Testing of Android apps

Tomáš Flek

Quality Assurance Engineer

Overview

1. What is testing
2. Role of QA in development process
3. Little bit of theory
4. JUnit
5. Watching health of app
6. Bug reporting
7. Automated testing
8. Espresso + UiAutomator
9. Continuous integration

avast

What is testing?

What is testing?

- Process of
  - Reviewing requirements
  - Finding defects
  - Gaining confidence about the level of quality
  - Preventing defects
  - …

avast

Why to test?

- People make mistakes
  - Product manager
  - Developer
  - QA engineer / Tester
  - 3rd party

- Human error can lead to defects/bugs inside app
  - Crash or worse

avast

Why to test?

- People make mistakes
  - Product manager
  - Developer
  - QA engineer / Tester
  - 3rd party

- Human error can lead to defects/bugs inside app
  - Crash or worse

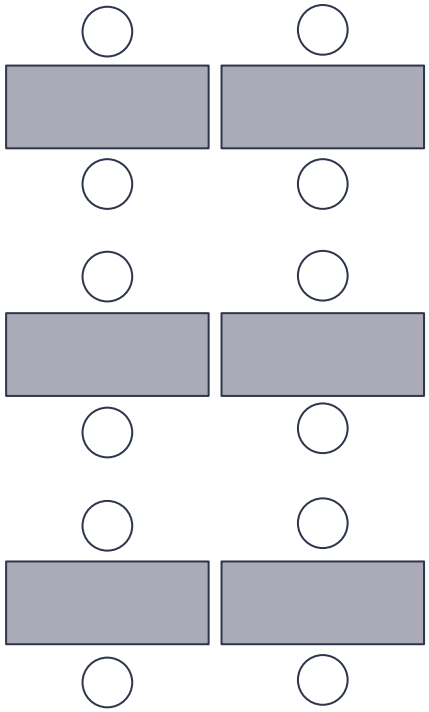
- 84% if users uninstall app after just **2 crashes!**

  techcrunch.com

avast

Role of QA in development process

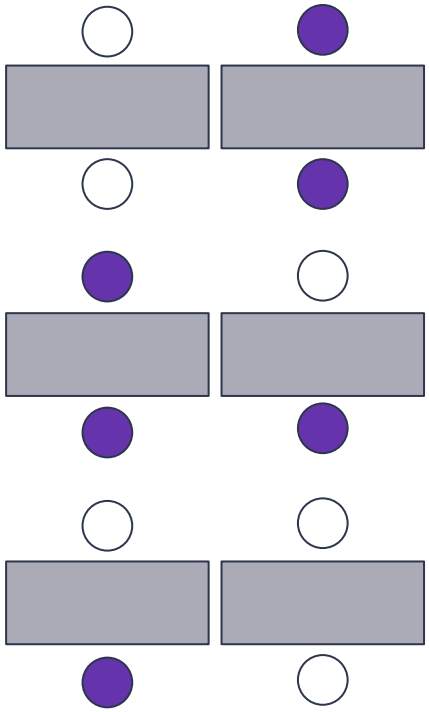
Team I work in at Avast



- ? Developer
- ? Product Manager
- ? Project Manager
- ? Quality Assurance Engineer
- ? Tester

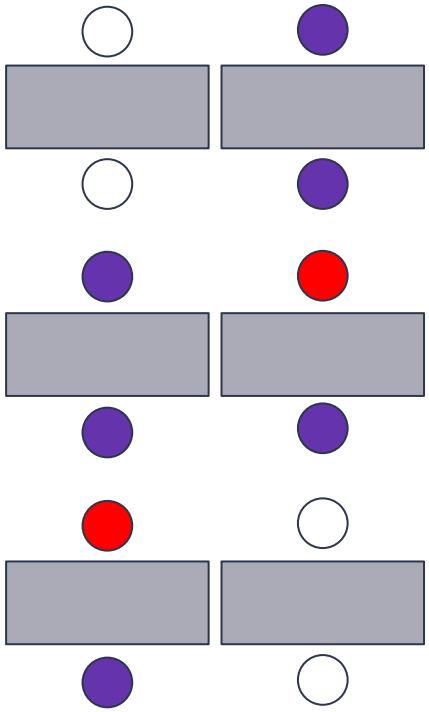
avast

Team I work in at Avast



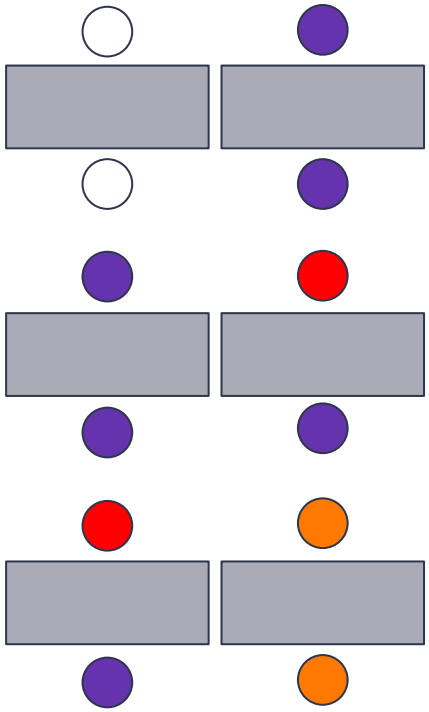
- **6 Developers**
- ? Product Manager
- ? Project Manager
- ? Quality Assurance Engineer
- ? Tester

Team I work in at Avast



- **6 Developers**
- **1 Product Manager**
- **1 Project Manager**
- ? Quality Assurance Engineer
- ? Tester

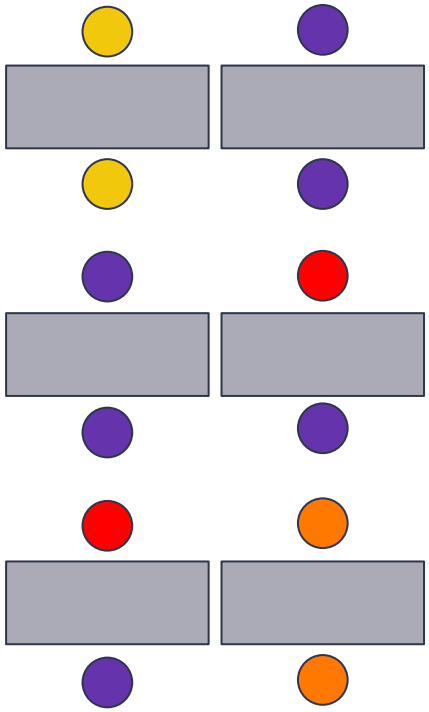
Team I work in at Avast



- 6 **Developers**
- 1 **Product Manager**
- 1 **Project Manager**
- 2 **Quality Assurance Engineers**
- ? Tester

avast

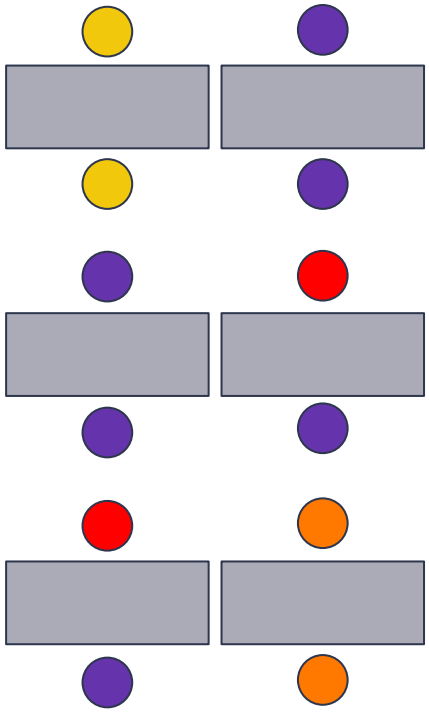
Team I work in at Avast



- 6 Developers
- 1 Product Manager
- 1 Project Manager
- 2 Quality Assurance Engineers
- 2 Testers

avast

Team I work in at Avast



- 6 Developers
- 1 Product Manager
- 1 Project Manager
- 2 Quality Assurance Engineers
- 2 Testers

- Develop app with millions of active users

What is the difference?

Tester

- Performs manual testing based on prepared scenarios
- Reports found bugs
- Verifies fixed bugs

Quality assurance engineer

- … everything what tester does
- Reviews requirements with PM and Developers
- Translates between technical and product manager’s language
- Organizes testing plans and approaches
- Performs risk analysis
- Develops automated tests
- Responsible for release preparation
- Post-release monitoring

avast

When to involve QA?

- At the very **beginning** when forming requirements

- QA/Testers, developers and PMs have to **talk to each other**
  - **“Testing is about questions and communication…“**

- How to achieve that?
  - Meet at regular basis and discuss new features/tasks
    - Does it make sense for user?
    - How it should behave in corner cases?
    - How much development effort it will take?
    - How it will be tested? Is it even testable?

avast

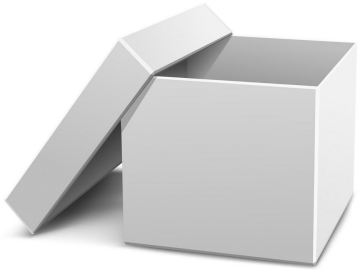
If you are on your own

- Do risk analysis and test important things first
- Ask friends to break your app
  - do not just test it on your own
- Carefully watch user's feedback
  - Beta groups, stage rollout
- Automate repetitive tasks

Little bit of theory

White box testing

- We know the code inside
- Focus on internal structures
- Is written code good enough?



Black box testing

- We do not know/care about the code inside
- Focus on program specification
- Is the program’s behavior correct for given input data?

Static testing

- Testing when program is not running
  - Static analysis of code
    - **FindBugs** (http://findbugs.sourceforge.net/)
    - **Lint** (https://developer.android.com/studio/write/lint.html)
    - **CheckStyle** (http://checkstyle.sourceforge.net/)
    - **Ktlint** (https://ktlint.github.io/)
    - **Detekt** (https://arturbosch.github.io/detekt/)
  - Code review

Dynamic testing

- Testing when program is running
  - Manual testing according to test cases
  - Automated test

avast

How many issues can you see?

```java
package com.avast.android.appfortests;

import android.util.Log;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

/**
 * Run FindBugs inspection to see what can be revealed by static analysis.
 *
 * @author Tomas Flek (flek@avast.com)
 */

public class ClassWithBugs {

    public void someMethod() throws SQLException {

        String dollars = "$";

        for (int i = 0; i < 100; i++) {
            dollars += "$$";
        }

        Connection con = DriverManager.getConnection("jdbc:mysql://database.name.com:1111",
                "root", "secretPassword");

        switch (dollars) {
            case "$":
                Log.i("test", "I have one dollar");
            case "$$":
                Log.i("test", "I have two dollars");
                break;
            case "$$$$$":
                Log.i("test", "I am rich!");
        }
    }
}
```

Demo

Types of test approaches

- Regression test
- Explorative test
- Smoke test
- Monkey test
- Acceptance test
- Integration test
- Penetration test
- Usability test
- A/B test
- ...

avast

Types of test approaches

- **Regression test**
- **Explorative test**
- **Smoke test**
- **Monkey test**
- Acceptance test
- Integration test
- Penetration test
- Usability test
- A/B test
- ...

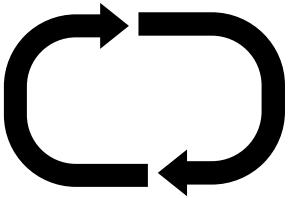
avast

Regression test

- Have I broken something that already worked?

- Usually requires lots of test cases
  - Test case:
    - Unique identifier
    - Environment
    - Input data
    - Steps with expected results

- Good candidate for automation



Explorative test

- Tester does not follow any steps (but records them)
- Usually finds lots of bugs with low time effort
- Requires experienced testers/QA engineers
- Should be performed on various environments
  - Android versions
  - Manufacturers with their modifications
  - Tablet layouts → rotations
- Cannot automate

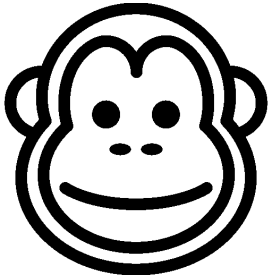
avast

Smoke test (Sanity check)

- Last check if nothing is broken miserably
  - Usually performed on already tested release candidate
- First check if build “worths testing effort”
  - Is it possible and reasonable to proceed with further testing?
- Quick test → several minutes
- Can be automated

Monkey test

- Generator of random input events
  - Taps
  - Scrolls
  - Swipes
  - ...



- Really easy to set up and run → finds nasty crashes

- Really easy to automate, just use ADB
  - https://developer.android.com/studio/test/monkey.html

```
adb shell monkey [options] <event-count>
```

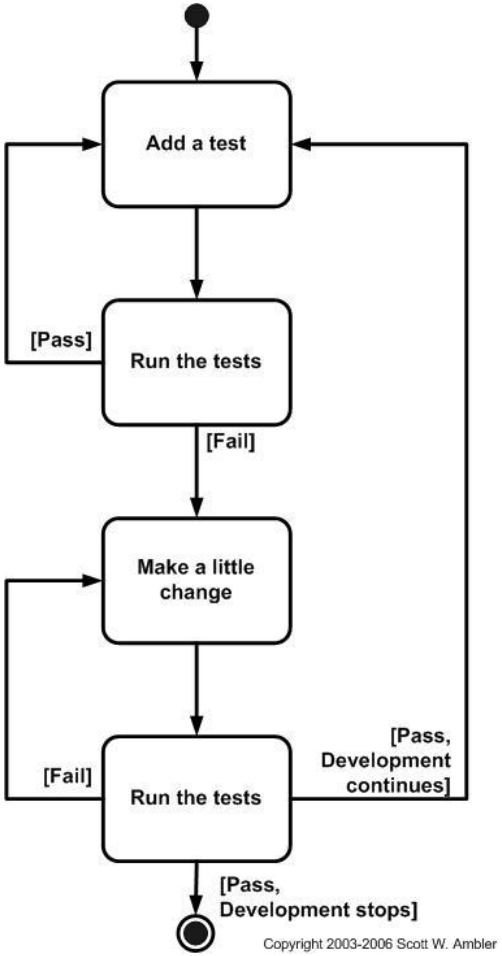
Demo

Test driven development

- Requirements are converted to small test cases

- Basic principle:
  - Write test
  - Run test to see where it failed
  - Write code
  - Run all tests
  - Refactor code
  - Repeat…

- What to use for such test? JUnit!



JUnit framework

Unit tests in general

- Unit == small and isolated piece of code
- Compares result of method call with expected result
- Lives next to the application code
  - Usually different package
  - Usually created by developers
- Triggered on demand
  - Even after each code change
  - Should be part of build process

avast

JUnit

- Framework for Java (http://junit.org)

- Based on method annotations
  - `@BeforeClass`
  - `@Before`
  - `@Test`
  - `@After`
  - `@AfterClass`

- Do not rely on tests order (usually alphabetic)
  - Each test should be independent unit anyway!
  - Order can be specified by special annotation

Test method structure

- Test methods must
  - be `public`
  - return `void`
  - have `@Test` annotation
  - contain at least one `assert`

- Assert variants
  - `assertTrue(), assertFalse()`
  - `assertEquals(), assertNotEquals()`
  - `assertSame(), assertNotSame()`
  - `assertNull(), assertNotNull()`
  - `assertArrayEquals()`
  - `assertThat(org.hamcrest.Matcher)`
  - `fail()`

avast

Hamcrest matchers

- `allOf` - matches if all matchers match
- `anyOf` - matches if any matchers match
- `not` - matches if the wrapped matcher doesn't match and vice
- `equalTo` - test object equality using the equals method
- `is` - decorator for `equalTo` to improve readability
- `hasToString` - test Object.toString
- `hasSize` - test size of collection
- `contains` - test elements of collection
- `containsInAnyOrder` - test elements of collection in any order
- `everyItem` - test if all items matches
- `hasProperty` - test if object has property with given name
- ...

(http://hamcrest.org/JavaHamcrest/javadoc/1.3/org/hamcrest/Matchers.html)

Demo

Watching health of app

What to watch

- Users feedback
  - Ratings
  - Comments
  - Uninstall rate

- Crash reporting
  - Amount of crashes
  - ANRs

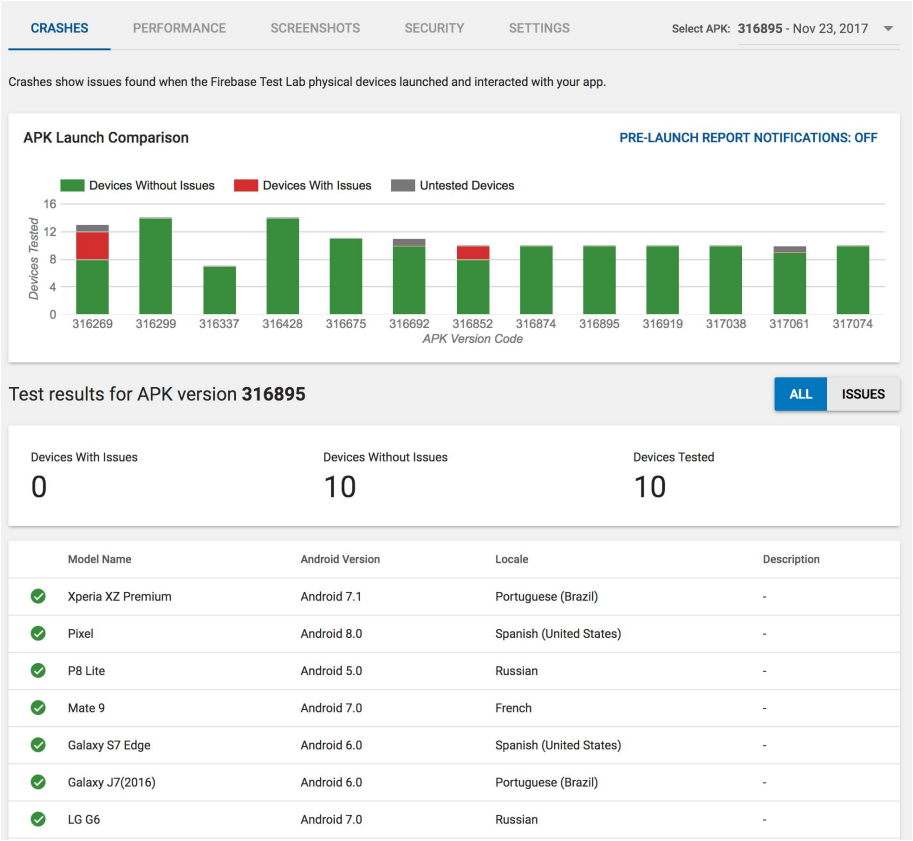
avast

Where to watch

- Google Play Developer Console
- Firebase
- Crashlytics (Fabric.io)
  - Currently part of Firebase

GP Developer Console

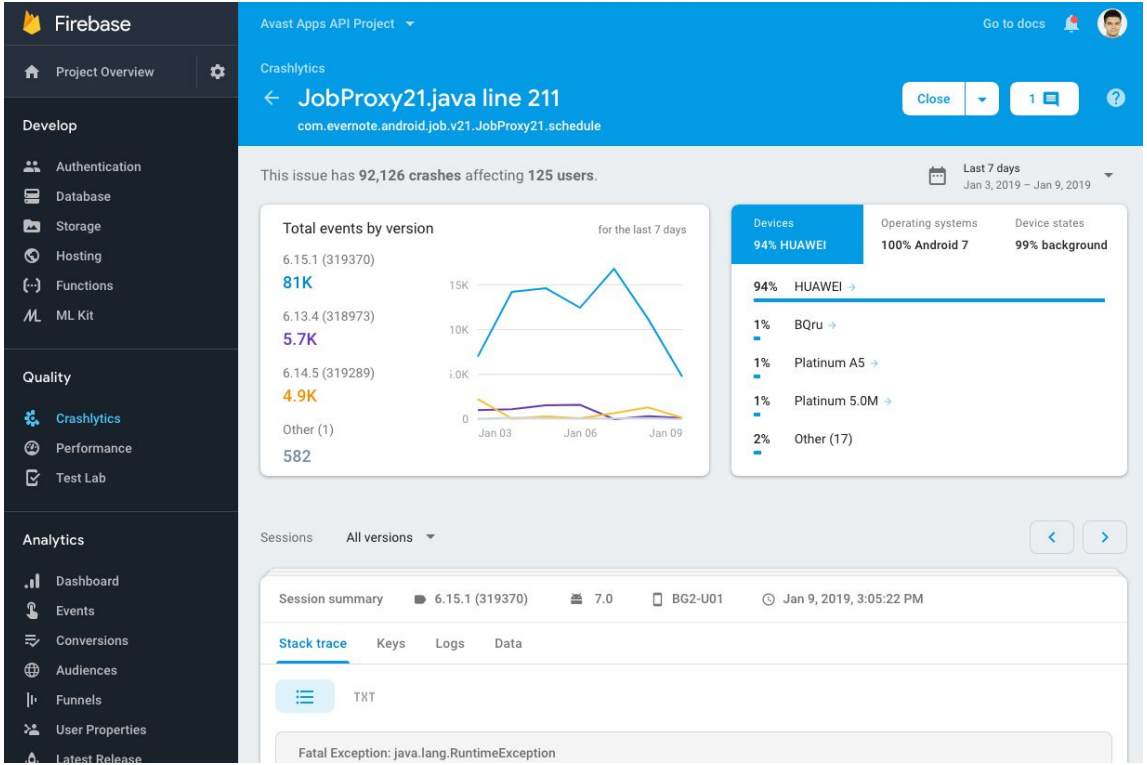
- Statistics
  - Install and uninstall rates
- Vitals
  - Crashes
  - ANRs
- Users feedback
  - Ratings
  - Reviews
  - Beta group feedback
- Release management
  - Pre-launch report
  - Alpha group
  - Beta group
  - Stage rollout (e.g. 1% of user base)



avast

Firebase (Crashlytics)

- Detailed crash reporting
  - Crash-free users
  - Crash-free sessions
  - Affected users
- Crashes analysed and sorted
  - Affected devices
  - Affected OS versions
  - Is it known Android bug?
  - Part of logcat before crash
- Deobfuscation mappings uploaded automatically



Bug reporting


Not this time Junior!
Today is release day.
BUG, BUG
There is a BUG!
Project Manager
Junior Tester

Keep bugs organized

- Have a place where to collect bug reports
- Prioritize
- Reject poorly described ones
- Have a workflow
  - Open
  - In progress
  - Waiting for code review
  - Waiting for QA review
  - Closed

avast

What should good bug report contain?

- Brief summary
- Priority
- Severity
- Affected version
- Fix version
- Environment
- Steps to reproduce
- Logs, screenshots, video
- Reference to related issues
- ...

avast

Tools (bug trackers)

- Bugzilla (https://www.bugzilla.org/)

- JIRA (https://www.atlassian.com/software/jira)

Mobile Quality Assurance / MQ-592

Application crashes on launch on tablets

Edit | Comment | Assign | More | Finish | Start Progress | Export

Details

Type:	Bug	Status:	OPEN (View Workflow)
Priority:	P2		
Affects Version/s:	None	Resolution:	Unresolved
Component/s:	None	Fix Version/s:	None
Labels:	None		

Description

When application is launched on tablets, it crashes immediately.

Environment:

- Samsung Galaxy Tab S2 (7.0.0)
- Lenovo Yoga Tab (6.0.1)

Steps to reproduce:

- install application
- launch it -> crash

Attachments

Drop files to attach, or browse.

log.txt
Just now 0.0 kB

People

Assignee:
Unassigned
Assign to me

Reporter:
Flek Tomáš

Votes:
0

Watchers:
0 Start watching this issue

Dates

Created:
4 minutes ago

Updated:
1 minute ago

Agile

View on Board

Automated testing

Before you start

- What to automate?
  - Parts of app that does not change often
  - Regression tests
  - Smoke tests

- What not to automate?
  - Parts of app that changes with each release
  - Localizations

- Great complement to manual testing, not replacement!
  - “What you do more than twice, make a script”

avast

Tools & frameworks for automation

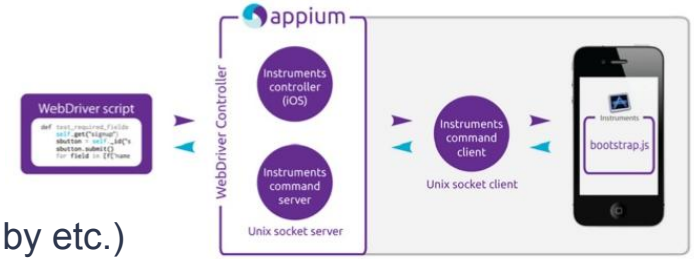
- **UIAutomator (Kotlin, Java)**
  - Android native apps
  - Testing outside of application
    - allows test many system stuff, access to any application, etc.
  - JUnit test-cases with special privileges
  - API level 16 or higher
  - WebView not supported

- **Espresso (Kotlin, Java)**
  - Android native apps
  - Testing inside of application
  - API level 8, 10, 15 or higher
  - WebView supported by Espresso Web

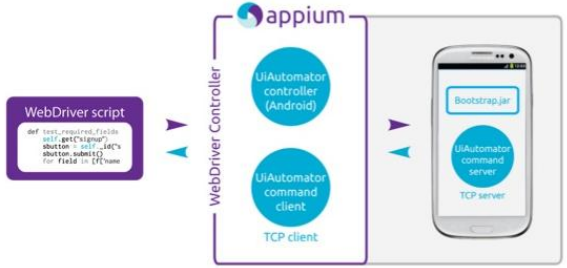
Tools & frameworks for automation

- **Appium**
  - Android/iOS/Windows native, hybrid and web apps
  - Any kind of programming language (Python, Java, Ruby etc.)
  - Client ↔ server architecture
    - Communication via `JSONWireProtocol`
  - WebView is not a problem
  - Using the WebDriver protocol
    - API 9 - 17 are supported via Appium's Selendroid Driver
    - API 18 and up are supported via Appium's UiAutomator
  - Great for apps that have same UI on both platforms

IOS



ANDROID



Find “Sign in” button and click it...

mDevice.findObject(By.res("com.twitter.android:id/sign_in")).click()

onView(withId(R.id.sign_in_button)).perform(click())

driver.find_element_by_id('com.twitter.android:id/sign_in').click()

Espresso + UiAutomator

Espresso

- Test can be performed only within the app
- Test has access to app resources (strings and IDs)
- Built-in synchronization of test actions and app UI → no sleeps needed!
  - Animations need to be turned off on test device
- Test syntax

```
onView(ViewMatcher)
    .perform(ViewAction)
    .check(ViewAssertion)
```

- Use Hamcrest matchers to find and assert more complex elements

https://developer.android.com/training/testing/ui-testing/espresso-testing

avast

UiAutomator

- Test multiple apps, interact with system UI

- Uses accessibility for UI interaction, no synchronization → sometimes sleep is needed

- Test syntax

```
UiObject2 obj = device.wait(Until.findObject(BySelector), timeout)
assertNotNull(obj)
assertEquals("Hello", obj.getText())
obj.click()
```

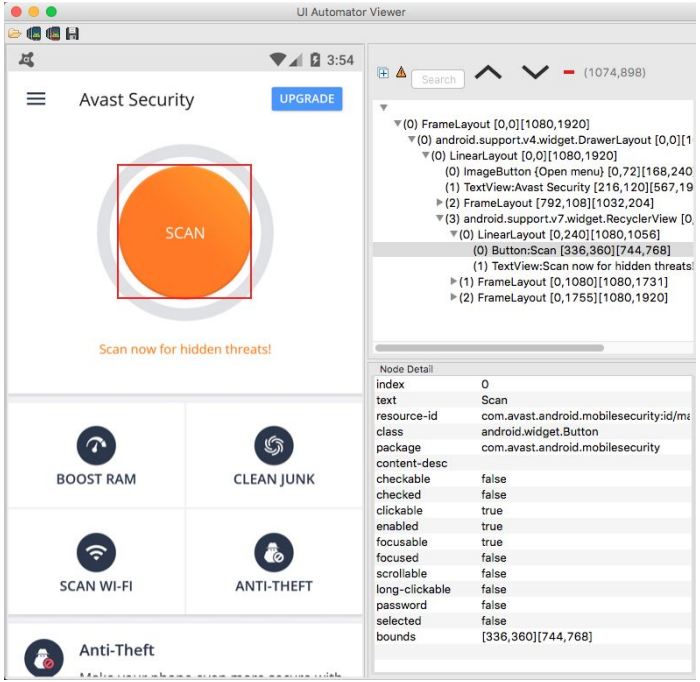
- Tests can be placed in separate module
  - You can force stop/clear data of tested app during test
  - You can test updates of your app from one version to another

https://developer.android.com/training/testing/ui-testing/uiautomator-testing

avast

UiAutomator Viewer

- GUI tool to analyze UI components
  - `<android-sdk>/tools/uiautomatorviewer`
- Creates XML dump of visible UI elements



Best practice

- Important elements should have unique resource IDs
- Avoid programmatically capitalized text
  - API 23- vs API 24+
- Avoid toaster messages for important stuff
  - It is possible to test them only in Espresso

avast

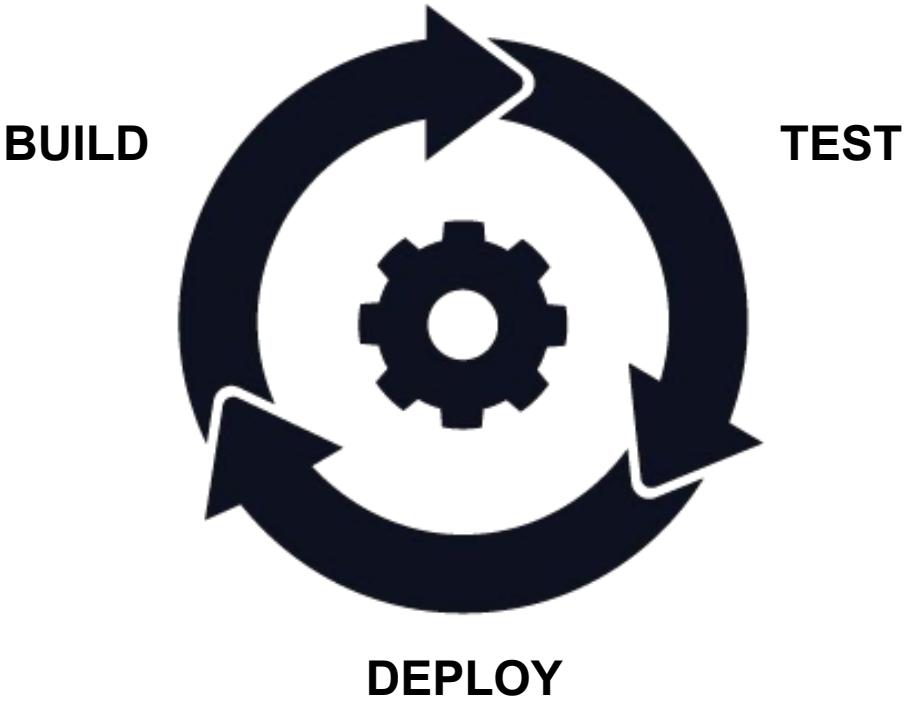
Best practice

- Important elements should have unique resource IDs
- Avoid programmatically capitalized text
  - API 23- vs API 24+
- Avoid toaster messages for important stuff
  - It is possible to test them only in Espresso
- **Now, let's create new test!**

avast

Demo

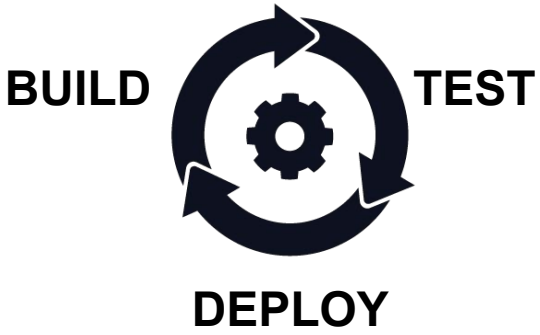
Continuous integration



avast

Why CI

- It takes long time to run tests locally
  - Developers will not do that
- Prevent problems that someone forgot run all tests
- Automatically
  - Build with new commit in VCS
  - Check code before merge to “master” branch
  - Run tests on remote servers
  - Run tests in parallel
  - Deploy tested build



CI Tools

- **Jenkins**
  - Open-source
  - Integration with a wide variety of tools and technologies
  - Community plugins
  - Community support
  - Harder to learn to use it

- **TeamCity**
  - Free with 3 build agents and up to 100 configurations, paid upgrade
  - Integration with a wide variety of tools and technologies
  - Paid support
  - Easy to use

Back to Project
Status
Changes
Console Output
View as plain text
View Build Information
Git Build Data
Previous Build

Console Output

Skipping 1.196 KB.. Full Log

```
KoEttlJQNjBwdnF2sVayAwD4vuoDpQAAAA==[0mskipping: [teamwiesn.com]

TASK [smtp-server : include] ***************************************************
skipping: [teamwiesn.com]

TASK [ssl : include] ************************************************************
included: /var/lib/jenkins/jobs/Rollout-system/workspace/roles/ssl/tasks/setup.yml for
teamwiesn.com

TASK [ssl : install ssl packages] ***********************************************
ok: [teamwiesn.com] => (item=[u'ssl-cert', u'sslscan'])

TASK [ssl : Install ssl certificates] *******************************************
ok: [teamwiesn.com] => (item=teamwiesn.crt)

TASK [ssl : Install ssl keys] ***************************************************
ok: [teamwiesn.com] => (item=teamwiesn.key)

TASK [ssl : Generate dhparam key] ***********************************************
ok: [teamwiesn.com]

TASK [ssl : Check ssl forward secrecy key permission] ***************************
ok: [teamwiesn.com]

TASK [ssl : include] ************************************************************
skipping: [teamwiesn.com]

TASK [pip : Check to see if pip is already installed.] **************************
ok: [teamwiesn.com]
```

IntegrationBuild (FreeBSD)			Pending (3)	1 queued		Run ...
#8059	Swabra	No artifacts	Changes (7)	2h:34m left		Stop
#8058	Tests failed: 2 (1 new), passed: 13658, ignored: 133, muted: 5	No artifacts	Dmitry Neverov (1)	2 minutes ago (2h:32m)		
IntegrationBuild (HSQLDB Incremental)			Pending (4)	1 queued		Run ...
#11787	Tests passed: 2205, ignored: 11, muted: 2; Running '[TeamCity] Server tests'	No artifacts	Evgeniy Koshkin (1)	1h:12m left		Stop
#11788	Success	No artifacts	Dmitry Neverov (1)	14 minutes ago (14m:02s)		
IntegrationBuild (Linux)			Pending (2)			Run ...
#7928	Tests failed: 2 (1 new), passed: 1897, ignored: 31; Running '[TeamCity] Server tests'	No artifacts	Changes (3)	1h:59m left		Stop
#7927	Tests failed: 1 (1 new), passed: 6926, ignored: 55, muted: 3; Running '[TeamCity] Server tests'	No artifacts	Evgeniy Koshkin (1)	1h:19m left		Stop
#7926	Tests passed: 12144, ignored: 75, muted: 4; Running '[TeamCity] Integration tests'	No artifacts	Changes (4)	40m:54s left		Stop
#7925	Tests failed: 1 (1 new), passed: 12301, ignored: 75, muted: 3; Running '[TeamCity] Integration tests...	No artifacts	Changes (2)	overtime: 8m:57s		Stop
#7924	Tests failed: 2 (1 new), passed: 13705, ignored: 91, muted: 4	No artifacts	Changes (3)	10 hours ago (2h:02m)		
IntegrationBuild (MacOS)	Test failures: 2 not investigated, 3 under investigation, 3 muted		Pending (3)	2 running	and 1 queued	Run ...
IntegrationBuild (MS SQL)	Test failures: 6 not investigated, 2 under investigation, 2 muted		Pending (4)	3 running	and 1 queued	Run ...
IntegrationBuild (MySQL)	Test failures: 2 not investigated, 3 under investigation, 5 muted		Pending (10)	1 running	and 1 queued	Run ...

How we do it

- Virtual Androids on Mac minis
- Android Emulators in Docker

- UiAutomator and Espresso tests triggered in TeamCity

- Own UiAutomator library for test support and execution

- Hundreds of tests runned every night

Outsourcing

- CI as a service
  - Travis, CircleCI, Shippable

- Testing as a service
  - Very expensive
  - “They will never know the product like you do”
  - Cleverlance

avast

What you should remember?

1. What is testing
2. Role of QA in development process
3. Little bit of theory
4. JUnit framework
5. Watching health of app
6. Bug reporting
7. Automated testing
8. Espresso + UiAutomator framework
9. Continuous integration

avast

What you should remember?

1. What is testing
   ➜ **Verifying that you app does what is should and does not piss off users**
2. Role of QA in development process
3. Little bit of theory
4. JUnit framework
5. Watching health of app
6. Bug reporting
7. Automated testing
8. Espresso + UiAutomator framework
9. Continuous integration

avast

What you should remember?

1. What is testing
   ➜ **Verifying that you app does what is should and does not piss off users**
2. Role of QA in development process
   ➜ **Testing starts with requirements**
3. Little bit of theory
4. JUnit framework
5. Watching health of app
6. Bug reporting
7. Automated testing
8. Espresso + UiAutomator framework
9. Continuous integration

avast

What you should remember?

1. What is testing
   ➜ **Verifying that you app does what is should and does not piss off users**
2. Role of QA in development process
   ➜ **Testing starts with requirements**
3. Little bit of theory
   ➜ **Choose testing strategy**
4. JUnit framework
5. Watching health of app
6. Bug reporting
7. Automated testing
8. Espresso + UiAutomator framework
9. Continuous integration

avast

What you should remember?

1. What is testing
   ➜ **Verifying that you app does what is should and does not piss off users**
2. Role of QA in development process
   ➜ **Testing starts with requirements**
3. Little bit of theory
   ➜ **Choose testing strategy**
4. JUnit framework
   ➜ **Write tests for methods where it makes sense**
5. Watching health of app
6. Bug reporting
7. Automated testing
8. Espresso + UiAutomator framework
9. Continuous integration

avast

What you should remember?

1. What is testing
   ➜ **Verifying that you app does what is should and does not piss off users**
2. Role of QA in development process
   ➜ **Testing starts with requirements**
3. Little bit of theory
   ➜ **Choose testing strategy**
4. JUnit framework
   ➜ **Write tests for methods where it makes sense**
5. Watching health of app
   ➜ **Monitor crashes and feedback of users**
6. Bug reporting
7. Automated testing
8. Espresso + UiAutomator framework
9. Continuous integration

avast

What you should remember?

1. What is testing
   ➜ **Verifying that you app does what is should and does not piss off users**
2. Role of QA in development process
   ➜ **Testing starts with requirements**
3. Little bit of theory
   ➜ **Choose testing strategy**
4. JUnit framework
   ➜ **Write tests for methods where it makes sense**
5. Watching health of app
   ➜ **Monitor crashes and feedback of users**
6. Bug reporting
   ➜ **Organize bug reports and communicate**
7. Automated testing
8. Espresso + UiAutomator framework
9. Continuous integration

avast

What you should remember?

1. What is testing
   ➜ **Verifying that you app does what is should and does not piss off users**
2. Role of QA in development process
   ➜ **Testing starts with requirements**
3. Little bit of theory
   ➜ **Choose testing strategy**
4. JUnit framework
   ➜ **Write tests for methods where it makes sense**
5. Watching health of app
   ➜ **Monitor crashes and feedback of users**
6. Bug reporting
   ➜ **Organize bug reports and communicate**
7. Automated testing
   ➜ **Automate repetitive boring tasks, but not everything**
8. Espresso + UiAutomator framework
9. Continuous integration

avast

What you should remember?

1. What is testing
   ➜ **Verifying that you app does what is should and does not piss off users**
2. Role of QA in development process
   ➜ **Testing starts with requirements**
3. Little bit of theory
   ➜ **Choose testing strategy**
4. JUnit framework
   ➜ **Write tests for methods where it makes sense**
5. Watching health of app
   ➜ **Monitor crashes and feedback of users**
6. Bug reporting
   ➜ **Organize bug reports and communicate**
7. Automated testing
   ➜ **Automate repetitive boring tasks, but not everything**
8. Espresso + UiAutomator framework
   ➜ **Automate end-to-end functional tests**
9. Continuous integration

avast

What you should remember?

1. What is testing
   ➜ **Verifying that you app does what is should and does not piss off users**
2. Role of QA in development process
   ➜ **Testing starts with requirements**
3. Little bit of theory
   ➜ **Choose testing strategy**
4. JUnit framework
   ➜ **Write tests for methods where it makes sense**
5. Watching health of app
   ➜ **Monitor crashes and feedback of users**
6. Bug reporting
   ➜ **Organize bug reports and communicate**
7. Automated testing
   ➜ **Automate repetitive boring tasks, but not everything**
8. Espresso + UiAutomator framework
   ➜ **Automate end-to-end functional tests**
9. Continuous integration
   ➜ **Automate building, testing and deployment**

avast

Thank you!

Q&A

flek@avast.com