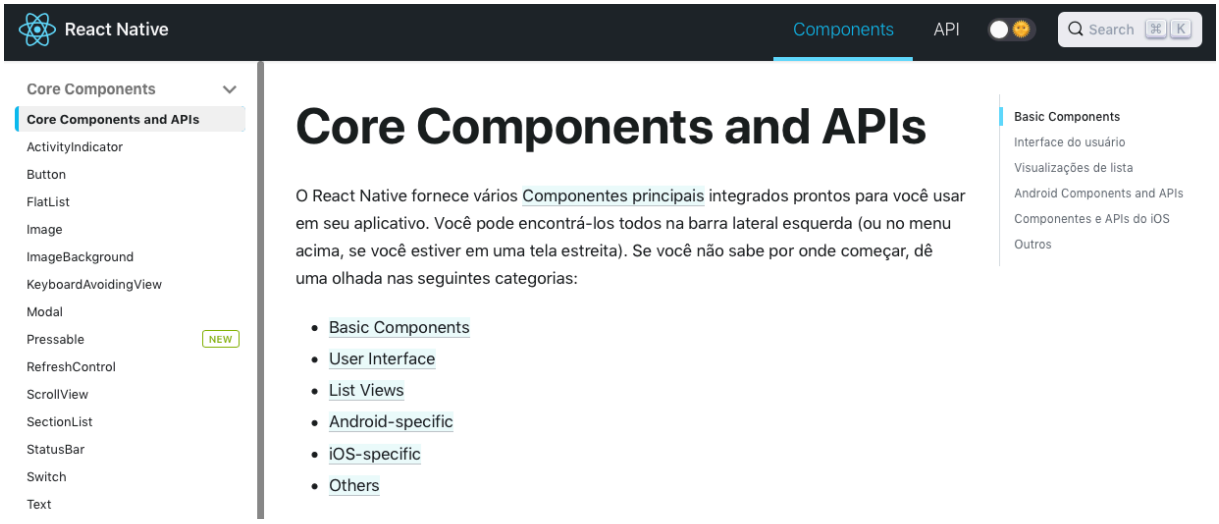
FIGURA 3. Tela inicial do site do manual de referência eletrônico.



Fonte: do autor.

Atualmente, na versão 0.63 da biblioteca *React Native*, na qual este trabalho foi baseado, são listados 24 (vinte e quatro) componentes principais (*Core Components*) agrupados em seis categorias, conforme apresentado na Figura 4. Essas categorias incluem os Componentes Básicos (*Basic Components*); Interface do Usuário (*User Interface*); Listagens (*List Views*); Específicos para *Android* e *iOS* (*Android-specific* e *IOS-specific*) e, finalmente a categoria Outros (*Others*). No menu a esquerda, apresentado na Figura 4, o desenvolvedor pode escolher entre todos os componentes disponíveis na versão 0.63 da biblioteca *React Native*.

FIGURA 4. Tela inicial dos componentes do manual de referência eletrônico.



Fonte: do autor.

Entre os componentes existentes, a Figura 5 apresenta em destaque, como exemplo, o componente *Button* (botão). Na parte central da figura são apresentadas informações relevantes para o desenvolvedor, incluindo um exemplo de como utilizar este componente. Também podem ser encontradas informações a respeito de outros componentes que podem ser utilizados com o mesmo objetivo, como o *TouchableOpacity* e o *TouchableWithoutFeedback.* Além disso, a página contém também *links* de acesso para outros componentes do tipo botão desenvolvidos pela comunidade. É importante mencionar que, além dos componentes documentados neste trabalho, inúmeros outros podem existir, uma vez que cada desenvolvedor pode construir seus próprios componentes, por exemplo, agrupando esses componentes básicos, mudando sua aparência ou mesmo o seu comportamento.

Sempre na parte à direita da descrição do componente são apresentados, em destaque, suas propriedades. A Figura 5 apresenta, como exemplo, as propriedades do componente *Button.* As propriedades que somente são válidas na plataforma *Android* recebem uma marcação verde. As propriedades que somente são válidas na plataforma iOS recebem uma marcação preta. Finalmente, a marcação azul é reservada para as propriedades que são válidas para aplicações executadas na TV, ao invés da tradicional execução sobre celular e *tablets*.

FIGURA 5. Tela inicial com as informações do componente Button.

Fonte: do autor.

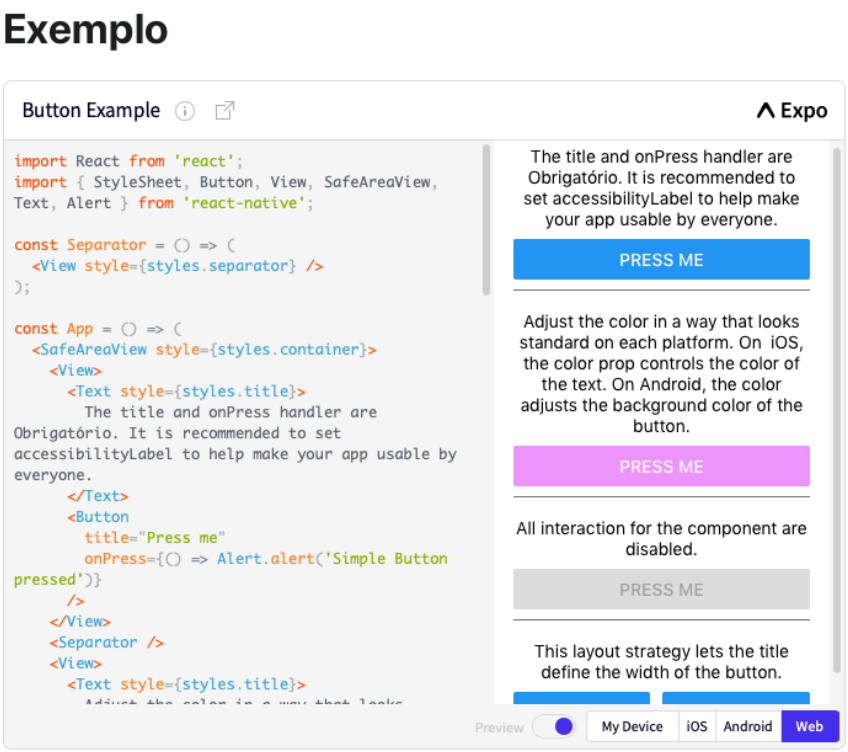
Considerando apenas os componentes principais do *React Native*, existem aproximadamente 360 (trezentos e sessenta) propriedades distribuídas de maneira

heterogênea sobre esses componentes. Em algumas dessas propriedades devem ser utilizados valores livres, em outras, somente são aceitos conjuntos de valores predeterminados. Algumas dessas propriedades são obrigatórias, outras são opcionais. Algumas, preferencialmente, não devem mais ser utilizadas, por serem obsoletas e só estão disponíveis por questões de compatibilidade. Enquanto algumas propriedades precisam receber um valor, incluindo, eventualmente, alguma função a ser chamada, outras precisam ser apenas mencionadas. Quando a propriedade é obrigatória, ele é mostrada em negrito, como é o caso das propriedades *onPress* e *title* do componente *ButtonI,* ambas apresentadas na Figura 5. A primeira dessas propriedades deve receber um evento a ser chamado quando o botão for pressionado e, a segunda, deve receber uma *string* que contém o texto a ser apresentado no botão.

Além da documentação, o manual de referência eletrônico permite que todos os componentes tenham seus exemplos testados diretamente no site. A Figura 6 apresenta esta situação para o componente *Button*. Na parte esquerda dessa figura é apresentado um código, como exemplo, que é executado, diretamente no site, cujo resultado é apresentado na parte direita dessa figura. Para realizar esta apresentação é utilizada uma ferramenta conhecida como *Snack* (Expo)[6]. Além de realizar a apresentação, essa ferramenta permite que o desenvolvedor escolha o ambiente para apresentação que pode ser o navegador, como no exemplo apresentado na Figura 6, uma máquina virtual *Android* ou *iOS* ou, até mesmo, o próprio dispositivo móvel do desenvolvedor. Nesse caso, o desenvolvedor precisa ter a ferramenta *Expo* instalada no celular e, ao escolher esta opção no menu inferior, apresentado na Figura 6, um QRCode (LIU, 2008) será apresentado, como chave para execução do exemplo no próprio celular do usuário.

[6] https://snack.expo.dev

FIGURA 6. Exemplo Interativo do componente *Button*.



Fonte: do autor.

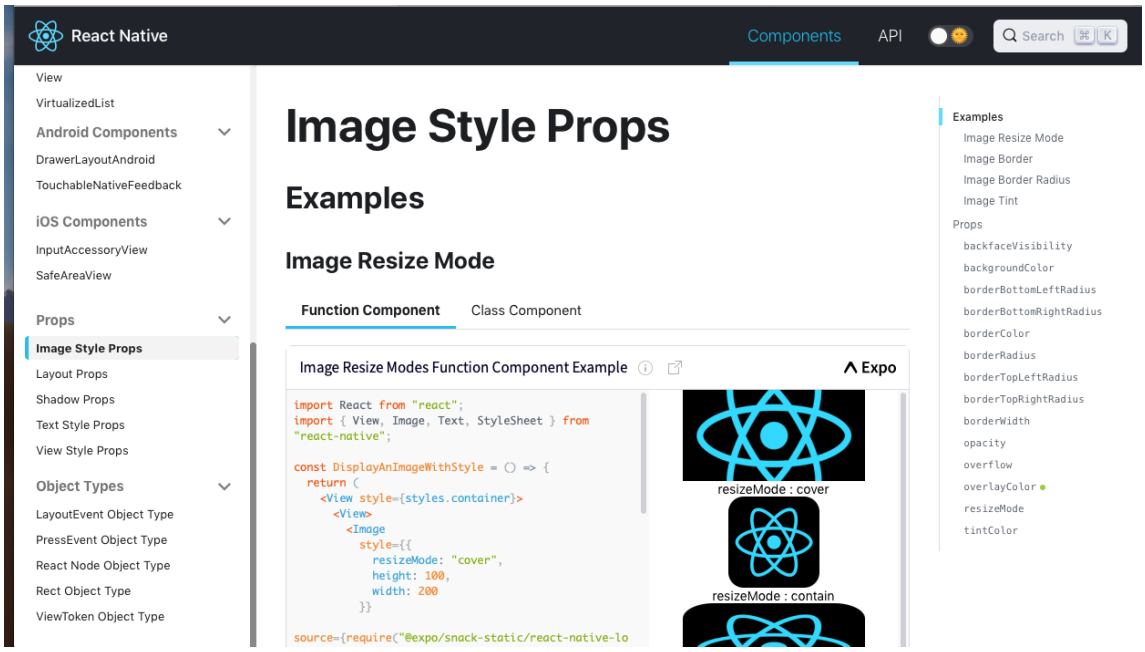
Para visualizar a edição on-line, a Figura 7 apresenta o exemplo, originalmente apresentado na Figura 6, associado à descrição do componente *Button*, modificado. Na modificação apresentada na Figura 7, o exemplo foi simplificado. Foram retirados os demais componentes e deixado apenas um componente *Button* dentro de um componente *SafeAreaView*, cujo funcionamento tem como objetivo definir a área de apresentação dos componentes (esse componente também é apresentado no manual de referência). No exemplo da Figura 7, o texto do botão foi alterado através da edição do conteúdo da sua propriedade *title*. O resultado dessas operações é apresentado em tempo real na tela do usuário, conforme pode ser visualizado na Figura 7.

FIGURA 7. Exemplo Interativo, modificado, do componente *Button*.

Fonte: do autor.

Além dos componentes propriamente ditos, adicionalmente, o manual de referência, apresenta também, nesta mesma aba (*Components*), algumas das principais propriedades utilizadas, sobretudo, na formatação de aspectos visuais, isto é, na apresentação dos componentes. A Figura 8 apresenta, em destaque, as propriedades de estilo utilizadas nas imagens (componente *image*). Embora essas propriedades já estejam descritas no componente *image*, através do menu, o desenvolvedor pode acessar, diretamente, as propriedades de estilo das imagens (*Image Style Props*), as propriedades do leiaute (*Layout Props*), as propriedades relacionadas à animação (*Shadow Props*), as propriedades de formatação dos textos (*Text Style Props*) e as propriedades de formatação das áreas visuais (*View Style Props*).

FIGURA 8. Tela com as propriedades de estilo das imagens.



Fonte: do autor.

Conforme apresentado na Figura 8, uma vez acessado cada conjunto de propriedades, entre as mencionadas, o desenvolvedor tem acesso à descrição deste conjunto. Na Figura 8 é apresentado, como exemplo, as propriedades de estilo das imagens. Além da descrição, são apresentados exemplos interativos, nos mesmos moldes daqueles apresentados para cada componente.

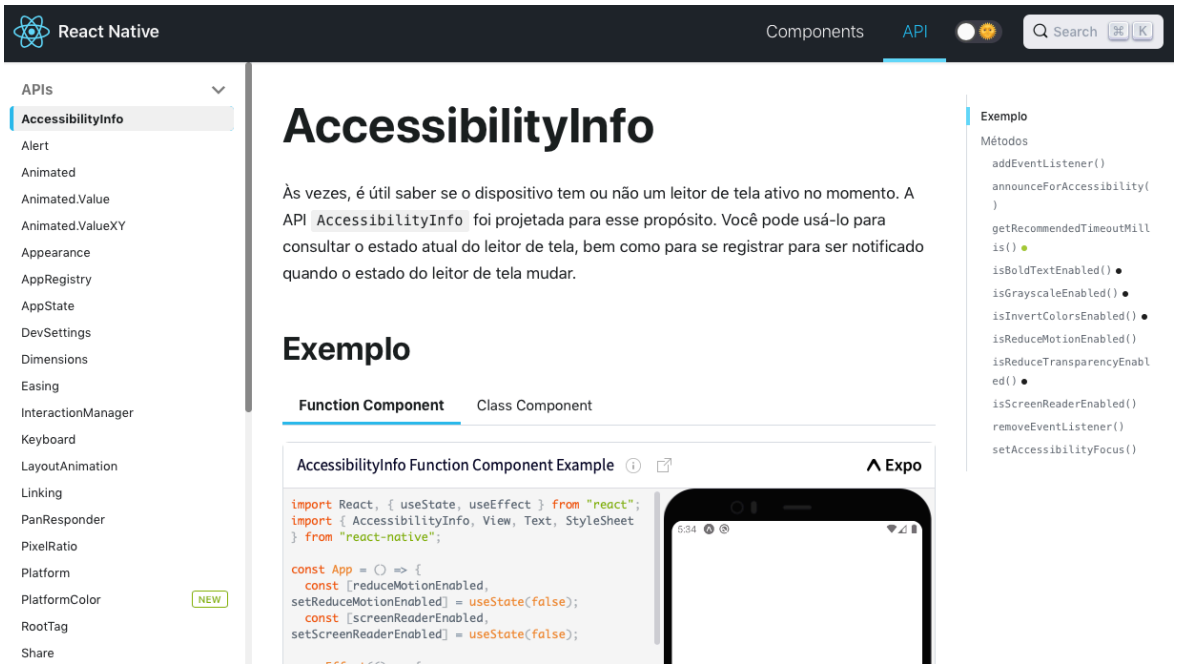
Por fim, ainda na aba *Components* do manual de referência, são descritos alguns tipos de objetos que são utilizados como parâmetros das propriedades de alguns componentes. Os tipos disponíveis são apresentados na Figura 8 (*LayoutEvent Object*, *PressEvent Object*, *React Node Object*, *React Object, ViewToken*) e, para cada um desses tipos, existe uma página com descrição, propriedades e exemplo interativo, nos mesmos moldes daquelas dos componentes, anteriormente apresentadas.

5 API REACT NATIVE

A outra aba atual do manual de referência eletrônico é a referente à API, cujo conteúdo inicial é apresentado na Figura 9. A partir da tela apresentada nessa figura, o desenvolvedor pode escolher navegar entre os elementos da API do *React Native* que, atualmente, na versão 0.63 dessa biblioteca, são formados por um conjunto de 33 (trinta e três) elementos, sendo 3 (três) exclusivos para a plataforma

Android e outros 3 (três) exclusivos para a plataforma *iOS*. Qualquer um desses elementos da API pode ser selecionado através do menu localizado à esquerda do manual de referência. Na Figura 9 o elemento selecionado é o *AccessibilityInfo*, que é mostrado automaticamente como primeiro elemento assim que a opção API é selecionada.

FIGURA 9.Tela inicial das APIs do manual de referência eletrônico.



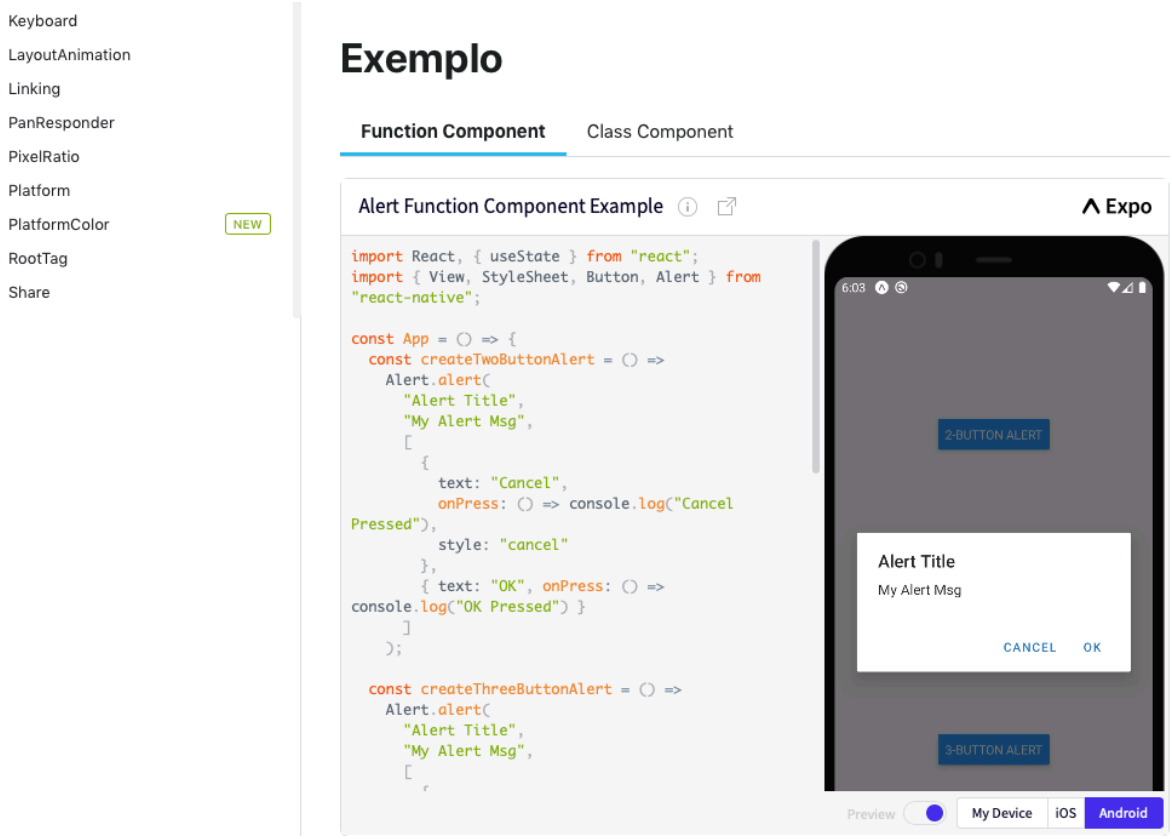
Fonte: do autor.

Como pode ser observado na Figura 9, as especificações do manual de referência eletrônico para os elementos da API do *React Native* seguem o mesmo formato daquela utilizada para os componentes. Para cada elemento selecionado é mostrada uma descrição da sua funcionalidade ao centro da página, com um ou mais exemplos que podem ser modificados de maneira interativa. Na parte direita da página são apresentadas as propriedades, bem como os métodos do elemento selecionado.

Diferente dos componentes, os elementos da API não necessitam ser renderizados, isto é, não precisam estar no método render da classe ou no retorno da função, como apresentado na Seção 2. Esses elementos, uma vez importados, podem ser diretamente utilizados. Para essa utilização, normalmente, os métodos desses elementos são utilizados. Como exemplo, a Figura 10 apresenta a tela de

execução com o elemento *Alert*. Para acessar este exemplo, o desenvolvedor deve escolher o elemento *Alert* no menu lateral esquerdo.

FIGURA 10. Tela inicial com um exemplo do elemento da API Alert.



Fonte: do autor.

O elemento *Alert* permite especificar nas aplicações uma janela do tipo modal (sobreposta aos elementos da interface). Através dessa janela é possível passar uma informação ao usuário da aplicação e, ainda, oferecer escolhas, através da seção de um botão com ações respectivas. Como apresentado no código da Figura 10, a apresentação do elemento é realizada através do seu método *alert*. Esse método recebe como parâmetros, o título da mensagem, o seu conteúdo e objetos que definem os botões, com textos e ações.

Dessa forma, os elementos da API são tipos de componentes que não necessitam ser instanciados e que, usualmente, tem suas ações acessadas através de métodos específicos. De forma similar aos componentes apresentados na seção anterior, cada um dos elementos da API, com detalhes de seus métodos e exemplos interativos são apresentados ao longo da aba API do manual de referência eletrônico.

6 RESULTADOS E DISCUSSÃO

Atualmente, a biblioteca *React Native* já oferece aos desenvolvedores um site com a documentação dos seus componentes e API. No entanto, essa documentação, na versão atual (0.63), está disponível apenas no idioma inglês, o que pode levar a falhas na interpretação e aumentar a dificuldade de uso dos desenvolvedores menos familiarizados com o idioma. Além disso, a construção de um manual de referência próprio, como este apresentado neste trabalho, permite adaptar o conteúdo, permitindo que ele seja simplificado ou, eventualmente, até mesmo estendido.

O manual desenvolvido neste trabalho foi utilizado como ferramenta de aprendizado na disciplina de Desenvolvimento de Software para Dispositivos Móveis dos cursos de Engenharia de Software e Sistemas de Informação. Nessa disciplina os alunos realizam o desenvolvimento de uma aplicação e, para isto, usualmente utilizam a biblioteca *React Native*.

Alguns exemplos existentes no manual foram colocados de maneira mais simples e direta, quando comparados com aqueles existentes na documentação do *React Native*. Foi identificado que os exemplos da documentação, muitas vezes, utilizam mais de um componente, cuja aplicação pode não ser fundamental ao exemplo, e que acabam por dispersar o foco na explicação do componente principal.

De forma similar, a construção do manual permitiu acrescentar novos exemplos àqueles existentes na documentação original. É possível, inclusive, adaptar o conteúdo acordo com a demanda da disciplina de Desenvolvimento de Software para Dispositivos Móveis, bem como de acordo com a necessidade dos alunos, acrescentando ou modificando os exemplos disponíveis.

Além do público em geral, os participantes do projeto tiveram a oportunidade de realizar um estudo aprofundado sobre a biblioteca, conhecendo em detalhes os componentes e sua API.

7 CONSIDERAÇÕES FINAIS

Neste trabalho foi desenvolvido um manual de referência eletrônico para a biblioteca *React Native*. Muito mais do que a simples tradução do conteúdo, foi possível realizar adaptações para adequar as informação aos desenvolvedores, sobretudo aos alunos.

A adoção de um conteúdo próprio para publicação das informações, como é o caso do manual de referência desenvolvido permite, no futuro, inserir novos conteúdos, criando, por exemplo, objetos de aprendizagem, voltados ao aprendizado da biblioteca *React Native* e, consequentemente, ao desenvolvimento de aplicações móveis.

Como trabalho futuro, além da construção dos objetos de aprendizagem mencionados, novas seções deverão ser adicionadas ao manual, incluindo um repositório de trabalhos já realizados pelos alunos, em um formato que o desenvolvedor consiga visualizar as aplicações, ainda que parcialmente, em funcionamento no próprio ambiente.

ABSCTRACT

This work presents the development of an electronic reference manual, in the form of a web site, with information about a React Native library. This manual, bulletin, has two sobs, one to define the components of this library and their properties and one to introduce its API elements, including its methods. In each of these sources are examples that can be sought on the platform. This manual, proposed to facilitate programmers' access to specifications for the development of mobile applications, uses the same structure as the language's official website (reactnative.dev)

Keywords: Mobile Applications. Components. React Native.

REFERÊNCIAS

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

iOS. **Build Your Apps for iOS 12.** Disponível em <https://developer.apple.com/ios>. Acesso em: 07 de Dezembro de 2021.

LECHETA, R. R. **Google Android: Aprenda a criar aplicações para dispositivos móveis com o Android SDK**. ISBN 978-8575224687. Novatec. 2015.

LIU, Y.; YANG, J.; LIU, M. **Recognition of QR Code with mobile phones**. Chinese Control and Decision Conference. Agosto, 2008.

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

SILBERSCHATZ, A; GALVIN, P. B. **Operating System Concepts**. ISBN0-201-54262-5. Addison-Wesley. 1998.

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