

# UI Автоматизация тестирования мобильных приложений

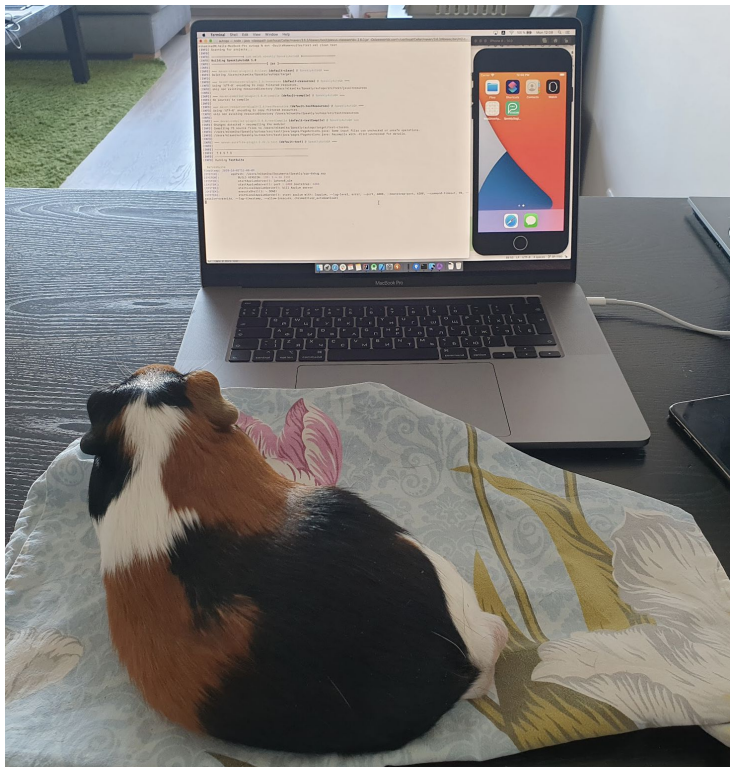
*Wolt*

Мирошниченко Михаил

[hikari.no.mikem@gmail.com](mailto:hikari.no.mikem@gmail.com)

<https://www.linkedin.com/in/miroshnichenkomichael/>

# Обо мне: Мирошниченко Михаил



15 лет в QA

8 лет в Mobile

4 года в Test Automation

3 года в Эстонии

QA Automation в **Wolt** (FI)

Тестировал в:

- Яндекс (RU)
- Доктор Веб (RU)
- Bolt (EE)

Как сделать автоматизацию для мобильных приложений: от выбора инструментов до выполнения тестов в CI

На примере того, как автотесты работают в Wolt Merchant App

# Первая серия: Сборка проекта автотестов на GitHub

После добавления тестов в проект с автотестами происходит сборка Docker образа на GitHub

[Code](#)[Issues](#)[Pull requests](#)[Actions](#)[Projects](#)[Wiki](#)

...

## MA-1159: Modify Merchant App settings with API commands (#58)

main pipeline #129

[Cancel workflow](#)

...

### Summary

#### Jobs

- Build and publish d...

Triggered via push now

Status:

mikamikaWolt pushed &lt;- 88ce2d4 main

**Queued**

Total duration

Artifacts

-

-

main.yml

# Вторая серия: Прогон тестов на новом билде

При Push нового кода приложения происходит сборка его дистрибутива и прогон тестов.

Тесты выполняются на ферме с мобильными устройствами

The image shows a web browser window with a dark-themed sidebar on the left and a main content area on the right. The sidebar contains a logo for 'credformot' and a list of navigation items: Dashboard, Projects, Insights, Organization Settings, and Plan. At the bottom of the sidebar, there are sections for 'Notifications' (with a blue indicator), 'Status' (with a green progress bar), and 'Help'. The main content area is titled 'Running tests' and displays a list of test results. Each result line includes a status (e.g., [PASS]), a test name, and a category (e.g., AppliedOrder). Below the list, there is a summary section with the following text:

```
27 [PASS] ... AppliedOrderCI
28 [PASS] ... AppliedOrderCI
29 [PASS] ... AppliedOrderCI
30 [PASS] ... AppliedOrderCI
31 [PASS] ... AppliedOrderCI
32 [PASS] ... AppliedOrderCI
33 [PASS] ... AppliedOrderCI
34 [PASS] ... AppliedOrderCI
35 [PASS] ... AppliedOrderCI
36 [PASS] ... AppliedOrderCI
37 [PASS] ... AppliedOrderCI
38 [PASS] ... AppliedOrderCI
39 [PASS] ... AppliedOrderCI
40 [PASS] ... AppliedOrderCI
41 [PASS] ... AppliedOrderCI
42 [PASS] ... AppliedOrderCI
43 [PASS] ... AppliedOrderCI
44 [PASS] ... AppliedOrderCI
45 [PASS] ... AppliedOrderCI
46 [PASS] ... AppliedOrderCI
47 [PASS] ... AppliedOrderCI
48 [PASS] ... AppliedOrderCI
49 [PASS] ... AppliedOrderCI
50 [PASS] ... AppliedOrderCI
51 [PASS] ... AppliedOrderCI
52 [PASS] ... AppliedOrderCI
53 [PASS] ... AppliedOrderCI
54 [PASS] ... AppliedOrderCI
55 [PASS] ... AppliedOrderCI
56 [PASS] ... AppliedOrderCI
57 [PASS] ... AppliedOrderCI
58 [PASS] ... AppliedOrderCI
59 [PASS] ... AppliedOrderCI
60 [PASS] ... AppliedOrderCI
61 [PASS] ... AppliedOrderCI
62 [PASS] ... AppliedOrderCI
63 [PASS] ... AppliedOrderCI
64 [PASS] ... AppliedOrderCI
65 [PASS] ... AppliedOrderCI
66 [PASS] ... AppliedOrderCI
67 [PASS] ... AppliedOrderCI
68 [PASS] ... AppliedOrderCI
69 [PASS] ... AppliedOrderCI
70 [PASS] ... AppliedOrderCI
71 [PASS] ... AppliedOrderCI
72 [PASS] ... AppliedOrderCI
73 [PASS] ... AppliedOrderCI
74 [PASS] ... AppliedOrderCI
75 [PASS] ... AppliedOrderCI
76 [PASS] ... AppliedOrderCI
77 [PASS] ... AppliedOrderCI
78 [PASS] ... AppliedOrderCI
79 [PASS] ... AppliedOrderCI
80 [PASS] ... AppliedOrderCI
81 [PASS] ... AppliedOrderCI
82 [PASS] ... AppliedOrderCI
83 [PASS] ... AppliedOrderCI
84 [PASS] ... AppliedOrderCI
85 [PASS] ... AppliedOrderCI
86 [PASS] ... AppliedOrderCI
87 [PASS] ... AppliedOrderCI
88 [PASS] ... AppliedOrderCI
89 [PASS] ... AppliedOrderCI
90 [PASS] ... AppliedOrderCI
91 [PASS] ... AppliedOrderCI
92 [PASS] ... AppliedOrderCI
93 [PASS] ... AppliedOrderCI
94 [PASS] ... AppliedOrderCI
95 [PASS] ... AppliedOrderCI
96 [PASS] ... AppliedOrderCI
97 [PASS] ... AppliedOrderCI
98 [PASS] ... AppliedOrderCI
99 [PASS] ... AppliedOrderCI
100 [PASS] ... AppliedOrderCI
```

27 [PASS] ... AppliedOrderCI  
28 [PASS] ... AppliedOrderCI  
29 [PASS] ... AppliedOrderCI  
30 [PASS] ... AppliedOrderCI  
31 [PASS] ... AppliedOrderCI  
32 [PASS] ... AppliedOrderCI  
33 [PASS] ... AppliedOrderCI  
34 [PASS] ... AppliedOrderCI  
35 [PASS] ... AppliedOrderCI  
36 [PASS] ... AppliedOrderCI  
37 [PASS] ... AppliedOrderCI  
38 [PASS] ... AppliedOrderCI  
39 [PASS] ... AppliedOrderCI  
40 [PASS] ... AppliedOrderCI  
41 [PASS] ... AppliedOrderCI  
42 [PASS] ... AppliedOrderCI  
43 [PASS] ... AppliedOrderCI  
44 [PASS] ... AppliedOrderCI  
45 [PASS] ... AppliedOrderCI  
46 [PASS] ... AppliedOrderCI  
47 [PASS] ... AppliedOrderCI  
48 [PASS] ... AppliedOrderCI  
49 [PASS] ... AppliedOrderCI  
50 [PASS] ... AppliedOrderCI  
51 [PASS] ... AppliedOrderCI  
52 [PASS] ... AppliedOrderCI  
53 [PASS] ... AppliedOrderCI  
54 [PASS] ... AppliedOrderCI  
55 [PASS] ... AppliedOrderCI  
56 [PASS] ... AppliedOrderCI  
57 [PASS] ... AppliedOrderCI  
58 [PASS] ... AppliedOrderCI  
59 [PASS] ... AppliedOrderCI  
60 [PASS] ... AppliedOrderCI  
61 [PASS] ... AppliedOrderCI  
62 [PASS] ... AppliedOrderCI  
63 [PASS] ... AppliedOrderCI  
64 [PASS] ... AppliedOrderCI  
65 [PASS] ... AppliedOrderCI  
66 [PASS] ... AppliedOrderCI  
67 [PASS] ... AppliedOrderCI  
68 [PASS] ... AppliedOrderCI  
69 [PASS] ... AppliedOrderCI  
70 [PASS] ... AppliedOrderCI  
71 [PASS] ... AppliedOrderCI  
72 [PASS] ... AppliedOrderCI  
73 [PASS] ... AppliedOrderCI  
74 [PASS] ... AppliedOrderCI  
75 [PASS] ... AppliedOrderCI  
76 [PASS] ... AppliedOrderCI  
77 [PASS] ... AppliedOrderCI  
78 [PASS] ... AppliedOrderCI  
79 [PASS] ... AppliedOrderCI  
80 [PASS] ... AppliedOrderCI  
81 [PASS] ... AppliedOrderCI  
82 [PASS] ... AppliedOrderCI  
83 [PASS] ... AppliedOrderCI  
84 [PASS] ... AppliedOrderCI  
85 [PASS] ... AppliedOrderCI  
86 [PASS] ... AppliedOrderCI  
87 [PASS] ... AppliedOrderCI  
88 [PASS] ... AppliedOrderCI  
89 [PASS] ... AppliedOrderCI  
90 [PASS] ... AppliedOrderCI  
91 [PASS] ... AppliedOrderCI  
92 [PASS] ... AppliedOrderCI  
93 [PASS] ... AppliedOrderCI  
94 [PASS] ... AppliedOrderCI  
95 [PASS] ... AppliedOrderCI  
96 [PASS] ... AppliedOrderCI  
97 [PASS] ... AppliedOrderCI  
98 [PASS] ... AppliedOrderCI  
99 [PASS] ... AppliedOrderCI  
100 [PASS] ... AppliedOrderCI

Test results: PASS  
[PASS] ... AppliedOrderCI

---

Fixed tests and execution time:  
Total Fixed tests: 1  
Average test execution time: 75 sec  
Total test execution time: 75 sec

Site execution time: 1 min 53 sec

# Сегодня сделаем такую же красоту:

Мобильная ОС: **iOS** и **Android**

Язык разработки: **Kotlin**

Управление проектом: **Maven**

Управление тестированием: **TestNG**

Драйвер для мобильных устройств: **Appium**

Паттерн разработки UI тестов: **PageFactory**

Хранилище кода и сборка проекта: **GitHub**: Actions + Secrets

Среда CI: **CircleCI** + **Docker**

Облачная ферма с устройствами: **BrowserStack**



# План сегодняшней встречи

1. Теория автоматизации
2. Создания проекта автотестов в IDE
3. Создание автотестов
4. Исполнение автотестов
5. Интеграция с CI

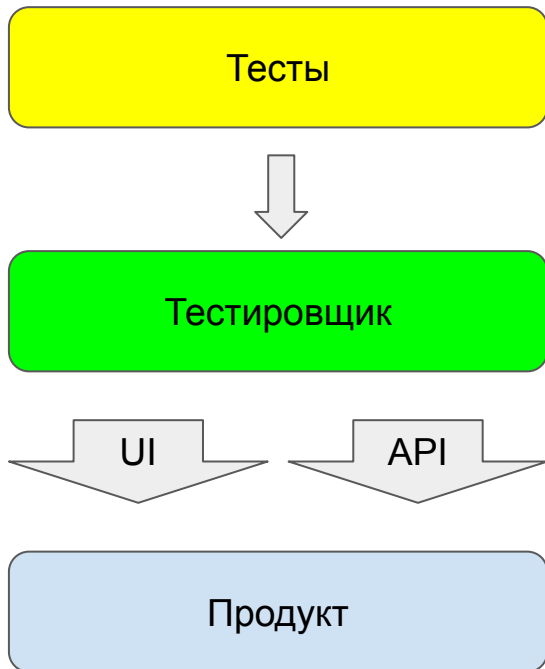
# Для чего автоматизировать тесты?

- Регрессионное тестирование
- Постоянный контроль качества во времени
- Проверить то, что долго или не возможно проверить вручную
- Быстрая обратная связь по качеству каждой сборки

# Что есть?

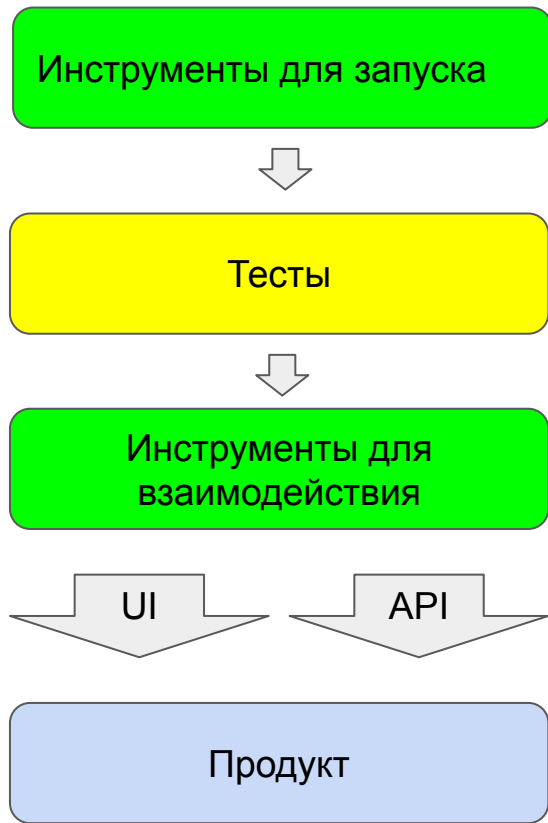
- Код проекта в GitHub
- Есть тест-кейсы

# Схема ручного тестирования



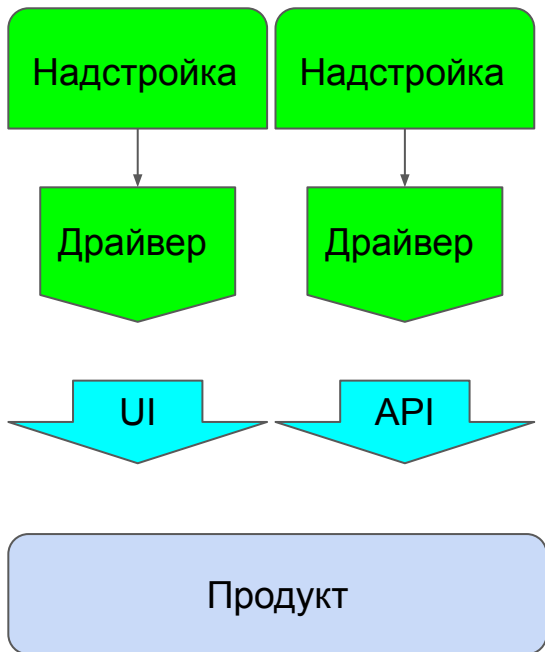
- **API** (Application Programming Interface): основной интерфейс для взаимодействия с другими программами
- **UI**: Графический интерфейс приложения для взаимодействиями с пользователями с UI автотестами

# Схема автоматизированного тестирования



Инструменты для запуска и взаимодействия с приложением называются “стеком автотестирования”

# Состав инструментов взаимодействия

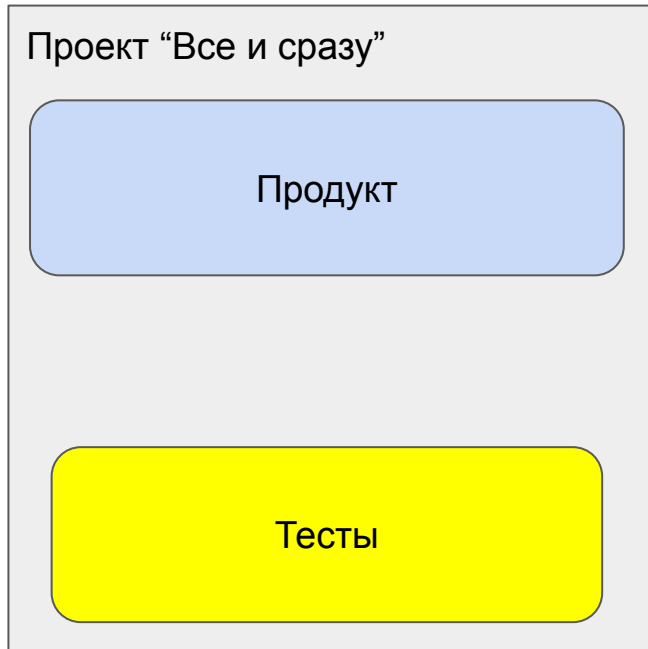


**Драйвер:** предоставляет программный интерфейс для одного из интерфейсов приложения

**Настройка:** делает использование драйвера проще и удобнее

# Автотесты: Где хранить тесты?

Внутри проекта



Как отдельный проект



# Инструменты: Два подхода к автоматизации

## **Нативная** автоматизация (Native):

- Напрямую используем встроенные драйверы автоматизации
- iOS: XCUITest
- Android: UIAutomator, Espresso
- Код приложения и код тестов - один проект

## **Не нативная** автоматизация (Not native):

- Используем встроенные драйверы через посредника (надстройку)
- Mobile: Appium; Web: Selenium
- Тесты - отдельный проект, не зависящий от проекта приложения



# Нативная автоматизация: “Белый ящик”

## Плюсы:

- Можно использовать компоненты и ресурсы приложения
- Максимальные возможности доступа к элементам интерфейсы и их свойствам
- Легче модифицировать поведение приложение

## Минусы:

- Работает только с приложением, для которого разработаны
- Обычно требуется использовать тот же язык, на котором написано приложение

# Не нативная автоматизация: “Черный ящик”

## Плюсы:

- Один проект автотестов для iOS, Android, Web
- Свобода в выборе языка и инструментов
- Компактность проекта

## Минусы:

- Труднее доступ к определенным свойствам элементов интерфейса
- Немного труднее реализовать: требуется управлять двумя продуктами (приложение и автотесты)

# Компоненты проекта not-native автоматизации

- **Framework:**
  - Управление порядком выполнения тестирования
  - Подготовка окружения
  - Подготовка отчетов о ходе тестирования и его результатах
  - Взаимодействие с другими приложениями
  - Предоставление стандартных функций для работы с интерфейсом приложения
  
- **Тесты**
  - iOS приложение
  - Android приложение
  - ...

## Часть 2: Проект автотестов

Перейдем от теории к практике и начнем писать код...

# С чего начать: среда разработки

IntelliJ Idea: <https://www.jetbrains.com/idea/>

- Бесплатная версия для некоммерческой разработки
- Поддерживает разные языки: Kotlin, Java
- Много полезных функций для быстрой разработки



Но вы можете использовать и свою любимую среду разработки! =)

Доклад:

<https://www.youtube.com/watch?v=zp0qC6JT0rE>

# Maven: система управления проектом

Используется для:

- загрузки всех необходимых компонентов
- установки параметров тестирования
- запуска тестов на исполнение
- сбора базовой информации о результатах тестирования

Представляет собой: Файл pom.xml

Сайт: <https://maven.apache.org>

Установка (macOS): **brew install maven**



# Maven: пример файла pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.wolt.merchant.autoqa</groupId>
  <artifactId>WoltAutoQA</artifactId>
  <version>1.0</version>

  <properties>
    <aspectj.version>1.9.2</aspectj.version>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <suiteName>suites/emptySuite.xml</suiteName>
    <env>staging</env> <!-- prod / staging -->
    <timestamp>${maven.build.timestamp}</timestamp>
    <results.directory>target/allure-results</results.directory>
    <kotlin.version>1.5.10</kotlin.version>
    <kotlin.compiler.incremental>>true</kotlin.compiler.incremental>
  </properties>
</project>
```

# Maven: добавление библиотек в проект

```
<!-- https://mvnrepository.com/artifact/io.appium/java-client -->  
<dependency>  
  <groupId>io.appium</groupId>  
  <artifactId>java-client</artifactId>  
  <version>7.3.0</version>  
</dependency>  
  
<!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->  
<dependency>  
  <groupId>org.projectlombok</groupId>  
  <artifactId>lombok</artifactId>  
  <version>1.18.10</version>  
  <scope>provided</scope>  
</dependency>  
  
<!-- https://mvnrepository.com/artifact/org.testng/testng -->  
<dependency>  
  <groupId>org.testng</groupId>  
  <artifactId>testng</artifactId>  
  <version>6.14.3</version>  
  <scope>test</scope>  
</dependency>
```



# Maven: подключение новых библиотек

Сайт: <https://mvnrepository.com> - используется для поиска библиотек

Пример кода для подключения новой библиотеки:

```
<!-- https://mvnrepository.com/artifact/io.appium/java-client -->
```

```
<dependency>
```

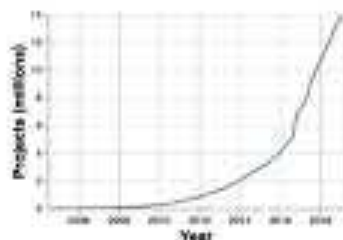
```
  <groupId>io.appium</groupId>
```

```
  <artifactId>java-client</artifactId>
```

```
  <version>7.3.0</version>
```

```
</dependency>
```

## Indexed Artifacts (23.4M)



## Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries

## What's New in Maven

**SLF4J API Module**

50,756 usages

org.slf4j &gt; slf4j-api &gt; 2.0.0-alpha5

MIT

The slf4j API

Last Release on Aug 30, 2021

**Deps Alpha**

64 usages

org.clojure &gt; tools.deps.alpha &gt; 0.12.1036

EPL

Deps Alpha

Last Release on Aug 30, 2021

**OSGi LogService Implemented Over SLF4J**

26 usages

org.slf4j &gt; osgi-over-slf4j &gt; 2.0.0-alpha5

MIT

OSGi LogService implementation over SLF4J

Last Release on Aug 30, 2021

# Maven: Как добавить тесты?

Maven может собрать ваш проект, но как сказать ему, что ваши классы содержат тестовые методы?

Поможет **TestNG!**

# TestNG: инструмент управления тестами

**Позволяет** с помощью suite.xml-файлов:

- настраивать параметры тестирования
- отделить параметры тестирования от параметров сборки проекта автотестов
- расширять возможности Maven

**Задаёт:**

- Действия перед началом и после окончания тестирования, до и после каждого теста
- Какие классы и функции являются запускаемыми тестами

# TestNG: Пример suite файла

```
<!DOCTYPE suite SYSTEM "http://beust.com/testng/testng-1.0.dtd" >
<suite name="Mobile UI Tests">
  <parameter name="product" value="merchantApp"/>
  <parameter name="deviceName" value="ipad_sim"/>
  <parameter name="startType" value="fastReset"/>
  <parameter name="env" value="staging"/>
  <parameter name="runner" value="local"/>
  <parameter name="launchParameters" value="-skipIntros"/>
  <test name="TEST suite">
    <groups>
      <run>
        <include name="ios"/> <!-- android / ios -->
      </run>
    </groups>
    <classes>
      <class name="tests.simpleChecks.SignInTest"/>
    </classes>
  </test>
</suite>
```

# TestNG: Параметры тестирования

```
<!DOCTYPE suite SYSTEM "http://beust.com/testng/testng-1.0.dtd" >
<suite name="Mobile UI Tests">
  <parameter name="product" value="merchantApp"/>
  <parameter name="deviceName" value="ipad_sim"/>
  <parameter name="startType" value="fastReset"/>
  <parameter name="env" value="staging"/>
  <parameter name="runner" value="local"/>
  <parameter name="launchParameters" value="-skipIntros"/>
  <test name="TEST suite">
    <groups>
      <run>
        <include name="ios"/> <!-- android / ios -->
      </run>
    </groups>
    <classes>
      <class name="tests.simpleChecks.SignInTest"/>
    </classes>
  </test>
</suite>
```

# TestNG: Список тестовых классов и групп тестов

```
<!DOCTYPE suite SYSTEM "http://beust.com/testng/testng-1.0.dtd" >
<suite name="Mobile UI Tests">
  <parameter name="product" value="merchantApp"/>
  <parameter name="deviceName" value="ipad_sim"/>
  <parameter name="startType" value="fastReset"/>
  <parameter name="env" value="staging"/>
  <parameter name="runner" value="local"/>
  <parameter name="launchParameters" value="-skipIntros"/>
  <test name="TEST suite">
    <groups>
      <run>
        <include name="ios"/> <!-- android / ios -->
      </run>
    </groups>
    <classes>
      <class name="tests.simpleChecks.SignInTest"/>
    </classes>
  </test>
</suite>
```

# Драйвер автоматизации: Appium

Appium:

- это сервер, к которому мы подключаемся для отправки команд устройству
- это драйверы для каждой отдельной операционной системы
- работает через HTTP

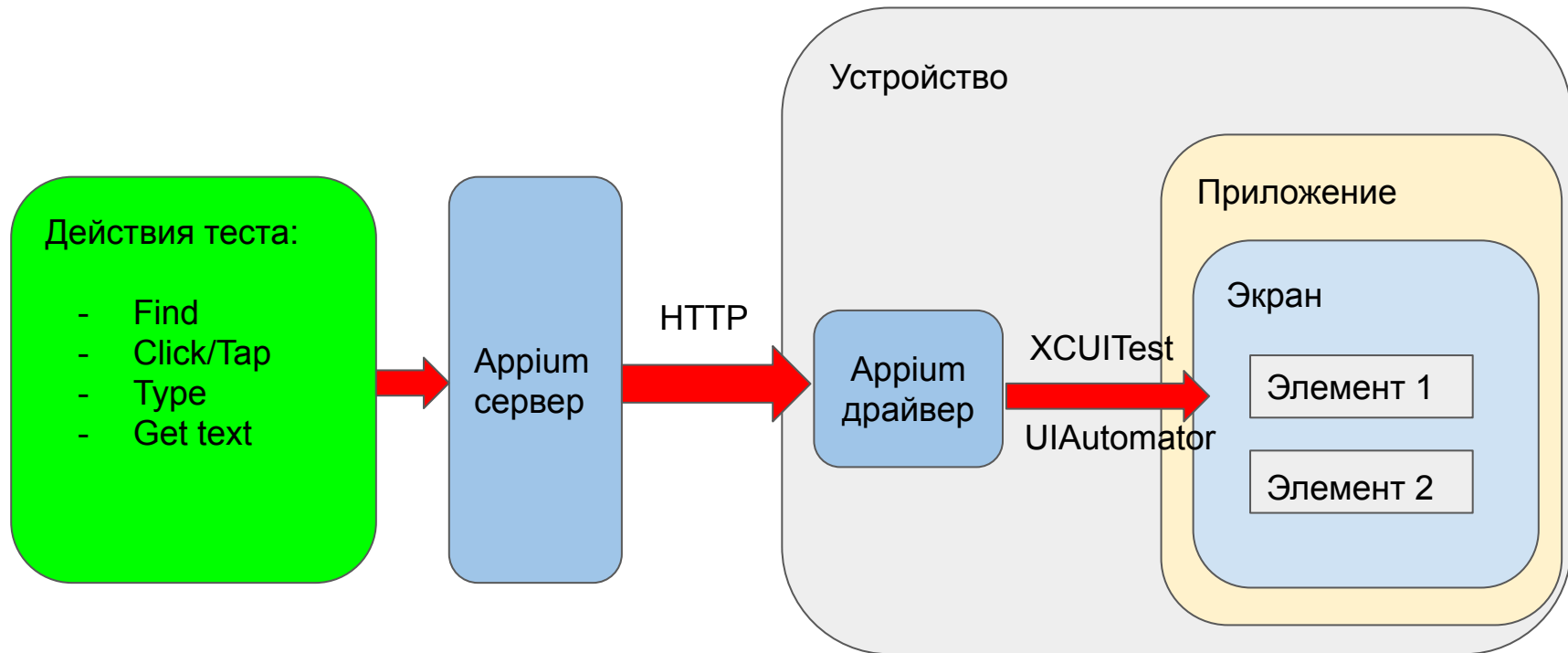
Сайт и установка: <https://appium.io>

Состоит из: сервер и клиент Appium Inspector





# Appium: схема работы сервера



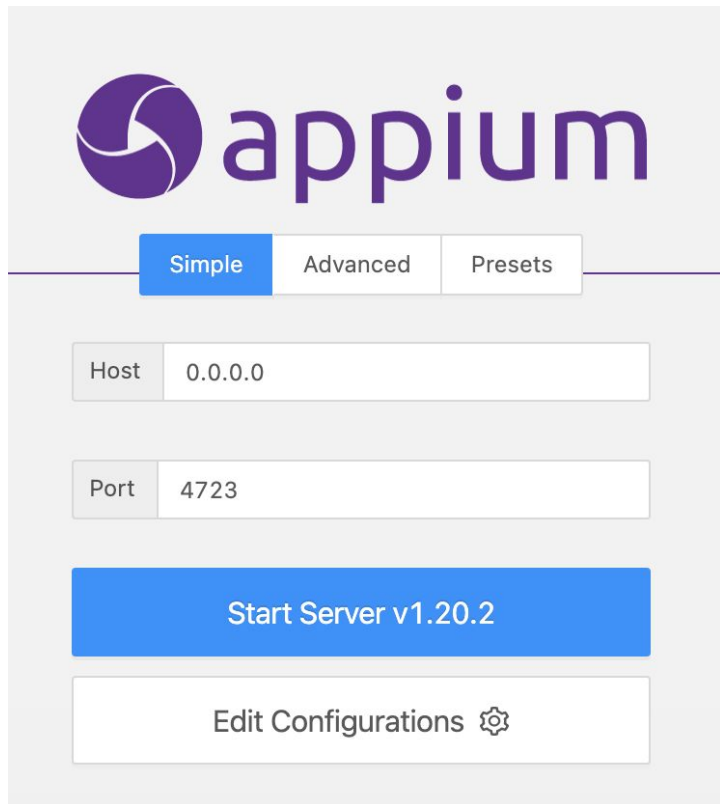
# Appium Inspector

Лучший друг разработчика автотестов: позволяет изучить интерфейс приложения

Типовой порядок работы с Appium (для “ручного” использования и в автотестах):

- Запуск сервера
- Задание параметров для подключения: Capabilities
- Подключение к устройству
- Исследование приложения

# Appium: запуск сервера











Стартовый экран Appium Desktop для запуска сервера.

Возможен запуск и без интерфейса



# Арриум: настройка подключения

| Desired Capabilities | Saved Capability Sets 2   | Attach to Session...        |
|----------------------|---|-----------------------------|
| platformName         | text     | iOS                         |
| deviceName           | text     | iPad (8th generation)       |
| bundleId             | text     | com.wolt.merchant-te        |
| automationName       | text     | XCuiTest                    |
| platformVersion      | text     | 14.5                        |
| noReset              | boolean  | <input type="radio"/> false |
| fullReset            | boolean  | <input type="radio"/> false |
| app                  | text    | /Users/mikhailmiroshn       |

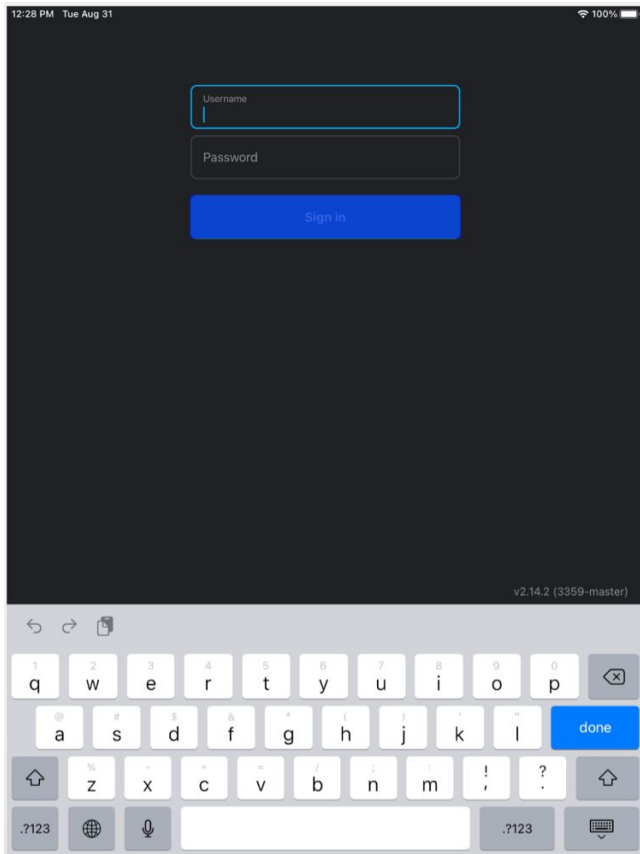
## Надо указать:

- Платформа
- Устройство
- Приложение
- Драйвер

## Дополнительно:

- Настройки очистки данных перед запуском

# Arriim: Изучение схемы интерфейса



Source
Actions

App Source

- <XCUIElementTypeWindow>
  - <XCUIElementTypeOther>
    - <XCUIElementTypeOther>
      - <XCUIElementTypeOther name="loginView.root">
        - <XCUIElementTypeOther name="loginView">
          - <XCUIElementTypeOther name="loginView">
            - <XCUIElementTypeButton name="loginView">
              - <XCUIElementTypeStaticText name="Sign in">
              - <XCUIElementTypeStaticText name="v2.14 (3359-master)">

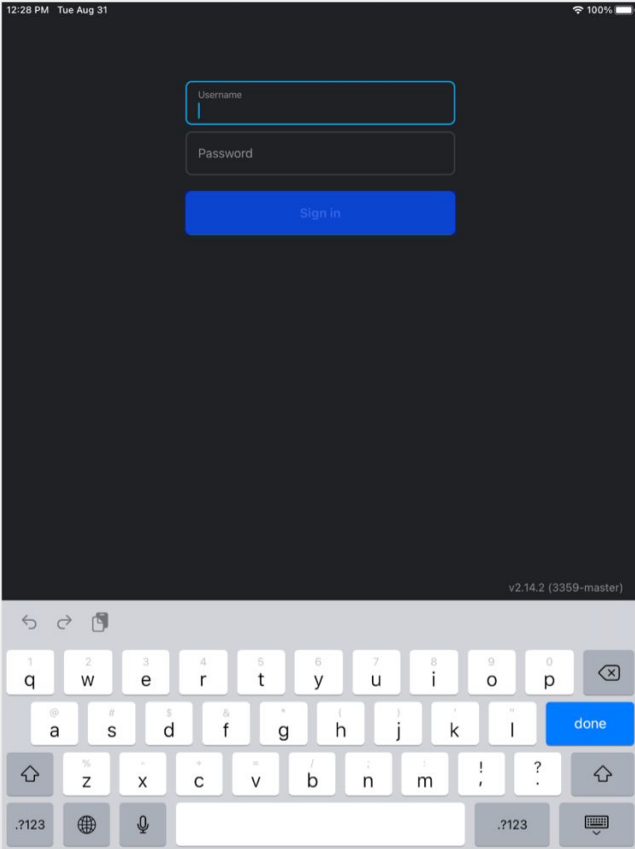
Selected Element

Tap
Send Keys
Clear
🗑️

| Find By                     | Selector  | Time (ms)  |
|-----------------------------|---|------------|
| accessibility id            | loginView.signInButton  | Get Timing |
| -ios class chain(beta)      | **/XCUIElementTypeButton[ label == "loginView.signInButton" ] | Get Timing |
| -ios predicate string(beta) | label == "loginView.signInButton"                             | Get Timing |
| xpath                       | //XCUIElementTypeButton[@name="loginView.signInButton"]       | Get Timing |

| Attribute | Value                                |
|-----------|--------------------------------------|
| elementId | 3C000000-0000-0000-0F7D-000000000000 |
| type      | XCUIElementTypeButton                |
| name      | loginView.signInButton               |

# Appium: Способы нахождения элемента



Source
Actions

App Source

- <XCUIElementTypeWindow>
  - <XCUIElementTypeOther>
    - <XCUIElementTypeOther>
      - <XCUIElementTypeOther name="loginView.root">
        - <XCUIElementTypeOther name="loginView">
          - <XCUIElementTypeOther name="loginView">
            - <XCUIElementTypeButton name="loginView">
              - <XCUIElementTypeStaticText name="Sign in">
              - <XCUIElementTypeStaticText name="v2.14 (3359-master)">

Selected Element

Tap
Send Keys
Clear
🗑️

| Find By                     | Selector  | Time (ms)  |
|-----------------------------|---|------------|
| accessibility id            | loginView.signInButton                                      | Get Timing |
| -ios class chain(beta)      | **/XCUIElementTypeButton[label == "loginView.signInButton"] | Get Timing |
| -ios predicate string(beta) | label == "loginView.signInButton"                           | Get Timing |
| xpath                       | //XCUIElementTypeButton[@name="loginView.signInButton"]     | Get Timing |

| Attribute | Value                                |
|-----------|--------------------------------------|
| elementId | 3C000000-0000-0000-0F7D-000000000000 |
| type      | XCUIElementTypeButton                |
| name      | loginView.signInButton               |

# Appium: какое приложение можно тестировать?

Самое главное требование к приложения для возможности использовать UI автоматизацию тестирования:

**Элементы интерфейса, которые используются в тестах, обязаны иметь уникальные идентификаторы!**

Без таких идентификаторов Appium не сможет найти элемент за минимальное время и абсолютно точно.



# Начало создания автоматизированного теста

- У нас есть проект с необходимыми модулями
- Настроено подключение к устройству
- Мы знаем структуру интерфейса и как находить элементы на экранах

# Часть 3: Создание автотестов

Построение эффективной структуры автоматизированных тестов

# Базовая структура автотеста

Подключиться к устройству



# Базовая структура автотеста

Подключиться к устройству

**Получить схему интерфейса**



# Базовая структура автотеста

Подключиться к устройству

Получить схему интерфейса

**Найти нужный элемент**



# Базовая структура автотеста

Подключиться к устройству

Получить схему интерфейса

Найти нужный элемент

**Выполнить действие**



# Базовая структура автотеста

Подключиться к устройству

Получить схему интерфейса

Найти нужный элемент

Выполнить действие

**Проверить результат**

# Базовая структура автотеста

Подключиться к устройству

Получить схему интерфейса

Найти нужный элемент

Выполнить действие

Проверить результат



# Пример команд для работы с элементом

```
// Find edit field element with ID
val editField: MobileElement = driver.findElement(By.id( id: "my_edit_field"))

// Define text to enter
val expectedValue = "Some text to enter"

// Click edit field and enter text
editField.click()
editField.setValue(expectedValue)

// Get actual text in edit field
val actualValue = editField.text

// Compare with expected result
Assert.assertEquals(
    actualValue, expectedValue,
    "Actual text '$actualValue' in edit field is not '$expectedValue'"
)
```

# Пример автотеста: Форма с логином и паролем

## LoginFormContainer

Login

input

Password

edit.password

Submit

```

fun testExample() {
    val testUserName = "autoqa"
    val testPassword = "iddqd"
    val formContainer = "LoginFormContainer"
    val loginFieldLocator = "//form[@id='$formContainer']//input"
    val loginField: MobileElement = driver.findElement(By.xpath(loginFieldLocator))
    with(loginField) { this: MobileElement
        click()
        clear()
        sendKeys(testUserName)
    }
    Assert.assertEquals(loginField.text, testUserName)
    val passwordFieldLocator = "edit.password"
    val passwordField: MobileElement = driver.findElement(By.id(passwordFieldLocator))
    with(passwordField) { this: MobileElement
        click()
        clear()
        sendKeys(testPassword)
    }
    val ctaButton: MobileElement =
        driver.findElement<MobileElement?>(By.id(formContainer)).findElement(By.linkText(linkText: "Submit"))
    ctaButton.click()
}

```

# Несколько тестов использующих один элемент

Тест 1: Вход в приложение с парой логин-пароль

- Найти элемент Login
- ...
- **Найти элемент Submit**
- **Нажать Submit**

Тест 2: Попытка входа без логина и пароля

- **Найти элемент Submit**
- **Нажать Submit**

The image shows a light blue rectangular container labeled "LoginFormContainer". Inside the container, there are three elements: a "Login" label above a red rounded rectangular button labeled "input"; a "Password" label above another red rounded rectangular button labeled "edit.password"; and a white rectangular button with a grey border and a 3D effect labeled "Submit".

# Решение: разделение тестов и их шагов

Экран\_A:  
- Локаторы  
- Шаги

...

Экран\_N:  
- Локаторы  
- Шаги



Тест А:  
- Экран\_A: Шаг\_2  
- Экран\_A: Шаг\_4

Тест С:  
- Экран\_A: Шаг\_1  
- Экран\_A: Шаг\_2  
- ...  
- Экран\_N: Шаг\_5  
- Экран\_N: Шаг\_7

# Page Factory

- Позволяет разнести тесты с локаторами элементов и шагами
- Буквально: Создаем Page Objects
- Дает возможность строить тесты из шагов

Чтобы узнать больше:

<https://www.guru99.com/page-object-model-pom-page-factory-in-selenium-ultimate-guide.html> EN

<https://victorz.ru/20171007671>

# Page Factory: принцип работы

## Класс описания экрана (Page Object):

- Локаторы: правила как искать элементы на экране
- Шаги - действия с элементами: поиск, нажатие, ввода текста, получение текста, прокрутка

## Использование:

- Инициализация объектов на странице
- Вызов шагов из теста

# Page Factory: пример описания экрана - локаторы

```
@iOSXCUITFindBy(id = "loginView.mainContainer")  
@AndroidFindBy(id = "loginScreen.mainView")  
private val mainContainer: List<MobileElement>? = null
```

```
@iOSXCUITFindBy(id = "loginView.usernameField")  
@AndroidFindBy(id = "loginScreen.userEditField")  
private val userNameInput: List<MobileElement>? = null
```

```
@iOSXCUITFindBy(id = "loginView.passwordField")  
@AndroidFindBy(id = "loginScreen.passwordEditField")  
private val passwordInput: List<MobileElement>? = null
```

```
@iOSXCUITFindBy(id = "loginView.signInButton")  
@AndroidFindBy(id = "loginScreen.ctaButton")  
private val signInButton: List<MobileElement?>? = null
```

# Page Factory: Шаги с переменными-элементами

```
fun setPassword(password: String): LogInScreen = also { it: LogInScreen  
|    passwordInput?.get(0)!!.sendKeys(password)  
}
```

```
fun tapSignInButton(): LogInScreen = also { it: LogInScreen  
|    signInButton?.get(0)!!.click()  
}
```

```
fun getSignInButtonText(): String {  
|    return signInButton?.get(0)!!.text  
}
```



# Page Factory: Способы поиска элемента

FindBy может находить элементы по:

- Tag (Android)
- Accessibility label (iOS)
- Class name
- XPath
- Image (base64 encoded image)
- Android UIAutomator UISelector
- Android Espresso View tag / Data matcher
- iOS UIAutomation classchain, predicate string

<https://appium.io/docs/en/commands/element/find-elements/>

# Page Factory: Переменная для элемента

```
val button: List<MobileElement>? = null
```

- **List** : найти все элементы с одинаковым ID одной командой
- **? = null** : изначально переменная пуста (null)

# Page Factory: Инициализация

- В начале работе класс экрана не указывает на реальные элементы приложения.
- Прежде чем работать с переменными, описывающими элементы экрана, их надо инициализировать.



# Page Factory: Инициализация в провайдере

```
class OnboardingProvider(  
    private val driver: AppiumDriver<*>?,  
    private val softAssert: SoftAssert  
) {  
  
    fun logInScreen(): LogInScreen {  
        return LogInScreen(driver, softAssert)  
    }  
}
```

Провайдер: объект, объединяющий экраны из одной области приложения

# Page Factory: Пример провайдеров

```
class ModalsProvider(  
    private val driver: AppiumDriver<*>?,  
    private val softAssert: SoftAssert  
) {  
  
    fun alertDialog(): AlertDialog {  
        return AlertDialog(driver, softAssert)  
    }  
  
    fun phoneDialog(): PhoneDialog {  
        return PhoneDialog(driver, softAssert)  
    }  
  
    fun intercom(): IntercomModal {  
        return IntercomModal(driver, softAssert)  
    }  
  
    fun moreActionsMenu(): MoreActionsMenu {  
        return MoreActionsMenu(driver, softAssert)  
    }  
}
```

# Page Factory: Корневой класс с провайдерами

```
class MerchantApp(  
    private val driver: AppiumDriver<*>?,  
    private val softAssert: SoftAssert  
) {  
  
    fun onboarding(): OnboardingProvider {  
        return OnboardingProvider(driver, softAssert)  
    }  
  
    fun menu(): MenuProvider {  
        return MenuProvider(driver, softAssert)  
    }  
  
    fun activeOrder(): ActiveOrderProvider {  
        return ActiveOrderProvider(driver, softAssert)  
    }  
}
```

# Page Factory: Корневой класс

- Служит как единая точка входа для всех экранов приложения
- Объединяет все провайдеры приложения
- Каждый продукт в проекте - отдельное “приложение”

# Page Factory: Выбор провайдера в приложении

merchantApp.|

|     |                |                     |
|-----|----------------|---------------------|
| m   | onboarding ()  | OnboardingProvider  |
| ↑ m | settings ()    | SettingsProvider    |
| ↓ m | activeOrder () | ActiveOrderProvider |
| m   | menu ()        | MenuProvider        |
| ↑ m | pastOrder ()   | PastOrdersProvider  |
| ↓ m | whatsNew ()    | WhatsNewProvider    |
| m   | modals ()      | ModalsProvider      |



# Page Factory: Выбор экрана в провайдере

```
merchantApp.modals().|
```

|   |   |                      |                 |
|---|---|----------------------|-----------------|
| ↑ | m | phoneDialog ()       | PhoneDialog     |
| ↓ | m | intercom ()          | IntercomModal   |
| ↓ | m | alertDialog ()       | AlertDialog     |
| ↑ | m | equals (other: Any?) | Boolean         |
| ↓ | m | moreActionsMenu ()   | MoreActionsMenu |

# Page Factory: Выбор шага, доступного на экране

```
merchantApp.modals().phoneDialog().|
```

|     |                       |             |
|-----|-----------------------|-------------|
| v   | isPhoneDialogShown    | PhoneDialog |
| ↑ v | TAG                   | String      |
| m   | getPhoneNumberText () | String      |
| ↓ m | getDisclaimerText ()  | String      |
| ↓ m | getTitleText ()       | String      |
| ↓ m | getUserNameText ()    | String      |
| m   | getWarningText ()     | String      |
| m   | tapCloseButton ()     | OrderCell   |

# Page Factory: Выбор шага для другого экрана

```
merchantApp.onboarding.logInScreen.
```

|     |                                |             |
|-----|--------------------------------|-------------|
| v   | isLoginScreenLoaded            | LogInScreen |
| ↑ m | tapSignInButton ()             | LogInScreen |
| ↑ m | setPassword (password: String) | LogInScreen |
| ↑ v | TAG                            | String      |
| ↓ m | setUserName (userName: String) | LogInScreen |

# Page Factory: Тестовый сценарий

```
@Test
fun exampleTest() {
    merchantApp.onboarding().loginScreen()
        .isLoginScreenLoaded
        .setUserName("autoqa")
        .setPassword("iddqd")
        .tapSignInButton()
}
```

# Page Factory: Тест

```
merchantApp.onboarding().loginScreen()  
    .isLoginScreenLoaded  
    .setUserName("user")  
    .setPassword("idkfa")  
    .tapSignInButton()
```

```
with(merchantApp.onboarding().loginScreen()) { this: LoginScreen  
    isLoginScreenLoaded  
    setUserName("user")  
    setPassword("idkfa")  
    tapSignInButton() ^with  
}
```

- Тест содержит только шаги: как тест-кейс
- Можно группировать шаги для наглядности

## Page Factory: Шаги, возвращающие значение

... а переводящие на следующий экранный элемент

```
val actualValue = merchantApp.menu().sideMenu().getUserNameLabelText()
```

```
val expectedValue = "demo user"
```

```
Assert.assertEquals(expectedValue, actualValue, "User name is not correct")
```

# Пример: Как мы напишем авто-тест

## Пример сценария:

- Запуск приложения
- Ввод логина и пароля
- Ожидание следующего экрана
- Открытие меню
- Выбор пункта
- Проверка что нужный экран показан

```

class SignInTest : BaseTest() {

    @Test(description = "my test")
    fun myTest() {

        merchantApp.onboarding().loginScreen()
            .isLoggedIn()
            .setUserName("user")
            .setPassword("1qkfa")
            .tapSignInButton()

        merchantApp.activeOrder().activeOrder()
            .isActiveOrderScreenShown()

        merchantApp.menu()
    }
}

```

• header()      Header
   
 • sideMenu()      SideMenu
   
 • equals(other: Any?)      Boolean
   
 • toString()      String
   
 • hashCode()      Int
   
 • to(that: B) for A in kotlin      Pair<MenuProvider, B>
   
 • javaClass for T in kotlin.jvm      Class<MenuProvider>
   
 • also {...} (block: (MenuProvider) -> Unit) fo\_ MenuProvider
   
 • apply {...} (block: MenuProvider.() -> Unit) - MenuProvider
   
 • let {...} (block: (MenuProvider) -> R) for T in kotlin
   
 • run {...} (block: MenuProvider.() -> R) for T in kotlin
   
 • runOnUiThread { } / AsyncTask / ...
   
 Press ⌘ to insert, ⌘ to replace, Next Tip



# “Оптимизация” Page Factory

Создадим объекты не страничек, а действий на страницах

# Библиотека действий с элементами

Основные действия над элементами также можно вынести в отдельную библиотеку:

- ожидание
- нажатие
- ввод текста
- получение текста
- прокрутка
- ...

```
@Step( value: "Set user name: '{userName}'")
fun setUsername(userName: String): LogInScreen = also {
    |   userNameInput?.get(0)!!.sendKeys(userName)
    }
}
```

# Использование библиотеки действий в шагах теста

```
fun setUsername(userName: String): LogInScreen = also { it: LogInScreen
|   sendKeys(userNameInput, userName)
| }

fun setPassword(password: String): LogInScreen = also { it: LogInScreen
|   sendKeys(passwordInput, password)
| }

fun tapSignInButton(): LogInScreen = also { it: LogInScreen
|   tap(signInButton)
| }

fun getSignInButtonText(): String {
|   return getText(signInButton)
| }
```

## “Оптимизация” действий над элементами

**Цель:** Мы хотим иметь возможность быстро понять где что-то пошло не так.

Возможные причины падения теста:

- Элемент отсутствует на экране
- Элемент имеет некорректное состояние: видимость, доступность...
- Нажатие не может быть выполнено
- Текст не был введен в поле
- ...

Appium падает так же как Selenium, только *мобильно* =)

# Обработка ошибок: реализация

## Идея:

- Действия над элементами должны возвращать Boolean значение своего результат.
- Шаги тестов проверяют результат действий.

## Что делаем:

- Добавляем: обработку результатов вызова Arrium команд в библиотеку действий
- Добавляем: проверки в шаги теста, использующие действия

# Пример: ввод текста

```
fun sendKeys(elements: List<MobileElement>?, num: Int, txt: String?): Boolean {  
    Засечем время начала попыток  
    do {  
        try {  
            Попытаемся просто ввести текст в элемент. Если  
            получилось, вернем “Все ок”  
        } catch (e: ElementNotInteractableException) {  
            Если не нажимается, можно попробовать еще раз  
        } catch (e: Exception) {  
            Если все упало, вернем “Не удача”  
        }  
        Сделаем паузу между попытками  
    } while Пока время на попытки не вышло  
    return Вернем результат что получилось  
}
```

## Пример: ввод текста

```
fun sendKeys(elements: List<MobileElement>?, num: Int, txt: String?): Boolean {
    val startTime = System.currentTimeMillis()
    do {
        try {
            (elements?.get(num) as MobileElement).sendKeys(txt)
            sleep(Timer.TAP_DELAY)
            return true
        } catch (e: ElementNotInteractableException) {
            log(text: TAG + "sendKeys(): retry...")
        } catch (e: Exception) {
            e.printStackTrace()
        }
        sleep(Timer.TAP_DELAY)
    } while (System.currentTimeMillis() < startTime + 10 * 1000) // 10 sec
    return false
}
```

## Шаги теста с проверкой действий с элементами

```
fun setUsername(userName: String): LogInScreen = also { it: LogInScreen
    Assert.assertTrue(sendKeys(userNameInput, userName), "${TAG}setUserName(): FAILED")
}
```

```
fun setPassword(password: String): LogInScreen = also { it: LogInScreen
    Assert.assertTrue(sendKeys(passwordInput, password), "${TAG}setPassword(): FAILED")
}
```

```
fun tapSignInButton(): LogInScreen = also { it: LogInScreen
    Assert.assertTrue(tap(signInButton), "${TAG}tapSignInButton(): FAILED")
}
```

**TAG** - Метка с именем текущего класса



# Расширение возможностей теста

Тест красив и стабилен =)

Пора провести небольшую подготовительную работу перед тестами =)

# TestNG: Шаги “вокруг” теста

Тест работает. Как добавить дополнительные действия до и после тестирования?



# TestNG: Управление тестом

**Функции** TestNG для дополнительных действий до и после теста

- До прогона тестов: `@BeforeSuite`
- Перед каждым тестом: `@BeforeMethod`
- После каждого теста: `@AfterMethod`
- После прогона всех тестов: `@AfterTest`
- После окончания тестирования: `@AfterSuite`

**Документация:**

<https://testng.org/doc/documentation-main.html#annotations>

# Управление тестом: применение TestNG

## Примеры:

- Запуск и остановка Appium сервера: `@Before/After Suite`
- Сброс кеша приложения, очистка “мусора” после теста: `@Before/After Method`
- Сбор статистики о тестировании: `@AfterTest`

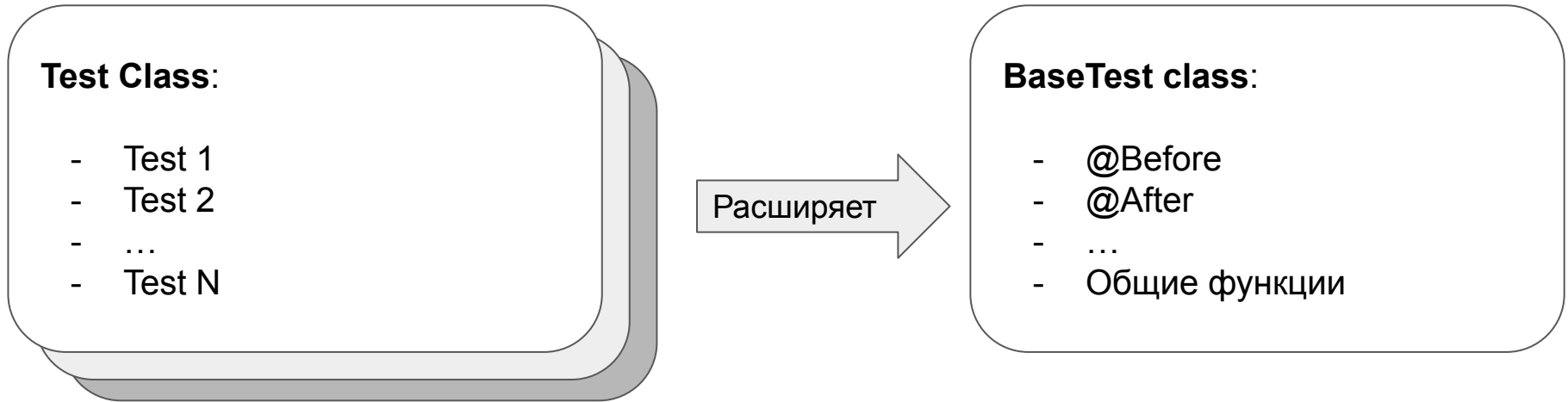
# Управление тестом: применение TestNG

## Функции с аннотациями TestNG

```
@Parameters( ...value: "product", "deviceName", "runner")
@BeforeSuite(alwaysRun = true)
fun beforeSuite(
    @Optional( value: "") product: String,
    @Optional( value: "") deviceName: String,
    @Optional( value: "") runner: String
) {...}
```

```
@AfterSuite(alwaysRun = true)
fun tearDown(iTestContext: ITestContext) {...}
```

# Иерархия тестовых классов



# Отчеты

Наш тест работает, умеет готовить окружение и даже подчищает за собой временные данные. Как представить результат работы теста?

```
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 300.554 s - in TestSuite
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 05:19 min
[INFO] Finished at: 2021-08-13T14:40:10+03:00
[INFO] -----
```

# Текстовый лог о ходе теста в консоли

Во время разработки и при тестирование продукта:

- Показывает что тест делает в данный момент
- Сохраняет полную историю выполнения теста для анализа
- Показывает ход выполнения, когда не видно экрана устройства



# Текстовый лог: реализация

```
fun setUsername(userName: String): LogInScreen = also { it: LogInScreen
    log( text: TAG + "setUserName(): " + userName)
    Assert.assertTrue(sendKeys(userNameInput, userName), "${TAG}setUserName(): FAILED")
}
```

```
fun setPassword(password: String): LogInScreen = also { it: LogInScreen
    log( text: TAG + "setPassword(): ")
    Assert.assertTrue(sendKeys(passwordInput, password), "${TAG}setPassword(): FAILED")
}
```

```
fun tapSignInButton(): LogInScreen = also { it: LogInScreen
    log( text: TAG + "tapSignInButton():")
    Assert.assertTrue(tap(signInButton), "${TAG}tapSignInButton(): FAILED")
}
```

# Текстовый лог в работе

```

BeforeMethod
[SYSTEM] Test parameters: checkSideMenuItemsListTest, product: merchantApp, env: staging
[SYSTEM] startDriver(): LOCAL
[SYSTEM] startDriver(): appFile: /Users/mikhailmiroshnichenko/Documents/Wolt/MerchantTest.app
[SYSTEM] startDriver(): Driver DO FAST RESET
[SYSTEM] startDriver(): Launch parameters: -skipIntros
[SYSTEM] startDriver(): deviceName: iPad (8th generation), OS version: 14.5
[SYSTEM] startDriver(): Local IOSDriver
Jul 07, 2021 5:28:16 PM io.appium.java_client.remote.AppiumCommandExecutor$1 lambda$0
INFO: Detected dialect: W3C
[SYSTEM] startDriver(): completed in 13 sec
[TEST] checkSideMenuItemsListTest LogInScreen() | isLoginScreenLoaded():
[TEST] checkSideMenuItemsListTest LogInScreen() | setUsername(): tablet
[TEST] checkSideMenuItemsListTest LogInScreen() | setPassword():
[TEST] checkSideMenuItemsListTest LogInScreen() | tapSignInButton():
[TEST] checkSideMenuItemsListTest Header() | isHeaderLoaded():
[TEST] checkSideMenuItemsListTest Header() | tapMenuButton():
[TEST] checkSideMenuItemsListTest SideMenu() | isSideMenuShown():
[TEST] checkSideMenuItemsListTest SideMenu() | getActiveOrdersItemText():
[TEST] checkSideMenuItemsListTest SideMenu() | getPastOrdersItemText():
[TEST] checkSideMenuItemsListTest SideMenu() | getEditMenuItemText():
[TEST] checkSideMenuItemsListTest SideMenu() | getTutorialVideosItemText():
[TEST] checkSideMenuItemsListTest SideMenu() | getWoltSupportItemText():
[TEST] checkSideMenuItemsListTest SideMenu() | getSettingsItemText():
[TEST] checkSideMenuItemsListTest SideMenu() | getWhatsNewItemText():

AfterMethod
Test result: PASSED
[SYSTEM] quit driver

```

## “Оптимизация” отчетов

От статичного текстового лога, привязанного в исполнению теста, к подробному отчету с картинками и на человеческом языке. Где-то на публичном сервере.

# Создание отчетов с помощью Allure

**Для чего:** Мы хотим иметь возможность легко понять и увидеть статус протестированного приложения.

**Как начать использовать:**

- Подключить библиотеку к проекту с Maven
- Добавить аннотации к классам, тестам и шагам
- Сгенерировать отчет

Важно: Allure - это веб-сервер.

Доклад А. Ерошенко на Heisenbug: <https://www.youtube.com/watch?v=tTV00kSutQI>

# Allure: Пример

Представим, что мы прогнали тесты. Посмотрим, как выглядит веб страничка с отчетом.

## Suites

group name duration status

Status: ● ● ● ● ● Marks:  

▼ Merchant iPad app full set - Thread 1

▼ Full

▼ tests.merchantApp.order.OrderTest

✔ #2 Get order with options and display options list in order cell autoqaConsumer, i... 22s 318ms

✔ #1 Get order, check details and complete it by moving cell autoqaConsumer, iPa... 1m 53s

▼ tests.merchantApp.settings.GeneralSettingsTest

✔ #1 Verify General Settings items list, Headers, Titles, Descriptions autoqaConsumer, iPa... 3m 37s

✔ #2 Verify Settings autoqaConsumer, iPa... 24s 131ms

tests.merchantApp.order.OrderTest.getAnd...

Passed **Get order with options and display options list in order cell**

Overview History Retries

Severity: normal

Duration: ⌚ 22s 318ms

### Parameters

consumer: autoqaConsumer  
 deviceName: iPad8th,iOS14-  
 env: staging  
 launchParameters: -skipintros  
 product: merchantApp  
 runner: browserstack  
 startType: fastReset  
 venue: autoqaVenue

### Execution

> Set up

▼ Test body

✔ Check 'Log In' screen loaded

224ms

# Allure: Добавление аннотаций к классу с тестами

```
@Epic( value: "autoqa demo")
@Feature( value: "Using Allure for reporting")
class SignInTest : BaseTest() {

    @Test(description = "Sign in with correct Venue credentials", groups = ["ios"])
    fun signInWithCorrectCredentialsTest() {

        merchantApp.onboarding().signIn()
    }
}
```

# Allure: Добавление аннотаций к классу с экраном

```
@Step("Set user name: '{userName}'")
fun setUsername(userName: String): LogInScreen = also { it: LogInScreen
    log( text: TAG + "setUserName(): " + userName)
    Assert.assertTrue(sendKeys(userNameInput, userName), "${TAG}setUserName(): FAILED")
}
```

```
@Step("Set password")
fun setPassword(password: String): LogInScreen = also { it: LogInScreen
    log( text: TAG + "setPassword(): ")
    Assert.assertTrue(sendKeys(passwordInput, password), "${TAG}setPassword(): FAILED")
}
```

```
@Step("Tap 'Sign Up' button")
fun tapSignInButton(): LogInScreen = also { it: LogInScreen
    log( text: TAG + "tapSignInButton():")
    Assert.assertTrue(tap(signInButton), "${TAG}tapSignInButton(): FAILED")
}
```



# Allure: Веб страничка с отчетом о тестировании

The screenshot displays the Allure test report interface. On the left, the 'Suites' section shows a tree view with the following structure:

- Suites (Search bar, Info, Download icons)
- order, name, duration, status (filters)
- Status: 0 0 1 0 0 (colored indicators)
- Marks: [Eye icon] [Warning icon]
- Mobile UI Tests (1 passed)
- TEST suite (1 passed)
- tests.merchantApp.signIn.SignInTest (1 passed)
- #1 Sign in with correct Venue credentials (8s 684ms)

On the right, the 'Parameters' section lists the following details:

- consumer: autoqaConsumer
- deviceName: iPad8th\_iOS14
- env: staging
- launchParameters: -skipIntros
- product: merchantApp
- runner: browserstack
- venue: autoqaVenue2

The 'Execution' section shows the following steps and durations:

- Set up
  - beforeMethod: 29s 383ms
  - beforeSuite: 41ms
- Test body
  - Check 'Log In' screen loaded: 547ms
  - Set user name: 'tablet2' 1 parameter: 1s 805ms
  - Set password 1 parameter: 1s 789ms
  - Tap 'Sign Up' button: 1s 856ms
  - Check 'Header' shown: 2s 447ms
- Tear down

# Allure: Обзор истории прогонов



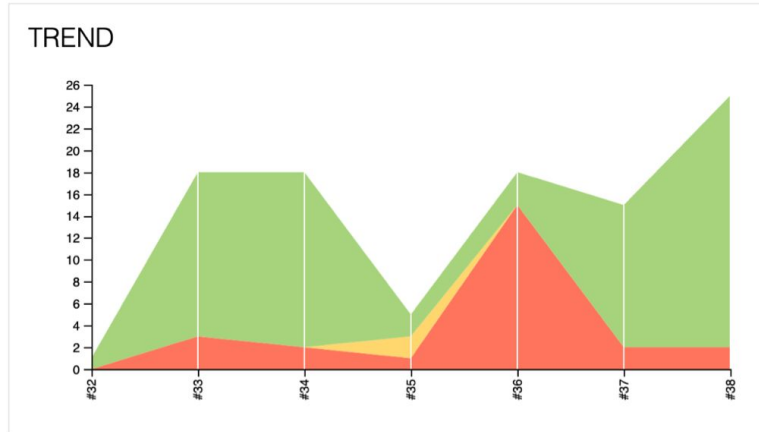
SUITES 4 items total

|   |   |    |
|---|---|----|
| ee.nortal.appium.tests.NewsScreenTest       | 2 | 13 |
| ee.nortal.appium.tests.LoginScreenTest      | 0 | 5  |
| ee.nortal.appium.tests.UserMenuTest         | 0 | 4  |
| ee.nortal.appium.tests.OnboardingScreenTest | 0 | 1  |

Show all

ENVIRONMENT

|            |                        |
|------------|------------------------|
| user.email | ta.PUB_VIEW@nortal.com |
| user.psw   | TestAutomationPsw      |



CATEGORIES 2 items total

|                 |   |
|-----------------|---|
| Product defects | 2 |
| Unknown problem | 2 |

Show all

EXECUTORS

Jenkins Mobile app UI Tests #38 [↗](#)

# Allure: Список текстов и их шаги

## Suites

order ⌵ name ⌵ duration ⌵ status ⌵

Marks: 👁 ⚠

Status: 2 0 23 0 0

ee.nortal.appium.tests.LoginScreenTest 5

- ✔ #3 Display error message when data entered to login form has errors 9s 091ms
- ✔ #5 Display error when entering incorrect credentials 23s 736ms
- ✔ #4 Login to app with valid credentials 26s 553ms
- ✔ #2 User can leave Login process 12s 193ms
- ✔ #1 User can return to interrupted log in process 10s 295ms

ee.nortal.appium.tests.NewsScreenTest 2 13

- ✔ #11 Add rating and comment to news with authorised user 1m 06s
- ✔ #6 Display News item details 36s 175ms
- ✔ #2 Filter news list with 'Filter form by Date range 18s 171ms
- ✔ #10 Filter news list with Filter form by Category 23s 652ms
- ✔ #8 Filter news list with Filter form by Location 29s 222ms
- ✔ #7 Filter news with top main menu 18s 492ms
- ✔ #12 Forbid adding rating and comment to news item to authorised user 24s 757ms
- ✘ #9 Open news item from Similar stream 42s 103ms
- ✔ #5 Open News screen 10s 258ms
- ✔ #15 Scroll news list 22s 381ms
- ✘ #13 Similar stream contains news with the same tags 35s 960ms
- ✔ #1 Sort news, by date (highest first) 22s 468ms
- ✔ #14 Sort news, by date (lowest first) 21s 309ms

ee.nortal.appium.tests.NewsScreenTest.should\_allow\_adding\_rating\_an...

Passed **Add rating and comment to news with authorised user**

Overview
History
Retries

Severity: normal

Duration: ⌵ 1m 06s

**Description**

User can add rating and comment to news, if he has logged in.

**Links**

🔗 WSAI2-565 , 📄 WSAI2-648

**Execution**

⌵ **Test body**

- Create News item 'News item 1' 4 parameters 1s 438ms
- ✔ Wait for Onboarding screen opened 4s 093ms
- ✔ Tap Skip Button 273ms
- ✔ Tap Lets Start button 1s 272ms
- ✔ Wait for Dashboard screen opened 601ms
- ✔ Tap user icon 720ms
- Enter ta.PUB\_VIEW@nortal.com to Email field 1 parameter 8s 317ms
- Enter TestAutomationPsw to Email field 1 parameter 6s 962ms
- ✔ Tap Login button 723ms
- ✔ Wait for Dashboard screen opened 1s 635ms
- ✔ Tap News button on toolbox 286ms

# Allure: Вывод причины падения теста

Categories

order  name  duration  status  Status: 4 0 0 0 0

Marks:

- Product defects 2
  - Similar section should be displayed! 2
    - #1 Open news item from Similar stream 42s 103ms
      - #2 Similar stream contains news with the same tags 35s 960ms
- Unknown problem 2
  - Similar section should be displayed! 2
    - #1 Open news item from Similar stream 42s 103ms
      - #2 Similar stream contains news with the same tags 35s 960ms

ee.nortal.appium.tests.NewsScreenTest.should\_display\_similar\_news\_d...

## Failed Open news item from Similar stream

Overview History Retries

Similar section should be displayed!

Categories: Unknown problem Product defects

Severity: normal

Duration: 42s 103ms

### Description

Tap on news item in Similar screen to view its details

### Links

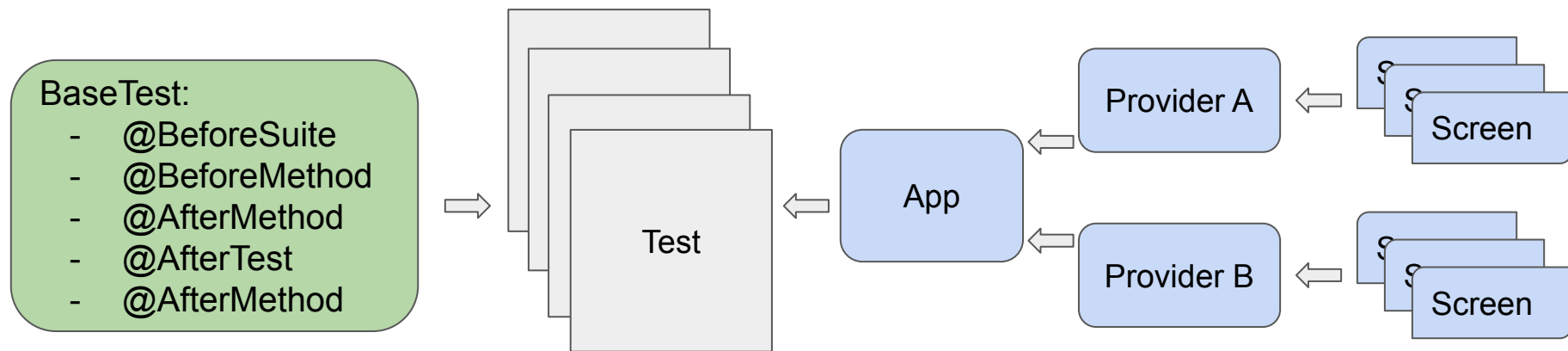
WSAI2-565, WSAI2-637

### Execution

#### Test body

- Create News item 'News item 1' 4 parameters 386ms
- Create News item 'News item 2' 4 parameters 351ms
- Wait for Onboarding screen opened 2s 955ms
- Tap Skip Button 394ms
- Tap Lets Start button 1s 688ms
- Wait for Dashboard screen opened 1s 055ms
- Tap News button on toolbox 664ms
- Wait for News screen opened 1s 243ms
- Select AT parent category 1 parameter 1 sub-step Re: 185ms

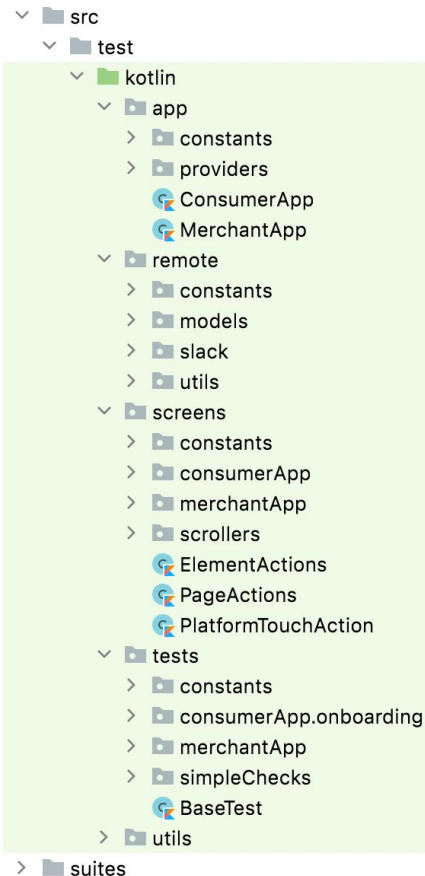
# Финальная архитектура проекта



# Где запускать автотесты?

Теперь у нас есть что запускать и чем запускать. Но где это запустить, чтобы получить быстрый отчет о состоянии приложения?

- **Самостоятельно:** разработчик и тестировщик вручную
- **На серверах Continuous Integration:**
  - Push в ветку новой задачи
  - Создание Pull request
  - Merge to Master



# Часть 4: Запуск автотестов

Запускаем созданные автотесты у себя и у соседа

# Запуск теста на локальной машине

- Реальное устройство
  - Эмулятор / Симулятор
  - Все необходимые драйверы и инструменты
- + Работает максимально быстро (поскольку рядом с агентом, исполняющим тест)
- Требуется соответствующее аппаратное обеспечение



# Реальное устройство

- + Не важна производительность агента
  - + Реальное программное окружение, как у пользователя
  - + Доступны все популярные сервисы Google, Apple и т.д.
- 
- Ограничения по доступу к определенным функциям
  - Надо покупать отдельно
  - Необходимо самостоятельно обслуживать устройства

# Эмулятор

- + Любая конфигурация без дополнительных затрат
- + Возможно все
  
- Нет популярных сервисов
- Высокие требования к производительности агента

# iOS устройства

## Включает:

- Симулятор iOS, идущий в составе IDE XCode для macOS
  
- iPhone / iPad

# iOS устройства

## Особенности:

- Нужен macOS
- Требуется аккаунт разработчика для сборки приложения на реальное устройство
- Многие функции заблокированы на реальном устройстве.
- Интерфейс строится по разному в зависимости от версии ОС
- Приложение для симулятора: **.app** x86 (ARM на устройстве с M1)
- Приложение для устройства: **.ipa** ARM

# Android устройства

## Включает:

- Эмулятор Android Studio: AVD - Android Virtual Device, Win / macOS
- Реальное устройство

# Android устройства

## Особенности:

- Можно собрать дистрибутив приложения apk, который будет работать и в эмуляторе, и на любом реальном устройстве
- Просто установить приложение на любое устройство

# Пример: Выполнение теста на симуляторе iPad

На примере **Wolt Merchant App**: приложения для ресторанов для приема и обработки заказов

**Тест:** Отмена полученного заказа

- Войти в приложение
- Отправить заказ в приложение
- Отменить принятый заказ

Как работает:

- На локальной машине
- На iPad симуляторе





## Запуск теста на другой машине: все и сразу

- **С компиляцией проекта и на подключенном устройстве**
  - Java
  - maven
  - Appium
  - iOS: XCode
  - Android: Android debug tools
  - Allure (для отчетов)

# Запуск теста на другой машине: на ферме устройств

- С компиляцией проекта и на подключенном устройстве
  - Java
  - maven
  - Appium
  - iOS: XCode
  - Android: Android debug tools
  - Allure (для отчетов)
- **С компиляцией и на удаленном устройстве**
  - Java
  - maven
  - Appium не нужен: он настроен на ферме устройств

# Запуск теста на другой машине: в образе системы

- С компиляцией проекта и на подключенном устройстве
  - Java
  - maven
  - Appium
  - iOS: XCode
  - Android: Android debug tools
  - Allure (для отчетов)
- С компиляцией и на удаленном устройстве
  - Java
  - maven
  - Appium не нужен: он настроен на исполнителе тестов
- **В Docker** с уже собранным проектом автотестов - отлично для CI

# Использование Docker: dockerfile

```
FROM maven:latest

# Create folders
RUN mkdir -p /usr/tests && \
    mkdir -p /usr/tests/src && \
    mkdir -p /usr/tests/suites \
    mkdir -p /usr/external_files

# Define working dir with test files
WORKDIR /usr/tests

# Copy source files
COPY src /usr/tests/src

# Copy suites
COPY suites /usr/tests/suites

# Copy main MVN file
COPY pom.xml /usr/tests

# Copy device settings file
COPY capabilities.json /usr/tests

# Pre install maven dependencies
RUN mvn install
```

# Использование Docker: Порядок сборки

Создание файла с описанием образа на основе Maven образа:

[https://hub.docker.com/\\_/maven](https://hub.docker.com/_/maven)

Сборка образа из файла:

```
docker build -t autoqa -f DockerfileTest .
```

Запуск тестов в контейнере:

```
docker run --rm autoqa /bin/bash -c "mvn  
-DsuiteName=mySuite.xml test"
```

# Использование Docker: Работа в контейнере

На чем запускать тесты при работе в контейнере?

Запуск тестов в контейнере:

```
docker run - - rm autoqa /bin/bash -c "mvn  
-DsuiteName=mySuite.xml test"
```

На ферме устройств

# Облачные фермы

Что делать, когда надо запускать приложения и тесты на реальных устройствах, но их у вас нет (или есть, но недостаточно).

Или девайсы купить можно, но вот поддерживать их не хочется.

**Решение:** Облачные сервисы с реальными устройствами (Mobile / Web)

- BrowserStack
- SauceLabs
- LambdaTest (beta)

# Облачная ферма устройств

- Предоставляется по подписке
- Имеет парк распространенных мобильных устройств с разными версиями ОС
- Имеет парк web-окружений с популярными браузерами и их версиями.
- Позволяет запускать автотесты
- Дает Live-доступ к устройствам через веб-браузер



# Облачная ферма устройств

Представим, что мы запустили наши тесты на устройстве сервиса BrowserStack...

- 4/4 PASSED + 4 messages + Mocked iPad app
- 2021-08-23T09:16:03
- 1/1 PASSED + 1 message + Mocked iPad app
- 2021-08-23T09:12:04
- 1/1 PASSED + 1 message + Mocked iPad app
- 2021-08-23T09:10:53
- 4/4 PASSED + 4 messages + Mocked iPad app
- 2021-08-23T09:05:19
- 6/6 PASSED + 6 messages + Mocked iPad app
- 2021-08-23T09:02:27
- 1/1 PASSED + 1 message + Mocked iPad app
- 2021-08-23T09:00:17
- 1/1 PASSED + 1 message + Mocked iPad app
- 2021-08-23T09:00:11

Local Testing  OFF

User Mikhal Miroshchikova

Session ID: 1baef94...

Protocol: MJSONWP

TEST RUN Success Test passed



| App      | Input Capabilities              | Device Capabilities                          |
|----------|---------------------------------|--|
| id       |                                 | 8c://51808154565078f0dc7966c63182175a1798a6c |
| appName  | com.walruslabs-test-integration |  |
| bundleId | com.walruslabs-test-integration |  |
| version  | 1.16.1                          |  |
| customId |                                 | MockedAppMocker                              |

# BrowserStack: Список прогонов тестов

DASHBOARD  ACCESS KEY

**FILTERS**

Merchant iPad app

1/1 PASSED • a day ago • Merchant iPad app

2021-07-21T16-07-10

1/1 PASSED • a day ago • Merchant iPad app

2021-07-21T16-00-39

1/5 FAILED • a day ago • Merchant iPad app

**2021-07-21T15-50-27**

1/5 FAILED • a day ago • Merchant iPad app

2021-07-21T12-44-53

1/1 PASSED • a day ago • Merchant iPad app

2021-07-21T15-36-01

5/5 PASSED • a day ago • Merchant iPad app

2021-07-21T15-27-05

2021-07-21T15-50-27

Merchant iPad app

Sessions **5** • FAILED Build Status **ERROR**

Last Updated **21 Jul 2021 12:55 UTC** User **Mikhail Miroshnichenko**

Duration **4m 33s** Build ID **e29becd...**

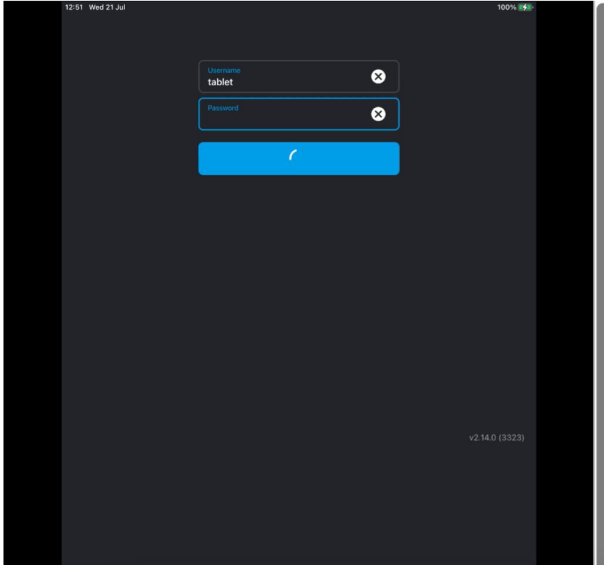
ALL SESSIONS (0) COMPLETED (0) TIMED OUT (0) ERROR (0)

| Session Name                               | Status  | REST API | OS Version | Devices  | Duration | Last Updated |
|--|---|----------|------------|----------|----------|--------------|
| checkPastOrdersScreenTest                  | <input checked="" type="checkbox"/> COMPLETED | PASSED   | iOS 14.0   | IPAD 8TH | 57s      | a day ago    |
| shouldDisplayWhatsNewPopUpAtSignIn         | <input checked="" type="checkbox"/> COMPLETED | PASSED   | iOS 14.0   | IPAD 8TH | 53s      | a day ago    |
| shouldOpenWhatsNewScreenByWhatsNewMenuItem | <input checked="" type="checkbox"/> COMPLETED | PASSED   | iOS 14.0   | IPAD 8TH | 50s      | a day ago    |

# BrowserStack: Просмотр информации о сессии

|          |                        |            |                       |               |         |
|----------|------------------------|------------|-----------------------|---------------|---------|
| OS       | iOS 14.0               | Status     | Completed             | Local Testing | Off     |
| Device   | iPad 8th               | REST API   | FAILED                | Protocol      | MJSONWP |
| Duration | 47s                    | Started At | 21 Jul 2021 12:50 UTC |               |         |
| User     | Mikhail Miroshnichenko | Session ID | 7908f68...            |               |         |

REST API Reason: Header() | isHeaderLoaded();



Text Logs   Network Logs   Device Logs   Appium Logs   Crash Logs (0)







SHOW:  Visuals    No Exceptions Found   [View Raw Logs](#)


- 00:00 ▶ Initialising Device
- 00:00 ▶ Downloading App
- 00:00 ▶ Installing App
- 00:00 ▶ Setting Up Appium
- 00:00 ▶ Setting Up Network Connection
- 00:00 ▶ Launching App
- 00:37 ▶ Retrieves the current context


NATIVE\_APP

# BrowserStack: Live сессия - настройка

**SELECT SOURCE**

-  Test with a Sample app  
apk/ipa files are ready, select any device
-  Upload your App  
Upload your `.ipa`, `.apk`, `.aab` files here.
-  Sync with App Center
-  Install via TestFlight
-  Install via Play Store
-  Install via App Store

 ANDROID  
REAL DEVICES

 iOS  
REAL DEVICES

|  | iPhone (32)       | iPad (22) |                |    |               |   |
|--|-------------------|-----------|----------------|----|---------------|---|
|  | iPhone 12         | 14        | iPhone 8       | 12 | iPhone 6      | 8 |
|  | iPhone 12 Mini    | 14        | iPhone 7       | 12 | iPhone 6 Plus | 8 |
|  | iPhone 12 Pro Max | 14        | iPhone 6S      | 12 |               |   |
|  | iPhone 12 Pro     | 14        | iPhone X       | 11 |               |   |
|  | iPhone XS         | 14        | iPhone 8       | 11 |               |   |
|  | iPhone XS         | 13        | iPhone 8 Plus  | 12 |               |   |
|  | iPhone XS         | 12        | iPhone 8 Plus  | 11 |               |   |
|  | iPhone XS Max     | 12        | iPhone SE 2020 | 13 |               |   |
|  | iPhone 11 Pro     | 13        | iPhone SE      | 11 |               |   |
|  | iPhone 11         | 14        | iPhone 6S      | 11 |               |   |
|  | iPhone 11         | 13        | iPhone 6S Plus | 11 |               |   |
|  | iPhone 11 Pro Max | 14        | iPhone 6       | 11 |               |   |
|  | iPhone 11 Pro Max | 13        | iPhone 7       | 10 |               |   |
|  | iPhone XR         | 12        | iPhone 6S      | 9  |               |   |
|  | iPhone 8          | 13        | iPhone 6S Plus | 9  |               |   |

Save manual upload steps using our API or integrations. [View documentation](#)



www.willmerhart.com/erc...

Refresh



www.willmerhart.com

Pages

Connecting to real Device Cloud

Starting device iPad 8th v14.0

Downloading app on iPad 8th v14.0

Installing app on iPad 8th v14.0.



# BrowserStack: Live сессия - использование

The screenshot displays the BrowserStack interface for a live session on an iPad 8th v14.0 device. The device screen shows a login form with fields for 'Username' and 'Password', and a 'Sign In' button. The BrowserStack toolbar includes options like 'Switch Device', 'Local testing', 'Zoom', 'DevTools', 'Report a bug', 'Settings', and 'Stop Session'. A 'Log' window on the right shows the session progress: 'Connecting to real Device Cloud', 'Starting device iPad 8th v14.0', 'Downloading app on iPad 8th v14.0', 'Installing app on iPad 8th v14.0', and 'Starting app on iPad 8th v14.0...'. A context menu is open over the device, listing actions such as 'Show Home Screen', 'Install New App', 'Kill/Uninstall', 'Rotate Device', 'Change Location', 'Throttle Network', 'Change Language', and 'iPad 8th v14.0'.

# Запуск теста в облаке

**Шаги** для реализации:

- **Получить** логин-пароль для доступа к BrowserStack
- **Загрузить** приложения в облако
- **Подключиться** к устройству из теста

Документация с примерами кода:

<https://www.browserstack.com/docs/app-automate/appium/getting-started/java>



# Настройка подключения к BrowserStack

```
capabilities.setCapability("device", "Samsung Galaxy S21 Ultra")
```

```
capabilities.setCapability("os_version", "10.0")
```

```
capabilities.setCapability("project", "My First Project")
```

```
capabilities.setCapability("build", "My First Build")
```

```
capabilities.setCapability("name", "Bstack-[Java] Sample Test")
```

```
capabilities.setCapability("app", "my_awesome_android_app")
```

```
driver = AndroidIOSDriver<AndroidElement>(
    URL("https://"+userName+": "+accessKey+"@hub-cloud.browserstack.com/wd/hub"), capabilities);
```

И это все!

# Тесты готовы.

Есть что запускать: тесты для продукта

Есть чем запускать: наш фреймворк

Есть где запускать: в контейнере Docker

Есть на чем запускать: на облачной ферме

Поехали в CI!

## Часть 5: Интеграция с CI

Дополним сборку продукта на CI прогоном автотестов

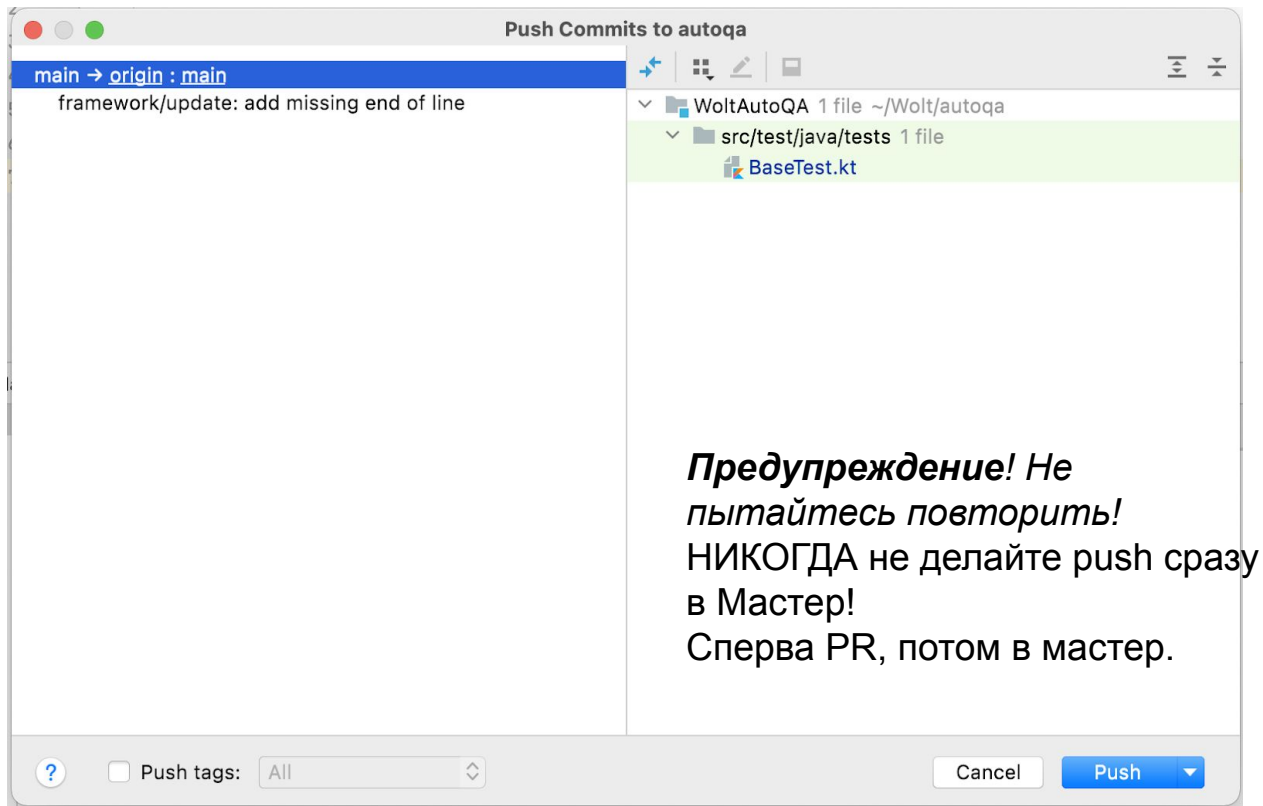
# Интеграция тестов в CI: план

- Хранение проекта в GitHub
- Сборка Docker образа с GitHub Actions
- Хранение ценной информации в GitHub Secrets
- Размещение образа в облаке Amazon Web Storage
- Сборка продукта и прогон тестов на CircleCI
- Отправка результатов тестирования в Slack

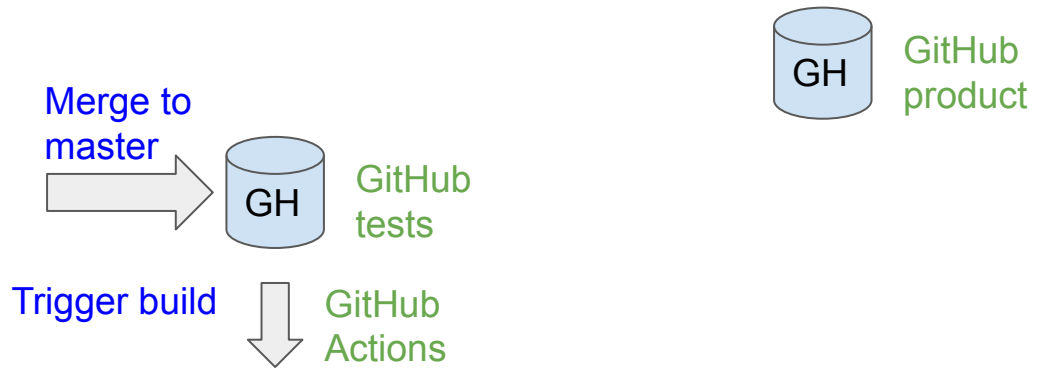
# CI: Репозитории с исходным кодом



# autoqa репозиторий: Push to Master



# CI: Добавление нового кода запустит GH Actions









# GitHub Actions: Сборка проекта автотестов запущена

Workflows

[New workflow](#)[All workflows](#)[main pipeline](#)

## All workflows

Showing runs from all workflows

| 60 workflow runs  |   | Event ▾ | Status ▾ | Branch ▾ | Actor ▾  |
|---|---|---------|----------|----------|--|
|  | <b>framework/update: add missing end of line</b><br>main pipeline #107: Commit dfb9a65 pushed by mikamikaWolt |         |          | main     |  15 seconds ago<br> <i>In progress</i> ... |
|  | <b>MA-1108: Settings (#45)</b><br>main pipeline #106: Commit 82ff121 pushed by mikamikaWolt                   |         |          | main     |  3 days ago<br> 1m 18s ...                 |



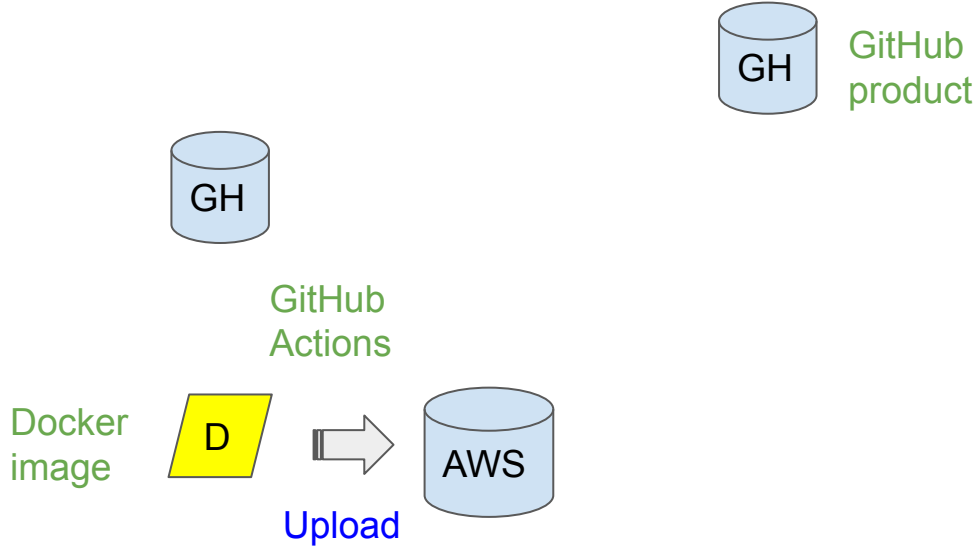
# CI: GH Actions собирает Docker image



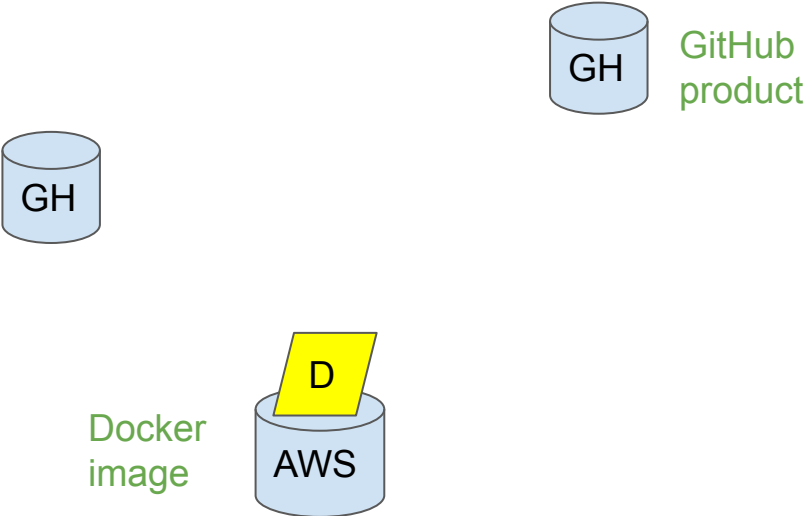
Docker  
image



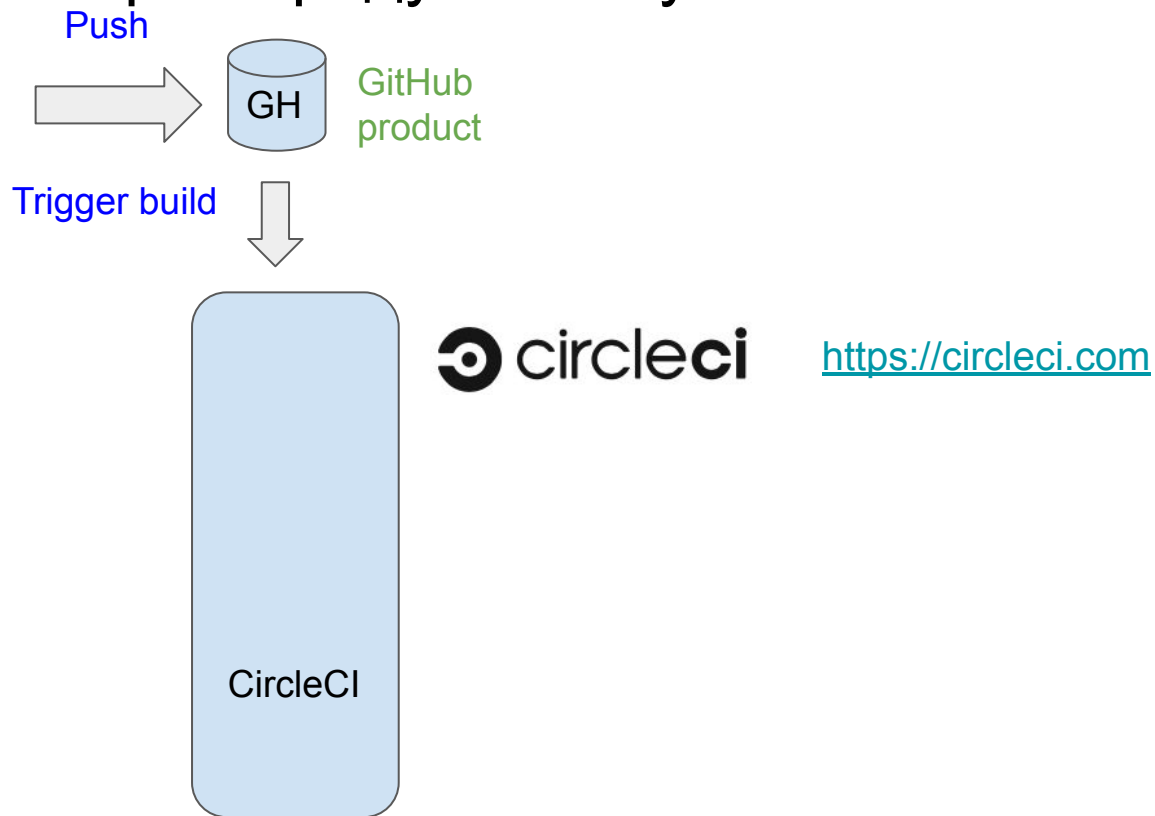
# CI: GH Actions загружает образ на AWS



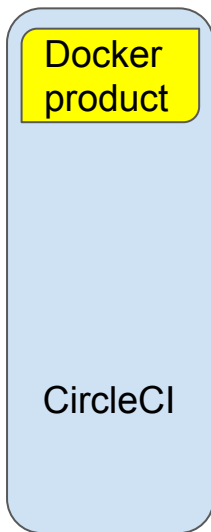
# CI: AWS содержит свежую версию autoqa







# CI: Push в репозиторий продукта запускает CircleCI



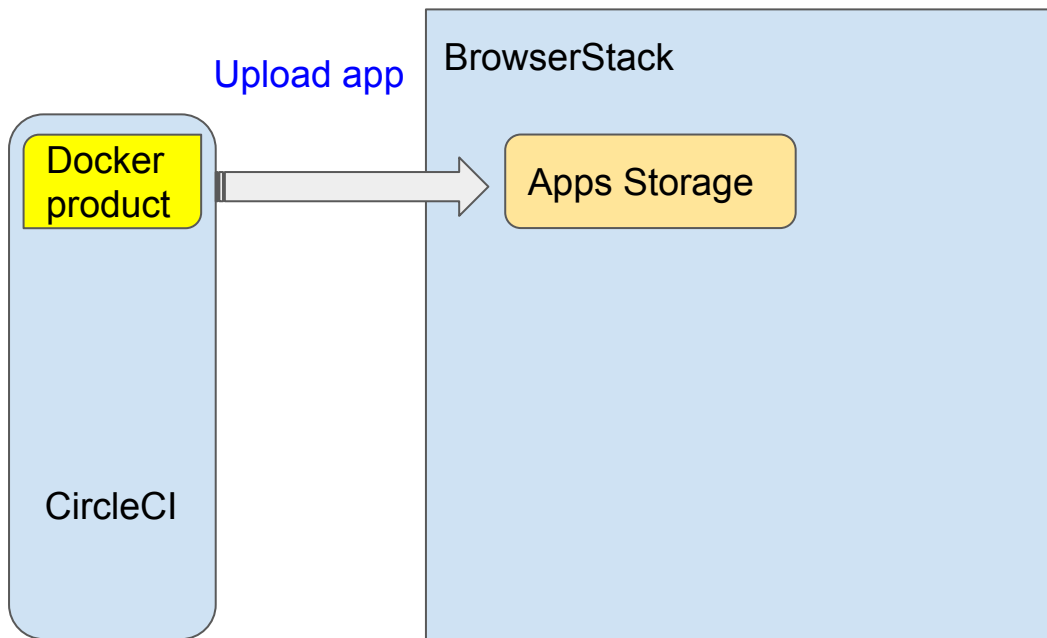
# CI: Сборка продукта в контейнере на базе macOS



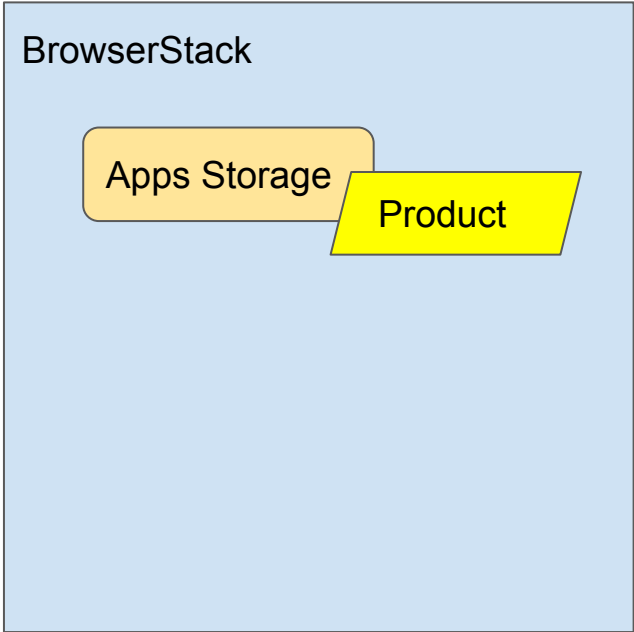
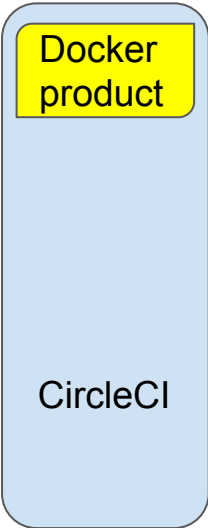
# CircleCI: Начало сборки продукта

| Pipeline               | Status  | Workflow  |
|------------------------|---|-----------|
| wolt-ios-merchant 3652 |  Running        | task_flow |
| <b>Jobs</b>            |   |           |
|                        |  swiftlint 8036  |           |
|                        |  build_task 8038 |           |
|                        |  test_task       |           |

# CI: Загрузка продукта в облачное хранилище

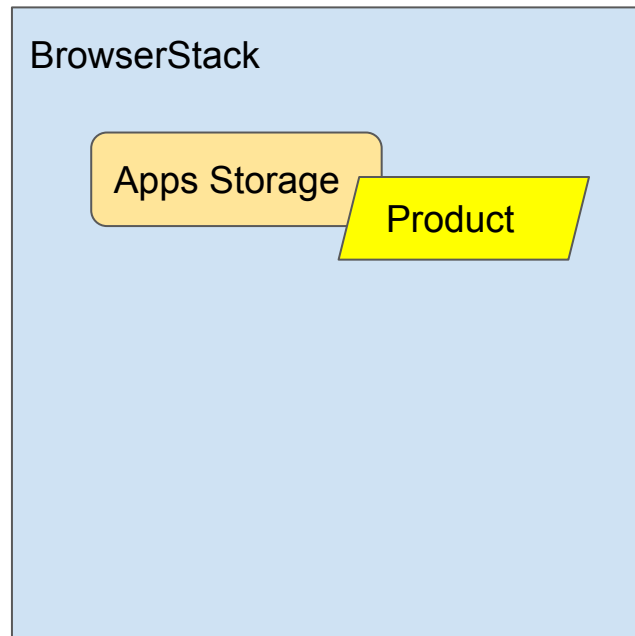
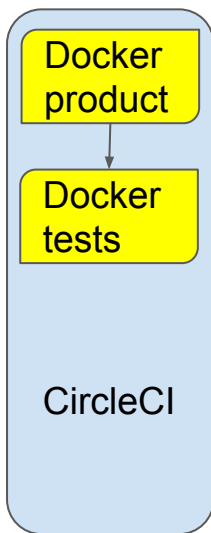


# CI: Продукт готов к использованию на BS






# CI: Начало прогона тестов после сборки продукта



# CircleCI: Job с тестами запущен

 **task\_flow** Running

 Insights

Duration

 15m 18s

Branch

 [feature/ma1120updateMenuEditorUI](#)

Commit

 [30c3c07](#)

Author & Message

 fixed animation flag

 swiftlint

33s

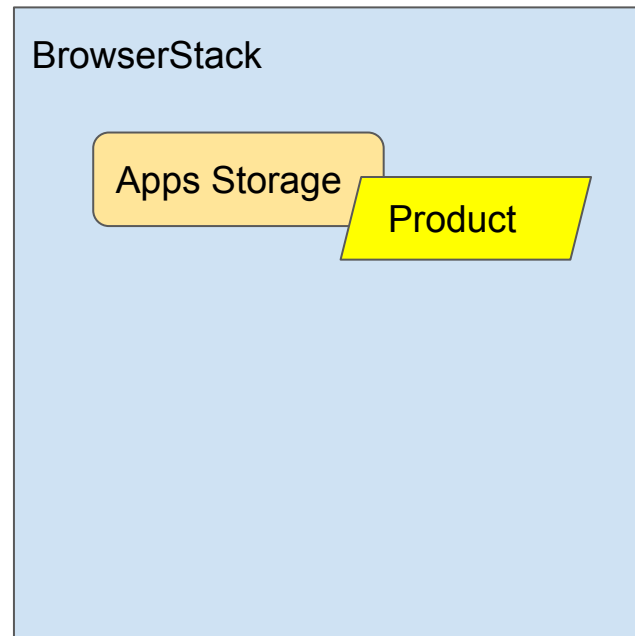
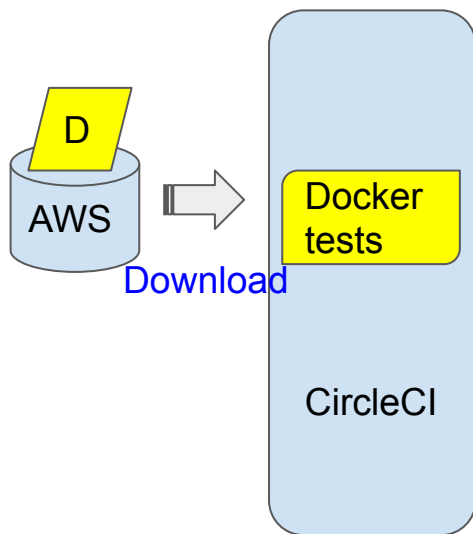
 build\_task

13m 13s

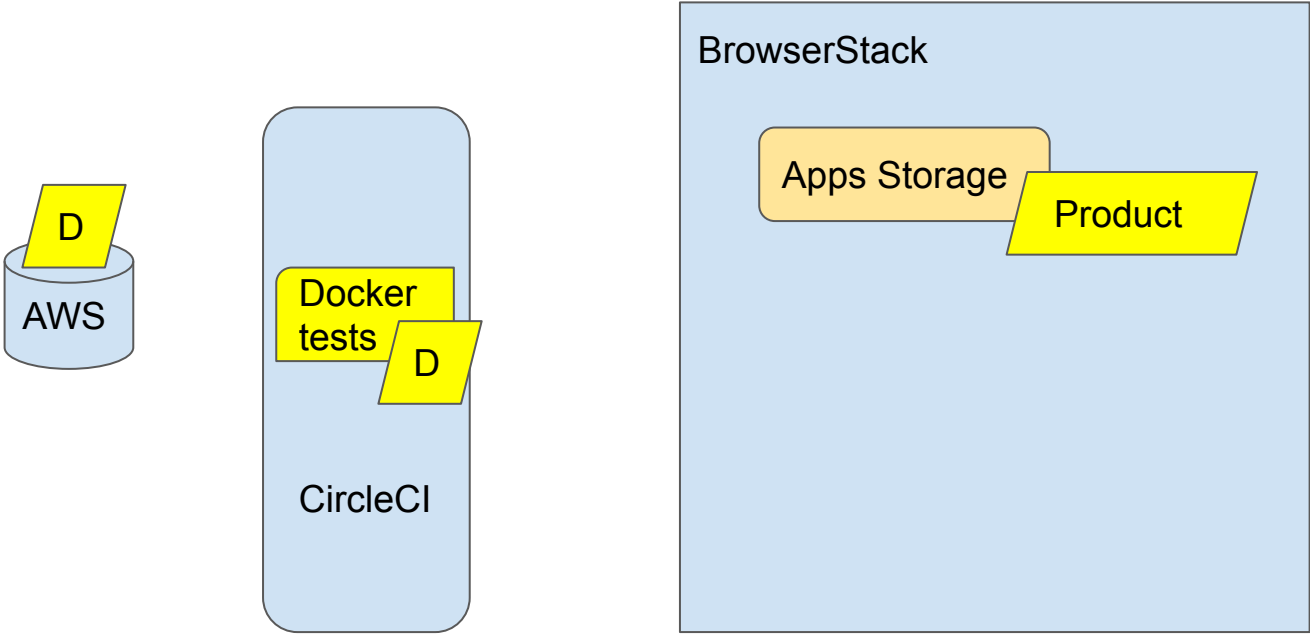
 test\_task

1m 19s

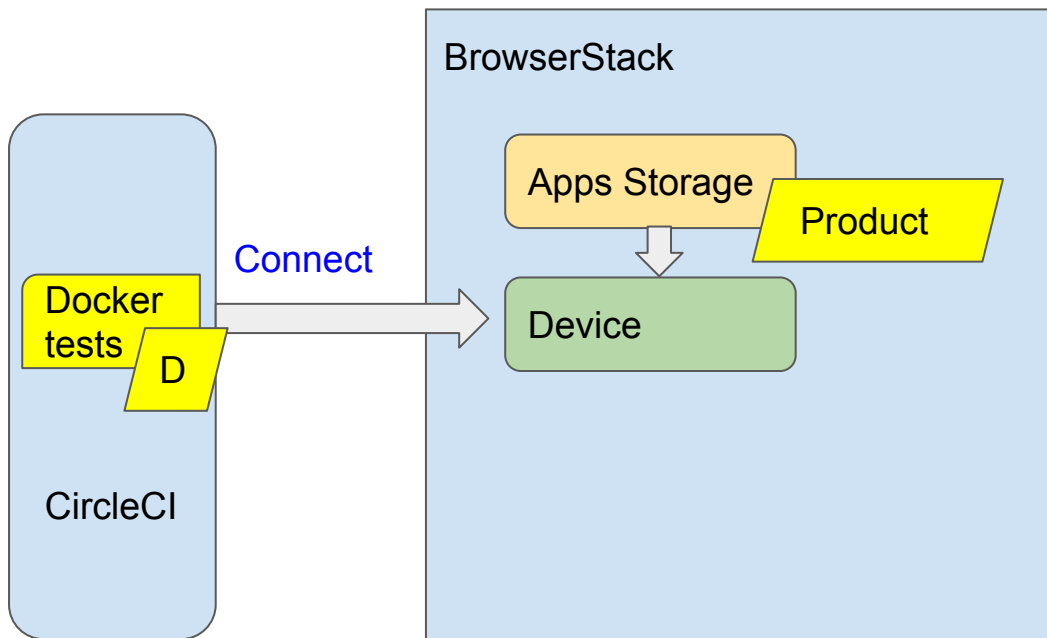
# CI: Загрузка последней версии autoqa



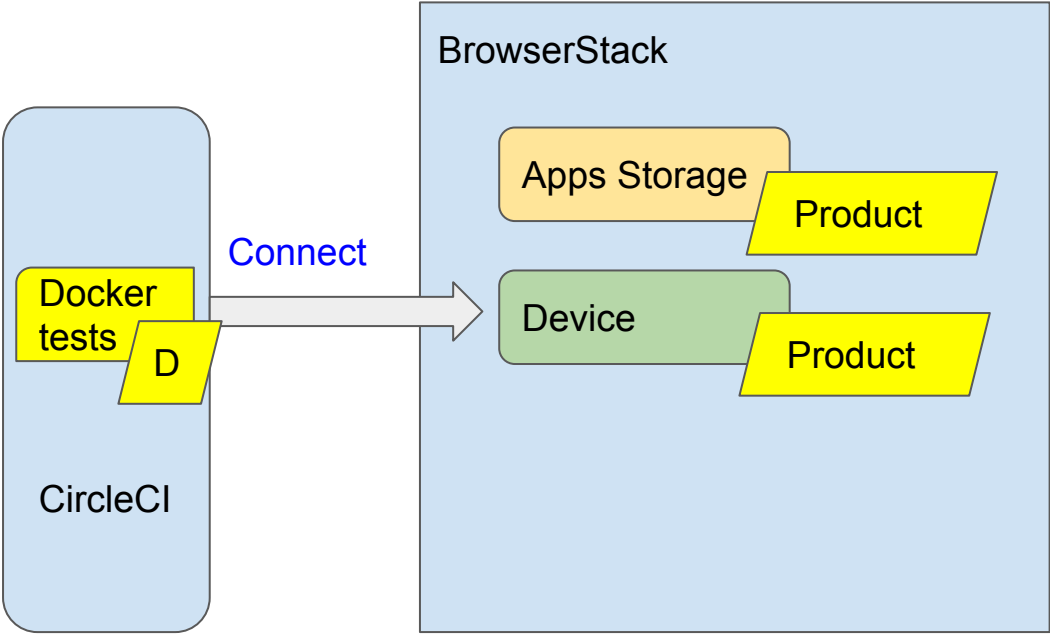
# CI: Тесты autoqa запущены внутри контейнера



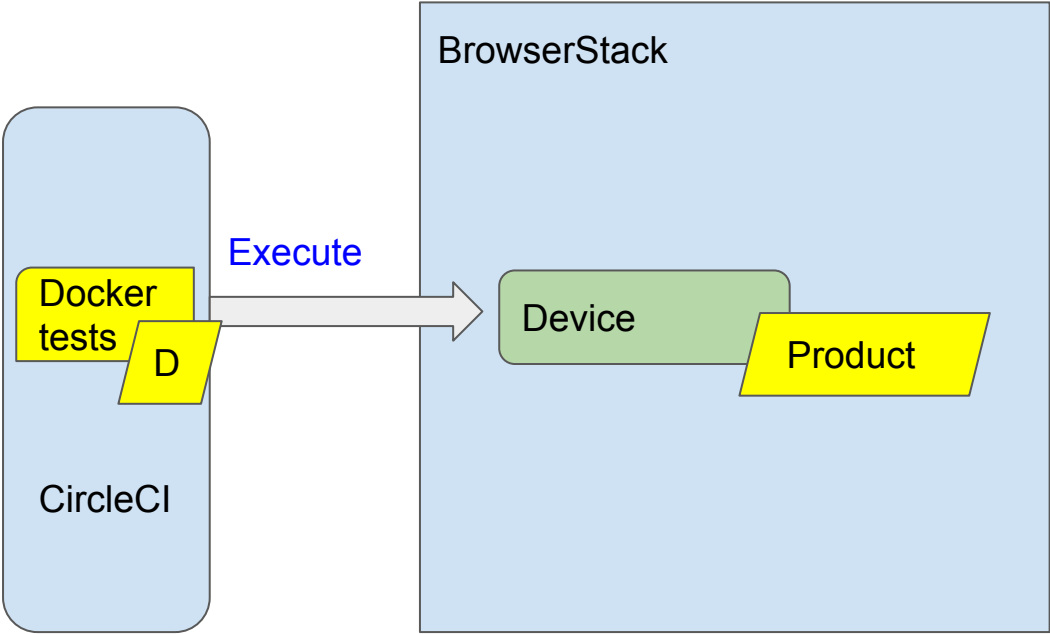
# CI: Подключение к устройству и установка продукта



# CI: Устройство с продуктов готово к использованию



# CI: Выполнение UI тестов



# CircleCI: Просмотр лога autoqa в свойствах Job

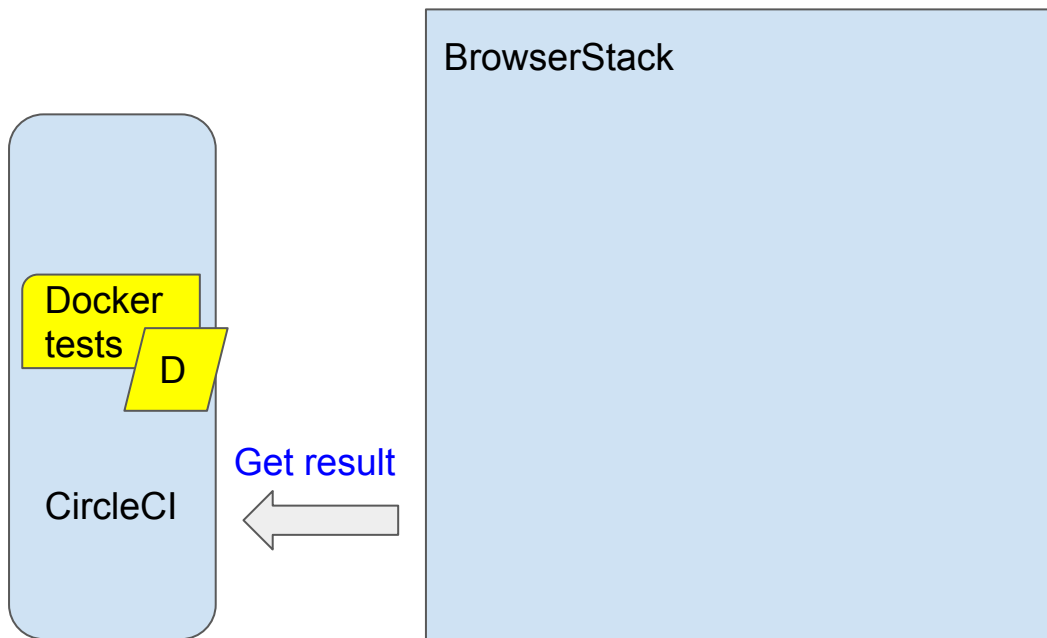
```

  ✓ Run smoke test set 1m 50s
64
65   BeforeMethod
66 [SYSTEM] Test parameters: getAndCompleteOrderByTapsTest, product: merchantApp, env: staging
67 [SYSTEM] startDriver(): BrowserStack
68 [SYSTEM] startDriver(): product: Merchant iPad app, link: feature-ma1120updateMenuEditorUI
69 [SYSTEM] startDriver(): Launch parameters: -skipIntros
70 [SYSTEM] startDriver(): deviceName: iPad 8th, OS version: 14
71 [SYSTEM] startDriver(): BrowserStack Cloud IOSDriver
72 Aug 19, 2021 12:51:52 PM io.appium.java_client.remote.AppiumCommandExecutor$1 lambda$0
73 INFO: Detected dialect: OSS
74 [SYSTEM] startDriver(): completed in 37 sec
75 [TEST] getAndCompleteOrderByTapsTest LogInScreen() | isLoginScreenLoaded():
76 [TEST] getAndCompleteOrderByTapsTest LogInScreen() | setUsername(): tablet
77 [TEST] getAndCompleteOrderByTapsTest LogInScreen() | setPassword():
78 [TEST] getAndCompleteOrderByTapsTest LogInScreen() | tapSignInButton():
79 [TEST] getAndCompleteOrderByTapsTest Header() | isHeaderLoaded():
80 [TEST] getAndCompleteOrderByTapsTest ApiHelper() | waitOrdersStackToBeEmpty()->Up to 300 seconds
81 [+]
82 [TEST] getAndCompleteOrderByTapsTest ApiHelper() | sendOrder()->Init:
83 [TEST] getAndCompleteOrderByTapsTest ApiHelper() | sendOrder()->SendRequest:
84 [TEST] getAndCompleteOrderByTapsTest ApiHelper() | sendOrder()->OrderCreated: 611e53f537e22da0
85 [TEST] getAndCompleteOrderByTapsTest OrderCell() | isOrderCellShown():
86 [TEST] getAndCompleteOrderByTapsTest OrderCell() | tapNextButton():
87 [TEST] getAndCompleteOrderByTapsTest OrderCell() | tapByXY(): x = 169, y = 646
88 [TEST] getAndCompleteOrderByTapsTest AcceptOrderScreen() | isAcceptOrderScreenShown():
89 [TEST] getAndCompleteOrderByTapsTest AcceptOrderScreen() | tapEstimationOptionByIndex(): 4
90 [TEST] getAndCompleteOrderByTapsTest OrderCell() | isOrderCellShown():
91 [TEST] getAndCompleteOrderByTapsTest OrderCell() | tapNextButton():
92 [TEST] getAndCompleteOrderByTapsTest OrderCell() | tapByXY(): x = 510, y = 646
93 [TEST] getAndCompleteOrderByTapsTest OrderCell() | tapNextButton():

```



# CI: Получение результатов тестирования

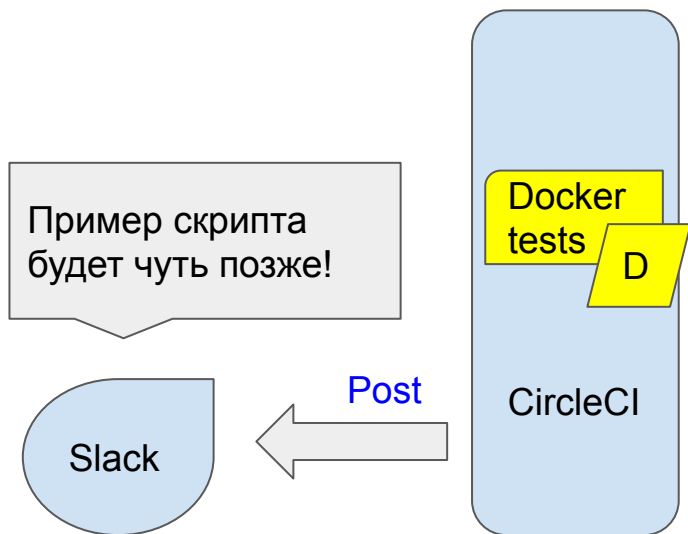


# CircleCI: Сводный отчет о тестировании в Job

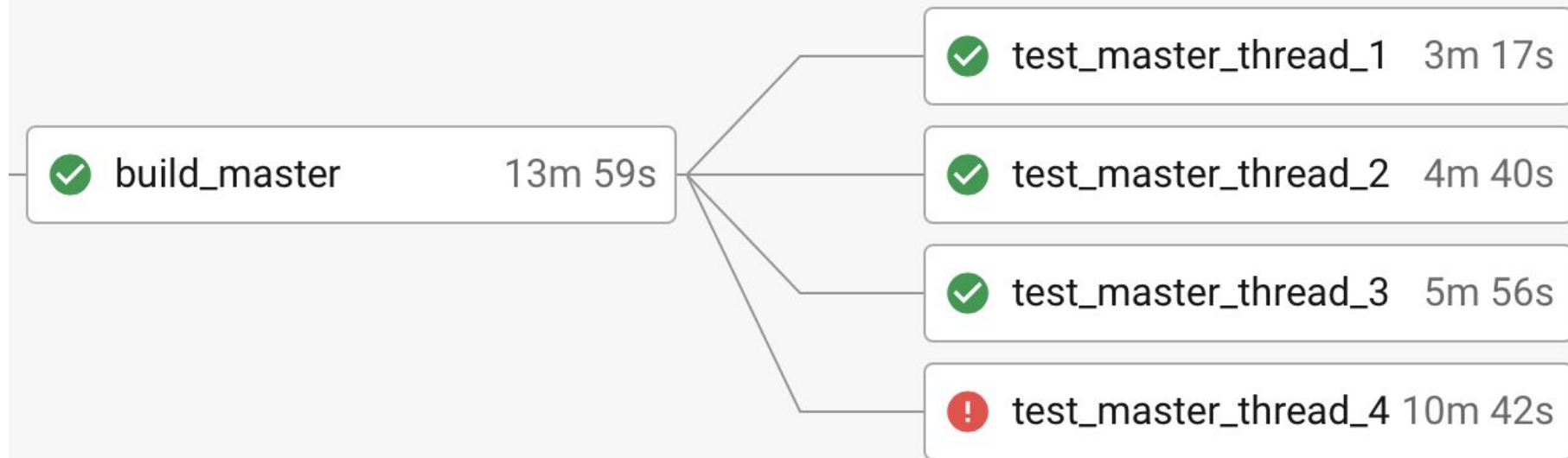
▼  Run full test set, thread 3

```
191
192     Passed tests and execution time:
193         tests.merchantApp.activeOrder.ActiveOrderScreenTest checkCloseRestaurantContentsTest [ ] - 32 sec
194         tests.merchantApp.activeOrder.ActiveOrderScreenTest checkRushModeModalContentsTest [ ] - 31 sec
195         tests.merchantApp.activeOrder.ActiveOrderScreenTest closeAndOpenRestaurantTest [ ] - 31 sec
196         tests.merchantApp.activeOrder.ActiveOrderScreenTest enableAndDisableRushModeTest [ ] - 20 sec
197     Total Passed tests: 4
198     Average test execution time: 28 sec
199     Total test execution time: 114 sec
200
201     Suit execution time: 4 min 4 sec
202
203     [SYSTEM]         Sending message to Slack 'merchant-app-testing' channel
204
205     AfterSuite
206     =====
207     Passed tests: 4
208     Skipped tests: 0
209     Failed tests: 0
210     =====
```

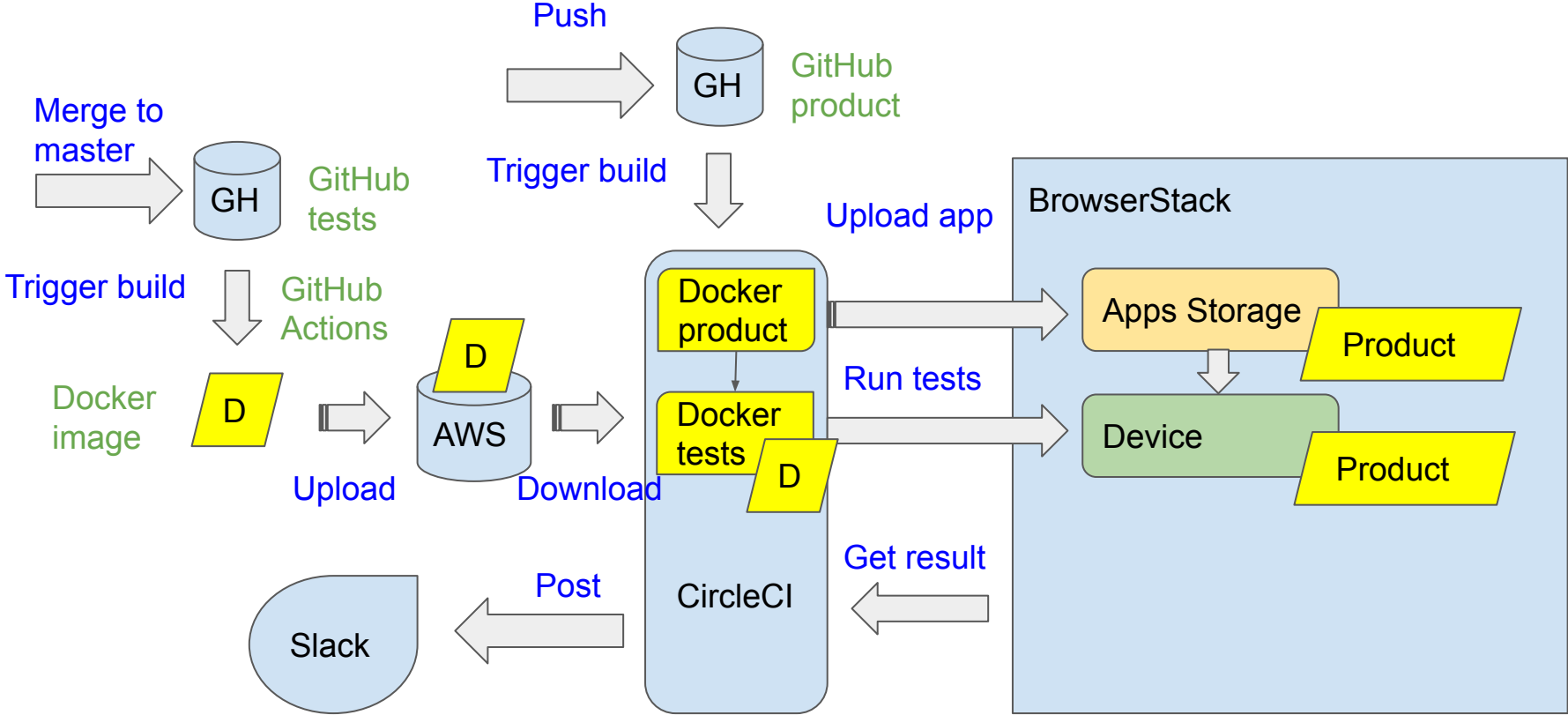
# CI: Отправка сообщения в Slack с результатами



# CircleCI: Сборка и тестирование завершены



# CI: Общая схема



# CI: Погружаемся в детали

Разберем детально, как работает этот план. И как, собственно, его реализовать.

# GitHub Actions

## Имплементация:

- Задание переменных
- Создание Конфигурационного файла
- Сборка на GitHub: раздел Action

Документация: <https://docs.github.com/en/actions>

# GH Actions: Secrets







|                     |
|---------------------|
| Options             |
| Manage access       |
| Security & analysis |
| Branches            |
| Webhooks            |
| Notifications       |
| Integrations        |
| Deploy keys         |
| Autolink references |
| Actions             |
| Secrets             |
| Actions             |
| Dependabot          |

## Actions secrets

[New repository secret](#)

Secrets are environment variables that are **encrypted**. Anyone with **collaborator** access to this repository can use these secrets for Actions.

Secrets are not passed to workflows that are triggered by a pull request from a fork. [Learn more](#).

| Repository secrets  |                   |   |
|---|-------------------|---|
|  CLOUD_USERS               | Updated on 7 Jul  | <a href="#">Update</a> <a href="#">Remove</a> |
|  ECR_AWS_ACCESS_KEY_ID     | Updated on 20 Apr | <a href="#">Update</a> <a href="#">Remove</a> |
|  ECR_AWS_SECRET_ACCESS_KEY | Updated on 20 Apr | <a href="#">Update</a> <a href="#">Remove</a> |
|  JENKINS_CREDENTIALS       | Updated on 20 Apr | <a href="#">Update</a> <a href="#">Remove</a> |
|  SLACK_ACCESS_KEY          | Updated on 23 Jul | <a href="#">Update</a> <a href="#">Remove</a> |
|  USERS                    | Updated on 28 Jul | <a href="#">Update</a> <a href="#">Remove</a> |



# GH Actions: Создание нового секрета

[Actions secrets](#) / New secret

---

Name

MY\_NEW\_SECRET

Value

No one will ever see this string

Add secret

# GH Actions: Конфигурационный файл

**name:** GitHub Actions Demo

**on:** [push]

**jobs:**

Explore-GitHub-Actions:

**runs-on:** ubuntu-latest

**steps:**

- name: List files in the repository

run: |

```
ls ${{ github.workspace }}
```

# GitHub Actions: настройка - импорт данных

```
on:
  push:
    branches:
      - main

env:
  AWS_REGION: eu-west-1
  IMAGE_VERSION: 1.0
  PROJECT: ${ github.event.repository.name }

jobs:
  build:
    name: Build and publish docker image
    runs-on: self-hosted
    steps:
      - name: Checkout code
        uses: actions/checkout@v2

      - name: Create file users.json
        run: |
          touch users.json

      - name: Fill users.json
        run: |
          echo ${ secrets.USERS } | base64 -d > users.json
```

# GitHub Actions: настройка - сборка образа

```
- name: Configure AWS credentials
uses: aws-actions/configure-aws-credentials@v1
with:
  aws-access-key-id: ${ secrets.ECR_AWS_ACCESS_KEY_ID }
  aws-secret-access-key: ${ secrets.ECR_AWS_SECRET_ACCESS_KEY }
  aws-region: ${ env.AWS_REGION }

- name: Login to Amazon ECR
id: ecr-login
uses: aws-actions/amazon-ecr-login@v1

- name: Build Docker image
env:
  ECR_REGISTRY: ${ steps.ecr-login.outputs.registry }
run: docker build -t $ECR_REGISTRY/$PROJECT:$IMAGE_VERSION .

- name: Push Docker image
env:
  ECR_REGISTRY: ${ steps.ecr-login.outputs.registry }
run: docker push $ECR_REGISTRY/$PROJECT:$IMAGE_VERSION

- name: Logout from Amazon ECR
if: always()
run: docker logout ${ steps.ecr-login.outputs.registry }
```

# Загрузка приложения в облако

В облаке у нас уже есть тесты, но еще нет продукта на котором их запускать.

# Загрузка файла приложения в облако

```
curl -u "USERNAME:ACCESSKEY"  
  
-X POST "https://api-cloud.browserstack.com/app-automate/upload"  
  
-F "file=@/Users/MyUser/Documents/apps/product.apk"  
  
-F 'data={"custom_id": "my_awesome_android_app"}'
```

**custom\_id** - это уникальный идентификатор приложения в облачном хранилище. Так вы можете хранить сколь угодно много разных версий вашего приложения одновременно.

# Fastlane

Fastlane - это инструмент для автоматизации процессов сборки и выкладки приложений

**Сайт:** <https://fastlane.tools>

```
platform :ios do

  desc "Builds, achieves and uploads ipa to Test server"

  lane :upload_lane do

    # Download repo

    # Build release app

    # Upload to Test server

  end

end
```

# Использования Fastlane для загрузки файла

```
desc "Upload build to BrowserStack"

lane :uploadToBSfromCI do |options|

  ipa_path = "/Users/distiller/project/output/gym/myProduct.ipa"

  browserstack_app_name = options[:cloudLink]

  curl = "curl -u \"\" + browserstack_login + "\" -X POST \"\" + browserstack_url + "\" -F \"file=@\" + ipa_path + "\" -F
  \data={\"custom_id\": \"\" + browserstack_app_name + "\"}\"\"

  UI.message(curl)

  app_upload_response = sh(curl)

  UI.message(app_upload_response)

end
```



# Запуск тестов на загруженном приложении

Теперь у нас есть команда для загрузки приложения.

Расширим пайплайн, собирающий наше приложение на CI.

# CircleCI

Для чего мы его используем:

- Имеет готовые образы с разными версиями XCode
- Имеет встроенные функции для сборки iOS приложения и работы с сертификатами

# CircleCI

- Расширим этап сборки приложения, добавив его загрузку в BrowserStack
  
- Добавим новый job с исполнением автотеста:
  - Загрузка Docker - образа из Amazon
  - Запуск теста в образе

# CircleCI: добавление этапа загрузки в облако

```
# Building the app
build_master:
  macos:
    xcode: 12.5.0
  environment:
    FL_OUTPUT_DIR: output
  shell: /bin/bash --login -o pipefail
  steps:
    - checkout
    - fix-xcode-checkouts
    - run:
      name: Install components
      command: bundle install
    - run:
      name: Build ipa file, upload to AppCenter and BS:MerchantAppMaster
      command: |
        bundle exec fastlane buildAndUpload
        bundle exec fastlane uploadToBSfromCI cloudLink:MerchantAppMaster
```

# CircleCI: прогон тестов на загруженном приложении

```
test_master:  
  docker:  
    - image: eu-west-1.amazonaws.com/autoqa:1  
  steps:  
    - run:  
      name: Run full test set  
      command: mvn -DsuiteName=suites/App/full.xml test
```

## CircleCI: добавление тестов в pipeline

```
workflows:  
# When merging to master branch  
  master_flow:  
    when:  
      and:  
        - equal: [ master, << pipeline.git.branch >> ]  
    jobs:  
      - swiftlint  
      - build_master:  
        requires:  
          - swiftlint  
      - test_master:  
        requires:  
          - build_master
```

# CircleCI: Нельзя просто так взять и запустить...

```
test_master:  
  docker:  
    - image: eu-west-1.amazonaws.com/autoqa:1  
  steps:  
    - run:  
      name: Run full test set  
      command: mvn -DsuiteName=suites/App/full.xml test
```

- CircleCI всегда запускает код в корне системы
- ... а тесты лежат в подкаталоге

## CircleCI: Прогон тестов - исправление =)

```
test_master:  
  docker:  
    - image: eu-west-1.amazonaws.com/autoqa:1  
  steps:  
    - run:  
      name: Run full test set  
      command: cd /usr/tests && mvn -DsuiteName=suites/App/full.xml test
```



# CircleCI: Список выполненных job в пайплайне

## All Pipelines



Everyone's Pipelines



All Projects



All Branches



Pipeline

Status

Workflow

Branch / Commit

wolt-ios-  
merchant 3545

Success

master\_flow



master

a0b606d Merge pull request #824 from  
creditornot/release/2.14.1

Jobs



swiftlint 7700



build\_master 7701



test\_master 7705

# CircleCI: Информация о пайплайне

|                     |        |                 |        |                           |
|---------------------|--------|-----------------|--------|---------------------------|
| Duration / Finished | Queued | Executor        | Branch | Commit                    |
| 7m 28s / 5h ago     | 0s     | Docker Medium ⓘ | master | 76250dd, cecaf5b, a0b606d |

Author & Message

Merge pull request #824 from creditornot/release/2.14.1

STEPS TESTS ARTIFACTS ●

▶ Spin up environment

▶ Preparing environment variables

▼ Run full test set

```
270     tests.merchantApp.pastOrders.PastOrdersTest checkPastOrdersScreenTest [ ] - 30 sec
271     tests.merchantApp.whatsNew.WhatsNewTest shouldDisplayWhatsNewPopUpAtSignIn [ ] - 14 sec
272     Total Passed tests: 5
273     Average test execution time: 42 sec
274     Total test execution time: 211 sec
275
276     Suit execution time: 6 min 31 sec
277
```

# Отправка сообщений в Slack

Что полезно добавить в сообщение:

- Статус (Все прошло или есть ошибки)
- Сведения о прогоне: ветка GitHub, окружение, suite
- Ссылка на пайплайн CI
- Ссылка на прогон на BrowserStack

# Пример сообщения в Slack с результатами теста



**autoqa\_notifications** APP 2:07 PM

**autoqa for Merchant iPad app: PASSED** ✓

**Suite:**

Merchant iPad app full set - Thread 3

**Branch:**


MerchantAppMaster

**Device:**

iPad 8th, OS version: 14

**CI pipeline:**

 Job

 Tests passed: 4 / 4

 Suite execution time: 4 min 24 sec

 browserstack

# Slack: Порядок настройки

- Создание Slack-бота для отправки сообщения и настройка его прав доступа
- Создание канала (channel)
- Имплементация кода, который будет формировать сообщение и отправлять его в соответствующий канал в определенной форме (JSON)

## Slack - полезные ссылки

- BrowserStack API: получение информации и прогонов и отдельных сессиях:

<https://www.browserstack.com/docs/app-automate/api-reference/appium/sessions#get-session-details>

- Slack API: составления и отправка сообщений:

<https://api.slack.com/authentication/basics>

<https://api.slack.com/messaging/sending>

- Конструктор сообщений Slack:

<https://app.slack.com/block-kit-builder/>

# Slack: пример кода для отправки сообщения

```
fun sendMessage(channel: SlackChannels, message: String, isMarkDown: Boolean = true) {
    syslog( text: "Sending message to Slack '${channel.description}' channel")

    val payload = JSONObject()
    payload["channel"] = channel.value
    if (isMarkDown) {
        payload["text"] = "autoga notification"
        payload["blocks"] = message
    } else {
        payload["text"] = message
    }

    val request = Request.Builder()
        .url(OutsourcedURLs.SLACK_POST_MESSAGE.url)
        .header( name: "Authorization", value: "Bearer $token")
        .addHeader( name: "content-type", value: "application/json")
        .post(payload.toString().toRequestBody())
        .build()

    client.newCall(request).execute().use { response ->
        val responseJson: JsonNode = ObjectMapper().readTree(response.body?.string()?.trim())
        Assert.assertTrue(
            responseJson.get("ok").asBoolean(),
            "${TAG}sendMessage(): Failed sending message to channel '${channel.name}'"
        )
    }
}
```

# Slack: пример кода для отправки сообщения

Используемые инструменты:

- **JSON**: формирование тела запроса
- **okhttp**: создание, отправка и обработка HTTP запросов
- **Jackson**: мощный инструмент работы с JSON объектами
- + **enum** класс `SlackChannel`: для удобного хранения констант



# CI: итоговый порядок исполнения

## Проект автотестов:

- Сборка образа тестов с GitHub Actions при merge to master
- Публикация образа на сетевое хранилище (Amazon)

# CI: итоговый порядок исполнения

## Проект приложения:

- Сборка приложения
- Загрузка дистрибутива в BrowserStack
- Скачивание последнего образа с тестами
- Запуск тестов внутри контейнере на загруженном приложении в облаке
- Публикация результатов

# Список использованной литературы

Idea: <https://www.jetbrains.com/idea/>

Maven: <https://maven.apache.org>

Appium: <https://appium.io>

TestNG: <https://testng.org/doc/index.html>

Docker: <https://www.docker.com>

Fastlane: <https://fastlane.tools>

GitHub Action: <https://github.com/features/actions>

CircleCI: <https://circleci.com>

Browserstack: <https://www.browserstack.com>

# Репозиторий с примером проекта

Тестовый проект с реализованной структурой Page Factory, использующий управление тестами TestNG и умеющий отправлять сообщения в Slack.

**GitHub:** <https://github.com/heavy-razzer/heisenbug-autoqa-demo>

# autoqa: Mobile UI test automation framework

Время удивительных вопросов и неожиданных ответов. Или наоборот.