Próximos Passos: Configuração do Projeto Expo

Este documento detalha os passos necessários para iniciar o desenvolvimento da versão nativa iOS do ModularCompany usando Expo.

1. Preparação do Ambiente

1.1 Instalação de Ferramentas

```
# Instalar o Expo CLI globalmente
npm install -g expo-cli

# Instalar o Expo Go no dispositivo iOS para testes
# (disponível na App Store)
```

1.2 Criação do Projeto

```
# Criar um novo projeto Expo com TypeScript
expo init ModularCompanyMobile

# Selecione o template: "Blank (TypeScript)"

# Navegar para o diretório do projeto
cd ModularCompanyMobile
```

1.3 Instalação de Dependências Essenciais

```
# Navegação
npm install @react-navigation/native @react-navigation/native-stack @react-
navigation/bottom-tabs
npm install react-native-screens react-native-safe-area-context

# UI
npm install react-native-paper react-native-vector-icons
npm install react-native-keyboard-aware-scroll-view

# Autenticação e Armazenamento
npm install expo-secure-store expo-auth-session expo-web-browser

# Ferramentas de Desenvolvimento
npm install --save-dev @types/react-native-vector-icons
```

2. Estrutura do Projeto

Criar a seguinte estrutura de diretórios:

```
ModularCompanyMobile/
├── assets/              # Ícones, imagens e outros recursos
├── src/
│   ├── api/             # Integração com a API do backend
│   ├── components/      # Componentes reutilizáveis
│   ├── hooks/           # Hooks personalizados
│   ├── navigation/      # Configuração de navegação
│   ├── screens/         # Telas da aplicação
│   ├── types/           # Definições de tipos TypeScript
│   └── utils/           # Utilitários
```

3. Configuração Inicial

3.1 Tema e Estilos

Criar um arquivo de tema que corresponda à identidade visual da web:

```
// src/theme/index.ts
import { DefaultTheme } from 'react-native-paper';

export const theme = {
  ...DefaultTheme,
  colors: {
    ...DefaultTheme.colors,
    primary: '#0f172a',
    secondary: '#f1f5f9',
    background: '#ffffff',
    surface: '#ffffff',
    error: '#ef4444',
    text: '#0f172a',
    disabled: '#a1a1aa',
  },
};
```

3.2 Configuração da Navegação

```
// src/navigation/index.tsx
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';

import LoginScreen from '../screens/LoginScreen';
import DashboardScreen from '../screens/DashboardScreen';
import TimeEntryScreen from '../screens/TimeEntryScreen';
```

```
import ProfileScreen from '../screens/ProfileScreen';

const Stack = createNativeStackNavigator();
const Tab = createBottomTabNavigator();

const AppTabs = () => {
  return (
    <Tab.Navigator>
      <Tab.Screen name="Dashboard" component={DashboardScreen} />
      <Tab.Screen name="TimeEntry" component={TimeEntryScreen} />
      <Tab.Screen name="Profile" component={ProfileScreen} />
    </Tab.Navigator>
  );
};

export default function Navigation() {
  // Estado de autenticação seria verificado aqui
  const isAuthenticated = false;

  return (
    <NavigationContainer>
      <Stack.Navigator>
        {isAuthenticated ? (
          <Stack.Screen
            name="Main"
            component={AppTabs}
            options={{ headerShown: false }}
          />
        ) : (
          <Stack.Screen
            name="Login"
            component={LoginScreen}
            options={{ headerShown: false }}
          />
        )}
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

4. Implementação de Componentes Base

4.1 Botão

Implementar o componente Button.tsx seguindo o design da versão web:

```
// src/components/Button.tsx
import React from 'react';
import { TouchableOpacity, Text, StyleSheet, ViewStyle, TextStyle } from
'react-native';
```

```tsx
interface ButtonProps {
  onPress: () => void;
  title: string;
  variant?: 'default' | 'destructive' | 'outline' | 'secondary' | 'ghost' |
'link';
  size?: 'default' | 'sm' | 'lg' | 'icon' | 'iconSm' | 'full';
  disabled?: boolean;
  style?: ViewStyle;
  textStyle?: TextStyle;
}

export const Button: React.FC<ButtonProps> = ({
  onPress,
  title,
  variant = 'default',
  size = 'default',
  disabled = false,
  style,
  textStyle,
}) => {
  // Implementação completa seguindo os esquemas da versão conceitual
};
```

4.2 Card

Implementar o componente Card.tsx seguindo o design da versão web.

4.3 Input

Criar componentes de entrada adaptados para dispositivos móveis.

5. Telas Principais

5.1 Tela de Login

Implementar a tela de login com as seguintes funcionalidades:

- Campos de email e senha
- Validação de formulário
- Integração com a API de autenticação
- Feedback visual para erros e carregamento

5.2 Dashboard do Funcionário

Implementar o dashboard com as seguintes funcionalidades:

- Visualização de estatísticas
- Lista de registros de horas recentes
- Ações rápidas para novo registro

5.3 Registro de Horas

Implementar a tela de registro de horas com:

- Seleção de data
- Seleção de horário de início e fim
- Campo de observações
- Validação de conflitos

6. Integração com API

6.1 Configuração do Cliente HTTP

```
// src/api/client.ts
import axios from 'axios';
import * as SecureStore from 'expo-secure-store';

const API_URL = 'https://api.modularcompany.com'; // Substituir pela URL real

export const apiClient = axios.create({
  baseURL: API_URL,
  headers: {
    'Content-Type': 'application/json',
  },
});

// Interceptor para adicionar token de autenticação
apiClient.interceptors.request.use(
  async (config) => {
    const token = await SecureStore.getItemAsync('auth_token');
    if (token) {
      config.headers.Authorization = `Bearer ${token}`;
    }
    return config;
  },
  (error) => {
    return Promise.reject(error);
  }
);
```

6.2 Implementação de Serviços

Criar serviços para cada endpoint da API necessário para a aplicação móvel.

7. Testes e Publicação

7.1 Testes em Dispositivos Reais

- Testar no Expo Go durante o desenvolvimento

- Testar em dispositivos iOS físicos para garantir a experiência

7.2 Build para TestFlight

```
# Gerar build para iOS
expo build:ios

# Seguir instruções para enviar para o TestFlight
```

7.3 Configuração de CI/CD

Configurar GitHub Actions ou outro serviço de CI/CD para automatizar builds e distribuição para TestFlight.

8. Recursos Adicionais

- Documentação do Expo
- Documentação do React Native
- Guia de Publicação na App Store