Documentação do Aplicativo EcoLight

Integrantes:

[RM552671] - Willian Daniel Oliveira Dantas

[RM554021] - Sara Ingrid da Silva

[RM554328] - João Vitor de Santana dos Santos

- **Link do Vídeo:**
  - https://youtu.be/YSCVVKc8_bA

Descrição Geral

O **EcoLight** é um aplicativo móvel desenvolvido para gerenciar e monitorar o consumo energético doméstico. Ele permite aos usuários cadastrar dispositivos, registrar consumos e metas mensais, calcular valores totais de consumo e impostos, além de exibir insights sobre o uso eficiente de energia elétrica.

Objetivo Principal: Promover a conscientização e a economia de energia elétrica por meio de um controle simples e intuitivo.

Informação importante:

Tivemos problemas com o OracleDB então utilizamos um banco PostgreSQL hospedado no Supabase. Como utilizamos o plano gratuito, influenciou no crud integrado com Java, o que gerou algumas mensagens de "Tente novamente" durante a apresentação, mas se trata de uma questão isolada do banco e da API que está publicada no Render, não se relaciona com a construção do App. Obrigado pela compreensão.

Funcionalidades

1. Gerenciamento de Dispositivos

- Cadastro de dispositivos com potência nominal.

- Listagem de dispositivos cadastrados.
- Atualização e exclusão de dispositivos.

2. Registro e Monitoramento de Consumos

- Registro de consumos por dispositivo.
- Exibição detalhada de consumos em uma lista:
  - Nome do consumo.
  - Potência (associada ao dispositivo).
  - Tempo de uso.
  - Preço total do consumo.

3. Configuração de Metas

- Registro de metas mensais de consumo em reais.
- Atualização de metas existentes.
- Exibição do progresso em relação à meta.

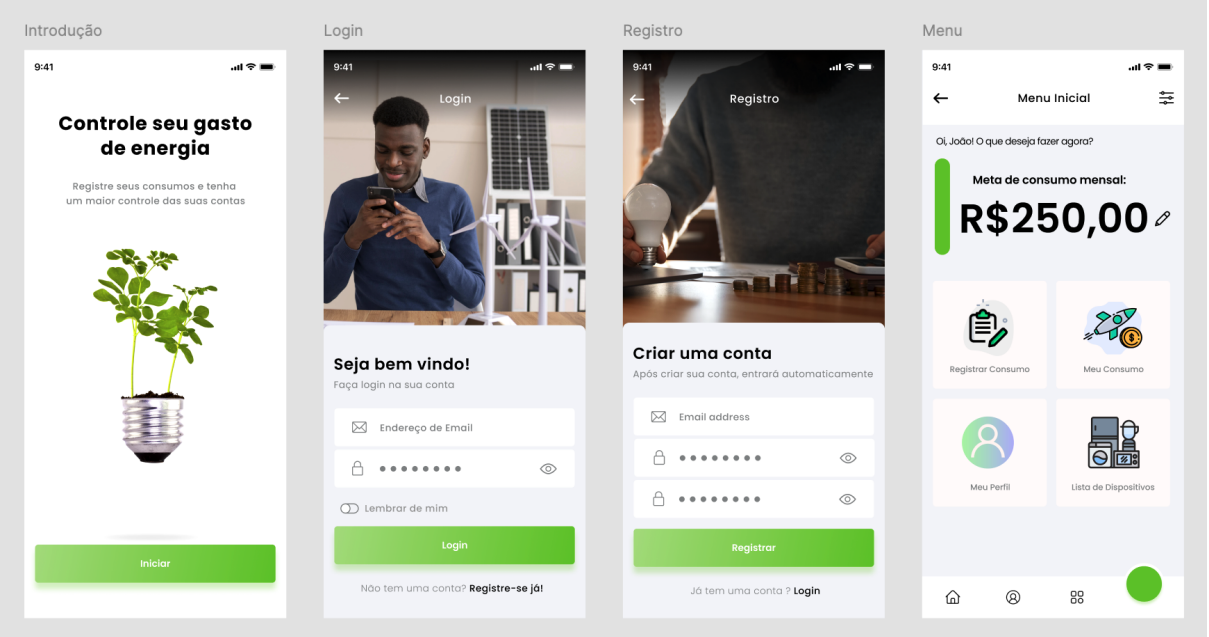
4. Cálculos Automáticos

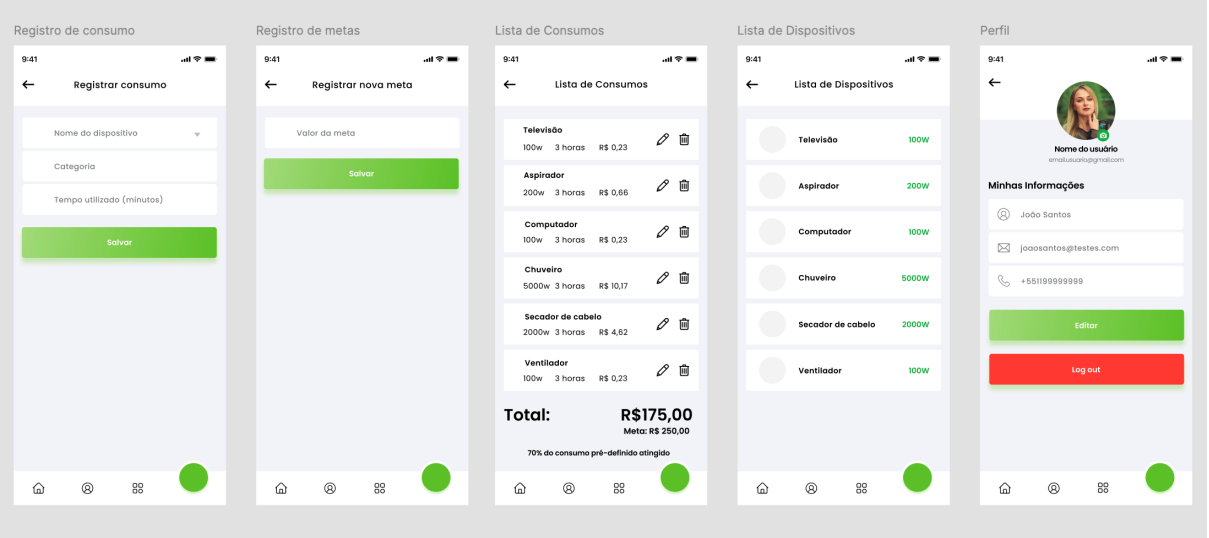
- Cálculo do consumo total mensal.
- Estimativa do valor de impostos baseados no consumo.

5. Integração com Firebase e API Externa

- Autenticação de usuários via Firebase Authentication.
- Sincronização dos dados com uma API REST para persistência.

Imagens do protótipo:





Requisitos do Sistema

Plataforma

- **Android**: Versão 5.0 (Lollipop) ou superior.

Tecnologias Utilizadas

- **Linguagem**: Kotlin.

- **Firebase**:
  - Authentication.
  - Realtime Database.
- **API REST**:
  - Comunicação com servidor externo via Retrofit.
- **Banco de Dados**:
  - PostgreSQL no backend da API.

Bibliotecas

- Retrofit: Comunicação com API REST.
- Firebase: Autenticação e banco de dados em tempo real.
- Gson: Conversão de JSON.
- ConstraintLayout: Layout responsivo.

Instalação

Passo 1: Clonar o Repositório

```
bash
git clone https://github.com/jvs4nt/EcoLight.git
```

Passo 2: Abrir no Android Studio

1. Abra o Android Studio.
2. Clique em **File > Open** e selecione a pasta clonada.
3. Aguarde o carregamento das dependências.

Passo 3: Configurar Firebase

1. Baixe o arquivo `google-services.json` do seu console Firebase.
2. Adicione-o à pasta `app/` do projeto.

Passo 4: Configurar Base URL

- Altere o valor da constante `BASE_URL` no arquivo `RetrofitClient`:

```
kotlin
private const val BASE_URL = "https://consumoenergiaapi.
onrender.com/"
```

Passo 5: Executar o Aplicativo

1. Conecte um dispositivo Android ou configure um emulador.
2. Clique em **Run > Run 'app'**.

Arquitetura do Aplicativo

O EcoLight utiliza a **Arquitetura MVVM**:

Componentes

1. **Models**:
   - Representam os dados e a lógica de negócios (ex.: `Device`, `Consumption`, `Meta`).
2. **Views**:
   - Representam as telas do aplicativo (XML + Activity/Fragment).
3. **ViewModels**:
   - Intermediam a comunicação entre as `Views` e os dados no `Model`.

Fluxo do Aplicativo

1. **Tela de Login**:
   - Usuário realiza login ou navega para a tela de registro.
2. **Tela Inicial**:
   - Exibe a meta mensal atual ou permite criar uma nova meta.
   - Botões para navegação:
     - Registrar consumo.

    - Ver lista de consumos.
    - Gerenciar dispositivos.
3. **Tela de Registro de Consumos**:
    - Seleciona um dispositivo.
    - Insere o tempo de uso.
    - Salva o consumo no Firebase.
4. **Tela de Lista de Consumos**:
    - Lista detalhada de todos os consumos registrados.
    - Botões para editar ou excluir um consumo.
5. **Tela de Registro/Edição de Metas**:
    - Insere ou atualiza a meta mensal.

Detalhes Técnicos

Firebase

- **Autenticação**:
    - Permite login e registro de usuários.
- **Realtime Database**:
    - Estrutura:

```
json
{
  "consumption": {
    "usuarioEmail": {
      "consumoId": {
        "name": "Consumo X",
        "timeUsed": "60",
        "deviceId": "123",
        "deviceName": "Dispositivo Y"
      }
    }
  },
```

```
  "devices": {
    "usuarioEmail": {
      "deviceId": {
        "name": "Dispositivo Y",
        "power": "200W"
      }
    }
  }
}
```

API REST

- **Endpoints Importantes**:
  - `POST /api/metas`: Cria uma nova meta.
  - `GET /api/metas?usuarioEmail=email`: Retorna a meta associada ao email.
  - `PUT /api/metas/{id}`: Atualiza uma meta pelo ID.

Endpoints da API

1. Registrar Meta

- **URL**: `POST /api/metas`
- **Body**:

```
json
Copiar código
{
  "valorMeta": 200.0,
  "dataCadastro": "2024-11-22",
  "usuarioEmail": "usuario@email.com"
}
```

- **Resposta**:
  - `200 OK`: Meta criada.

- `500 Internal Server Error`: Problema no servidor.

2. Buscar Meta por Email

- **URL**: `GET /api/metas?usuarioEmail=usuario@email.com`
- **Resposta**:

```
json
Copiar código
{
  "id": 1,
  "valorMeta": 200.0,
  "dataCadastro": "2024-11-22",
  "usuarioEmail": "usuario@email.com"
}
```

3. Atualizar Meta

- **URL**: `PUT /api/metas/{id}`
- **Body**:

```
json
Copiar código
{
  "valorMeta": 250.0,
  "dataCadastro": "2024-11-23",
  "usuarioEmail": "usuario@email.com"
}
```

Considerações de Segurança

1. **Autenticação Firebase**:
   - Garantir que apenas usuários autenticados acessem dados.
   - Restringir acesso ao Firebase Database usando regras de segurança.

2. **Validação de Dados**:
    - Validar todos os dados enviados para o servidor (e.g., emails, valores numéricos).
3. **HTTPS**:
    - Certifique-se de que a API está servindo via HTTPS para proteger os dados em trânsito.
