Aplicativo Analisador de Texto (Versão Completa)

1. Objetivo

Consolidar todos os conceitos da Unidade 2 (Arquitetura MVVM, Persistência de Dados com Banco de Dados Local e Gerenciamento Avançado de Eventos/Estado) para entregar uma versão completa e robusta do aplicativo "Analisador de Texto".

O foco desta entrega é a implementação de um fluxo completo de autenticação de usuário (Cadastro e Login), com persistência de dados em banco de dados e validações de formulário complexas.

2. Requisitos Funcionais (Fluxo de Telas)

O aplicativo deverá conter 4 telas principais, integradas em um fluxo de usuário coeso.

Tela 1: Cadastro de Usuário (tela_cadastro.dart)

Esta é a tela com os requisitos de validação mais complexos.

Campos Obrigatórios:

- **Nome Completo:** Deve ser obrigatório. Deve conter **nome e sobrenome** (a validação deve falhar se o usuário digitar apenas uma palavra, ex: "Diego").
- **CPF:** Deve ser obrigatório. (aplicar uma máscara de formato).
- **Data de Nascimento:** Deve ser obrigatória. (usar um DatePicker).
- **E-mail:** Deve ser obrigatório e validar o formato de e-mail.
- **Senha:** Deve ser obrigatória. Deve possuir um **ícone de visibilidade** (alternar obscureText).
- **Confirmação de Senha:** Deve ser obrigatória. Deve ser idêntica ao campo "Senha".

Validação de Senha (Requisito Avançado):

A senha só será considerada válida se atender a **todas** as seguintes regras:

1. Conter pelo menos um caractere maiúsculo (A-Z).
2. Conter pelo menos um caractere minúsculo (a-z).
3. Conter pelo menos um número (0-9).
4. Conter pelo menos um caractere especial (ex: !@#$%^&*()).

O formulário deve conter um **validador visual** (ex: uma lista de checkboxes ou ícones) que informe ao usuário, em tempo real, quais regras ele já cumpriu.

O botão "Cadastrar" só deve ficar habilitado quando **todos** os campos do formulário estiverem preenchidos e válidos.

Tela 2: Login (tela_login.dart)

- Deve conter campos para **E-mail** e **Senha**.
- Deve possuir um link ou botão "Ainda não tem conta? Cadastre-se", que navega para a tela_cadastro.dart.
- Ao pressionar "Entrar", o aplicativo deve consultar o banco de dados local.
- Se o usuário e senha estiverem corretos, navegar para a tela_principal.dart.
- Se incorretos, exibir uma mensagem de erro (ex: SnackBar ou Text).

Tela 3: Tela Principal (tela_principal.dart)

- Esta é a tela principal do Analisador de Texto (desenvolvida em aula).
- Esta tela **não pode** ser acessada sem que o usuário tenha feito o login.
- Exibir o nome do usuário logado, ex: "Bem-vindo, **Nome do Usuario**!".

Tela 4: Tela de Resultados (tela_resultados.dart)

- Esta é a tela que exibe os resultados da análise (desenvolvida em aula).
- Ela deve receber e exibir corretamente os dados processados pela Tela Principal.

3. Requisitos Técnicos (Não-Funcionais)

- **Arquitetura MVVM:** O projeto **deve** estar estruturado no padrão Model-View-ViewModel.
- **Views(Telas e Widgets):** Devem conter apenas a lógica de UI.
- **ViewModels:** Devem conter toda a lógica de estado da tela e validações de formulário.
- **Service:** Contém toda a lógica de negócios e chamadas ao banco de dados.
- **Models:** Devem representar as entidades de dados (ex: Usuario.dart).
- **Banco de Dados Local (sqflite):**
- O aplicativo deve inicializar um banco sqflite.
- Todos os dados do usuário (nome, cpf, email, senha criptografada*) devem ser salvos no banco durante o cadastro.
- A tela de Login deve consultar este banco para autenticar o usuário.
- Armazenar a senha usando um pacote de hash, como crypto, e não em texto puro.
- **Gerenciamento de Eventos/Estado:**
- Deve-se fazer uso de Form com GlobalKey para validação.
- O estado da UI (ex: habilitação de botões, exibição de erros, status de carregamento) deve ser controlado pelo ViewModel (ex: usando ChangeNotifierProvider, ValueNotifier, etc.).

4. Entrega e Avaliação

Para facilitar a correção e garantir a execução do ambiente, a entrega deverá ser feita da seguinte forma:

1. **Método Preferencial (Repositório Git):**

- Disponibilizar o projeto completo em um **repositório Git público** (ex: GitHub).
- O professor irá clonar o repositório para testar o funcionamento.
- *(Recomendação: Se utilizar o **Codespaces** para construir o projeto, basta compartilhar o link do repositório público criado).*

2. **Métodos Alternativos (App Compilado):**

- Quem preferir, pode entregar o aplicativo já compilado e funcional e mandar pelo blackboard o arquivo.rar com o código:
- **Opção A:** Gerar o **APK** (Android) e enviar o arquivo.
- **Opção B:** Fazer o *deploy* da versão **Web** e enviar o link público.