ConTudo - Projeto de Aplicativo (App) para dispositivos “Android”

1. Instalar o Node.js:

Baixar no site <https://nodejs.org/pt-br/> e instalar.

2. Instalar o Expo (Plataforma para construção de apps nativos com React Native):

Abra o Terminal (Prompt de Comando) e execute o seguinte código: “npm install -g expo-cli” (sem as aspas)\*.

**\* Lembrar de sempre pressionar “Enter” no teclado para executar cada comando digitado.**

3. Instalar o Visual Studio Code:

Baixar o Visual Studio Code no site <https://code.visualstudio.com/download> e proceder a instalação.

4. Criar uma pasta (no Explorer do computador) onde os projetos serão salvos, voltar ao terminal e acessar a pasta onde os projetos serão salvos:

Exemplo:

No disco local “D:”;

Foi criada a pasta “Android”;

Dentro da pasta “Android” foi criada a pasta “Projetos”;

Então para acessar uma sub-pasta chamada “Projetos” que está na pasta “Android” que foi criada no disco “D:” usaremos os seguintes comandos no Terminal (sem o cifrão):

$ D:

$ cd Android

$ cd Projetos

O Terminal deve apresentar algo assim:

$ D:Android/Projetos>\_

5. Criar o projeto dentro da pasta selecionada, através do Terminal, utilizando o seguinte comando (sem o cifrão):

$ expo init “<nome\_do\_projeto>” (digitar o nome do projeto sem as aspas)

Deve ficar algo como abaixo:

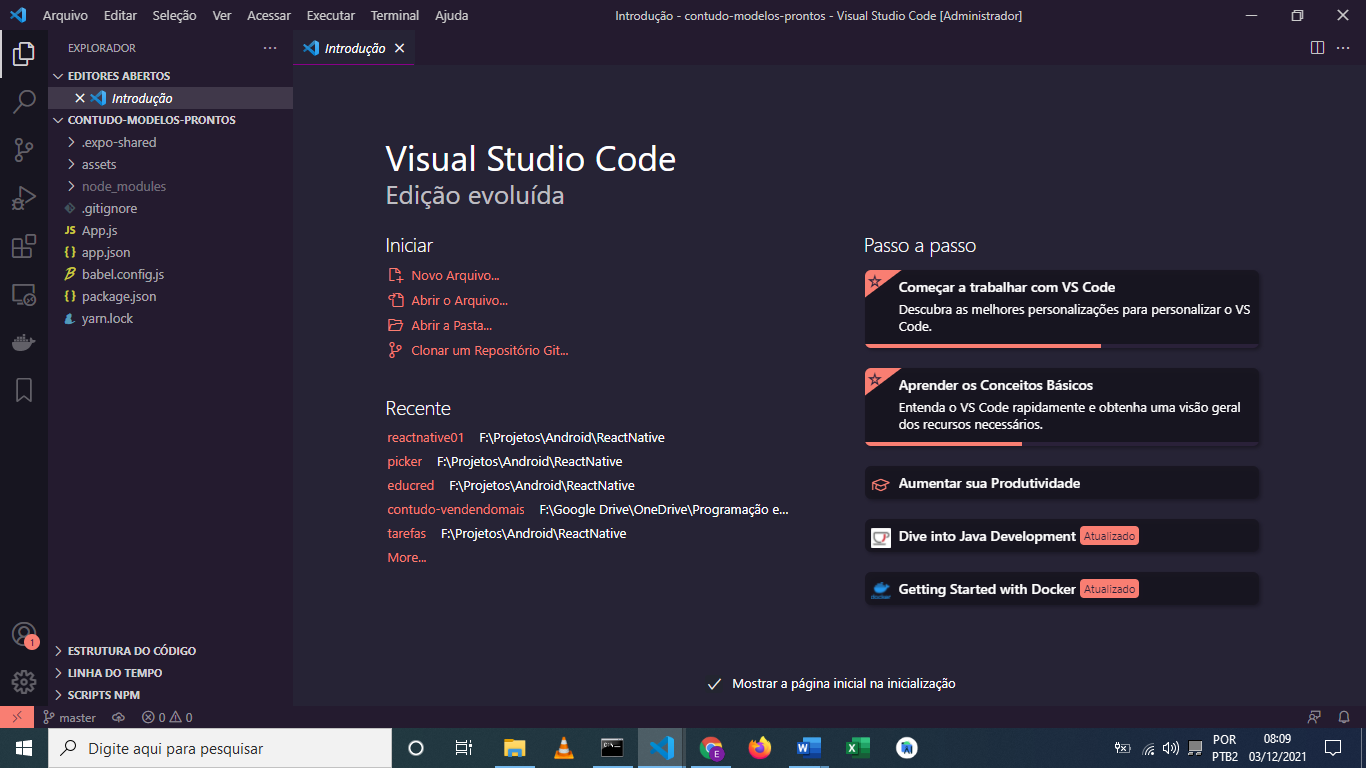
D:Android/Projetos>expo init primeiro\_projeto

O processo de criação do projeto deve levar um tempo, aguarde até o término.

6. Abrir o Visual Studio Code:

Abrir a pasta do projeto criado;

A tela deve parecer com o exemplo abaixo:



7. Dependências: O que são? O que fazem?

P.S.: Sempre verificar a versão da dependência, para instalar a versão correspondente ao ano em que o recurso que vai utilizar foi criado.

Dependências são todos os pacotes que seu projeto necessita ter instalado para que possa ser executado de maneira correta.

Em outras palavras, todos os módulos (pacotes) de terceiros que utilizamos em nosso projeto, se tornam dependências, já que sem eles, o funcionamento da nossa aplicação estaria comprometida.

Exemplo: Quando precisamos “navegar” entre telas do aplicativo, precisamos de uma dependência que crie essa funcionalidade. Então essas dependências têm que ser instaladas através de um Prompt de Comando.

Abaixo, seguem, as primeiras dependências que serão instaladas, através do Terminal (Prompt de Comando), acesse a pasta do projeto e digite os comandos, pressionando “Enter”, após cada comando.

À medida que formos implementando recursos ao projeto, novas dependências serão instaladas.

Se você precisar instalar uma versão mais antiga de um pacote, especifique-a

npm install <package>@<version>

Por exemplo: npm install express@3.0.0

Você também pode adicionar o --save sinalizador a esse comando para adicioná-lo às dependências do package.json ou --save --save-exact sinalizadores, se desejar a versão exata especificada nas dependências do package.json.

**Dependências de Tela de Embarque (OnBoarding):**

|  |  |  |
| --- | --- | --- |
|  | npm install --save react-native-onboarding-swiper |  |

**Dependências de Trilhas (stacks) e Navegação entre Telas:**

|  |  |
| --- | --- |
| npm install --save @react-navigation/native | Instala uma série de dependências-chave para que tenhamos os comportamentos nativos de navegação/transição de telas |
| npm install --save @react-navigation/stack | Instala uma lib para definir como a navegação será gerenciada pelo React Native |
| expo install react-native-gesture-handler react-native-reanimated react-native-screens react-native-safe-area-context | Instalar uma série de dependências necessárias pelo Expo |
| expo install @react-native-community/masked-view | Instalar mais uma dependência necessárias pelo Expo |

**Dependências do banco de dados (Firebase):**

|  |  |  |
| --- | --- | --- |
|  | expo install firebase |  |
|  | npm install --save react-native-firebase | Desinstalar |
|  | npm install --save @react-native-firebase/app | Desinstalar |
|  | npm install --save @react-native-firebase/auth | Desinstalar |

[Passos a serem realizados na conta do Firebase.](#Firebase)

Acessar o site do Firebase: “<https://console.firebase.google.com/>”;

Fazer login com uma conta Google;

Clicar em “Adicionar Projeto”;

Digite um nome para o projeto e clique em “Continuar”;

Desmarque a opção: “Habilitar Google Analytics para este projeto” e clique em “Criar Projeto”;

Aguarde o projeto ser criado e clique em “Continuar”;

No menu lateral clique em “Authentication” e depois em “Primeiros Passos”;

Clique em “Email/Password”, depois clique na primeira opção “Ativar” para habilitar esta forma de autenticação e depois “Salvar”;

No menu lateral clique no ícone de “catraca” ao lado do nome “Visão geral do projeto” e depois clique em “Configurações do projeto”;

Role até o fim da página e “Selecione uma plataforma para começar” clicando no ícone “</>”;

Digite um nome para o App (pode ser o mesmo do projeto) e clique em “Registrar”;

Na tela que aparecer role para baixo e clique em “Continue no console”;

Aparecerá uma nova tela, role para baixo e veja o código de configuração do banco de dados do App.

Volte para a pasta do seu projeto e crie um arquivo de configuração do Firebase como abaixo, substituindo as informações pelas que estão no App do seu Firebase:

|  |
| --- |
| import firebase from "firebase";  import 'firebase/storage';  const firebaseConfig = {  apiKey: "AIzaSyDlck5Cvx4QXAaCKQwhs\_AmrRO0WDTCdpk",  authDomain: "nome-do-seu-app-f65e1.firebaseapp.com",  projectId: " nome-do-seu-app-f65e1",  storageBucket: " nome-do-seu-app-f65e1.appspot.com",  messagingSenderId: "11113432203",  appId: "1:11113432203:web:b7778f6b9447dfc12f008e"  };    // Initialize Firebase  if(firebase.apps.length === 0){  firebase.initializeApp(firebaseConfig);  }  export default firebase; |

Login no App com Conta Google:

**Dependências de login com a conta Google:**

|  |  |  |
| --- | --- | --- |
|  | expo install expo-google-app-auth |  |

Na página do Firebase, em “Authetication”, Escolha fazer login com uma conta Google, informe o nome do seu App e o email de suporte

Acesse o site: “<https://docs.expo.dev/versions/latest/sdk/google/>”;

Role para baixo, até encontrar “Using it inside of the Expo app”, e clique no link: “[Credentials Page](https://console.developers.google.com/apis/credentials)”;

Na página de “APIs e Serviços”, clique na “setinha para baixo” que fica logo após a frase “Google Cloud Platform”;

Na janela “Selecione um projeto”; clique na aba “TODOS” e selecione o App que deseja trabalhar;

Clique em “ + CRIAR CREDENCIAIS” e depois em “ID do Cliente OAUTH”;

Clique em “CONFIGURAR TELA DE CONSENTIMENTO”; na próxima tela marque “Externo”, digite as informações do seu App e clique em “SALVAR E CONTINUAR”;

Feche a janela “Verificação Obrigatória” clicando em “OK”;

Na tela “Escopos” clique em “SALVAR E CONTINUAR”;

Na tela “Informações opcionais” clique em “SALVAR E CONTINUAR”;

Na tela “Resumo” clique em “VOLTAR PARA O PAINEL”;

Na tela “Credenciais”, clique novamente em “ + CRIAR CREDENCIAIS” e depois em “ID do Cliente OAUTH”;

Escolha “Tipo de aplicativo”: “Android”, “Nome”: defina o nome do cliente (Exemplo: Cliente nome\_do\_aplicativo);

Para obter a “Impressão Digital para Certificação SHA-1” siga os passos:

Baixe a versão do “openssl” compatível com seu sistema operacional e descompacte em uma pasta de sua preferência. Links: [OpenSSL para 64 bits](http://code.google.com/p/openssl-for-windows/downloads/detail?name=openssl-0.9.8k_X64.zip) e [OpenSSL para 32 bits](https://code.google.com/p/openssl-for-windows/downloads/detail?name=openssl-0.9.8k_WIN32.zip);

No terminal acesse a pasta “bin”, dentro da pasta onde descompactou o “openssl” e digite o seguinte comando: “openssl rand -base64 32 | openssl sha1 -c” (sem as aspas).

Copie e cole o código SHA-1 gerado no campo “Impressão Digital para Certificação SHA-1” e clique em “CRIAR”.

Na tela “Credenciais”, clique no ícone “Copiar cliente OAuth”, para copiar o “ID do cliente”;

Volte para a edição do seu Projeto e no trecho de código onde aparecer “androidClienteId”, cole, entre aspas simples, o “ID do Cliente” que foi copiado, após os dois pontos.

Lembrar de alterar as Regras do Firestore Database e do Storage.

Descrição das opções de “Criação” do Firebase:

|  |  |  |
| --- | --- | --- |
|  | Authetication |  |
|  | Firebase Database | Cloud Firestore  Atualizações em tempo real, consultas eficientes e escalonamento automático |
|  | Realtime Database | Armazenamento e sincronização de dados em tempo real |
|  | Storage | Armazenar e recuperar arquivos gerados pelo usuário, como imagens, áudio e vídeos, sem o código do servidor |
|  | Hosting | Implementar apps da Web e apps da Web para dispositivos móveis em segundos usando uma rede de fornecimento de conteúdo global segura |
|  | Functions | Execute seu código de back-end para dispositivos móveis sem gerenciar servidores. |
|  | Machine Learning | Resolva problemas comuns nos seus apps com o aprendizado de máquina |
|  |  |  |

**Dependência para processamento de imagem Node.js de alto desempenho (sharp-cli):**

|  |  |  |
| --- | --- | --- |
|  | npm install sharp-cli | CLI para sharp , um módulo de processamento de imagem Node.js de alto desempenho. |

**Para rodar projetos Expo no Browser:**

Todos os projetos após o SDK 32 vêm com suporte web por padrão. Para adicionar suporte web a um aplicativo Expo existente, você pode fazer o seguinte:

**Dependências da web:**

|  |  |  |
| --- | --- | --- |
|  | expo install react-native-web |  |
|  | expo install react-dom |  |

Certifique-se de que seu projeto tenha pelo menos expo@^33.0.0instalado.

Inicie seu projeto com expo start, pressione w, para iniciar o Webpack e abra o projeto no navegador.

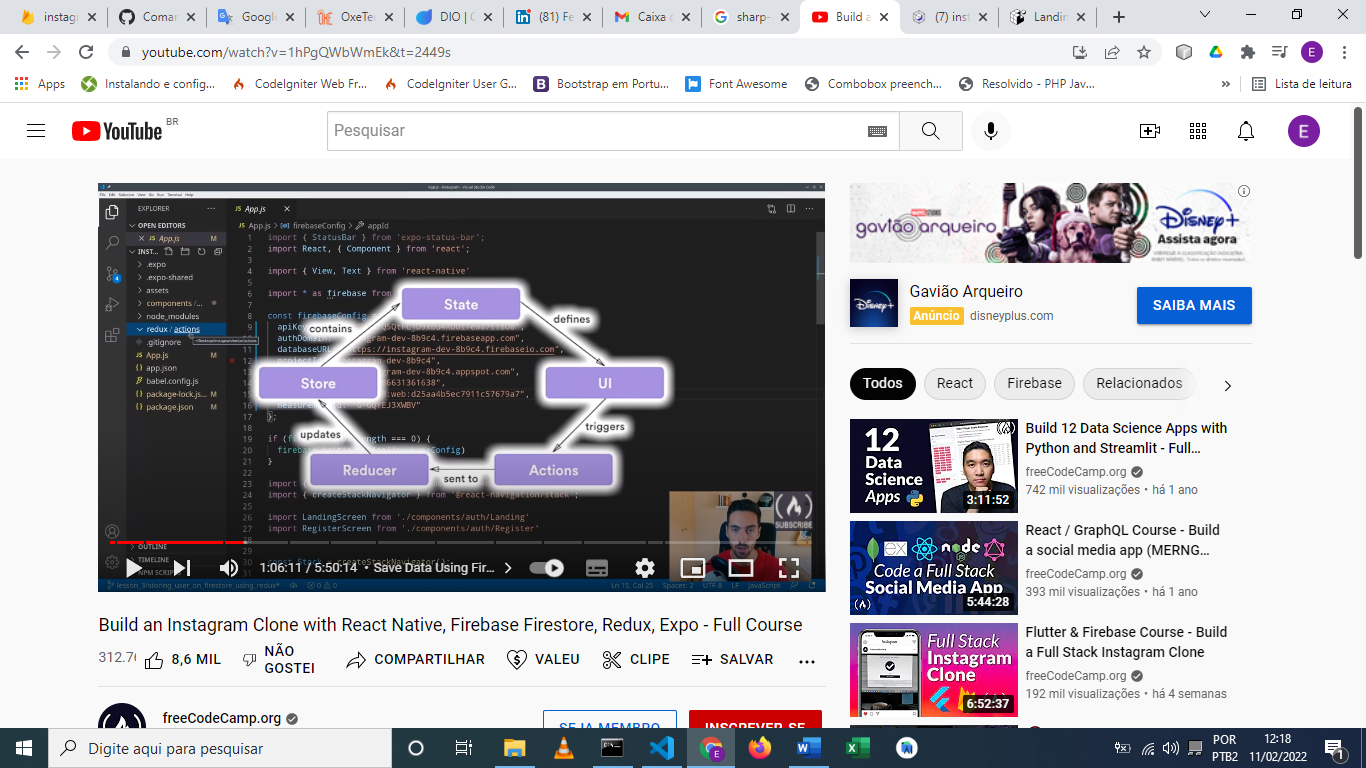
Pontas

Inicialização mais rápida no modo somente web executandoexpo web

Teste APIs protegidas, como Câmera e Compartilhamento, usando os sinalizadores --httpsou .--no-https

**Dependências do Redux: (REDUX FOI SUBSTITUÍDO POR USESTATE**

|  |  |  |
| --- | --- | --- |
|  | npm install redux |  |
|  | npm install react-redux |  |
|  | npm install redux-thunk |  |



**Dependências de Bottom Tabs e Icones:**

|  |  |  |
| --- | --- | --- |
|  | npm install @react-navigation/bottom-tabs --save | Bottom tabs – menu inferior |
|  | npm install --save react-native-vector-icons |  |
|  | npm install --save @react-navigation/material-bottom-tabs |  |
|  | npm install --save react-native-paper |  |

**Dependências de Camera e Image Picker:**

|  |  |  |
| --- | --- | --- |
|  | expo install expo-camera |  |
|  | expo install expo-image-picker |  |

Com o projeto criado e primeiras dependências instaladas, vamos começar a atualizar os arquivos existentes e criar novos arquivos com suas respectivas funcionalidades. Abaixo serão mostradas imagens de como as telas devem se parecer.

Arquivos do projeto:

|  |  |  |
| --- | --- | --- |
| **Arquivo** | **Atribuição** | **Pasta(s)/Caminho** |
| App.js | O primeiro a ser “chamado” ao abrir o aplicativo:  Importa as dependências necessárias;  Importa a constante (const) “Providers” do arquivo “index.js” da pasta “./navigation”, depois define a constante “App” e suas atribuições, exporta a constante “App” no fim do arquivo. | / |

Navegação entre telas do aplicativo:

|  |  |  |
| --- | --- | --- |
| index.js | Importa as dependências necessárias;  Importa as contantes (const) “AuthProvider” e “Routes” de seus respectivos arquivos, depois define a constante “Providers” e suas atribuições, exporta a constante “Providers” no fim do arquivo. | /navegacao/ |
| AuthProvider.js | Importa as dependências necessárias;  Trata as ações de **AUTENTICAÇÃO** entre o Aplicativo e o Firebase, tais como:  Define e exporta uma constante (const) com contexto de autenticação “AuthContext”;  Define e exporta a constante (const) “AuthProvider”;  Define uma constante (const) “user”;  Trata o Login, Cadastro e Logout do usuário entre o Aplicativo e o Firebase (Authentication). | /navegacao/ |
| Routes.js | Importa as dependências necessárias;  Importa a constante (const) “AuthContext” do arquivo “AuthProvider” e as constantes (const) “AppStack” e “AuthStack” de seus respectivos arquivos.  Define as constantes “user”, “initializing” e “onAuthStateChange”;  No “useEffect” define a constante (const) “subscribe” e verifica o estado de autenticação do usuário (auth.onAuthStateChange);  Verifica a constante “initializing” usando a cláusula “if”;  Exporta a constante “Routes” no fim do arquivo. | /navegacao/ |
| AuthStack.js | Importa as dependências necessárias;  Importa o arquivo da tela de embarque “TelaEmbarque”, a tela inicial “TelaInicial” e os arquivos das telas de login e cadastro de usuários, “LoginTutor”, “CadastrarTutor”, “LoginJogador” e “CadastrarJogador”;  Define a constante (const) “Stack” (Pilha) igual a “createStackNavigator()”;  Define a constante (const) “AuthStack”;  Define e “seta” a constante (const) “isFirstLaunch”;  Define a variável (let) “routeName”;  Usa “AsyncStorage” para verificar se o aplicativo já foi iniciado pela primeira vez “alreadyLaunch” atribuindo um “value”, se nunca foi iniciado (value == null) “seta” “setIsFirstLaunch” como “true”;  Verifica o estado de “isFirstLaunch”, se “true” define “routeName” para a tela de embarque “TelaEmbarque”, senão define “routeName” para a tela de login “TelaInicial”;  Define a “Stack.Navigator” para navegação entre telas;  Define as “Stack.Screen” das telas acessíveis antes do usuário estar logado.  Exporta a constante “AuthStack” no fim do arquivo. | /navegacao/ |
| AppStack.js | Importa as dependências necessárias;  Importa o arquivo da tela inicial “TelaInicial” e demais páginas disponíveis ao usuário após realização do Login;  Define a constante (const) “Stack” (Pilha) igual a “createStackNavigator()”;  Define a constante (const) “AppStack”;  Define a “Stack.Navigator” para navegação entre telas;  Define as “Stack.Screen” das telas acessíveis após o usuário estar logado.  Exporta a constante “AppStack” no fim do arquivo. | /navegacao/ |

Tela de Embarque:

|  |  |  |
| --- | --- | --- |
| TelaEmbarque.js | Importa as dependências necessárias;  Define a constante (const) “TelaEmbarque”;  Define as características da tela de embarque.  Exporta a constante “AuthStack” no fim do arquivo. | /pages/TelaEmbarque/ |

|  |  |  |
| --- | --- | --- |
| Nº | Descrição | Comando |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | Instala a dependência de uma lista de seleção, componente que não existe nativo no React Native | npm install --save react-native-picker-select |
|  |  | npm install --save http |
|  |  | npm install --save express |
|  |  | npm install --save cors |
|  |  | npm install --save body-parser |
|  |  | npm install --save axios |

|  |  |  |
| --- | --- | --- |
|  | npm install --save @expo/vector-icons |  |
|  |  |  |
|  | npm install --save @material-ui/icons |  |
|  | npm install --save react-native-elements |  |
|  | npm install --save styled-components |  |
|  | npm install --save react-native-action-button |  |
|  | npm install --save react-native-image-crop-picker --save |  |
|  | npm install --save @react-native-masked-view/masked-view react-native-linear-gradient |  |
|  | npm install react-native-skeleton-placeholder --save |  |
|  | npm install --save react-native-gifted-chat |  |
|  | npm install --save @react-native-async-storage/async-storage |  |
|  | npm install react-native-hide-show-password-input --save |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Para usar o Login Google em um projeto Expo:

|  |  |  |
| --- | --- | --- |
|  | npm install expo-google-app-auth |  |
|  | npm install --save @react-native-community/google-signin |  |

Execute o comando abaixo após instalar e configurar o Firebase:

|  |  |  |
| --- | --- | --- |
|  | npx react-native run-android |  |
|  | npm install --save react-native-unimodules |  |

Redux

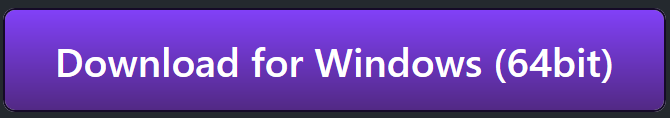
|  |  |  |
| --- | --- | --- |
|  | npm install redux --save |  |
|  | npm install react-redux --save |  |
|  | npm install redux-thunk --save |  |

Passos a serem realizados na conta do Firebase.

1-GitHub:

2-Passos para Instalar:

1. Acesse a [página de download do GitHub Desktop](https://desktop.github.com/).
2. Clique em **Download para Windows**.



1. Na pasta Downloads do seu computador, clique duas vezes no arquivo de configuração **GitHub Desktop**.Arquivo GitHubDesktopSetup
2. GitHub Desktop será lançado após a instalação ser concluída.

3-[Criando e enviando arquivos para seu repositório no GitHub](http://gabsferreira.com/criando-e-enviando-arquivos-para-seu-repositorio-no-github/)

Antes de mais nada: se tem dúvida sobre as diferenças entre Git e Github, dá uma conferida [nesse vídeo](https://www.youtube.com/watch?v=P4BNi_yPehc?source=gabsferreira.com).

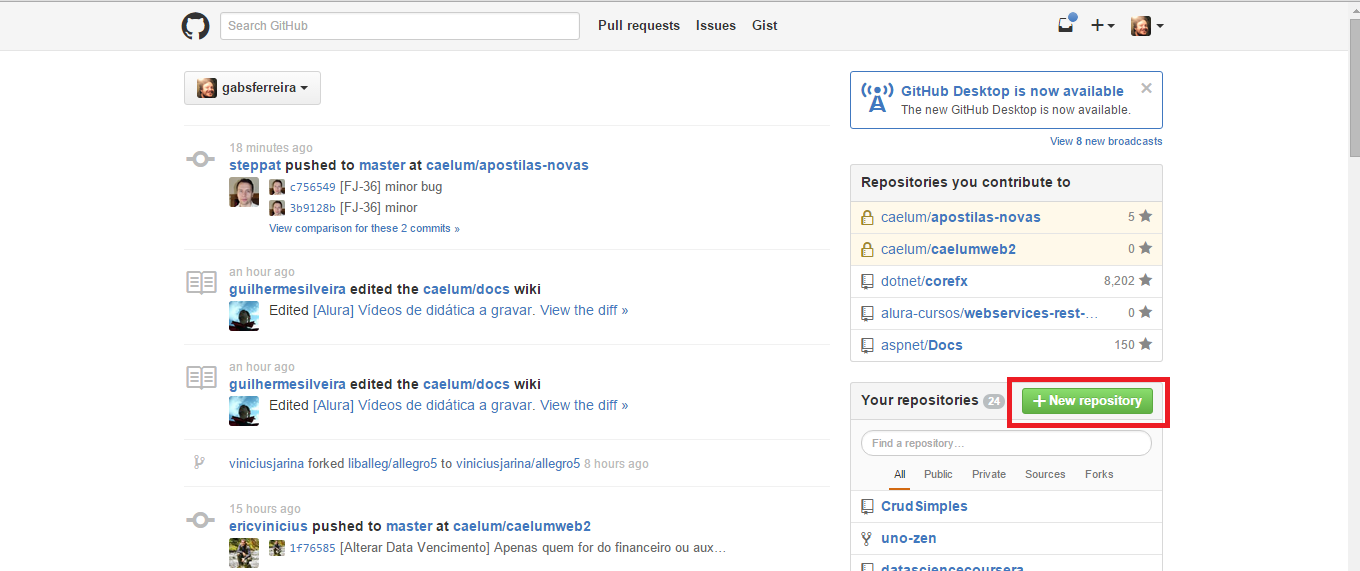
No [primeiro](http://gabsferreira.com/instalando-o-git-e-configurando-github/) post sobre Git e Github, ensinei a fazer a parte de configuração e instalação.  
No [segundo](http://gabsferreira.com/forkando-e-clonando-um-repositorio-no-github/), mostrei como fazer o fork de um repositório existente e baixar ele pra sua máquina.

Agora, chegou a hora de aprender a **criar** seu próprio repositório e interagir com ele.

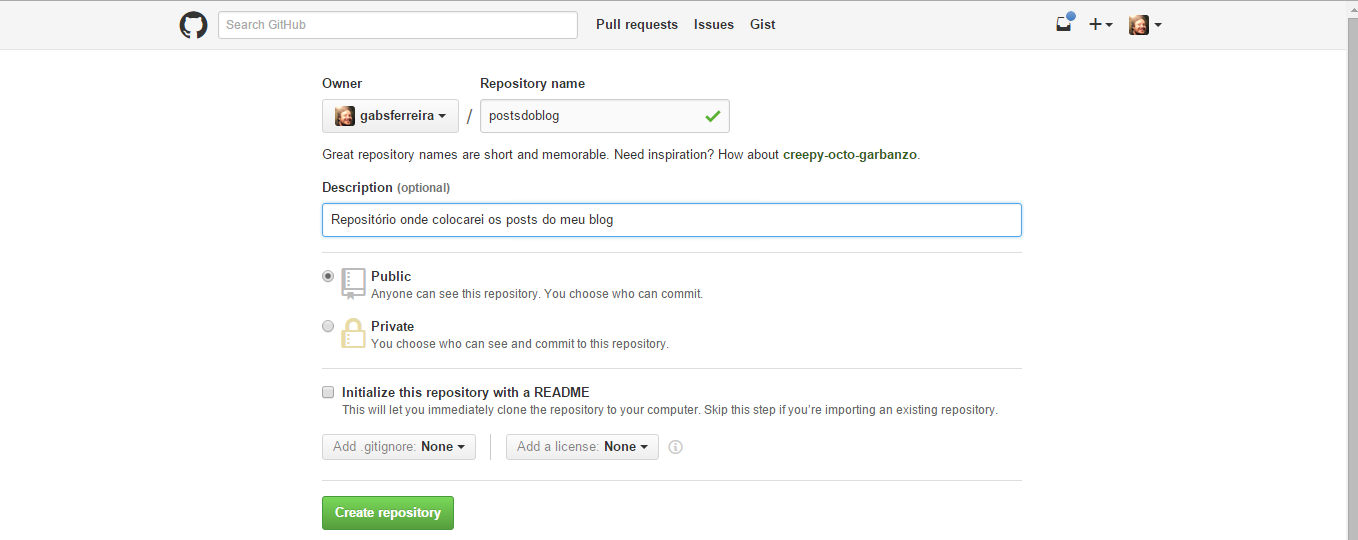
Geralmente as pessoas criam repositórios no Github para hospedar seus projetos pessoais. E esses projetos não precisam ser necessariamente código.  
Como exemplo nesse post, eu vou criar um repositório para guardar os posts do meu blog. Os comandos que vou usar são **os mesmos** pra qualquer tipo de projeto/arquivo.

# **Criando o repositório**

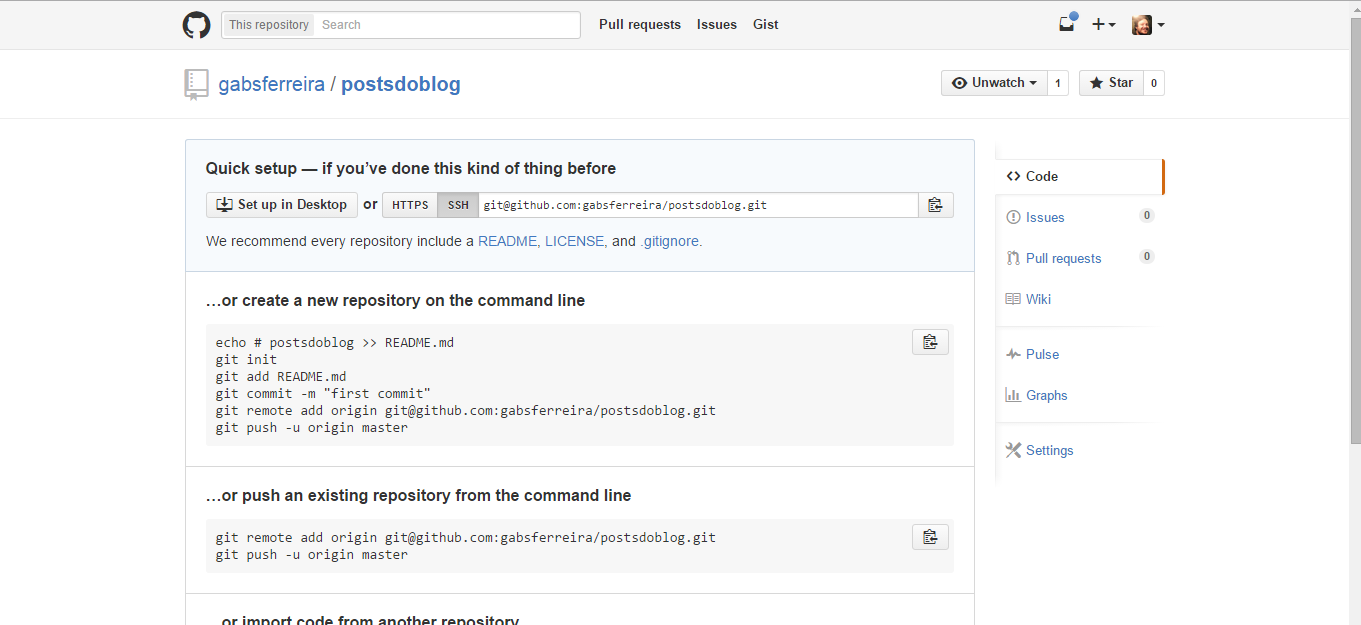
Abra o [Github](https://github.com/), faça login com a sua conta e clique em New repository:



Em seguida, basta colocar o nome e descrição do repositório que você está criando e clicar em Create repository:



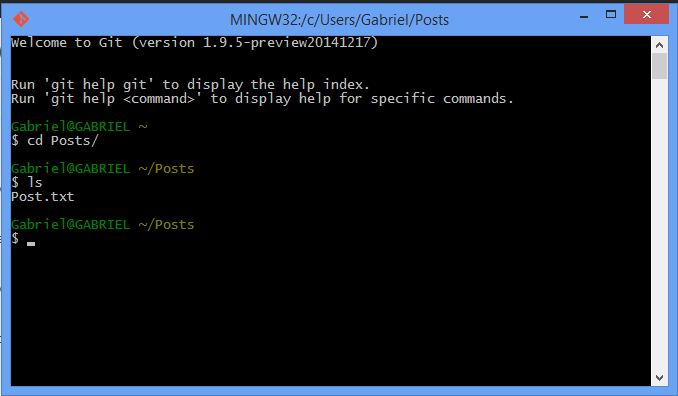
Depois, aparecerá essa tela:



O repositório foi criado, só que ainda está vazio.  
Vamos enviar nossos arquivos para lá!

# **Inicializando um repositório na sua máquina**

Abra o Git Bash e vá até a pasta onde está o seu projeto.  
Eu no caso vou até a minha pasta Posts, onde tenho um post meu salvo no formato txt:

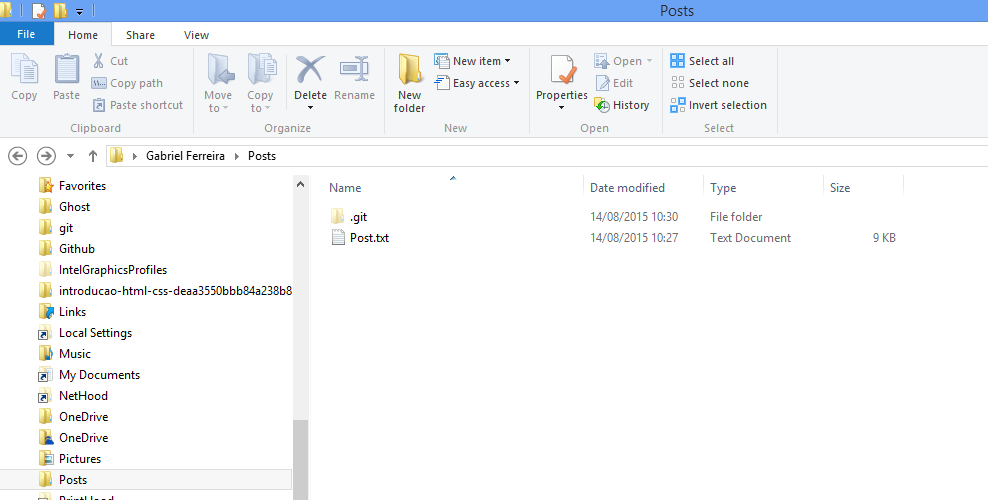


Agora, quero transformar essa minha pasta em um **repositório Git**.  
Pra isso, basta digitar **git init** e dar enter:

$ git init

Initialized empty Git repository in c:/Users/Gabriel/Posts/.git/

Ele está falando que iniciou um repositório vazio nessa pasta. Como eu sei que agora tenho um repositório nessa pasta?  
Se você abrir a pasta no File Explorer e configurar ele para mostrar os arquivos ocultos, verá que tem uma pasta a mais chamada **.git**:



O Git usa essa pasta pra controlar as alterações feitas no seu repositório.

Ok, agora temos um repositório na sua máquina e um no Github...como **sincronizar** os dois?

# **Adicionando uma origin**

Vamos agora executar esse comando:

git remote add origin https://github.com/seu\_nome\_no\_github/nome\_do\_repositorio.git

Basicamente o que estamos dizendo nessa linha de comando é:  
"Git, esse meu repositório **local** se conectará com um **remoto**, o caminho dele(**origin**) é git@github.com:gabsferreira/postsdoblog.git. Estabeleça essa comunicação pra mim".

Se você abrir aquela pasta **.git** e abrir o arquivo chamado **config** lá dentro, verá a seguinte linha:

[remote "origin"]

url = git@github.com:gabsferreira/postsdoblog.git

fetch = +refs/heads/\*:refs/remotes/origin/\*

É onde está configurada nossa **origin**.

# **Enviando alterações para o Github**

Quando estou dentro de uma pasta que é um repositório Git e quero saber o que eu fiz de alterações, eu digito o seguinte comando:

git status

Esse comando é um dos que você **mais** vai usar no Git.  
Ele irá mostrar:

On branch master

Initial commit

Untracked files:

(use "git add <file>..." to include in what will be committed)

Post.txt

nothing added to commit but untracked files present (use "git add" to track)

Ele está dizendo aí que o arquivo Post.txt está untracked, ou seja, não está sendo **monitorado/gerenciado** pelo Git. Quero que ele seja!  
Então:

git add Post.txt

Com esse comando, eu **adicionei** o meu arquivo ao Git. Se eu der git status de novo:

$ git status

On branch master

Initial commit

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: Post.txt

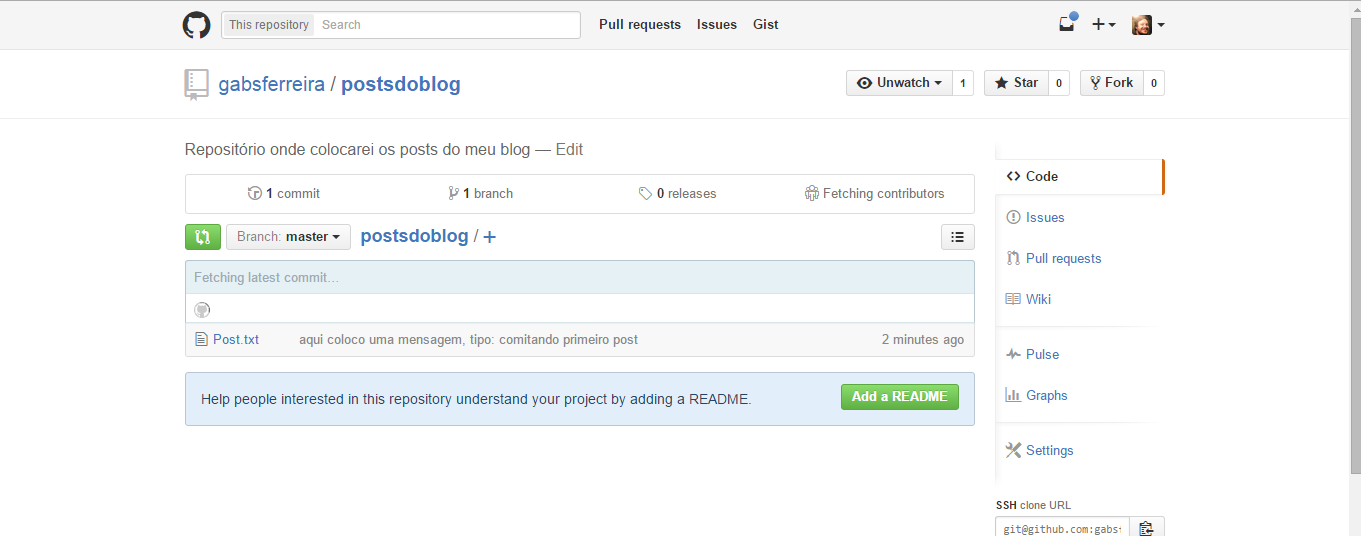
Agora ele me mostra que o arquivo está para **ser comitado**. Ou seja, ele ainda não guardou o estado do arquivo **permanentemente**, é só temporário.  
Se eu quiser que seja permantente, uso o git commit:

git commit -m "aqui coloco uma mensagem, tipo: comitando primeiro post"

Beleza, comitado.  
Agora quero enviar isso para o meu repositório remoto:

git push -u origin master

Se você for até o seu repositório no Github agora, voilá:



## React Native + FireBase(CRUD Completo) – Tasks

**Passo 1**

Você precisará de um conta Gmail para poder acessar o firebase e criar seu primeiro projeto.

**Passo 2**

Criar um banco no Firebase. Ao criar o banco devemos acessar: FireStore Database/Coud FireStore/Regras e em editar regras configurar nosso allow como true.

Default



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | rules\_version = '2';  service cloud.firestore {    match /databases/{database}/documents {      match /{document=\*\*} {        allow read, write: if true;      }    }  } |

Depois vamos em dados para criar um coleção. Vamos chamar essa coleção de Tasks e adicionar um objeto em nossa collection Tasks.

Default



|  |  |
| --- | --- |
| 1  2  3  4 | {  description: "estudar javascript", (sting)  status: True (boolean)  } |

Em visão geral do projeto podemos acessar nossas credenciais para acesso ao banco, vamos selecionar WEB e copiar nossas credenciais.

### Criando nosso projeto

Default



|  |  |
| --- | --- |
| 1  2 | expo init tasks  cd tasks |

Vamos fazer algumas instalações que iremos utilizar durante o desenvolvimento do nosso projeto.  
Vamos utilizar react navigation dentro do nosso app, então rode em seu terminal dentro da pasta do projeto:

yarn add @react-navigation/native

yarn add @react-navigation/stack

expo install react-native-gesture-handler react-native-reanimated react-native-screens react-native-safe-area-context @react-native-community/masked-view

yarn add firebase

Com tudo configurado vamos agora iniciar a configuração dos arquivos dentro do nosso projeto.  
Para iniciar nosso projeto vamos montar nossa estrutura de pastas:

src

config

firebase.js

pages

Task

index.js

style.js

NewTask

index.js

style.js

DeTails

index.js

style.js

App.js

Configurando nosso firebase.js dentro de src/config/

import firebase from "firebase"

import 'firebase/storage';

var firebaseConfig = {

apiKey: "8asd48as1das1dasd8asd1a9s8d1d81as",

authDomain: "tasks-1s6a5d16a.firebaseapp.com",

databaseURL: "https://tasks-1s6a5d16a-default-rtdb.com.firebaseio.com",

projectId: "tasks-1s6a5d16a",

storageBucket: "tasks-1s6a5d16a.appspot.com",

messagingSenderId: "15561561651",

appId: "1:15561561651:web:1d5a6s1d5a61d65s"

};

// Initialize Firebase

firebase.initializeApp(firebaseConfig);

const database = firebase.firestore();

export default database

**Configurando as Rotas**

Agora, vamos trabalhar dentro do nosso arquivo principal.  
No App.js, abra seu arquivo, vamos editá-lo:

import React from "react";

import { NavigationContainer } from "@react-navigation/native";

import { createStackNavigator } from "@react-navigation/stack";

import Task from "./src/pages/Task/";

import NewTask from "./src/pages/NewTask/";

import Details from "./src/pages/Details/";

const Stack = createStackNavigator();

export default function App() {

return (

<NavigationContainer>

<Stack.Navigator initialRouteName="Task">

<Stack.Screen

name="Task"

component={Task}

options={{

headerTintColor: "#F92E6A",

}}

/>

<Stack.Screen

name="New Task"

component={NewTask}

options={{

headerTintColor: "#F92E6A",

}}

/>

<Stack.Screen

name="Details"

component={Details}

options={{

headerTintColor: "#F92E6A",

}}

/>

</Stack.Navigator>

</NavigationContainer>

);

}

## **Page Task**

Construindo o index.js da Page Task:

1 – Vamos realizar os imports necessários para nossa page Task onde todas as tasks serão carregadas:

Default



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | import React, { useState, useEffect } from "react";  import {    View,    Text,    TouchableOpacity,    FlatList,  } from "react-native";  import database from "../../config/firebase.js";  import { FontAwesome } from "@expo/vector-icons";  import styles from "./style"; |

2 – Vamos criar nosso componente que irá retornar a page Task e criar a parte lógica do nosso componente, onde teremos o state task, o useEfect e a função deleteTask para deletar uma task criada:

Default



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | export default function Task({ navigation }) {    const [task, setTask] = useState([]);      function deleteTask(id) {      database.collection("Tasks").doc(id).delete();    }      useEffect(() => {      database.collection("Tasks").onSnapshot((query) => {        const list = [];        query.forEach((doc) => {          list.push({ ...doc.data(), id: doc.id });        });        setTask(list);      });    }, []); |

3 – Vamos prepara nosso return onde iremos renderizar a lista de tasks que vamos trazer do Firebase:

Default



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47 | return (      return (      <View style={styles.container}>        <FlatList          showsVerticalScrollIndicator={false}          data={task}          renderItem={( { item } )=>{             return(            <View style={styles.Tasks}>              <TouchableOpacity                style={styles.deleteTask}                onPress={() => {                  deleteTask(item.id)                }}              >              <FontAwesome                name="star"                size={23}                color="#F92e6A"              >              </FontAwesome>              </TouchableOpacity>              <Text                style={styles.DescriptionTask}                onPress={()=>                  navigation.navigate("Details",{                    id: item.id,                    description: item.description,                  })                }              >              {item.description}              </Text>              </View>            )          }}        />        <TouchableOpacity style          style={styles.buttonNewTask}          onPress={() => navigation.navigate("New Task")}        >          <Text style={styles.iconButton}>+</Text>        </TouchableOpacity>      </View>    )  } |

Para finalizar, vamos adicionar algum estilo para deixar nosso app mais bonito:

Default



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48 | import { StyleSheet } from 'react-native'    const styles = StyleSheet.create({    container: {      flex: 1,      backgroundColor:"#fff",      paddingTop: 20  },  Tasks:{    width:"100%",    flexDirection:"row",    justifyContent:"space-between",    marginTop:5  },  deleteTask:{     justifyContent:"center",     paddingLeft:15,  },  DescriptionTask:{    width:"75%",    alignContent:"flex-start",    backgroundColor:"#f5f5f5cf",    padding:12,    paddingHorizontal: 20,    borderRadius:50,    marginBottom: 5,    marginRight:15,    color:"#282b2db5",  },  buttonNewTask:{    width:60,    height:60,    position:"absolute",    bottom: 30,    left:20,    backgroundColor:"#F92e6a",    borderRadius:50,    justifyContent:"center",    alignItems: "center"  },  iconButton:{    color:"#ffffff",    fontSize:25,    fontWeight:"bold",  },  });    export default styles |

### Page NewTask

Constuindo o index.js da Page New Task:

1- Vamos realizar os imports necessários para nossa page New Task onde iremos criar uma nova task:

Default



|  |  |
| --- | --- |
| 1  2  3  4 | import React, { useState } from "react";  import { View, Text, TextInput, TouchableOpacity } from "react-native";  import database from '../../config/firebase.js'  import styles from "./style"; |

2 – Vamos criar nosso componente que irá retornar a page New Task e criar a parte lógica do nosso componente, onde teremos o state description e a função addTask para adicionar uma nova task:

Default



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | export default function NewTask({ navigation }, props) {      const [description, setDescription] = useState(null);      function addTask(){      database.collection('Tasks').add({        description: description,        status: false      })      navigation.navigate("Task");    } |

3 – Vamos preparar nosso return onde iremos ter nosso textinput para adicionarmos uma nova task:

Default



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | return(      <View style={styles.container}>        <Text style={styles.label}>Description</Text>        <TextInput        style={styles.input}        placeholder="Ex: estudar javascript"        onChangeText={setDescription}        value={description}        />        <TouchableOpacity          style={styles.buttonNewTask}          onPress={()=>{            addTask()          }}        >          <Text style={styles.iconButton}>Save</Text>        </TouchableOpacity>      </View>    )  } |

Para finalizar, vamos adicionar algum estilo para deixar nosso app mais bonito:

Default



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44 | import { StyleSheet } from 'react-native'    const styles = StyleSheet.create({  container: {     flex:1,     backgroundColor:'#fff'  },  label:{     width:"90%",     marginTop: 20,     fontSize:16,     marginLeft: 20,     color:"#F92E6A"  },  input:{    width:"90%",    marginTop:10,    padding:10,    height:50,    borderBottomWidth: 1,    borderBottomColor:"#F92E6A",    marginLeft:"auto",    marginRight:"auto"  },  buttonNewTask:{    width:60,    height:60,    position:"absolute",    bottom: 30,    left:20,    backgroundColor:"#F92e6a",    borderRadius:50,    justifyContent:"center",    alignItems: "center"  },  iconButton:{    color:"#ffffff",    fontSize:16,    fontWeight:"bold",  }    });    export default styles |

### P****age Details****

Constuindo o index.js da Page Details:

1- Vamos realizar os imports necessários para nossa page Details onde poderemos ver uma task e editá-la:

Default



|  |  |
| --- | --- |
| 1  2  3  4  5 | import React, { useState } from "react"  import {View, Text, TextInput, TouchableOpacity}  from "react-native"    import database from "../../config/firebaseconfig"  import styles from "./style" |

2 – Vamos criar nosso componente que irá retornar a page Details e criar a parte lógica do nosso componente, onde teremos o state descriptioEdit, teremos uma const idTask e a função editTask:

Default



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | export default function Details({navigation, route}){    const [descriptionEdit, setDescriptionEdit] = useState(route.params.description)    const idTask = route.params.id      function editTask(description, id){      database.collection("Tasks").doc(id).update({        description: description,      })      navigation.navigate("Task")    } |

3 – Vamos preparar nosso return, que irá permitir a visualização e a edição da nossa task:

Default



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | return(      <View style={styles.container}>        <Text style={styles.label}>Description</Text>        <TextInput        style={styles.input}        placeholder="Ex: estudar javascript"        onChangeText={setDescriptionEdit}        value={descriptionEdit}        />        <TouchableOpacity          style={styles.buttonNewTask}          onPress={()=>{            editTask(descriptionEdit, idTask)          }}        >          <Text style={styles.iconButton}>Save</Text>        </TouchableOpacity>      </View>    )  } |

Para finalizar, vamos adicionar algum estilo para deixar nosso app mais bonito:

Default



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44 | import { StyleSheet } from 'react-native'    const styles = StyleSheet.create({  container: {     flex:1,     backgroundColor:'#fff'  },  label:{     width:"90%",     marginTop: 20,     fontSize:16,     marginLeft: 20,     color:"#F92E6A"  },  input:{    width:"90%",    marginTop:10,    padding:10,    height:50,    borderBottomWidth: 1,    borderBottomColor:"#F92E6A",    marginLeft:"auto",    marginRight:"auto"  },  buttonNewTask:{    width:60,    height:60,    position:"absolute",    bottom: 30,    left:20,    backgroundColor:"#F92e6a",    borderRadius:50,    justifyContent:"center",    alignItems: "center"  },  iconButton:{    color:"#ffffff",    fontSize:16,    fontWeight:"bold",  }    });    export default styles |

### Resolvendo Warnings

Se tiver warnings ao adicionar uma nova task, siga os seguintes passos:

* Acesse **node\_modules / react-native / Libraries / Core / Timer / JSTimers.js**
* Procure a variável **MAX\_TIMER\_DURATION\_MS**
* Altere 60 \* 1000 para **10000 \* 1000**
* Salve as alterações e reconstrua seu aplicativo.