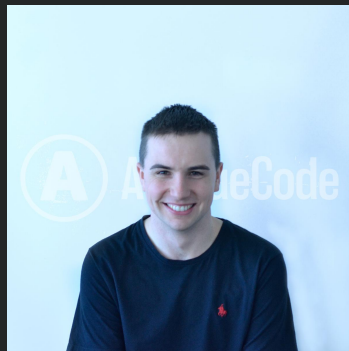


Testando uma aplicação com Arquitetura Hexagonal e Spring Boot

Dezembro/2020





Henrique Schmidt



1

Arquitetura
Hexagonal

2

Pirâmide de testes

3

Testando a lógica
de negócio

4

Testando os
adaptadores

5

Testando a
integração entre
componentes

Arquitetura Hexagonal

Ports & Adapters

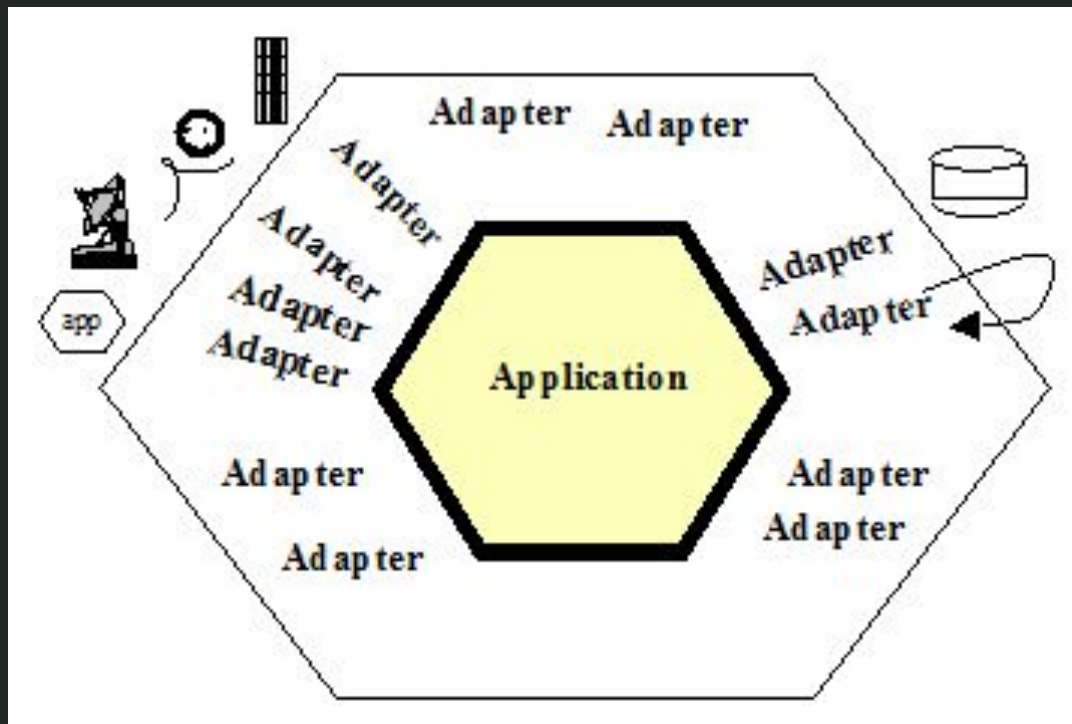
Objetivo #1

Permitir que uma aplicação seja igualmente guiada por usuários, programas, testes automatizados ou scripts.

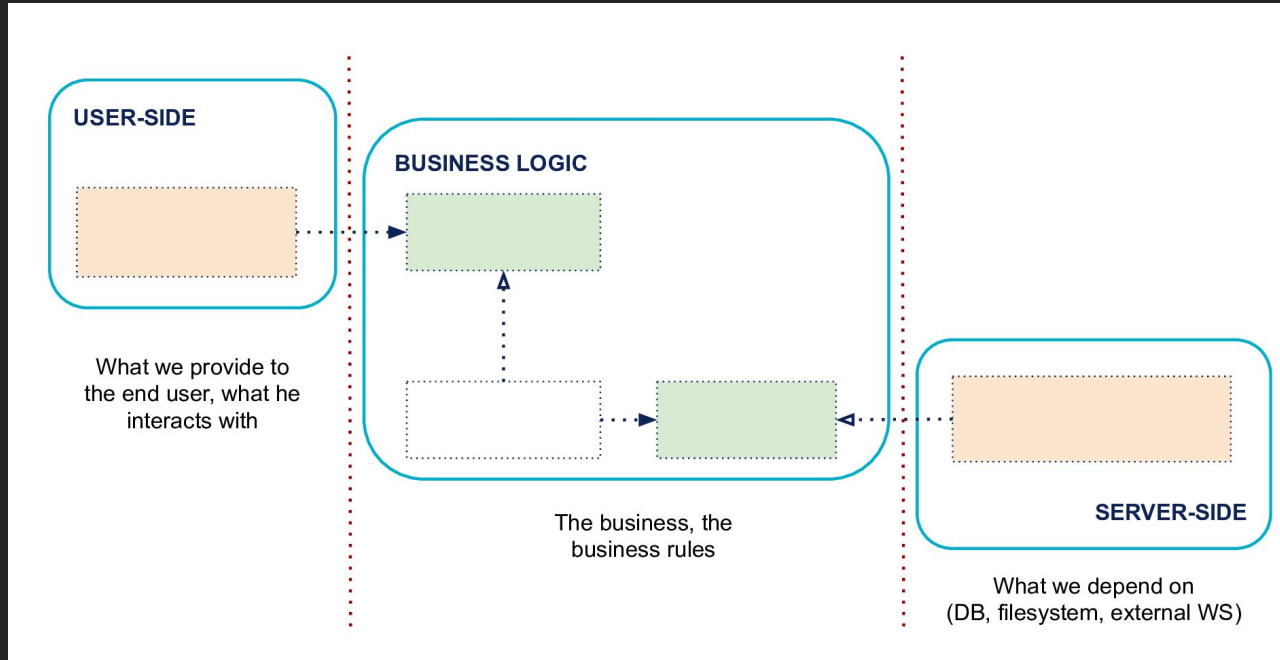
Objetivo #2

Permitir que uma aplicação seja desenvolvida de forma isolada dos banco de dados e dispositivos necessários na hora da execução.

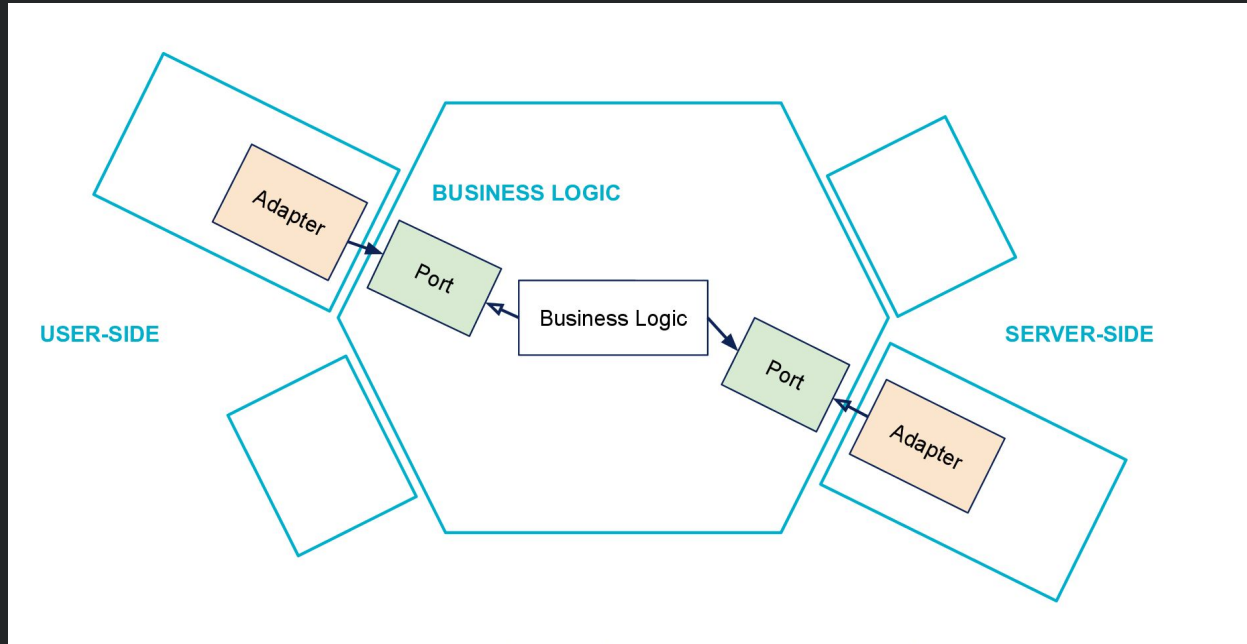
1 Arquitetura Hexagonal



<https://alistair.cockburn.us/wp-content/uploads/2018/02/Hexagonal-architecture-basic-1.gif>



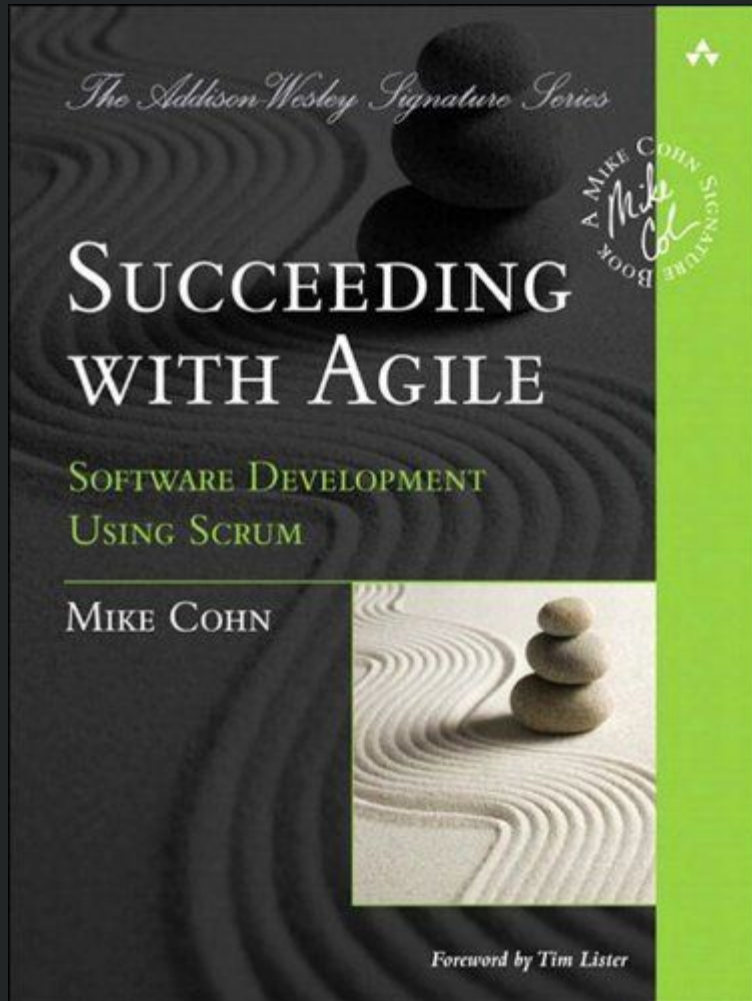
<https://blog.octo.com/en/hexagonal-architecture-three-principles-and-an-implementation-example/>



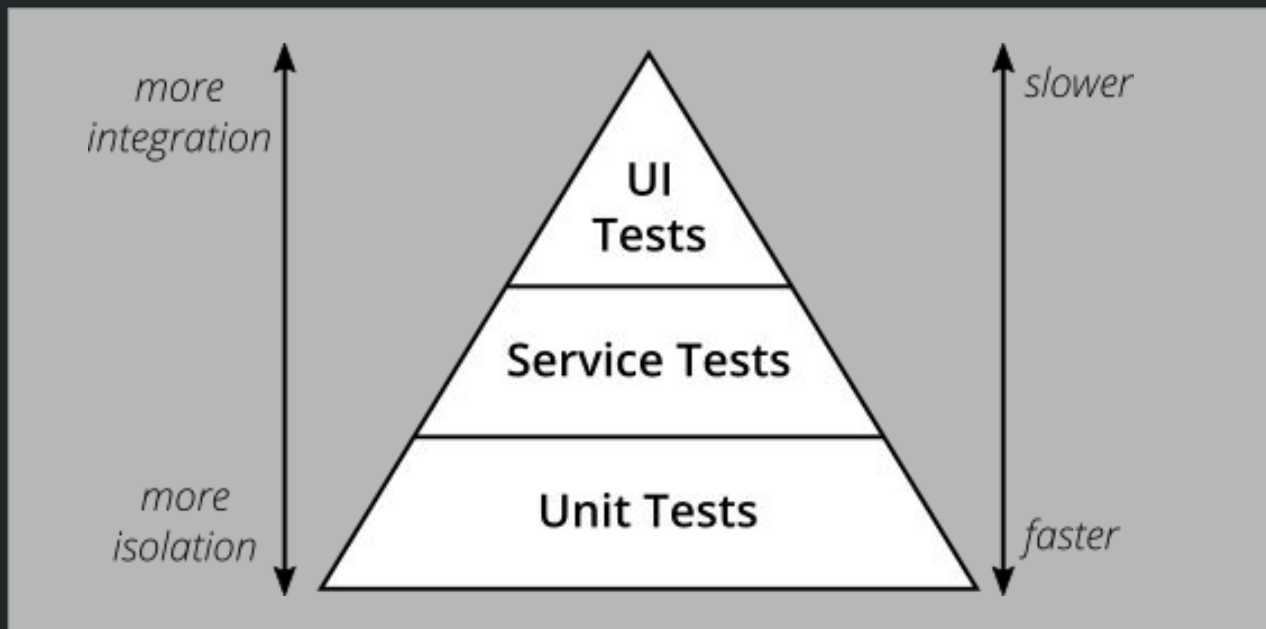
<https://blog.octo.com/en/hexagonal-architecture-three-principles-and-an-implementation-example/>



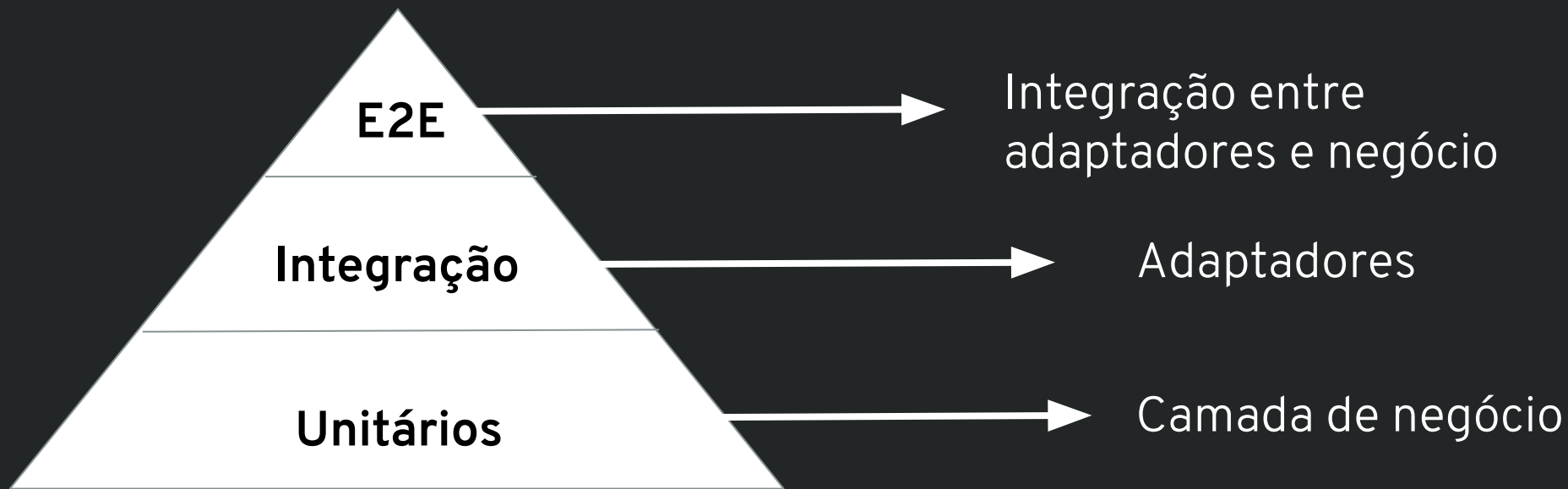
Pirâmide de testes



@mikewcohn

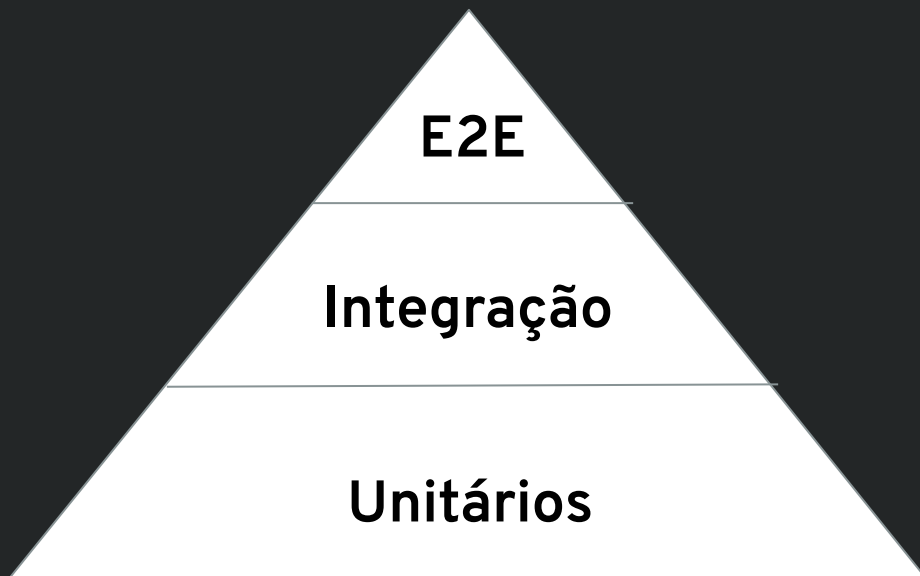


<https://martinfowler.com/articles/practical-test-pyramid.html>





Testando a lógica de negócio



Camada de negócio

1. Não depende de infra-estrutura (banco de dados, kafka, etc);
2. As regras de negócio devem ser escritas nesta camada;
3. Deve depender de interfaces (portas) para se comunicar com o mundo externo;
4. Deve ser chamado pelos adaptadores primários (lado do usuário).
5. Também pode ser chamada de domínio ou core.

**Como não há infra-estrutura
complexa envolvida,
precisamos apenas de uma
biblioteca de testes**

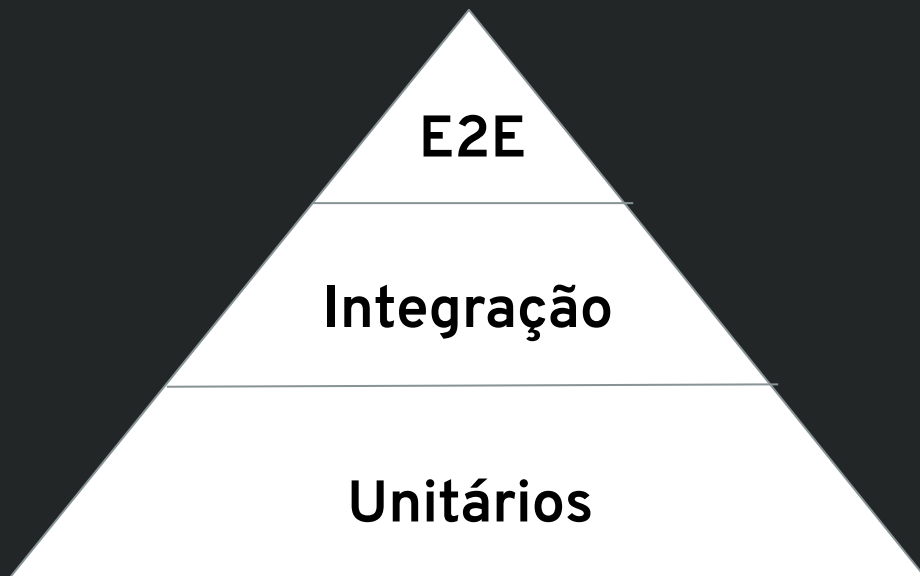


```
15
16 @ExtendWith(MockitoExtension.class)
17 class PlayerFacadeTest {
18
19     @Mock
20     private PlayerRepository playerRepository;
21
22     @InjectMocks
23     private PlayerFacade facade;
24
25     @Test
26     void shouldReturnTheListOfPlayers() {
27         when(playerRepository.findAll()).thenReturn(Flux.just(henrique(), fernando()));
28
29         StepVerifier.create(facade.allPlayers())
30             .expectNext(henrique())
31             .expectNext(fernando())
32             .expectComplete()
33             .verify();
34     }
35
```

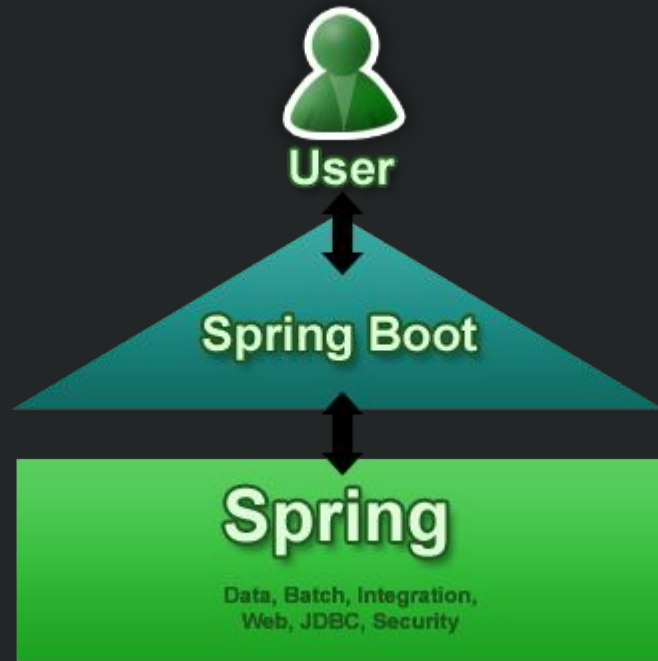
```
36     @Test
37     void shouldSaveANewPlayer() {
38         when(playerRepository.save(diego())).thenReturn(Mono.just("id1"));
39
40         StepVerifier.create(facade.create(diego()))
41             .expectNext("id1")
42             .expectComplete()
43             .verify();
44
45         verify(playerRepository).save(diego());
46     }
47
```

4

Testando os adaptadores



O propósito do Spring Boot é tornar fácil a criação de aplicações Spring.



<https://spring.io/blog/2013/08/06/spring-boot-simplifying-spring-for-everyone/>

Provê a infraestrutura para diversos tipos de teste, entre eles:

- Banco de dados relacional;
- MongoDB;
- Kafka;
- Chamada para API externa.

**Objetivo é garantir que
nosso código integra de
forma correta com o mundo
externo**



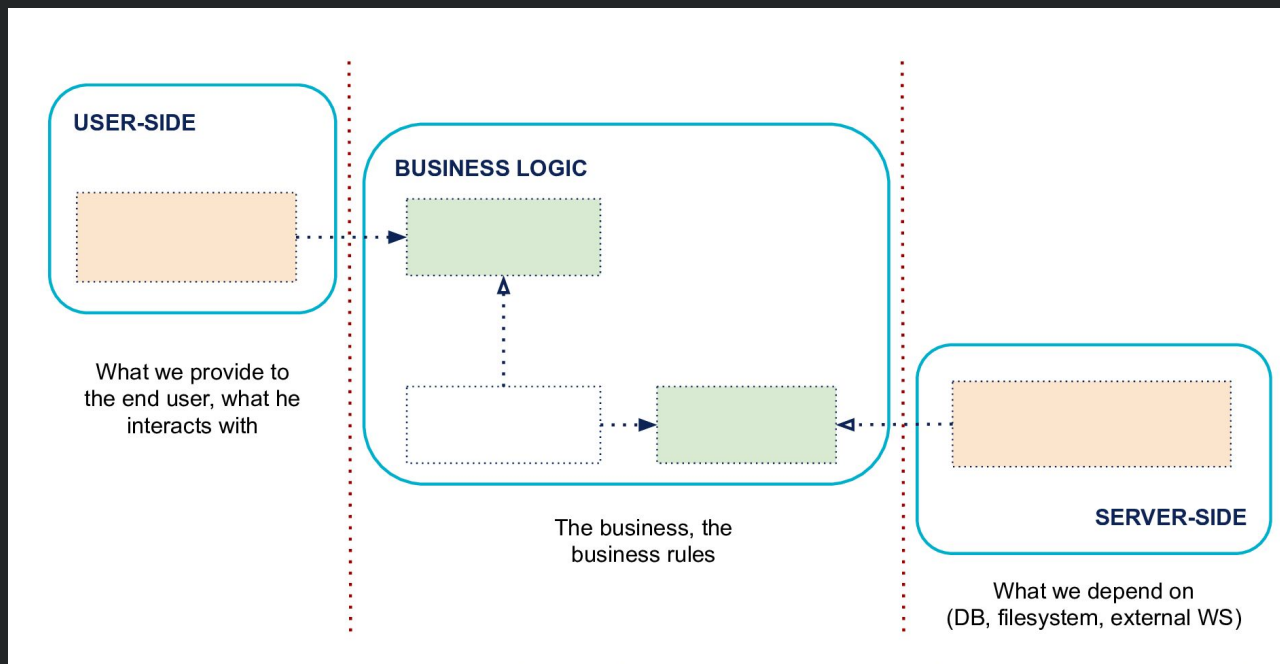
**Não crie mocks para tipos
que você não é o dono**

How to write good tests - Documentação do mockito

<https://github.com/mockito/mockito/wiki/How-to-write-good-tests>



4 Testando os adaptadores



https://blog.octo.com/en/hexagonal-architecture-three-principles-and-an-implementation-example
/

Testando o Controller

Contexto Spring

O Spring Boot fornece diversas anotações para levantar apenas uma parte de uma aplicação.

@WebMvcTest

@WebFluxTest


```
19  @WebFluxTest(PlayerController.class)
20  @Import(PlayerMapperImpl.class)
21  class PlayerControllerTest {
22
23      @Autowired
24      private WebTestClient webTestClient;
25
26      @MockBean
27      private PlayerFacade playerFacade;
28
```

```
@IntegrationTest
void shouldReturnTheListOfPlayers() {
    when(playerFacade.allPlayers()).thenReturn(Flux.just(henrique(), fernando()));

    webTestClient.get().uri("/players")
        .exchange()
        .expectStatus().isOk()
        .expectBody()
        .jsonPath("[0].id").isEqualTo("idHenrique")
        .jsonPath("[0].name").isEqualTo("Henrique")
        .jsonPath("[0].position").isEqualTo("GK")
        .jsonPath("[0].team").isEqualTo("Gremio")
        .jsonPath("[1].id").isEqualTo("idFernando")
        .jsonPath("[1].name").isEqualTo("Fernando")
        .jsonPath("[1].position").isEqualTo("CF")
        .jsonPath("[1].team", equalTo("Barcelona"));
}
```

```
@IntegrationTest
void shouldNotCreateAPlayerWithoutName() {
    Player playerWithoutName = new Player(null, null, "CF", "Gremio");

    webTestClient
        .post().uri("/players")
        .bodyValue(playerWithoutName)
        .exchange()
        .expectStatus().isBadRequest()
        .expectBody()
        .jsonPath("$.code").isEqualTo("field_validation_error")
        .jsonPath("$.description").doesNotExist()
        .jsonPath("$.details").value(hasSize(1))
        .jsonPath("$.details[0].name").isEqualTo("name")
        .jsonPath("$.details[0].description").isEqualTo("must not be empty");

    verifyNoInteractions(playerFacade);
}
```

MongoDB

@DataMongoTest

1. A aplicação se conectará a um banco de dados MongoDB embarcado, por padrão.
2. Todos documentos e repositórios estarão disponíveis para teste.

```
15  @DataMongoTest
16  @Import({MongoPlayerRepository.class, PlayerDocumentMapperImpl.class})
17  class MongoPlayerRepositoryTest {
18
19      @Autowired
20      private ReactiveMongoTemplate reactiveMongoTemplate;
21
22      @Autowired
23      private MongoPlayerRepository mongoPlayerRepository;
24
```

```
40     @IntegrationTest
41     void shouldSaveAPlayer() {
42         Player savedHenrique = toPlayer(reactiveMongoTemplate.save(toDocument(henrique()))).block();
43
44         String diegoId = mongoPlayerRepository.save(diego()).block();
45         Player savedDiego = diegoWithId(diegoId);
46
47         StepVerifier.create(mongoPlayerRepository.findAll())
48             .expectNext(savedHenrique)
49             .expectNext(savedDiego)
50             .expectComplete()
51             .verify();
52     }
```

API Externa

@AutoConfigureWireMock

1. Esta anotação subirá um servidor WireMock onde a aplicação fará chamadas como se fosse um servidor real.

```
14  @SpringBootTest(classes = WebClientAutoConfiguration.class)
15  @Import(RestTeamClient.class)
16  @AutoConfigureWireMock(port = 0)
17  class RestTeamClientTest {
18
19      @Autowired
20      private TeamClient client;
21
```

```
22  @Test
23  void shouldReturnTrueWhenTeamExists() {
24      stubFor(get(urlEqualTo("/teams/existsTeamId"))
25              .willReturn(aResponse().withBodyFile("exists_team.json")
26                  .withHeader("Content-Type", "application/json")));
27
28      StepVerifier.create(client.exists("existsTeamId"))
29          .expectNext(true)
30          .verifyComplete();
31
32      verify(getRequestedFor(urlEqualTo("/teams/existsTeamId")));
33  }
```

Kafka



Testando a integração entre componentes



@SpringBootTest

1. Essa anotação permite fazer um teste levantando todo o contexto de uma aplicação Spring Boot.
2. Com isso, é possível validar que a implementação dos adaptadores funcionam corretamente quando integrados com a camada de negócio.

5 Testando a integração entre os componentes

```
19 @SpringBootTest
20 @AutoConfigureDataMongo
21 @AutoConfigureWebTestClient
22 @Import(MongoTestHelper.class)
23 class PlayerAcceptanceTest {
24
25     @Autowired
26     private WebTestClient webClient;
27
28     @Autowired
29     private MongoTestHelper mongoTestHelper;
30
```



```
112     @DisplayName("As a user, I want to find a player by id")
113     @AcceptanceTest
114     void findPlayerById() {
115         String idHenrique = mongoTestHelper.save(henrique()).block();
116
117         webClient.get().uri("/players/{id}", idHenrique)
118             .exchange()
119             .expectStatus().isOk()
120             .expectBody()
121             .jsonPath("$.id").isEqualTo(idHenrique)
122             .jsonPath("$.name").isEqualTo("Henrique")
123             .jsonPath("$.position").isEqualTo("GK")
124             .jsonPath("$.team").isEqualTo("Gremio");
125     }
126
127
```

```
91     @DisplayName("As a user, I want to delete a player")
92     @AcceptanceTest
93     void deletePlayer() {
94         String idHenrique = mongoTestHelper.save(henrique()).block();
95         mongoTestHelper.save(fernando()).block();
96         webClient.get().uri("/players/")
97             .exchange()
98             .expectStatus().isOk()
99             .expectBody()
100             .jsonPath("$").value(hasSize(2));
101
102         webClient.delete().uri("/players/{id}", idHenrique)
103             .exchange().expectStatus().isNoContent();
104
105         webClient.get().uri("/players/")
106             .exchange()
107             .expectStatus().isOk()
108             .expectBody()
109             .jsonPath("$").value(hasSize(1));
110     }
```

Thank you!

Questions?

Henrique Schmidt
henriquels25@gmail.com



Tell a Story

An optional subtitle could be here.

This presentation is meant to help craft your best slideshow ever at Avenue Code. When creating it, keep it simple and focus on a few key points. Leave the rest for oral communication, and make sure to stress key insights.

Use these slides as templates, replacing the content with your own. When you need to emphasize something, **make it bold**. Sometimes, **colored text can help you too**, but **DO NOT** combine a lot of different colored text like this sentence. Choose only among these eight colors when highlighting your sentences. And remember: use different colors only when there's a need to differentiate an emphasized information from another.

**Use these as your
section breaks**

**Make this
colored to
draw
attention**

Another optional subtitle could be here. It should be 14pt and on Overpass Semi-Bold.

1. You can use numbered lists like this one too.
2. Numbered lists should be used when there's a sequential logic to your slide.
3. Otherwise, please refrain to use bullet points.
4. If your list is longer than this text box, please duplicate this slide and continue on another one.
5. You can also reevaluate and keep things brief. ;)

A design optimized for you

Font sizes and spacing are already set for your best interest, so no need to change them. There are many layouts to choose, so choose wisely.

For instance, there's no need for another column here. Maybe this information would be better displayed in a bullet list, don't you think?

Compare two things

Comparison Item #1

- This is a preferred layout for comparing two thing.
- Don't forget to keep it brief!
- Otherwise, duplicate this one and continue the comparison in another slide.

Comparison Item #2

- If there are too many items, you may want to switch over to layout 04 Long Text, 2 Columns.
- Or you can create a table too.

Or compare three things

Comparison Item #1

- If you need to compare more than three things...
- Item #1 is so good!


Comparison Item #2

- ...probably you will be better off creating an infographic...
- Item #2 is so so.

Comparison Item #3


- ...instead of using this layout.
- Ugh, item #3 is the worst.

Playing with big words



**Luke, I'm your father.
Also, this is a quote
in your presentation.**

Darth Vader—or whoever said it should have their name here.
You can even put when or where this quote was said here if needed.



3

Replace this with your section title

**Big words like these are
used to display powerful
insights in your slideshow.**

**Let's add some
images, shall we?**

4 Replace this with your section title

Images are everything

You should try to always use images in your presentations – like it was said before, keep it simple and focus on a few key points.

Use images that make sense to your presentation and to your audience. Try to use high quality images whenever you can.

Place your image here. Try to keep your image within these boundaries.

And remove this text box later!

Again, remove this text box later!

One more time: remove this text box later!

4 Replace this with your section title

Place your image here. Try to keep your image within these boundaries.

And remove this text box later!

Again, remove this text box later!

One more time: remove this text box later!

C'mon, place a chart there

- Combine bullet points and images like charts to quickly show about it
- Summarize the image information here
- Keep duplicating this slide if there's too much to explain: make it easy for your audience!

4 Replace this with your section title

Place your image here. Try to keep your image within these boundaries.

And remove this text box later!

Again, remove this text box later!

One more time: remove this text box later!

It could be an illustration too

If you're planning on using an image with a transparent background (.png), you should fit it inside the text box boundaries and align it to the top.

Place wide images like flow charts below and write their title here

Place your image here. Try to keep your image within these boundaries.

And remove this text box later!

Again, remove this text box later!

One more time: remove this text box later!

Place your full page image here. Your image shouldn't necessarily fill the whole page, but it must be sized equally as this page height **OR** width.

And remove this text box later! Especially if it's an image with transparency (a .png image).

Again, remove this text box later!

One more time: remove this text box later!

4 Replace this with your section title



- Multiple images should be used for comparison purposes



- You can add brief key points here on why you're comparing these images, what do you think?

4 Replace this with your section title



- You can also compare three different images, for instance



- Keep in mind this can distract your audience, and you should guide them on what's the big picture here



- If more than three images is necessary, consider continuing this on another slide

4 Replace this with your section title. And here's a blank slide for you.

