Seção 13: Desafios Práticos e Critérios de Avaliação

Desenvolvimento de Aplicativos Móveis - Flutter

Introdução

Chegou a hora de colocar em prática tudo que você aprendeu! Esta seção apresenta três desafios de complexidade crescente, cada um com requisitos claros e critérios objetivos de avaliação.

Os desafios são estruturados para:

- **Consolidar o aprendizado** através da prática
- **Simular cenários reais** de desenvolvimento
- **Permitir auto-avaliação** com critérios mensuráveis
- **Encorajar criatividade** dentro de requisitos técnicos

Cada desafio inclui:

- Lista de requisitos obrigatórios
- Requisitos opcionais (para pontuação extra)
- Rubrica de avaliação detalhada
- Dicas de implementação
- Checklist de entrega

Como usar esta seção:

1. **Para professores**: Use os desafios como projetos avaliativos, adaptando conforme necessário
2. **Para alunos**: Escolha o nível adequado ao seu conhecimento e trabalhe progressivamente
3. **Para auto-estudo**: Complete todos os três níveis para domínio completo

---

Estrutura dos Desafios

Níveis de Dificuldade

🟢 Nível Básico - "AuthApp Foundation"

- Foco: Fundamentos de autenticação

- Tempo estimado: 8-12 horas
- Pré-requisitos: Conhecimento básico de Flutter
- Pontuação máxima: 100 pontos

🟡 **Nível Intermediário - "AuthApp Plus"**

- Foco: Integração completa com backend
- Tempo estimado: 16-24 horas
- Pré-requisitos: Nível Básico concluído
- Pontuação máxima: 150 pontos

🔴 **Nível Avançado - "AuthApp Pro"**

- Foco: Funcionalidades avançadas e polimento
- Tempo estimado: 24-40 horas
- Pré-requisitos: Nível Intermediário concluído
- Pontuação máxima: 200 pontos

Desafio Nível Básico: AuthApp Foundation

Objetivo

Criar um aplicativo Flutter com autenticação básica funcional, focando nos fundamentos: cadastro, login, logout e persistência de sessão.

Requisitos Obrigatórios

1. Funcionalidades Core (40 pontos)

1.1. Tela de Login (10 pontos)

- [ ] Campos de email e senha
- [ ] Validação de campos (email válido, senha mínimo 6 caracteres)
- [ ] Botão de login funcional
- [ ] Link para tela de cadastro
- [ ] Link para recuperação de senha

1.2. Tela de Cadastro (10 pontos)

- [ ] Campos: nome, email, senha, confirmar senha

- [ ] Validação de todos os campos
- [ ] Verificação de correspondência de senhas
- [ ] Botão de cadastro funcional
- [ ] Link para voltar ao login

1.3. Tela Home/Principal (10 pontos)

- [ ] Mostra informações do usuário logado (nome, email)
- [ ] Botão de logout funcional
- [ ] Navegação protegida (só acessa se autenticado)

1.4. Autenticação com Supabase (10 pontos)

- [ ] Integração com Supabase configurada
- [ ] SignUp funcionando
- [ ] SignIn funcionando
- [ ] SignOut funcionando

2. Gerenciamento de Estado (20 pontos)

2.1. Provider/Riverpod (15 pontos)

- [ ] AuthNotifier implementado
- [ ] Estados modelados (loading, authenticated, unauthenticated, error)
- [ ] Telas consumindo estado corretamente
- [ ] Auth Wrapper implementado

2.2. Persistência de Sessão (5 pontos)

- [ ] Sessão persiste após fechar o app
- [ ] App verifica sessão ao iniciar
- [ ] Usuário não precisa fazer login novamente

3. Interface e UX (20 pontos)

3.1. Design Visual (10 pontos)

- [ ] Material Design 3 aplicado
- [ ] Tema consistente (cores, tipografia)
- [ ] Campos de formulário bem estilizados
- [ ] Botões seguem padrão estabelecido

3.2. Feedback e Loading (10 pontos)

- [ ] Indicadores de loading durante operações assíncronas
- [ ] Mensagens de erro claras e amigáveis

- [ ] Mensagens de sucesso quando apropriado
- [ ] Toggle de visibilidade de senha

4. Qualidade de Código (20 pontos)

4.1. Organização (10 pontos)

- [ ] Estrutura de pastas clara (screens, widgets, models, services)
- [ ] Separação de responsabilidades
- [ ] Nomenclatura consistente e descritiva

4.2. Boas Práticas (10 pontos)

- [ ] Controllers são descartados (dispose)
- [ ] Sem hardcoded strings (usar constantes)
- [ ] Código comentado em pontos-chave
- [ ] Sem warnings do Dart Analyzer

Requisitos Opcionais (Pontos Extra)

- [ ] **+5 pontos**: Tela de recuperação de senha funcional
- [ ] **+5 pontos**: Animações de transição entre telas
- [ ] **+5 pontos**: Tema dark mode funcional
- [ ] **+5 pontos**: Splash screen personalizada
- [ ] **+10 pontos**: Testes unitários básicos (auth service)

Pontuação máxima com extras: 130 pontos

Rubrica de Avaliação Detalhada

Funcionalidades Core (40 pontos)

Critério	Excelente (10)	Bom (7-8)	Satisfatório (5-6)	Insuficiente (0-4)
Telas implementadas	Todas as telas completas e funcionais	1-2 pequenos bugs	Telas incompletas ou com bugs significativos	Funcionalidades principais não funcionam
Validação	Validação robusta, mensagens claras	Validação básica funciona	Validação inconsistente	Sem validação
Integração Supabase	Todas operações funcionam perfeitamente	1 operação com problema	Múltiplas operações com problemas	Não integrado ou não funciona

Gerenciamento de Estado (20 pontos)

Critério	Excelente (15)	Bom (10-12)	Satisfatório (7-9)	Insuficiente (0-6)
AuthNotifier	Implementação completa, todos estados modelados	Estados principais implementados	Implementação básica	Não implementado ou muito incompleto
Persistência	Sessão persiste perfeitamente	Funciona com pequenos problemas	Inconsistente	Não funciona

Interface e UX (20 pontos)

Critério	Excelente (10)	Bom (7-8)	Satisfatório (5-6)	Insuficiente (0-4)
Design	Material Design 3, consistente, profissional	Bom visual, pequenas inconsistências	Visual básico mas funcional	Visual pobre ou inconsistente
Feedback	Loading, erros e sucessos bem implementados	Feedback básico funciona	Feedback inconsistente	Sem feedback adequado

Qualidade de Código (20 pontos)

Critério	Excelente (10)	Bom (7-8)	Satisfatório (5-6)	Insuficiente (0-4)
Organização	Estrutura exemplar, fácil de navegar	Bem organizado, pequenas melhorias possíveis	Organização básica	Desorganizado
Boas práticas	Código limpo, sem warnings, bem comentado	Poucas melhorias necessárias	Várias melhorias necessárias	Muitos problemas

Dicas de Implementação

1. Comece pelo básico:

```dart
// 1. Configure o Supabase no main.dart
// 2. Crie o AuthService com signUp, signIn, signOut
// 3. Implemente AuthNotifier com estados básicos
// 4. Crie telas sem estilo primeiro (foco na funcionalidade)
// 5. Adicione estilo e polimento depois
```

2. Teste incrementalmente:

- Teste cada funcionalidade antes de passar para a próxima
- Use `print()` statements para debugar
- Teste cenários de erro (email inválido, senha errada, etc)

3. Use os componentes da Seção 12:

- Copie CustomTextField, AuthButton, etc
- Adapte conforme necessário
- Reutilize código sempre que possível

4. Gerenciamento de estado simplificado:

```dart
// Não complique! Estados básicos suficientes:
// - AuthStateInitial (verificando sessão)
// - AuthStateLoading (operação em andamento)
// - AuthStateAuthenticated (usuário logado)
// - AuthStateUnauthenticated (usuário não logado)
// - AuthStateError (erro ocorreu)
```

Checklist de Entrega

Antes de submeter, verifique:

Funcionalidade:

- [ ] Posso criar uma conta nova
- [ ] Posso fazer login com a conta criada
- [ ] Posso fazer logout
- [ ] Sessão persiste (fechar e abrir app mantém login)
- [ ] Validações impedem dados inválidos

Interface:

- [ ] App não trava ou dá erro fatal
- [ ] Campos têm labels claros
- [ ] Botões indicam loading quando necessário
- [ ] Mensagens de erro são claras

Código:

- [ ] Projeto compila sem erros
- [ ] Não há warnings críticos

- [ ] Código está organizado em pastas
- [ ] README.md explica como rodar o projeto

Documentação mínima (README.md):

```markdown
# AuthApp Foundation

## Como Rodar

1. Clone o repositório
2. Execute `flutter pub get`
3. Configure suas credenciais Supabase em `lib/config/supabase_config.dart`
4. Execute `flutter run`

## Credenciais Supabase

- URL: [sua URL]
- Anon Key: [sua key]

## Funcionalidades Implementadas

- [x] Login
- [x] Cadastro
- [x] Logout
- [x] Persistência de sessão
```

Desafio Nível Intermediário: AuthApp Plus

Objetivo

Expandir o app básico com recuperação de senha, login social, validações avançadas e melhorias de UX.

Requisitos Obrigatórios

1. Funcionalidades Estendidas (50 pontos)

1.1. Recuperação de Senha (15 pontos)

- [ ] Tela de solicitação de recuperação (email)
- [ ] Envio de email funcional via Supabase
- [ ] Tela de redefinição de senha (após clicar no link)
- [ ] Deep linking configurado (Android + iOS)

- [ ] Fluxo completo testado e funcional

1.2. Login Social (20 pontos)

- [ ] Login com Google configurado e funcional
- [ ] Login com GitHub configurado e funcional
- [ ] OAuth redirect URLs configurados
- [ ] Tratamento de callback do OAuth
- [ ] Botões de social login na tela de login

1.3. Magic Link (15 pontos)

- [ ] Tela de magic link implementada
- [ ] Envio de email funcional
- [ ] Deep linking funciona para magic link
- [ ] Sessão estabelecida após clicar no link

2. Validações Avançadas (20 pontos)

2.1. Validação de Senha Forte (10 pontos)

- [ ] Indicador visual de força de senha
- [ ] Requisitos claros (minúscula, maiúscula, número, especial)
- [ ] Feedback em tempo real
- [ ] Impede cadastro com senha fraca

2.2. Validação Assíncrona (10 pontos)

- [ ] Verificação se email já existe (durante cadastro)
- [ ] Debouncing implementado (não valida a cada letra)
- [ ] Feedback visual durante validação
- [ ] Mensagens claras

3. Melhorias de UX (30 pontos)

3.1. Estados e Transições (15 pontos)

- [ ] Splash screen com verificação de sessão
- [ ] Transições suaves entre telas
- [ ] Animação de shake em campos com erro
- [ ] Loading states em todos os botões

3.2. Tratamento de Erros (15 pontos)

- [ ] Erros de rede tratados graciosamente
- [ ] Mensagens de erro específicas (não genéricas)

- [ ] Retry automático quando apropriado
- [ ] Timeout configurado

4. Perfil de Usuário (30 pontos)

4.1. Tela de Perfil (20 pontos)

- [ ] Mostra informações do usuário
- [ ] Permite editar nome
- [ ] Permite alterar senha
- [ ] Atualização funcional via Supabase

4.2. Avatar (10 pontos)

- [ ] Mostra avatar do usuário (letra inicial ou imagem)
- [ ] Permite escolher avatar (picker de imagem)
- [ ] Upload para Supabase Storage (ou URL externa)

5. Qualidade e Testes (20 pontos)

5.1. Testes (10 pontos)

- [ ] Testes unitários para AuthService
- [ ] Testes unitários para validators
- [ ] Testes de widget para componentes principais
- [ ] Cobertura mínima de 50%

5.2. Documentação (10 pontos)

- [ ] README completo com instruções
- [ ] Comentários em código complexo
- [ ] Documentação de configuração (OAuth, deep linking)
- [ ] Screenshots no README

Requisitos Opcionais (Pontos Extra)

- [ ] **+10 pontos**: Autenticação biométrica implementada
- [ ] **+10 pontos**: Modo offline (cache de sessão)
- [ ] **+10 pontos**: Internacionalização (PT + EN)
- [ ] **+10 pontos**: Tema customizável (usuário escolhe)
- [ ] **+15 pontos**: CI/CD configurado (GitHub Actions)
- [ ] **+5 pontos**: Análise de código (linter rigoroso)

Pontuação máxima com extras: 210 pontos

Rubrica de Avaliação Detalhada

Funcionalidades Estendidas (50 pontos)

Critério	Excelente	Bom	Satisfatório	Insuficiente
Recuperação senha	Fluxo completo, deep linking perfeito (15pts)	Funciona com pequenos bugs (10-12pts)	Funcionalidade básica (7-9pts)	Não funciona (0-6pts)
Login social	Ambos funcionam perfeitamente (20pts)	Um funciona perfeitamente (14-16pts)	Ambos com problemas (10-13pts)	Não implementado (0-9pts)
Magic link	Funcional e bem implementado (15pts)	Funciona com problemas menores (10-12pts)	Implementação básica (7-9pts)	Não funciona (0-6pts)

Validações Avançadas (20 pontos)

Critério	Excelente	Bom	Satisfatório	Insuficiente
Força senha	Indicador bonito, feedback claro (10pts)	Funcional mas básico (7-8pts)	Implementação simples (5-6pts)	Não implementado (0-4pts)
Validação assíncrona	Debouncing, UX excelente (10pts)	Funciona bem (7-8pts)	Básica (5-6pts)	Não implementado (0-4pts)

Melhorias de UX (30 pontos)

Critério	Excelente	Bom	Satisfatório	Insuficiente
Estados e transições	Polido, animações suaves (15pts)	Boas transições (10-12pts)	Básico (7-9pts)	Pobre (0-6pts)
Tratamento erros	Robusto, mensagens claras (15pts)	Bom tratamento (10-12pts)	Básico (7-9pts)	Pobre (0-6pts)

Perfil de Usuário (30 pontos)

Critério	Excelente	Bom	Satisfatório	Insuficiente
Tela perfil	Completa e funcional (20pts)	Funcional, melhorias possíveis (14-16pts)	Básica (10-13pts)	Incompleta (0-9pts)
Avatar	Upload funciona perfeitamente (10pts)	Funciona com problemas (7-8pts)	Muito básico (5-6pts)	Não funciona (0-4pts)

Qualidade e Testes (20 pontos)

Critério	Excelente	Bom	Satisfatório	Insuficiente
Testes	Cobertura >70%, bem escritos (10pts)	Cobertura 50-70% (7-8pts)	Alguns testes (5-6pts)	Poucos ou nenhum (0-4pts)
Documentação	Completa, clara, screenshots (10pts)	Boa documentação (7-8pts)	Básica (5-6pts)	Pobre (0-4pts)

Dicas de Implementação

1. Deep Linking:

```dart
// Teste deep links com ADB (Android):
// adb shell am start -a android.intent.action.VIEW \
//   -d "io.supabase.seuapp://reset-password"

// No iOS, use xcrun simctl:
// xcrun simctl openurl booted "io.supabase.seuapp://reset-password"
```

2. OAuth Configuration:

```dart
// Google OAuth:
// 1. Configure no Google Cloud Console
// 2. Adicione redirect URI: https://seu-projeto.supabase.co/auth/v1/callback
// 3. Configure no Supabase Dashboard: Authentication > Providers > Google
// 4. Adicione Client ID e Secret

// GitHub OAuth:
// 1. Configure no GitHub Developer Settings
// 2. Mesma redirect URI do Google
// 3. Configure no Supabase Dashboard
```

3. Validação Assíncrona:

```dart
```

```dart
// Use Timer para debouncing
Timer? _debounce;

void _onEmailChanged(String value) {
  if (_debounce?.isActive ?? false) _debounce!.cancel();

  _debounce = Timer(const Duration(milliseconds: 500), () async {
    // Fazer validação assíncrona aqui
    final exists = await checkEmailExists(value);
    // Atualizar UI
  });
}
```

4. Testes:

dart

```dart
// Exemplo de teste unitário para AuthService
test('signIn with valid credentials should return user', () async {
  final authService = AuthService();

  final result = await authService.signIn(
    email: 'test@example.com',
    password: 'password123',
  );

  expect(result.user, isNotNull);
  expect(result.user!.email, 'test@example.com');
});
```

Checklist de Entrega

Além do checklist do Nível Básico:

Funcionalidades Estendidas:

- [ ] Recuperação de senha funciona end-to-end
- [ ] Login com Google funciona
- [ ] Login com GitHub funciona
- [ ] Magic link funciona
- [ ] Deep linking configurado e testado

Validações:

- [ ] Indicador de força de senha funcional
- [ ] Validação de email duplicado funciona

- [ ] Todas validações têm feedback visual

Perfil:

- [ ] Posso editar meu nome
- [ ] Posso alterar minha senha
- [ ] Posso adicionar/atualizar avatar

Qualidade:

- [ ] Testes passam (`flutter test`)
- [ ] Cobertura >= 50%
- [ ] README com screenshots
- [ ] Configuração documentada

Desafio Nível Avançado: AuthApp Pro

Objetivo

Criar uma experiência de autenticação de nível profissional com recursos avançados, segurança reforçada e polimento máximo.

Requisitos Obrigatórios

1. Segurança Avançada (40 pontos)

1.1. Autenticação Multifator - 2FA (20 pontos)

- [ ] Configuração de 2FA na tela de perfil
- [ ] Geração de QR code para app autenticador
- [ ] Validação de código TOTP funcional
- [ ] Códigos de backup gerados
- [ ] Fluxo de login com 2FA

1.2. Gerenciamento de Sessões (20 pontos)

- [ ] Lista de dispositivos/sessões ativas
- [ ] Mostra informações (dispositivo, localização aproximada, última atividade)
- [ ] Permite revogar sessões específicas
- [ ] Logout de todos os dispositivos
- [ ] Notificação de novo login

2. Funcionalidades Pro (40 pontos)

2.1. Histórico de Atividades (10 pontos)

- [ ] Log de ações importantes (login, troca senha, etc)
- [ ] Timestamp de cada atividade
- [ ] Filtragem por tipo de atividade
- [ ] Persistido no Supabase

2.2. Modo Offline (15 pontos)

- [ ] Cache de dados do perfil
- [ ] Operações offline enfileiradas
- [ ] Sincronização ao reconectar
- [ ] Indicador de status de conexão
- [ ] Tratamento de conflitos

2.3. Verificação de Email Aprimorada (15 pontos)

- [ ] Reenvio de email de confirmação
- [ ] Status de verificação visível
- [ ] Banner persistente se não verificado
- [ ] Deep link de confirmação funciona
- [ ] Atualização de email com re-verificação

3. UX/UI Profissional (40 pontos)

3.1. Onboarding (15 pontos)

- [ ] 3-4 telas explicando o app
- [ ] Animações e transições suaves
- [ ] Skip option
- [ ] Indicador de progresso
- [ ] Mostra apenas na primeira vez

3.2. Animações Avançadas (10 pontos)

- [ ] Hero animations entre telas
- [ ] Micro-interações em botões e campos
- [ ] Loading skeletons em vez de spinners
- [ ] Transições de página customizadas

3.3. Acessibilidade (15 pontos)

- [ ] Suporte a leitores de tela
- [ ] Semântica apropriada em todos widgets
- [ ] Contraste adequado (WCAG AA)
- [ ] Tamanhos de toque >= 48px

- [ ] Testado com TalkBack/VoiceOver

4. Arquitetura e Qualidade (40 pontos)

4.1. Arquitetura Limpa (20 pontos)

- [ ] Clean Architecture implementada (domain, data, presentation)
- [ ] Use cases isolados
- [ ] Repositories com interfaces
- [ ] Injeção de dependências
- [ ] Separação clara de camadas

4.2. Testes Abrangentes (20 pontos)

- [ ] Testes unitários (>80% cobertura)
- [ ] Testes de integração
- [ ] Testes de widget
- [ ] Testes E2E (pelo menos login flow)
- [ ] Mocks para Supabase

5. DevOps e Documentação (40 pontos)

5.1. CI/CD (20 pontos)

- [ ] GitHub Actions configurado
- [ ] Build automático em PRs
- [ ] Testes automáticos
- [ ] Lint e análise de código
- [ ] Deploy automático (Firebase App Distribution ou TestFlight)

5.2. Documentação Profissional (20 pontos)

- [ ] README completo e profissional
- [ ] Documentação de arquitetura (diagramas)
- [ ] API docs geradas (dartdoc)
- [ ] Contributing guidelines
- [ ] Changelog mantido

Requisitos Opcionais (Pontos Extra)

- [ ] **+20 pontos**: PWA (Progressive Web App) funcional
- [ ] **+15 pontos**: Rate limiting client-side
- [ ] **+15 pontos**: Captcha em login/cadastro
- [ ] **+10 pontos**: Export de dados do usuário (GDPR)
- [ ] **+10 pontos**: Deletar conta com confirmação

- [ ] **+10 pontos**: Telemetria e analytics (Sentry, Firebase)
- [ ] **+10 pontos**: Feature flags
- [ ] **+5 pontos**: Monorepo (packages separados)

Pontuação máxima com extras: 295 pontos

Rubrica de Avaliação Detalhada

Segurança Avançada (40 pontos)

Critério	Excelente	Bom	Satisfatório	Insuficiente
2FA	Implementação completa e segura (20pts)	Funcional com pequenos problemas (14-16pts)	Implementação básica (10-13pts)	Não funciona (0-9pts)
Gerenciamento sessões	Todas features, UX excelente (20pts)	Funcional, pode melhorar (14-16pts)	Básico (10-13pts)	Incompleto (0-9pts)

Funcionalidades Pro (40 pontos)

Critério	Excelente	Bom	Satisfatório	Insuficiente
Histórico	Completo, filtros, UX ótima (10pts)	Funcional (7-8pts)	Básico (5-6pts)	Pobre (0-4pts)
Modo offline	Robusto, sincronização perfeita (15pts)	Funciona bem (10-12pts)	Básico (7-9pts)	Não funciona (0-6pts)
Verificação email	Todas features, polido (15pts)	Funcional (10-12pts)	Básico (7-9pts)	Incompleto (0-6pts)

UX/UI Profissional (40 pontos)

Critério	Excelente	Bom	Satisfatório	Insuficiente
Onboarding	Polido, animações lindas (15pts)	Bom onboarding (10-12pts)	Básico (7-9pts)	Pobre (0-6pts)
Animações	Múltiplas animações suaves (10pts)	Algumas animações (7-8pts)	Animações básicas (5-6pts)	Sem animações (0-4pts)
Acessibilidade	Totalmente acessível, testado (15pts)	Boa acessibilidade (10-12pts)	Parcial (7-9pts)	Pobre (0-6pts)

Arquitetura e Qualidade (40 pontos)

Critério	Excelente	Bom	Satisfatório	Insuficiente
Clean Architecture	Implementação exemplar (20pts)	Boa arquitetura (14-16pts)	Arquitetura básica (10-13pts)	Pobre (0-9pts)
Testes	Cobertura >80%, E2E (20pts)	Cobertura 60-80% (14-16pts)	Cobertura 40-60% (10-13pts)	<40% (0-9pts)

DevOps e Documentação (40 pontos)

Critério	Excelente	Bom	Satisfatório	Insuficiente
CI/CD	Pipeline completo, automático (20pts)	Pipeline funcional (14-16pts)	Básico (10-13pts)	Não configurado (0-9pts)
Documentação	Profissional, completa (20pts)	Boa documentação (14-16pts)	Documentação básica (10-13pts)	Pobre (0-9pts)

Dicas de Implementação

1. 2FA com TOTP:

dart

```dart
// Use o pacote otp para gerar códigos TOTP
dependencies:
  otp: ^3.1.4

// Gerar secret:
final secret = OTP.randomSecret();

// Gerar QR code URL:
final otpauth = 'otpauth://totp/AuthApp:user@email.com?secret=$secret&issuer=AuthApp';

// Validar código:
bool isValid = OTP.generateTOTPCodeString(
  secret,
  DateTime.now().millisecondsSinceEpoch,
) == userCode;
```

2. Clean Architecture:

```
lib/
├── core/              # Utilities, constants
├── domain/            # Entities, repositories (interfaces), use cases
│   ├── entities/
│   ├── repositories/
│   └── usecases/
├── data/              # Repository implementations, data sources
│   ├── datasources/
│   ├── models/
│   └── repositories/
└── presentation/      # UI, state management
    ├── pages/
    ├── widgets/
    └── providers/
```

3. Testes E2E:

dart

```dart
// Use integration_test package
testWidgets('Complete login flow', (tester) async {
  app.main();
  await tester.pumpAndSettle();

  // Preencher email
  await tester.enterText(
    find.byKey(const Key('email_field')),
    'test@example.com',
  );

  // Preencher senha
  await tester.enterText(
    find.byKey(const Key('password_field')),
    'password123',
  );

  // Clicar em login
  await tester.tap(find.byKey(const Key('login_button')));
  await tester.pumpAndSettle();

  // Verificar que chegou na home
  expect(find.text('Bem-vindo'), findsOneWidget);
});
```

4. CI/CD GitHub Actions:

yaml

```yaml
# .github/workflows/flutter.yml
name: Flutter CI

on: [push, pull_request]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v3
    - uses: subosito/flutter-action@v2
      with:
        flutter-version: '3.19.0'

    - name: Install dependencies
      run: flutter pub get

    - name: Analyze
      run: flutter analyze

    - name: Run tests
      run: flutter test --coverage

    - name: Upload coverage
      uses: codecov/codecov-action@v3

    - name: Build APK
      run: flutter build apk --release
```

Checklist de Entrega

Além dos checklists anteriores:

Segurança:

- [ ] 2FA funciona completamente
- [ ] Posso ver e revogar sessões
- [ ] Notificações de segurança funcionam

Funcionalidades Pro:

- [ ] Histórico registra atividades
- [ ] App funciona offline

- [ ] Verificação de email aprimorada

UX/UI:

- [ ] Onboarding na primeira vez
- [ ] Animações suaves em transições
- [ ] Testado com leitor de tela
- [ ] Contraste adequado

Arquitetura:

- [ ] Clean Architecture implementada
- [ ] Camadas bem separadas
- [ ] Use cases isolados

Testes:

- [ ] Cobertura >= 80%
- [ ] Testes E2E passam
- [ ] Todos testes passam no CI

DevOps:

- [ ] CI/CD funciona
- [ ] Build automático
- [ ] Deploy automático (opcional)

Documentação:

- [ ] README profissional
- [ ] Diagramas de arquitetura
- [ ] API docs geradas
- [ ] Changelog atualizado

Formato de Entrega

Estrutura do Repositório

```
meu-auth-app/
├── .github/
│   └── workflows/
│       └── flutter.yml
├── lib/
│   ├── core/
```

```
│   ├── domain/
│   ├── data/
│   └── presentation/
├── test/
├── integration_test/
├── assets/
├── docs/
│   ├── architecture.md
│   └── screenshots/
├── README.md
├── CHANGELOG.md
├── CONTRIBUTING.md
└── pubspec.yaml
```

README Template

markdown

[Nome do App] - Sistema de Autenticação Flutter

🎯 Objetivo

[Breve descrição do projeto e nível do desafio]

✨ Funcionalidades

Implementadas
- ✅ Cadastro de usuário
- ✅ Login com email/senha
- ✅ Login social (Google, GitHub)
- ✅ Recuperação de senha
- ✅ Magic links
- ✅ Perfil de usuário
- ✅ 2FA (apenas nível avançado)
- ✅ [outras funcionalidades]

Opcionais Implementadas
- ✅ [lista de opcionais]

🏗️ Arquitetura

[Diagrama ou descrição da arquitetura]

🚀 Como Rodar

Pré-requisitos
- Flutter >= 3.19.0
- Conta Supabase
- [outros pré-requisitos]

Instalação

1. Clone o repositório:
```bash
git clone https://github.com/usuario/meu-auth-app.git
cd meu-auth-app
```

2. Instale dependências:
```bash
flutter pub get
```

3. Configure Supabase:
   - Crie um projeto em supabase.com
   - Copie `.env.example` para `.env`
   - Preencha com suas credenciais

4. Execute:
```bash
flutter run
```

🧪 Testes
```bash
# Testes unitários
flutter test

# Testes com cobertura
flutter test --coverage

# Testes de integração
flutter test integration_test
```

📸 Screenshots

[Capturas de tela do app]

🛠️ Tecnologias

- Flutter 3.19.0
- Dart 3.3.0
- Supabase
- Riverpod (gerenciamento de estado)
- [outras bibliotecas]

📝 Licença

[Sua licença]

👤 Autor

[Seu nome e contato]

Critérios de Apresentação (Opcional)

Se houver apresentação oral:

Estrutura sugerida (10-15 minutos):

1. **Introdução (2min)**
    - Nome do projeto
    - Nível do desafio
    - Objetivo
2. **Demo (5-7min)**
    - Fluxo principal (cadastro → login → home)
    - Funcionalidades destaque
    - Interações ao vivo
3. **Arquitetura (2-3min)**
    - Diagrama de arquitetura
    - Decisões técnicas importantes
    - Padrões utilizados
4. **Desafios e Aprendizados (2-3min)**
    - Dificuldades encontradas
    - Como foram resolvidas
    - Principais aprendizados
5. **Q&A (tempo variável)**

Dicas para apresentação:

- Teste seu demo antes (gravação de backup é boa ideia)
- Prepare ambiente de desenvolvimento limpo
- Tenha exemplos de código-chave prontos para mostrar
- Seja honesto sobre o que não funcionou ou ficou pendente

Auto-Avaliação

Checklist de Auto-Avaliação

Use esta checklist para avaliar seu próprio trabalho antes de submeter:

Nível Básico:

Funcionalidades: [ ] /40 pontos
- Tela login funcional
- Tela cadastro funcional
- Tela home protegida
- Integração Supabase

Estado: [ ] /20 pontos
- AuthNotifier implementado
- Persistência funciona

Interface: [ ] /20 pontos
- Material Design aplicado
- Feedback visual

Código: [ ] /20 pontos
- Bem organizado
- Sem warnings críticos

TOTAL: [ ] /100 pontos

Nível Intermediário:

[Adicione as seções do intermediário]

TOTAL: [ ] /150 pontos

Nível Avançado:

[Adicione as seções do avançado]

TOTAL: [ ] /200 pontos

Perguntas Reflexivas

Após completar o desafio, reflita:

1. **O que funcionou bem?** Liste 3-5 aspectos do seu projeto que você considera bem-feitos.
2. **O que foi mais desafiador?** Identifique os principais obstáculos e como os superou.
3. **O que faria diferente?** Se pudesse recomeçar, o que mudaria na abordagem?
4. **Próximos passos:** Como você expandiria ou melhoraria este projeto?
5. **Aprendizados técnicos:** Quais conceitos técnicos você dominou melhor através deste projeto?
6. **Aprendizados pessoais:** O que aprendeu sobre seu próprio processo de desenvolvimento?

Recursos Adicionais

Bibliotecas Úteis

Autenticação e Segurança:

- `supabase_flutter` - Cliente Supabase oficial
- `flutter_secure_storage` - Armazenamento seguro
- `local_auth` - Biometria
- `otp` - TOTP para 2FA

UI/UX:

- `google_fonts` - Fontes Google
- `flutter_svg` - Suporte SVG
- `cached_network_image` - Cache de imagens
- `shimmer` - Loading skeletons
- `lottie` - Animações Lottie

Utilitários:

- `connectivity_plus` - Status de conexão
- `image_picker` - Selecionar imagens
- `path_provider` - Diretórios do sistema
- `share_plus` - Compartilhar conteúdo

Testes:

- `mockito` - Mocking
- `integration_test` - Testes E2E
- `golden_toolkit` - Golden tests

DevOps:

- `sentry_flutter` - Error tracking
- `firebase_analytics` - Analytics

- `flutter_launcher_icons` - Ícones do app
- `flutter_native_splash` - Splash screen

Referências

Documentação Oficial:

- Flutter Documentation
- Supabase Flutter Docs
- Riverpod Documentation
- Material Design 3

Tutoriais e Exemplos:

- Flutter Auth Examples
- Clean Architecture Flutter
- Flutter Testing

Comunidades:

- Flutter Discord
- r/FlutterDev
- Stack Overflow - Flutter

Conclusão da Seção

Você agora tem três desafios progressivos para consolidar todo o aprendizado sobre autenticação em Flutter. Cada desafio foi cuidadosamente estruturado para:

✅ **Ensinar através da prática** - Aplicação real dos conceitos
✅ **Avaliar objetivamente** - Critérios claros e mensuráveis
✅ **Incentivar excelência** - Requisitos opcionais para ir além
✅ **Desenvolver profissionalmente** - Práticas usadas na indústria

Próximos passos:

1. Escolha o nível adequado ao seu conhecimento atual
2. Leia todos os requisitos antes de começar
3. Planeje sua abordagem (não pule direto para o código)

4. Implemente incrementalmente (funcionalidade por funcionalidade)
5. Teste frequentemente
6. Documente conforme desenvolve
7. Use a auto-avaliação para validar seu trabalho

Boa sorte, e lembre-se: o objetivo não é apenas completar o desafio, mas dominar os conceitos através da prática deliberada!

Na próxima e última seção, você encontrará um guia de troubleshooting com os erros mais comuns e como resolvê-los.

Fim da Seção 13: Desafios Práticos e Critérios de Avaliação