**Guia para Configuração e Execução de um Projeto React Native (Expo)**

Este guia é uma explicação passo a passo extremamente detalhada de como instalar ferramentas, configurar um ambiente de desenvolvimento, entender a estrutura de um projeto React Native.

**Passo 1: Preparando o Ambiente de Desenvolvimento**

**1. Instalar o Visual Studio Code (VSCode)**

O Visual Studio Code é o editor de texto recomendado para desenvolvimento de projetos React Native.

Acesse o site oficial: <https://code.visualstudio.com/>.

Baixe a versão apropriada para o seu sistema operacional (Windows, macOS ou Linux).

Siga o processo de instalação. No Windows, você pode simplesmente clicar em "Próximo" até o final.

Após a instalação, abra o VSCode e familiarize-se com ele. É uma ferramenta poderosa com extensões como:

**ESLint**: Para verificar problemas no código.

**Prettier**: Para formatar o código.

**React Native Tools**: Extensão específica para desenvolvimento React Native.

**2. Instalar Node.js**

O Node.js é necessário para executar comandos npm (Node Package Manager) e rodar o servidor do projeto.

Baixe o Node.js no site oficial: <https://nodejs.org/>.

Escolha a versão LTS (Long Term Support), pois é mais estável.

Instale o Node.js e marque a opção para adicioná-lo ao PATH (isso facilita o uso de comandos node e npm no terminal).

Verifique a instalação:

Abra o terminal (Prompt de Comando, PowerShell ou terminal do VSCode) e execute:

**node -v**

Isso deve exibir a versão do Node.js.

Verifique também o npm:

**npm -v** - Esse comando confirma a instalação do Node Package Manager.

3**. Instalar o Expo CLI**

Expo é uma plataforma para rodar e construir apps React Native de forma simplificada. O CLI permite gerenciar e rodar projetos Expo.

Abra o terminal e execute o seguinte comando:

**npm install -g expo-cli**

Esse comando instala o Expo CLI globalmente no sistema.

Verifique a instalação:

**expo --version**

Isso retorna a versão do Expo CLI.

**Passo 2: Configurando o Projeto**

1. **Baixar o Projeto**

Se você recebeu o projeto em um arquivo ZIP:

Extraia o arquivo usando o gerenciador de arquivos do sistema.

Certifique-se de que a pasta do projeto contém arquivos como App.js, package.json, app.json, etc.

2. **Abrir o Projeto no VSCode**

Abra o Visual Studio Code.

Clique em File > Open Folder (ou Arquivo > Abrir Pasta).

Navegue até a pasta do projeto e selecione-a.

3. **Instalar Dependências**

As dependências do projeto estão listadas no arquivo package.json. Para instalá-las:

No terminal integrado do VSCode (abra-o com `Ctrl + ``):

npm install -Isso baixa todas as bibliotecas necessárias para rodar o app.

**Passo 3: Executando o Projeto**

1. Iniciar o Servidor Expo

No terminal, estando na raiz do projeto, execute:

npx expo start - Esse comando inicia o servidor de desenvolvimento e abre uma interface no navegador.

2. Rodar no Emulador Android

Certifique-se de que o emulador Android está em execução (abra-o no Android Studio, se necessário).

Na interface do Expo aberta no navegador, clique em Run on Android device/emulator.

O Expo instalará automaticamente o app no emulador e iniciará o aplicativo.

**Estrutura do Projeto**

**Arquivos e Diretórios Importantes**

**App.js:**

**Ponto de entrada do aplicativo.**

**Contém a lógica principal e configura a navegação.**

**app.json:**

**Arquivo de configuração do Expo (nome do app, ícone, permissões, etc.).**

**firebaseConfig.js:**

**Configuração do Firebase, usada para conectar o app a serviços de backend (ex.: autenticação e banco de dados).**

**package.json:**

**Lista as dependências do projeto e scripts, como:**

**start: Inicia o servidor.**

**build: Cria uma versão de produção.**

**screens/:**

**Contém as telas do aplicativo, como login, perfil, etc.**

**assets/:**

**Armazena recursos estáticos, como imagens e ícones.**

**Funcionalidades do Projeto**

Com base na estrutura dos arquivos:

Autenticação com Firebase:

O app pode autenticar usuários por e-mail, senha ou outros métodos do Firebase.

Navegação entre Telas:

Provavelmente implementada com react-navigation.

Armazenamento de Dados:

Pode usar Firebase Firestore ou Realtime Database para salvar informações.

Interface Responsiva:

Utilizando componentes React Native para adaptar o design a diferentes dispositivos.

**Resolvendo Problemas Comuns**

Firebase não conecta

Verifique se as credenciais em firebaseConfig.js correspondem ao projeto correto no console do Firebase.

Descrição do Projeto: Funcionalidades, Ferramentas e Tecnologias Utilizadas

**Autenticação com Firebase**

O código utiliza o Firebase para autenticar usuários por e-mail e senha.

Funções como login e registro são implementadas com Firebase Authentication.

Caso um usuário não esteja autenticado, ele é redirecionado para a tela de login.

**2. Tela de Login**

A tela de login permite que o usuário insira seu e-mail e senha para autenticação.

Ao inserir os dados corretamente, o usuário é autenticado e redirecionado para a tela principal (provavelmente de navegação ou home).

**3. Tela de Cadastro**

Na tela de cadastro, o usuário pode criar uma conta fornecendo um e-mail e uma senha.

Após o cadastro, o usuário é autenticado e redirecionado para a tela principal.

**4. Tela Principal (Home)**

Após o login ou registro bem-sucedido, o usuário é direcionado para a tela principal do app.

Nessa tela, você pode adicionar funcionalidades específicas do seu aplicativo, como exibir uma lista de dados ou navegar para outras seções.

**5. Navegação entre Telas**

O código implementa navegação entre telas usando a biblioteca react-navigation.

O fluxo de navegação inclui transições entre telas de login, cadastro e a tela principal (Home).

**6. Gerenciamento de Sessão**

O aplicativo gerencia a sessão do usuário por meio do Firebase, garantindo que a autenticação seja persistente.

Se o usuário já estiver autenticado, ele é redirecionado diretamente para a tela principal, sem precisar se autenticar novamente.

**7. Tela de Recuperação de Senha (Possível)**

Embora não mencionada diretamente, o código pode ser facilmente modificado para incluir uma funcionalidade de recuperação de senha, aproveitando os recursos do Firebase Authentication para enviar um e-mail de redefinição de senha.

**8. Verificação de Usuário Logado**

Antes de carregar a interface principal, o app verifica se há um usuário logado (verificando o estado de autenticação do Firebase).

Se o usuário não estiver logado, ele será redirecionado para a tela de login.

**- Autenticação com Firebase:**

A autenticação de usuários é realizada através do Firebase Authentication, permitindo login com e-mail e senha.

O app oferece a funcionalidade de registro de novos usuários, com a verificação do e-mail para garantir que o usuário é legítimo.

Além disso, o app também permite a recuperação de senha via envio de e-mail de reset para o usuário.

**- Navegação entre Telas:**

O projeto utiliza a biblioteca React Navigation para gerenciar a navegação entre diferentes telas do app.

As telas principais incluem a tela de Login, Cadastro, Tela Principal, Tela de Coleta, entre outras.

A navegação é fluida e permite a transição entre as telas de forma intuitiva, utilizando stacks, tabs e outros tipos de navegação.

Armazenamento de Dados no Firebase:

O app usa o Firebase Firestore (ou possivelmente o Firebase Realtime Database) para armazenar dados de forma eficiente e escalável.

Os dados incluem informações de usuários, coletas registradas, status das coletas (pendentes ou aprovadas) e relatórios.

**Interface Responsiva:**

Para garantir que o app seja acessível em diferentes dispositivos, o design foi construído de forma responsiva, utilizando componentes flexíveis do React Native.

A interface foi projetada para oferecer uma boa experiência em diversos tamanhos de tela, desde smartphones de pequenas dimensões até tablets maiores.

**Adicionar Nova Coleta:**

O app permite que os usuários registrem novas coletas, preenchendo informações como a data, a localização e o tipo de coleta.

Os dados das coletas são salvos no banco de dados Firebase e podem ser acessados posteriormente para consultas e relatórios.

**Aprovar Coleta:**

O administrador ou um usuário com permissões adequadas pode aprovar ou rejeitar coletas que estão pendentes.

Uma lista de coletas pendentes de aprovação é exibida no app, e o administrador pode marcar as coletas como aprovadas ou rejeitadas.

Quando a coleta é aprovada, o status no banco de dados é atualizado para "aprovada", e o usuário é notificado.

**Ver Relatório da Coleta:**

O app possui uma funcionalidade para que administradores ou usuários autorizados possam visualizar relatórios das coletas realizadas.

Relatórios podem ser filtrados por data, status (aprovada ou pendente), e outros critérios específicos.

Cada relatório contém detalhes sobre a coleta, como data, localização, tipo de coleta e o status de aprovação.

**Ferramentas e Tecnologias Usadas**

React Native:

O React Native é a principal tecnologia utilizada para construir o app. Ele permite a criação de aplicativos móveis nativos utilizando JavaScript e React.

Com o React Native, o app pode ser executado tanto em dispositivos Android quanto iOS, aproveitando a maioria dos recursos nativos de cada plataforma.

**Firebase:**

O Firebase é um conjunto de ferramentas fornecido pelo Google para facilitar o desenvolvimento de aplicativos móveis e web.

No projeto, são usados os seguintes serviços do Firebase:

Firebase Authentication para autenticação de usuários.

Firebase Firestore ou Realtime Database para o armazenamento de dados como coletas, usuários e relatórios.

Firebase Email Verification para garantir que os e-mails dos usuários são válidos.

**React Navigation:**

O React Navigation é a biblioteca utilizada para gerenciar a navegação entre as diferentes telas do app.

Ele permite a criação de navegação em stack, tabs e drawers, garantindo que a transição entre as telas seja rápida e suave.

**Expo:**

O Expo é uma plataforma que facilita o desenvolvimento de aplicativos React Native. Ele fornece uma série de ferramentas para inicializar e rodar o projeto de maneira simplificada, sem a necessidade de configurar o ambiente nativo manualmente.

O Expo CLI foi usado para inicializar o projeto, o que agilizou o processo de desenvolvimento e testes.

**Visual Studio Code (VSCode):**

O Visual Studio Code foi o editor utilizado para escrever o código do projeto. Ele é uma ferramenta poderosa e leve, com extensões que ajudam a identificar problemas no código, formatar automaticamente e facilitar a navegação entre arquivos.

Extensões como ESLint (para checagem de erros de código), Prettier (para formatação) e React Native Tools (para facilitar o desenvolvimento com React Native) foram utilizadas no projeto.

**Node.js e npm:**

O Node.js é a plataforma JavaScript usada para rodar o servidor de desenvolvimento e executar os comandos de instalação de pacotes. Ele permite que o código JavaScript seja executado no servidor, fora do navegador.

O npm (Node Package Manager) foi usado para gerenciar as dependências do projeto, como o Expo, React Navigation, Firebase, entre outros.

**Android Studio:**

O Android Studio foi utilizado para configurar o emulador Android, permitindo rodar o app em um ambiente simulado. Ele também fornece ferramentas para depuração e teste de aplicativos Android.

**Axios (ou Fetch):**

O Axios (ou o método nativo fetch) foi utilizado para fazer requisições HTTP, como para o envio e recebimento de dados com o backend, seja para autenticação ou operações de coletas.