soluções mobile

prof. Thyerri Mezzari

React Native: acelerando

Recapitulando...

Ecossistema

- **JavaScript** (e/ou TypeScript) para 90% de sua codebase
- Java ou Objective-C em casos mais "hardcore"
- Use qualquer editor de código (recomendo o VS Code com plugins específicos)
- Android Studio para "ligar o emulador" ou debugs/análises mais complexas

Recapitulando...

Componentes

- **Sintaxe JSX**
- <View></View>
- <MeuProprioComponente></MeuProprioComponente>
- Não se esqueça de importar os componentes antes de usar:

```
import { Text } from "react-native";
```

Recapitulando...

Functional Components

```
function HelloMessage(props) {
    return <Text>Olá, {props.name}!</Text>;
}
```

Recapitulando...

Estilização / Personalização

- **Sintaxe JSS**
- Muito parecido com o CSS da web
- A maioria das regras de CSS funcionam, menos float e outras específicas de *background* e *box-shadow*.

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "#fff",
    alignItems: "center",
    justifyContent: "center",
  },
});
```

Recapitulando...

Frameworks visuais

Por padrão o React Native não têm e não possui nenhum componente de UI requintado para uso, mas existem alguns frameworks bacana no mercado para composição de UI:

https://callstack.github.io/react-native-paper
https://nativebase.io https://react-native-training.github.io/react-native-elements
https://github.com/wix/react-native-ui-lib

Navegação entre telas

createStackNavigator()

Rotas e Navegação

No início do projeto do React Native o framework se dispunha a oferecer algumas soluções oficiais de como criar múltiplas telas (e uma navegação entre elas) em um aplicativo. O problema encontrado na época foi que existia uma solução "interna" distinta para iOS e Android.

A do iOS possuía uma performance 100% nativa, já a do Android era "semi-emulada" e por vezes existia uma perda de desempenho entre a troca de telas.

Dada estas dificuldades com o passar do tempo a comunidade open-source se dispôs a resolver estes problemas criando soluções alternativas e complementares as oferecidas pelo RN.

Rotas e Navegação

As soluções criadas ultrapassaram em muito a performance das soluções fornecidas "de fábrica" e o core team do React Native achou por bem remover as soluções oficiais e passar a recomendar as soluções da comunidade.

Atualmente as duas principais alternativas de navegação entre tela são os plugins adicionais chamados **React Navigation** e o **React Native Navigation**, que apesar do nome parecido são bem distintos.

React Navigation

Projeto 100% open-source capitaneado pela empresa responsável pelo Expo. É atualmente a solução mais popular do mercado recomendada inclusive pelo Facebook:

https://reactnavigation.org/

***Spoiler**: É esta a solução que vamos abordar em sala de aula.*

React Native Navigation

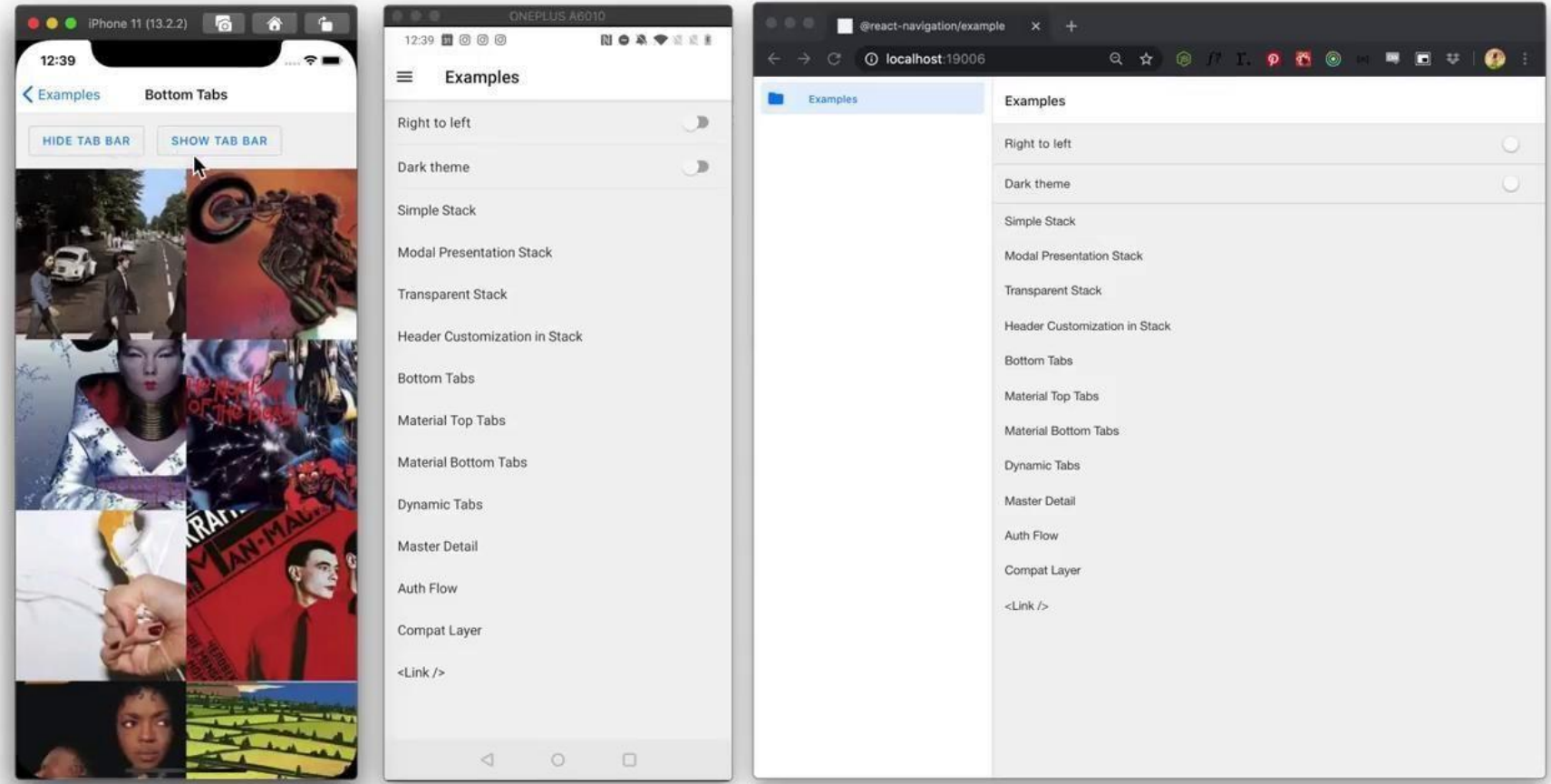
Projeto 100% open-source capitaneado pela empresa responsável pelo WIX. Este projeto têm como foco a performance máxima, a ideia é usar ao máximo a camada nativa de cada plataforma e trazer o mínimo de processamento possível para a camada do JavaScript.

Em seu uso não têm tanta atração quando a solução anterior, mas é tão bom quanto:

https://github.com/wix/react-native-navigation

Minha recomendação: REACT NAVIGATION

E minha experiência profissional com RN e Expo nunca tive uma necessidade de usar outra solução sem ser o **React Navigation** oficialmente em um projeto, sendo um projeto super robusto e sólido, proporciona uma fácil instalação e configuração tanto em um projeto ***RN padrão*** quando em um projeto baseado em **Expo**.


iPhone 11 (13.2.2)
12:39
Examples
Bottom Tabs
HIDE TAB BAR
SHOW TAB BAR
ONEPLUS A6010
12:39
Examples
Right to left
Dark theme
Simple Stack
Modal Presentation Stack
Transparent Stack
Header Customization in Stack
Bottom Tabs
Material Top Tabs
Material Bottom Tabs
Dynamic Tabs
Master Detail
Auth Flow
Compat Layer
<Link />
@react-navigation/example
localhost:19006
Examples
Examples
Right to left
Dark theme
Simple Stack
Modal Presentation Stack
Transparent Stack
Header Customization in Stack
Bottom Tabs
Material Top Tabs
Material Bottom Tabs
Dynamic Tabs
Master Detail
Auth Flow
Compat Layer
<Link />

Instalando o React Navigation

O pacote principal do projeto está localizado em **@react-navigation/native** por tanto para instalá-lo use o comando:

```
npm install @react-navigation/native
```

Além deste pacote principal será necessário configurarmos e instalarmos algumas outras dependências adicionais.

EXPO: Instalando o React Navigation

No caso do **Expo** para instalar as dependências adicionais siga este comando:

```
npx expo install react-native-screens react-native-safe-area-context
```

PS.: Esta aula está atualizada com a última versão do React Navigation, em versões anteriores esse processo de instalação era bem mais burocrático (e com mais dependências).

RN: Instalando o React Navigation

No caso de um projeto *RN padrão*, o comando para iniciar é este:

```
npm install react-native-screens react-native-safe-area-context
```

E diferente do Expo (que te ajuda em algumas coisas) no caso de estar seguindo o setup em um RN padrão será necessário finalizar a instalação com mais alguns outros passos:

https://reactnavigation.org/docs/getting-started#installing-dependencies-into-a-bare-react-native-project

Começando o React Navigation

Logo após completarmos a instalação base, devemos "focar" em analisar os requisitos de nossa aplicação e definir qual o tipo de navegação a mesma terá.

Pode parecer uma pergunta estranha, mas alguns apps são baseados em navegação simples, outros baseados em abas, alguns têm aquele menu que vem da lateral esquerda (NavigationDrawer) e em apps mais complexos podem ter tudo isso junto (e muito mais).

Começando o React Navigation

No atual estágio de evolução do projeto React Navigation os tipos principais de navegação foram "desacoplados" do projeto principal, assim poderemos instalar e usar apenas os tipos que nós precisamos em nosso app.

Iremos começar pelo tipo mais simples e mais comum, que envolve trocas de telas simples, baseado no ***native stack navigator***.

Navegação do tipo *Stack*

Stack na língua inglesa quer dizer "pilha", ou seja a ideia é "empilhar" um monte de telas e ir alternando entre elas e seria o tipo de navegação mais básico.

Para iniciarmos precisamos instalar também este "navigator" como dependência de nosso projeto:

```
npm install @react-navigation/native-stack
```

Navegação do tipo *Stack*

Uma vez que estejamos com nosso navegador instalado e a disposição, podemos começar a estruturar nosso app.

Os primeiros passos envolvem o uso de um ***NavigationContainer*** e a criação de nosso *StackNavigator* através do comando ***`createNativeStackNavigator`***.

```
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
```

Navegação do tipo *Stack*

O ***NavigationContainer*** é um componente de base, mais versátil, como o nome já diz ele serve para "conter" toda nossa aplicação no que diz respeito a organização de navegação.

Por exemplo nosso *App* principal ficaria assim:

```
function App() {
  return (
    <NavigationContainer>
      ... aqui vai o resto da aplicação
    </NavigationContainer>
  );
}

export default App;
```

Navegação do tipo *Stack*

Na sequência podemos finalmente começar a "empilhar" nossas telas e preparar uma possível navegação entre elas. Primeiro temos que criar nosso **Stack**.*Navigator*:

```
const Stack = createNativeStackNavigator();
```

Navegação do tipo *Stack*

Depois começamos a usá-lo dentro do *NavigationContainer*:

```
function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen name="Home" component={HomeScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

Navegação do tipo *Stack*

Cada tela de nosso aplicativo é considerada um *Component* por mais que seja algo "grande" e complexo. Este *Component* (ou tela) deverá ser listado dentro de nossa Stack de Navegação através de um componente do tipo ***<Stack.Screen />***.

Este componente recebe no mínimo dois atributos, um é o ***name*** (*que é como iremos localizar as telas)* e o outro é o ***component*** que é "o que" deve ser carregado em cada tela.

Navegação do tipo *Stack*

Depois de definirmos nossa Stack e listamos nossas telas dentro de nosso Navigator, fica a nosso critério iniciar a troca de telas.

O primeiro passo é entender que cada componente listado em nossa Stack receberá automaticamente uma propriedade (prop) chamada `**navigation**`.

Por exemplo, em um app com duas telas uma chamada ***HomeScreen*** e outra ***ProdutoScreen*** podemos seguir desse jeito...

Navegação do tipo *Stack*

```
function HomeScreen({ navigation }) {
  return (
    <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
      <Text>Estamos na Home</Text>
      <Button
        title="Go to Details"
        onPress={() => navigation.navigate('Produto')}
      />
    </View>
  );
}
```

Navegação do tipo *Stack*

```
function ProdutoScreen({ navigation }) {
  return (
    <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
      <Text>Estamos visualizando um Produto</Text>
      <Button
        title="Voltar"
        onPress={() => navigation.goBack()}
      />
    </View>
  );
}
```

Navegação do tipo *Stack*

Recebendo a propriedade (prop) navigation dentro de nossa "tela" podemos utilizar comandos comuns como "***navigate***" (para abrir uma nova tela) ou "***goBack***" (para voltar a tela anterior).

Agora vamos montar essa navegação básica em um exemplo de App usando Expo.

Passo a passo - BasicNavigation

- **Criar e abrir o projeto**
- npx create-expo-app BasicNavigation --template blank
- cd BasicNavigation
- code . (abre o projeto no VSCode

...

- **Instalar o React Navigation + native stack**
- npm install @react-navigation/native
- npx expo install react-native-screens react-native-safe-area-context
- npm install @react-navigation/native-stack

Passo a passo - BasicNavigation

- **Criar a HomeScreen**

```
function HomeScreen() {
  return (
    <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
      <Text>Home Screen</Text>
    </View>
  );
}
```

Passo a passo - BasicNavigation

- **Criar o Stack Navigator dentro do NavigationContainer**

```
export default function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen name="Home" component={HomeScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}

const Stack = createNativeStackNavigator();
```

- **Teste seu progresso até aqui**

Passo a passo - BasicNavigation

- **Avançando, criar uma segunda tela**

```
function DetailsScreen() {
  return (
    <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
      <Text>Details Screen</Text>
    </View>
  );
}
```

- **E add ela ao nosso Stack Navigator**

- `<Stack.Screen name="Details" component={DetailsScreen} />`

Passo a passo - BasicNavigation

- **Para iniciar nosso app abrindo na telha de detalhes usamos**
- `<Stack.Navigator initialRouteName="Details">`

- É preciso recarregar o app no emulador usando “R” para ver o resultado.

- Em seguida, poderá voltar novamente para Home, para continuarmos a atividade

- `<Stack.Navigator initialRouteName=“Home">`

Passo a passo - BasicNavigation

- Agora vamos navegar usando botões.
- Incluir um botão na HomeScreen e usar a prop navigation

```
function HomeScreen({ navigation }) {
  return (
    <View style={{ flex: 1, alignItems: "center", justifyContent: "center" }}>
      <Text>Home Screen</Text>
      <Button
        title="Ir para detalhes"
        onPress={() => navigation.navigate("Details")}
      ></Button>
    </View>
  );
}
```

Passo a passo - BasicNavigation

- Agora vamos navegar usando botões.
- Incluir um botão na DetailsScreen e usar a prop navigation

```
function DetailsScreen({ navigation }) {
  return (
    <View style={{ flex: 1, alignItems: "center", justifyContent: "center" }}>
      <Text>Details Screen</Text>
      <Button
        title="Voltar para home"
        onPress={() => navigation.goBack()}
      ></Button>
    </View>
  );
}
```

Recomendo

Recomendo dar uma lida nestes links também:

https://reactnavigation.org/docs/headers

https://reactnavigation.org/docs/header-buttons

https://reactnavigation.org/docs/nesting-navigators

https://reactnavigation.org/docs/params

Outros tipos de navegação

Tab navigation, Drawer...

Tab navigation

Tipo de navegação bem popular, costuma organizar um menu de abas na parte de baixo da tela.

Caso necessite desse tipo de navegação siga o link abaixo:

https://reactnavigation.org/docs/tab-based-navigation

Tab navigation *Android Like*

Tipo de navegação bem popular, costuma organizar um menu de abas na parte de cima da tela, bem comum em apps Android.

Caso necessite desse tipo de navegação siga o link abaixo:

https://reactnavigation.org/docs/material-top-tab-navigator

Drawer navigation

Tipo de navegação baseado em menu "sacado" da lateral esquerda de seu app (normalmente).

Caso necessite desse tipo de navegação siga o link abaixo:

https://reactnavigation.org/docs/drawer-based-navigation

bonus round:
Consumindo dados on-line

Requisições HTTP

Uma das coisas que mais facilitam o uso do RN é como algumas tarefas se tornam triviais de acordo com o ecossistema JavaScript.

Na web temos a possibilidade de consumir dados on-line através de técnicas baseadas em "AJAX".

Temos como usar nativamente o método ***fetch*** ou o objeto mais clássico ***XMLHttpRequest*** e em React Native é **exatamente igual**.

Requisições HTTP

Por exemplo, supondo que desejamos acessar uma API que lista filmes, uma busca no caso, podemos usar o fetch para fazer algo simples assim:

```
const response = await fetch('http://www.omdbapi.com/?s=spider%20man&apikey=1cd66749');
```

Vale a pena dar uma olhada no guia sobre "Networking" da própria documentação do RN:

https://reactnative.dev/docs/network

Requisições HTTP

Outros projetos mais avançados de requisições HTTP da comunidade React / JS também são bem usados em ***React Native***:

https://github.com/tannerlinsley/react-query

https://swr.vercel.app/

https://github.com/axios/axios

Requisições HTTP

Agora vamos implementar algumas ideias de requisições on-line em nosso app de exemplo.

Passo a passo - BasicNavigation - Http

- **Crie uma nova tela - MoviesScreen.**
- **Incluir um botão na DetailsScreen para chamar a Movies**

- `<Stack.Screen name="Movies" component={MoviesScreen} />`
-
```
<Button
      title="Listar filmes"
      onPress={() => navigation.navigate("Movies")}
      ></Button>
```

Passo a passo - BasicNavigation - Http

- **Na movies screen, precisamos buscar os dados dos filmes**

```
const [isLoading, setLoading] = useState(true);
const [data, setData] = useState([]);

const getMovies = async () => {
    try {
      const response = await fetch("https://reactnative.dev/movies.json");
      const json = await response.json();
      setData(json.movies);
    } catch (error) {
      console.error(error);
    } finally {
      setLoading(false);
    }
  };
```

Passo a passo - BasicNavigation - Http

- **E chamamos a busca usando useEffect (no inicio do componente)**

```
useEffect(() => {
    getMovies();
  }, []);
```

Passo a passo - BasicNavigation - Http

- **E aqui a camada visual da tela, usando FlatList**

```
<View style={{ flex: 1, padding: 24 }}>
      {isLoading ? (
        <ActivityIndicator />
      ) : (
        <FlatList
          data={data}
          keyExtractor={({ id }) => id}
          renderItem={({ item }) => (
            <Text>
              {item.title}, {item.releaseYear}
            </Text>
          )}
        ></FlatList>
      )}
    </View>
```

Próximos passos

Em nossas próximas aulas iremos partir para os conceitos finais de RN:

- Permissions
- AsyncStorage
- expo-sqlite
- react-native-sqlite-storage

Fique atento as datas

Fique atento as nossas próximas aulas e entregas:

- 28/05/2024 – Aula **remota** – Produção do ABP.
- 04/06/2024 – Persistência de dados, Permissões e publicação de App
- 11/06/2024 23h59 – Entrega atividade To Do List - Peso 1,5
- 11/06/2024 – Produção supervisionada do ABP. Aula **Presencial**. Vou avaliar como está o desenvolvimento do projeto. Aproveitem também para tirar as últimas dúvidas para a N2.
- 18/06/2024 – N2 – Avaliação individual peso 8,0 – Desenvolvimento híbrido.
- 25/06/2024 – Apresentação ABP (entrega do fonte pode ser até 27/06).

obrigado 🚀