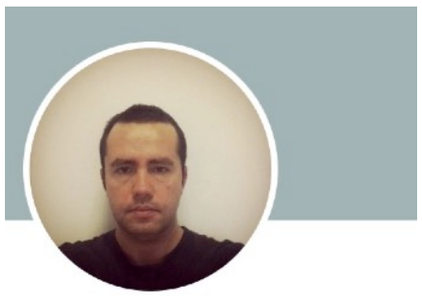
Embrace the evolution of a Mobile Application Product

Mindset and tools for a domain/test driven design approach for mobile application development

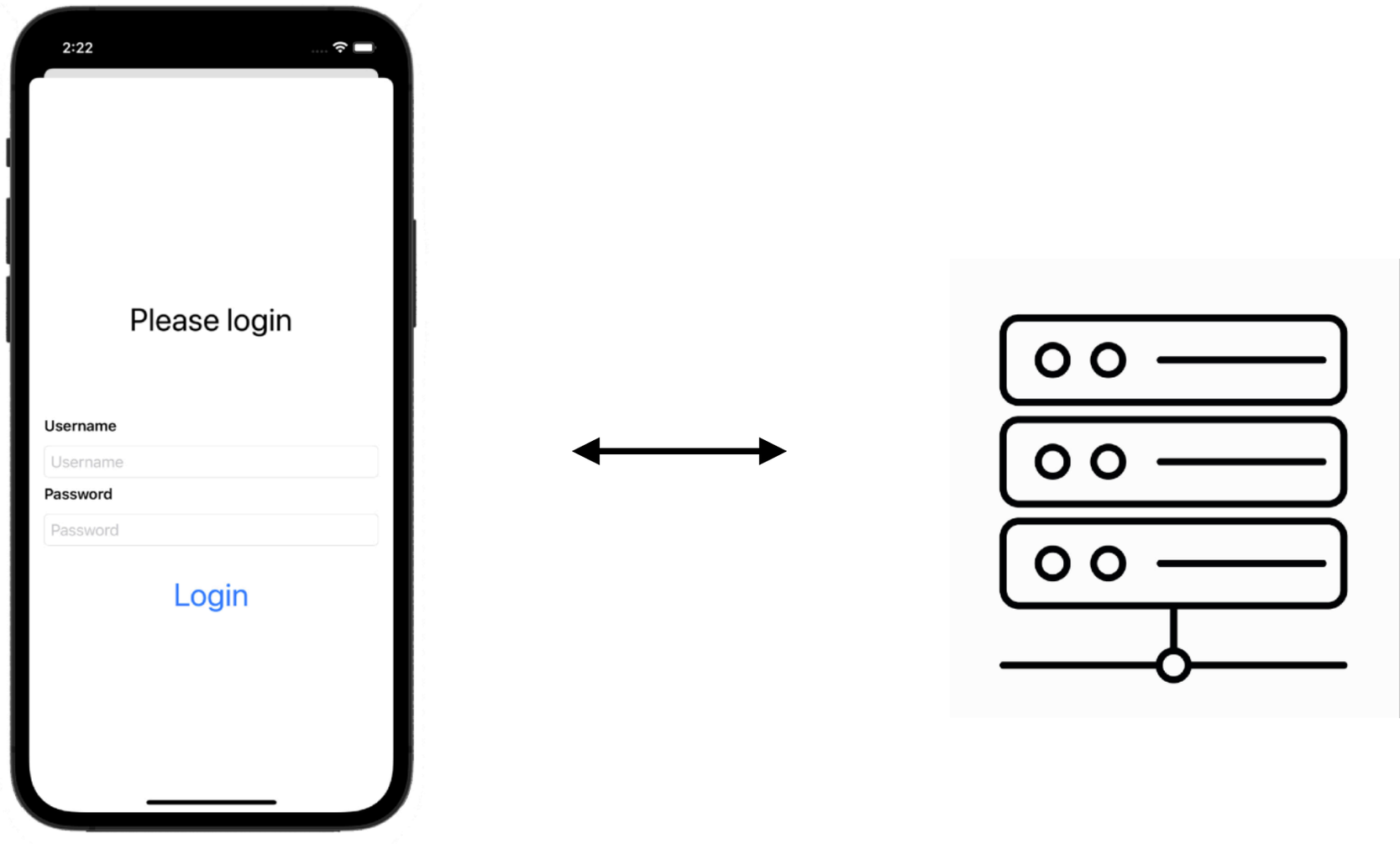


Alessandro Chiarotto
Developer presso Sky Italia

Alex Gotev
Principal Engineer at Sky Italia

Context - Production Environment

Mobile Application & Backend Server

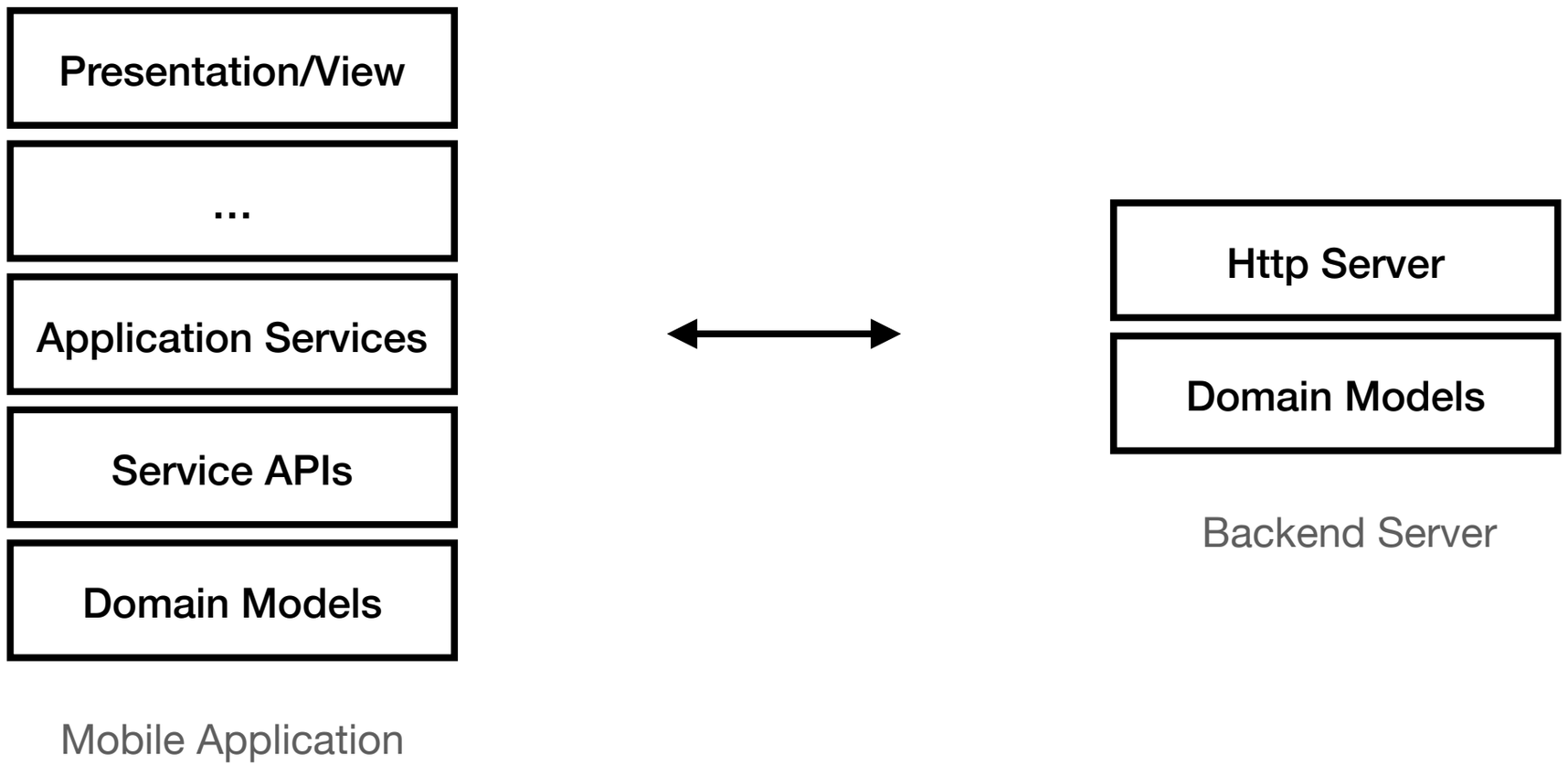


Mobile Application

Backend Server

Layers

Mobile Application / Backend Server

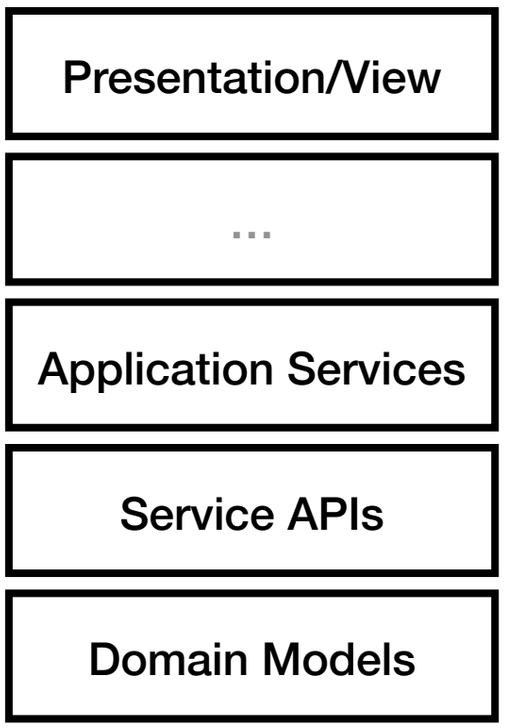


Evolution of a mobile application product

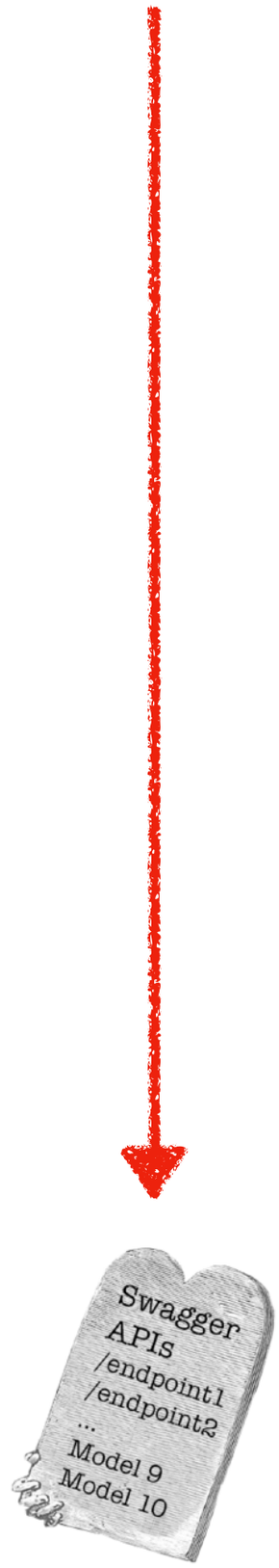
New feature/user story to implement over time

Usually new user story's implementation implies:

- Edit or write source code belonging to different layers of mobile application
- New services API
- New or new versions of domain data models



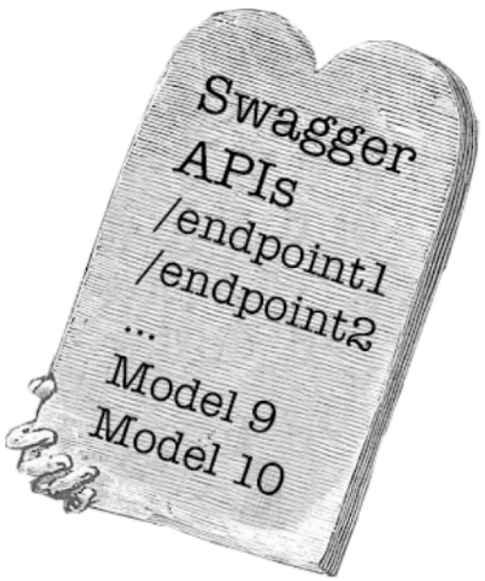
When a new feature is defined



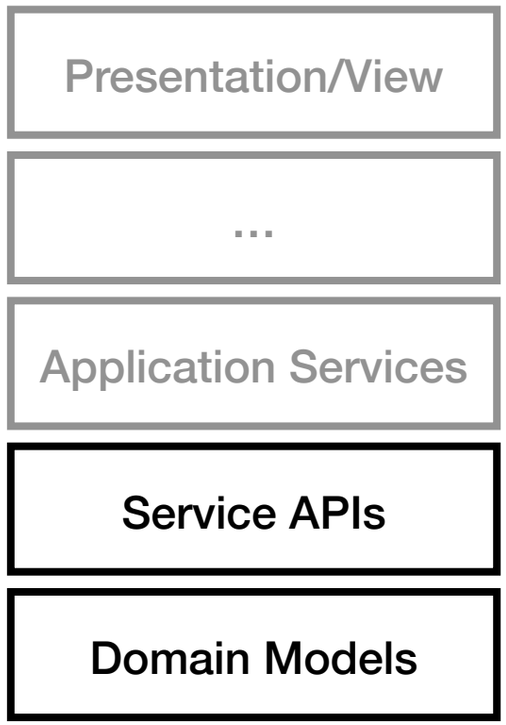
New Swagger’s version

Goal

Write minimum amount of code given new swagger version



Code Generation



Domain model

Code generation

```
public struct Pet: Codable, Hashable {

  public let _id: Int

  public let category: Category?

  public let name: String

  public let status: Status?

  public init(_id: Int, category: Category? = nil, name: String, status: Status? = nil) {
      self._id = _id
      self.category = category
      self.name = name
      self.status = status
  }

  public enum Status: String, Codable, Hashable, CaseIterable, {
      case available = "available"
      case pending = "pending"
      case sold = "sold"
  }

}
```

Service Api

Code generation

```
open class PetAPI: ReactiveAPI {

    public func addPet(body: Pet) -> Single<Void> {
        request(.post, url: absoluteURL(“/pet"), body: body)
    }

    public func getPetById(petId: Int) -> Single<Pet> {
        request(.get, url: absoluteURL("/pet/\(petId)"))
    }

    …
}
```

Mock

Code generation

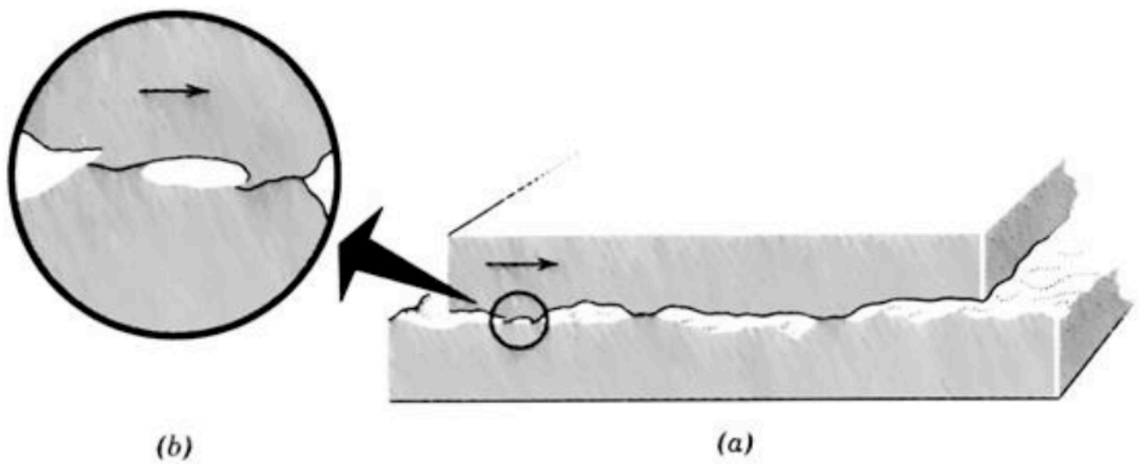
```
public extension Pet {

    static func mock(_id: Int = Int.mock(),
                     category: Category? = nil,
                     name: String = String.mock(),
                     status: Pet.Status? = nil) -> Pet {

        Pet(_id: _id, category: category, name: name, status: status)
    }

}


public extension Pet.Status {

    static func mock() -> Pet.Status {
       Pet.Status.allCases.randomElement()!
    }

}
```

#typesafe #mocks

#default #mandatory #fields #with #random #value

Goal

Reduce friction during user story's implementation



Test Driven Development

(Sky Test Foundation)

Developers don't wait for API implementation by *Backend Devs*

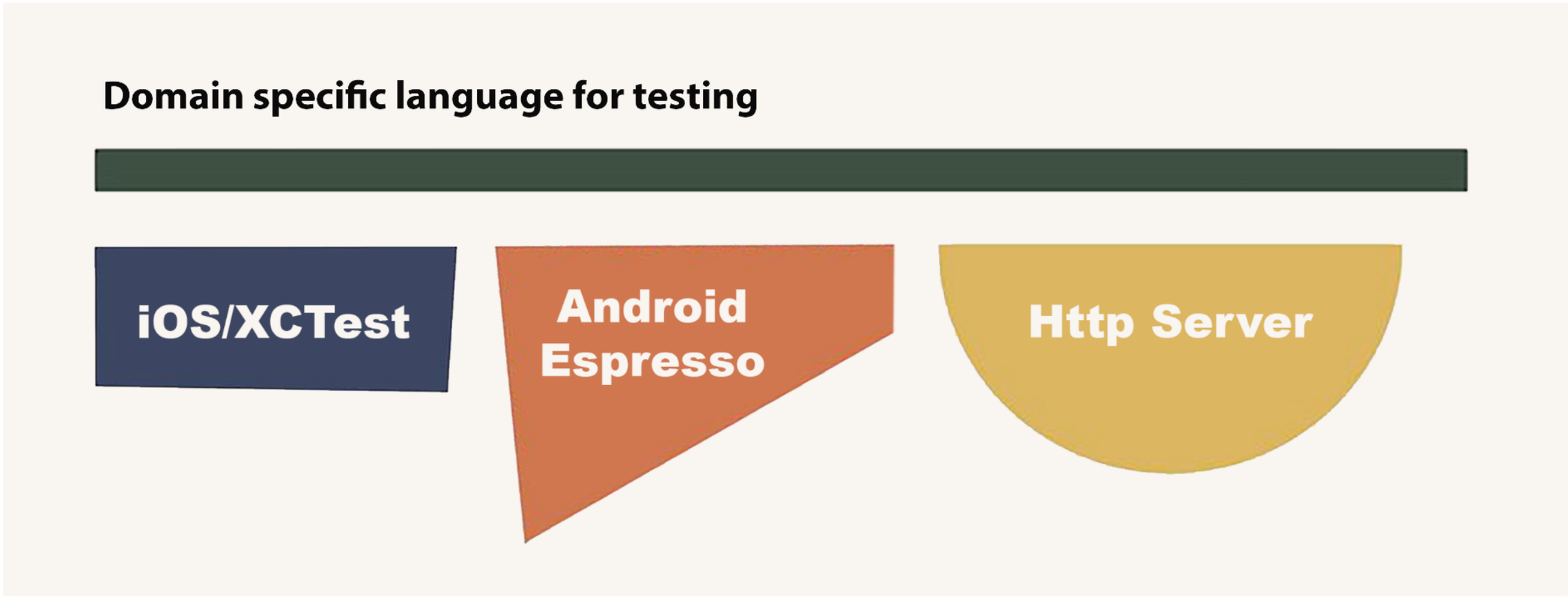
Goal

Prove implementation’s correctness

Test Campaign

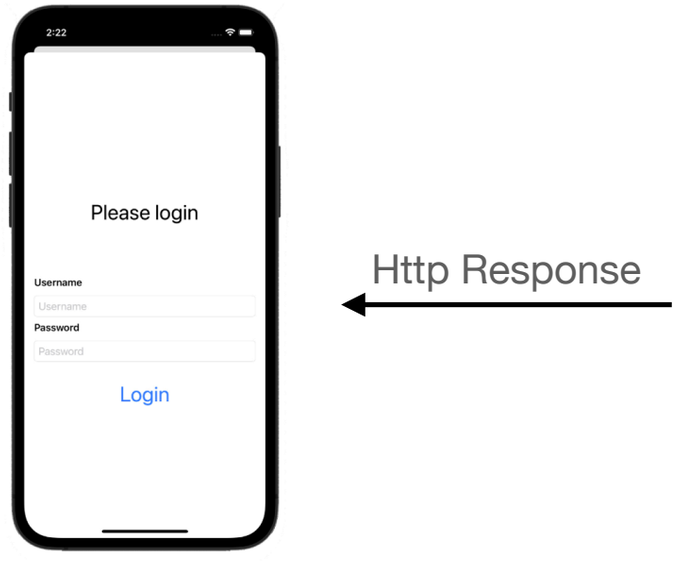
(Sky Test Foundation)

Sky Test Foundation

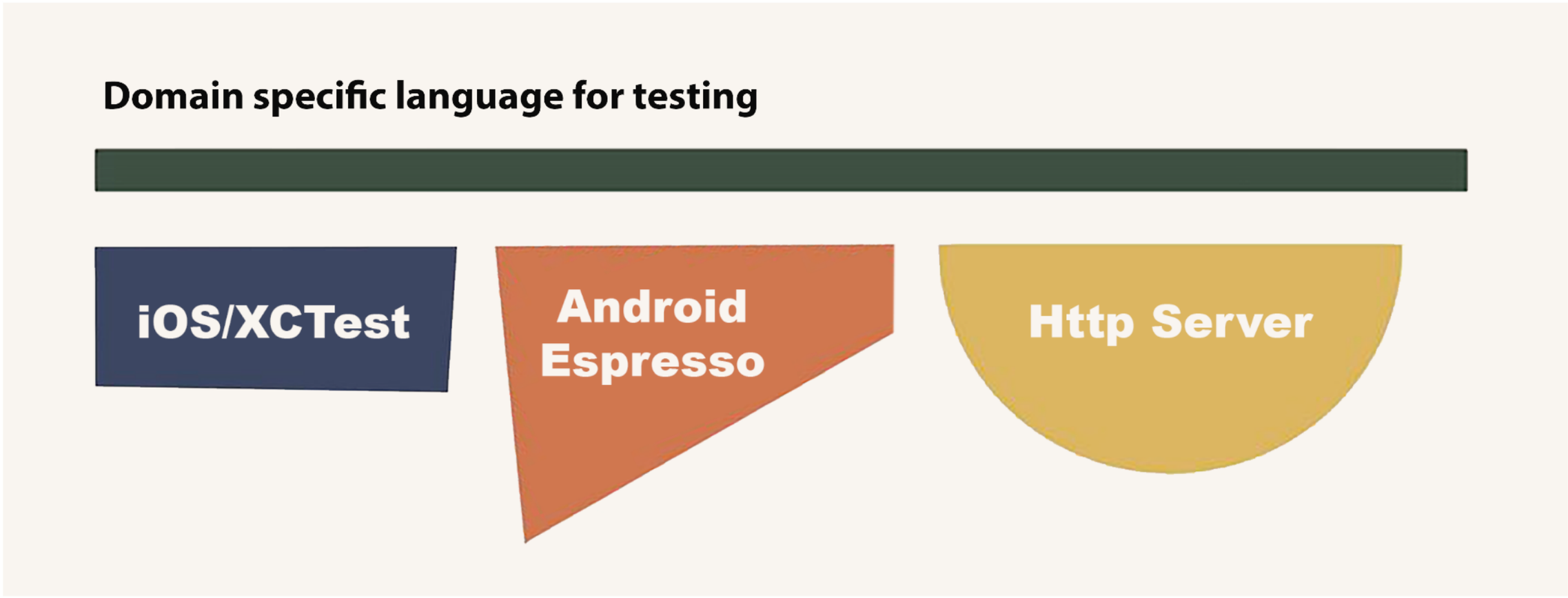


During test execution, Sky Test Foundation allows you to:

- **mock HTTP responses (received by the App)**
- define assertions on each HTTP request sent by mobile application
- assert UI elements existence in view hierarchy
- simulate user gestures

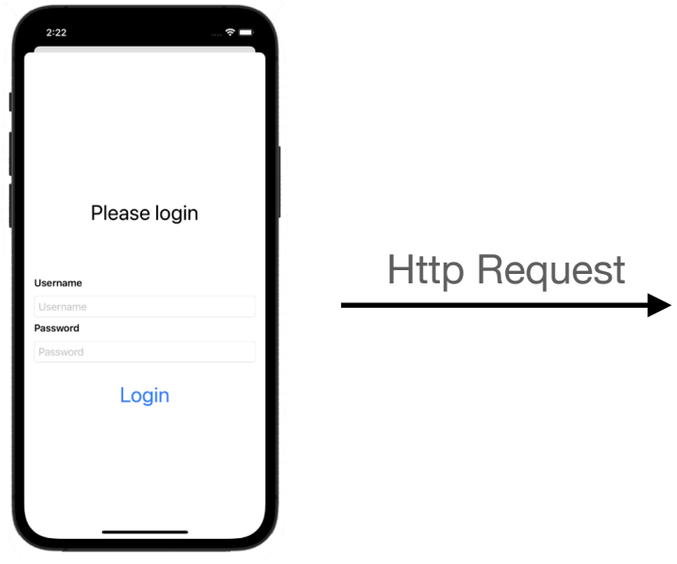


Sky Test Foundation

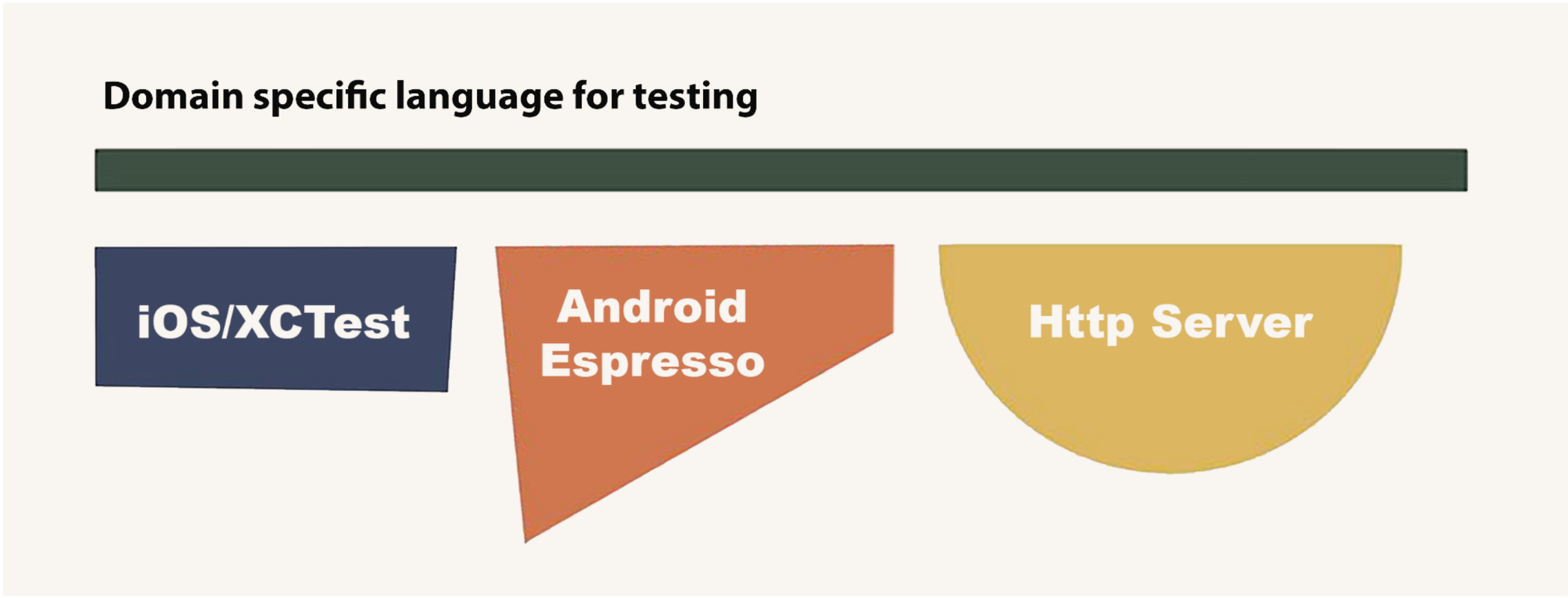


During test execution, Sky Test Foundation allows you to:

- mock HTTP responses (received by the App)
- **define assertions on each HTTP request sent by mobile application**
- assert UI elements existence in view hierarchy
- simulate user gestures

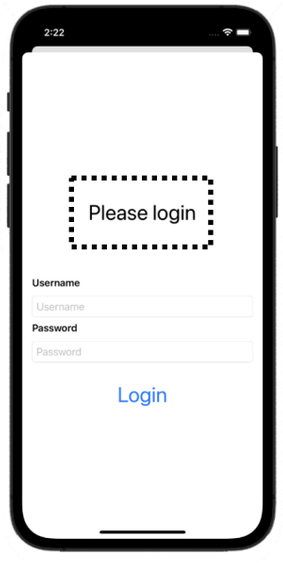


Sky Test Foundation

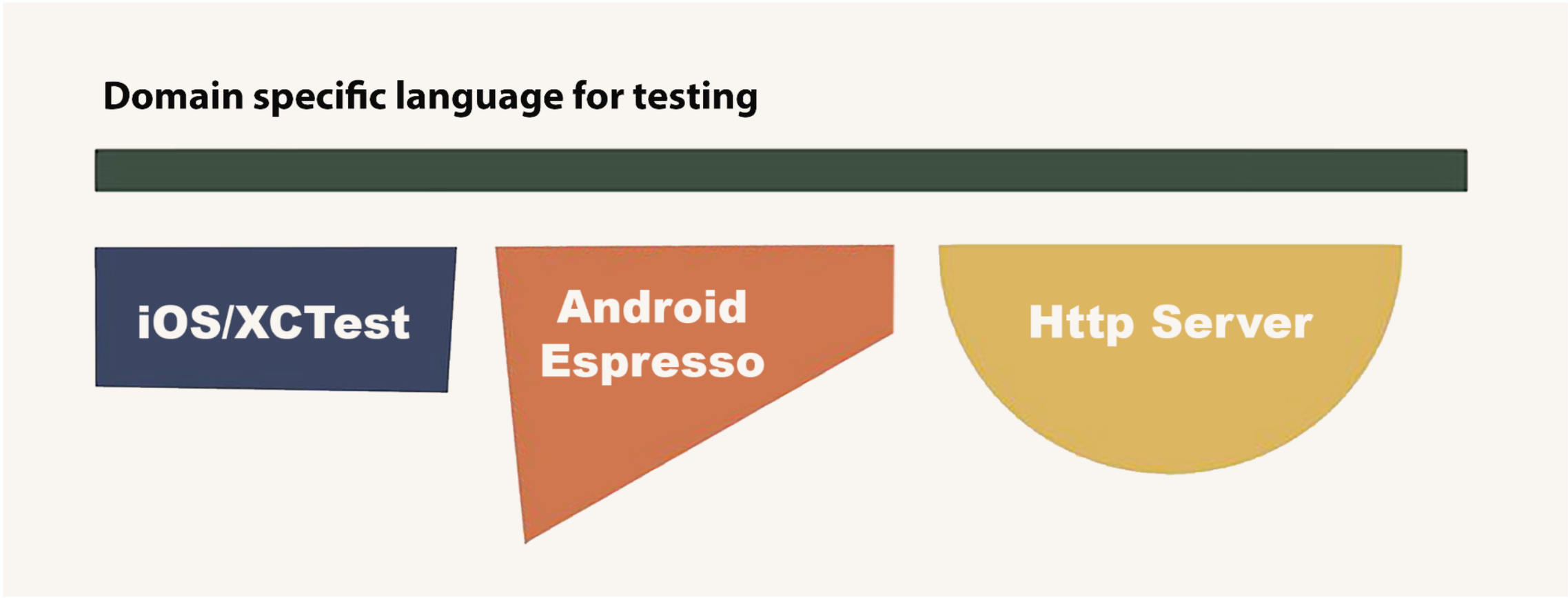


During test execution, Sky Test Foundation allows you to:

- mock HTTP responses (received by the App)
- define assertions on each HTTP request sent by mobile application
- **assert UI elements existence in view hierarchy**
- simulate user gestures

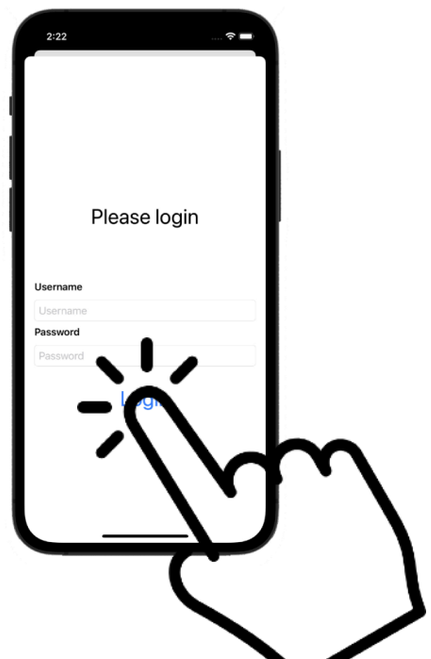


Sky Test Foundation



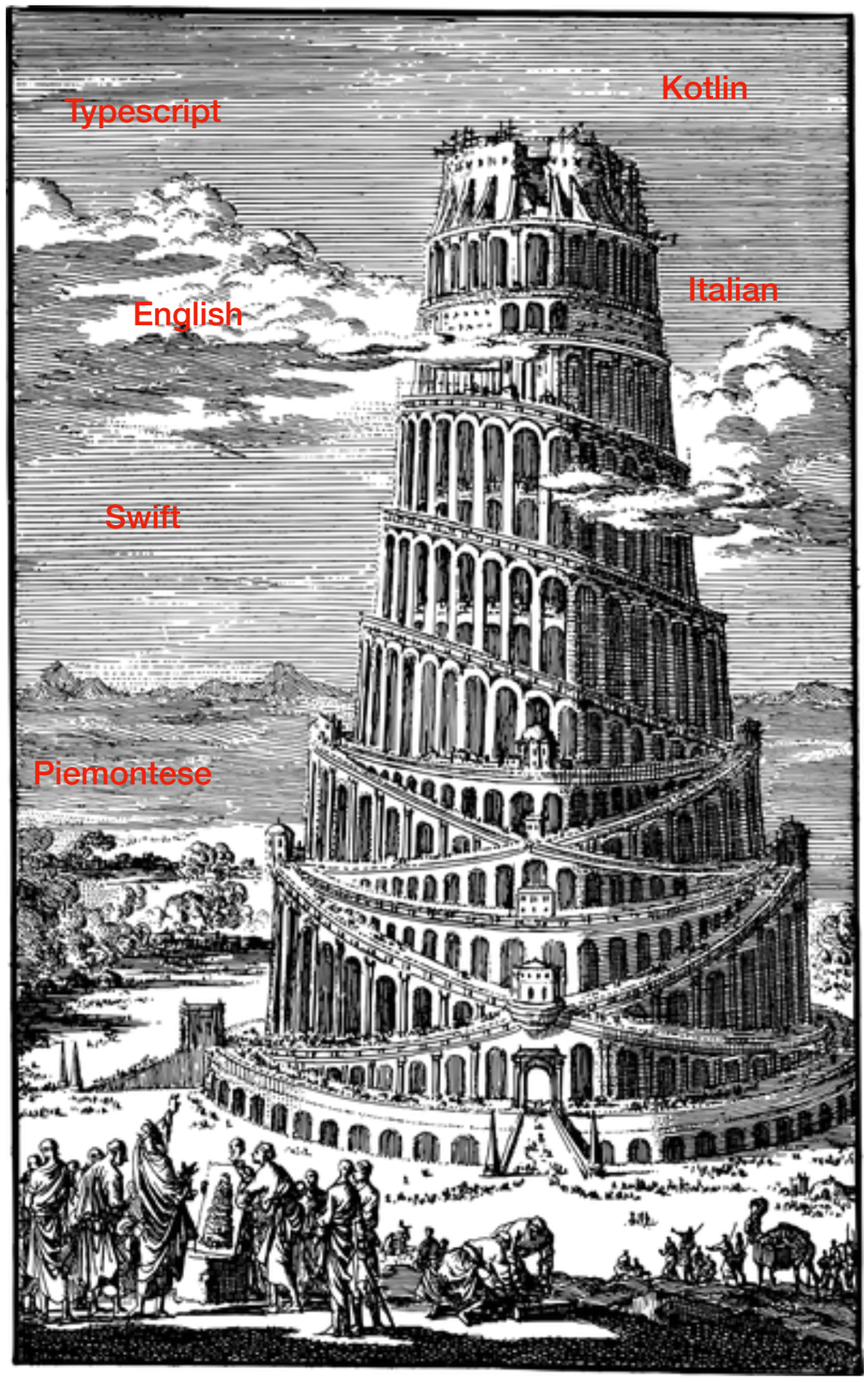
During test execution, Sky Test Foundation allows you to:

- mock HTTP responses (received by the App)
- define assertions on each HTTP request sent by mobile application
- assert UI elements existence in view hierarchy
- **simulate user gestures**



Write a test once

Run on different platforms (iOS/Android)



Tower of Babel

Only one language to describe an expected behaviour

Copy & Paste approach

Write test once

Write test in Swift/Xcode —> Copy in Kotlin/Android Studio

the other way around

Copy in Swift/Xcode <— Write test in Kotlin/Android

Domain specific language for testing

Copy & Paste

```swift
func testTapDetailThenValidUI() {
    // Given
    let tom = Pet.mock(name: "Tom")
    let jerry = Pet.mock(name: "Jerry")
    let pets = [jerry, tom]

    httpServerBuilder
            .route(MockResponses.User.successLogin())
            .route(MockResponses.Pet.findByStatus(pets: pets))
            .route(MockResponses.Pet.getPetById(tom))
            .buildAndStart()

    // When
    appLaunch()


    // Then
    exist(withTextEquals("Please login"))
    typeText(withTextInput("Username"), "Alessandro")
    typeText(withTextInput("Password"), "Secret")
    tap(withButton("Login"))

    tap(withTextEquals(tom.name))
}
```

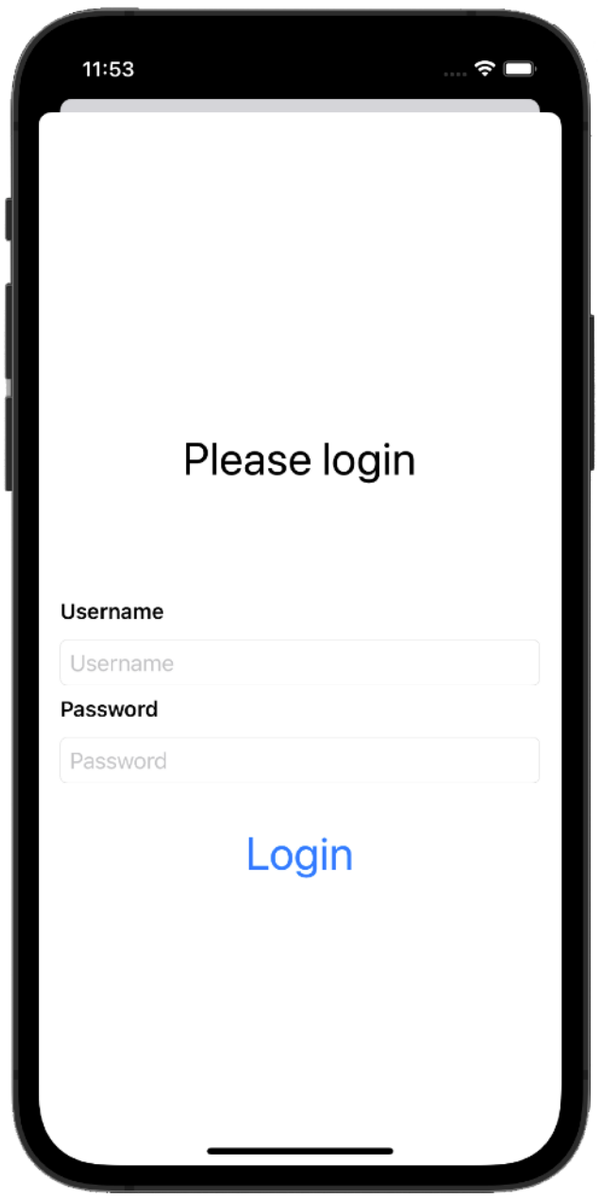
Swift

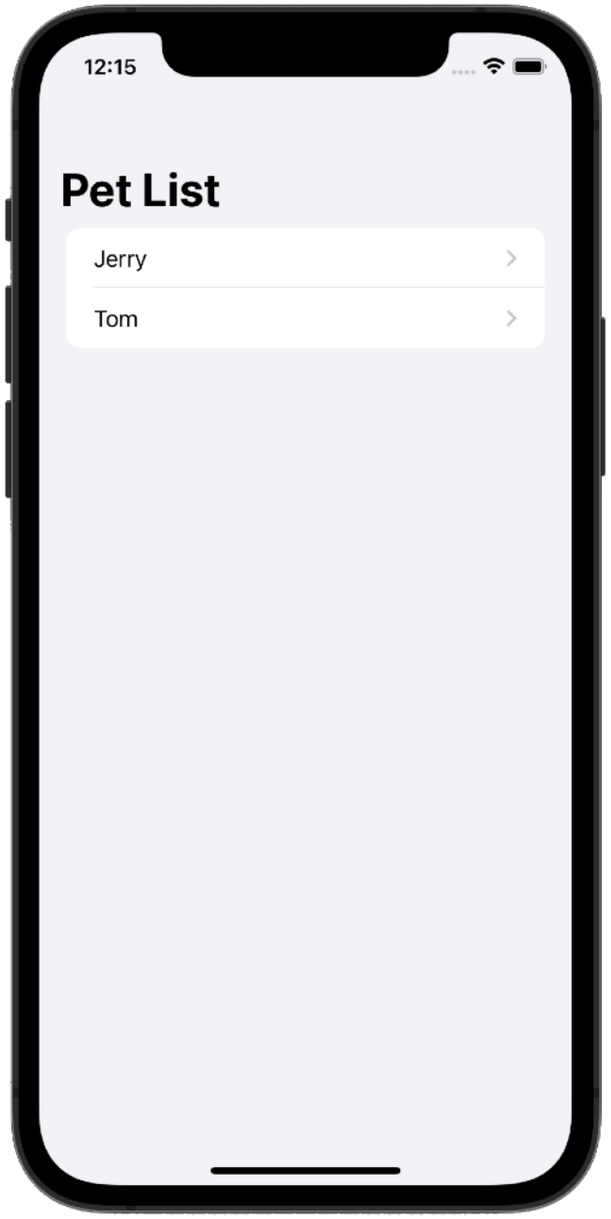
```kotlin
@Test
fun testTapDetailThenValidUI() {
    // Given
    val tom = Pet.mock(name = "Tom")
    val jerry = Pet.mock(name = "Jerry")
    val pets = arrayOf(jerry, tom)

    httpServerBuilder
        .route(MockResponses.User.successLogin())
        .route(MockResponses.Pet.findByStatus(pets = pets))
        .route(MockResponses.Pet.getPetById(tom))
        .buildAndStart()

    // When
    appLaunch()


    // Then
    exist(withTextEquals("Please login"))
    typeText(withTextInput("Username"), "Alessandro")
    typeText(withTextInput("Password"), "Secret")
    tap(withButton("Login"))

    tap(withTextEquals(tom.name))
}
```

Kotlin

A user story - Example 1

```
As User
I want to login
So that
I can see a list of available pets
```





From User Story definition —> Implementation & Test

```
As User
I want to login
So that
I can see a list of available pets
```

Swagger

API
/user/login
/pet/findByStatus

Model
Pet
ApiResponse

Service API

```
open class PetAPI: ReactiveAPI {
  public func findByStatus() -> [Pet] {

  }

}
```

Domain Models

```
public struct Pet {

}
```

Mocks

```
public extension Pet {
    static func mock() -> Pet {

    }
}
```

Generated code

Presentation/View

...

Application Services

Service APIs

Domain Models

Mobile Application Target

Http Routes

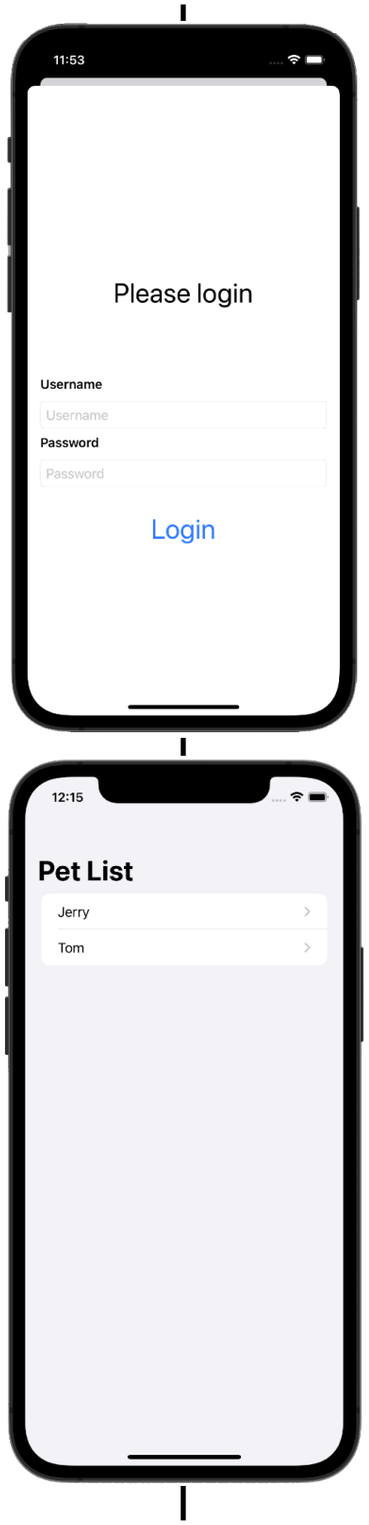
Mocks

SkyTestFoundation

Test Target

User story details…

"As User I want to login
So that I can see a list of available pets"



Mobile Application

type text "Alessandro" into "Username" text field

type text "Secret" into "Password" text field

tap button "Login"

/user/login

ApiResponse(code = 200)

/pet/findByStatus

[Pets]

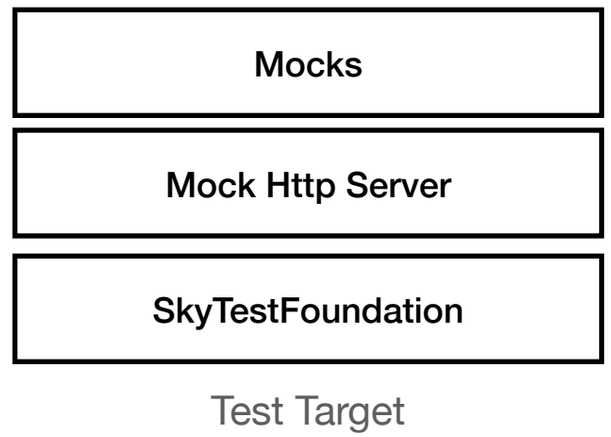
Assert the existence of tom.name text

Mock Server

UI Test using Sky Test Foundation

Domain specific language for testing

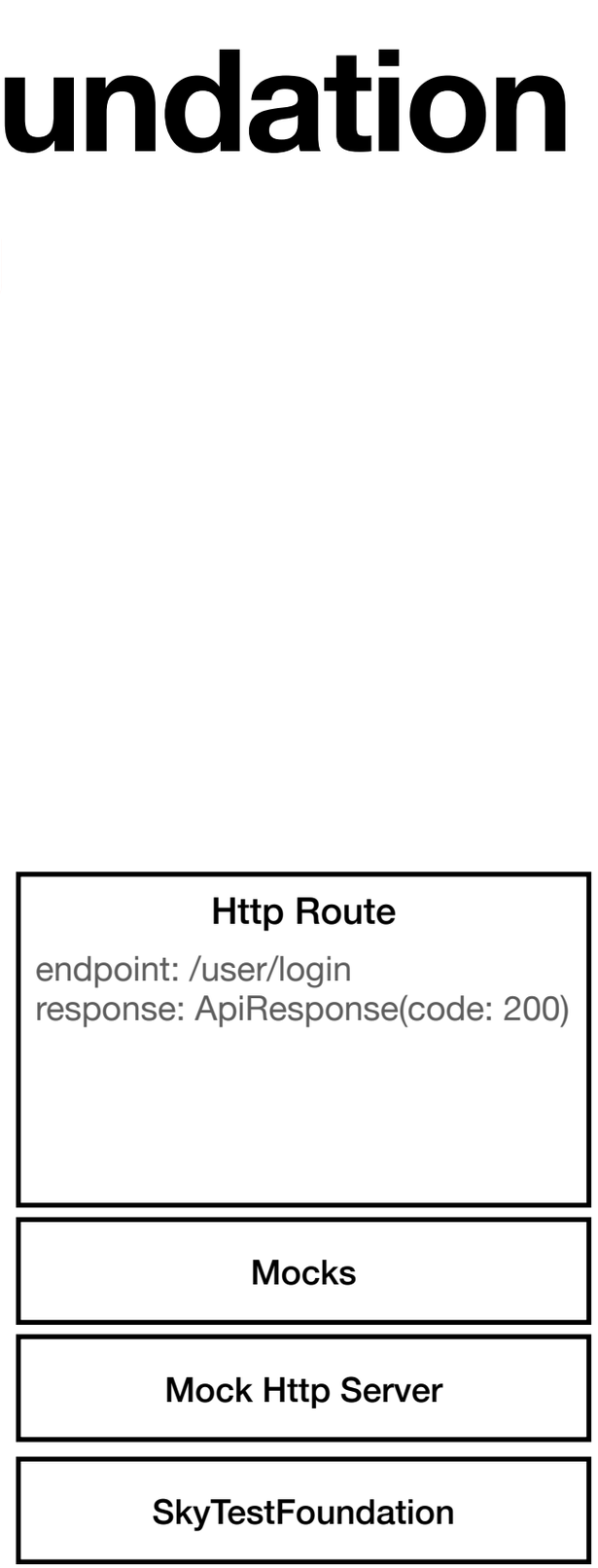
```
func testTapDetailThenValidUI() {

}
```



UI Test using Sky Test Foundation

Domain specific language for testing

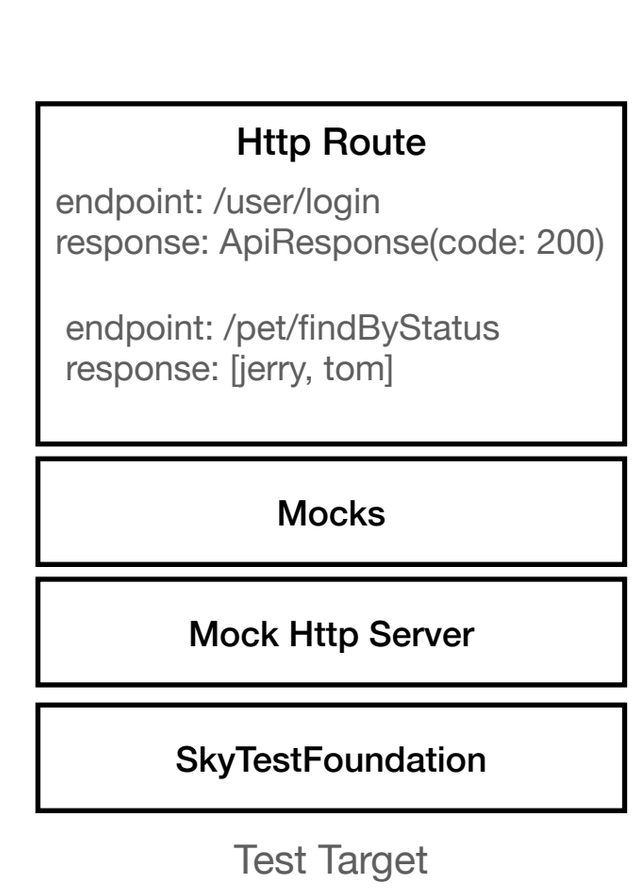
```
func testTapDetailThenValidUI() {
  // Given
  let tom = Pet.mock(name: "Tom")
  let jerry = Pet.mock(name: "Jerry")
  let pets = [jerry, tom]
  httpServerBuilder
      .route(MockResponses.User.successLogin())



}
```



UI Test using Sky Test Foundation

Domain specific language for testing

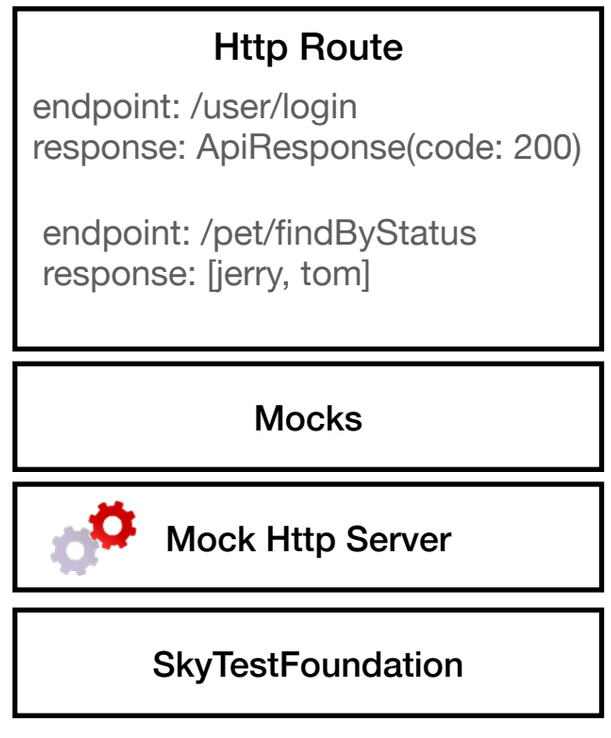
```
func testTapDetailThenValidUI() {
  // Given
  let tom = Pet.mock(name: "Tom")
  let jerry = Pet.mock(name: "Jerry")
  let pets = [jerry, tom]
  httpServerBuilder
     .route(MockResponses.User.successLogin())
     .route(MockResponses.Pet.findByStatus(pets: pets))
}
```



UI Test using Sky Test Foundation

Domain specific language for testing

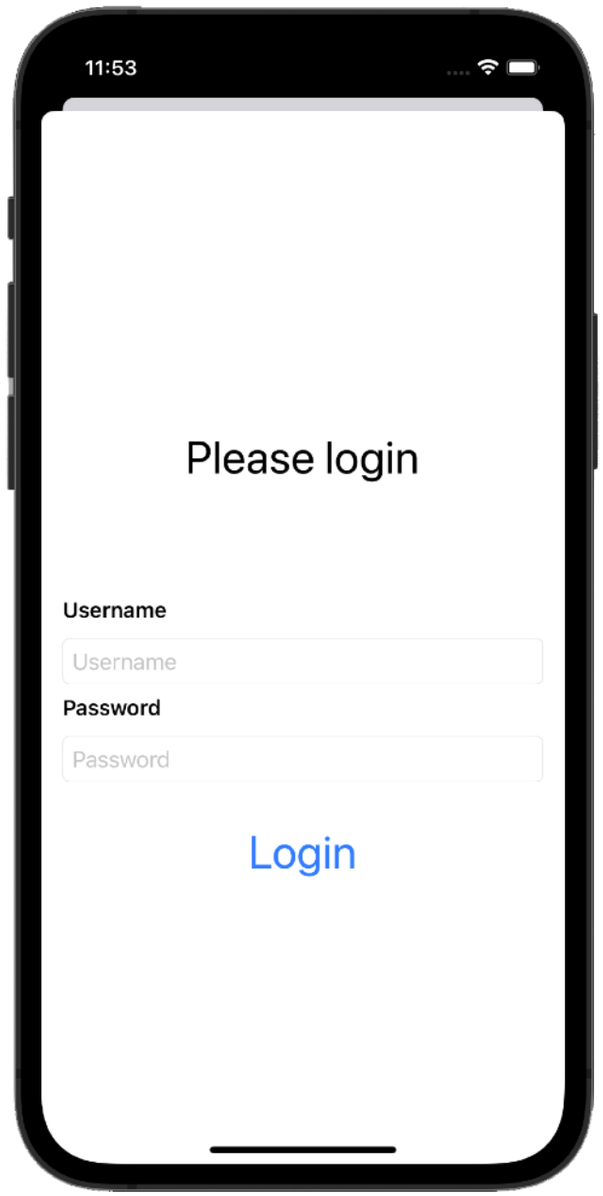
```
func testTapDetailThenValidUI() {
  // Given
  let tom = Pet.mock(name: "Tom")
  let jerry = Pet.mock(name: "Jerry")
  let pets = [jerry, tom]
  httpServerBuilder
     .route(MockResponses.User.successLogin())
     .route(MockResponses.Pet.findByStatus(pets: pets))
     .buildAndStart()
}
```



UI Test using Sky Test Foundation

Domain specific language for testing

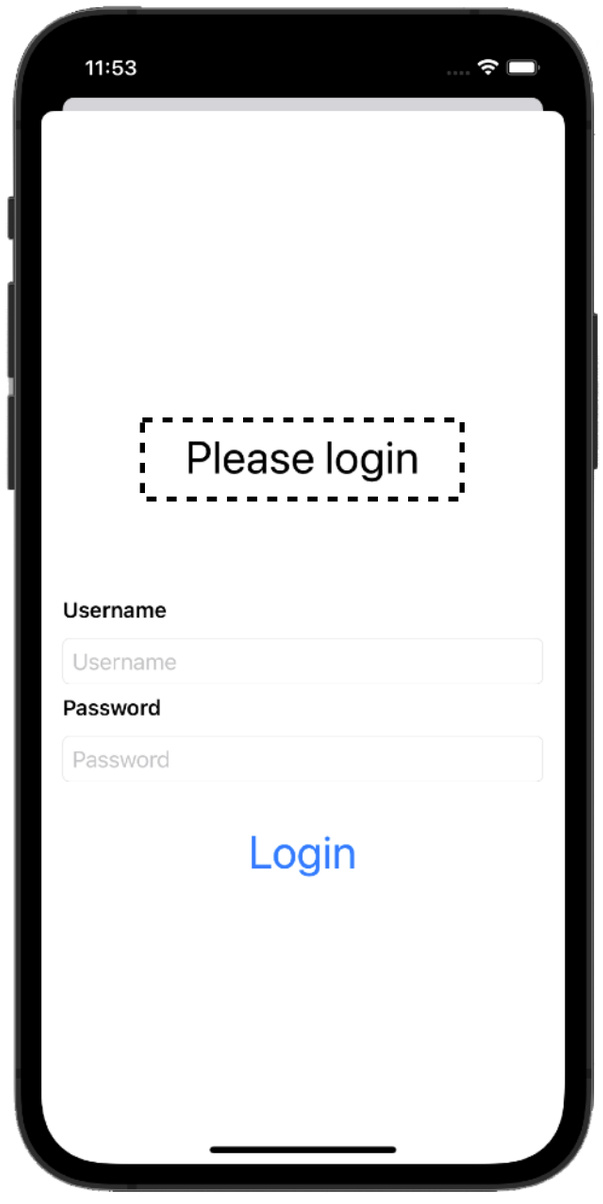
```
func testTapDetailThenValidUI() {
  // Given
  let tom = Pet.mock(name: "Tom")
  let jerry = Pet.mock(name: "Jerry")
  let pets = [jerry, tom]
  httpServerBuilder
     .route(MockResponses.User.successLogin())
     .route(MockResponses.Pet.findByStatus(pets: pets))
     .buildAndStart()

  // When
  appLaunch()


}
```



UI Test using Sky Test Foundation

Domain specific language for testing

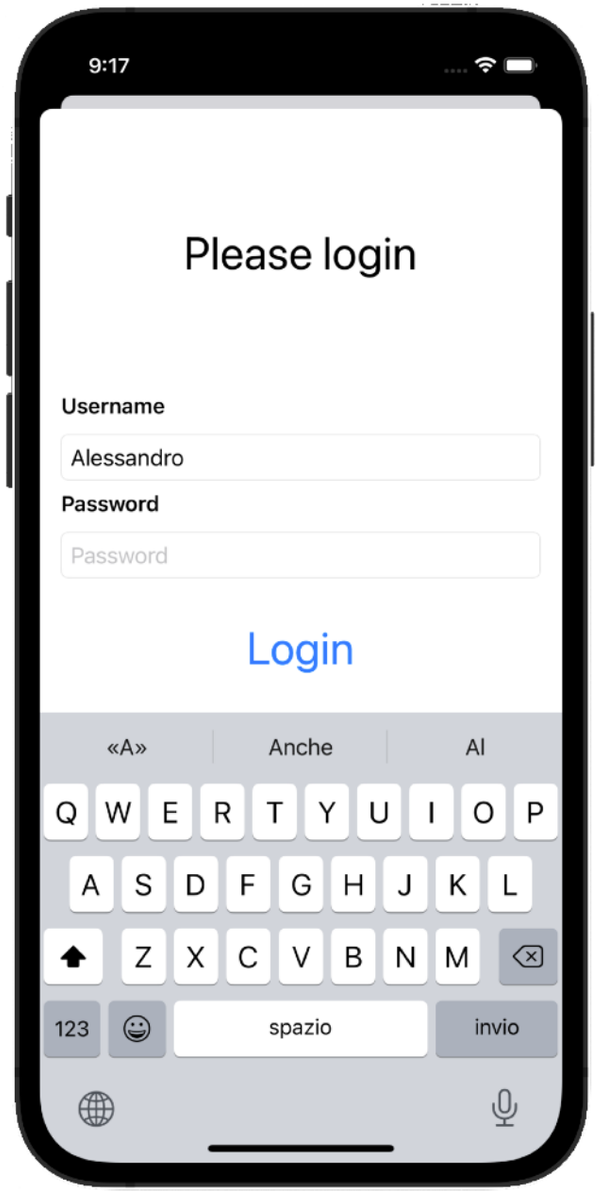
```
func testTapDetailThenValidUI() {
  // Given
  let tom = Pet.mock(name: "Tom")
  let jerry = Pet.mock(name: "Jerry")
  let pets = [jerry, tom]
  httpServerBuilder
     .route(MockResponses.User.successLogin())
     .route(MockResponses.Pet.findByStatus(pets: pets))
     .buildAndStart()

  // When
  appLaunch()

  // Then
  exist(withTextEquals("Please login"))



}
```



UI Test using Sky Test Foundation

Domain specific language for testing

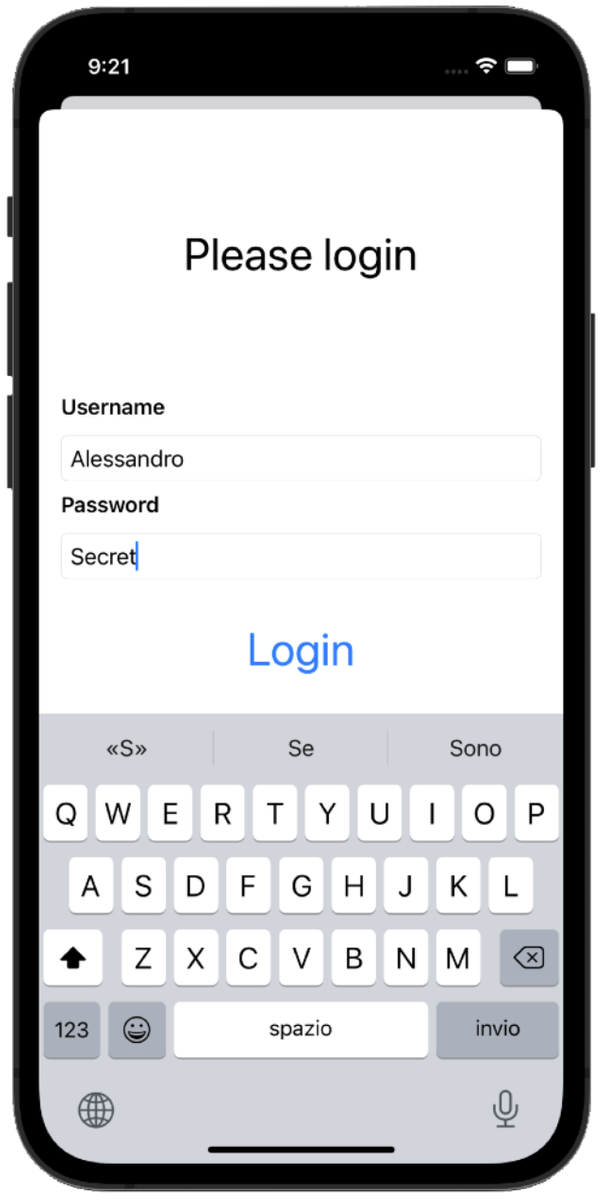
```
func testTapDetailThenValidUI() {
  // Given
  let tom = Pet.mock(name: "Tom")
  let jerry = Pet.mock(name: "Jerry")
  let pets = [jerry, tom]
  httpServerBuilder
     .route(MockResponses.User.successLogin())
     .route(MockResponses.Pet.findByStatus(pets: pets))
     .buildAndStart()

  // When
  appLaunch()

  // Then
  exist(withTextEquals("Please login"))
  typeText(withTextInput("Username"), "Alessandro")



}
```



UI Test using Sky Test Foundation

Domain specific language for testing

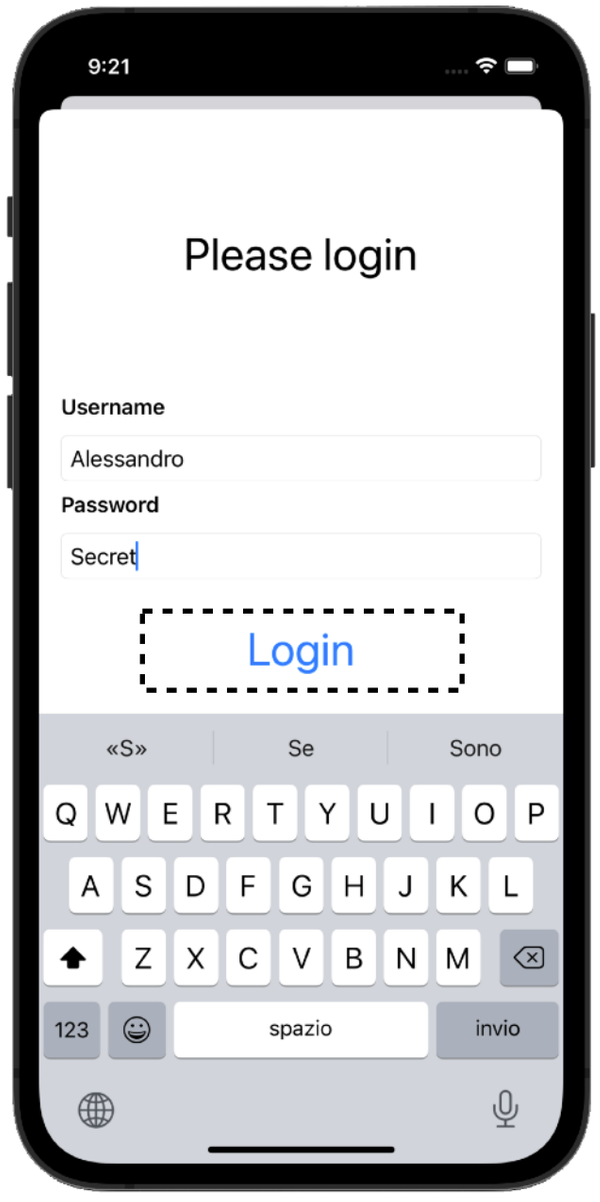
```
func testTapDetailThenValidUI() {
  // Given
  let tom = Pet.mock(name: "Tom")
  let jerry = Pet.mock(name: "Jerry")
  let pets = [jerry, tom]
  httpServerBuilder
     .route(MockResponses.User.successLogin())
     .route(MockResponses.Pet.findByStatus(pets: pets))
     .buildAndStart()

  // When
  appLaunch()

  // Then
  exist(withTextEquals("Please login"))
  typeText(withTextInput("Username"), "Alessandro")
  typeText(withSecureTextInput("Password"), "Secret")


}
```



UI Test using Sky Test Foundation

Domain specific language for testing

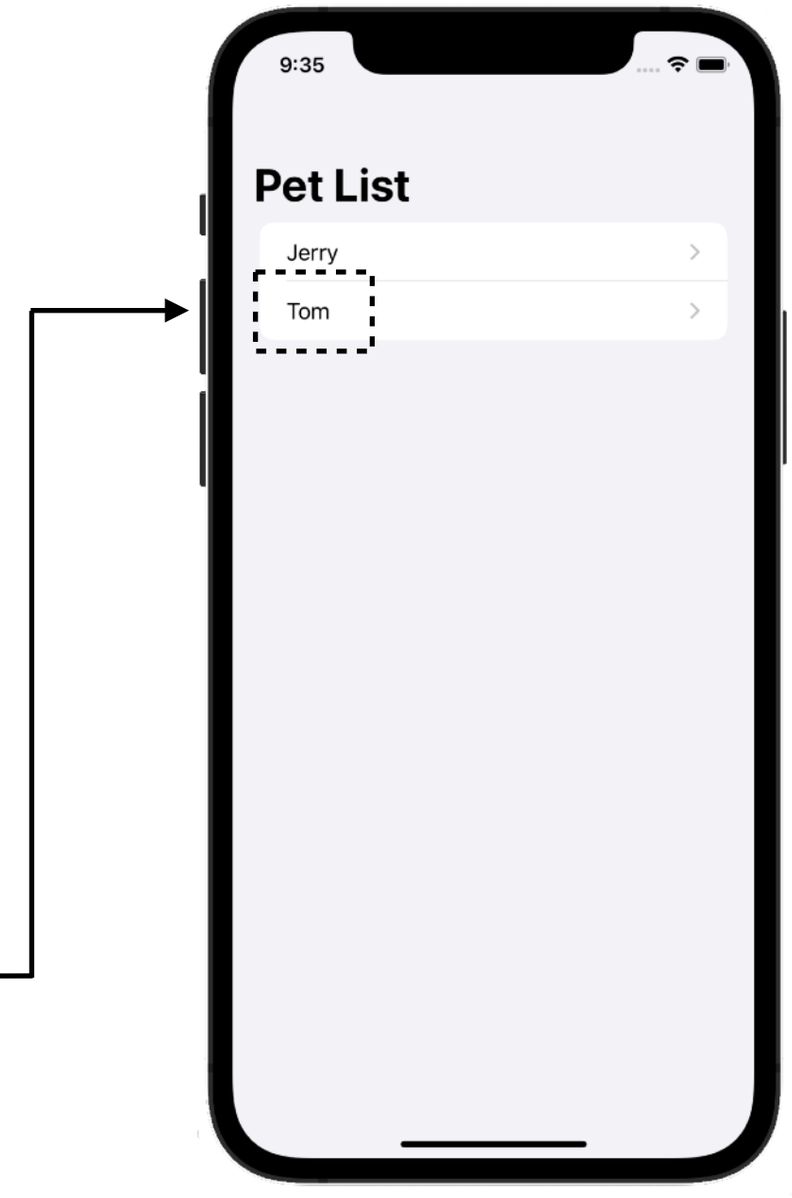
```
func testTapDetailThenValidUI() {
  // Given
  let tom = Pet.mock(name: "Tom")
  let jerry = Pet.mock(name: "Jerry")
  let pets = [jerry, tom]
  httpServerBuilder
     .route(MockResponses.User.successLogin())
     .route(MockResponses.Pet.findByStatus(pets: pets))
     .buildAndStart()

  // When
  appLaunch()

  // Then
  exist(withTextEquals("Please login"))
  typeText(withTextInput("Username"), "Alessandro")
  typeText(withSecureTextInput("Password"), "Secret")
  tap(withButton("Login"))


}
```



UI Test using Sky Test Foundation

Domain specific language for testing

```
func testTapDetailThenValidUI() {
  // Given
  let tom = Pet.mock(name: "Tom")
  let jerry = Pet.mock(name: "Jerry")
  let pets = [jerry, tom]
  httpServerBuilder
     .route(MockResponses.User.successLogin())
     .route(MockResponses.Pet.findByStatus(pets: pets))
     .buildAndStart()

  // When
  appLaunch()

  // Then
  exist(withTextEquals("Please login"))
  typeText(withTextInput("Username"), "Alessandro")
  typeText(withSecureTextInput("Password"), "Secret")
  tap(withButton("Login"))

  exist(withTextEquals(tom.name))
}
```



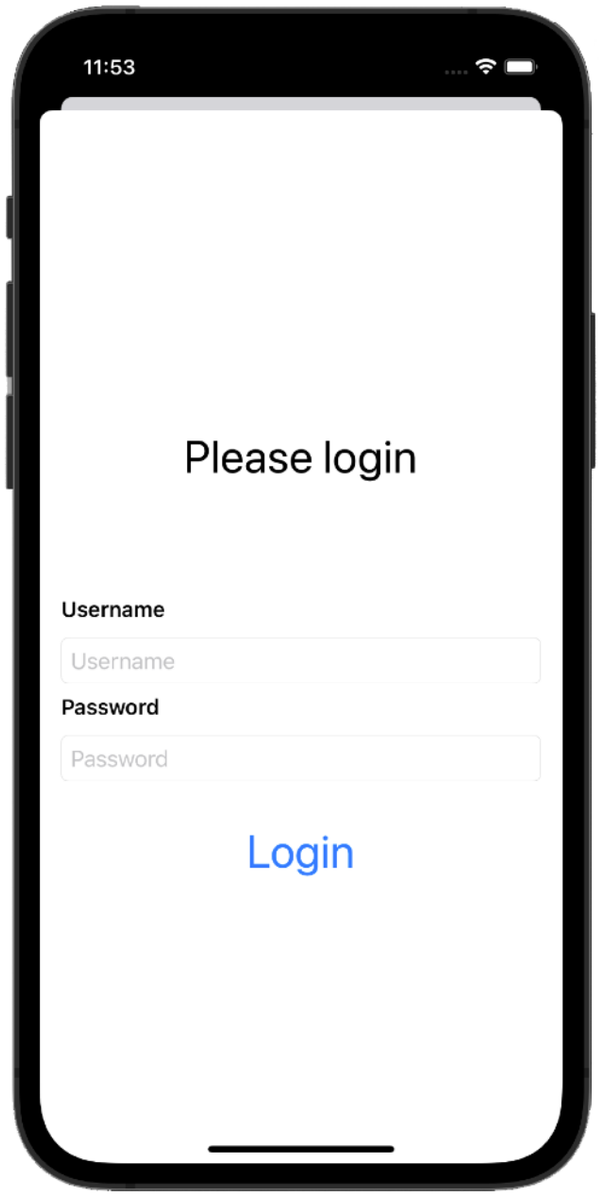
UI Test using Sky Test Foundation

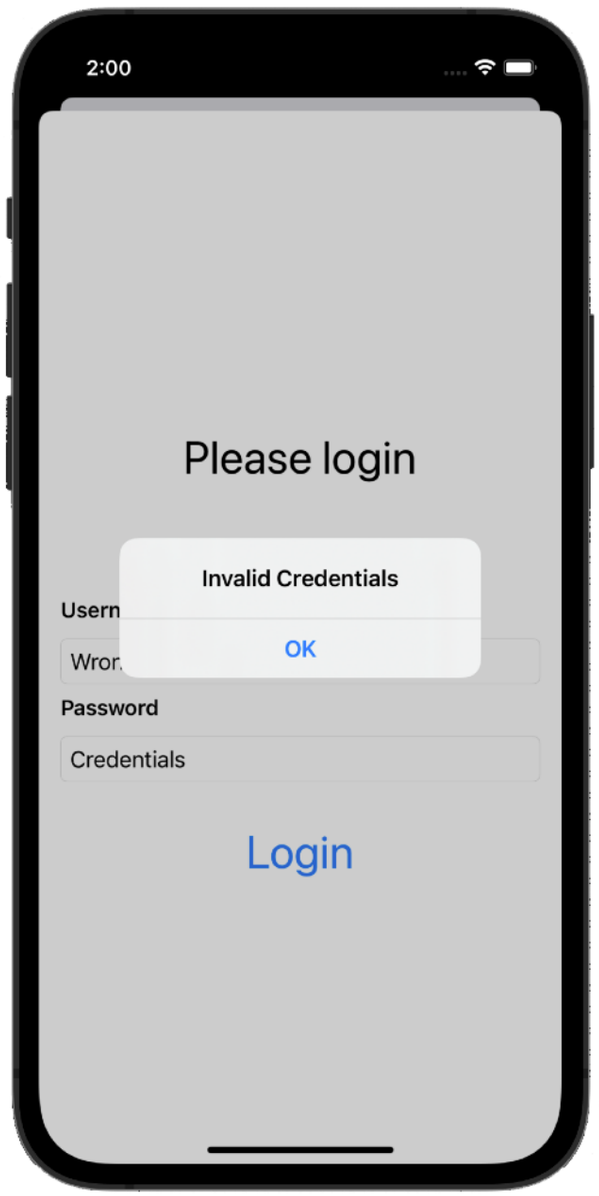
Domain specific language for testing

```
func testTapDetailThenValidUI() {
    // Given
    let tom = Pet.mock(name: "Tom")
    let jerry = Pet.mock(name: "Jerry")
    let pets = [jerry, tom]
    httpServerBuilder
        .route(MockResponses.User.successLogin())
        .route(MockResponses.Pet.findByStatus(pets: pets))
        .buildAndStart()

    // When
    appLaunch()

    // Then
    exist(withTextEquals("Please login"))
    typeText(withTextInput("Username"), "Alessandro")
    typeText(withSecureTextInput("Password"), "Secret")
    tap(withButton("Login"))

    exist(withTextEquals(tom.name))
}
```

A user story - Example 2

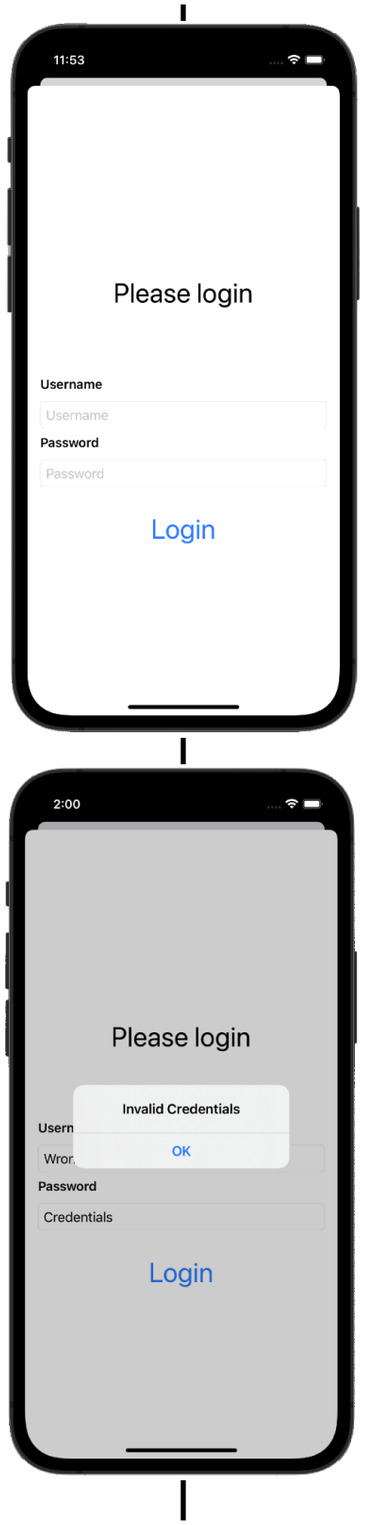
```
As User
I want to type invalid credentials
So that
I can see “Invalid Credentials” alert
```





User story details…

“As User I want to type invalid credentials
So that I can see “Invalid Credentials” alert”



type text “Wrong” into “Username" text field

type text “Credentials" into “Password" text field

tap button "Login"

/user/login

ApiResponse(code = **404**)

Assert the existence of “Invalid Credentials” text

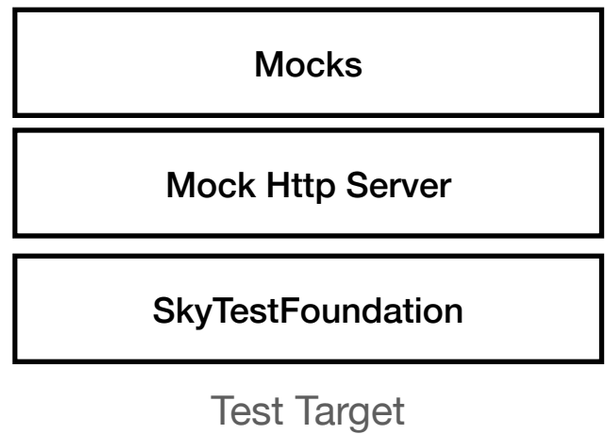
Mobile Application

Mock Server

UI Test using Sky Test Foundation

Domain specific language for testing

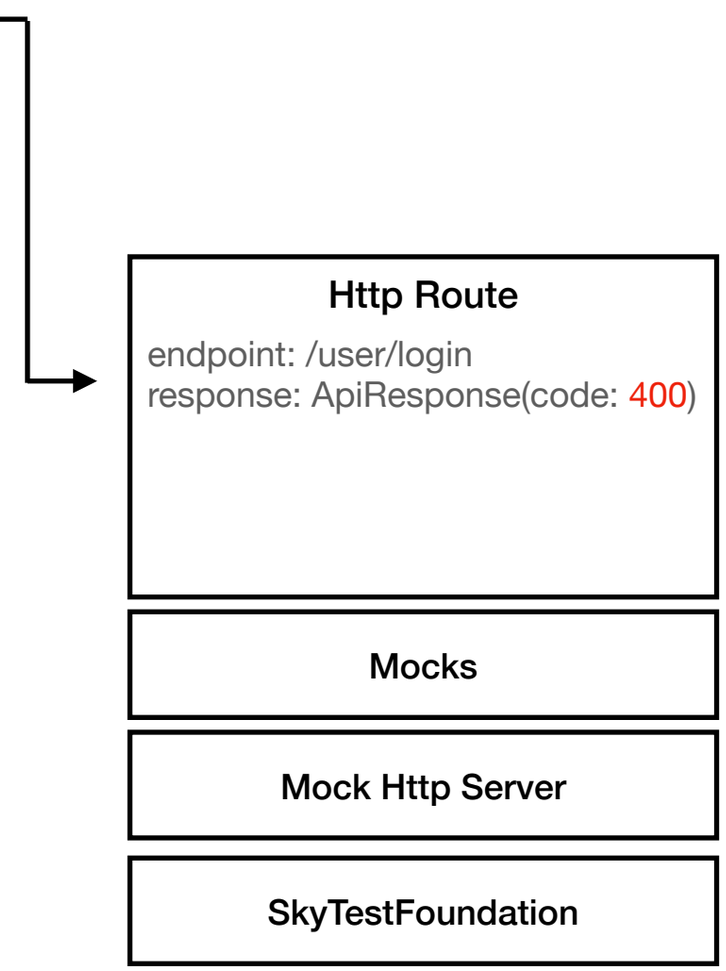
```
func testLoginGivenInvalidCredentials() {



}
```



UI Test using Sky Test Foundation

Domain specific language for testing

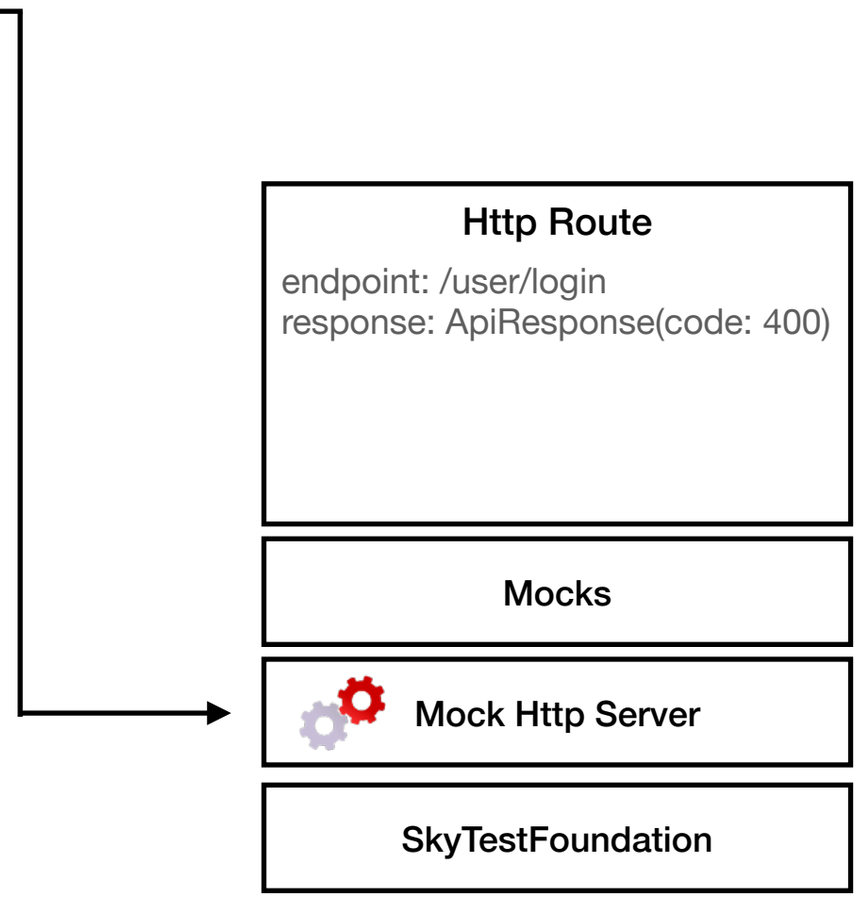
```
func testLoginGivenInvalidCredentials() {
    // Given
    httpServerBuilder
        .route(MockResponses.User.unauthorizedLogin())

}
```



UI Test using Sky Test Foundation

Domain specific language for testing

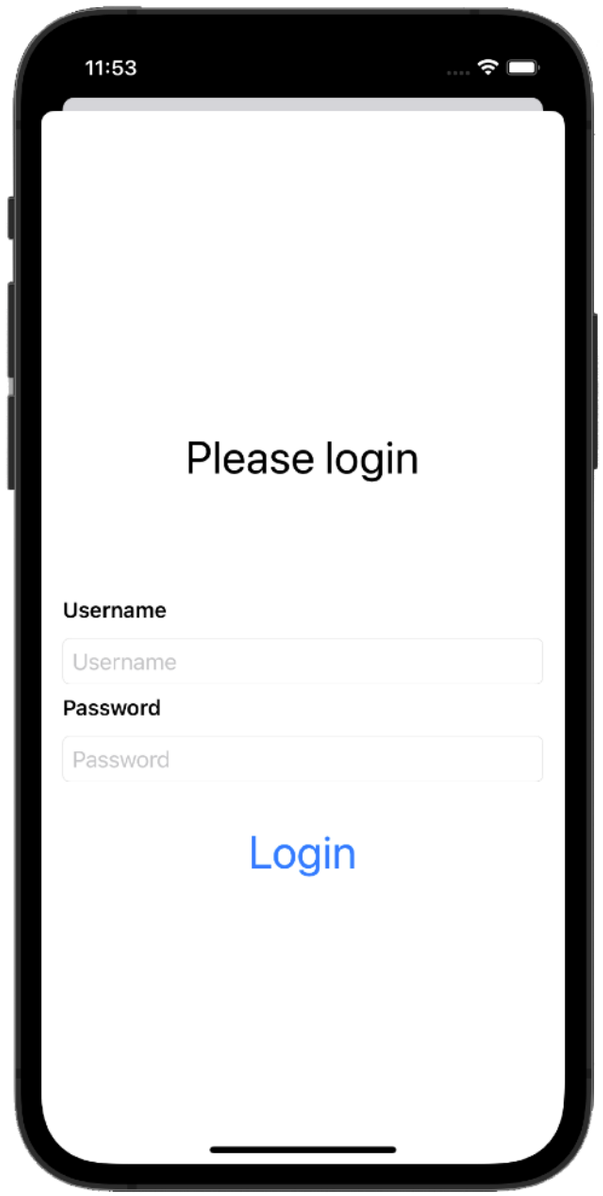
```
func testLoginGivenInvalidCredentials() {
    // Given
    httpServerBuilder
        .route(MockResponses.User.unauthorizedLogin())
        .buildAndStart()

}
```



UI Test using Sky Test Foundation

Domain specific language for testing

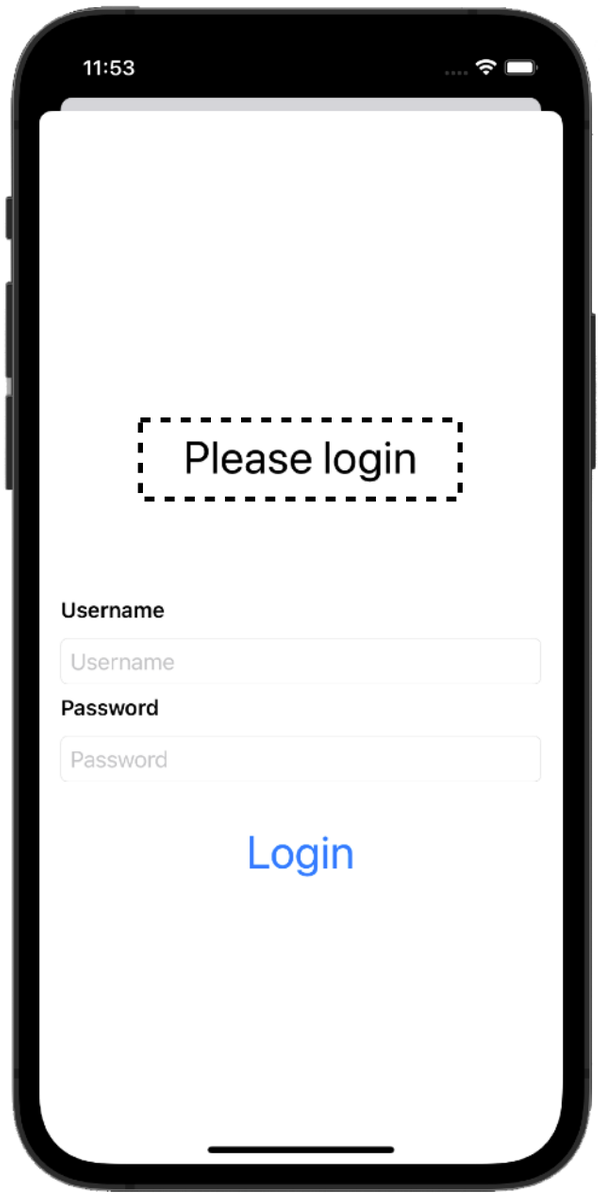
```
func testLoginGivenInvalidCredentials() {
    // Given
    httpServerBuilder
        .route(MockResponses.User.unauthorizedLogin())
        .buildAndStart()

    // When
    appLaunch()

}
```



UI Test using Sky Test Foundation

Domain specific language for testing

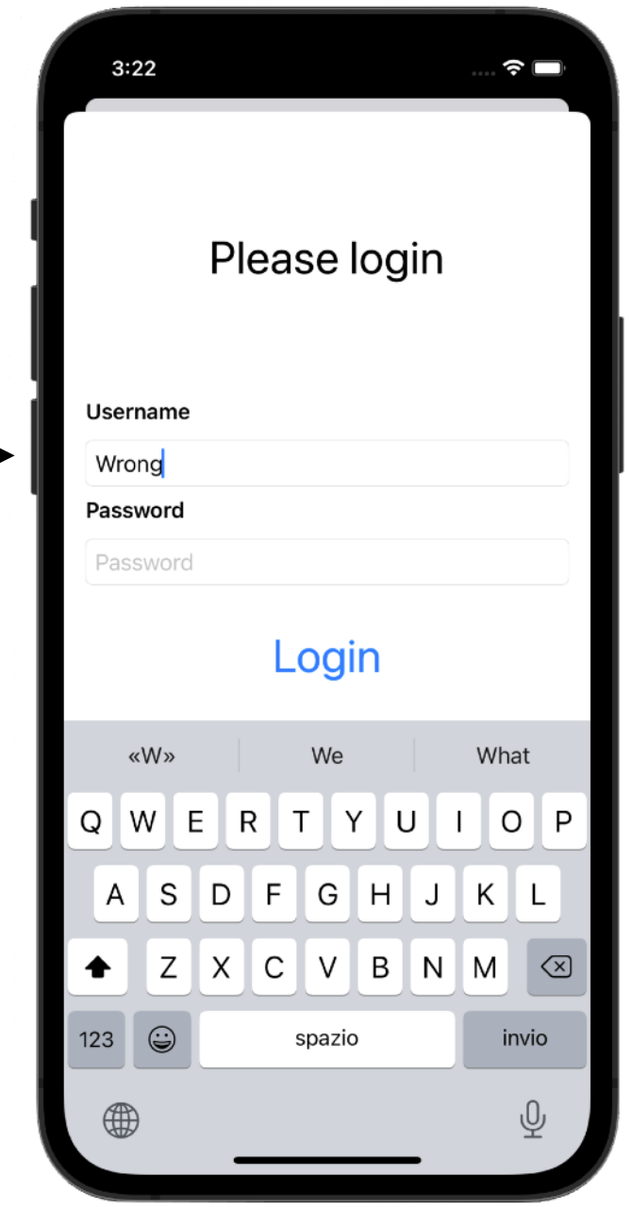
```
func testLoginGivenInvalidCredentials() {
    // Given
    httpServerBuilder
        .route(MockResponses.User.unauthorizedLogin())
        .buildAndStart()

    // When
    appLaunch()

    // Then
    exist(withTextEquals("Please login"))



}
```



UI Test using Sky Test Foundation

Domain specific language for testing

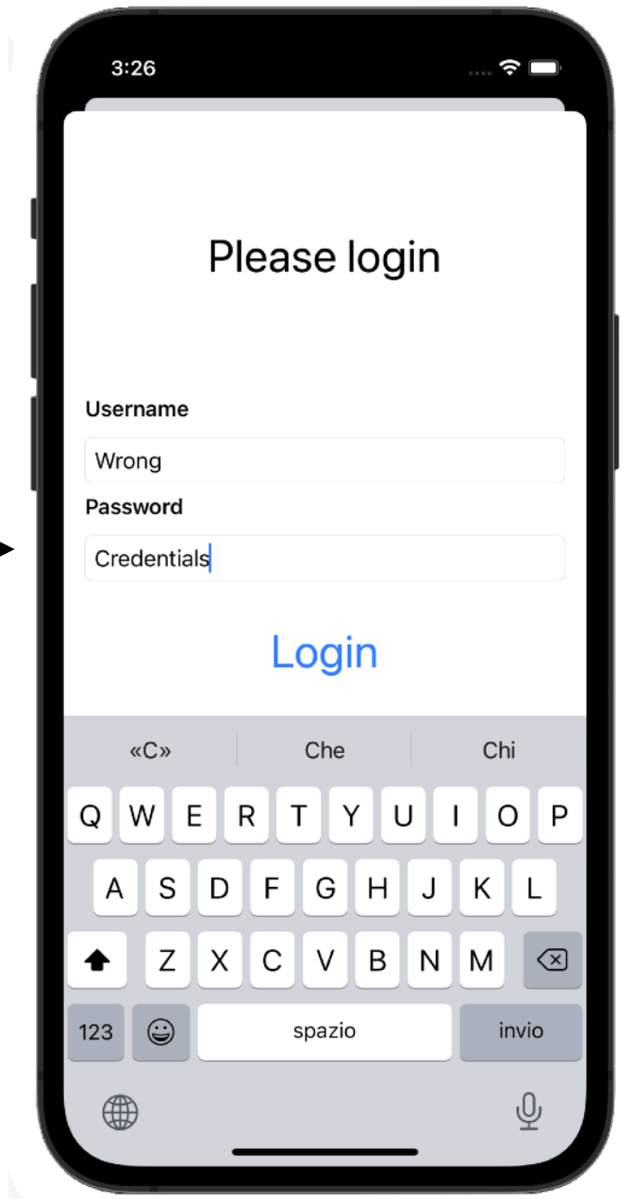
```
func testLoginGivenInvalidCredentials() {
    // Given
    httpServerBuilder
        .route(MockResponses.User.unauthorizedLogin())
        .buildAndStart()

    // When
    appLaunch()

    // Then
    exist(withTextEquals("Please login"))
    typeText(withTextInput("Username"), "Wrong")



}
```



UI Test using Sky Test Foundation

Domain specific language for testing

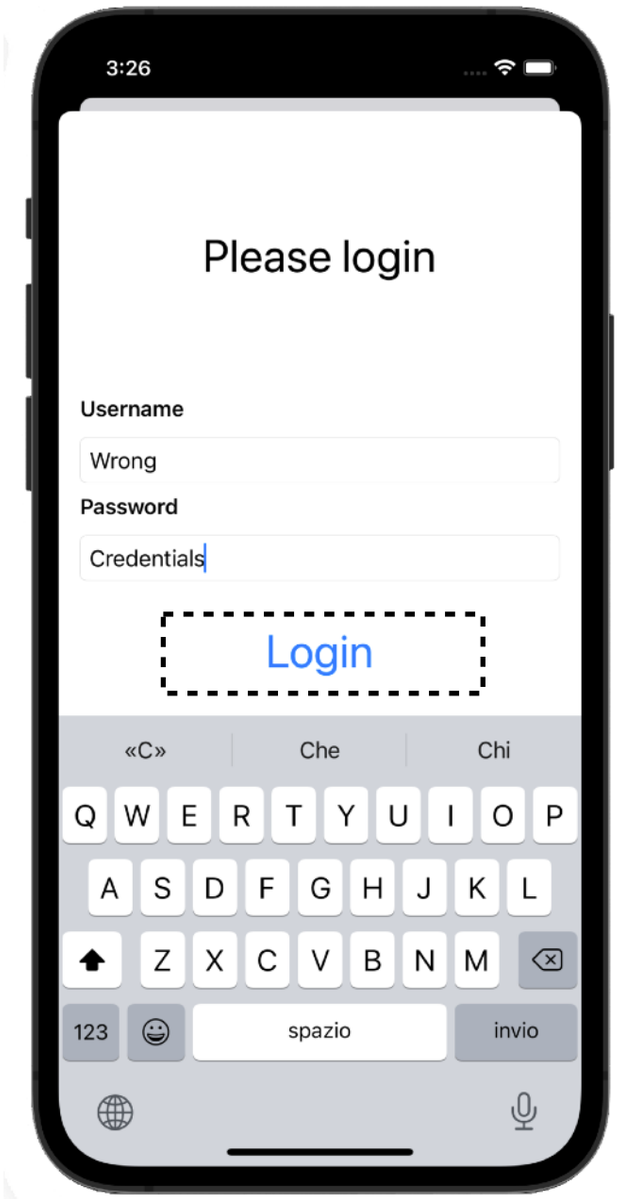
```
func testLoginGivenInvalidCredentials() {
    // Given
    httpServerBuilder
        .route(MockResponses.User.unauthorizedLogin())
        .buildAndStart()

    // When
    appLaunch()

    // Then
    exist(withTextEquals("Please login"))
    typeText(withTextInput("Username"), “Wrong")
    typeText(withTextInput("Password"), "Credentials")



}
```



UI Test using Sky Test Foundation

Domain specific language for testing

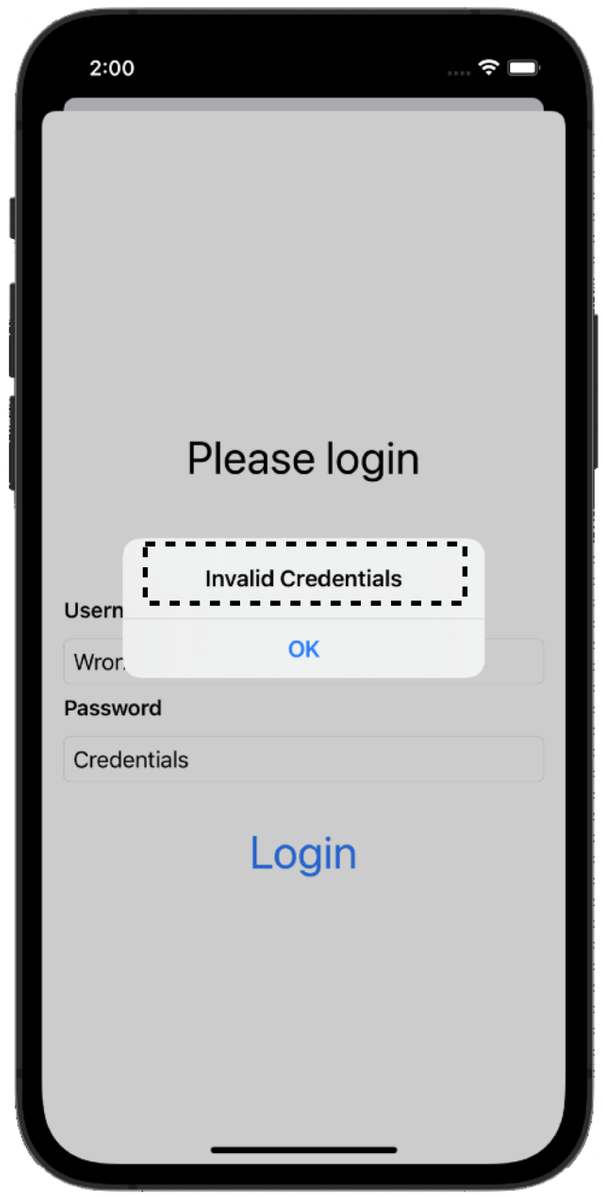
```
func testLoginGivenInvalidCredentials() {
    // Given
    httpServerBuilder
        .route(MockResponses.User.unauthorizedLogin())
        .buildAndStart()

    // When
    appLaunch()

    // Then
    exist(withTextEquals("Please login"))
    typeText(withTextInput("Username"), "Wrong")
    typeText(withTextInput("Password"), "Credentials")
    tap(withButton("Login"))


}
```



UI Test using Sky Test Foundation

Domain specific language for testing

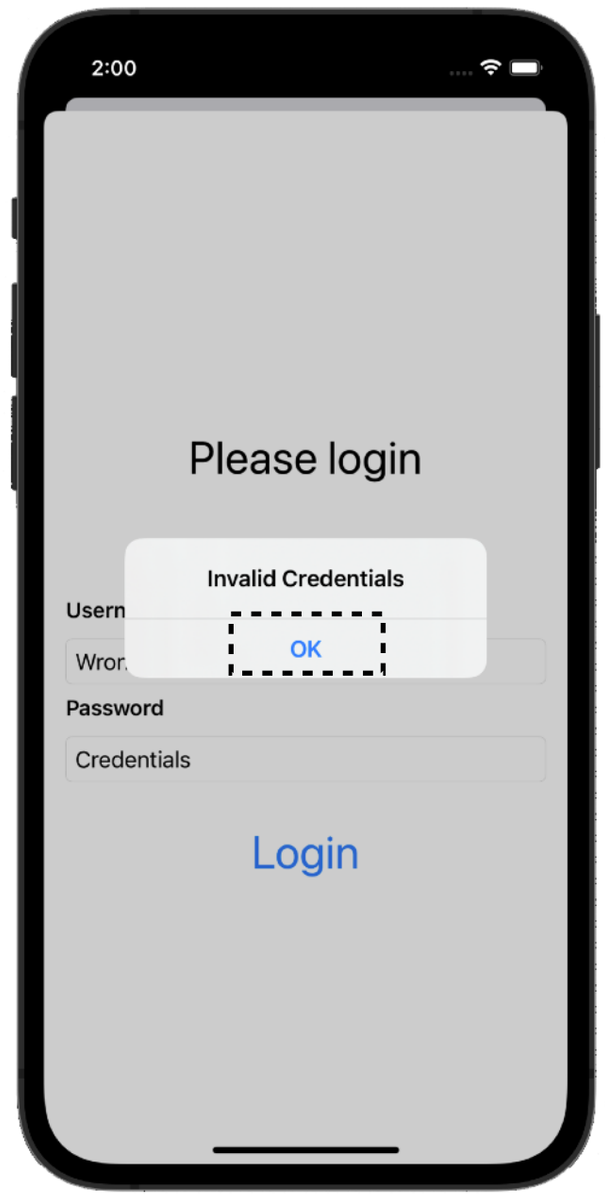
```
func testLoginGivenInvalidCredentials() {
    // Given
    httpServerBuilder
        .route(MockResponses.User.unauthorizedLogin())
        .buildAndStart()

    // When
    appLaunch()

    // Then
    exist(withTextEquals("Please login"))
    typeText(withTextInput("Username"), "Wrong")
    typeText(withTextInput("Password"), "Credentials")
    tap(withButton("Login"))

    exist(withTextEquals("Invalid Credentials"))


}
```



UI Test using Sky Test Foundation

Domain specific language for testing

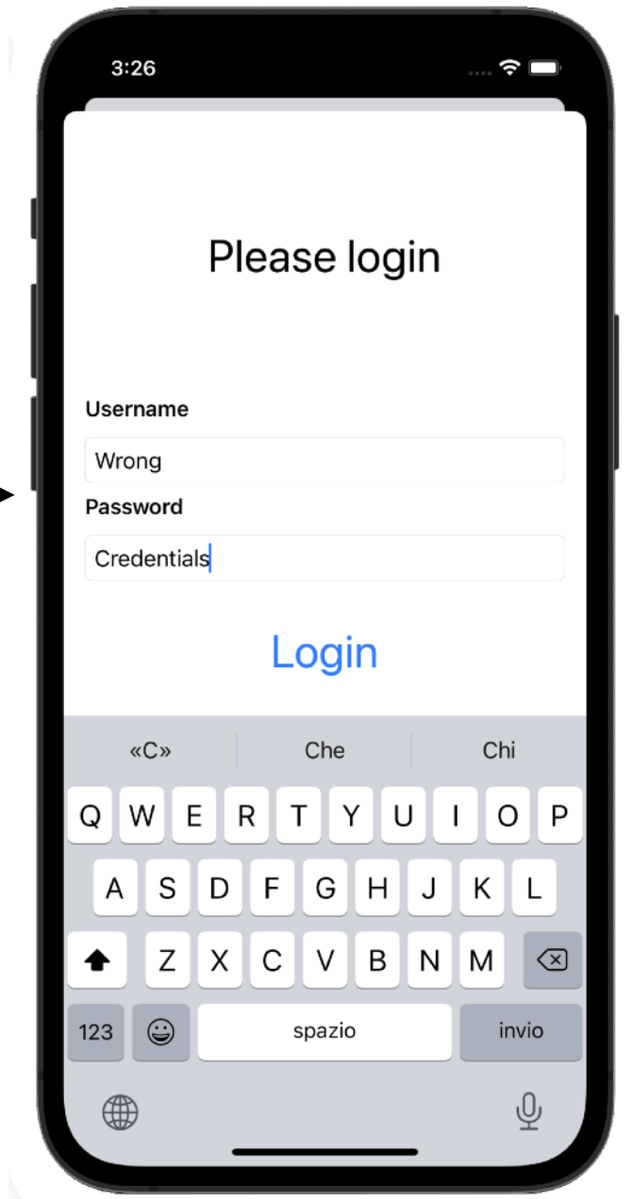
```
func testLoginGivenInvalidCredentials() {
    // Given
    httpServerBuilder
        .route(MockResponses.User.unauthorizedLogin())
        .buildAndStart()

    // When
    appLaunch()

    // Then
    exist(withTextEquals("Please login"))
    typeText(withTextInput("Username"), "Wrong")
    typeText(withTextInput("Password"), "Credentials")
    tap(withButton("Login"))

    exist(withTextEquals("Invalid Credentials"))
    tap(withButton("OK"))

}
```



UI Test using Sky Test Foundation

Domain specific language for testing

```
func testLoginGivenInvalidCredentials() {
    // Given
    httpServerBuilder
        .route(MockResponses.User.unauthorizedLogin())
        .buildAndStart()

    // When
    appLaunch()

    // Then
    exist(withTextEquals("Please login"))
    typeText(withTextInput("Username"), "Wrong")
    typeText(withTextInput("Password"), "Credentials")
    tap(withButton("Login"))

    exist(withTextEquals("Invalid Credentials"))
    tap(withButton("OK"))
    notExist(withTextEquals("Invalid Credentials"))
}
```



UI Test using Sky Test Foundation

Domain specific language for testing

```
func testLoginGivenInvalidCredentials() {
    // Given
    httpServerBuilder
        .route(MockResponses.User.unauthorizedLogin())
        .buildAndStart()

    // When
    appLaunch()

    // Then
    exist(withTextEquals("Please login"))
    typeText(withTextInput("Username"), "Wrong")
    typeText(withTextInput("Password"), "Credentials")
    tap(withButton("Login"))

    exist(withTextEquals("Invalid Credentials"))
    tap(withButton("OK"))
    notExist(withTextEquals("Invalid Credentials"))
}
```

2nd part

Embrace the evolution of a Mobile Application Product

Alex Gotev
Principal Engineer at Sky Italia

A concrete example with a running mobile application

#codegeneration #mocks #test #TDD #DDD #DSL #SkyTestFoundation

2022

Where are we using it?

~2M

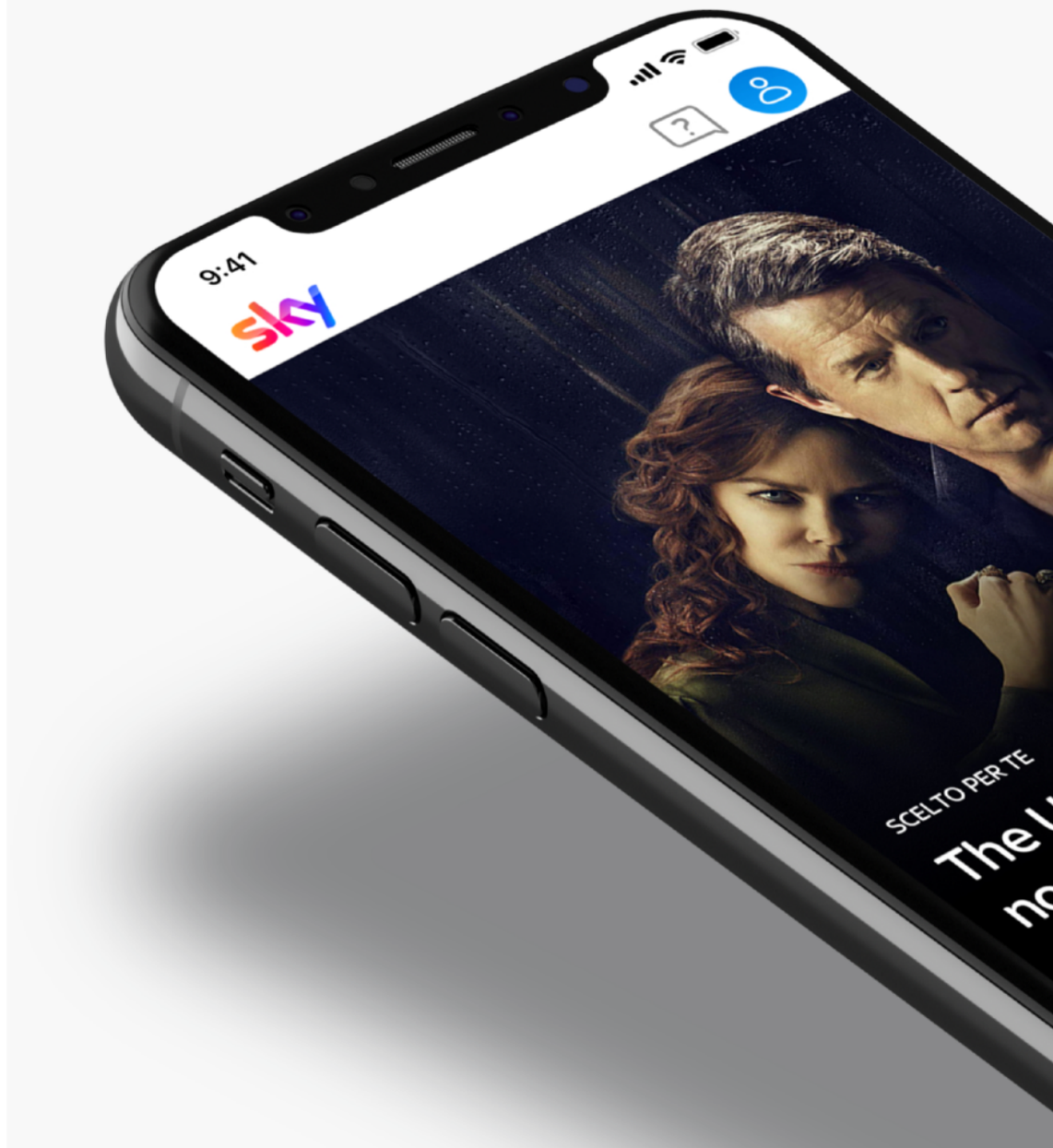
users

99.9%

crash free

85%

code coverage



One more thing...

Unveiling
Sky Test Foundation iOS