

## DOCUMENTAÇÃO DE IMPLANTAÇÃO DO APP ANDROID COM EXPO

### 1. OBJETIVO

Transformar o sistema de combustível existente (web) em um aplicativo Android usando:

- React Native com Expo
- WebView para carregar o sistema em <http://combustivel.blomag.com.br/>
- EAS Build para gerar o APK instalável

### 2. PRÉ-REQUISITOS

- Node.js e npm instalados
- Conta na Expo (expo.dev) com login feito no terminal
- Sistema de combustível acessível pela URL: <http://combustivel.blomag.com.br/>
- Repositório do sistema em: c:\Users\luizb\Downloads\Locador-RequisicaoCombustivel

### 3. CRIAÇÃO DO PROJETO EXPO

3.1 Comando para criar o projeto mobile (executado na pasta raiz do sistema):

```
cd c:\Users\luizb\Downloads\Locador-RequisicaoCombustivel npx create-expo-app app-combustivel
```

3.2 Estrutura principal criada (template com Expo Router):

- app-combustivel/app/\_layout.tsx
- app-combustivel/app/(tabs)/index.tsx
- app-combustivel/app/(tabs)/explore.tsx, etc.

3.3 No package.json do app-combustivel:

- main configurado como "expo-router/entry", por isso não existe App.js/App.tsx na raiz.

### 4. CONFIGURAÇÃO DA WEBVIEW APONTANDO PARA O SISTEMA

4.1 Dependência react-native-webview

- A dependência react-native-webview já está presente em app-combustivel/package.json.
- Se necessário, o comando para instalar seria:

```
npx expo install react-native-webview
```

4.2 Substituição da tela inicial pela WebView

Arquivo modificado: app-combustivel/app/(tabs)/index.tsx

Conteúdo configurado para abrir o sistema de combustível:

- Importação de View, StatusBar e WebView
- Constante SYSTEM\_URL configurada como:

```
const SYSTEM_URL = 'http://combustivel.blomag.com.br/';
```

- Componente HomeScreen renderiza:
  - Uma View com estilo flex: 1
  - StatusBar
  - WebView com:
    - source={{ uri: SYSTEM\_URL }}

- `javaScriptEnabled`
- `domStorageEnabled`
- `startInLoadingState`
- `scalesPageToFit`

Resultado: ao abrir o app, a aba principal (Home) carrega diretamente o sistema web de combustível dentro da WebView.

## 5. CONFIGURAÇÃO VISUAL DO APP (APP.JSON)

Arquivo: `app-combustivel/app.json`

Configurações principais:

- `name: "Combustivel Blomaq"`
- `slug: "combustivel-blomaq"`
- `version: "1.0.0"`
- `orientation: "portrait"`
- `icon: "./assets/images/icon.png"`
- `splash:`
  - `image: "./assets/images/splash-icon.png"`
  - `resizeMode: "contain"`
  - `backgroundColor: "#ffffff"`
- `android:`
  - `adaptiveIcon.foregroundImage: "./assets/images/android-icon-foreground.png"`
  - `adaptiveIcon.backgroundColor: "#ffffff"`
  - `package: "com.blomaq.combustivel"`

### 5.1 Correções aplicadas em relação ao template

Foram corrigidos caminhos inválidos que causavam falhas no prebuild:

- `icon:`
  - Anterior: `"./assets/icon.png"`
  - Correto: `"./assets/images/icon.png"`
- `splash.image:`
  - Anterior: `"./assets/splash.png"`
  - Correto: `"./assets/images/splash-icon.png"`
- `android.adaptiveIcon.foregroundImage:`
  - Anterior: `"./assets/adaptive-icon.png"`
  - Correto: `"./assets/images/android-icon-foreground.png"`

Motivo: os arquivos reais estão na pasta `assets/images`, e não diretamente em `assets`.

## 6. CONFIGURAÇÃO DO EAS BUILD PARA ANDROID

### 6.1 Configuração inicial

Dentro da pasta `app-combustivel` foi executado:

`eas login eas build:configure`

Isso gerou o arquivo eas.json em app-combustivel/eas.json.

## 6.2 Ajuste do perfil preview para gerar APK

Conteúdo relevante de eas.json:

- cli.version: ">= 16.28.0"
- build.development:
  - developmentClient: true
  - distribution: "internal"
- build.preview:
  - distribution: "internal"
  - android.buildType: "apk"
- build.production:
  - autoIncrement: true

Importante: buildType: "apk" garante que o EAS gere um arquivo APK para instalação direta em Android.

## 7. ERROS DE PREBUILD E CORREÇÕES (IMAGENS)

Durante os primeiros builds, o EAS apresentou erros ENOENT, por exemplo:

- ENOENT: no such file or directory, open './assets/icon.png'
- ENOENT: no such file or directory, open './assets/splash.png'
- ENOENT: no such file or directory, open './assets/adaptive-icon.png'

Causa:

- Os caminhos no app.json apontavam para arquivos em ./assets/, mas os arquivos reais estavam em ./assets/images/.

Correção:

- Atualização de todos os caminhos de ícones e splash para assets/images, conforme descrito na seção 5.

Após essa correção, o prebuild passou a encontrar as imagens corretamente.

## 8. ERRO NO APP: ERR\_CLEARTEXT\_NOT\_PERMITTED

### 8.1 Sintoma

Ao abrir o APK instalado no celular, a WebView exibiu:

- Error loading page
- Description: net::ERR\_CLEARTEXT\_NOT\_PERMITTED

### 8.2 Causa

- A URL do sistema usa HTTP sem HTTPS:  
<http://combustivel.blomaq.com.br/>
- Versões recentes do Android bloqueiam tráfego HTTP (cleartext) por padrão, por segurança.
- Resultado: a WebView não consegue carregar a página sem uma configuração explícita permitindo HTTP.

### 8.3 Decisão

- Enquanto a aplicação estiver publicada apenas em HTTP, foi necessário habilitar o uso de cleartext no app Android.
- Isso foi feito com o plugin expo-build-properties, adicionando usesCleartextTraffic=true no AndroidManifest gerado pelo Expo.

## 9. ADIÇÃO DO PLUGIN EXPO-BUILD-PROPERTIES

### 9.1 Dependência no package.json do app-combustivel

No arquivo app-combustivel/package.json, em "dependencies", foi adicionada a linha:

- "expo-build-properties": "^0.12.3"

Em seguida, foi executado:

```
cd app-combustivel npm install
```

### 9.2 Configuração do plugin no app.json

No arquivo app-combustivel/app.json foi adicionado o bloco de plugins:

```
"plugins": [ [ "expo-build-properties", { "android": { "usesCleartextTraffic": true } } ] ]
```

Estrutura final relevante do app.json:

- android:
  - adaptivelcon.foregroundImage: "./assets/images/android-icon-foreground.png"
  - adaptivelcon.backgroundColor: "#ffffff"
  - package: "com.blomaq.combustivel"
- plugins:
  - expo-build-properties com android.usesCleartextTraffic = true

Com isso, o prebuild passou a gerar o AndroidManifest com:

- android:usesCleartextTraffic="true"

Permissão que libera o carregamento de HTTP pela WebView.

## 10. GERAÇÃO DO APK FINAL

### 10.1 Comando de build

Na pasta app-combustivel foi utilizado o comando:

```
eas build -p android --profile preview
```

Fluxo do build:

- Compressão do projeto e upload para a infraestrutura da Expo.
- Execução do prebuild (geração do código nativo Android).
- Compilação do aplicativo com as configurações ajustadas.

### 10.2 Resultado

- Ao final do processo, o EAS disponibiliza um link na interface web da Expo (builds do projeto) para download do arquivo APK.
- Esse APK pode ser instalado diretamente em dispositivos Android.
- Ao abrir o app, a WebView carrega o sistema de combustível a partir da URL <http://combustivel.blomaq.com.br/>, agora sem o erro ERR\_CLEARTEXT\_NOT\_PERMITTED.

## 11. RESUMO DOS PRINCIPAIS PONTOS

- Projeto mobile criado com Expo (create-expo-app) na pasta app-combustivel.
- WebView configurada em app/(tabs)/index.tsx apontando para <http://combustivel.blomaq.com.br/>.
- app.json ajustado para usar ícones e splash existentes em assets/images.
- eas.json configurado com perfil preview para gerar APK (buildType: "apk").
- Correção de erros de prebuild causados por caminhos de imagem incorretos.
- Tratamento do erro net::ERR\_CLEARTEXT\_NOT\_PERMITTED permitindo HTTP via expo-build-properties e usesCleartextTraffic=true.
- Geração de APK funcional para Android por meio do comando eas build -p android --profile preview.