Flutter Avançado - Aula 5

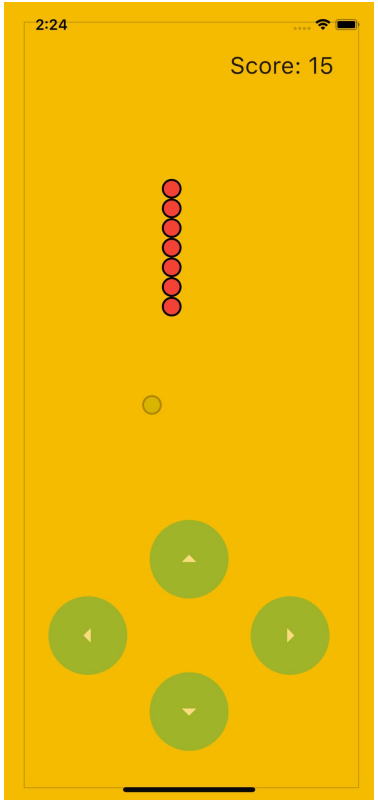
Exercícios 14 a 16

Fernando Duwe

v 1.1

Dicas de hoje

- Artigo: How to Create a 2D Snake Game in Flutter [Ver mais]



Garantia de qualidade

Ver mais

- Conjunto de procedimentos para prevenir erros/falhas nos produtos/serviços
- Diferente de controle de qualidade
- QA (*Quality Assurance*): Prevenção de problemas nos processos de desenvolvimento, através de atividades planejadas e sistemáticas
- QC (*Quality Control*): Atividades ou técnicas utilizadas para capturar problemas, tratar e manter a qualidade do produto

Testes unitários

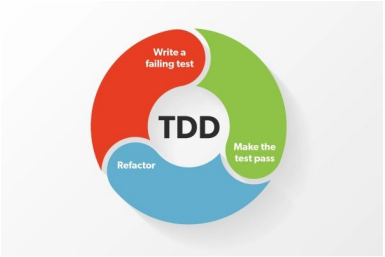
Ver mais

- Testes que verificam se uma parte específica do código, costumeiramente a nível de função, está funcionando corretamente
- Sozinho, o teste unitário não pode verificar a funcionalidade de uma parte do software

TDD

Ver mais

- Baseado em testes unitários
- Nesse modelo, os testes unitários são escritos pelo desenvolvedor antes mesmo de a funcionalidade ser implementada
- Dessa forma, o desenvolvedor pode se familiarizar melhor o que deve ser implementado, além de criar um legado de testes unitários



- Para criarmos testes unitários em Flutter, vamos utilizar a biblioteca flutter_test
- Essa biblioteca pode ser adicionada no *pub_spec.yaml*, podendo ser adicionada nas *dev_dependencies*
- Após a dependência ser adicionada ao nosso projeto, vamos criar um arquivo de testes

Um teste básico

```
import 'package:counter_app/counter.dart';
import 'package:test/test.dart';

void main() {
  test('Counter value should be incremented', () {
    final counter = Counter();

    counter.increment();

    expect(counter.value, 1);
  });
}
```

Um teste negativo

```
import 'package:counter_app/counter.dart';
import 'package:test/test.dart';

void main() {
  test('Counter value should be incremented', () {
    final counter = Counter();

    counter.increment();

    expect(counter.value != 5, true);
  });
}
```

- Vamos criar uma classe que calcule as médias de um aluno
- Crie uma função que receba uma lista de notas e retorne a média
- Escreva os seguintes testes:

    1. Testar se a função retorna 0 se você passa uma lista em branco de notas
    2. Testar o arredondamento das notas
    3. Testar também com um teste negativo

- Com os testes unitários, nós conseguimos testar pequenas porções do nosso código
- O flutter também disponibiliza funções para testarmos o comportamento de um widget
- Para isso, vamos utilizar também o pacote *flutter_test*

Criando um widget dentro de um teste

```
import 'package:flutter_test/flutter_test.dart';
import 'package:projeto_aula_avancado/models/questionModel.dart';
import 'package:projeto_aula_avancado/screens/millionShow.dart';

void main () {
  testWidgets("Teste 1", (widgetTester) async {
    await widgetTester.pumpWidget(TestScreen());
  });
}
```

Ver mais

Buscando widgets, usando o Finder

Com o Finder, podemos procurar widgets a partir de suas características, como texto, tipo, ícones, etc. Podemos a partir do resultado do Finder, verificar se um ou mais widgets foram encontrados. Isso serve para sabermos se o widget está sendo renderizado conforme o esperado.

```
final vrTextFinder = find.text(“Teste”);
```

Verificando o resultado da busca com o finder

Após buscarmos um ou vários widgets, podemos agora verificar os resultados encontrados, usando o Matcher.

```
expect(vrTextFinder, findsOneWidget);
```

O findsOneWidget testa se o Finder encontrou somente um widget conforme a pesquisa informada. Na próxima página veremos mais opções que temos com o Matcher.

Ver mais

- **findsNothing:** Nenhum widget encontrado com o filtro informado
- **findsOneWidget:** Somente um widget encontrado
- **findsWidgets:** Um ou mais widgets foram encontrados
- **findsNWidgets:** Um número exato de widgets foram encontrados (n)
- **matchesGoldenFile:** Verifica se o widget encontrado é renderizado igualmente a uma imagem bitmap em específico

Efetuando um scroll em uma lista

```
final listFinder = find.byType(Scrollable);
final itemFinder = find.byKey(const ValueKey('item_50_text'));

// Scroll until the item to be found appears.
await tester.scrollUntilVisible(
  itemFinder,
  500.0,
  scrollable: listFinder,
);

// Verify that the item contains the correct text.
expect(itemFinder, findsOneWidget);
```

Escrevendo texto em uma caixa de texto

No exemplo abaixo, escrevemos o texto hi, no widget do tipo TextField.

```
await tester.enterText(find.byType(TextField), 'hi');
```

Arrastando um widget

No exemplo abaixo, arrastamos um *dismissible*. A função *pumpAndSettle* irá atualizar o *widget* e esperar a animação do dismissible finalizar sua execução. Após atualizar o *widget*, verificará se o item “hi” ainda existe em tela.

```
await tester.drag(find.byType(Dismissible), const Offset(500.0, 0.0));

// Build the widget until the dismiss animation ends.
await tester.pumpAndSettle();

// Ensure that the item is no longer on screen.
expect(find.text('hi'), findsNothing);
```

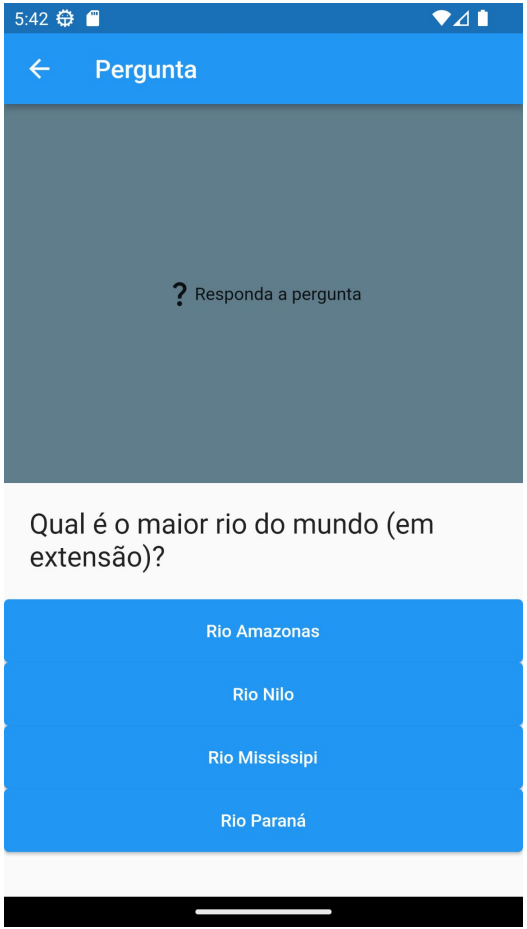
Pressionando botões

```
// Procurando um botão a partir do texto e clicando
await widgetTester.tap(find.text(“Login”));

// Redesenhando a tela após o clique do botão
await widgetTester.pump();

// Verificando se encontrou o texto esperado
expect(find.text("Acesso Liberado, bem vindo!"), findsOneWidget);
```

- Vamos criar um widget do show do milhão
- Para isso, crie um *model* da pergunta e suas possíveis respostas
- Crie uma tela para responder a pergunta
- Na próxima página, o comportamento esperado da tela

Exercício 15 - cont

Ver mais

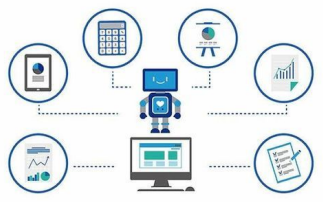


Testes para implementar

- Primeiro teste: Verificar se todas as perguntas foram renderizadas. Verificar se a quantidade correta de botões foi criada, conforme a quantidade de opções
- Segundo teste: Verificar se existe um texto com a pergunta em tela. Verificar se existe um texto com o mesmo texto da pergunta
- Terceiro teste: Testar a resposta da pergunta. O teste deve responder a pergunta correta e verificar se a tela se comporta corretamente

Ver mais

- Os testes de integração (ou E2E) testam toda a aplicação
- E2E = End to End
- Ao executar o teste, o emulador será chamado, executando a aplicação conforme o teste escrito
- Os testes escritos se assemelham muito aos testes de widgets
- Uma boa prática é escrever os códigos em uma nova pasta: *integration_test*
- Para os testes de integração, utilizaremos a biblioteca *integration_test*



Testes de Integração - cont

Ver mais

```
import 'package:flutter/cupertino.dart';
import 'package:flutter_test/flutter_test.dart';
import 'package:integration_test/integration_test.dart';
import 'package:admin_test/main.dart' as app;

void main() {
  IntegrationTestWidgetsFlutterBinding.ensureInitialized();

  group("Teste do App de IMC", () {
    testWidgets("Abrir, preencher form e clicar em Ok", (widgetTester) async {
      app.main();

      await widgetTester.pumpAndSettle();

      await widgetTester.enterText(
          find.byKey(ValueKey<String>("form_input_user")), "admin");

      await Future.delayed(Duration(seconds: 1));

      await widgetTester
          .tap(find.byKey(ValueKey<String>("form_button_next")));

      await widgetTester.pumpAndSettle();
    });
```

Pego meu projeto e encapsulo no *alias* app

```
import 'package:imc/main.dart' as app;
```

Aguardando o início do app, verificando se todos os recursos já estão inicializados

```
IntegrationTestWidgetsFlutterBinding.ensureInitialized();
```

Em cada teste, inicializo o meu aplicativo

```
app.main();
```

Executando os testes em outras plataformas

Para executar o teste em uma plataforma diferente, utilize o comando abaixo:

```
flutter test integration_test/foo_test.dart -d <DEVICE_ID>
```

Para saber o ID do device que você deseja testar, utilize o comando

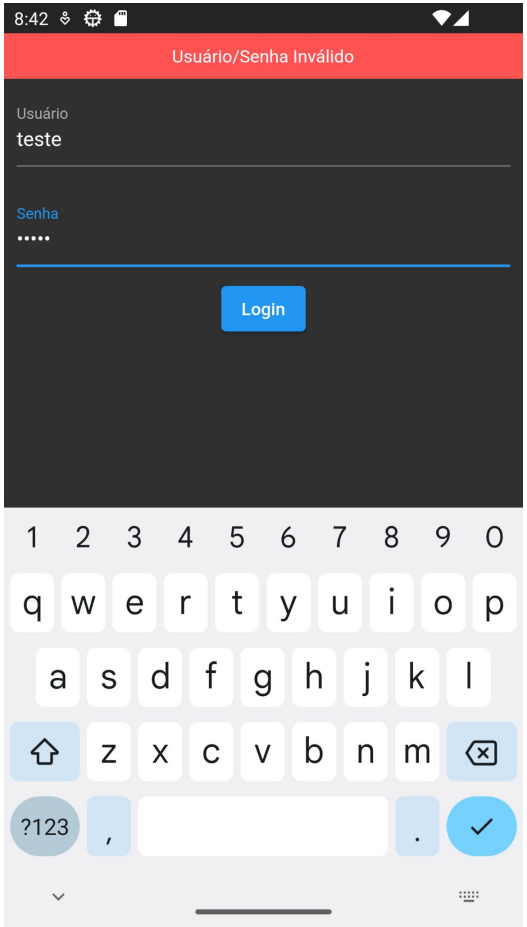
```
flutter devices
```

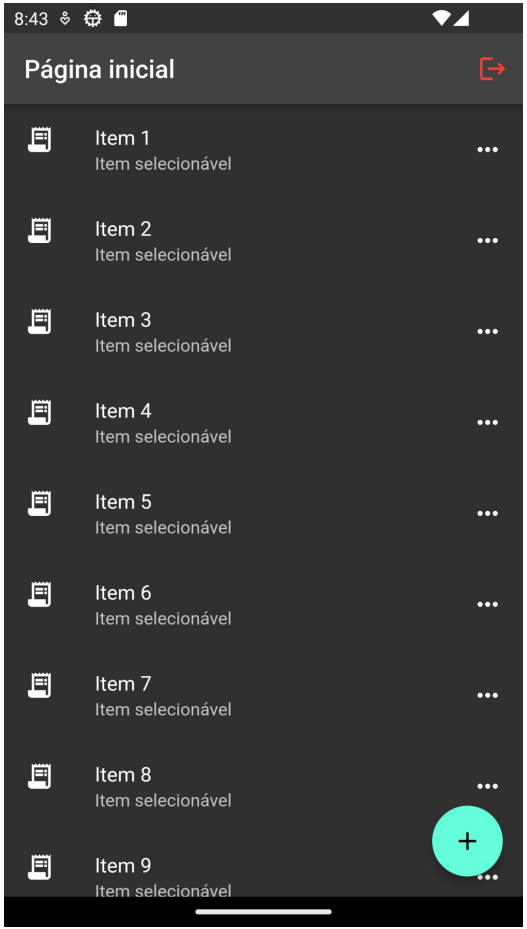
- Vamos criar uma área de administração
- Ao abrir o app, deve ser exibido um formulário de login
- Caso o usuário e senha sejam inválidos, apresenta uma mensagem de erro
- Caso o usuário e senha estejam corretos, abre a tela de admin

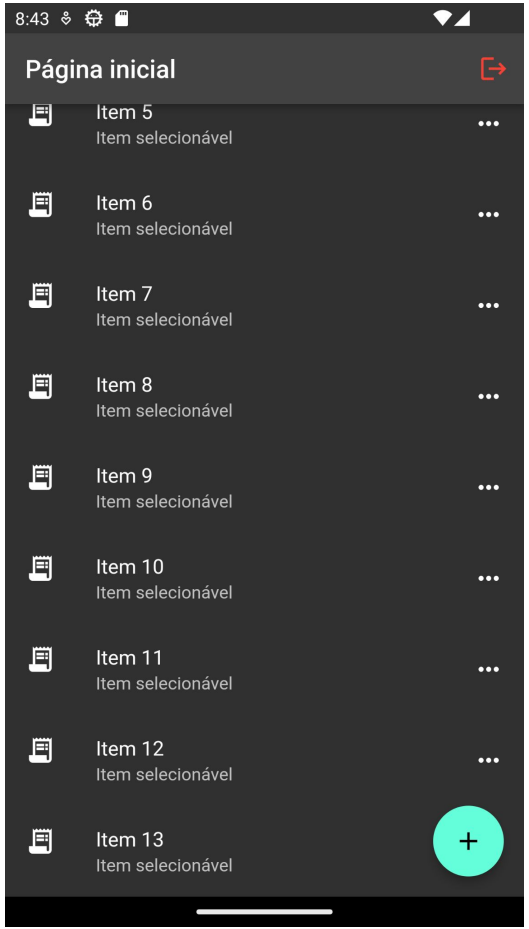
1. A tela de admin deve possuir um botão de adicionar
2. A tela de admin deve possuir um botão de logoff

Exercício 16 - cont

Ver mais







Testes para implementar

- Primeiro teste: Login: Entrar e Sair. Efetuar o Login, entrar na página de admin, esperar alguns segundos e sair
- Segundo teste: Login: Usuário/senha inválido. Testar o formulário de login com credenciais inválidas. Verificar se a mensagem de erro é exibida
- Terceiro teste: Login: Entrar e clicar no botão adicionar 10x