

# Utilização do Appium como ferramenta de testes de UI em aplicações iOS

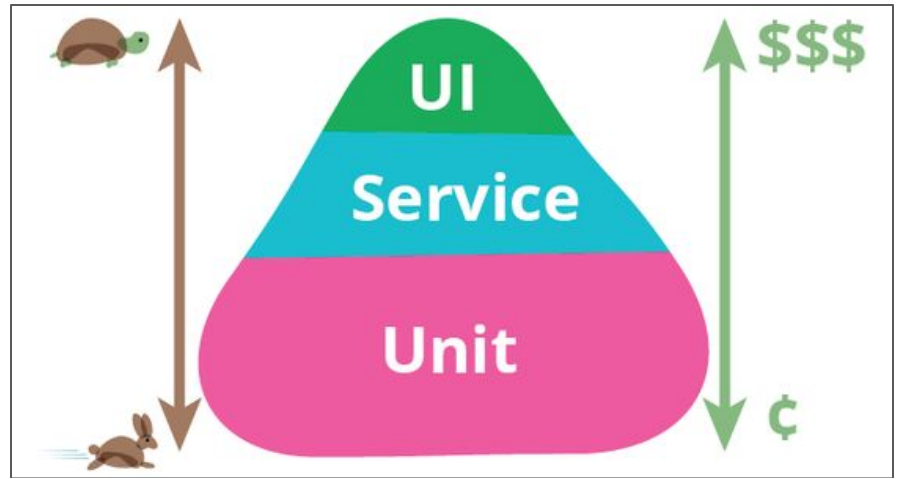
Thiago Garcia

# Diferença entre fases de teste, tipos de teste e formas de execução

# O que são níveis de teste?

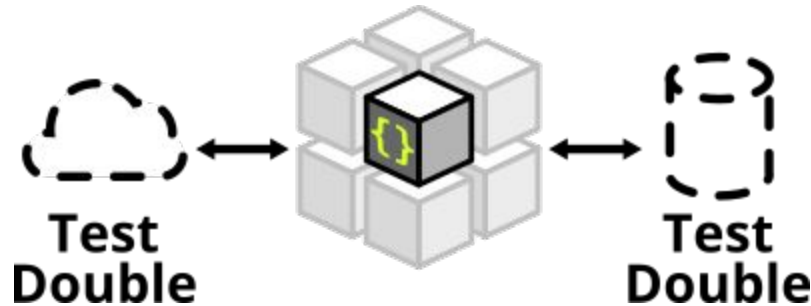
Baseado na literatura do ISTQB (International Software Testing Qualifications Board), os níveis de teste são grupos de atividades de teste que são organizados e gerenciados juntos. Cada nível de teste é uma instância do processo de teste.

- Teste de Unidade
- Teste de integração
- Teste do sistema
- Teste de aceite



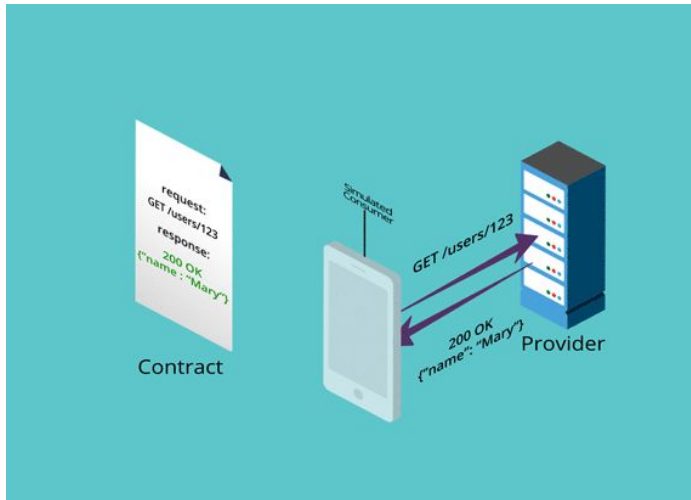
# Teste de Unidade

- O Teste de unidade ou de módulo, verifica o funcionamento da menor unidade de um código testável da aplicação, independente da interação dela com outras partes do nosso código.
- Esse tipo de teste geralmente é realizado isoladamente do resto do sistema. Pode utilizar objetos simulados, virtualização de serviços, simuladores e controladores. O teste unitário pode cobrir funcionalidade (por exemplo: correção de cálculos), características não funcionais (por exemplo, busca de vazamentos de memória) e propriedades estruturais (por exemplo: teste de decisão).



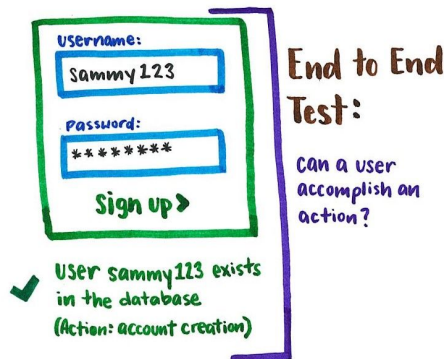
# Teste de integração

- Os testes de integração são os mais comuns de serem esquecidos na hora de escrever testes para um projeto. Mesmo testando duas unidades que interagem entre si separadamente, usando mocks, virtualização etc, e concluindo que ambas estão funcionando como esperado. Ainda assim, é possível que as duas unidades não funcionem bem em conjunto.



# Teste do sistema

- O teste de sistema se concentra no comportamento e nas capacidades de todo um sistema ou produto. Geralmente considerando as execuções das tarefas de ponta a ponta do sistema e os comportamentos não funcionais exibidos ao executar tais tarefas.
- O teste do sistema geralmente produz informações que são usadas pelos stakeholders para tomar decisões de liberação. O teste do sistema também pode satisfazer requisitos ou padrões legais e regulatórios. Esse tipo de teste é o mais comum de se implantar, pois simula a experiência do usuário. São, portanto, muito importantes, pois são os testes mais próximos do que o usuário realmente vai encontrar ao utilizar a aplicação.



# Appium

# Appium

Appium é uma das ferramentas mais populares para automatizar aplicativos nativos e híbridos em plataformas móveis iOS, móveis Android e desktop Windows. É importante ressaltar que o Appium é “cross-platform”: ele permite que você escreva testes em várias plataformas (iOS, Android), usando a mesma API.

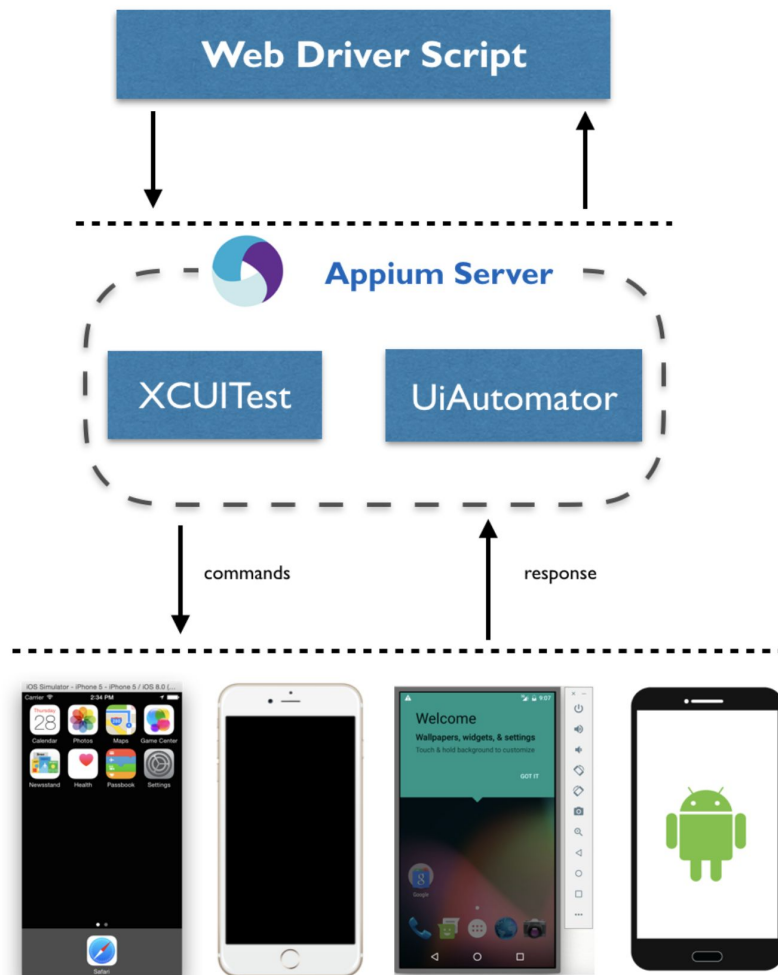


# Benefícios Appium

- É uma ferramenta de código aberto
- Pode ser usado para automatizar testes de aplicativos nativos, da web e híbridos em plataformas Android e iOS
- "cross-platform", o que significa que a API Appium pode ser usada para escrever testes em várias linguagens
- ele utiliza a biblioteca WebDriver, que foi expandida por funções adicionais úteis para automação de teste móvel.
- A ideia por trás do Appium é permitir a automação de testes sem a necessidade de recompilar o aplicativo.

# Funcionamento

A arquitetura do Appium é muito simples. Em primeiro lugar, é importante ressaltar que o Appium é baseado na arquitetura cliente-servidor. O cliente se conecta ao servidor Appium na forma de um objeto JSON por meio do protocolo HTTP. Em seguida, o servidor cria uma sessão, dando ao cliente um número de sessão individual (ID), que fica ativo durante a operação do servidor. Em seguida, os testes são executados como parte da sessão que você criou.



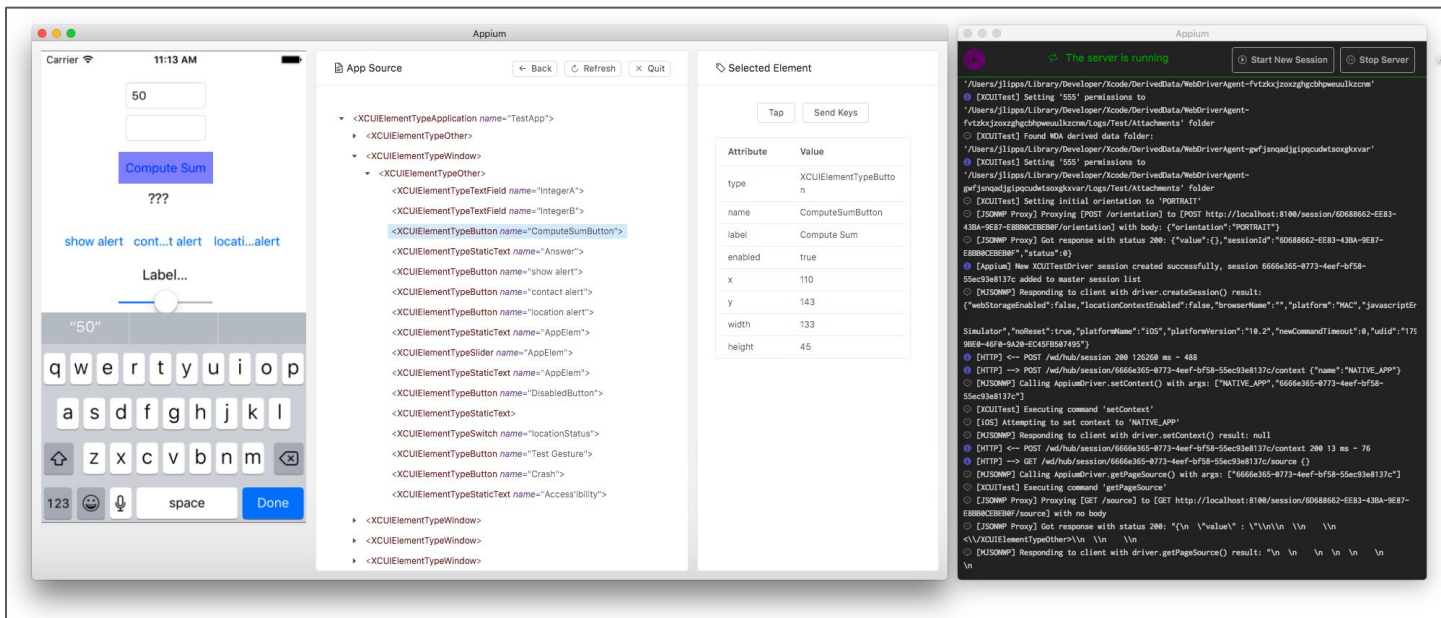
# Instalação do Appium

# Instalação do Appium

1. `brew install node`      `# get node.js`
2. `npm install -g appium`    `# get appium`
3. `npm install wd`            `# get appium client`
4. `appium &`                    `# start appium`

# Instalação do Appium desktop

1. <https://github.com/appium/appium-desktop/releases/tag/v1.21.0>









# Preparação da Aplicação



# Preparação da Aplicação




# Preparação da Aplicação



**Custom Class**

Class   

Module  

**Identity**

Restoration ID

**User Defined Runtime Attributes**

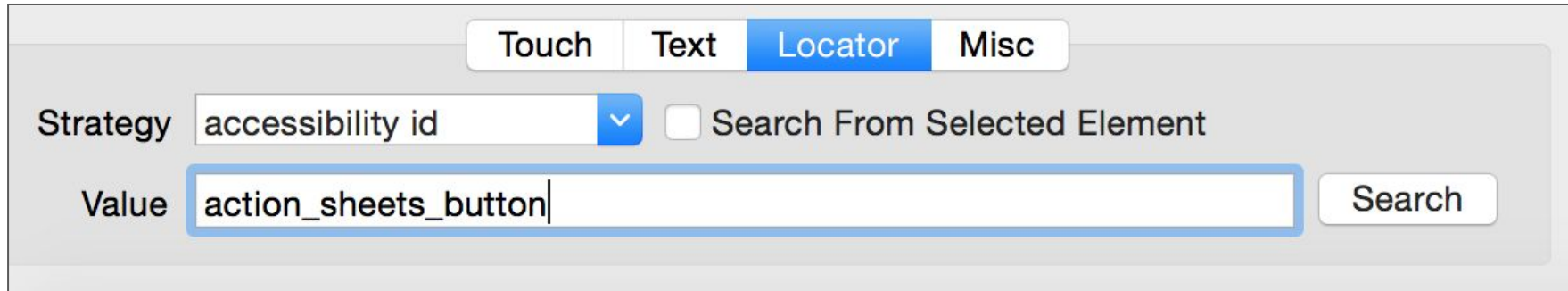
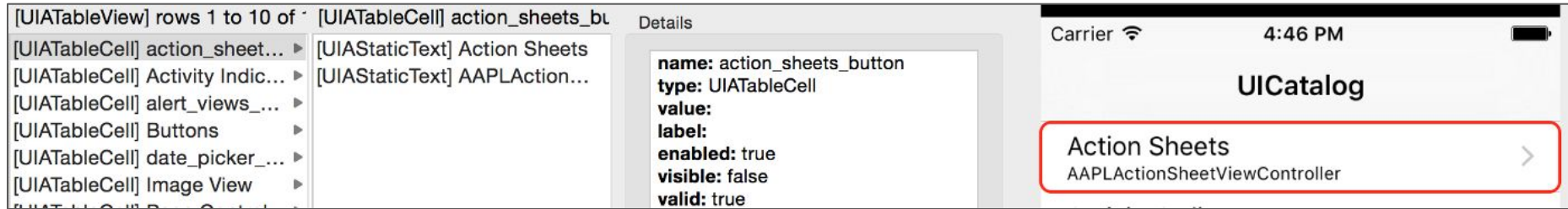
Key Path	Type	Value
+ —		

User Defined Runtime Attributes		
Key Path	Type	Value
accessibilityIdentifier	String	⬢ action_sheets_button
+ —		

**Document**



# Preparação da Aplicação



# Estruturação dos testes

# Estruturação dos testes

- Usaremos o IntelliJ IDEA
  - <https://www.jetbrains.com/idea/download/#section=mac>
- Devemos importar todas as dependências necessárias para o projeto

```
<!-- https://mvnrepository.com/artifact/io.appium/java-client -->
<dependency>
  <groupId>io.appium</groupId>
  <artifactId>java-client</artifactId>
  <version>7.5.1</version>
</dependency>

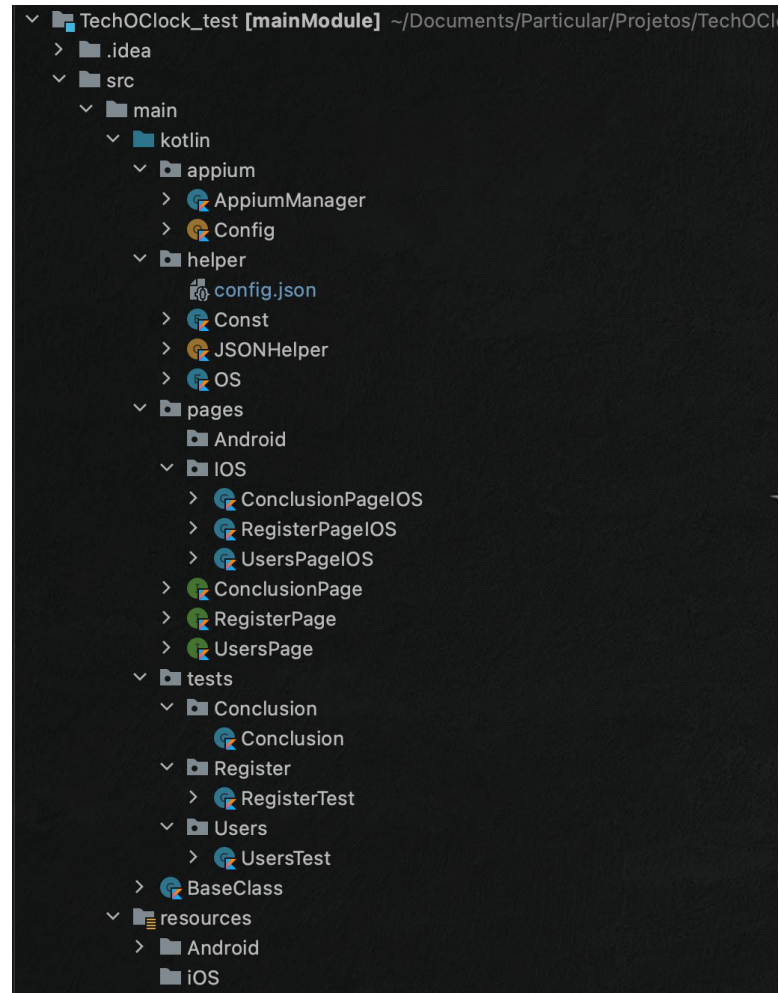
<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
<dependency>
  <groupId>org.seleniumhq.selenium</groupId>
  <artifactId>selenium-java</artifactId>
  <version>3.141.59</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.testng/testng -->
<dependency>
  <groupId>org.testng</groupId>
  <artifactId>testng</artifactId>
  <version>7.4.0</version>
  <scope>test</scope>
</dependency>
<dependency>
  <groupId>org.testng</groupId>
  <artifactId>testng</artifactId>
  <version>7.4.0</version>
  <scope>compile</scope>
</dependency>

<!-- https://mvnrepository.com/artifact/com.googlecode.json-simple/json-simple -->
<dependency>
  <groupId>com.googlecode.json-simple</groupId>
  <artifactId>json-simple</artifactId>
  <version>1.1.1</version>
</dependency>
```

# Estruturação dos testes

- Ficheiro “Main”
  - Responsável pela programação dos testes
    - **Appium**: responsável pela configuração do appium
    - **Helper** : responsável pela configuração do S.O.
    - **Pages** : implementação dos testes por ecrã da App
    - **Testes** : ficheiros base para cada testes em particular
    - **Base Class**: Responsável por dar início aos testes
- Ficheiro “Resources”
  - Local onde inserimos as apps depois de compiladas (.apk / .ipa)

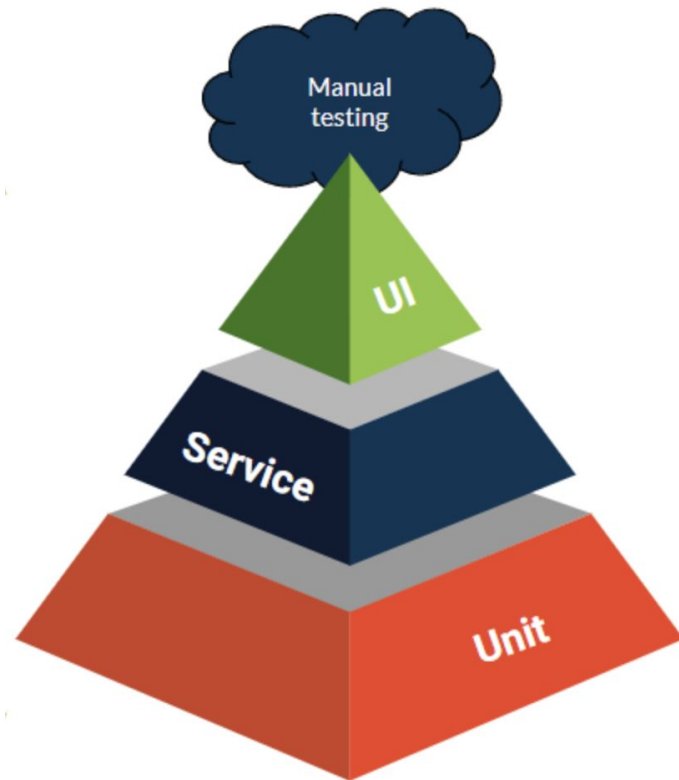


# Demonstração

# Próximos passos...

# Próximos passos...

- Testes de serviço
- Testes unitários
- Automação de publicação e integração com os testes



- **E-mail**

- *thiago.garcia@itsector.pt*

- **Github**

- *<https://github.com/programacaoThiagoGarcia/TechOClock>*

ThiagoGarcia Ficheiro de anotações		5eae39 8 hours ago	🕒 3 commits
📁 Screenshot	first commit	8 hours ago	
📁 TechOClock_ios	first commit	8 hours ago	
📁 TechOClock_test	first commit	8 hours ago	
📁 TechOClock_Android	first commit	8 hours ago	
📄 README.md	Update README.md	8 hours ago	
📄 annotation.txt	Ficheiro de anotações	8 hours ago	



**Obrigado!**

