
# Treinamento Aplicativo

Área de Produto

Versão 1.0

# *React Native* – Changelog

| Versão | Data | Autor | Descrição |
|---|---|---|---|
| 0.1 | 06/12/2019 | Marcelo Higuti | Criação do documento. |
| 1.0 | 10/01/2020 | Renato Yamashita | Transcrição do documento original para este formato. |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# *React Native* – Conteúdo

Neste documento estão as atividades relacionadas ao Aplicativo da nossa aplicação:

- **Introdução**
- **Configurações iniciais**;
- **Estrutura do projeto**
- **Exercícios**

# *React Native* – Introdução

O *React Native* é um *framework* utilizado para desenvolver aplicativos para *Android* e para *iOS* de forma nativa.

A seguir, será apresentado um treinamento de *React Native* para que você entenda os conceitos básicos da tecnologia e também verifique como é a estrutura básica do projeto do nosso aplicativo.

Para a criação do treinamento foi utilizado o *React Native 0.61*.

# *React Native* – Configurações iniciais I

- Clone o repositório do treinamento se não o tiver feito;
- Abra o cmd.exe, entre na pasta do repositório e navegue até o diretório "***\react-native-tutorial\app***";
- Execute o comando:
  - ***npm install***
- Assim que terminarem as instalações, execute o comando:
  - ***react-native run-android***

# *React Native* – Configurações iniciais II

- Em casos de erros durante a compilação, siga os seguintes passos:
  - Entre na pasta "***\react-native-tutorial\app\android***" pelo *cmd.exe*;
  - Execute o comando:
    - ***gradlew clean***
  - Volte para a raiz com o comando:
    - **cd ..**
  - Rode o projeto novamente com:
    - ***react-native run-android***

# *React Native* – Estrutura do projeto I

A estrutura de pastas do projeto tem um organização específica:

- ***\App.js*** – é o primeiro script a ser executado quando o aplicativo é aberto, aqui são feitas configurações globais que serão aplicadas para todo o projeto (ex.: configuração de fontes, *redux*, componente a ser 'renderizado' inicialmente);
- ***\package.json*** – é o ponto de partida de qualquer projeto *Node.js*. Ele é responsável por descrever o seu projeto, informar as *engines* (versão do *node* e do *npm*), versão do projeto, dependências de produção e de desenvolvimento, dentre outras coisas;

# *React Native* – Estrutura do projeto II

- ***android* e *ios*** – pastas que contêm as configurações específicas para cada plataforma que devem ser configuradas de acordo com as funcionalidades desejadas (bibliotecas externas, *links*, permissões);
- ***src*** – pasta principal com os códigos *JavaScript* que serão utilizado para construir o aplicativo para as duas plataformas (*Android* e *iOS*).

# *React Native* – Estrutura do projeto III

O projeto utiliza uma estrutura **MVC (Model, View, Controller)** e a pasta "***\src***" está segmentada dessa forma:

- ***\assets*** – utilizada para armazenar arquivos estáticos como fontes, ícones, imagens, etc.;
- ***\classes*** – *scripts* de classes "pais" que são herdadas por outros componentes e que já estão configuradas para certas funcionalidades (ex.: conexão com o banco de dados);
- ***\components*** – aqui estão todos os componentes criados para serem reutilizados no aplicativo, garantindo uma identidade visual e modularização, (ex.: botão, *input*, listas);
- ***\navigation*** – os arquivos aqui são utilizados para configurar as rotas, telas e *tabs* que serão utilizadas para navegar dentro do aplicativo.

# *React Native* – Estrutura do projeto IV

- ***\pages*** – todos os *scripts* de telas do aplicativo devem ser armazenados dentro dessa pasta. Normalmente aqui são criadas pastas com três arquivos diferentes, vamos utilizar como exemplo a entidade "*customer*" (*src\pages\customer*):
  - ***CustomerModel.js*** – arquivo que estende a "*Model.js*" da pasta "\classes" e define a estrutura e informações que devem ser recuperados do banco de dados;
  - ***CustomerController.js*** – arquivo que possui as lógicas e regras de negócio que a entidade deve seguir para realizar as operações necessárias, normalmente importa o arquivo e funções da "*Model*";
  - ***Customer.js*** – arquivo que representa a "*View*", aqui estão os componentes que serão utilizados para construir a tela e importa as lógicas do arquivo "*Controller*".

# *React Native* – Estrutura do projeto V

- ***redux*** – arquivos de configuração básica do *redux*, possui a *store* e e o *RootReducer* que junta todos os *Reducers* dos diversos componentes;
- ***utils*** – contém o arquivo *"Utils.js"* que possui diversas funções genéricas e que normalmente são utilizadas em vários componentes diferentes do *app*.

# *React Native* – Exercício

Finalizando esta atividade você terá realizado as seguintes tarefas:

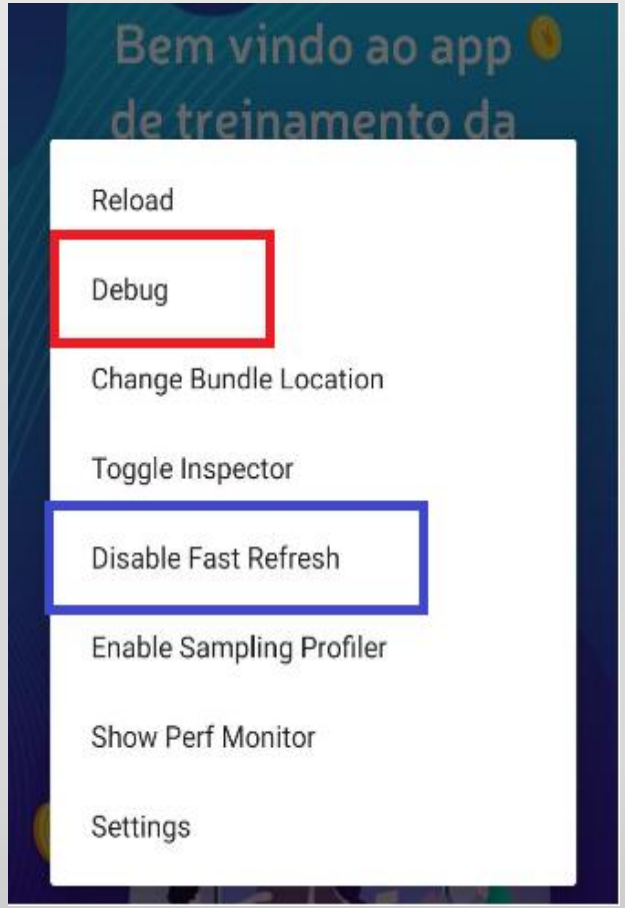
A documentação oficial do *React Native* (https://facebook.github.io/react-native/docs/getting-started) é um ótimo guia caso tenha alguma dúvida

- Criar um componente modular para customizá-lo através de *props*;
- Utilizar o *state* para armazenar valores e perceber a mudança em tempo real;
- Aprender os *lifecycles* do *React Native;*
- Navegar entre as telas utilizando *tabs*, *navigation* e passando dados;
- Utilizar a biblioteca "*axios*" para realizar chamadas em APIs;
- Fazer um ***CRUD*** com o banco de dados *MySQLite*;
- Utilizar as classes criadas internamente para realizar as operações de banco de dados;
- Utilizar o *Redux* para verificar o compartilhamento de dados entre diferentes telas da aplicação.

# *React Native* – Dicas

Depois de compilado e instalado, com o aplicativo aberto você pode "chacoalhar" o celular pros lados para abrir o menu de opções de desenvolvedor.

- Ative a opção de "*Debug*" que uma aba do *Google Chrome* será aberta, nela você pode pressionar "*F12*" e verificar no *console* todos os *logs* do código em *React Native* que possuem "*console.log()*";
- **Obs.**: certifique-se que a opção de "*Fast Refresh*" está ativada para ver as mudanças em tempo real.



# *React Native* – Dicas

Esse passo a passo serve como um guia para você realizar o treinamento, não é obrigatório seguir estritamente os passos, o importante é você absorver os conceitos e aprender a utilizá-los da melhor maneira possível. Sinta-se livre para finalizar o exercício da forma que preferir e pedir ajuda em dúvidas conceituais, estruturais e boas práticas.

# *React Native* – Exercício

1 – Vamos começar criando um componente simples de botão:

- Dicas:
  - Quando for necessário criar componentes genéricos que serão utilizados em qualquer lugar do *App* (ex.: botão), importe os componentes da própria biblioteca do “*react-native*”;
  - Para fazer os *layouts* de telas e se quiser componentes diferentes (ex.: lista, dropdown) utilize a biblioteca que quiser. Recomendamos o *NativeBase*:
    - https://docs.nativebase.io/Components.html#Components

# *React Native* – Exercício

- Na pasta "***\src\components***" crie um arquivo chamado "***TrainingButton.js***", esse componente deve ser capaz de alterar:
  - A cor de fundo do botão;
  - Sua largura e altura;
  - O texto dentro dele;
  - A função que será executada em seu "*click*";
  - Utilize uma variação do componente de botão do React Native:
    - *import { TouchableOpacity } from 'react-native'*

# *React Native* – Exercício

2 - Altere o arquivo "***Training01.js***" localizado na pasta "***\src\pages\training***"

- Instale a biblioteca "*axios*" (https://www.npmjs.com/package/axios) e faça uma requisição em seu servidor local criado no exercício de treinamento do Back-End;
- Utilize o *lifecycle* "*componentDidMount*" do *React Native* para fazer a requisição dos dados (https://pt-br.reactjs.org/docs/state-and-lifecycle.html);
- Os dados devem ser mostrados em uma lista com a disposição da figura ao lado.

| Nome | Salário | Aniversário |
|---|---|---|
| N'Golo Kanté<br>Status: Ativo | R$ 670.000,00 | 29/03/1991 |
| Ángel Romero<br>Status: Inativo | R$ 130.000,00 | 04/07/1992 |
| João Cardozo<br>Status: Ativo | R$ 950,00 | 26/08/1993 |
| Maria Silva<br>Status: Ativo | R$ 1.200,00 | 02/12/1985 |
| Ronaldo Fenômeno<br>Status: Ativo | R$ 950.000,00 | 18/09/1976 |
| Adriano Imperador<br>Status: Inativo | R$ 650.000,00 | 17/02/1982 |
| Ronaldinho Gaúcho<br>Status: Ativo | R$ 850.000,00 | 21/03/1980 |

# *React Native* – Exercício

3 – Crie o arquivo "***Training02.js***" na pasta "***\src\pages\training***"

- Essa nova tela deve ter os seguintes componentes:
  - 2 *inputs* de texto para o nome e data de nascimento;
  - 1 *input* de *dropdown* com as seguintes opções de salário: R$500,00, R$1000,00, R$1500,00, R$2000,00 e R$2500,00;
  - 1 *input* de *checkbox* para o *status* que salvará os valores: "Ativo" ou "Inativo";
  - 1 botão com a cor de fundo "#E6A000" e texto: "Construir Card";
  - 1 botão com a cor de fundo "blue" e texto: "Visualizar Detalhes";
  - 1 *card* que atualizará o texto com as informações do *input* após o *click* do botão.

# *React Native* – Exercício

- - Crie uma nova aba para acessar essa nova tela. Ver os arquivos:
  - *\src\navigation\BottomTabNavigation.js;*
  - *\src\navigation\StackNavigator.js.*
- Referências:
  - Referências para criação de *layout*, posicionamento e *grid* das telas:
    - https://facebook.github.io/react-native/docs/flexbox
  - Referências da biblioteca utilizada para fazer a navegação estre telas:
    - https://reactnavigation.org/docs/en/hello-react-navigation.html

# *React Native* – Exercício

## 4 – Utilizando o *state*

- Ao preencher todos os inputs da tela, o botão “Construir Card” deve armazenar esses valores em um objeto dentro do *state* e o *Card* deve mostrar essas informações;
- Toda vez que o botão for pressionado, o *Card* deve mostrar os valores atuais dos *inputs*;
- Todos os campos são obrigatórios, então se algum deles estiver em branco, deve ser mostrado uma mensagem de alerta para o usuário preencher o que está faltando;

# *React Native* – Exercício

- A data deve ser inserida nesse formato “YYYY-MM-DD” e no Card ela deve ser mostrada nesse formato: “DD/MM/YYY”;
- O botão “Visualizar Detalhes” deve abrir uma tela nova (na mesma aba) e essa tela nova deve conter um *Card* com todas as informações dos *inputs*, porém também deve conter uma imagem e um título.

# *React Native* – Exercício

## 5 – Salvar os valores dos inputs no banco de dados

- Veja a documentação da biblioteca de *React Native* para *"sqlite3"*:
  - https://www.npmjs.com/package/react-native-sqlite-storage
- Já existe um banco de dados *"sqlite3"* instalado no aplicativo:
  - **Nome do banco de dados**: *training.db;*
  - **Nome da tabela**: *rst_employees;*
  - **Colunas da tabela rst_employees**: *"id"*, *"name"*, *"status"*, *"birth"*, *"salary"* e as colunas padrão.

# *React Native* – Exercício

- Crie outra tela com o nome "***Training03.js***" e coloque em outra aba da aplicação;
- Construa novamente um formulário com os campos:
  - 2 *inputs* de texto para o nome e data de nascimento;
  - 1 *input* de *dropdown* com as seguintes opções de salário: R$500,00, R$1000,00, R$1500,00, R$2000,00 e R$2500,00;
  - 1 *input* de *checkbox* para o status que salvará os valores: "Ativo" ou "Inativo";
  - 1 botão para salvar essas informações no banco de dados;
  - 1 botão que abra uma nova tela na mesma aba e mostre uma lista de todos os dados salvos na tabela "*rst_employees".*

# *React Native* – Exercício

- Nessa nova tela que mostra a lista de todos os *employees*, ao clicar em um item da lista, outra tela deve ser aberta agora mostrando um formulário com os respectivos dados preenchidos e caso alguma informação seja alterada, isso deve ser salvo no banco de dados. (Ao clicar no botão de salvar ou cancelar o aplicativo deve voltar a tela anterior que possui a lista de *employees*).

# *React Native* – Exercício

## 6 – Utilizando as classes criadas pela Pulse para realizar as operações do banco de dados

- Adicione mais um botão no arquivo "***Training03.js***", ele irá abrir uma nova tela na mesma aba para mostrar novamente uma lista com todos os dados do banco "*rst_employees*".
- Para isso crie os arquivos de *Model* e *Controller*. Siga o exemplo da pasta "**\src\pages\database-example**";
- O arquivo de Model deve estender a classe *Model* e criar funções para as operações básicas do CRUD (*select*, *insert* e *update*, não é necessário fazer o *delete*). Aqui são definidos os parâmetros da *query* como: nome da tabela, colunas, condições, etc.;

# *React Native* – Exercício

- O *Controller* irá criar novas funções, com as funções definidas na *Model*, mais as regras de negócio que os botões da tela devem aplicar;
- No final desse exercício a tela deve estar exatamente igual ao exercício 5, com as mesmas funcionalidades, porém ao invés de utilizar as funções do *"sqlite3"*, as operações no banco de dados devem utilizar as funções criadas internamente e configuradas no seu *Model* e *Controller*.

# *React Native* – Exercício

## 7 – Utilizando o ***redux***

- Primeiro vamos entender um pouco da estrutura do *Redux* utilizada no projeto;
- Vá até a pasta "***\src\redux***", aqui ficam os arquivos de configuração do *redux* no projeto. (O único que você deverá modificar será o arquivo "***\src\redux\RootReducer.js***");
- A *store.js* faz as configurações básicas para o *redux* funcionar. Para mais detalhes:
    - https://medium.com/reactbrasil/iniciando-com-redux-c14ca7b7dcf
- O "***Reducer.js***" possui o *state* inicial e aqui são definidas as lógicas que serão aplicadas para alterar esse *state* dependendo da *action* que será disparada;

# *React Native* – Exercício

- A "***Actions.js***" define as funções que serão utilizadas pelas telas que querem alterar a *store* do *redux* e mandam qual o tipo da ação a ser disparada;
- O "***RootReducer.js***" junta todos os diferentes *Reducers* criados pela aplicação para que suas respectivas *stores* possam ser acessadas por qualquer tela da aplicação;
- O *redux* é utilizado para definir a tela inicial que será mostrada ao carregar o aplicativo, a tela "***Welcome.js***" só é mostrada na primeira execução do *app*, após isso o *app* sempre inicia nas abas mirando para a "***Home.js***";
- Você deve ter percebido que montamos o mesmo formulário nos arquivos "***Training02.js***" e "***Training03.js***". Nas situações que queremos compartilhar dados entre diferentes telas que não possuem a hierarquia de pai e filho, utilizamos o *redux*;

# *React Native* – Exercício

- O objetivo do exercício é fazer com que os dados que são digitados no arquivo "***Training02.js***" sejam refletidos no arquivo "***Training03.js***" e vice-versa. Consulte o guia se tiver dúvidas, não é necessário fazer as configurações pois o *app* já possui tudo pronto, foque na criação do *Reducer*, das *Actions* e na utilização do ***mapStateToProps*** e ***mapDispatchToProps*** (https://medium.com/reactbrasil/iniciando-com-redux-c14ca7b7dcf);
- Crie o arquivo "***TrainingReducer.js***" dentro da pasta "***\src\pages\training***". Esse arquivo terá a *state* inicial que armazenará os valores dos inputs e a lógica para armazenar esses valores;

# *React Native* – Exercício

- Crie o arquivo "***TrainingActions.js***" dentro da pasta "***\src\pages\training***". Esse arquivo irá exportar as funções que serão chamadas em outras telas para definir o tipo da ação disparada no *Reducer*;
- Utilize o ***mapStateToProps*** e ***mapDispatchToProps*** nos arquivos "***Training02.js***" e "***Training03.js***" para compartilhar os valores dos inputs que estão na *store* do *redux*. Veja os exemplos no arquivo "***\src\pages\home\Home.js***".

# *React Native*

FIM!