BIBLIOTECA REACT NATIVE SLIDING MODAL

Everthon Z. Carniel, João Marcos G. D. R. Costa, Me. Fabrício Gustavo Henrique

Faculdade de Tecnologia de Ribeirão Preto (FATEC)

Ribeirão Preto, SP – Brasil

everthon.carniel@fatec.sp.gov.br,
joao.costa31@fatec.sp.gov.br,
fabricio.henrique@fatec.sp.gov.br

Resumo. *Este artigo abordará o desenvolvimento de uma biblioteca em React Native, para implementar um modal responsivo à gestos panorâmicos, que utilizará recursos para a criação de animações fluidas e bibliotecas adicionais para manipular gestos e propriedades.*

Abstract. *This paper will approach the development of a React Native library, to implement a modal responsive to panoramic gestures, which will use resources to create fluid animations and additional libraries to manipulate gestures and properties.*

1. INTRODUÇÃO

Pesquisando pela comunidade do React Native, foi identificada a necessidade de criar um *modal* dinâmico capaz de ler e interpretar gestos panorâmicos dos usuários e de produzir animações gráficas, que seja compatível com a biblioteca de navegação de telas, chamada React Navigation.

Portanto, foi proposta a criação de uma biblioteca visando o desenvolvimento e otimização dessa ferramenta para contribuir com a comunidade React Native e facilitar a integração desse recurso com diversos projetos. Esta biblioteca será chamada *react-native-sliding-modal* e durante este artigo será mencionada como *sliding-modal* ou *modal*.

Esta biblioteca será publicada de maneira *open source* na comunidade do React Native e também possuirá compatibilidade com a biblioteca React Native Gesture Handler, utilizada para a manipulação de gestos, e com a Prop Types, criada pelo Facebook e utilizada para a checagem e tipagem de propriedades.

Este projeto descreve a abstração e o desenvolvimento desta ferramenta gráfica, assim como sua funcionalidade, integração e aplicação prática em desenvolvimentos de projetos na comunidade React Native.

2. JUSTIFICATIVA

O objetivo deste trabalho é colaborar com a comunidade do React Native, desenvolvendo um novo componente de interface para ser usado de forma gratuita.

3. PROBLEMA DE PESQUISA

Como desenvolver um *modal* responsivo a gestos panorâmicos e reutilizável em projetos React Native, e publicá-lo na comunidade?

4. OBJETIVO GERAL

O objetivo geral desse projeto é desenvolver uma biblioteca *open source*, para contribuir com o desenvolvimento de aplicações móveis em React Native suprindo a necessidade de um *modal* responsivo à gestos panorâmicos.

4.1. Objetivos específicos

- Pesquisar e entender quais as tecnologias necessárias para a implementação da biblioteca;
- Escrever e documentar a estrutura do projeto;
- Desenvolver e testar o código da biblioteca.
- Publicar a biblioteca de maneira *open source*.

5. DESENVOLVIMENTO - MÉTODOS E MATERIAIS

A seguir serão abordadas as tecnologias utilizadas no desenvolvimento do *sliding-modal*, como os métodos e materiais disponibilizados pelo *framework* escolhido, por sua comunidade e por terceiros.

5.1. Manipulação de Gestos

Todo conceito de interação humano computador (IHC) em dispositivos móveis é realizado com base na manipulação de gestos dos usuários. Portanto, deve ser feito de forma prática, pois, apenas assim será possível proporcionar uma boa experiência aos usuários.

Rodrigo Roncaglio (2016) esclarece que gestos são a base de como você interage com seus dispositivos móveis, e que estes podem ser divididos em duas etapas, o toque mecânico e as atividades resultado do toque. Onde toques mecânicos são interações que usuários fazem com a tela e as atividades dessas interações são os resultados dos toques mecânicos precedentes.

Neste projeto a biblioteca utilizada para a manipulação de gestos é o React Native Gesture Handler. As animações ou atividades resultantes a essas interações serão

fornecidas pela Animated API, uma biblioteca virtual pertencente ao React Native.

Inicialmente o *sliding-modal* vai suportar dois tipos de gestos panorâmicos, o *swiping* e *flinging*, gestos que são contínuos e executados por usuários, que consistem em deslizar ou arrastar componentes gráficos.

O *swiping* consiste em pressionar um elemento na tela movendo-o de um lado para outro, até que ele pare de ser pressionado. Esse evento será manipulado quando o usuário segurar e mover a área arrastável do *sliding-modal* para expandir ou diminuir seu tamanho, bem como para fechá-lo.

O *flinging* funciona assim como o *swiping*, mas necessita de uma maior velocidade para ser definido. Ou seja, sua atividade resulta em uma animação que arremessa o componente selecionado. Esse gesto será utilizado para fechar o *sliding-modal*.

A imagem abaixo exemplifica como esses gestos serão utilizados na manipulação desta ferramenta.

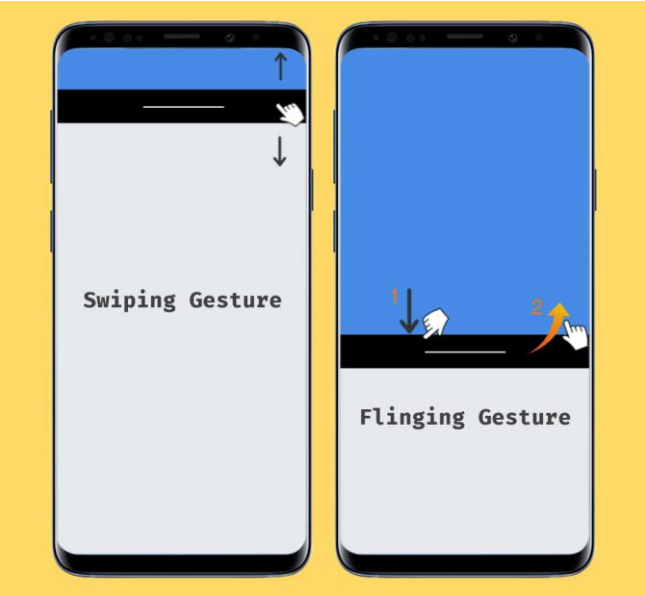

Figura 1. Gestos de manipulação do sliding-modal
Fonte: (João Marcos G. D. R. Costa, 2020)

5.2. React Native

Um *framework* é o resultado da unificação de códigos comuns para atingir uma funcionalidade específica para abstrair um resultado. O *framework* escolhido para o desenvolvimento do *sliding-modal* é o React Native.

O React Native é utilizado para desenvolver aplicações móveis e é estruturado em React JS, um *framework* para desenvolvimento *web*. Ambos são baseados em linguagem JavaScript, pois é utilizada principalmente em desenvolvimento de interfaces gráficas, é orientada a eventos e é compatível com a linguagem de marcação HTML e folhas de estilo CSS.

"JavaScript é uma linguagem de alto nível, dinâmica, interpretada e não tipada, conveniente para estilos de programação orientados a objetos e funcionais." (FLANAGAN, 2013, cap. 1, p. 1)

Dentre as principais características deste *framework*, a mais importante é a orientação a objetos declarativos, que possibilita a manipulação do *ciclo de vida* de componentes conforme os dados são atualizados. Outra característica muito importante é a criação de componentes que gerenciam seu próprio *ciclo de vida*, esse método é chamado de *componentização*.

Douglas Novato (2020) explica que o conceito principal do React Native é a construção de interfaces a partir de componentes para renderizar aplicações para Android e iOS (Sistema Operacional da Apple) de forma nativa.

Portanto, é possível afirmar que o React Native possui uma arquitetura *component-based*. Esse tipo de arquitetura consiste na decomposição do design em componentes funcionais ou lógicos que representam interfaces de comunicação bem definidas contendo métodos, eventos e propriedades.

Um componente React Native é controlado por dois tipos de dados: *props* e *states*. As *props* são propriedades ou valores de leitura que os componentes utilizam para estabelecer uma comunicação entre si, enquanto os *states* são valores de leitura e escrita gerenciados e inicializados internamente por componentes.

A seguir serão abordados os materiais e métodos essenciais disponibilizados pelo React Native para o desenvolvimento do *sliding-modal*.

5.2.1 Modal

O Modal é um componente do React Native utilizado para exibir conteúdos acima de uma visualização (tela) principal na aplicação.

Este componente é definido como a estrutura do *sliding-modal*, e consequentemente será possível exibi-lo em uma camada superior da aplicação e utilizá-lo em qualquer tela, garantindo que outros componentes não sobreponham o seu conteúdo.

É por conta do Modal que o *sliding-modal* possui compatibilidade com os componentes de navegação do React Navigation, que geralmente são carregados em uma camada elevada da aplicação localizada abaixo da camada deste componente.

5.2.2 Animated API

A Animated API também é uma biblioteca do React Native, ela é projetada para criar animações fluidas de fácil construção e manutenção. Ela consiste em relações declarativas entre *inputs* e *outputs*, transformações configuráveis entre eles e métodos para controlar a execução de animações baseadas em tempo.

Para criar uma animação com o Animated é necessário utilizar um *valor animado* ou o atributo *Animated.Value*, que será vinculado a propriedades de estilo de componentes animados e será atualizado por meio de funções de animação providas pela Animated. Quando alterado, esse atributo causa atualizações gráficas em componentes

dependentes, alterando suas propriedades de estilo.

Atualmente a Animated fornece as funções de animação *decay*, *spring* e *timing*. Para o desenvolvimento deste projeto foram necessárias as funções *spring* e *timing*.

Cada função de animação é capaz de gerar uma curva de animação específica, que controla como seus valores devem ser animados. Ou seja, elas controlam a maneira como um intervalo de valores é atribuído a um *valor animado*, determinando o comportamento do efeito gráfico.

A função *spring* fornece um modelo básico de *física de mola* para animações, que será utilizado sempre que o *sliding-modal* for exibido na aplicação.

A função *timing*, "anima" um valor durante o tempo definido, usando funções de atenuação para transmitir diferentes movimentos físicos nesse intervalo de tempo, gerando animações de efeito saltitante, animações inerciais ou animações com efeito de elasticidade. Ela será utilizada ao final da execução do gesto *flinging*, arremessando o *modal* para baixo até que ele feche se o gesto não for finalizado.

O componente que irá monitorar os gestos dos usuários e será vinculado a um *valor animado* para executar as animações descritas anteriormente é o *Animated.View*, um componente com suporte a layout com *flexbox*, estilização, tratamentos de toque e controles de acessibilidade. Ele será a área arrastável utilizada para fechar o *sliding-modal* através dos gestos já definidos.

Para que o desenvolvedor possa exibir conteúdos grandes dentro do *sliding-modal* e permitir que usuários arrastem o conteúdo na direção vertical, sem interferir nessas animações, serão necessários os *componentes de rolagem*.

5.2.3 Componentes de rolagem

Inicialmente o *sliding-modal* poderá utilizar dois componentes de rolagem do React Native, o *ScrollView* e o *FlatList*. Ambos englobarão o conteúdo recebido.

O *ScrollView* é um componente de rolagem genérico capaz de hospedar vários componentes e conteúdos, possuindo uma rolagem vertical ou horizontal.

O *FlatList* é um componente de rolagem performático para exibir listas simples, também possuindo rolagem vertical e horizontal. Ele possui alguns recursos adicionais como suporte a cabeçalho e rodapé, suporte de separadores, carregamento de rolagem, entre outros.

O *sliding-modal* poderá receber uma propriedade para definir qual será o componente de rolagem, caso essa propriedade não seja recebida, o *ScrollView* será definido como componente de rolagem padrão.

Portanto, para possibilitar que propriedades sejam definidas e garantir que o desenvolvedor conheça quais são elas, é necessária a utilização de validadores de *props*. Neste caso, a biblioteca utilizada para esse tipo de validação é a *PropTypes*, que será abordada na seção a seguir.

5.3. PropTypes

Esta é uma biblioteca adicional do React utilizada para validação, checagem e tipagem de propriedades em modo de desenvolvimento. Ou seja, é ela quem valida as *props* que são recebidas por um componente para auxiliar o desenvolvimento.

Neste projeto, o *sliding-modal* irá possuir propriedades mutáveis, que serão gerenciadas por objetos validadores exportados da PropTypes para indicar quais propriedades são requeridas, os seus tipos, nomes e valores padrão.

5.4. React Native Gesture Handler

Este material é uma biblioteca popular utilizada para a manipulação de gestos, que tem como objetivo substituir o sistema de toque integrado do React Native, chamado Gesture Responder System.

Para detectar os gestos panorâmicos dos usuários na área arrastável do *modal*, será utilizado um manipulador chamado *PanGestureHandler* fornecido por essa biblioteca. Ele gerencia os gestos contínuos que permitem rastrear os movimentos na tela. Ele é ativado quando o dedo é colocado na tela e movido por alguma distância, fornecendo dados traduzidos desde o início do gesto, como sua velocidade e amplitude.

A Gesture Handler está sendo utilizada neste projeto, porque, diferentemente do Gesture Responder System, ela é capaz de detectar os gestos que acontecem na mesma camada do componente Modal citado anteriormente, que será utilizado neste trabalho por questões de compatibilidade com a biblioteca React Navigation, que será abordada na próxima seção.

5.5. React Navigation

A React Navigation é a principal biblioteca de navegação para React. Ela disponibiliza um *navegador de pilha* ou *stack navigator*, que fornece recursos para que aplicativos possam gerenciar seu histórico de navegação, roteamento e efetuar transições entre telas. Entre os principais recursos dessa biblioteca estão os componentes de navegação, como o cabeçalho, o navegador de abas e o navegador de gaveta.

Essa biblioteca é frequentemente utilizada no desenvolvimento de aplicações por conta de sua praticidade, pois, a criação de um sistema próprio de navegação exige um algoritmo robusto. É justamente por esse motivo que o *sliding-modal* deve possuir compatibilidade com seus recursos, oferecendo uma maneira prática de implementação.

5.6. Desenvolvimento Prático

Para o desenvolvimento do algoritmo, foi escolhido o editor de *código-fonte* Visual Studio Code. Além de ser leve, ele possui suporte integrado para JavaScript, TypeScript, Node e *plugins* auxiliares para React Native. Portanto, nesta ocasião ele é o editor de texto mais recomendável.

Para compilar e testar o código, foram utilizados emuladores do Android Studio, ambiente de desenvolvimento integrado do Android, e do XCode, ambiente de desenvolvimento integrado da Apple.

Todo o controle de versão desta biblioteca foi feito via repositório no GitHub e no Node Package Manager (NPM).

O GitHub é um sistema de gerenciamento de projetos e versões em *nuvem*, essencial para o desenvolvimento de bibliotecas.

Semelhantemente, o NPM é um gerenciador utilizado para publicar projetos *Node* e um utilitário de linha de comando para interagir com repositórios e suas versões, auxiliando na instalação de pacotes e dependências. É por meio dele que o *sliding-modal* poderá ser instalado e utilizado.

A próxima seção documenta a instalação e utilização do *sliding-modal*, e também suas propriedades.

6. Documentação da biblioteca

O *sliding-modal* possui dois componentes básicos de controle de interface, a *SwipeableArea*, ou área arrastável, que será utilizada para fechá-lo e gerenciar os gestos de *swiping* e *flinging* do usuário por meio da React Native Gesture Handler, e o componente de rolagem, que poderá ser tanto o *ScrollView* quanto o *FlatList* citados anteriormente.

6.1 Instalação

O *sliding-modal* possui duas dependências, a PropTypes e a React Native Gesture Handler. Para instalá-los juntos, é necessário executar o comando *npm install --save react-native-gesture-handler prop-types react-native-sliding-modal*.

Entretanto, se o desenvolvedor estiver utilizando o React Native 0.59 ou inferior, é necessário executar o comando *react-native link react-native-gesture-handler*. Desde a versão 0.60 comandos de vinculação de bibliotecas acontecem automaticamente.

Caso o desenvolvedor possua essas dependências, basta executar o comando *npm install --save react-native-sliding-modal*.

6.2 Utilização

Após a instalação da biblioteca e suas dependências, o próximo passo é importar o componente *sliding-modal* no arquivo desejado, digitando a linha de código demonstrada abaixo.

```
import SlidingModal from 'react-native-sliding-modal';
```

Figura 2. Código para importar o sliding-modal
Fonte: (Everthon Z. Carniel, 2020)

Após importar o *sliding-modal*, é possível utilizá-lo como no exemplo abaixo.

```
<SlidingModal
 backgroundColor='white'
 borderRadius={20}
 containerType='FlatList'
 customIcon={<Icon />}
 direction='top-to-bottom'
 swipeableColor='black'
 swipeableSize={30}
 isVisible={visible}
 minimizeBorderRadius={60}
 minimizeColor='white'
 minimizeHeight={6}
 minimizeWidth={80}
 onClose={handleVisible}
>
 <ChildComponent />
</SlidingModal>
```

Figura 3. Código de uso do componente sliding-modal
Fonte: (Everthon Z. Carniel, 2020)

Na imagem acima, o *ChildComponent* representa o componente que engloba o conteúdo que será exibido no *modal* e o *Icon*, que é um componente customizado que será renderizado dentro da área arrastável, ambos podem ser definidos pelo desenvolvedor.

Ainda na imagem anterior, é possível ver um exemplo de uso de cada propriedade fornecida pelo *sliding-modal* na cor verde. Lembrando que além dessas propriedades o desenvolvedor também pode passar ao *modal* todas as *props* nativas do componente de rolagem definido, exceto a propriedade *style*.

6.3 Propriedades do sliding-modal

Na tabela a seguir serão listadas todas as *props* da biblioteca desenvolvida neste projeto, bem como suas definições. Para mais informações a respeito das *props* nativas do *ScrollView* e do *FlatList* acesse a documentação oficial do React Native disponível em https://reactnative.dev/docs/components-and-apis.

Tabela 1. Propriedades do sliding-modal

Props	Padrão	Tipo	Descrição
backgroundColor	white	string	Define a cor de fundo do componente.

borderRadius	0	number	Define o raio das bordas do componente.
containerType	ScrollView	string	Define qual componente de rolagem será usado. Identifica apenas “ScrollView” ou “FlatList” .
customIcon	undefined	element	Recebe um ícone customizado.
direction	bottom-to-top	string	Define a direção em que o *modal* deverá expandir quando for aberto. Identifica apenas “top-to-bottom” ou “bottom-to-top” como parâmetro.
swipeableColor	black	string	Define a cor de fundo da área arrastável.
swipeableSize	35	number	Define a altura da área arrastável.
isVisible	false	bool	Define a visibilidade do sliding-modal.
minimizeBorderRadius	50	number	Define o raio das bordas do ícone padrão. Será ignorada se a propriedade “customIcon” existir.
minimizeColor	white	string	Define a cor do ícone padrão. Será ignorada se a propriedade “customIcon” existir.
minimizeHeight	4	number	Define a altura do ícone padrão. Será ignorada se a propriedade “customIcon” existir.
minimizeWidth	100	number	Define a largura do ícone padrão. Será ignorada se a propriedade “customIcon” existir.
onClose	() => {}	func	Recebe uma função para ser chamada no momento que o sliding-modal for fechado.
handleVisible	() => {}	func	Recebe uma função para modificar o *state isVisible*

style	{}	object	Recebe propriedades de estilo para o componente de rolagem.

Fonte: (Everthon Z. Carniel, 2020)

Na tabela é possível identificar duas propriedades essenciais para o controle da exibição do *sliding-modal: isVisible* e *handleVisible*. Elas definem se ele está visível ou não.

Para controlar a visibilidade do *sliding-modal*, é preciso atribuir um *state* à propriedade *isVisible*, para que ele possa ser lido pelo *modal*. O *state* deve possuir uma função para ser atribuída à propriedade *handleVisible*. Desta forma, quando fechado o *modal*, o *state* será alterado, garantindo assim que o *sliding-modal* possa ser exibido e utilizado novamente.

7. CONCLUSÃO

Como dito anteriormente, o objetivo desse trabalho era desenvolver uma biblioteca que forneça um *modal* responsivo à gestos panorâmicos dos usuários. Uma vez que esta ferramenta é reutilizável e de fácil integração, ela permitirá a otimização do tempo para os desenvolvedores que a utilizarem.

Esta ferramenta denominada *sliding-modal* é capaz de detectar os gestos panorâmicos dos usuários sobre uma camada superior da aplicação, com o auxílio da biblioteca da ferramenta de gestos, e de produzir animações eficientes, sendo também um recurso funcional para iOS e Android. Além disso, esta ferramenta é compatível com a React Navigation, uma das principais características almejadas por esse trabalho, e que, consequentemente, fará com que ela possa ser utilizada em projetos React Native que utilizem essa biblioteca.

A biblioteca *sliding-modal* foi publicada em uma versão estável e totalmente funcional. Os resultados obtidos com o seu desenvolvimento foram muito animadores, o que permite que ela seja disponibilizada para a comunidade do React Native. Além disso, essa biblioteca foi publicada no NPM para que possa ser instalada em qualquer projeto para dispositivos móveis gerenciado por ele.

É fato que tudo que é novo gera certa insegurança, porém com a necessidade de uma ferramenta como essa, se torna favorável sua implementação para contribuir com o desenvolvimento móvel. Sabendo que o crescimento tecnológico é cada vez mais visível, por conta da criação de diversos recursos reutilizáveis e otimizados, se não houver contribuições que facilitem o uso desses recursos importantes, não será possível focar em características futuras e na expansão de ferramentas atuais.

8. REFERÊNCIAS

CHIJO, LETÍCIA. (2019) Como funciona a API Animated do React Native?, https://medium.com/reactbrasil/como-funciona-a-api-animated-do-react-native-288b800f68da, Outubro.

DEL COL, ARMANDO. (2019) React Native Basics: Componentes Funcionais vs Classes, https://devsamurai.com.br/react-native-componentes-funcionais-vs-classes/, Novembro.

FACEBOOK, (2017) Prop Types, https://github.com/facebook/prop-types, Abril.

FLANAGAN, DAVID. (2013) JavaScript: o guia definitivo, Bookman, cap. 1, p. 1.

NABORS, RACHEL AND WHITE, ELI. (2020) React Native: Core Components and APIs, https://reactnative.dev/docs/components-and-apis, Julho.

NOVATO, DOUGLAS. (2020) Um guia completo de React Native, https://blog.geekhunter.com.br/um-guia-completo-de-react-native/, Março.

REACT. (2020) Getting Started, https://pt-br.reactjs.org/docs/getting-started.html, Novembro.

REACT NATIVE GESTURE HANDLER. (2020) Getting Started, https://docs.swmansion.com/react-native-gesture-handler/docs/, Novembro.

RONCAGLIO, RODRIGO. (2016) Gestos básicos em mobile e a importância - que ainda não damos - a eles, https://medium.com/catarinas-design/projetando-para-gestos-6d35609fab9, Abril.