Eina de generació assistida de proves automatitzades per a webs

Artal Laudo Salvà

Resum— La creació de codi de prova automatitzada per a webs és una tasca repetitiva i en molts casos no és viable d'implementar a causa dels costos de manteniment derivats de la variabilitat de les pàgines web. L'objectiu d'aquest projecte és aconseguir automatitzar la creació del codi de prova automatitzada en Selenium WebDriver mitjançant una aplicació per a escriptori desenvolupada en Java que permeti reduir el cost de la creació i manteniment de les proves funcionals automatitzades per als formularis web. A l'article es detalla com s'ha desenvolupat una eina orientada a assistir l'escriptura del codi de proves automatitzades per a fer proves funcionals dels formularis web. L'eina permet analitzar una pàgina a partir del seu URL per a identificar automàticament els camps dels formularis de la web, posteriorment crear casos de test assignant valors a cada camp, i finalment crear el codi de les proves automatitzades.

Paraules clau— Prova automatitzada, proves funcionals, casos de prova, web scraping, Selenium WebDriver

Abstract— Creating automated test code for websites is a repetitive task and in many cases is not feasible to implement due to the maintenance costs resulting from the variability of web pages. The goal of this project is to automate the creation of automated test code in Selenium WebDriver using a Java-based desktop application that will reduce the cost of creating and maintaining automated functional tests for web forms. The article details how a tool designed to assist automated test code writing has been developed for functional testing of web forms. The tool allows you to analyze a page from its URL to automatically identify the fields of the web forms, then create test cases by assigning values to each field, and finally create the code of the automated tests.

Index Terms— Automated testing, functional testing, test cases, web scraping, Selenium WebDriver

1 Introducció

AQUEST any passat l'empresa *IN2 Ingeniería de la Información*, que es dedica, entre altres coses a desenvolupar i mantenir pàgines web, ha aplicat *Kaizen*. El mètode *Kaizen*, amb origen japonès, serveix per a analitzar les metodologies de treball dels diferents equips i identificar els diferents punts on es pot millorar el rendiment. S'ha identificat que a la majoria dels equips es fa poc test i pràcticament totes les proves que es realitzen són proves manuals de validació per a cada tasca i només es fan proves de regressió abans d'entregar el producte o en alguns casos en finalitzar cada *Sprint* del projecte, depenent de la metodologia que aplica cada equip. Els equips no veuen viable aplicar proves de regressió automatitzades per les constants modificacions a la que se sotmeten algunes pàgines i pel cost que suposa aprendre a fer servir una eina de prova automatitzada del mercat, principalment degut a l'elevat temps de manteniment que requereix i afegir-la dins de la metodologia de treball de l'equip. L'empresa vol evitar haver d'obtenir una llicència de pagament d'una eina del mercat.

Un dels objectius principals de l'empresa és unificar les metodologies dels diferents equips, ja que actualment cada equip treballa amb una metodologia diferent, de manera que quan una persona canvia d'equip requereix d'un elevat temps d'adaptació. Es vol aprofitar aquesta unificació de metodologies i el desenvolupament d'aquest projecte per a introduir de manera formal les proves automatitzades a tots els equips de l'empresa. D'aquesta manera es podran identificar els defectes en les fases inicials de desenvolupament, abaratir els costos, reduir temps de desenvolupament i millorar la qualitat dels productes entregats al client.

L'article està estructurat en 8 seccions incloent la introducció. En primer lloc es presenten els objectius, es contextualitza el projecte a l'estat de l'art, i s'explica la metodologia que s'ha seguit durant el transcurs del projecte. A continuació es detalla el procediment de desenvolupament del

- *E-mail de contacte: artal.laudo@e-campus.uab.cat*
- *Menció realitzada: Tecnologies de la Informació.*
- *Treball tutoritzat per: Rubén Rubio Barrera (DEIC)*
- *Curs 2019/20*

projecte i un exemple d'ús de l'aplicació. Per acabar es fa una crítica del treball i es comenten les conclusions que s'han extret de la realització del treball. Finalment es veuen els agraïments seguits de la llista de les referències i la bibliografia consultada.

2 Objectius

L'objectiu del treball és aconseguir desenvolupar una eina que permeti assistir la creació del codi de proves automatitzades. A més a més aquesta eina ha de poder ser utilitzada per gent sense coneixements elevats en l'eina que s'ha emprat per a desenvolupar les proves automatitzades i no ha de requerir aprenentatge per a fer-la servir.

A continuació es llisten les tasques que s'han plantejat per al desenvolupament del projecte:

1. Investigar i analitzar les tecnologies de prova automatitzada.
2. Identificar els formularis d'una pàgina web a partir d'un URL.
3. Crear casos de prova i fer l'assignació de valors als camps.
4. Persistir les dades.
5. Exportar el codi de la prova automatitzada.
6. Crear una interfície d'usuari per a ordinadors amb Microsoft Windows.
7. Permetre la configuració del programa.
8. Generar informes de resultats de les proves automatitzades. (Opcional)
9. Permetre la importació i exportació de casos de prova mitjançant fulls de càlcul. (Opcional)

3 Estat de l'art

Aquest apartat està dividit en 3 seccions on es mostra l'estat de l'art de les proves, les eines per a facilitar-ne la tasca de proves, els diferents tipus de selectors i les diferents interfícies d'usuari.

3.1 Proves i els seus tipus

Les proves del software estan presents d'alguna manera a totes les metodologies de desenvolupament, permetent detectar possibles errors causats per la introducció de defectes durant qualsevol fase a qualsevol producte derivat del desenvolupament.

Com més aviat es realitzen les proves, el defecte s'haurà propagat menys a la resta del producte i per tant serà menys costós realitzar la correcció pertinent. Realitzar proves a un producte no garanteix la seva correctesa, només fa menys probable que s'hagin pogut passar per alt defectes en el producte que puguin causar errors. Realitzar més proves només permet que es trobin una major quantitat d'errors. Encara que sempre es puguin realitzar més proves, és impossible realitzar unes proves exhaustives (excepte en casos específics), de manera que habitualment les proves se centren a cobrir una quantitat més gran de casos diferenciats mitjançant un seguit de metodologies i bones pràctiques.

D'entre les proves que es poden realitzar al software, s'identifiquen:

- Les proves funcionals comproven que el sistema fa el que s'ha definit als requisits funcionals del producte.
- Les proves no funcionals comproven com de bé es comporta el sistema en diferents aspectes, com ara la usabilitat, la compatibilitat, el rendiment, la seguretat i les bases de dades.
- Les proves de caixa blanca es basen en la implementació del sistema i procuren cobrir un major percentatge de les sentències, decisions i condicions del codi entre d'altres mètriques.
- Les proves relacionades amb canvis ja siguin proves de confirmació per a confirmar que s'ha implementat correctament el canvi, o bé proves de regressió per a assegurar que la resta del codi conserva el seu correcte funcionament [1].

Aquest projecte tracta l'exportació del codi equivalent amb Selenium WebDriver en Java de proves automatitzades de formularis de pàgines web, de manera que està orientat a les proves funcionals, de validació i de regressió.

3.2 Eines per a facilitar proves

D'entre les eines més similars a la que es desenvolupa al treball hi ha Ranorex Studio Test Automation, LambdaTest, Sauce Labs i altres, moltes d'aquestes implementen Selenium Web Driver per a les proves automatitzades [2]. La majoria d'aquestes aplicacions requereixen l'ús d'una llicència de pagament i són complexes de fer servir i d'implementar al fluxe de treball a causa de la quantitat de funcionalitats que incorporen.

Selenium Web Driver és un framework d'automatització web de codi obert, multiplataforma, multinavegador i programable en diversos llenguatges, entre els quals hi ha Ruby, JavaScript, Java, Python i C# [3]. Des del juny de 2018 l'ús de Selenium WebDriver per a les proves de pàgines web és una recomanació del *World Wide Web Consortium* (W3C), una comunitat internacional que treballa per a desenvolupar i promocionar estàndards web [4].

Els components de Selenium WebDriver són: el WebDriver, el controlador del navegador i el navegador. Els

WebDrivers són una interfície de programació que es pot comunicar amb un controlador específic del navegador per a poder realitzar accions al navegador. Selenium WebDriver no inclou els WebDrivers específics de cada navegador, ja que aquests varien segons la versió del navegador, i per tant cal descarregar-los a part i referenciar-los en el codi.

Existeixen diverses configuracions possibles segons el tipus de comunicació que s'estableix entre cada component, com ara: tots els components al mateix sistema; amb una comunicació remota mitjançant un servidor Selenium o un WebDriver remot; o bé mitjançant un framework extern que es comuniqui amb un servidor Selenium o un Grid Selenium [5].

El WebDriver obre una finestra de navegador real, per tant no és una bona eina per a fer test de rendiment, ja que cada finestra del navegador té un consum de recursos elevat, és per això que es recomanen altres eines per a fer aquests tipus de proves.

Hi ha altres projectes de Selenium com ara Selenium IDE, una extensió per a navegador capaç de registrar i reproduir un conjunt d'interaccions a una pàgina; i Selenium Grid, un servidor que permet executar proves de Selenium WebDriver en paral·lel a diverses màquines, amb l'objectiu de reduir els temps d'execució de conjunts de proves [6].

Existeixen eines que permeten fer proves d'altres tipus com ara JMeter [7], un projecte Apache programat en Java, especialment orientat a proves de rendiment, o SoapUI [8], més orientat a proves d'integració, rendiment i seguretat.

En aquest treball es farà servir Selenium WebDriver com a motor, procurant desenvolupar una eina que funcionalment tingui un ús intuitiu com Selenium IDE, però que permeti obtenir el codi Java de proves automatitzades equivalent a Selenium WebDriver.

3.3 Selectors

Selenium WebDriver permet accedir als elements del llenguatge de marques d'hipertext (HTML) mitjançant selectors als quals es poden realitzar diferents tipus d'accions, entre les més habituals hi ha l'acció de fer clic i l'acció d'introduir text. Existeixen els següents tipus de selectors [9]:

- Selector per nom de la classe: Permet seleccionar els elements que coincideixen amb una classe HTML.
- Selector per fulls d'estils en cascada (CSS): Permet seleccionar els elements que coincideixen amb un selector CSS.
- Selector per identificador: Permet seleccionar l'element que coincideix amb un identificador HTML.
- Selector per nom: Permet seleccionar els elements que coincideixen amb un nom HTML.
- Selector per valor de l'enllaç: Permet seleccionar els elements que tenen un enllaç HTML igual al valor definit.
- Selector per valor parcial de l'enllaç: Permet seleccionar els elements que tenen un enllaç HTML que conté el valor definit.
- Selector per etiqueta HTML: Permet seleccionar els elements que coincideixen amb l'etiqueta HTML.
- Selector per llenguatge de camins de llenguatge de marques extensible (XPath): Permet seleccionar els elements que coincideixen amb un XPath.

3.4 Interfícies d'usuari

Sun Microsystems va desenvolupar JavaFX el 2008, i el 2009 va ser adquirida per Oracle. Des de llavors JavaFX ha estat el framework d'interfície d'usuari de referència per a desenvolupar en Java, substituint a Java Swing, també desenvolupat per Sun Microsystems el 1996.

4 METODOLOGIA

Aquest apartat està dividit en 3 seccions on es mostren la metodologia, la planificació de tasques i les observacions.

4.1 Metodologia

Per a desenvolupar el projecte s'ha optat per fer servir una metodologia de tipus àgil en espiral, per a aconseguir adaptar el projecte a les situacions imprevistes i una gestió més flexible. L'objectiu de basar la metodologia emprada en el projecte a metodologies àgils ha estat aconseguir obtenir els principals avantatges d'aquestes, com ara millorar la qualitat del producte, eliminar característiques innecessàries, reducció dels riscos i una detecció ràpida dels errors. [10]

Per al projecte s'ha volgut fer servir algunes característiques i bones pràctiques de Scrum [11] i altres de *Extreme Programming* (XP) [12], adaptant la metodologia al context del projecte, on només hi ha un sol desenvolupador i hi ha limitacions en la possibilitat de sol·licitar reunions.

De la metodologia Scrum s'ha volgut fer servir un seguiment per Sprints, on en cada període de unes dues setmanes s'han fet revisions informals de l'evolució del projecte i acompliment dels objectius planificats. S'ha creat una pissarra de Trello amb totes les tasques per a dur el seguiment del projecte. Els rols de desenvolupador, Product Owner i Scrum Master els ha assumit l'estudiant; mentre que el tutor d'empresa i el tutor del treball han assumit el rol de Stakeholder.

Respecte a XP s'ha procurat realitzar les bones pràctiques

amb el codi, realitzant revisions del codi per a deixar-lo simplificat, llegible i mantenible, realitzant millores de forma continuada a funcionalitats desenvolupades amb anterioritat.

4.2 Tasques a realitzar i planificació

A continuació s'aprofundeix en cada tasca que s'ha plantejat per al desenvolupament del projecte.

4.2.1 Investigar i analitzar les tecnologies de prova automatitzada

S'ha d'investigar les diferents tecnologies de prova automatitzada i les seves característiques, les possibilitats que ofereixen i la personalització que permeten per a adaptarse a les necessitats que puguin sorgir a l'empresa i les necessitats de l'eina, així com s'ha de definir les tecnologies i arquitectura que es faran servir.

4.2.2 Identificar els formularis d'una pàgina web a partir de l'URL

L'eina ha de poder fer Web Scraping a partir del Localitzador Uniforme de Recursos (URL) per a identificar els formularis i els seus camps, i guardant-ne les dades rellevants segons el tipus de camp. Ha de permetre identificar cada camp de manera única per a poder tornar a identificar l'element mitjançant l'eina seleccionada per a les proves automatitzades. L'aplicació no ha de requerir que l'usuari hagi de conèixer cap dada més enllà de l'URL.

S'ha començat desenvolupant el *Web Scraping* amb la llibreria jsoup de Java, però finalment s'ha decidit fer servir l'eina Selenium WebDriver per a fer el Web Scraping, permetent facilitar la tasca d'exportació del codi automatitzat. Inicialment es va plantejar fer que cada camp s'identifiqués amb selectors CSS o XPath que l'usuari pogués editar, finalment s'han fet servir selectors per etiqueta i nom. Aquests canvis han fet que hi hagués un endarreriment en la planificació però alhora han permés facilitar el desenvolupament de les següents tasques.

4.2.3 Crear de casos de prova i fer l'assignació de valors als camps

L'eina ha de permetre a l'usuari afegir cada cas de prova que vulgui crear, assignant els valors dels camps de manera semblant a la que es faria en la interfície d'una pàgina web. Ha de permetre afegir informació addicional com el nom del cas de prova.

4.2.4 Persistir les dades

L'eina ha de permetre que l'aplicació desi els formularis, casos de prova i configuració a la màquina de manera transparent a l'usuari. Aquest sistema de persistència es farà en un fitxer local.

4.2.5 Exportar el codi de la prova automatitzada

L'eina ha de permetre exportar el codi de cada cas de prova en Selenium WebDriver en llenguatge Java a un fitxer .java que es pugui afegir a un projecte. Aquest codi haurà de reomplir el formulari segons els valors assignats al cas de prova.

4.2.6 Crear una interfície d'usuari per a ordinadors amb Microsoft Windows

Es vol crear una interfície d'usuari per a ordinadors amb Microsoft Windows, ja que és el tipus d'equip amb el que treballen tots els empleats de l'empresa. La interfície ha de ser senzilla i ha de permetre que l'usuari entengui com ha de fer cada acció sense que se li hagi d'explicar el funcionament de l'eina. Es vol crear una interfície adequada als estils actuals, orientada a Material Design [13]. Inicilment es va plantejar l'ús de JavaFX, però degut als endarreriments s'ha acabat optant per l'ús de Java Swing, per a accelerar el desenvolupament de la interfície.

4.2.7 Permetre la configuració del programa

El programa ha de permetre a l'usuari seleccionar diverses opcions de configuració, com ara la carpeta on té guardat el Web Driver de cada navegador, per a quins navegadors vol tenir les proves automatitzades. Inicialment es volia afegir la possibilitat de configurar l'idioma de l'aplicació, implicant la pertinent internacionalització i localització de tots els camps de text de la interfície, però no s'ha pogut afegir per culpa de les limitacions de temps.

4.2.8 Generar informes de resultats de les proves automatitzades

Com a objectiu opcional afegit a la generació del codi de la prova automatitzada, es vol generar un informe de resultats per a cada cas de prova. En un inici es volia mostrar la informació del cas de prova i una captura del navegador per a mostrar el resultat, finalment només es desa una captura del navegador.

4.2.9 Permetre la importació i exportació de casos de prova mitjançant fulls de càlcul

Es volia oferir la possibilitat de gestionar els casos de prova mitjançant fulls de càlcul, permetent extreure i importar casos de prova. permetent realitzar modificacions més fàcilment als casos de prova en cas de que es modifiquin els elements del formulari. Aquesta funcionalitat no s'ha pogut desenvolupar per culpa dels endarreriments.

4.3 Observacions

Es valora positivament la metodologia emprada, ja que s'ha aconseguit que el codi obtingut sigui net, seguint els principis de *Clean Code* [14]. L'ús d'XP ha permès que les funcionalitats desenvolupades en etapes anteriors millorin per a facilitar el desenvolupament de les següents tasques.

Durant el desenvolupament del projecte s'ha detectat que

el fet de no haver dividit les tasques més grans en altres subtasques més petites ha causat que molts cops fos complicat detectar els endarreriments en la planificació fins que no s'ha completat el temps calculat per a la realització de la tasca. Algunes tasques llargues que s'han endarrerit, especialment a les últimes fases del projecte, han causat que no hi hagués una velocitat de reacció prou ràpida davant les inconveniències. Amb unes tasques més curtes s'hauria pogut planificar millor l'evolució real de les tasques més llargues i detectar possibles endarreriments en la planificació per a poder prendre mesures abans d'acabar les tasques que s'han allargat respecte a la planificació.

5 DESENVOLUPAMENT

En aquest apartat es mostren els detalls del desenvolupament de les diferents tasques.

5.1 Arquitectura de l'aplicació

S'ha implementat l'arquitectura de Selenium WebDriver de tenir tots els components al mateix sistema degut a que en molts casos es treballa amb webs que només estan accessibles des de una Xarxa Privada Virtual (VPN), i cada client en té almenys una. Per tant, es realitzarà una aplicació que pugui ser executada a un ordinador amb Windows 10, l'entorn de treball que fan servir els empleats de l'empresa i des d'on ja es connecten a les VPN.

5.2 Identificació dels formularis a partir de l'URL

Per a identificar els formularis a partir de l'URL és necessari fer servir un selector per a detectar els camps, depenent del tipus de selector que es faci servir hi haurà una major tendència a haver d'anar canviant els selectors, per tant, és una de les decisions més importants per a l'aplicació.

Els selectors per enllaç i per enllaç parcial no tenen cap cabuda en les proves de formularis. Els selectors per classe, els selectors CSS i els selectors per XPath tenen una tendència força gran a canvis un cop es modifiquen estils de la pàgina, de manera que són un tipus de selectors poc aconsellats per a l'ús que volem donar, que està orientat a funcionalitat. Aquests selectors poden ser més útils per a eines orientades a estils, per exemple per a identificar si un element amb una classe està tenint sobreescrits uns altres estils per una altra regla CSS.

D'altra banda els selectors per identificador són perfectes per a pàgines creades pensant en les proves, inclús són la recomanació per part de Selenium, ja que l'identificador és un camp que rarament es modifica, però moltes vegades els elements no tenen un identificador, i no és viable modificar totes les pàgines web que es vulguin provar per a adaptar-les a aquest selector. No es descarta que en un futur l'ús d'identificadors als elements sigui una pràctica habitual i aquest sigui el sistema més indicat per a seleccionar un element.

Els sistemes selectors mitjançant la classe, mitjançant el CSS, mitjançant el XPath i mitjançant l'identificador tenen el desavantatge de requerir que l'usuari (habitualment dedicat a realitzar proves) necessiti conèixer detalls d'implementació de cada canvi en els estils que pugui causar que els selectors deixin de funcionar.

Finalment, els selectors per etiqueta i nom són els selectors ideals per a la tasca a desenvolupar en aquest projecte, ja que les etiquetes dels formularis són fàcilment identificables i degut a que el nom es fa servir per a rebre els valors al servidor, aquest és un camp que presenta poca variabilitat. Selenium adverteix de que aquest és un camp que pot estar present varies vegades en un formulari. Dos exemples clars relacionats amb formularis poden ser el cas de dos formularis, un de registre i un d'identificació, que puguin tenir el mateix nom per al camp d'usuari o de contrasenya, o bé el cas d'un camp de tipus radio. Per a resoldre el primer cas es fa primer una cerca per formulari i després per els seus camps, per a resoldre el segon cas es recorre el conjunt de camps amb el mateix nom i es diferència pel camp de valor.

Inicialment al projecte es va plantejar la possibilitat de requerir que l'usuari introduís els selectors CSS o XPath de cada element dels formularis, però mitjançant els selectors per etiqueta i nom aquesta funcionalitat s'ha pogut automatitzar completament. Tot i que cal tenir en compte que hi ha casos excepcionals on els camps del formulari no es carreguen fins que no es doni una circumstància en concret, i aquests no seran processats per el WebDriver, com per exemple casos on una part de formulari es carregui per ajax després de prémer un botó. Aquest aspecte no s'ha desenvolupat degut a la complexitat que pot tenir cobrir tots els casos excepcionals. Depenent del tipus d'element d'entrada es guarden diferents dades a la clase pertinent, com ara el valor o si està clicat, que poden variar depenent del tipus de camp. No s'han afegit tots els tipus de camps possibles per a reduir el volum de treball, tot i que el procediment a seguir és equivalent.

El programa és capaç de detectar els formularis i camps que es troben a una pàgina a partir de l'URL, tant durant la càrrega de la pàgina com en un altre instant indicat per l'usuari. Tot i això l'automatització de la captura està subjecte a la inexistència de camps o conjunts d'accions que no es puguin automatitzar, com ara les Proves de Turing Públiques i Automàtiques per a diferenciar a Màquines d'Humans (CAPTCHA) o l'execució d'esdeveniments JavaScript que generin codi de formulari HTML.

Degut a que el programa s'executa a la banda del usuari, aquest pot accedir a les mateixes pàgines que l'usuari podria accedir mitjançant el seu navegador. Això permet que es pugui accedir als URL que requereixin estar connectat a la VPN únicament requerint realitzar la connexió des de la màquina de l'usuari sense configuracions addicionals. Aquest punt es considera rellevant ja que un desenvolupament que executés aquest programa en remot hagués requerit un apartat de configuració de connexió a la VPN per

a poder accedir a pàgines allotjades a les VPN externes de cada client.

5.3 Creació de casos de prova

Per a generar cada cas de prova en un inici s'havia pensat en crear els formularis a la interfície d'usuari i que l'usuari els reomplís allà, fent que la introducció dels casos de prova es fes directament al programa. Això afegia una complexitat molt més gran a l'hora d'implementar la interfície d'usuari, ja que obligava a generar un formulari completament adaptable, així que es va plantejar si podia existir una alternativa viable que ho pogués evitar.

Finalment s'ha optat per carregar la pàgina en qüestió al navegador mitjançant el WebDriver i oferir a la interfície un botó per a escanejar tots els camps del formulari i guardar-ne els valors. Amb aquesta solució s'aconsegueix que es puguin guardar alguns formularis amb camps generats a partir d'esdeveniments JavaScript provocats per reomplir els camps anteriors del formulari. I també permet fer un escaneig dels formularis d'un formulari dividit en parts que es carreguin mitjançant JavaScript. S'ha hagut de tornar a implementar aquesta part de nou, provocant que el temps dedicat fos major a l'estimació inicial, però obtenint un resultat semblant a la implementació de Selenium IDE.

El desenvolupament de creació de casos de prova s'ha modificat a la fase final per a adaptar-se a una solució que ofereix la possibilitat d'introduir les dades al formulari de la pàgina enlloc de fer-ho a la interfície. Aquest desenvolupament es va haver de realitzar degut a que durant la codificació de la interfície es va comprovar que la possibilitat de crear una interfície adaptable a tots els camps del formulari era una tasca molt més complicada del que s'havia calculat inicialment. Una interfície adaptable a tots els camps del formulari requereix posicionar correctament cada camp, mostrar-lo segons el seu tipus i fer que l'acció del botó "Create Test Case" agafi tots els camps en l'acció.

L'alternativa que s'ha fet servir consisteix en executar un escaneig de la pàgina URL un cop l'usuari ha introduït les dades als camps del formulari, i aquest interpreta els camps de la pàgina. D'aquesta manera s'obté la possibilitat de permetre que alguns formularis generats per esdeveniments JavaScript que requereixin reomplir camps anteriors puguin ser desats i executats. El canvi no ha suposat un increment substancial de les hores ja que el desenvolupament plantejat ha permès aprofitar una part del desenvolupament previ.

Per a poder organitzar de forma més còmoda les proves creades s'ha generat una jerarquia amb pàgines i proves, on una pàgina consta d'un URL, d'un nom i d'un conjunt de proves; i una prova consta d'un nom i el seu conjunt de valors de cada camp.

5.4 Persistència de dades

S'ha decidit fer servir l'emmagatzematge local de l'usuari per a persistir les dades del programa, ja que no existeix la necessitat de compartir els casos de prova generats al programa a altres persones de l'empresa i a més a més el programa només genera el codi, així que el codi de prova automatitzada resultant es pot compartir al repositori GitLab de l'empresa. Per a emmagatzemar les dades de forma local s'ha implementat guardant l'objecte en un corrent de bytes, i carregant-lo a l'inici abans que la interfície d'usuari. Les classes del programa s'han estructurat de manera que només sigui necessari guardar i carregar la classe pare, de la que en depenen la resta.

El programa guarda totes les dades després de cada acció i al tancar el programa, de manera que si es finalitza mitjançant l'administrador de tasques els canvis queden guardats igualment. Les dades del programa es guarden en un fitxer que es crea automàticament quan es guarda per primer cop a la primera execució. Quan el programa s'executa primer comprova si el fitxer existeix i si aquest no existeix carrega l'aplicació com si fos la primera vegada que s'obre. La gestió de la persistència de dades queda completament transparent a l'usuari, que no cal que sàpiga que les dades del programa es guarden en un fitxer.

5.5 Exportació a codi de la prova automatitzada

Per a generar el codi de prova automatitzada s'ha fet servir una classe dedicada a generar la estructura bàsica comuna a tots els casos de prova amb modificacions condicionals segons la configuració i les proves, i també cada tipus de camp de formulari genera el seu propi conjunt de *Strings* per a escriure el codi equivalent per a omplir el formulari.

L'usuari haurà d'introduir manualment el codi de clic al botó que desitgi, ja que s'ha valorat que en moltes situacions hi pot haver més d'un botó de tipus *Submit*, o bé el botó que s'ha de clicar és un botó normal que crida a una funció JavaScript de validació i posteriorment clica el botó de tipus *Submit*. El codi que es genera no s'ha tabulat per a simplificar el desenvolupament d'aquesta part, ja que es considera una funcionalitat mínima i s'ha decidit que hi havia altres punts més rellevants per desenvolupar.

El codi automatitzat es genera en unes classes anomenades Test seguit d'un número incremental. Dins del fitxer generat hi ha un comentari que indica els detalls de la pàgina i del cas de prova i s'hi troba el codi de la prova automatitzada sense tabular. El codi de la prova automatitzada inclou la introducció de les variables als camps i la creació d'un fitxer d'imatge amb la captura de pantalla de la pàgina i el mateix nom de la classe, d'aquesta manera es pot crear fàcilment un recull de captures i resultats.

El procés de generació de codi de prova automatitzada es fa a nivell de pàgina, ja que s'ha valorat que generar el codi de prova en conjunts és més pràctic que haver-ho de fer per a cada cas. Per a crear un codi de prova automatitzada serà necessari que s'hagi configurat el programa, s'hagi creat una pàgina, s'hagi creat un cas de prova per a la pàgina amb els seus valors als camps, i es procedeixi a

generar el codi d'aquella pàgina.

5.6 Interfície d'usuari

Per a la interfície s'ha fet servir Java Swing degut a la simplicitat, claredat de documentació i guies, i degut al reduït marge de temps sobrant per a dedicar a la interfície. D'altra manera s'hagués optat per JavaFX, que era la plataforma pensada inicialment. Per a la interfície s'ha procurat simplificar al màxim possible la vista d'usuari per a tal de que es vegi clar que es pot fer en cada cas. S'ha creat una interfície que més endavant es pugui modificar fàcilment a l'estil *Material Design* i que alhora s'hi assembli el màxim possible. S'han pres les següents decisions de disseny:

- S'han evitat llistes de taules amb dades a la interfície i s'han fet servir camps desplegables al seu lloc.
- S'han reduït els canvis de finestra o vista per a permetre que l'usuari es familiaritzi més fàcilment amb l'eina i fer l'ús més intuïtiu.
- S'ha evitat tenir molts camps a completar per a cada acció, minimitzant el nombre de clics necessaris.
- S'han emprat colors blancs i grisos pastel sense els degradats per defecte que genera Java Swing.
- S'ha creat la interfície amb els textos en anglès perquè és un idioma que tothom de l'empresa coneix.
- S'ha creat la interfície amb pocs textos per a facilitar-ne la posterior internacionalització.

L'eina s'ha creat de manera que el seu ús sigui intuïtiu, deixant totes les accions possibles a la vista, de manera que l'usuari no hagi de buscar entre menús les diferents accions a realitzar.

5.7 Configuració del programa

S'ha desenvolupat una classe per a emmagatzemar la configuració del programa, on trobem la localització dels controladors dels navegadors i més endavant s'hi afegirà la selecció d'idioma.

Es pot editar la configuració per defecte per a que el programa pugui accedir als controladors pertinents, una acció important per al correcte funcionament del WebDriver, ja que en algunes actualitzacions de navegador els controladors necessiten actualitzar-se també. Dins la configuració també es pot escollir per a quins navegadors es volen crear els casos de prova, triant entre Google Chrome, Mozilla Firefox i Microsoft Internet Explorer. El programa es pot fer servir si es guarden els WebDrivers a la carpeta que està assignada per defecte, tot i que s'aconsella configurar-lo abans de començar a fer-ne ús.

5.8 Generar informes de resultats de les proves automatitzades

S'ha afegit a la generació del codi de la prova automatitzada una captura de pantalla del resultat final.

El programa genera el codi de prova automatitzada afegint una captura de pantalla al final de l'execució, tot i que finalment no s'ha pogut desenvolupar una documentació per text a causa de la falta de temps.

6 Exemple d'ús

A continuació es mostra una guia de com realitzar les diferents accions possibles a l'aplicació amb un exemple d'ús.

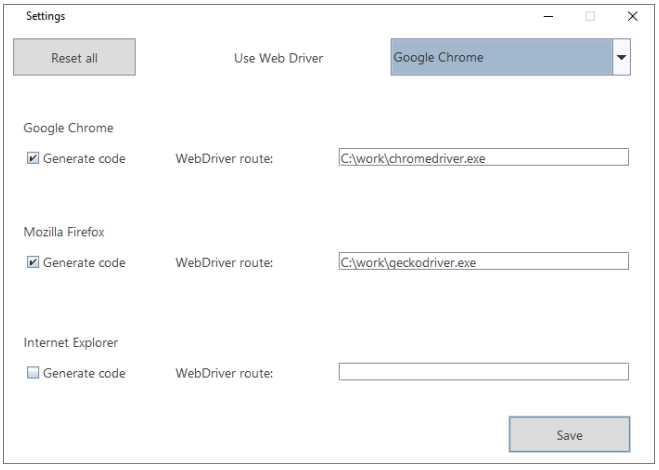

Figura 1. Captura de la interfície de configuració

Configurar l'aplicació:

1. Descarregar i descomprimir els Web Drivers dels navegadors que es vulguin fer servir, preferiblement fent servir els oficials de Selenium [15].
2. Prémer el botó "*Settings*" de la interfície principal del programa.
3. S'obre la interfície de configuració tal i com es mostra a la figura 1.
4. Seleccionar el navegador que es vol fer servir per a introduir els casos de prova al camp "*Use Web Driver*".
5. Seleccionar els navegadors als que es vol exportar el codi.
6. Afegir la ruta absoluta dels Web Drivers seleccionats.

7. Prémer el botó "*Save*" per a guardar els canvis.

8. Tancar la finestra de configuració per la creueta superior.

9. Es visualitza altre cop la interfície principal del programa tal i com es mostra a la figura 2.

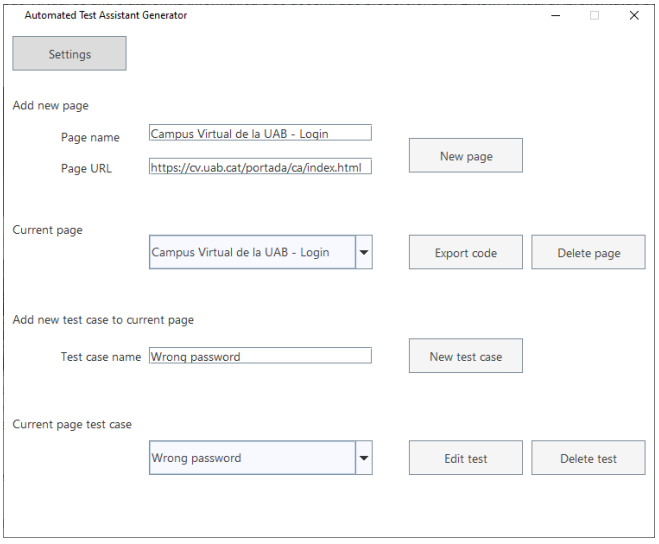

Figura 2. Captura de la interfície principal

Crear una pàgina:

1. Afegir el nom i l'URL de la pàgina als camps del formulari "*Add new page*". A l'exemple s'ha fet un cas de prova amb la página del Campus Virtual de l'Universitat Autònoma de Barcelona amb l'URL https://cv.uab.cat/portada/ca/index.html.

2. Prémer el botó "*New page*".

Crear un cas de prova:

1. Seleccionar la pàgina a la que es vol crear el cas de prova al camp "*Current page*".

2. Afegir el nom del nou cas d'ús al camp "*Test case name*". A l'exemple de la figura 2 s'ha creat el cas de prova "*Wrong password*", on es vol probar el cas de prova d'una contrassenya incorrecta.

3. Prémer el botó "*New test case*".

4. Seleccionar el cas de prova a "*Page test cases*".

5. Prémer el botó "*Edit test*".

6. S'obren dues finestres, en una s'obre el navegador amb la pàgina carregada, com es veu a la figura 3 (la barra blanca que hi apareix a la part de dalt anuncia "*Un software automatizado de pruebas está controlando Chrome*") i a l'altra s'obre una finestra de diàleg que diu "*Press OK to save changes*", com es veu a la figura 4.

7. Omplir el formulari de la pàgina que s'ha obert a la nova finestra de navegador segons el cas de prova que es vol realitzar.

8. Prémer el botó d'acceptar de la finestra de diàleg "*Press OK to save changes*".

9. Les dues finestres es tanquen i es mostra la interfície principal del programa.

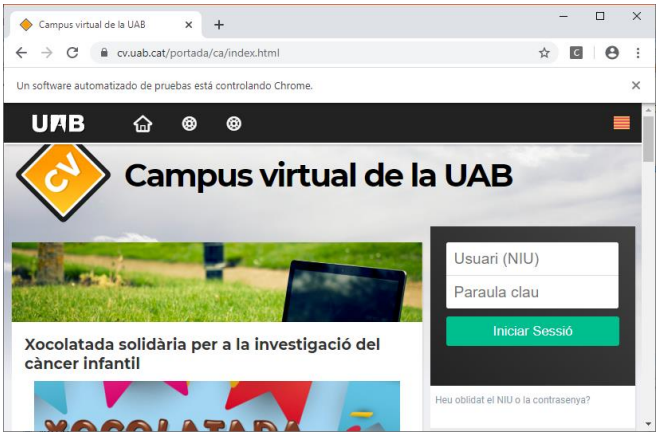

Figura 3. Captura de l'execució del navegador desde Selenium WebDriver

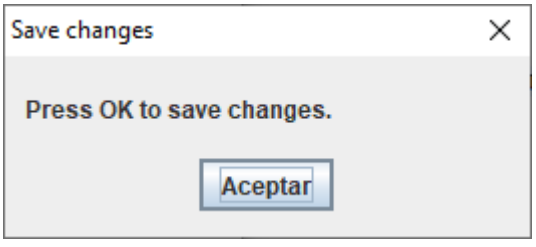

Figura 4. Captura del diàleg per a desar un cas de prova

Exportar els casos de prova a codi:

1. Seleccionar la pàgina de la que es vulgui exportar el codi al camp "*Current page*".

2. Prémer el botó "*Export code*".

3. A la mateixa carpeta on hi ha el programa s'han generat els fitxers amb nomenclatura "*test*" + número de cas de prova + navegador.

4. El codi es pot copiar a un projecte de qualsevol entorn integrat de desenvolupament (IDE) per a editar-lo i executar-lo.

A més a més de les funcionalitats que s'han anomenat, es poden eliminar les pàgines, els casos de prova i es pot resetejar tota l'aplicació com si fos la primera execució.

7 Conclusions

L'eina desenvolupada compleix els objectius plangejats, tot i no haver aconseguit desenvolupar les tasques opcionals, la qualitat de les tasques assolides és major del que en un principi s'havia plantejat. Finalment s'han assolit les 7 tasques principals que s'havien planejat, així com s'ha desenvolupat la generació de codi de captura de pantalla, una de les tasques opcionals, que s'ha assolit parcialment a causa dels endarreriments en el desenvolupament d'algunes tasques.

Selenium IDE ha presentat millores al llarg de l'any 2019 incorporant al programa la funcionalitat per a exportar les proves a C#, JavaScript, Java, Python i Ruby, de manera que l'aplicació desenvolupada en aquest projecte presenta poques millores respecte a la versió actual de Selenium IDE més enllà de la possibilitat de personalitzar el programa, mentre que Selenium IDE ofereix moltes més funcionalitats [16]. Aquest fet demostra que hi havia una necessitat de mercat per a realitzar el desenvolupament que s'ha portat a terme en aquest projecte.

Com que Selenium IDE ha desenvolupat la mateixa funcionalitat que l'objectiu principal del treball, els futurs desenvolupaments sobre l'eina seran a caràcter personal en lloc de necessitats laborals. Tot i això, encara hi ha alguns camps on l'eina que s'ha desenvolupat es podria millorar respecte a les funcionalitats que ofereix Selenium IDE, com ara, la possibilitat d'afegir als codis generats la funcionalitat de documentar el cas de prova i dels seus resultats, que es podria redactar seguint l'estàndard de Gherkin; importar i exportar els casos de prova en fulls de càlcul; i també es considera rellevant aplicar la internacionalització per a la interfície.

Agraïments

Vull agrair al Rubén Rubio Barrera, que ha tingut molta disponibilitat per a fer reunions i ha proposat millores que han permès que el projecte sigui de més qualitat. També vull agrair a tots els meus companys de l'empresa IN2 Ingeniería de la Información per tot el suport per a poder accedir a cursos, certificacions i reunions que han ofert informació sobre el funcionament de les proves a l'empresa i l'entorn laboral. Finalment m'agradaria fer una menció especial per a la Cristina Carceller Bosch i a tota la meva família per tot el suport durant aquests mesos.

Bibliografia

[1] International Software Testing Qualifications Board, "Certified Tester Foundation Level Syllabus 2018 Version", 2018. [Online] Available: https://www.istqb.org/downloads/send/2-foundation-level-documents/281-istqb-ctfl-syllabus-2018-v3-1.html, [Accesed: 19-Nov-2019]

[2] Selenium, "Ecosystem", 2019. [Online] Available: https://selenium.dev/ecosystem/, [Accessed: 27-Dec-2019]

[3] Selenium, "Installing Selenium Libraries", 2020. [Online] Available: https://selenium.dev/documentation/en/selenium_installation/installing_selenium_libraries/, [Accessed: 11-Jan-2020]

[4] Simon Stewart, David Burns, "WebDriver W3C Recommendation", 2018. [Online] Available: https://www.w3.org/TR/webdriver1/, [Accessed: 27-Dec-2019]

[5] Selenium, "Understanding the components", 2020. [Online] Available: https://selenium.dev/documentation/en/webdriver/understanding_the_components/, [Accessed: 11-Jan-2020]

[6] Selenium, "Selenium Projects", 2019. [Online] Available: https://selenium.dev/projects/, [Accessed: 27-Dec-2019]

[7] Apache Software Foundation, "Apache JMeter", 2019. [Online] Available: https://jmeter.apache.org/, [Accessed: 14-Sep-2019]

[8] Smartbear, "Powerful Automated Testing Tool for REST APIs", 2019. [Online] Available: https://www.soapui.org/professional/soapui-pro.html, [Accessed: 15-Sep-2019]

[9] Selenium, "Locating elements", 2020. [Online] Available: https://selenium.dev/documentation/en/getting_started_with_webdriver/locating_elements/ , [Accessed: 11 Jan - 2020]

[10] Mark C. Layton, "Ten Benefits of Agile Project Management", 2019. [Online] Available: https://www.dummies.com/careers/project-management/ten-benefits-of-agile-project-management/, [Accessed: 3-Jan-2020]

[11] Ken Schwaber, Jeff Sutherland, "The Scrum Guide", 2017. [Online] Available: https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf#zoom=100, [Accessed: 12-Sep-2019]

[12] Don Wells, "The Rules of Extreme Programming", 1999. [Online] Available: http://www.extremeprogramming.org/rules.html, [Accessed: 12-Sep-2019]

[13] Google, "Introduction – Material Design", 2019. [Online] Available: https://material.io/design/introduction/#principles, [Accessed: 06-Jan-2020]

[14] Robert Cecil Martin, "Clean Code A Handbook of Agile Software Craftmanship", 14th Edition, Massachusetts, Courier, 2015.

[15] Selenium, "Third party divers and plugins", 2020. [Online] Available: https://selenium.dev/documentation/en/getting_started_with_webdriver/third_party_drivers_and_plugins/, [Accessed: 11-Jan-2020]

[16] SeleniumHQ, "Tags · SeleniumHQ/selenium-ide · GitHub", 2019. [Online] Available: https://github.com/SeleniumHQ/selenium-ide/tags, [Accessed: 12-Jan-2020]