Treinamento NTec 20.2

Frontend PiuPiuwer - React Native

Responsáveis: Vinicius Rechuan
Augusto Iryoda
Gustavo Palma
Marcelo Kentaro

Summerados: Victor Santana
Daniel Jovchelevich Carvalho
Lívia Nishimura
Felipe Muralha
Felipe Rodrigues
Bruno Yoshioka
Rafael Menas
José Eduardo
Ygor Acacio
Luiz Felipe
Rossana Kallas
Victor Renzo

Introdução

O Piupiuwer é um sucesso! O projeto Web está sendo utilizado por todos os membros da Poli Júnior, e o melhor de tudo está começando a crescer e outras pessoas estão utilizando e o melhor de tudo está 100% seguro! Devido ao enorme sucesso foi decidido que precisamos expandir a solução para que possamos usá-lo sem precisar de um computador.

A missão para nossos incríveis trainees é fazer um aplicativo do Piupiuwer! E claro será feito em React Native para aproveitarmos o que já temos e colocar trazer ele a vida o mais rápido possível!

Desafio Proposto

Utilizando novamente como guia a prototipagem que vocês criaram no **Figma**, nesta etapa vocês deverão desenvolver, ao mínimo, uma **página de Login** e uma outra de **Feed.** Com as ferramentas do **React Native** e do **Typescript**, vocês deverão implementar, em ambas as páginas, as **mesmas funcionalidades básicas** que foram implementadas no último projeto, inicialmente apenas realizando as **mudanças necessárias** para uma adaptação para essa nova plataforma.

A ideia proposta é realmente **reutilizar grande parte do código** desenvolvido na etapa de **ReactJS**, apenas fazendo alterações de código **específicas da framework** desta etapa, assim como mudanças de **UI** e **UX**.

Visto isso, é importante ressaltar que, por conta do **tamanho de tela** de um **smartphone**, é aconselhável que vocês realizem certas mudanças para que os elementos na tela **não fiquem desorganizados**. Exemplos de possíveis boas mudanças seriam como, ao invés de permitir que **a caixa de criar um novo piu** permaneça na própria página de **Feed**, criar um botão que leve o usuário para uma **nova tela**, onde ele poderia **escrever o seu piu** com **espaço de sobra**, **sem confusão**.

Assim, ao completarem mudanças como as descritas acima, vocês poderiam seguir com a implementação das **funcionalidades opcionais** deste projeto. Essas envolveriam a criação de telas como a de **Cadastro**, ou até mesmo novas funcionalidades que seriam utilizadas pelos usuários **já logados** no sistema.

Assim como no **WhatsApp** ou **Instagram**, a navegação entre telas após o usuário ter logado no app é realizada por meio de **abas** na parte inferior da tela. Utilizando a biblioteca **BottomTabNavigation**, vocês poderiam criar, junto da página de **Feed**, algumas novas páginas, como a de **Perfil** e a de **Pesquisa de usuários**.

Com isso, aconselhamos que, em questão dessas funcionalidades opcionais, vocês apenas as implementem após terem finalizado as páginas obrigatórias de **Login** e **Feed**.

Com isso, seguem o **curso** que vocês deverão seguir e as **descrições das tasks**!

Curso da Rocketseat

https://www.notion.so/Next-Level-Week-e9fd0aaf9a5a466092987985d53aaf7e

Neste curso, vocês entrarão em contato com alguns **conceitos essenciais** do React Native, como:

- **Components específicos da framework**, que substituem as **tags de HTML** (não terá mais **div**, **p**, **a**, **input** etc.)
- Rotas do aplicativo utilizando **StackNavigator**
- Navegação por abas utilizando o **BottomTabNavigator**
- Utilização do **AsyncStorage** (similar ao LocalStorage, do ReactJS)

Porém, há algumas coisas que o Diego faz nesse curso que **não deveriam ser seguidas**:

- Uso do **StyleSheet**: o **StyleSheet** é biblioteca básica de estilização de Components do React Native. O próprio Diego menciona que o **StyledComponents** possui **mais funcionalidades** em relação ao **StyleSheet**, porém não o utiliza neste curso. De tal modo, como na etapa de **ReactJS**, o **StyledComponents deve ser utilizado** para estilizar seus components. A diferença é que, ao invés de utilizar **"styled.div"** ou **"styled.p"**, você deverá utilizar os nomes dos components do próprio React Native, transformando-os em **"styled.View"** e **"styled.Text"**, por exemplo. Além disso, a importação dele deve ser feita com "import styled from 'styled-components**/native**';". Caso estiver com dúvidas convido a ler a documentação do styled-components em react-native.
- Declarar **funções literais** dentro de um component: sempre se deve utilizar os **hooks básicos** do **React**. No caso de funções, declare-as dentro de um **useCallback**, para que a aplicação não tenha que declarar essa função **100x por minuto**
- **"then"**: sempre que possível, deve-se utilizar a estrutura de **async/await**. Com esta mudança, seu código torna-se mais legível e a lógica por trás mais intuitiva
- **AsyncStorage** fora do hook do **useAuth**: o AsyncStorage deve apenas ser utilizado como um método de armazenar uma **pequena quantidade** de informações com o intuito de serem **acessadas após o aplicativo ter sido fechado**. Utilizá-lo para acessar os dados do usuário logado durante o funcionamento da aplicação, por exemplo, causaria **problemas de rapidez** para o usuário. Como substituto, utilize o **ContextProvider** (como o **useAuth** — que

armazena dados e métodos de autenticação — e o **usePius** — que armazena dados e métodos de interação, criação e carregamento de pius)

Instalação do Expo

Para a instalação do Expo, entre no link a seguir (**Obs.:** Siga as instruções da sessão **Com Expo**, e não do link da Rocketseat).

https://www.notion.so/Ambiente-React-Native-f779604efae84e45a9cb13b44172455f

Tela de Login

- Apenas os usuários **registrados** no banco de dados podem se logar (Cada trainee com sua própria conta, ou com a hackeada do amiguinho)
- Caso algum dos inputs do usuário esteja **incorreto**, uma **mensagem** informando esse erro deve aparecer. A verificação de erros deve ser feita utilizando-se o **Yup** (clique aqui para ver a documentação do Notion)
- Um sistema de armazenamento de conta e login automático é desejável
- Para implementar a **autenticação**, vide a **documentação da API**, cujo link está abaixo

Tela de Feed

- Devem aparecer **todos os pius** do banco de dados
- Os pius devem ser mostrados em **ordem cronológica** (mais recentes no topo)
- O usuário deverá poder interagir com os pius de algumas formas. Implementem **pelo menos duas** das seguintes interações abaixo
    - Cada piu terá um botão "**favorito**" que, ao ser clicado, deve ser adicionado a uma lista na página de perfil do usuário
    - Cada piu terá um botão "**like**", que acrescenta 1 ao contador de likes
    - Os pius poderão ser **deletados** apenas se for do **usuário logado**
- Deve haver uma caixa para enviar **novos pius**
    - O piu só pode ser enviado se **não estiver vazio**
    - O piu só pode ser enviado se contiver **menos que 140 caracteres**
    - Deve haver um **contador** que atualiza conforme o usuário digita
    - Se o input do usuário chegar em **140 caracteres** tanto o contador quanto o campo de texto devem **mudar de estilo**

- Caso algum dos erros acima ocorra, deve-se exibir uma **mensagem de erro** ao usuário
- O piu criado pelo usuário deve ser **registrado** no banco de dados usado pela **API** (vide a **documentação da API** para saber mais)

Tela de Perfil (opcional)

- Devem ser dispostas todas as **informações pessoais** do usuário (nome completo, foto, sobre)
- Todos os **pius do usuário** devem ser mostrados da mesma maneira que no Feed
- Deve haver um botão de "**seguir**" o usuário, caso não seja o usuário logado

Tela de Pesquisa de usuários (opcional)

- Ela procura os **usuários** do Piupiuwer e dá acesso a seus perfis

Tela de Pesquisa de usuários (opcional)

- Esta tela deverá ter os **seguintes campos** para preenchimento:
  - **Username** (obrigatório)
  - **Nome** (obrigatório)
  - **Sobrenome** (obrigatório)
  - **Email** (obrigatório)
  - **Senha** (obrigatório)
  - **Sobre**
  - **Foto de perfil**
- A verificação de erros deve ser feita utilizando-se o **Yup** (clique aqui para ver a documentação do Notion)

Documentação da API

https://www.notion.so/pjntec/API-PiuPiuwer-ada2a5cc9dd2462e96c0e1d12f2d000e

OBS: Caso tenham dúvida de como usar a **API**, sintam-se livres para mandar mensagem no grupo dos Trainees, ou para o **Augusto Iryoda** ou **Gabriel Mascarenhas**.

Documentação de Aprendizados para Projetos (Notion)

https://www.notion.so/Aprendizados-para-Projetos-9454ba8daf9d4fcda21b99fb03ec53f8

OBS: Por mais que esteja no tópico de ReactJS, muitos dos conceitos e das bibliotecas do ReactJS também podem ser utilizadas dentro do React Native.

Documentação de Typescript (Notion)

https://www.notion.so/Typescript-c5e2d402bc5f4cb0ab9da7175a760e84

Monitores

Caso tenham dúvidas, vocês podem contar conosco sempre que precisarem! Tentaremos ajudar o máximo possível. Podem nos chamar no **grupo do Whats**, ou no **privado**, ou nas **monitorias no Discord** que faremos.

Passaremos mais pra frente os dias e horários das monitorias do **Discord**!

Entrega

O **prazo para a entrega** do projeto final é **quarta (21/10) às 23h59**. Vocês deverão deixar o repositório do classroom atualizado com a versão mais recente do seu código.

Boa sorte a todos, e **fritem muito**!
Atenciosamente,

Henrique Falconer
Líder do Treinamento 20.2
henrique.falconer@polijunior.com.br
Menção Honrosa

Vinicius Rechuan
Gerente do Núcleo de Tecnologia
vinicius.rechuan@polijunior.com.br
+55 21 99507-2844

Augusto Iryoda
Líder do Treinamento 21.1
augusto.iryoda@polijunior.com.br
+55 11 99654-2296

Gustavo Palma
Líder de Research 21.1
gustavo.palma@polijunior.com.br
+55 11 94162-6691

Marcelo Kentaro
Líder de Research 21.1
henrique.falconer@polijunior.com.br
+55 11 97450-4505

Victor Ken
Gerente do Núcleo de Tecnologia
victor.ken@polijunior.com.br