Curso: Desenvolvimento de Software Multiplataforma

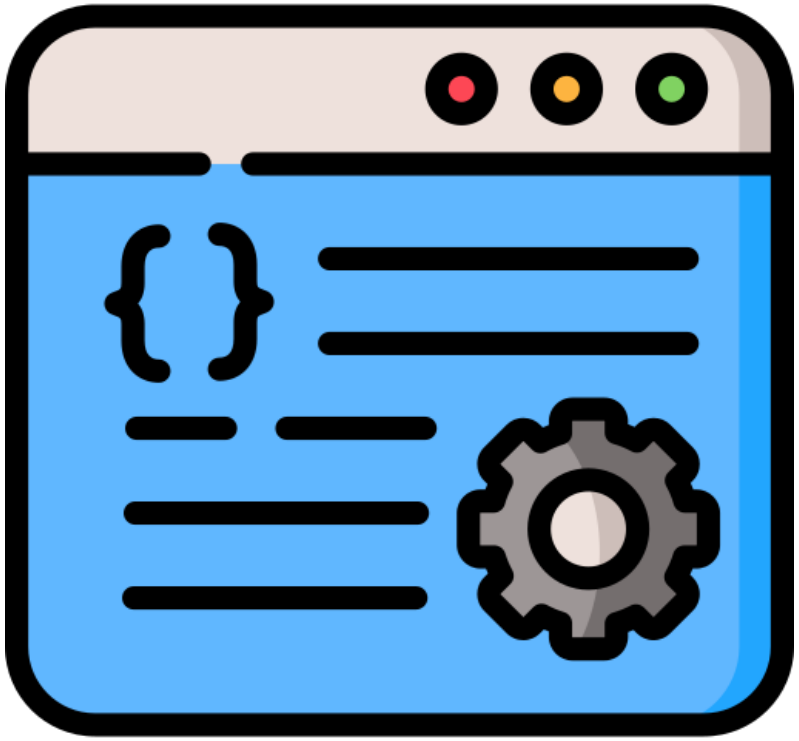
Disciplina: Programação para Dispositivos Móveis I

Professor: André Olímpio

Programação Para Dispositivos Móveis I

Shadow Tree

- É uma estrutura de dados interna que o RN utiliza para gerenciar a interface do usuário (UI).
- O código em RN é definido através de componentes e suas propriedades no JS.
- Esses componentes são convertidos em uma árvore de componentes nativos, conhecida como **“shadow tree”.**



Fonte: Flaticon.com

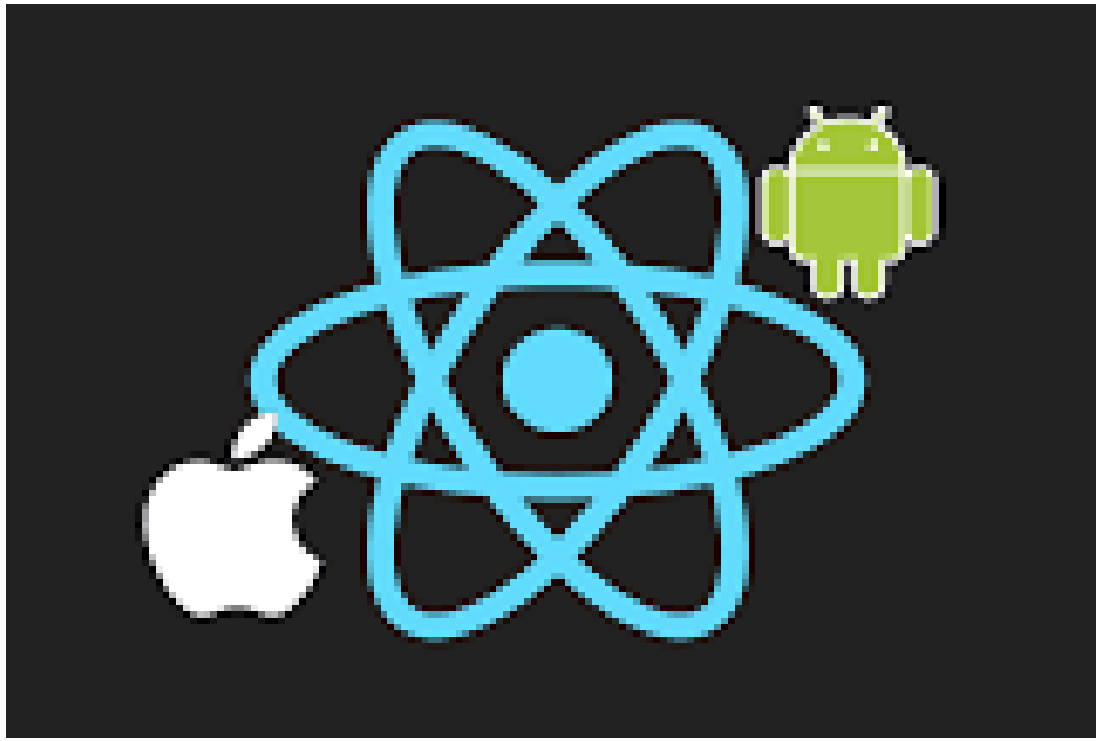
Shadow Tree

- **Funções e benefícios do Shadow Tree:**

1. **Gerenciamento de layout:** é responsável por calcular o layout de todos os componentes da UI, utilizando o algoritmo de layout Flexbox para determinar a posição e o tamanho dos componentes.
2. **Intermediação entre JS e nativo:** serve como uma representação intermediária dos componentes JS antes de serem convertidos em componentes nativos, facilitando a comunicação eficiente entre o código JS e a UI nativa.
3. **Performance:** ao gerenciar o layout e outras operações de UI, o RN pode reduzir a quantidade de operações que precisam ser executadas diretamente na UI thread, melhorando o desempenho e a suavidade da aplicação.
4. **Isolamento:** permite que o layout e outras operações sejam calculados de forma independente do thread de UI principal, reduzindo a latência e melhorando a responsividade da aplicação.

Native Modules

- São componentes que permitem que o código JS interaja com APIs nativas do sistema operacional (iOS e Android), sendo essenciais para acessar funcionalidades específicas do dispositivo que não estão disponíveis diretamente no JS.



Fonte: Medium.com

Native Modules

- **Funções e Benefícios dos Native Modules:**

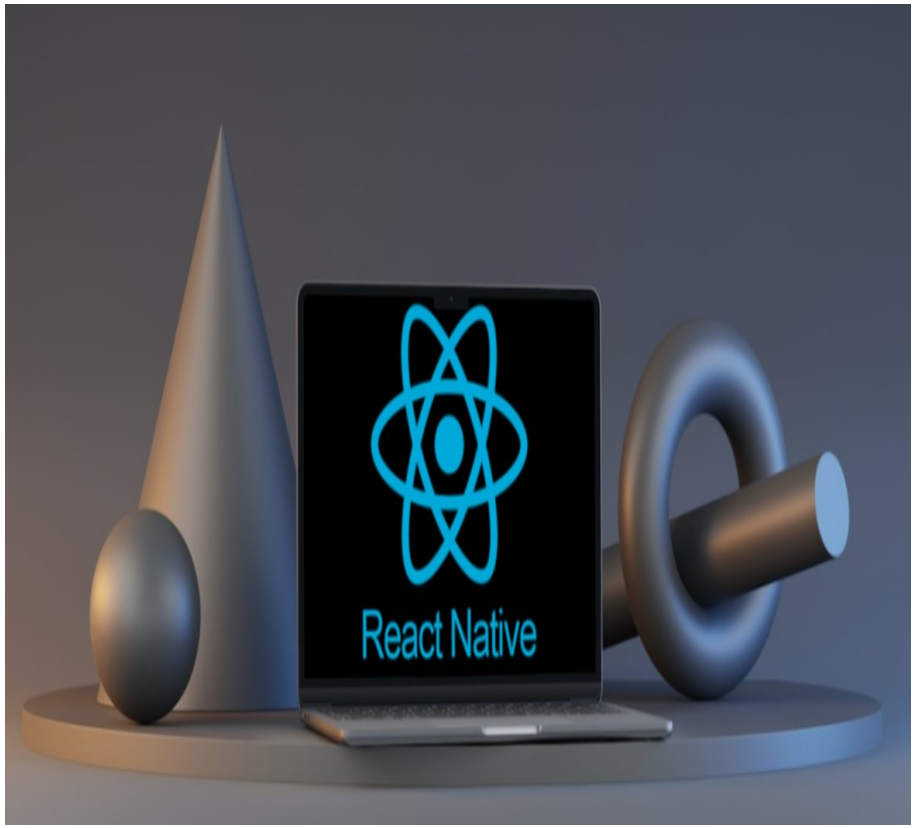
1. **Acesso a funcionalidades nativas:** permitem que o código JS chame funcionalidades específicas do sistema operacional, como acesso ao GPS, câmera ou armazenamento.
2. **Interoperabilidade:** os native modules permitem escrever código nativo em Java (para Android) ou Objective-C/Swift (para iOS) e exponham essas funcionalidades para o JS através de uma interface consistente.
3. **Performance:** alguns cálculos ou operações podem ser mais eficientes quando executados no código nativo, e os native modules permitem essa otimização.
4. **Extensibilidade:** permitem que os desenvolvedores estendam as capacidades do RN, integrando bibliotecas e SDKs nativos que não têm uma contraparte JS.

Blunders

- Conhecidos como **empacotadores**.
- Na construção de aplicativos usando frameworks como RN e Expo, os bundlers desempenham papéis distintos em relação ao processamento e à preparação do código para as diferentes plataformas.
- São dois:
  - **Metro Bundler.**
  - **Web Bundler.**

Metro Bundler

- **Papel:** é o empacotador de módulos utilizado pelo RN para aplicativos móveis, responsável por compilar e agrupar todos os módulos JS do aplicativo em um único arquivo JS que pode ser carregado pelo aplicativo móvel.
- **Uso:** é automaticamente utilizado quando executamos o comando expo start para iniciar o servidor de desenvolvimento.



Fonte: MicroVerse.org

Metro Bundler

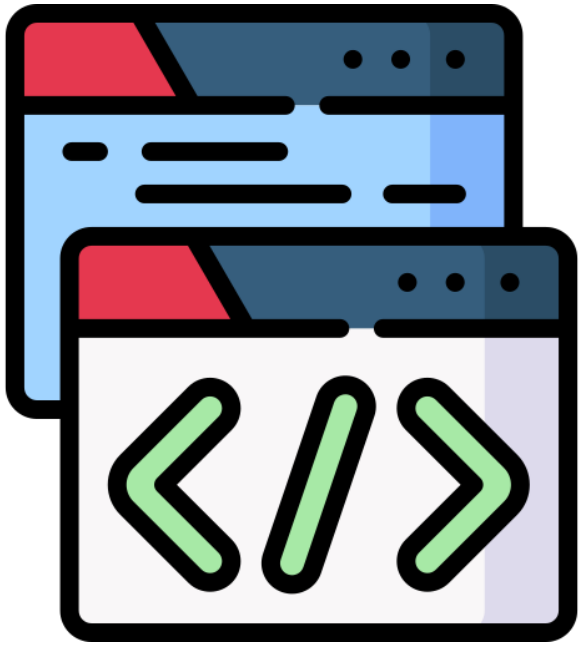
- **Funções:**
  - **Transformação de código:** converte o código JS moderno e JSX (sintaxe utilizada pelo React) em um formato que pode ser compreendido pelos ambientes de execução em dispositivos móveis.
  - **Empacotamento de módulos:** agrupa todos os módulos JS e dependências em um único arquivo para otimizar o carregamento do aplicativo.
  - **Atualizações e hot reloading:** suporta atualização rápida e hot reloading para que possamos ver as mudanças no código quase instantaneamente sem precisar recompilar o aplicativo inteiro.
  - **Resolução de dependências:** gerencia e resolve dependências de módulos JS, garantindo que todas as dependências sejam incluídas no pacote final.

Hot Realod

- É uma técnica que permite a atualização instantânea de uma aplicação em execução, sem a necessidade de reiniciar o servidor ou recompilar o código.
- Essa funcionalidade é especialmente popular entre desenvolvedores web, pois proporciona uma experiência de desenvolvimento mais ágil e eficiente.
- Com o Hot Reload, as alterações feitas no código são refletidas imediatamente na interface do usuário, permitindo que os desenvolvedores visualizem as mudanças em tempo real.

Web Bundler

- **Papel:** é utilizado para criar pacotes otimizados de código JS. Em um contexto de desenvolvimento com RN para Web ou projetos Expo, o Web Bundler é responsável por preparar o código para execução em navegadores.
- **Uso:** é utilizado automaticamente quando executamos comandos como expo start para a versão web do projeto ou quando utilizamos ferramentas como Webpack em projetos React.



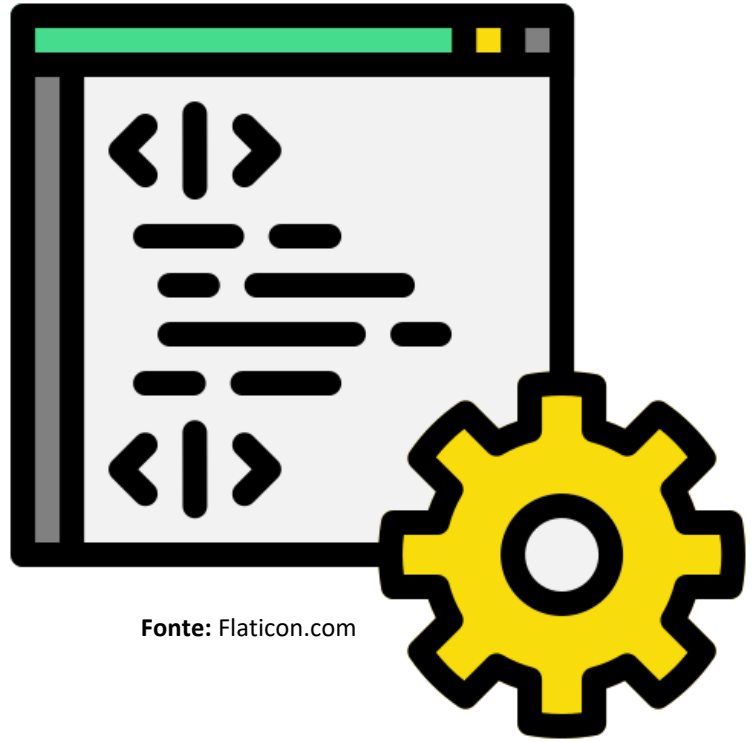
Fonte: Flaticon.com

Web Bundler

- **Funções:**
  - **Transformação de código:** transforma código JS moderno e JSX para um formato compatível com navegadores web.
  - **Empacotamento e minificação:** agrupa e minifica o código JS e os recursos estáticos (como CSS e imagens) para otimizar o desempenho no navegador.
  - **Resolução de dependências:** resolve e inclui todas as dependências necessárias para a execução do aplicativo no ambiente web.
  - **Hot reloading:** para desenvolvimento rápido na web, semelhante ao que é suportado pelo Metro Bundler para dispositivos móveis.

Web Bundler

- **O que é minificação?**
  - Minificar significa reduzir o tamanho do código JS.
  - Isso pode ser feito antes de implementar um site em produção.
  - Código JS não minificado pode tornar uma página mais lenta para os usuários.



Fonte: Flaticon.com

React Navigation

- A navegação entre telas é um aspecto fundamental na construção de aplicativos mobile com RN.
- Permite que o usuário se mova fluidamente entre diferentes partes do aplicativo, proporcionando uma experiência intuitiva.
- A biblioteca mais utilizada para gerenciar a navegação é o **React Navigation (**https://reactnavigation.org**)**.
- Oferece uma interface flexível e poderosa para criar diferentes tipos de fluxos de navegação, como pilhas (stacks), abas (tabs) e menus (drawers).

React Navigation - Fundamentos

- **Navegadores:** são os componentes que gerenciam a transição entre as telas.
- Os principais tipos são:
  - StackNavigator: cria uma pilha de telas, onde cada tela é empilhada sobre a anterior. É ideal para navegação hierárquica.
  - TabNavigator: cria uma barra de abas, permitindo navegar entre diferentes telas.
  - DrawerNavigator: cria um menu lateral com as opções de navegação.

React Navigation - Fundamentos

- **Rotas:** definem as telas do aplicativo e seus nomes. Cada rota é associada a um componente de tela.
- **Navegação:** O processo de mover o usuário entre as rotas. É realizado através de métodos como `navigation.navigate` e `navigation.goBack`.

Fonte: Flaticon.com

React Navigation - Fundamentos

- Para configurar a navegação no RN utilizando a biblioteca **@react-navigation/native** é necessário instalar algumas dependências.
- Cada uma delas desempenha um papel específico na configuração e no funcionamento da navegação no aplicativo:
  - **npm i @react-navigation/native @react-navigation/native-stack**
  - **npm i react-native-screens react-native-safe-area-context**

Dependências do React Navigation

1. **@react-navigation/native** (https://www.npmjs.com/package/@react-navigation/native): biblioteca central para navegação.

- **Função:** esta é a biblioteca principal que fornece as funcionalidades básicas de navegação no RN, gerenciando o estado da navegação e o deep linking entre telas;

- **Necessidade:** sem esta dependência, não é possível implementar a navegação em um aplicativo RN de forma estruturada e consistente.

Dependências do React Navigation

2. **@react-navigation/native-stack** (https://www.npmjs.com/package/@react-navigation/native-stack): implementa a navegação em pilha.

- **Função:** esta dependência fornece a implementação da navegação em pilha (stack navigation), que é uma das formas mais comuns de navegação. Uma pilha permite navegar de uma tela para outra e retornar à tela anterior.
- **Necessidade:** a navegação em pilha é essencial para a maioria dos aplicativos, onde o usuário pode navegar por várias telas e, em seguida, voltar para a tela anterior.

Dependências do React Navigation

3. **@react-native-screens** (https://www.npmjs.com/package/react-native-screens): otimiza o desempenho de navegação.

- **Função:** esta biblioteca melhora o desempenho da navegação no RN ao usar componentes nativos de gerenciamento de telas, permitindo que as telas sejam renderizadas de forma mais eficiente, especialmente em dispositivos móveis.
- **Necessidade:** embora opcional, react-native-screens é altamente recomendado para melhorar o desempenho e a experiência do usuário em aplicativos que utilizam navegação.

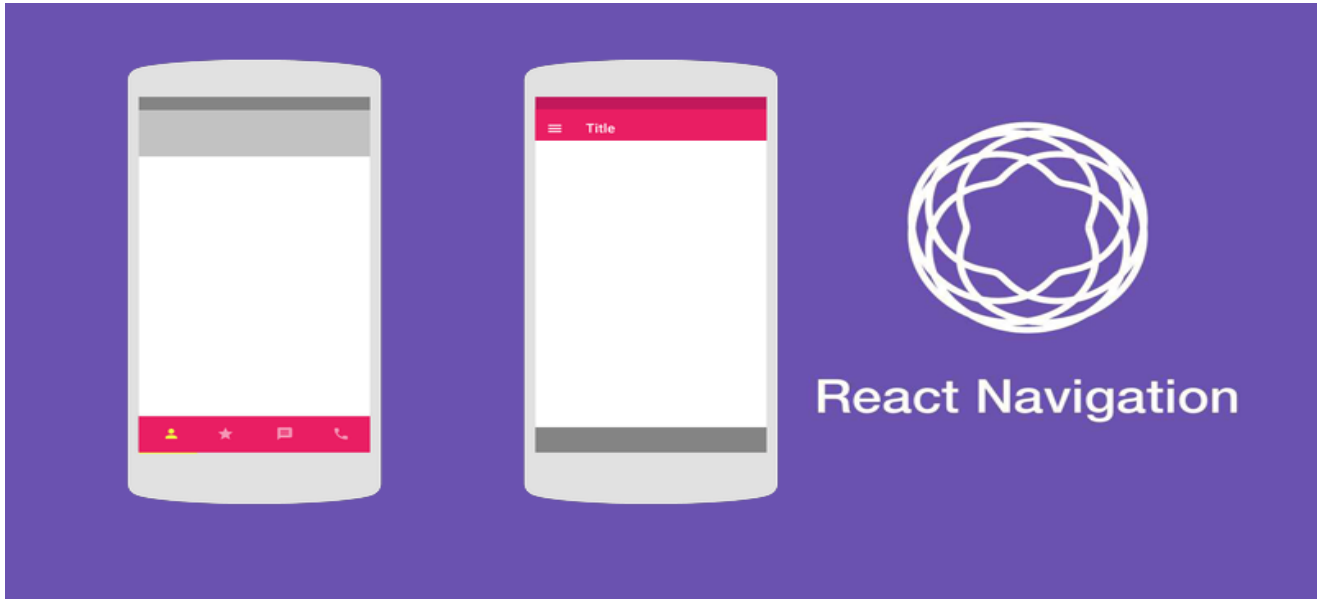
Dependências do React Navigation

4. **@react-native-safe-area-context** (https://www.npmjs.com/package/react-native-safe-area-context): gerencia áreas seguras para evitar sobreposição de conteúdo.

- **Função:** esta biblioteca lida com as áreas seguras (safe areas) do dispositivo, como as margens ao redor do *notch* em iPhones, bordas arredondadas e barras de navegação em dispositivos Android.
- **Necessidade:** garantir que o conteúdo não seja sobreposto por áreas não utilizáveis da tela é crucial para a experiência do usuário. Esta biblioteca nos permite gerenciar essas áreas com facilidade.

Navegação com StackNavigator

- Para implementar a navegação entre telas utilizando **@react-navigation/native**, três componentes desempenham papéis fundamentais:
  - **NavigationContainer**
  - **Stack.Navigator**
  - **Stack.Screen**



Fonte: Medium.com

Navegação com StackNavigator

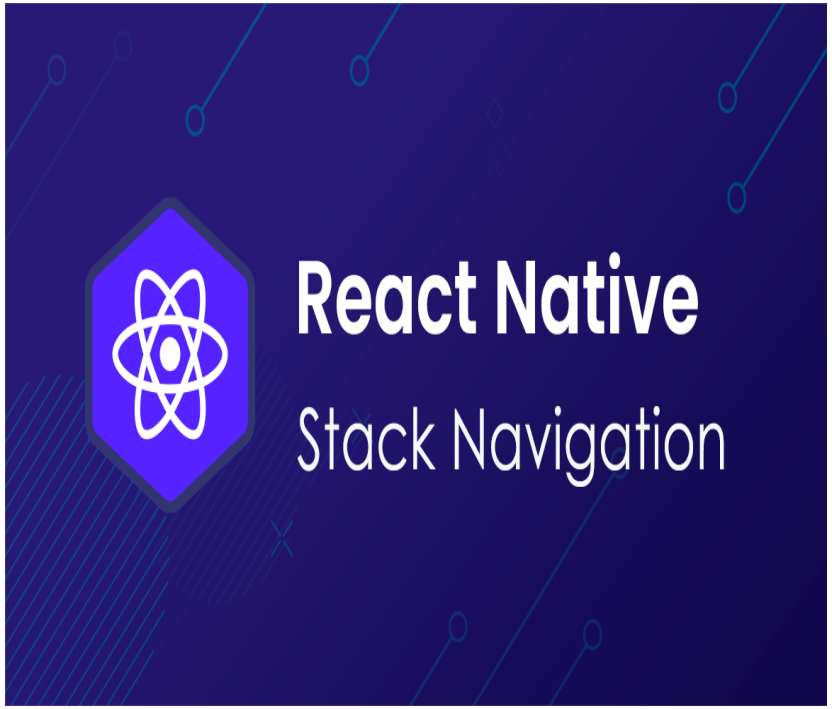
- **NavigationContainer:**
  - É o componente principal que gerencia o estado de navegação de toda a aplicação.
  - Atua como um contêiner para todos os navegadores (navigators) e deve envolver o `Stack.Navigator`.
  - Sem o `NavigationContainer`, a navegação não funcionará, pois é responsável por fornecer o contexto de navegação e garantir que o histórico de navegação seja mantido;

Navegação com StackNavigator

- **Stack.Navigator:**
  - É o componente que define o escopo do *stack navigator*, ou seja, organiza as telas em uma estrutura de pilha.
  - É responsável por criar o contexto de navegação entre as telas definidas como Stack.Screen.
  - Também permite configurar as opções de navegação, como o *comportamento do botão de retorno, a animação entre as telas* e *o título do cabeçalho.*

Navegação com StackNavigator

- **Stack.Screen:**
  - É utilizado para definir cada tela que faz parte do stack navigator.
  - Cada Stack.Screen corresponde a uma rota no navegador, especificando um nome (name) e um componente (component) que será renderizado quando essa rota estiver ativa.



Fonte: DevMedia.com.br

Navegação com TabNavigator

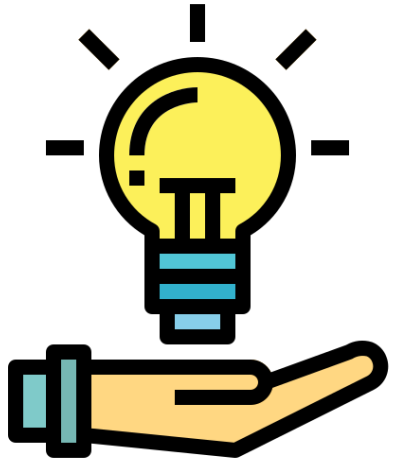
- O ***TabNavigator*** é um conjunto de abas na parte superior ou inferior da tela do dispositivo.
- O primeiro passo é instalar o pacote **@react-navigation/bottom-tabs** (https://www.npmjs.com/package/@react-navigation/bottom-tabs):
  - **npm i @react-navigation/bottom-tabs**
- No componente App temos de definir a navegação por abas utilizando:
  - **const Tab = createBottomTabNavigator<RootStackParamList>();**

Fonte: Flaticon.com

Navegação com DrawerNavigator

- O *DrawerNavigator* é o menu na lateral esquerda e as vezes colocados na direita.
- Será necessário instalar a seguinte dependência:
  - **npm i @react-navigation/drawer**
- No componente App é preciso definir a navegação utilizando **Drawer**:
  - **const Drawer = createDrawerNavigator<RootStackParamList>();**

Leitura Complementar



Fonte: Flaticon.com

- **Antes de Criar um APP: Tudo que você precisa saber sobre o Expo**

https://www.youtube.com/watch?v=2xIpIwil3-I

Tipos de navegação no React Native

https://blog.rocketseat.com.br/navegacao-react-native/

- **React Navigation: Routing and navigation for React Native and Web apps**

https://reactnavigation.org/