□ Documentação Técnica - EnerCheck Mobile

Versão: 1.0.0
Data: 25/10/2023
Framework: React Native (Expo SDK 54)
Autor: Arquitetura Mobile

---

1. □ Wireframes e Navegação

□ Visão Geral

EnerCheck é uma aplicação mobile desenvolvida para gerenciamento de projetos elétricos, verificação de conformidade e assinatura de planos.

□ Telas Principais

1. LoginScreen (Autenticação)

Propósito: Tela de login para usuários existentes.
Fluxo:\

- Valida credenciais → Faz login via API → Verifica se usuário tem plano.\
- **SE tem plano →** Dashboard (GeralScreen).\
- **SE não tem plano →** PlanosScreen.

2. RegisterScreen (Cadastro)

Propósito: Cadastro de novos usuários.
Validações: Email regex, CREA (6 dígitos), senha forte.
Fluxo: Cria usuário → Login automático → Redireciona para PlanosScreen.

3. PlanosScreen (Seleção de Plano)

Propósito: Exibir planos (Básico, Pro, Enterprise).
Dados: GET `/api/Planos`.
Fluxo: Seleciona plano → FinalizarEscolhaAssinaturaScreen.

4. FinalizarEscolhaAssinaturaScreen (Pagamento)

Propósito: Finalizar assinatura (Cartão, PIX, Boleto).
Fluxo: Pagamento → Vincula plano (`PUT /api/Usuarios/add/plano`) → Dashboard.

5. GeralScreen (Dashboard Principal)

Propósito: Visão geral com estatísticas e projetos recentes.
Features:\

- `useFocusEffect` para recarregar dados.\
- Badges de status.

6. UploadProjetoScreen (Upload)

Propósito: Envio de arquivos PDF/DWG/XLSX.
Componente: `expo-document-picker`.

7. ProjetoScreen (Detalhes)

Propósito: Visualizar status (Aprovado/Pendente) e detalhes técnicos.

8. SettingsScreen (Configurações)

Propósito: Gerenciar perfil, segurança e assinaturas.

□ Fluxo de Navegação

```
flowchart TD
    Start([Abrir App]) --> CheckAuth{Autenticado?}

    CheckAuth -->|Não| Login[LoginScreen]
    CheckAuth -->|Sim| CheckPlan{Tem Plano?}

    Login -->|Novo Usuário| Register[RegisterScreen]
    Login -->|Login OK| CheckPlan

    Register -->|Cadastro OK| Planos[PlanosScreen]

    CheckPlan -->|Não| Planos
    CheckPlan -->|Sim| Dashboard[GeralScreen - Dashboard]

    Planos -->|Escolher Plano| Payment[FinalizarEscolhaAssinatura]

    Payment -->|Pagar| VincularPlano[Vincular Plano - API]
    VincularPlano --> Dashboard

    Dashboard --> Upload[UploadProjetoScreen]
    Dashboard --> Projeto[ProjetoScreen]
    Dashboard --> Settings[SettingsScreen]

    Settings --> Perfil[Perfil Component]
    Settings --> Assinaturas[Assinaturas Component]

    Assinaturas -->|Alterar Plano| Planos

    style Login fill:#0D6EFD,color:#fff
    style Dashboard fill:#22c55e,color:#fff
    style Payment fill:#fbbf24,color:#000
```

---

2. □ Gerenciamento de Estado

□ Tecnologia Utilizada

- **Context API (React Context)**
- **AsyncStorage**

□ Estrutura da Store

ThemeContext (contexts/ThemeContext.js)

- Propósito: Gerenciar tema (light/dark/automático).\
- Persistência: Salva preferência no AsyncStorage.\
- Sincronização: Ouve alterações do sistema via `useColorScheme`.

□ AsyncStorage -- Persistência Local

Chave Descrição

Token JWT Bearer Token (Auth) refreshToken Token de renovação userData Objeto completo do usuário rememberMe Flag "Lembrar de mim"

□ Fluxo de Dados (Login)

1. Usuário insere credenciais.\
2. API responde `{ token, user }`.\

3. Auth.js salva tokens.\
4. App salva `userData` no cache.\
5. Estado global muda para autenticado.

□ Decisão Arquitetural

Optou-se por **não usar Redux**, pois o estado global é mínimo (apenas Tema).
Os dados de negócio são tratados como **Server State** via API + cache local.

---

3. □ Uso de API e Integração

□ Configuração da API

- Cliente: **Axios v1.13.2**\
- Base URL: `https://enercheck.onrender.com`\
- Autenticação: JWT Bearer Token

Interceptors

- **Request:** Adiciona `Authorization: Bearer {token}`.\
- **Response:**
  - Caso 401 → tenta `/Usuario/refresh` para renovar token.

□ Principais Endpoints

Método Endpoint Descrição Auth

POST /Usuario/login Login □ POST /Usuario/refresh Renovar Token □ GET /api/Usuarios/me Dados do Usuário □ PUT /api/Usuarios/add/plano Vincular Plano □ GET /api/Planos Listar Planos □

□ Estratégia Offline (Planejada)

- Monitoramento de conexão via **NetInfo**\
- Fila de requisições pendentes em **AsyncStorage**\
- Sincronização automática ao reconectar

---

4. ⚡ Performance e Build

□ Otimizações Implementadas

- `enableScreens()` para navegação nativa.\
- `useMemo` e `useCallback` em operações pesadas.\
- `FlatList` otimizada.\
- Uso de ícones vetorizados.

□ Guia de Build e Publicação (EAS)

Pré-requisitos

```
npm install -g eas-cli
eas login
```

---

1. Build Android (AAB)

```
eas build --platform android --profile production
eas submit --platform android
```

---

2. Build iOS (IPA)

```
eas build --platform ios --profile production
eas submit --platform ios --latest
```

Métricas de Performance (Alvo)

Métrica Target

Bundle Size APK ~30 MB Bundle Size IPA ~25 MB JS Bundle ~2 MB Cold Start < 3s Warm Start < 1s Memória (Peak) ~150 MB

Testing Checklist

Antes do Build

- ☐ Remover `console.log` \
- ☐ Testar build release local\
- ☐ Verificar permissões\
- ☐ Validar assets

Após o Build

- ☐ Testar fluxo completo\
- ☐ Testar em dispositivos low-end\
- ☐ Testar rotação e dark mode

Troubleshooting

Erro: *Unable to resolve module*

```
npx expo start --clear
rm -rf node_modules package-lock.json
npm install
```

Falha no Build Android

```
cd android && ./gradlew clean
```

Falha no Build iOS

```
rm -rf ~/Library/Developer/Xcode/DerivedData
cd ios && pod deintegrate && pod install
```

Recursos Úteis

- Expo Build Docs\
- EAS Submit\
- React Native Performance\
- Hermes Engine