Wrocław , data 24.05.2019r.

STUDIA PODYPLOMOWE - TESTER OPROGRAMOWANIA

Wyższa Szkoła Bankowa we Wrocławiu

MINI PROJEKT – AUTOMATYZACJA TESTÓW DLA APLIKACJI MOBILNEJ ANDROID

Testowanie mobilne z Appium

Sprawdził:

mgr inż. Grzegorz Mazur

Opracował/Opracowali:

Ewa Kwaśniewska

Katarzyna Gabrysiak

SEKCJA A

HARDWARE	
Oprogramowanie	
Typ Windows: Wersja: „Ilobitowy”: 64 bitowy sysytem operacyjny Procesor: PAMIĘĆ RAM:	Windows 10 HOME 1803 64 bitowy sysytem operacyjny Intel(R) Core(TM) i7-8750H CPU@ 2.20GHz 2.21 GHz 8,00 GB
Nazwa komputera:	LAPTOP-8A0DAI0S ideapad 330-15ICH
TELEFON	
Nazwa telefonu: Model: Wersja systemu Android: Pamięć RAM: Ekran:	Huawei P Smart FIG-LX1 8.0.0 3.0 GB 2160x1080
EMULATOR	
Emulator Model System operacyjny Instalacja AVD Name	Emulator-5554 Nexus 5 Android Marshmallow dla x86 Intel HAXM mojemulator

SOFTWARE	wersja
Android Studio Do pobrania: https://developer.android.com/studio/index.html Skonfigurować z Android SDK Skonfigurować emulator AVD (Android Virtual Device)	v. 3.3.2
Java W wierszu poleceń CMD sprawdzam java-version	v. 1.8.0_192
Node.js Sprawdzić czy jest zainstalowany /w ścieżce: C:\Program Files\nodejs\ Jeśli nie- zainstalować dla Windows https://nodejs.org/en/download/ Sprawdzamy w cmd: node -v enter	v. 10.15.3
Npm Sprawdzamy w cmd: npm -v enter	v. 6.4.1
Appium Sprawdzenie czy jest zainstalowane C:\Users\48793\AppData\Local\Programs\appium-desktop	1.13.0

Jeśli nie, w cmd: npm install -g appium	
Appium doctor Sprawdzam w cmd: appium-doctor Jeśli nie, w cmd: npm install appium-doctor -g	v. 1.10.0
PyCharm https://www.jetbrains.com/pycharm/ może być wersja Community	v. 2018.3.5
Python https://www.python.org/downloads/windows/	v. 2.7.16
Appium Python Client Instalacja: https://github.com/appium/python-client lub cmd: pip install Appium-Python-Client sprawdzam wersję: pip list	v. 0.40
Selenium cmd: python enter w konsoli pythona: import selenium help (selenium) exit	v. 3.141
APK-Info APK-Info.exe > wybieram aplikację Potrzebne 2 informacje: Package launchable-activity:name=	v. 1.32 (08.12.2018)

DODANO ŚCIEŻKI do zmiennej systemowej
PATH:
C:\Program Files\Android\Android Studio\bin C:\Program Files\Java\jdk1.8.0_192\bin\ C:\Users\48793\AppData\Local\Android\Sdk\tools\bin C:\Users\48793\AppData\Local\Android\Sdk\tools C:\Users\48793\AppData\Local\Android\Sdk\platform-tools C:\Program Files\nodejs\node_modules\npm\bin C:\Python27 C:\Python27/Scripts Ścieżka do APK np. C:\ Chilternrailways\Chilternraylways Tickets_v4.0_apkpure.com.apk
JAVA HOME:
C:\Program Files\Java\jdk1.8.0_192\
ANDROID_HOME
C:\Users\48793\AppData\Local\Android\Sdk

NODE_HOME
C:\Program Files\nodejs\

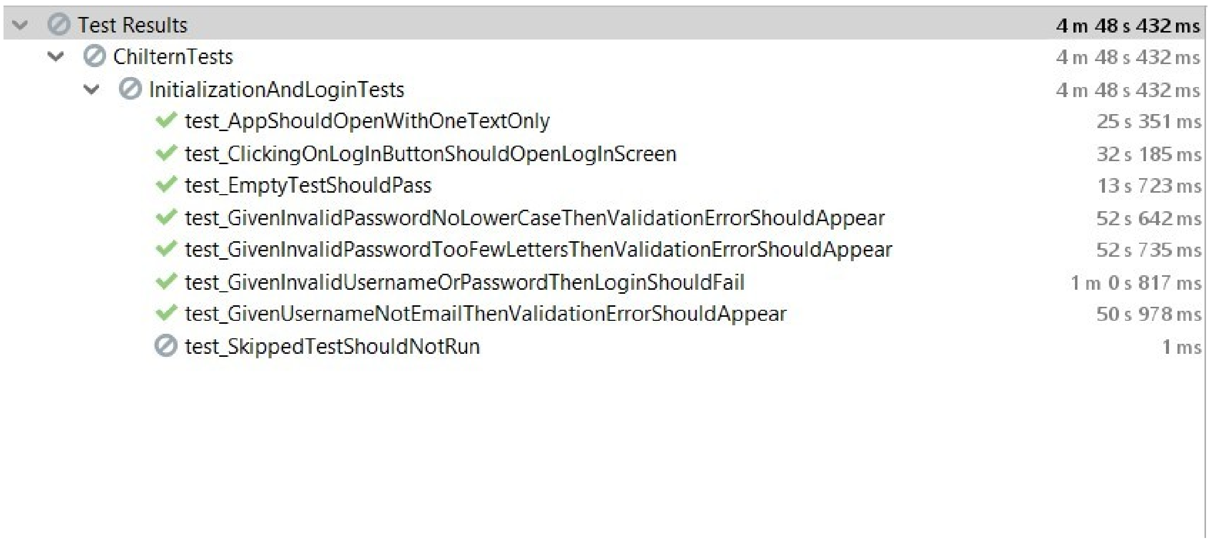
ABY ZACZĄĆ PISAĆ TESTY /URUCHOMIĆ TESTY:

- W wierszu poleceń cmd uruchomić: Appium. Działa w tle.
- W nowym oknie cmd > C:\Users\48793\AppData\Local\Android\Sdk\tools\bin >
  > uia +tab = na końcu bat (do wykonywania zdjęc emulatora -
  lokalizowania (inspekcja) elementów: xpath, id, text atrributes, classNames etc.
- Pobieram aplikację *.apk dla Androida / lub *.ipk dla iOS
  Wystarczy przenieść myszą na ekran emulatora i zostaje zainstalowana
- Uruchomić Android Studio > wybieram projekt > uruchamiam emulator
  albo w cmd sprawdzić działanie emulatora przez ADB (Android Debug Bridge):
  cd C:\Users\48793\AppData\Local\Android\Sdk\tools
  > emulator -avd mojemulator - uruchamia emulator
- Uruchamiam PyCharm; nowy plik .py > pamiętam o Configure Python Interpreter
- w def setUp(self) określamy desired_capabilities:
  wersja androida / nazwa emulatora/urządzenia / ścieżka do aplikacji.APK)
  pakiet package, activity)
- analiza logów Appium Server
- aby sprawdzić czy ADB widzi urządzenia wpisuje w cmd adb devices. Wymagany tryb deweloperski dla androida-USB debugging na ON.
- Android Studio: Configure:
  AVD
  SDK Manager

SEKCJA B

Przypadki testowe: 8 testów.
Link: https://screencast-o-matic.com/watch/cqh0DHTl0L

Nazwa testowanej aplikacji: Chilternrailways



```
Test Results                                                              4 m 48 s 432 ms
  ChilternTests                                                           4 m 48 s 432 ms
    InitializationAndLoginTests                                           4 m 48 s 432 ms
      ✓ test_AppShouldOpenWithOneTextOnly                                      25 s 351 ms
      ✓ test_ClickingOnLogInButtonShouldOpenLogInScreen                        32 s 185 ms
      ✓ test_EmptyTestShouldPass                                               13 s 723 ms
      ✓ test_GivenInvalidPasswordNoLowerCaseThenValidationErrorShouldAppear    52 s 642 ms
      ✓ test_GivenInvalidPasswordTooFewLettersThenValidationErrorShouldAppear  52 s 735 ms
      ✓ test_GivenInvalidUsernameOrPasswordThenLoginShouldFail             1 m 0 s 817 ms
      ✓ test_GivenUsernameNotEmailThenValidationErrorShouldAppear              50 s 978 ms
      ⊘ test_SkippedTestShouldNotRun                                                1 ms
```

```
import os
import unittest
from appium import webdriver
from time import sleep
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By

class InitializationAndLoginTests(unittest.TestCase):

    def waitUntil(self, timeout, path):
        wait = WebDriverWait(self.driver, timeout)
        wait.until(EC.element_to_be_clickable((By.XPATH, path)))

    def getElements(self, class_name):
        elements = self.driver.find_elements_by_class_name(class_name)
        for el in elements:
            print "%s: text:'%s', resourceId:'%s'" % (class_name, el.text,
el.get_attribute('resourceId'))
        return elements

    def moveToLoginPage(self):

        self.waitUntil(30, '//android.widget.TextView[@text="Log in"]')
        self.getElements('android.widget.TextView')

        button =
self.driver.find_element_by_xpath('//android.widget.TextView[@text="Log
in"]')
        button.click()

        text_view_elements = self.getElements('android.widget.TextView')
        edit_text_elements = self.getElements('android.widget.EditText')
        button_elements = self.getElements('android.widget.Button')

        self.assertEquals(len(edit_text_elements), 2)
        self.assertEquals(edit_text_elements[0].text, "")
        self.assertEquals(edit_text_elements[1].text, "")
        self.assertEquals(len(button_elements), 1)
        self.assertEquals(button_elements[0].text, "Log in")

    def setUp(self):
        print "setUp: %s " % self.id()
        desired_caps = {'platformName': 'Android',
                        'platformVersion': '6.0',
                        'deviceName': 'emulator-5554',
                        'appPackage': 'com.ormlondon.chilternace',
                        'appActivity':
'com.ormlondon.chilternace.MainActivity'}

        self.driver = webdriver.Remote('http://localhost:4723/wd/hub',
desired_caps)

    def tearDown(self):
        print "tearDown: %s \n" % self.id()
        self.driver.quit()

    @unittest.skip("skipping")
    def test_SkippedTestShouldNotRun(self):
        print "Running: %s ..." % self.id()
        self.fail("shouldn't happen")

    # @unittest.skip("skipping")
    def test_EmptyTestShouldPass(self):
        print "Running: %s" % self.id()
        self.assertTrue(True)

    # @unittest.skip("skipping")
    def test_AppShouldOpenWithOneTextOnly(self):
        print "Running: %s" % self.id()
        text_view_elements =
self.driver.find_elements_by_class_name('android.widget.TextView')

        self.assertEquals(len(text_view_elements), 1, "there should be only
one element")
        self.assertEquals(text_view_elements[0].text, "The fastest way to \
nbook tickets", "someone has changed this text, ups!")

    # @unittest.skip("skipping")
    def test_ClickingOnLogInButtonShouldOpenLogInScreen(self):
        print "Running: %s" % self.id()

        self.waitUntil(30, '//android.widget.TextView[@text="Log in"]')

        text_view_elements =
self.driver.find_elements_by_class_name('android.widget.TextView')
        for el in text_view_elements:
            print "text_view_element: text:'%s', resourceId:'%s'" %
(el.text, el.get_attribute('resourceId'))

        button =
self.driver.find_element_by_xpath('//android.widget.TextView[@text="Log
in"]')
        button.click()

        text_view_elements =
self.driver.find_elements_by_class_name('android.widget.TextView')
        for el in text_view_elements:
            print "text_view_element: text:'%s', resourceId:'%s'" %
(el.text, el.get_attribute('resourceId'))

        edit_text_elements =
self.driver.find_elements_by_class_name('android.widget.EditText')
        for el in edit_text_elements:
            print "edit_text_element: text:'%s', resourceId:'%s'" %
(el.text, el.get_attribute('resourceId'))

        button_elements =
self.driver.find_elements_by_class_name('android.widget.Button')
        for el in button_elements:
            print "button_element: text:'%s', resourceId:'%s'" % (el.text,
el.get_attribute('resourceId'))

        self.assertEquals(len(edit_text_elements), 2)
        self.assertEquals(edit_text_elements[0].text, "")
        self.assertEquals(edit_text_elements[1].text, "")

        self.assertEquals(len(button_elements), 1)
        self.assertEquals(button_elements[0].text, "Log in")

    # @unittest.skip("skipping")
    def test_GivenUsernameNotEmailThenValidationErrorShouldAppear(self):
        print "Running: %s" % self.id()

        self.moveToLoginPage()

        input_email =
self.driver.find_element_by_id('com.ormlondon.chilternace:id/account_login_
input_email')
        input_password =
self.driver.find_element_by_id('com.ormlondon.chilternace:id/account_login_
input_password')
        login_button =
self.driver.find_element_by_id('com.ormlondon.chilternace:id/login_button')

        input_email.send_keys("tester")
        input_password.send_keys("123456")

        login_button.click()

        self.getElements('android.widget.TextView')
        self.getElements('android.widget.EditText')

        textinput_error =
self.driver.find_element_by_id('com.ormlondon.chilternace:id/textinput_erro
r')
        self.assertEquals(textinput_error.text, "Please provide a valid
email address")

    # @unittest.skip("skipping")
    def
test_GivenInvalidPasswordTooFewLettersThenValidationErrorShouldAppear(self)
:
        print "Running: %s" % self.id()

        self.moveToLoginPage()

        input_email =
self.driver.find_element_by_id('com.ormlondon.chilternace:id/account_login_
input_email')
        input_password =
self.driver.find_element_by_id('com.ormlondon.chilternace:id/account_login_
input_password')
        login_button =
self.driver.find_element_by_id('com.ormlondon.chilternace:id/login_button')

        input_email.send_keys("tester@testerzy.pl")
        input_password.send_keys("123456")

        login_button.click()

        self.getElements('android.widget.TextView')
        self.getElements('android.widget.EditText')
        self.getElements('android.widget.Button')

        textinput_error =
self.driver.find_element_by_id('com.ormlondon.chilternace:id/textinput_erro
r')
        self.assertEquals(textinput_error.text, "At least 8 characters
required")

    # @unittest.skip("skipping")
    def
test_GivenInvalidPasswordNoLowerCaseThenValidationErrorShouldAppear(self):
        print "Running: %s" % self.id()

        self.moveToLoginPage()

        input_email =
self.driver.find_element_by_id('com.ormlondon.chilternace:id/account_login_
input_email')
        input_password =
self.driver.find_element_by_id('com.ormlondon.chilternace:id/account_login_
input_password')
        login_button =
self.driver.find_element_by_id('com.ormlondon.chilternace:id/login_button')

        input_email.send_keys("tester@testerzy.pl")
        input_password.send_keys("12345678")

        login_button.click()

        self.getElements('android.widget.TextView')
        self.getElements('android.widget.EditText')
        self.getElements('android.widget.Button')

        textinput_error =
self.driver.find_element_by_id('com.ormlondon.chilternace:id/textinput_erro
r')
        self.assertEquals(textinput_error.text, "At least one lowercase
letter is required")

    # @unittest.skip("skipping")
    def test_GivenInvalidUsernameOrPasswordThenLoginShouldFail(self):
        print "Running: %s" % self.id()

        self.moveToLoginPage()

        input_email =
self.driver.find_element_by_id('com.ormlondon.chilternace:id/account_login_
input_email')
        input_password =
self.driver.find_element_by_id('com.ormlondon.chilternace:id/account_login_
input_password')
        login_button =
self.driver.find_element_by_id('com.ormlondon.chilternace:id/login_button')

        input_email.send_keys("tester@testerzy.pl")
        input_password.send_keys("1qaz@WSX")

        login_button.click()

        self.waitUntil(30, '//android.widget.TextView[@text="Please
wait"]')

        alertTitle =
self.driver.find_element_by_id('android:id/alertTitle')
        message = self.driver.find_element_by_id('android:id/message')

        self.assertEquals(alertTitle.text, "Please wait")
        self.assertEquals(message.text, "Operation in progress")

        self.waitUntil(30, '//android.widget.Button[@resource-
id="android:id/button1"]')

        alertTitle =
self.driver.find_element_by_id('android:id/alertTitle')
        message = self.driver.find_element_by_id('android:id/message')
        button1 = self.driver.find_element_by_id('android:id/button1')

        self.assertEquals(alertTitle.text, "Log in")
        self.assertEquals(message.text, "User tester@testerzy.pl not
found")
        self.assertEquals(button1.text, "OK")


if __name__ == "__main__":
    suite =
unittest.TestLoader().loadTestsFromTestCase(InitializationAndLoginTests)
    unittest.TextTestRunner(verbosity=2).run(suite)
```