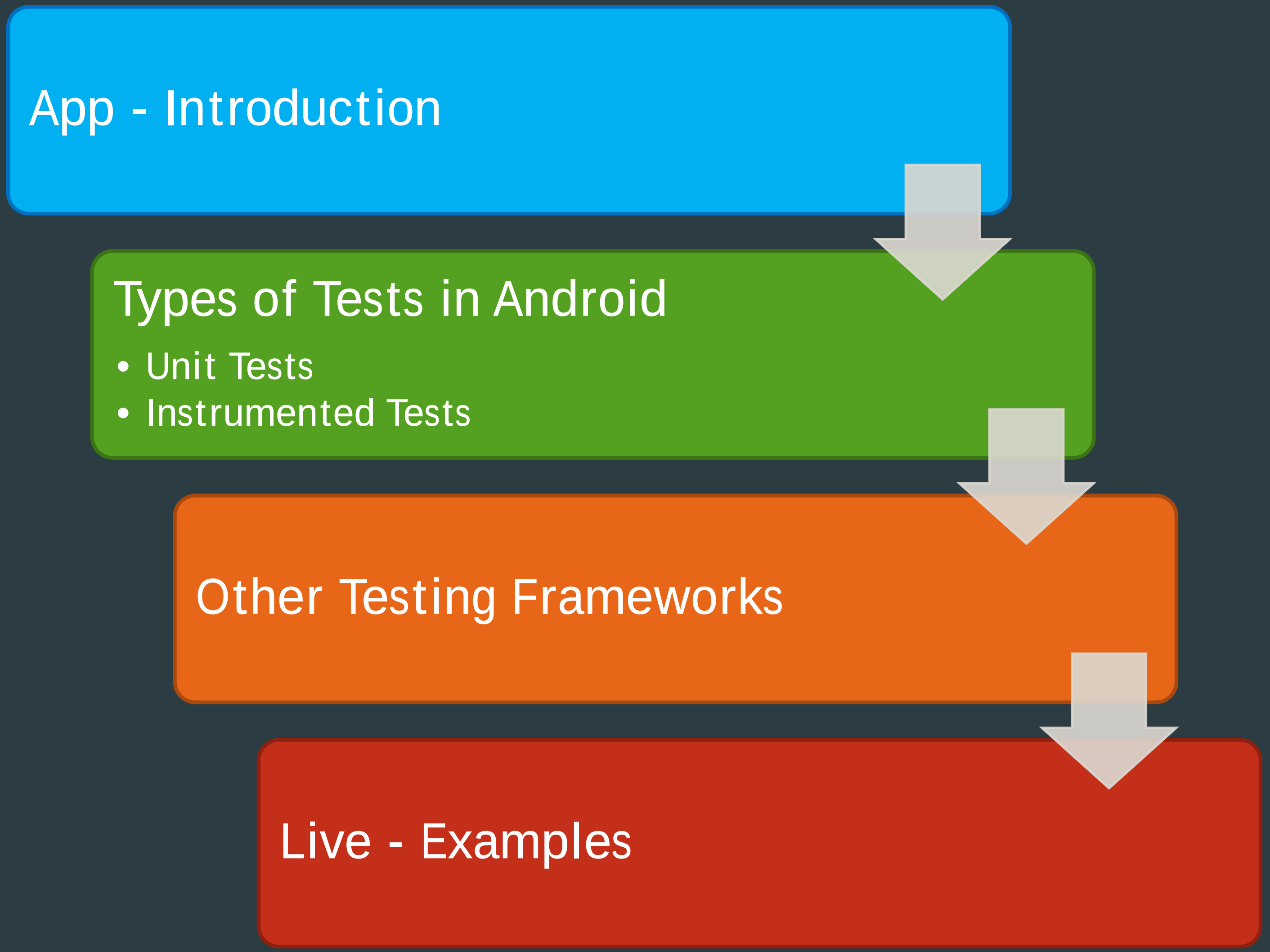
Mobile Testing II

Tobias Redl

25.05.2018

AGENDA



App - Introduction

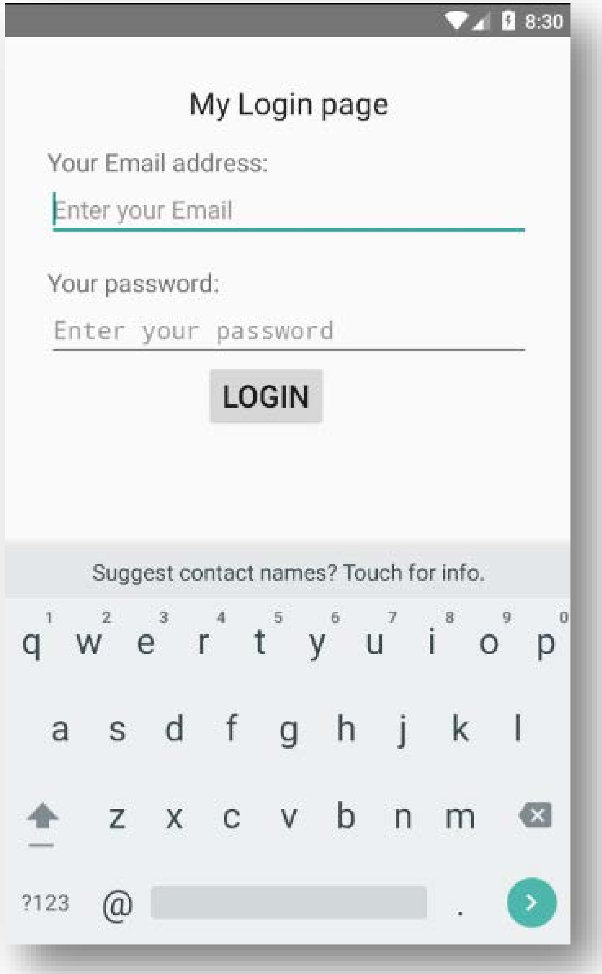
Types of Tests in Android

- Unit Tests
- Instrumented Tests

Other Testing Frameworks

Live - Examples

App - Introduction



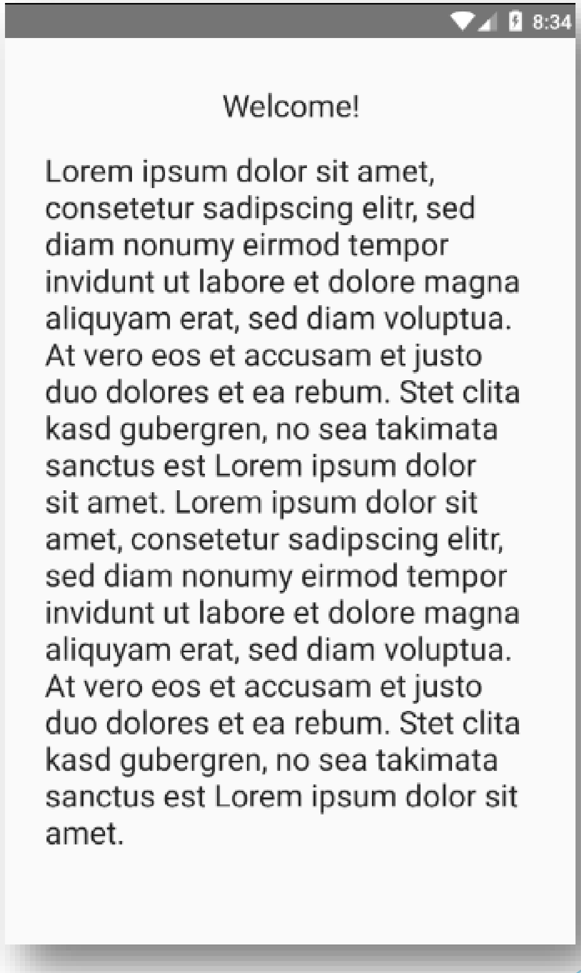
My Login page

Your Email address:

Enter your Email

Your password:

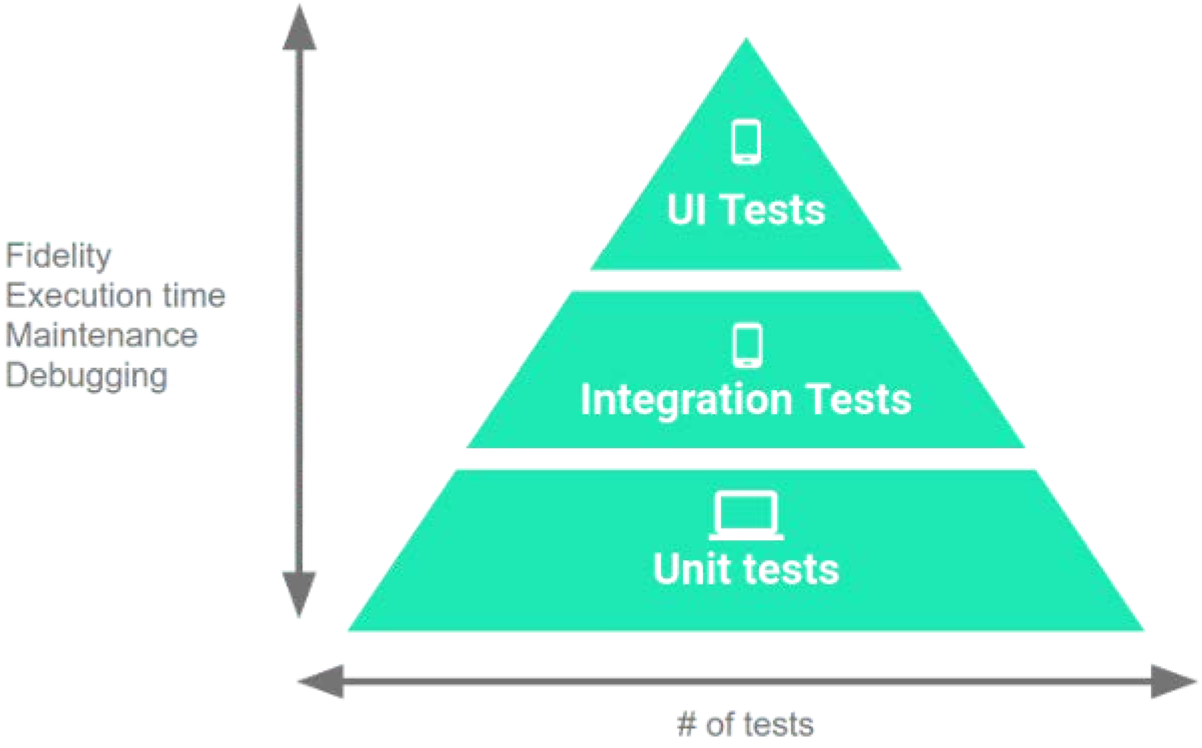
Enter your password



Welcome!

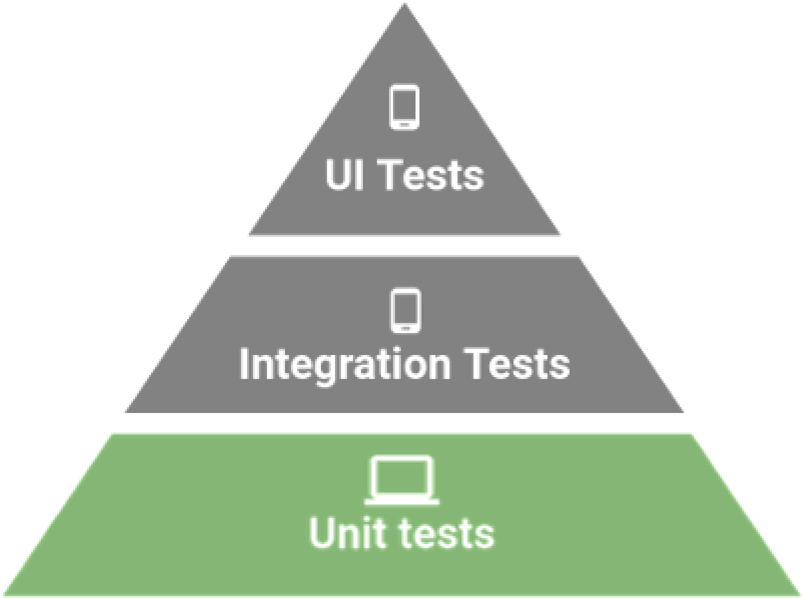
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Types of Tests in Android



Unit Tests

- Tests a single function / method
- Do not need a running app
- Normally no Android dependencies
  - Try to mock them
- It‘s fast
  - Runs on local JVM



Unit Tests

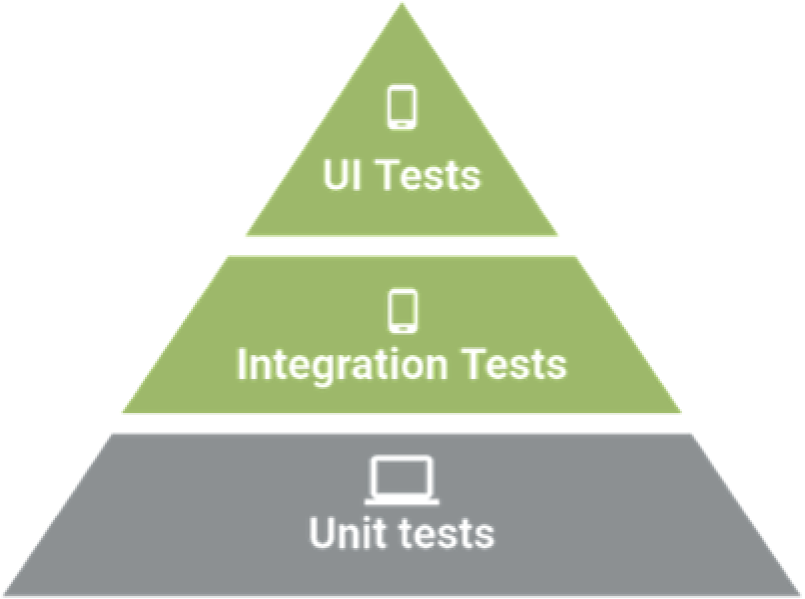
Examples

- E-Mail validation
  - Some Text in front of @
  - @ - symbol
  - Domain after @
- Password validation
  - Long enough
  - All needed symbols are included

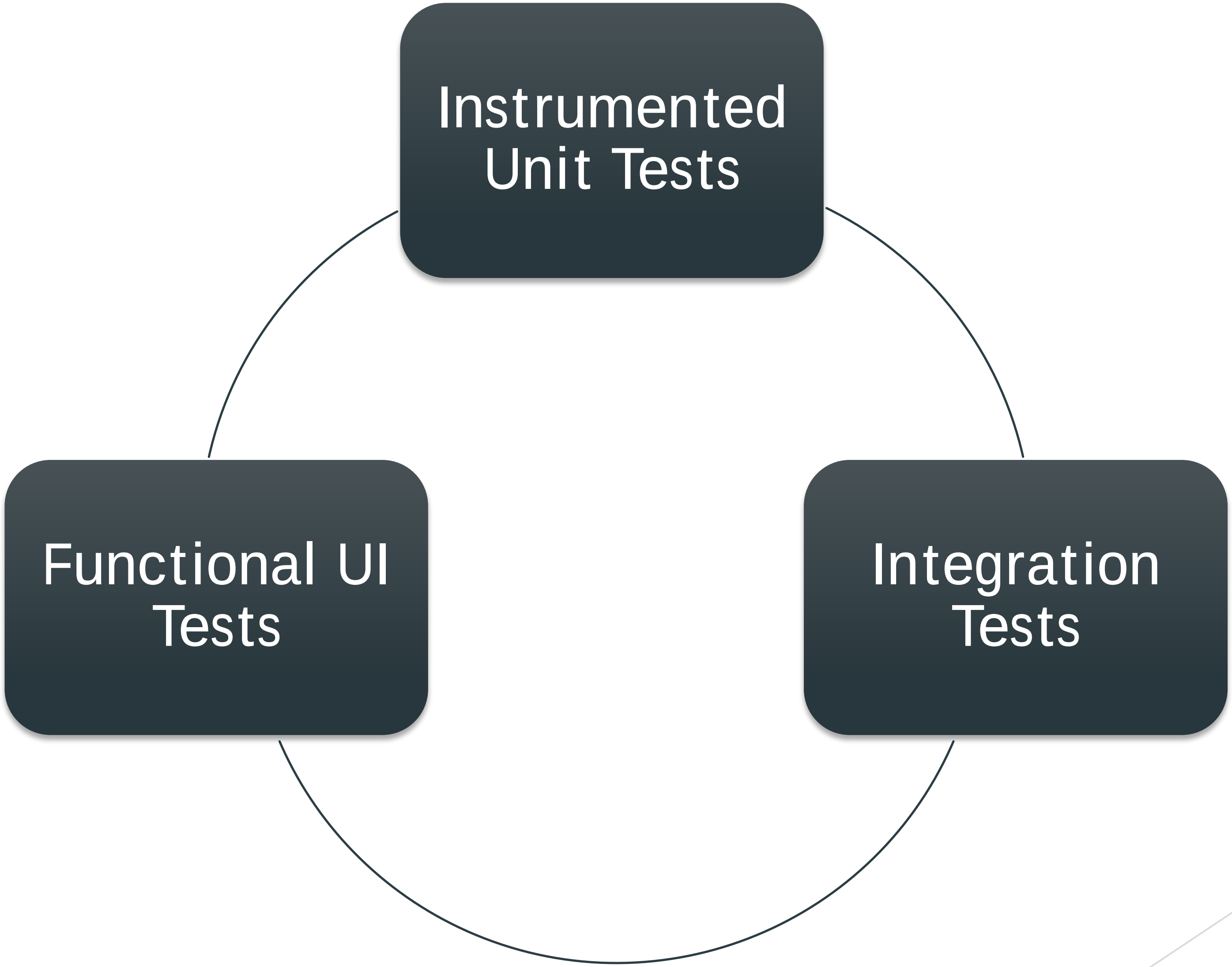
- Login validation (with mock)
- Content delivery (with mock)
- Store data locally (with mock)

Instrumented Tests

- Run on real / virtual device
- Slower than unit Tests
- Have more access
  - App – Context
  - Android dependencies like ressources



Instrumented Tests



Instrumented Unit Tests

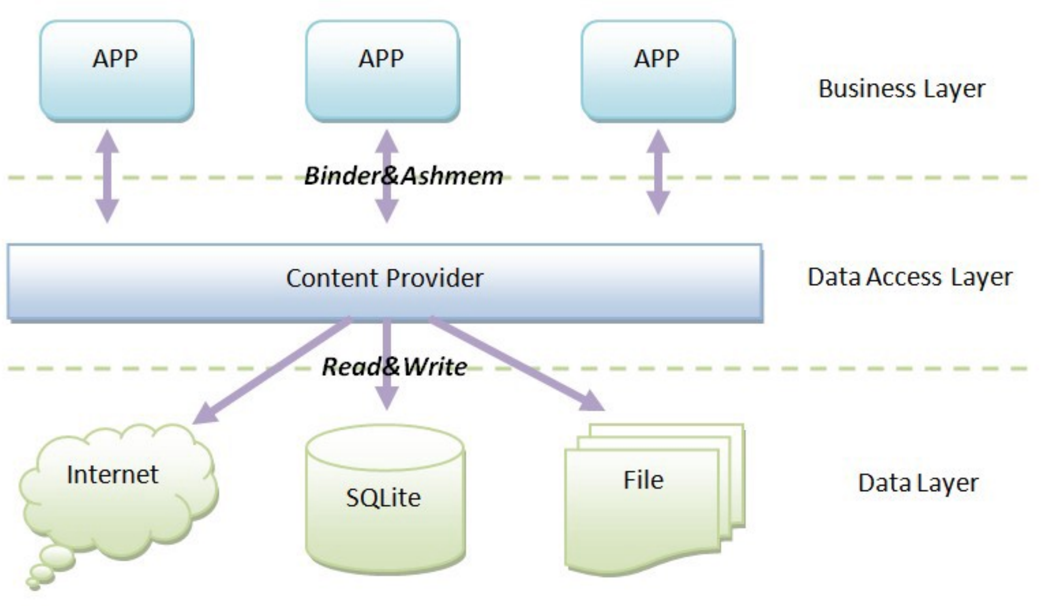
Integration Tests

Functional UI Tests

Instrumented Tests

IntegrationTests

Services



Content Provider

Instrumented Tests

Integration Tests - Examples

- Login validation (no mock)
  - Connect to server
  - Transfer data to and from server
- Content delivery (no mock)
  - Connect to server
  - Transfer data from server
- Sharing data
  - Send data to other apps
  - Receive data from other apps
- Store data locally (no mock)
  - Store in Database

Instrumented Tests

Functional UI Tests

Single App

- Target app behaviour
- Always the same context
- User like interactions

UI TESTING FOR ANDROID

espresso

Multiple Apps

- Interaction between apps
- Changing context
- User and system like interactions

UiAutomator

Instrumented Tests

Functional UI Tests - Examples

- Login Form
  - E-Mail input is available
  - Password input is available
  - Login Button is available
- Failed Login is visible
- Successfull Login in visible

- Content Page
  - Content is displayed
  - Textboxes are visible
- Keep logged in
  - Send app to background an bring it back to foreground

Other Testing Frameworks

iOS - Frameworks

- XCTest
  - Unit Tests
  - Integration Tests
  - Functional UI Tests (XCUITests)
- KIF
- Earl Grey (Espresso for iOS)

Other Testing Frameworks

Appium

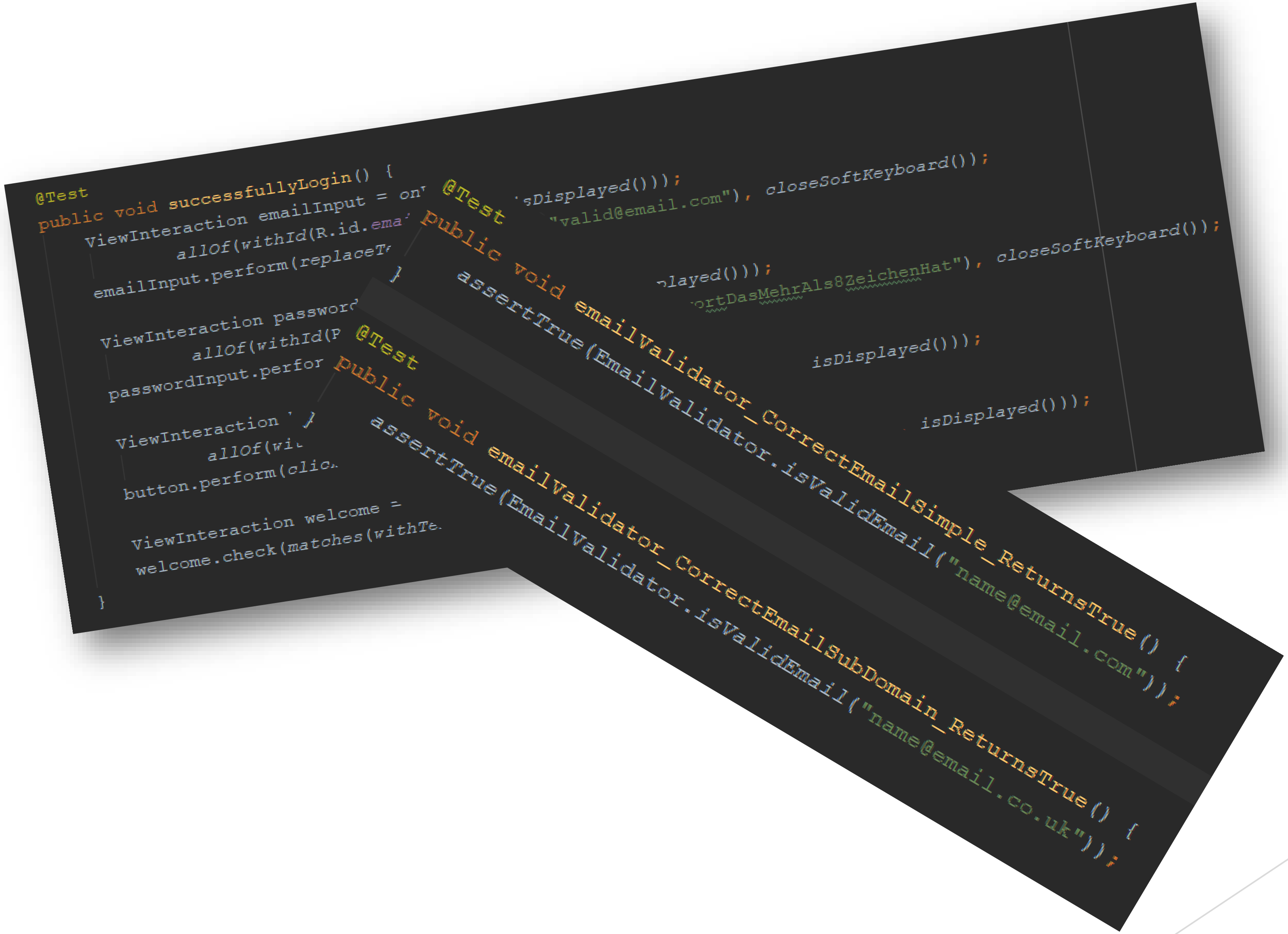
- Based on Selenium (browser testing tool)
- Webserver with REST API
- Client / Server architecture
- Possibility to write tests in different languages (http client API)
- Server and tests can be on different machines
- Tests use XCUITest (iOS)
- Tests use UIAutomator (Android)

Other Testing Frameworks

Roboelectirc & Mockito

- Mocking frameworks
- Mock up
  - Data
  - Classes
  - System parts

Live - Examples



```
@Test
public void successfullyLogin() {
    ViewInteraction emailInput = on        isDisplayed()));
            allOf(withId(R.id.ema          "valid@email.com"), closeSoftKeyboard());
    emailInput.perform(replaceT

    ViewInteraction passwor                played()));
            allOf(withId(P                 ortDasMehrAls8ZeichenHat"), closeSoftKeyboard());
    passwordInput.perfor

    ViewInteraction                        isDisplayed()));
            allOf(wi
    button.perform(clic                    isDisplayed()));

    ViewInteraction welcome =
    welcome.check(matches(withTe

}
```

```
@Test
public void emailValidator_CorrectEmailSimple_ReturnsTrue() {
    assertTrue(EmailValidator.isValidEmail("name@email.com"));
}
```

```
@Test
public void emailValidator_CorrectEmailSubDomain_ReturnsTrue() {
    assertTrue(EmailValidator.isValidEmail("name@email.co.uk"));
}
```