ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ ΠΛΗΡΟΦΟΡΙΑΣ ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΦΑΡΜΟΓΗ ΜΕΘΟΔΟΛΟΓΙΩΝ ΚΑΙ ΤΕΧΝΙΚΩΝ ΔΙΑΣΦΑΛΙΣΗΣ ΠΟΙΟΤΗΤΑΣ ΣΕ ΔΙΑΔΙΚΤΥΑΚΟΥΣ ΙΣΤΟΤΟΠΟΥΣ

Υπεύθυνος Φοιτητής
ΚΑΦΦΕ ΚΩΝΣΤΑΝΤΙΝΑ

Νοέμβριος 2021

Επιβλέπων Καθηγητής
ΧΑΤΖΗΓΕΩΡΓΙΟΥ ΑΛΕΞΑΝΔΡΟΣ

Στόχος

- Μελέτη της σημασίας του αυτοματοποιημένου και του χειροκίνητου ελέγχου ενός λογισμικού
- Μελέτη μεθοδολογιών και τεχνικών του αυτοματοποιημένου ελέγχου εφαρμογών ιστού
- Μελέτη εργαλείων για αυτοματοποιημένο έλεγχο εφαρμογών ιστού
- Εφαρμογή τεχνικών και εργαλείων σε συγκεκριμένη εφαρμογή ιστού

Εργαλεία / Τεχνολογίες που χρησιμοποιήθηκαν

- Java (Λειτουργικότητα)
- Intellij IDEA (Περιβάλλον ανάπτυξης)
- Selenium (Εργαλείο ανάπτυξης αυτοματοποιημένων σεναρίων)
- JUnit (Εργαλείο ανάπτυξης αυτοματοποιημένων σεναρίων)
- TestNG (Εργαλείο ανάπτυξης αυτοματοποιημένων σεναρίων)
- Cucumber (Εργαλείο ανάπτυξης αυτοματοποιημένων σεναρίων)
- Jsoup, Bootstrap, JQuery, CSS (Δημιουργία αναφοράς)

Selenium

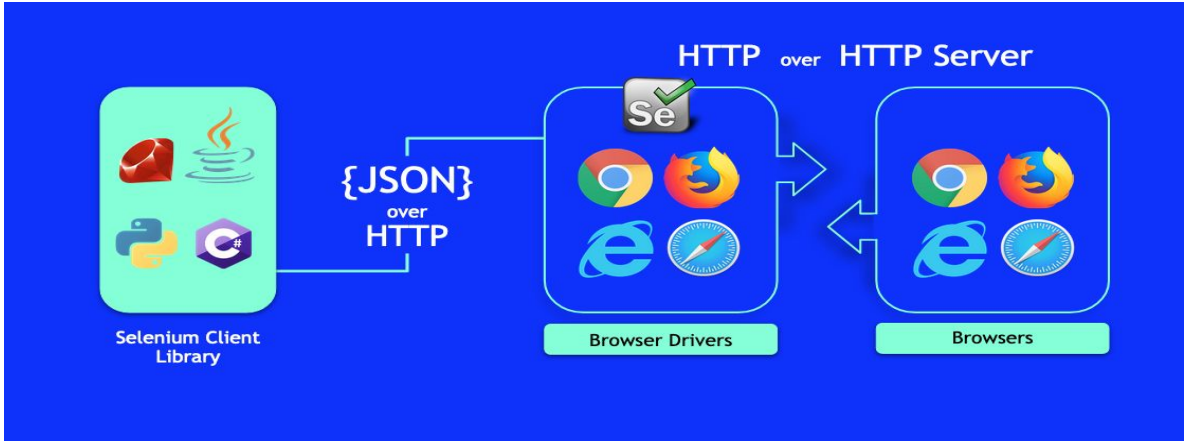
Το Selenium είναι ένα πλαίσιο αυτοματοποιημένων δοκιμών ανοικτού κώδικα που χρησιμοποιείται για την επικύρωση εφαρμογών ιστού σε διαφορετικά προγράμματα περιήγησης και πλατφόρμες. Το λογισμικό Selenium δεν είναι μόνο ένα εργαλείο αλλά μια σουίτα λογισμικού, κάθε κομμάτι της οποίας καλύπτει διαφορετικές ανάγκες Selenium QA testing ενός οργανισμού.

Τα 4 κύρια συστατικά της πλατφόρμας Selenium είναι τα εξής:

- Selenium Integrated Development Environment (IDE)
- Selenium Remote Control
- Selenium WebDriver
- Selenium Grid

Selenium WebDriver

Στη παρούσα εργασία ασχοληθήκαμε με το Selenium WebDriver, η αρχιτεκτονική του οποίου αναλύεται παρακάτω. Το test γράφεται στην επιθυμητή γλώσσα προγραμματισμού, αυτό μετατρέπεται σε JSON και κοινοποιείται μέσω HTTP (REST API) στον συγκεκριμένο browser-driver που χρησιμοποιείται και αυτό επικοινωνεί με τη σειρά του μέσω HTTP με τον ίδιο τον browser, όπου επικοινωνεί πίσω ξανά με μία HTTP απάντηση.



Στην εργασία, πριν από την εκτέλεση κάθε σεναρίου σετάρουμε και ξεκινάμε έναν Chromedriver, και ανοίγουμε την ιστοσελίδα στην οποία θα πραγματοποιήσουμε τα δοκιμαστικά σενάρια.

Selenium WebDriver Locators

Στο WebDriver αυτοματισμό, όλα σχετίζονται με τα web elements. Τα στοιχεία ιστού είναι αντικείμενα DOM που υπάρχουν στην ιστοσελίδα. Προκειμένου να εκτελεστούν λειτουργίες σε ένα web element πρέπει να εντοπιστούν ακριβώς τα στοιχεία. Οι locators βοηθούν στον εντοπισμό των στοιχείων GUI*(web elements) μέσω των οποίων μπορούν να εκτελεστούν πολλαπλές ενέργειες χρήστη. Το Locator είναι μια εντολή που λέει στο Selenium IDE σε ποια στοιχεία GUI (π.χ. πλαίσιο κειμένου, κουμπιά, πλαίσια ελέγχου κ.λπ.) πρέπει να λειτουργήσει. Ο προσδιορισμός των σωστών στοιχείων GUI αποτελεί προϋπόθεση για τη δημιουργία ενός σεναρίου αυτοματισμού.

Selenium WebDriver Locators

Οι διάφοροι τύποι CSS Locators στο Selenium IDE είναι οι εξής:

- id
- name
- className
- tagName
- cssSelector
- xpath
- linkText

Η σύνταξη που χρησιμοποιείται για τον εντοπισμό των στοιχείων είναι ο εξής:

- WebElement element = driver.findElement (By.<Locator>);
- List list = driver.findElements(By.<Locator>);

Selenium Assertions

Τα assertions χρησιμοποιούνται για την επικύρωση μιας δοκιμαστικής περίπτωσης και μας βοηθούν να καταλάβουμε εάν μια δοκιμαστική περίπτωση έχει περάσει ή απέτυχε. Τα assertions θεωρείται ότι πληρούνται εάν το πραγματικό αποτέλεσμα ταιριάζει με αυτό του αναμενόμενου αποτελέσματος. Το κύριο πλεονέκτημα της ύπαρξης ισχυρισμών είναι ο εντοπισμός ελαττωμάτων σε ένα πρόγραμμα. Ένας ισχυρισμός είναι μια boolean έκφραση σε ένα συγκεκριμένο σημείο ενός προγράμματος, η οποία θα είναι αληθής εκτός αν υπάρχει κάποιο σφάλμα στο πρόγραμμα. Ένας ισχυρισμός δοκιμής ορίζεται ως μια έκφραση, η οποία περικλείει κάποια ελέγξιμη λογική που καθορίζεται για έναν υπό δοκιμή στόχο.

Selenium - native frameworks

Υπάρχει μια σειρά από πλαίσια που αναπτύχθηκαν εξαρχής ως, Selenium-native frameworks ώστε να καλύψει τους περιορισμούς του Selenium(υποβολή αναφορών, δυνατότητα εκτέλεσης παράλληλων δοκιμών από έναν υπολογιστή κ.λπ). Μερικά από αυτά τα frameworks είναι τα JUnit, TestNG και Cucumber.

JUnit

Το JUnit είναι ένα πλαίσιο ανοικτού κώδικα που έχει σχεδιαστεί για τη συγγραφή και εκτέλεση δοκιμών στη γλώσσα προγραμματισμού Java. Το Junit χρησιμοποιεί annotations. Το annotation είναι ένα meta-data/meta-tag που βοηθάει στο να προσδιοριστεί ποια ενέργεια πρέπει να εκτελεστεί με μία μέθοδο, επίσης επιτρέπει στους προγραμματιστές να οργανώνουν, να ομαδοποιούν και να διατηρούν τα test cases, και επιπλέον παρέχουν πρόσθετες πληροφορίες σχετικά με τις μεθόδους και τις κλάσεις που ορίζονται στον κώδικα.

- Το JUnit διασφαλίζει ότι κάθε τμήμα ή λειτουργία του λογισμικού δοκιμάζεται.
- Κάθε φορά που κάνουμε μια μικρή ή μεγάλη τροποποίηση στον κώδικα (σε οποιαδήποτε λειτουργία), μπορούμε να βεβαιωθούμε ότι η λειτουργία αποδίδει καλά και ότι δεν έχει σπάσει κάποια παλαιότερη λειτουργικότητα, εκτελώντας όλες τις περιπτώσεις δοκιμών JUnit που έχουν γραφτεί για τη συγκεκριμένη λειτουργία.
- Το JUnit έχει γίνει πρότυπο για τη δοκιμή στη γλώσσα προγραμματισμού Java και υποστηρίζεται από σχεδόν όλα τα IDE, π.χ. Netbeans, Eclipse, IntelliJ κ.λπ. Έτσι, μπορούμε να εργαζόμαστε σε οποιοδήποτε τυποποιημένο περιβάλλον IDE.
- Μπορούμε επίσης να ενσωματώσουμε το Ant με το JUnit για να επιτρέψουμε την εκτέλεση συνόλων δοκιμών (όλων των περιπτώσεων δοκιμών μονάδας) ως μέρος της διαδικασίας κατασκευής, την καταγραφή της εκροής τους και τη δημιουργία πλούσιων αναφορών ενισχυμένων με χρώμα.

TestNG

Το TestNG είναι ένα πλαίσιο δοκιμών αυτοματοποίησης στο οποίο το NG σημαίνει "Next Generation" (επόμενη γενιά). Το TestNG είναι εμπνευσμένο από το JUnit το οποίο χρησιμοποιεί annotations (@). Χρησιμοποιώντας το TestNG, μπορούμε να δημιουργούμε μια κατάλληλη αναφορά και μπορούμε εύκολα να μάθουμε πόσες περιπτώσεις δοκιμών έχουν περάσει, αποτύχει και παραλειφθεί. Επίσης παρέχει τη δυνατότητα εκτέλεσης των αποτυχημένων περιπτώσεων δοκιμής ξεχωριστά.

- Πολυάριθμες περιπτώσεις δοκιμών μπορούν να οργανωθούν με ευκολία μεταφράζοντάς τες σε αρχείο testng.xml.
- Ορίζει προτεραιότητες στην εκτέλεση περιπτώσεων δοκιμής.
- Μια περίπτωση δοκιμής μπορεί να εκτελεστεί σε διαφορετικούς χρόνους χρησιμοποιώντας τη λέξη-κλειδί invocationCount αναθέτοντας έναν αριθμό επανάληψης στη μεταβλητή invocationCount.
- Υποστηρίζει δοκιμές cross-browser και υποστηρίζει εργαλεία όπως το Maven, το Jenkins κ.λπ.
- Σε αντίθεση με το WebDriver, το TestNG διαθέτει ενσωματωμένο μηχανισμό για τη δημιουργία αναφορών σε αναγνώσιμη μορφή.
- Αποτελείται από προεπιλεγμένες συναρτήσεις JDK για το χρόνο εκτέλεσης και την καταγραφή.

TestNG - Suite Test

Μια σουίτα δοκιμών είναι μια συλλογή περιπτώσεων δοκιμών που αποσκοπούν στον έλεγχο μιας συμπεριφοράς ή ενός συνόλου συμπεριφορών ενός προγράμματος λογισμικού(ουσιαστικά είναι ο τρόπος να εκτελέσουμε όλα τα δοκιμαστικά σενάρια). Στο TestNG, δεν μπορούμε να ορίσουμε μια σουίτα στον πηγαίο κώδικα δοκιμών, αλλά αναπαρίσταται από ένα αρχείο XML. Επιτρέπει επίσης την ευέλικτη διαμόρφωση των προς εκτέλεση δοκιμών. Μια σουίτα μπορεί να περιέχει μία ή περισσότερες δοκιμές και ορίζεται από την ετικέτα <suite>.
Η <suite> είναι η ετικέτα ρίζα του testng.xml. Περιγράφει μια σουίτα δοκιμών, η οποία με τη σειρά της αποτελείται από διάφορα τμήματα <test>.

- <suite> - Η ετικέτα σουίτας (tag: suite) μπορεί να έχει οποιοδήποτε όνομα και δηλώνει το όνομα της δοκιμαστικής σουίτας.
- <test> - Η ετικέτα δοκιμής(tag: test) μπορεί να έχει οποιοδήποτε όνομα και υποδεικνύει τα δοκιμαστικά μας σύνολα.
- <classes> - Αυτός είναι ο συνδυασμός του ονόματος πακέτου και του ονόματος της κλάσης που περιλαμβάνει τα δοκιμαστικά σενάρια.

TestNG - Data Driven Testing

Το Data Driven Testing(παροχή παραμέτρων στις μεθόδους δοκιμής) πραγματοποιείται όταν δεδομένα αποθηκεύονται σε μία βάση δεδομένων και στη συνέχεια χρησιμοποιούνται ως είσοδοι σε μία μέθοδο δοκιμής. Είναι μια τεχνική αυτοματοποίησης δοκιμών στην οποία τα δεδομένα δοκιμής και η λογική δοκιμής διατηρούνται χωριστά. Το Data Driven Testing μπορεί να πραγματοποιηθεί μέσω του TestNG χρησιμοποιώντας τον σχολιασμό @DataProvider. Μια μέθοδος με τον σχολιασμό @DataProvider πάνω της επιστρέφει έναν πίνακα 2D του αντικειμένου όπου οι γραμμές καθορίζουν τον αριθμό των επαναλήψεων και οι στήλες καθορίζουν τον αριθμό των παραμέτρων εισόδου που περνούν στη μέθοδο Test με κάθε επανάληψη. Αυτός ο σχολιασμός λαμβάνει ως παράμετρο μόνο το όνομα του παρόχου δεδομένων, το οποίο χρησιμοποιείται για τη σύνδεση του παρόχου δεδομένων με τη μέθοδο Test.

```
@Test (dataProvider = "provideLogInData")
public void login(String email, String password) {
    WebDriverSetUp.driver.findElement(By.id("ap_email")).sendKeys(email);
    WebDriverSetUp.driver.findElement(By.id("continue")).click();
    WebDriverSetUp.driver.findElement(By.id("ap_password")).sendKeys(password);
    WebDriverSetUp.driver.findElement(By.id("signInSubmit")).click();
}

@DataProvider(name = "provideLogInData")
public Object [] [] loginData ()
{
    Object [] [] data = new Object [3] [2];
    data [0] [0] = "kons.kaffe@gmail.com";     data [0] [1] = "konstantina123";
    data [1] [0] = "konakaffe@yahoo.com";      data [1] [1] = "konstantina1234";
    data [2] [0] = "InvalidLogIn@Test.com";    data [2] [1] = "InvalidPassword";
    return data;
}
```

TestNG - Reporting

Τα TestNG Reports είναι οι default αναφορές HTML που δημιουργούνται όταν εκτελούνται οι δοκιμαστικές περιπτώσεις χρησιμοποιώντας το TestNG. Αυτές οι αναφορές βοηθούν να προσδιοριστούν οι πληροφορίες σχετικά με τις δοκιμαστικές περιπτώσεις και την κατάσταση ενός έργου.

Υπάρχουν δύο είδη default αναφορών:

- Detailed report -Συνδυάζει λεπτομερείς πληροφορίες, όπως σφάλματα, ομάδες δοκιμών(test groups), χρόνο εκτέλεσης, αρχεία καταγραφής βήμα προς βήμα.

- Summary report -Είναι η συνοπτική αναφορά της τρέχουσας δοκιμαστικής εκτέλεσης. Αφορά Passed/Failed/Skipped σενάρια δοκιμών.

Cucumber

Το Cucumber είναι ένα εργαλείο δοκιμών που υποστηρίζει την ανάπτυξη δοκιμών με γνώμονα το Behavior Driven Development - BDD. Προσφέρει έναν τρόπο για τη συγγραφή δοκιμών που μπορεί να κατανοήσει ο καθένας, ανεξάρτητα από τις τεχνικές του γνώσεις. Στο BDD, οι χρήστες (επιχειρηματικοί αναλυτές, ιδιοκτήτες προϊόντων) γράφουν πρώτα σενάρια ή δοκιμές που περιγράφουν τη συμπεριφορά του συστήματος από την οπτική γωνία του πελάτη, πριν οι προγραμματιστές γράψουν τους κώδικες τους.

- Είναι χρήσιμο να εμπλέκονται οι ενδιαφερόμενοι επιχειρηματίες που δεν μπορούν εύκολα να διαβάσουν κώδικα.
- Το εργαλείο δοκιμών Cucumber εστιάζει στην εμπειρία του τελικού χρήστη.
- Ο τρόπος συγγραφής των δοκιμών επιτρέπει την ευκολότερη επαναχρησιμοποίηση του κώδικα στις δοκιμές.
- Γρήγορη και εύκολη ρύθμιση και εκτέλεση.

Cucumber - Behavior Driven Development

Το BDD είναι ένας τρόπος εργασίας για τις ομάδες λογισμικού που κλείνει το χάσμα μεταξύ των επιχειρηματιών και των τεχνικών. Η επικοινωνία μεταξύ των δύο μερών είναι συχνά το εμπόδιο στην πρόοδο του έργου όταν οι μηχανικοί συχνά παρεξηγούν τι χρειάζεται πραγματικά η επιχείρηση από το λογισμικό της και οι επαγγελματίες επιχειρήσεων παρεξηγούν τις δυνατότητες της τεχνικής τους ομάδας. Περιτριγυρισμένο από κακή επικοινωνία και πολυπλοκότητα, το τελικό προϊόν είναι συχνά τεχνικά λειτουργικό, αλλά δεν πληροί τις ακριβείς απαιτήσεις της επιχείρησης. Το BDD αποσκοπεί στο να το αλλάξει αυτό. Εστιάζει στον καθορισμό απαιτήσεων με βάση την επιθυμητή συμπεριφορά του λογισμικού. Αυτές οι απαιτήσεις(σενάρια δοκιμών) αρχικά γράφονται σε μία επίσημη γλώσσα(Gherkin) και στη συνέχεια αυτά τα σενάρια μετατρέπονται σε κώδικα.

Cucumber - Gherkin

Το Gherkin είναι μια επίσημη γλώσσα για την περιγραφή λειτουργιών λογισμικού και σεναρίων δοκιμών. Χρησιμοποιεί ένα σύνολο λέξεων - κλειδιών. Τα πιο γνωστά από αυτά είναι το τρίδυμο Given-When-Then που χρησιμοποιείται για να περιγράψει κάθε σενάριο δοκιμής.

```
@LoginFeature
Feature: Login

  Background:
    Given I visit the "Decolight" page

  Scenario Outline: Login
    When I click the "Σύνδεση"
    And I enter <email> as email
    And I enter <password> as password
    When I click the "Σύνδεση" button
    Then I am <status> and the account used is <email>
    When I click "Αποσύνδεση" if <status> is logged in
    Then I am redirected to "Log out" page

    Examples:
      | email | password | status |
      | kons.kaffe@gmail.com | konstantina123 | logged in |
      | konakaffe@yahoo.com | konstantina1234 | logged in |
      | konakaffe@yahouu.com | drggttrrrttr | not logged in |
```

Cucumber - Step Definitions

Οι κλάσεις step definition είναι κλάσεις Java που περιέχουν μεθόδους που δίνουν λειτουργικότητα στα σενάρια γραμμένα σε γλώσσα Gherkin. Κάθε βήμα (ένα βήμα ή και περισσότερα) στο Gherkin αντιστοιχίζεται σε ένα step definition που με τη σειρά του περιέχει κώδικα που εκτελείται όταν εκτελείται το σενάριο. Όταν το Cucumber εκτελέσει ένα βήμα Gherkin σε ένα σενάριο, θα αναζητήσει έναν αντίστοιχο ορισμό βημάτων (Step Definitions) για εκτέλεση.

```java
//Steps Implementation
public LoginSteps() {
    Given( expression: "^I visit the \"([^\"]*)\" page$", (String arg0) -> {
        openPage(arg0);
    });
    When( expression: "^I click the \"([^\"]*)\"$", (String arg0) -> {
        clickToLogin();
    });
    And( expression: "^I enter (.*) as email$", (String arg0) -> {
        addEmail(arg0);
    });
    And( expression: "^I enter (.*) as password$", (String arg0) -> {
        addPassword(arg0);
    });
    When( expression: "^I click the \"([^\"]*)\" button$", (String arg0) -> {
        clickLogin();
    });
    Then( expression: "^I am (.*) and the account used is (.*)$", (String arg0,String arg1) -> {
        assertLogIn(arg0, arg1);
    });
    When( expression: "^I click \"([^\"]*)\" if (.*) is logged in$", (String arg0, String arg1) -> {
        clickToLogout(arg1);
    });
    Then( expression: "^I am redirected to \"([^\"]*)\" page$", (String arg0) -> {
        assertIfLoggedOut(arg0);
    });
}
```

Cucumber - Hooks

Το Cucumber υποστηρίζει Hooks, τα οποία είναι blocks of code, που εκτελούνται πριν ή μετά από κάθε σενάριο. Μπορούν να οριστούν οπουδήποτε στο έργο, χρησιμοποιώντας τις μεθόδους @Before και @After. Τα Cucumber Hooks μας επιτρέπουν να διαχειριζόμαστε καλύτερα τη ροή εργασίας του κώδικα και μας βοηθά να μειώσουμε τον περιττό κώδικα.

```
public static WebDriver driver;
public static WebDriverWait wait;

//Setup
@Before("@LoginFeature")
public void browserIsOpen(){
    System.setProperty("webdriver.chrome.driver", "C:\\AutomationTest\\chromedriver.exe");
    driver = new ChromeDriver();
    driver.manage().window().maximize();
    driver.manage().timeouts().implicitlyWait( time: 5, TimeUnit.SECONDS);
    wait = new WebDriverWait(driver, timeOutInSeconds: 3);
}

@After("@LoginFeature")
public void destroy() { driver.quit(); }
```

Cucumber - Test Runner class, Cucumber Options

Το Cucumber προκειμένου να εκτελέσει τα δοκιμαστικά σενάρια χρησιμοποιεί το framework Junit. Καθώς χρησιμοποιείται το Junit, για την εκτέλεση των δοκιμαστικών σεναρίων πρέπει να δημιουργήσουμε μία Test Runner κλάση. Αυτή η κλάση χρησιμοποιεί το Junit annotation @RunWith(), που λέει στο Junit ότι αυτή είναι η test runner κλάση. Το @CucumberOptions είναι σαν property/settings αρχεία για τις δοκιμές. Για παράδειγμα μπορούμε να καθορίσουμε τη διαδρομή προς τα features files, τη διαδρομή προς τα step definitions κλπ.

```
package cucumberdecolight.tools;

import cucumber.api.CucumberOptions;
import cucumber.api.junit.Cucumber;
import org.junit.runner.RunWith;

@RunWith(Cucumber.class)
@CucumberOptions(
    features = "src\\test\\resources\\cucumberdecolight\\features",
    plugin = {"html:target/cucumber-html-report", "json:target/cucumber.json", "pretty:target/cucumber-pretty.txt",
              "usage:target/cucumber-usage.json", "junit:target/cucumber-results.xml"},
   glue = {"cucumberdecolight.steps"}
    )
public class CucumberRunner {}
```

Cucumber - Reporting

Το Cucumber χρησιμοποιεί reporter plugins για να παράγει αναφορές που περιέχουν πληροφορίες σχετικά με τα σενάρια που πέρασαν ή απέτυχαν. Ορισμένα plugins είναι ενσωματωμένα, άλλα πρέπει να εγκατασταθούν ξεχωριστά. Στην παρούσα εργασία δημιουργήσαμε μόνοι μας ένα historical report αντλώντας πληροφορίες από το auto-generated cucumber report. Το historical report περιλαμβάνει έναν πίνακα με όλα τα passed/failed scenarios/steps, την διάρκεια εκτέλεσης τους, καθώς επίσης και την ημερομηνία/ώρα που εκτελέστηκαν οι δοκιμαστικοί έλεγχοι, για κάθε φορά που εκτελέστηκαν αυτοί. Για την δημιουργία του historical report, δημιουργήθηκε ένα java εκτελέσιμο εργαλείο, το οποίο με τη χρήση της βιβλιοθήκης Jsoup, ανοίγει και διαβάζει τα αποτελέσματα των δοκιμών από το auto-generated report του cucumber. Στη συνέχεια, προσθέτει τη συγκεκριμένη πληροφορία στο συγκεντρωτικό report που βρίσκεται στο package ‘reporting-history’. Για τη δημιουργία αυτού του historical report, χρησιμοποιήθηκαν οι βιβλιοθήκες Bootstrap, JQuery, καθώς επίσης για την μορφή της εμφάνισης του έγινε συγγραφή της απαραίτητης CSS.

Ευχαριστώ πολύ!