

BDD, from feature to unit testing

Henrique **Valcanaia**

BDD, from feature to unit testing

Henri Haia



Henrique **Valcanaia**



programming since '09



~10y writing "hacks"



Computer Engineering @ UFRGS

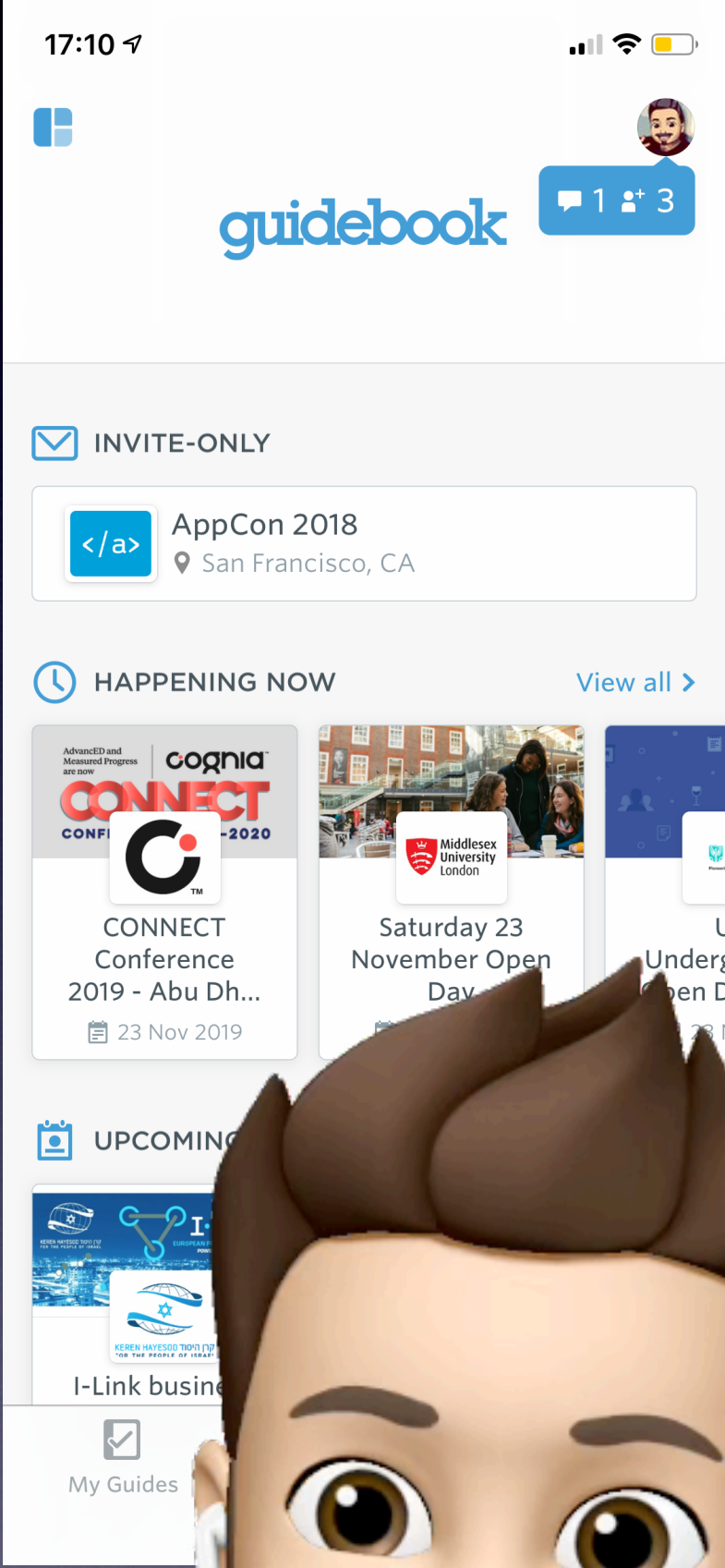
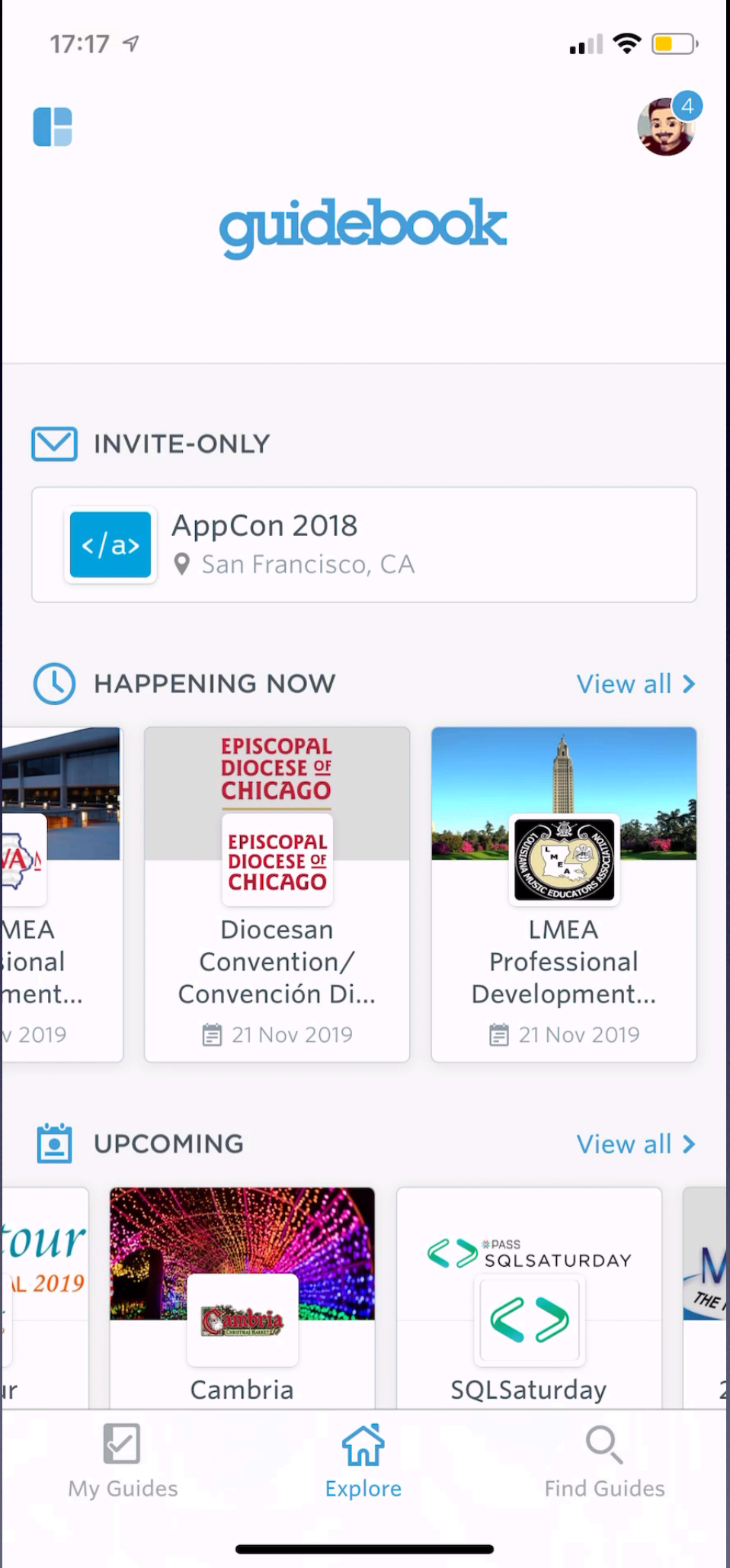
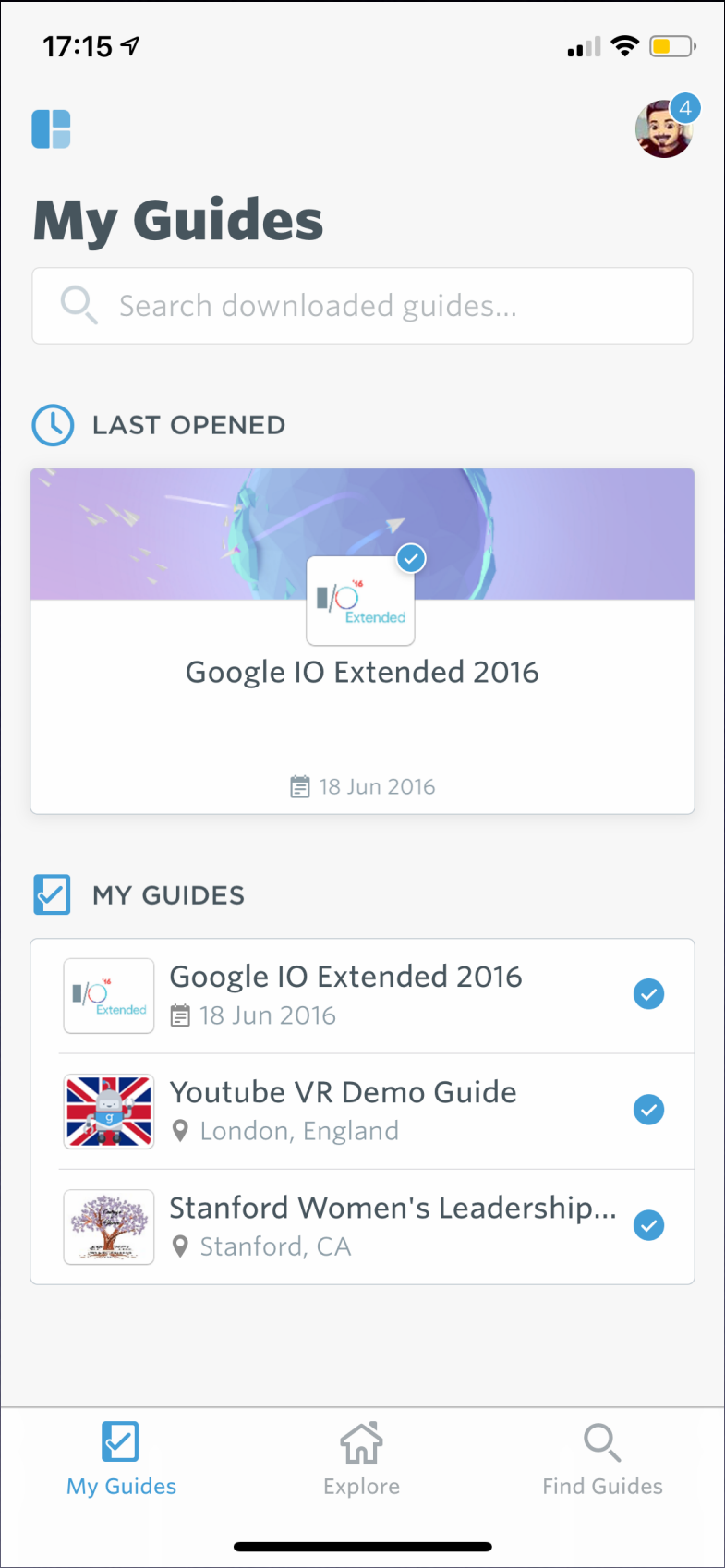


Apple Developer Academy 2015/2016



iOS ~5 years







<https://bit.ly/projetobdd>



2:20



Marvel Comics

A

Amazing Spider-Man
(1999) #558 (Turner
Variant)

Amazing Spider-Man
500 Covers Slipcase -
Book II (Trade
Paperback)

Ant-Man (2003) #1

Ant-Man (2003) #2

Ant-Man (2003) #3

Ant-Man (2003) #4

C

Cap Transport (2005)
#1

how this talk was **born**?



what's the **goal** with this talk



awesome **experience** to end **user**

robust applications

define: robustness

*ability to withstand or overcome adverse
conditions or **rigorous testing***

how did I **start**
seeking robustness?

~*la **garantia** soy yo*~



I DON'T MAKE **MISTAKES**



next **professional** step?

?

tests, *of course*



let's see **how test works**




```
protocol Calculator {  
    func double(_ value: Int) -> Int  
}  
  
final class FastCalculator: Calculator {  
    func double(_ value: Int) -> Int {  
        return value << 1  
    }  
}
```

```
import XCTest  
  
final class FastCalculatorTest: XCTestCase {  
    lazy var calculator: Calculator = {  
        return FastCalculator()  
    }()  
  
    func testDoubleSmallNumber() {  
        let two = 2  
        let expectedFour = self.calculator.double(two)  
        XCTAssertEqual(expectedFour, 4)  
    }  
}
```


I don't do TDD



I don't **always** do TDD



things that
change constantly



business rules



critical systems



critical and real time
systems???

define: critical system

*"a system which must be **highly reliable**
and **retain this reliability** as they **evolve**
without incurring prohibitive costs"*



*"increasingly software can be considered to be **critical**, due to the **business** or other functionality which it supports"*



"upgrades or changes to such software are expensive and risky, primarily because the software has not been designed and built for ease of change"



*"expertise, **tools** and methodologies
which support the design and
implementation of software systems
that evolve without risk (of failure or
loss of quality) are **essential**"*



Boeing 737 **MAX**

and?

Lion Air **737 MAX** **crash** in Indonesia

3.2 Contributing Factors

Contributing factors defines as actions, omissions, events, conditions, or a combination thereof, which, if eliminated, avoided or absent, would have reduced the probability of the accident or incident occurring, or mitigated the severity of the consequences of the accident or incident. The presentation is based on chronological order and not to show the degree of contribution.

1. During the design and certification of the Boeing 737-8 (MAX), assumptions were made about flight crew response to malfunctions which, even though consistent with current industry guidelines, turned out to be incorrect.
2. Based on the incorrect assumptions about flight crew response and an incomplete review of associated multiple flight deck effects, MCAS's reliance on a single sensor was deemed appropriate and met all certification requirements.
3. MCAS was designed to rely on a single AOA sensor, making it vulnerable to erroneous input from that sensor.
4. The absence of guidance on MCAS or more detailed use of trim in the flight manuals and in flight crew training, made it more difficult for flight crews to properly respond to uncommanded MCAS.
5. The AOA DISAGREE alert was not correctly enabled during Boeing 737-8 (MAX) development. As a result, it did not appear during flight with the mis-calibrated AOA sensor, could not be documented by the flight crew and was therefore not available to help maintenance identify the mis-calibrated AOA sensor.
6. The replacement AOA sensor that was installed on the accident aircraft had been mis-calibrated during an earlier repair. This mis-calibration was not detected during the repair.
7. The investigation could not determine that the installation test of the AOA sensor was performed properly. The mis-calibration was not detected.
8. Lack of documentation in the aircraft flight and maintenance log about the continuous stick shaker and use of the Runaway Stabilizer NNC meant that information was not available to the maintenance crew in Jakarta nor was it available to the accident crew, making it more difficult for each to take the appropriate actions.
9. The multiple alerts, repetitive MCAS activations, and distractions related to numerous ATC communications were not able to be effectively managed. This was caused by the difficulty of the situation and performance in manual handling, NNC execution, and flight crew communication, leading to ineffective CRM application and workload management. These performances had previously been identified during training and reappeared during the accident flight.

3.2 Contributing Factors

Contributing factors defines as actions, omissions, events, conditions, or a combination thereof, which, if eliminated, avoided or absent, would have reduced the probability of the accident or incident occurring, or mitigated the severity of the consequences of the accident or incident. The presentation is based on chronological order and not to show the degree of contribution.

1. During the design and certification of the Boeing 737-8 (MAX), assumptions were made about flight crew response to malfunctions which, even though consistent with current industry guidelines, turned out to be incorrect.
2. Based on the incorrect assumptions about flight crew response and an incomplete review of associated multiple flight deck effects, MCAS's reliance on a single sensor was deemed appropriate and met all certification requirements.
3. MCAS was designed to rely on a single AOA sensor, making it vulnerable to erroneous input from that sensor.
4. The absence of guidance on MCAS or more detailed use of trim in the flight manuals and in flight crew training, made it more difficult for flight crews to properly respond to uncommanded MCAS.

who's **guilty**?



does this question
even make **sense**?



future society about software?



why are we talking about this?!

¬_(ツ)_/

Robustness

Behaviour-driven development

Behaviour
Driven
Development

Test-Driven Development++ Acceptance Test-Driven Planning++

[https://web.archive.org/
web/20150901151029/
http://behaviourdriven.org/](https://web.archive.org/web/20150901151029/http://behaviourdriven.org/)



but what **is** BDD?

BDD is **not** a **silver bullet**



*an Agile software development **process** that encourages **collaboration** among developers, QA and **non-technical or business participants** in a software project*

[https://en.wikipedia.org/wiki/
Behavior-driven_development](https://en.wikipedia.org/wiki/Behavior-driven_development)



improve communication **between**
tech and non tech people



prevent **wrong assumptions**



Introducing BDD.

Daniel Terhorst-North.

Better Software, March 2006.

<https://dannorth.net/introducing-bdd/>



Dan was very happy **doing Agile, TDD**, etc.



same **confusion** and **misunderstandings**



extreme TDD

DEV: Where do I **start**?

DEV: What should I **test**?

DEV: How much to **test at the moment**?

do you see where we're **heading to?**



perfect testable beautiful code, of course



"if someone had told me that!"



improve communication **between**
tech and non tech people



prevent **wrong assumptions**



makes **sense**?!



BDD to the rescue



Software = **User Story** + **Acceptance Criteria**

similar to **Domain-Driven Development**

As a [person who will benefit],
I want [some feature],
so that [the benefit or value of the feature].

Given [some initial context],
when [an event occurs],
then [ensure some outcomes].

checkpoint



user stories
acceptance criterias
tests
code

shut up and talk about iOS



BDD frameworks in **Swift**




```
XCTContext.runActivity(named: "Given") { (activity: XCTActivity) in
    XCTContext.runActivity(named: "When") { (activity: XCTActivity) in
        XCTContext.runActivity(named: "Then") { (activity: XCTActivity) in
            XCTAssertTrue(true)
        }
    }
}
```


Quick (8.5k★)

a **behavior-driven development framework for Swift**
and Objective-C. Inspired by RSpec, Specta, and Ginkgo

[https://github.com/
Quick/Quick](https://github.com/Quick/Quick)



Nimble (3.5k★)

express the expected outcomes of Swift or Objective-C expressions. Inspired by Cedar

[https://github.com/
Quick/Nimble](https://github.com/Quick/Nimble)




```
it('') { }
```

defines an example


```
describe('') { }  
context('') { }
```

defines a logical groupings of examples


```
import Quick
import Nimble
final class QuickA: QuickSpec {
    override func spec() {
        describe("Given you watched this talk") {
            let you = Person(name: "Your Name IV", watchedThisTalk: true)
            context("When someone ask you about BDD") {
                let someone = Person(name: "Some One", watchedThisTalk: false)
                let yourResponse = someone.askAboutBDD(to: you)
                it("Then you should be able to say something nice") {
                    expect(yourResponse).to(equal("Something nice"))
                }
            }
        }
    }
}
```


As a user

I want to see a list of Marvel's comics

So that I can learn more about it

Given that the app has internet connection
When the app opens
Then show the comics list


```
describe("Given that the app has internet connection") {  
    beforeEach {  
        reachability.hasConnection = true  
    }  
    context("When the app opens") {  
        beforeEach {  
            viewController.viewDidLoad()  
        }  
  
        it("Then should show the comics list") {  
            expect(viewController._didAskToPresentComics).toBeTrue()  
        }  
    }  
}
```


BDD is about **integration** tests



*"an application behavior can be described by
the integration of its components behaviors"*

Robustness

increase agility on Xcode



TARGETS



BDDComicsList



BDDComicsListiOSTests



BDDComicsListMacOSTests



BDDComicsListUITests

TARGETS



BDDComicsList



BDDComicsListiOSTests



BDDComicsListMacOSTests



BDDComicsListUITests

default iOS target

TARGETS



BDDComicsList



BDDComicsListiOSTests



BDDComicsListMacOSTests



BDDComicsListUITests

unit testing


```
context("When the view finished loading") {
    it("Should have the activity indicator view initially hidden") {
        expect(self.charactersListViewController.activityIndicator.isHidden)
            .to(beTrue())
        expect(self.charactersListViewController.activityIndicator.isAnimating)
            .to(beFalse())
    }
    it("Should set an empty footer for the tableview to remove the empty lines at the end") {
        expect(self.charactersListViewController.tableView.tableFooterView).toNot(beNil())







        expect(self.charactersListViewController.tableView.tableFooterView)
            .to(beAnInstanceOf(UITableView.self))
    }
    it("Should have the tableView datasource set") {
        expect(self.charactersListViewController.tableView.dataSource)
            .to(be(self.charactersListViewController))
    }
    it("Should have table selection disabled") {
        expect(self.charactersListViewController.tableView.allowsSelection)
            .to(beFalse())
    }
    it("Should use automatic cell height") {
        expect(self.charactersListViewController.tableView.rowHeight)
            .to(equal(UITableView.automaticDimension))
    }
}
```


BDDComicsListiOSTests

All Passed Failed Mixed

Tests

▼ CharactersListModuleSpecs > BDDComicsListiOSTests 6 passed (100%) in 0,0073s

- ✓  Given_that_I_have_internet_connection__When_the_server_response_is_invalid__Then_present_an_error_message_informing_that_the_content_couldn_t_be_read()
- ✓  Given_that_I_don_t_have_internet_connection__When_it_tries_to_load_the_comic_characters__Then_present_a_message_informing_that_there_is_no_internet_connection()
- ✓  Given_that_I_have_internet_connection__When_the_server_response_is_valid__Then_present_a_list_with_the_name_of_the_characters_of_that_comic_sorted_ascending_by_the_first_letter()
- ✓  Given_that_a_comic_was_selected__When_the_view_with_the_list_of_comic_characters_is_presented__Then_show_the_name_of_the_comic_as_the_view_s_title()
- ✓  Given_that_a_comic_was_selected__When_the_app_is_loading_the_comic_characters__Then_present_a_UI_activity_indicator()
- ✓  Given_that_a_comic_was_selected__When_the_app_finished_loading_the_comic_characters__Then_hide_the_UI_activity_indicator()

Given_that_I_have_internet_connenction__

When_the_server_response_is_inlivad__

Then_present_an_error_message_informing_
that_the_content_couldn_t_be_read()

Given_that_I_have_internet_connenction__

When_the_server_response_is_inlivad__

Then_present_an_error_message_informing_
that_the_content_couldn_t_be_read()

TARGETS



BDDComicsList



BDDComicsListiOSTests



BDDComicsListMacOSTests



BDDComicsListUITests

macOS unit test







```
describe("Given that I don't have internet connection") {  
  beforeEach {  
    self.dataGatewayMock._fetchComicsResponseMock = .failure(.noInternetConnection)  
  }  
  context("When it tries to load the comics") {  
    beforeEach {  
      self.presenterMock._didAskToPresentNoInternetConnectionErrorMessage = false  
      self.interactor.loadListOfComics()  
    }  
    it("Then present a message informing that there is no internet connection") {  
      expect(self.presenterMock._didAskToPresentNoInternetConnectionErrorMessage).to(beTrue())  
    }  
  }  
}
```


BDDComicsListMacOSTests

All Passed Failed Mixed

Tests

▼ CharactersListDataGatewaySpecs > BDDComicsListMacOSTests 5 passed (100%) in 0,0911s

- ✓  Given_that_I_don_t_have_internet_connection__When_trying_to_fetch_comics__Should_call_the_completion_block_passing_noInternetConnection_result()
- ✓  Given_that_I_have_internet_connection__When_asked_to_fetch_comics__Should_ask_the_remote_data_for_the_comics()
- ✓  Given_that_I_have_internet_connection__When_the_fetch_comics_response_could_not_be_parsed__Should_call_the_completion_block_passing_responsesInvalid()
- ✓  Given_that_I_have_internet_connection__When_the_fetch_comics_response_could_be_parsed_and_succeeded__Should_call_the_completion_block_passing_the_comics()
- ✓  Given_that_I_have_internet_connection__When_the_fetch_comics_response_could_be_parsed_but_response_code_is_different_than_200__Should_call_the_completion_block_passing_responsesInvalid()

TARGETS



BDDComicsList



BDDComicsListiOSTests



BDDComicsListMacOSTests



BDDComicsListUITests

UI tests


BDDComicsListUITests

```
func testCharactersListTransition() {
    XCTContext.runActivity(named: "Given that there are comics") { _ in
        XCTContext.runActivity(named: "When I select a comic", block: { _ in
            XCTContext.runActivity(named: "Then present the list of characters", block: { _ in
                let app = XCUIApplication()
                let tablesQuery = app.tables
                let mockedComic = "Amazing Spider-Man (1999) #558 (Turner Variant)"

                // Opens Spider Man details
                let spiderMan = tablesQuery.cells.staticTexts[mockedComic]
                XCTAssertTrue(spiderMan.exists)
                spiderMan.tap()

                // Check list of characters
                let charactersNames = ["Archangel", "Avalanche", "Blob", "Colossus", "Destiny",
                                       "Nightcrawler", "Pyro", "Storm", "Wolverine", "X-Men"]
                charactersNames.forEach { (characterName: String) in
                    XCTAssertTrue(tablesQuery.cells.staticTexts[characterName].exists)
                }

                // Navigate back to comics
                app.navigationBar[mockedComic].buttons["Marvel Comics"].tap()
            })
        })
    }
}
```


 XCTAssertTrue failed

BDDComicsListUITests

▼ BDDComicsListUITests > BDDComicsListUITests 1 failed (100%) in 3s

▼   testCharactersListTransition()

▶ Start Test at 2019-11-23 21:55:17.469 (Start) 

▶ Set Up (0.06s) 



▼ Given that there are comics 

▼ When I select a comic 

▼ Then present the list of characters 

Checking existence of `"Amazing Spider-Man (1999) #558 (Turner Variant)" StaticText` (3.44s)


▼ Assertion Failure: BDDComicsListUITests.swift:35: XCTAssertTrue failed

 Automatic Screenshot 


Tear Down (3.51s)

BDDComicsListUITests

▼ BDDComicsListUITests > BDDComicsListUITests 1 failed (100%) in 3s

▼   testCharactersListTransition()

▶ Start Test at 2019-11-23 21:55:17.469 (Start) 

▶ Set Up (0.06s) 



▼ Given that there are comics 

▼ When I select a comic 

▼ Then present the list of characters 

Checking existence of ` "Amazing Spider-Man (1999) #558 (Turner Variant)" StaticText ` (3.44s)

▼ Assertion Failure: BDDComicsListUITests.swift:35: XCTAssertTrue failed


 Automatic Screenshot 


Tear Down (3.51s)

BDDComicsListUITests

▼ BDDComicsListUITests > BDDComicsListUITests 1 failed (100%) in 3s

▼   testCharactersListTransition()

▶ Start Test at 2019-11-23 21:55:17.469 (Start) 

▶ Set Up (0.06s) 



▼ Given that there are comics 

▼ When I select a comic 

▼ Then present the list of characters 

Checking existence of ` "Amazing Spider-Man (1999) #558 (Turner Variant)" StaticText ` (3.44s)

▼ Assertion Failure: BDDComicsListUITests.swift:35: XCTAssertTrue failed


 Automatic Screenshot 


Tear Down (3.51s)

BDDComicsListUITests

▼ BDDComicsListUITests > BDDComicsListUITests 1 failed (100%) in 3s

▼   testCharactersListTransition()

▶ Start Test at 2019-11-23 21:55:17.469 (Start) 

▶ Set Up (0.06s) 



▼ Given that there are comics 

▼ When I select a comic 

▼ Then present the list of characters 

Checking existence of `"Amazing Spider-Man (1999) #558 (Turner Variant)" StaticText` (3.44s)

▼ Assertion Failure: BDDComicsListUITests.swift:35: XCTAssertTrue failed


 Automatic Screenshot 


Tear Down (3.51s)

BDDComicsListUITests

▼ BDDComicsListUITests > BDDComicsListUITests 1 failed (100%) in 3s

▼   testCharactersListTransition()

▶ Start Test at 2019-11-23 21:55:17.469 (Start) 

▶ Set Up (0.06s) 



▼ Given that there are comics 

▼ When I select a comic 

▼ Then present the list of characters 

Checking existence of `"Amazing Spider-Man (1999) #558 (Turner Variant)" StaticText` (3.44s)

▼ Assertion Failure: BDDComicsListUITests.swift:35: XCTAssertTrue failed


 Automatic Screenshot 


Tear Down (3.51s)

BDDComicsListUITests

▼ BDDComicsListUITests > BDDComicsListUITests 1 failed (100%) in 3s

▼   testCharactersListTransition()

▶ Start Test at 2019-11-23 21:55:17.469 (Start) 

▶ Set Up (0.06s) 



▼ Given that there are comics 

▼ When I select a comic 

▼ Then present the list of characters 

Checking existence of `"Amazing Spider-Man (1999) #558 (Turner Variant)" StaticText` (3.44s)

▼ Assertion Failure: BDDComicsListUITests.swift:35: XCTAssertTrue failed


 Automatic Screenshot 


Tear Down (3.51s)

BDDComicsListUITests

▼ BDDComicsListUITests > BDDComicsListUITests 1 failed (100%) in 3s

▼   testCharactersListTransition()

▶ Start Test at 2019-11-23 21:55:17.469 (Start) 

▶ Set Up (0.06s) 



▼ Given that there are comics 

▼ When I select a comic 

▼ Then present the list of characters 

Checking existence of ` "Amazing Spider-Man (1999) #558 (Turner Variant)" StaticText ` (3.44s)

▼ **Assertion Failure: BDDComicsListUITests.swift:35: XCTAssertTrue failed**


 Automatic Screenshot 


Tear Down (3.51s)

BDDComicsListUITests

▼ BDDComicsListUITests > BDDComicsListUITests 1 failed (100%) in 3s

▼   testCharactersListTransition()

▶ Start Test at 2019-11-23 21:55:17.469 (Start) 

▶ Set Up (0.06s) 



▼ Given that there are comics 

▼ When I select a comic 

▼ Then present the list of characters 

Checking existence of `"**Amazing Spider-Man (1999) #558 (Turner Variant)**" StaticText` (3.44s)

▼ Assertion Failure: BDDComicsListUITests.swift:35: XCTAssertTrue failed


 Automatic Screenshot 


Tear Down (3.51s)

BDDComicsListUITests

▼ BDDComicsListUITests > BDDComicsListUITests failed

▼   testCharactersListTransition()

▶ Start Test at 2019-11-23 21:55:17.469 (Start) 

▶ Set Up (0.06s) 


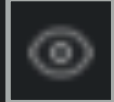
▼ Given that there are comics 

▼ When I select a comic 

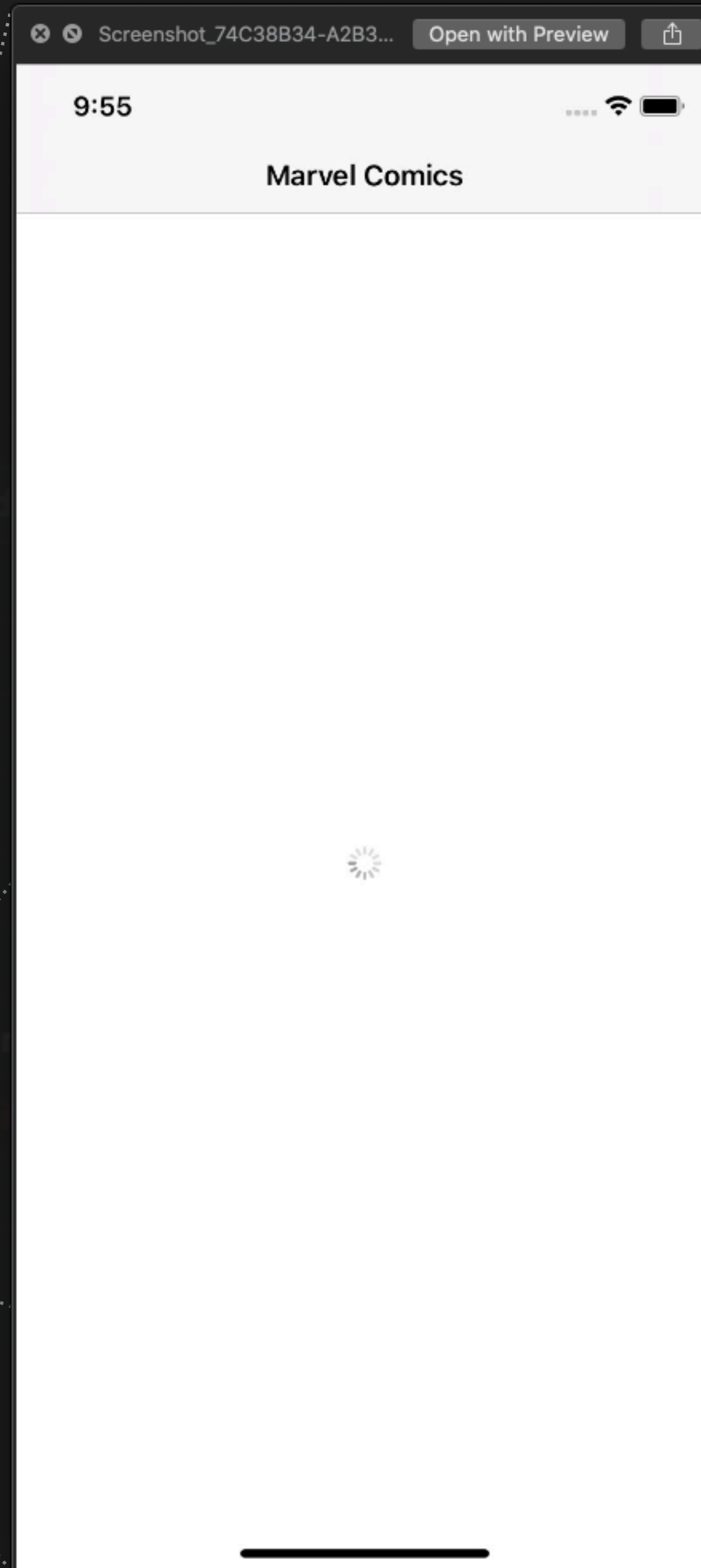
▼ Then present the list of character 

Checking existence of `*Amazing Spider-Man` in the list of characters (3.44s)

▼ Assertion Failure: BDDComicsListUITests.swift:100: Expected to find the character

 Automatic Screenshot 

Tear Down (3.51s)



we're testing
integration between modules



what about **unit testing**?



when do I get to
choose my **architecture**



when you have all the **specs**



behaviours



user stories



acceptance criterias



iOS Architecture Generator

SHR

Store Helper Router

GENERATE ANOTHER ONE

Powered by Swift.
Created by [Guilherme Rambo](#)
[Open Source](#)



<https://iosarchitecture.top/>

now **TDD**, right?




use BDD to define your
components behavior


helps you define
components **responsibilities**




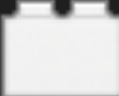
decoupled and specific targets

TARGETS

 BDDComicsList

 BDDComicsListiOSTests

 BDDComicsListMacOSTests

 BDDComicsListUITests

go with behavior
until you have **a unit**



now you do **TDD!**



checkpoint

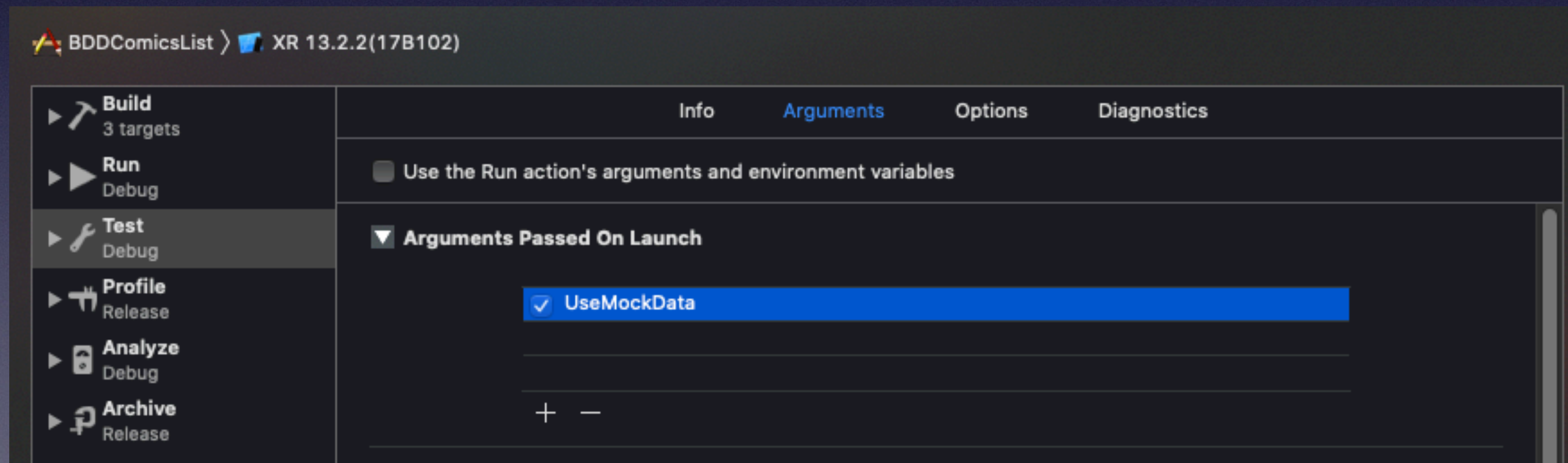


describe **behavior until**
you have a unit

extra



launch arguments 🚀




```
extension UIApplication {  
    var shouldUseMockData: Bool {  
        return ProcessInfo().arguments.contains("UseMockData")  
    }  
}
```


issues



- Insufficient **budget**
- **Size** of the team
- Poor outside **feedback**
- **Minimum** business logic
- **Too many** meetings
- Bad **planning** and unrealistic **expectations**
- Inadequate **communication**
- Treating **quality as solely the testers' responsibility**
- **Inconsistent formatting** for BDD scenarios

recap



- **BDD is about the process**
- BDD is **not a silver bullet**
- BDD helps **specify the product**
- BDD can help in **business validation and clarification**
- Integration tests can be felt as an overhead, but are **essencial to scalable/critical applications**
- Tests outputs are easier to **understand**
- **Low coupling**
- **Communication++**



<https://bit.ly/projetobdd>



<https://iosdevbr.herokuapp.com/>

that's all folks

thanks!



Henrique **Valcanaia**



@ricovalcanaia



@valcanaia



@hvsw



