Teste Técnico para Desenvolvedor - Flutter com Firebase

Objetivo:

Desenvolver um aplicativo móvel utilizando Flutter para autenticação via OAuth, consumo de uma API pública

Especificações Técnicas:

1. Autenticação OAuth (E-mail e Senha):

- **Tecnologia:**
  - Utilizar o Firebase Authentication com suporte a login com email e senha.

- **Funcionalidade:**
  - O usuário deve ser apresentado a uma tela de login ao abrir o app.
  - Implementar a autenticação via Firebase, retornando o perfil do usuário (nome, e-mail, e foto de perfil).
  - Após o login, o app deve armazenar a sessão do usuário e redirecionar para a tela de listagem de posts.
  - Implementar o logout, removendo a sessão do Firebase e redirecionando o usuário para a tela de login.

2. Tela de Listagem de Posts:

- **API:** Consumir a API pública de posts do JSONPlaceholder.
- **Tecnologia:** Utilizar Dio para realizar a requisição à API
- **Funcionalidade:**
  - Exibir uma lista dos posts retornados pela API, mostrando:
    - **Título** (exibir completo)
    - **Corpo**: (limitar a 100 caracteres com opção "Ver mais" se truncado)
  - Implementar para carregar 10 posts por vez utilizar um widget para indicar que está carregando.

3.Tela de Detalhes do Post:

- **Funcionalidade:**
  - Ao clicar em um post, o app deve redirecionar para uma página de detalhes do post
  - Exibir as seguintes informações:
    - **Título completo.**
    - **Corpo completo.**
    - **Autor do Post.**
  - Botão de "Voltar" para retornar à listagem.

4. Tela detalhe do Perfil:

- **Funcionalidade:**
  - Na tela de listagem de post ao clicar no avatar o usuário deve abrir detalhes do perfil**.**

    - Exibir as seguintes informações que podem ser salvas manualmente no Firestore:
        - **Imagem (mock)**
        - **Nome**
        - **Quantidade de Posts**
        - **Idade**
        - **Gostos**

5. Testes Automatizados:

- **Tecnologia:** Flutter Test, Mocktail. (Diferencial teste integraçao)
    - Testar o serviço de interação com Firebase Authentication e Firestore.
    - Testar os componentes de listagem e detalhes dos posts para garantir o carregamento correto dos dados.
    - Mockar Firebase e API externa para os testes unitários.
    - Diferencial teste integração ou Golden Test.

Entrega:

- Submeter o código via GitHub.
- Adicionar usuário e senha no README.
- Instruções para rodar os testes.
- Explicação da arquitetura usada.

Critérios de Avaliação:

1. Organização e Estrutura do Código

- Uso de boas práticas de organização de pastas e arquivos.
- Nomeação clara e consistente de classes, métodos e variáveis.
- Separação adequada entre camadas (ex.: `UI`, `Business Logic`, `Data Layer`).

2. Qualidade do Código

- Leitura e clareza do código.
- Uso de conceitos como SOLID, Clean Architecture.
- Redução de código desnecessário ou duplicado.

3. Implementação de Funcionalidades

- Funcionalidades entregues conforme os requisitos definidos.
- Correção e completude na implementação das features.
- Uso de widgets Flutter para criar layouts e interações de forma eficiente.

4. UI/UX

- Experiência do usuário fluida e responsiva.
- Uso adequado de widgets.

5. Gerenciamento de Estado

- Escolha do gerenciamento de estado , recomendado uso do **Bloc** , na escolha de outro justificar o uso.

- Implementação eficiente e escalável do gerenciamento de estado.
- Atualização de UI sincronizada com as mudanças no estado.

6. Conexão com APIs ou Banco de Dados

- Configuração e consumo eficiente de APIs.
- Uso do **Dio** para chamadas HTTP.
- Implementação de Firestore/SQLite ou outro banco de dados, se aplicável.

7. Tratamento de Erros

- Implementação de mensagens de erro claras para o usuário.
- Tratamento de exceções e falhas de rede de forma robusta.
- Logs e debug claros no código.

8. Testes

- Cobertura de testes unitários e/ou de integração (opcional, dependendo do desafio).
- Uso adequado de ferramentas como `flutter_test`.
- Implementação de testes automatizados para validar a lógica principal.

9. Uso de Recursos do Flutter

- Exploração de pacotes da comunidade e recursos nativos do Flutter.
- Implementação de animações, se relevante para o desafio.
- Uso eficiente de temas (`ThemeData`) e personalização de UI.

10. Documentação

- README claro com instruções de execução e justificativas de escolhas técnicas.
- Comentários no código explicando trechos complexos.
- Descrição de como expandir ou escalar a solução no futuro.