**Desafio :: Versão 2.0 do Happy**

Nessa versão nós vamos implementar novas funcionalidades para nossa aplicação que acabamos de desenvolver durante a Next Level Week 3.0 da [Rocketseat](https://rocketseat.com.br/), a **Happy.**

Esse vai ser um desafio para elevar ainda mais o seu nível como programador, onde você irá implementar novas funcionalidades para tornar a sua aplicação ainda melhor!

# **Layout da aplicação**

Durante o desenvolvimento de todos os desafios aqui propostos, você sempre terá algo para fazer no layout das suas aplicações.

Por exemplo, no desafio de autenticação de usuários você terá que criar a página de login nas versões Web e Mobile.

Então para saber como acessar o Layout dos desafios, veja na página abaixo:

[Layout dos desafios Happy](https://www.notion.so/Layout-dos-desafios-Happy-7a187b8bc861416aa5e539653af07bfd)

# **Acesso restrito**

A primeira nova funcionalidade da sua aplicação será a a implementação do Acesso Restrito, que será uma página de Dashboard que apenas os usuários cadastrados no seu banco de dados terão acesso. Para isso, nós primeiramente devemos criar, no nosso servidor, a funcionalidade para cadastrar usuários, inserindo seu nome, e-mail e senha.

Para cumprir esses requisitos, recomendamos criar uma tabela de nome users no banco de dados e uma rota para cadastro de usuários, e cadastrar um usuário padrão para ter acesso a aplicação.

**Dica:** Antes de salvar a senha do usuário no banco de dados, lembre-se sempre de criptografar a senha para manter ela segura. Para isso você pode utilizar bibliotecas como o [bcrypt](https://www.npmjs.com/package/bcrypt).

Depois de permitir a criação de uma senha, você precisa de uma rota para o login do usuário. Essa rota deve validar a senha do usuário no banco de dados e uma boa estratégia para manter o usuário logado por um tempo determinado é utilizar tokens JWT.

**Dica:** Nós temos um post no nosso blog sobre autenticação no Node.js utilizando bcrypt e JWT, deixamos no final dessa sessão em conteúdo extra.

O usuário deve poder logar em sua conta para acessar a Dashboard e poder cadastrar orfanatos ou aprovar cadastros de orfanatos.

**Dica:** Para manter o usuário logado na aplicação, caso ele clique na opção "Lembrar-me", você pode salvar o token JWT no LocalStorage do navegador (Web). Caso ele não selecione para se lembrar, você pode apenas mantê-lo autenticado durante uma sessão (até fechar a aplicação).

📖 **Conteúdo extra**:

* **Vídeo:** [Autenticação no React Native / ReactJS com Context API & Hooks | Masterclass #12](https://www.youtube.com/watch?v=KISMYYXSIX8)
* **Blog:** [Autenticação JWT no React Native com API REST em NodeJS](https://blog.rocketseat.com.br/autenticacao-react-native-nodejs/)
* **Blog:** [Fluxo de autenticação com Token JWT no React Native](https://blog.rocketseat.com.br/fluxo-de-autenticacao-com-react-native/)

# **Recuperação de senhas**

Nem sempre um usuário vai se lembrar da sua senha para logar na aplicação, principalmente se há muito tempo que ele acessou pela última vez, então uma funcionalidade essencial é a funcionalidade de recuperar a senha do usuário.

**Dica:** Nos conteúdos extras deixamos alguns conteúdos que você pode utilizar para entender como funcionaria o fluxo de recuperação de senhas no seu servidor.

📖 **Conteúdo extra:**

* **Vídeo:** [Recuperação de senha e envio de e-mail no Adonis | Behind the Code #02](https://www.youtube.com/watch?v=R7WTFLM1lto)
* **Vídeo:** [[API NodeJS + Express + Mongo] Recuperação de senha com NodeMailer | Diego Fernandes](https://www.youtube.com/watch?v=Zwdv9RllPqU)

# **Cadastro de orfanatos**

Agora, ao cadastrar um orfanato, ele não deve ser exibido em tempo real na aplicação para todos os usuários, todos os cadastros de orfanatos precisam ficar pendentes até a aprovação de usuário/administrador da aplicação.

Para isso, no seu banco de dados você pode adicionar um novo campo na tabela de orfanatos informando se o cadastro dela está pendente ou não. E no seu backend, na listagem de orfanatos, você deve retornar apenas os orfanatos que não estão com cadastro pendente.

# **Splash Screen no React Native com Expo**

Aqui você deverá configurar a splash screen do app mobile do Happy. Para isso, nós temos um conteúdo mais que especial sobre Deploy da aplicação mobile com expo, onde ensinamos a configurar a splash screen.

📖 **Conteúdo extra:**

* **Biblioteca:** [Gerando APK (Android) e IPA (iOS) com React Native & Expo | Code/Drops #15](https://www.youtube.com/watch?v=wYMvzbfBdYI)

# **Onboarding do usuário**

Nessa funcionalidade, você deve implementar o onboarding ao usuário abrir a aplicação para que tenha uma pequena apresentação da aplicação para o usuário.

Vale lembrar que essa apresentação só deve ser exibida na primeira vez que o usuário abrir a aplicação.

📖 **Conteúdo extra:**

* **Biblioteca:** [react-native-onboarding-swiper](https://github.com/jfilter/react-native-onboarding-swiper)

# **Localização real do usuário**

Em ambas nossas aplicações Web e Mobile, trabalhamos com a localização do usuário de forma fixa, utilizando uma latitude e longitude que informamos a mão. Agora o seu desafio é utilizar a localização real do usuário no mapa.

📖 **Conteúdo extra:**

* **Documentação:** [Utilizando localização com o expo-location](https://docs.expo.io/versions/latest/sdk/location/)
* **Vídeo:** [Utilizando a localização do usuário com o expo](https://www.facebook.com/rocketseat/posts/653992205147257)
* **Vídeo:** [Utilizando a localização do usuário pelo browser](https://www.facebook.com/watch/?v=649582815857758)

# **Cadastro em múltiplas etapas**

No cadastro de orfanatos do mobile, se você olhar no layout, perceberá que ele foi retrabalhado para funcionar em várias etapas. Seu trabalho é implementar essa funcionalidade.

Para isso, você apenas precisa guardar os dados enquanto muda de tela, e você pode escolher de várias opções para isso como por exemplo usar a ContextAPI ou mesmo passar os dados por parâmetros de navegação.

📖 **Conteúdo extra:**

* **Blog:** [Context API do React 16.3.0 e métodos de ciclo de vida](https://blog.rocketseat.com.br/context-api-react-16-ciclo-de-vida/)
* **Vídeo:** [Autenticação no React Native / ReactJS com Context API & Hooks | Masterclass #12](https://www.youtube.com/watch?v=KISMYYXSIX8)

# **Logout da aplicação**

Como última funcionalidade, mas não menos importante, você deve implementar uma função de logout na Dashboard da aplicação Web, para que o usuário possa trocar de conta caso ele deseje.

Aqui, você pode aproveitar o conteúdo sobre o fluxo de autenticação que deixamos na parte de autenticação para entender o funcionamento do Logout.

📖 **Conteúdo extra**:

* **Vídeo:** [Autenticação no React Native / ReactJS com Context API & Hooks | Masterclass #12](https://www.youtube.com/watch?v=KISMYYXSIX8)

# **Deploy da aplicação**

Agora é a hora de se desafiar a publicar a sua aplicação na WEB para todo mundo poder acessar! Nós criamos um guia **exclusivo** para essa edição da NLW ensinando do zero o processo de deploy da nossa aplicação, e o melhor, você ainda ainda vai aprender a configurar o PostgreSQL na sua aplicação!

📖 **Conteúdo extra:**

**Guia:** [**Deploy NLW**](https://www.notion.so/Deploy-NLW-ccd6afdf9e644673b9cb22c4aebaf39c)