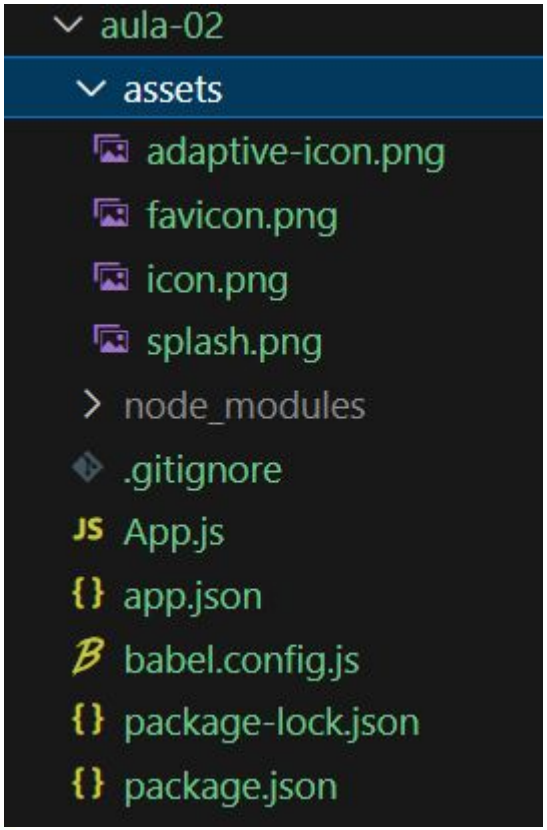
Programação Mobile
React Native - Aula 02

Professor: João Felipe Bragança

Relembrando Aulas Anteriores

- É necessário ter o Node.JS instalado na máquina para rodar o React Native
- Existem duas formas de iniciar projetos, com Expo e com CLI
- CLI exige configurações avançadas enquanto Expo basta digitar um comando
- Apps Nativos utilizam linguagens próprias da plataforma, são mais performáticos e oferecem integração total com o hardware
- Apps Híbridos são multiplataforma, reduzindo o tempo e o custo do projeto
- Para criar um projeto com Expo, basta digitar *npx create-expo-app nome-do-projeto*
- Para rodar o projeto, é necessário estar no mesmo nível que o *package.json*
- Para entrar na pasta digite *cd nome-do-projeto*
- Para iniciar digite *npx expo start*
- Podemos visualizar o app no celular através do Expo Go, utilizando o emulador do Android Studio, Genymotion ou através de visualizações web como o Navegador ou a extensão Mobile View do vs code.
- Também é possível espelhar a tela do celular no computador com programas como o LetsView

Estrutura de um Projeto Expo



Ao criar um projeto Expo com o comando npx create-expo-app, a estrutura padrão vem como na figura ao lado.

Dentro da pasta assets, temos:

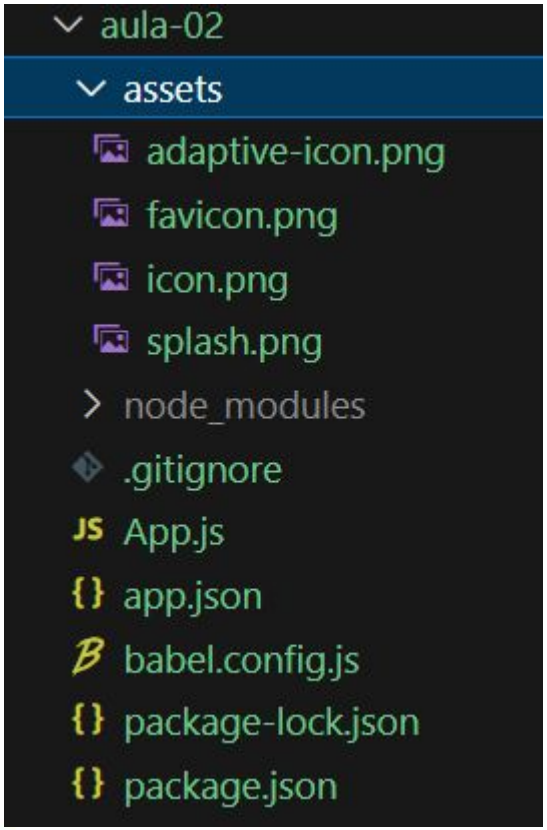
- O adaptive-icon.png usado como ícone no android
- O icon.png usado como ícone no IOS
- O favicon.png usado como ícone na web
- A splash.png, imagem exibida rapidamente quando iniciamos um aplicativo. Chamado de splash screen

Podemos ter imagens em qualquer pasta do projeto, não apenas dentro desta pasta.

Atenção: O Expo não consegue exibir diretamente imagens .svg diretamente, são necessários passos adicionais.

Saiba mais em:
https://docs.expo.dev/ui-programming/using-svgs/

Estrutura de um Projeto Expo



- node_modules contém o arquivo de todas as dependências do projeto. Essa pasta pode ser excluída e é ignorada pelo git
- .gitignore lista todos os arquivos que não precisam e não devem ser versionados
- App.js é o ponto de entrada de um projeto Expo, todos os demais componentes são chamados a partir daqui.
- app.json contém algumas configurações do app, como ícones, splash screen, orientação, etc.
- babel.config.js configura o babel, uns transpilador javascript que transforma o javascript moderno em JS compatível com versões anteriores, entre outras funcionalidades

Estrutura de um Projeto Expo

```
{
  "name": "aula-02",
  "version": "1.0.0",
  "main": "node_modules/expo/AppEntry.js",
  ▷ Debug
  "scripts": {
    "start": "expo start",
    "android": "expo start --android",
    "ios": "expo start --ios",
    "web": "expo start --web"
  },
  "dependencies": {
    "expo": "~50.0.14",
    "expo-status-bar": "~1.11.1",
    "react": "18.2.0",
    "react-native": "0.73.6"
  },
  "devDependencies": {
    "@babel/core": "^7.20.0"
  },
  "private": true
}
```

- package.json é o coração de todo projeto node. Ele contém informações sobre nome do app, versão, quais scripts podem ser executados e as dependências e dependências de desenvolvimento
- Este arquivo é necessário para startar um projeto com o comando *npx expo start*
- Quando um projeto é clonado de um repositório, o comando *npm install* baixa todas as dependências listadas e armazena tudo na pasta node_modules. Por isso não é necessário versionar tal pasta.
- Dependências de Desenvolvimento servem para nos auxiliar em tempo de desenvolvimento, como por exemplo o typescript.

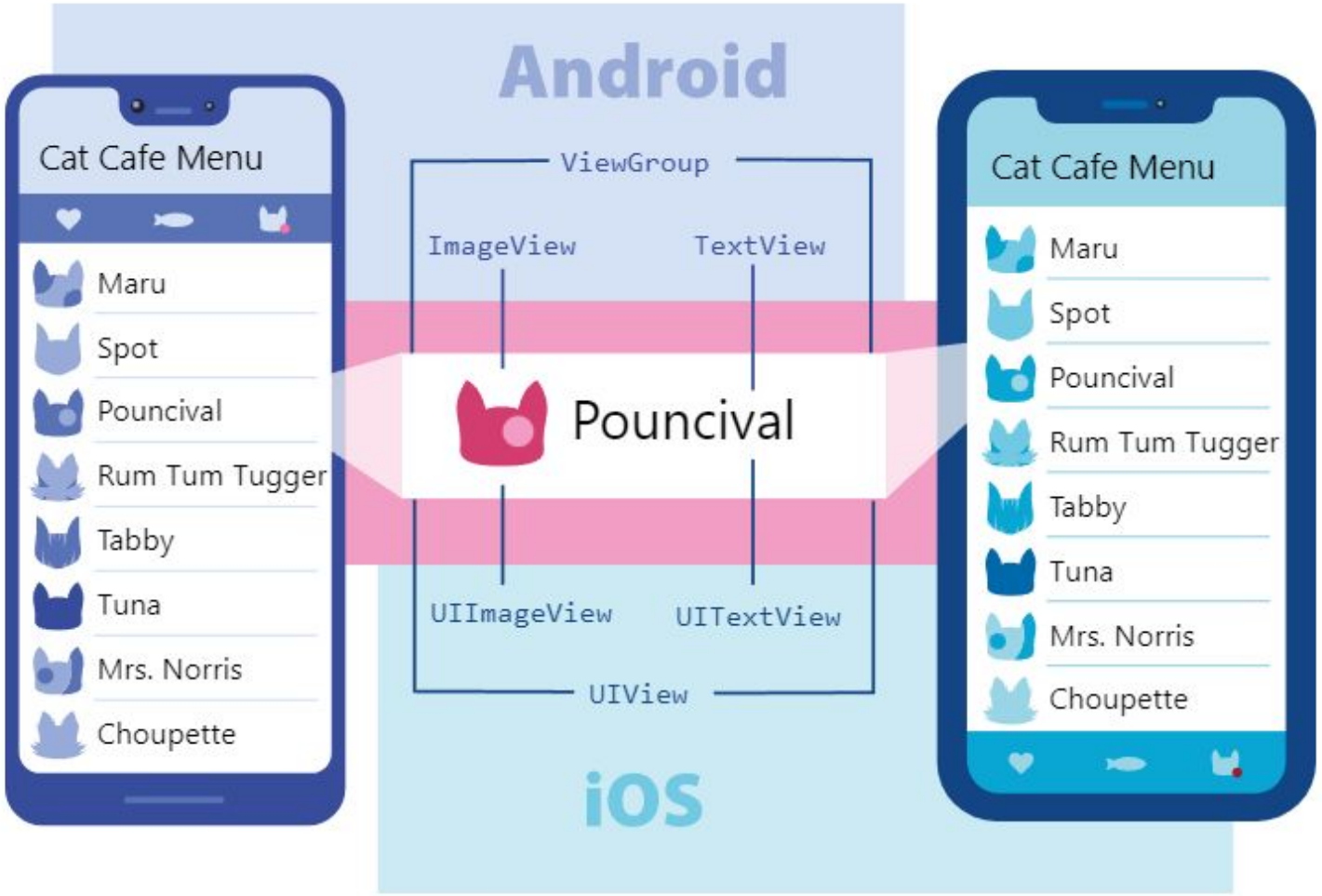
App.js

```
1  //Área onde são importados dos os componentes utilizados
2  import { StatusBar } from "expo-status-bar";
3  import { StyleSheet, Text, View } from "react-native";
4
5  //Componetente funcional (Uma função que começa com a letra maiúscula e retorna um elemento visual - JSX)
6  //Componentes precisam ser exportados para poderem ser utilizados pela aplicação
7  export default function App() {
8    //Área para variáveis, estados e funções
9    //Aqui é feita toda a lógica da aplicação
10
11   //Dentro do return temos uma sintaxe chamada JSX!
12   //Pode Retornar Apenas 1 elemento pai, como uma <View></View> ou um fragmente <></>
13   //Dentro do elemento pai podem ter quantos elementos se queira
14   return (
15     <View style={styles.container}>
16       {/* A maioria dos componentes possuem o atributo style, onde é possível passar objetos de estilização*/}
17       <Text>Hello World!!!</Text>
18       {/* Para comentar dentro do JSX, é necessário envolver o bloco de comentário com chaves (bigodes) {} */}
19       <StatusBar style="auto" />
20     </View>
21   );
22 }
23
24 //Para criar objetos de estilização de forma parecida com CSS, utiliza-se o Stylesheet do react-native
25 const styles = StyleSheet.create({
26   container: {
27     flex: 1,
28     backgroundColor: "#fff",
29     alignItems: "center",
30     justifyContent: "center",
31   },
32 });
```

Conceitos Importantes

- StyleSheet não é CSS, mas tem um conjunto de estilos super parecidos
- As propriedades são escritas utilizando o padrão lowerCamelCase. A primeira letra é minúscula, não tem espaços nem traços, e cada palavra subsequente começa com letra maiúscula
- Exemplos: backgroundColor, fontSize, flexDirection, justifyContent, alignItems
- Diferentemente do CSS, não existe herança. Os estilos devem ser aplicados diretamente onde se deseja utilizar
- Diferentemente do HTML, não podemos ter textos “soltos”, sempre é necessário ter um componente em volta do texto para deixar explícito para o React Native saber como converter para o nativo.

Componentes Nativos



Comparativo Entre Componentes

REACT NATIVE UI COMPONENT	ANDROID VIEW	IOS VIEW	WEB ANALOG	DESCRIPTION
`<View>`	`<ViewGroup>`	`<UIView>`	A non-scrolling `<div>`	A container that supports layout with flexbox, style, some touch handling, and accessibility controls
`<Text>`	`<TextView>`	`<UITextView>`	`<p>`	Displays, styles, and nests strings of text and even handles touch events
`<Image>`	`<ImageView>`	`<UIImageView>`	`<img>`	Displays different types of images
`<ScrollView>`	`<ScrollView>`	`<UIScrollView>`	`<div>`	A generic scrolling container that can contain multiple components and views
`<TextInput>`	`<EditText>`	`<UITextField>`	`<input type="text">`	Allows the user to enter text

Saiba Mais

- Mais sobre Nativo e React Native: https://reactnative.dev/docs/intro-react-native-components
- Core Components: https://reactnative.dev/docs/components-and-apis
- View: https://reactnative.dev/docs/view
- Text: https://reactnative.dev/docs/text
- StyleSheet: https://reactnative.dev/docs/stylesheet
- Image: https://reactnative.dev/docs/image
- Button: https://reactnative.dev/docs/button

Obrigado(a)!