UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale in Ingegneria del software II

Un processo CI/CD con tecniche di generazione di test E2E robusti

Anno Accademico 2020/2021

relatore

Ch.mo Prof Porfirio Tramontana

candidato
Gianluca Talitro
matr. M63001001

Ringrazio il professore Porfirio Tramontana, per avermi seguito lungo la preparazione della tesi di laurea.

Ringrazio la mia ragazza Angela per avermi sempre supportato in questo lungo percorso.

Ringrazio la mia famiglia di avermi dato la possibilità di studiare e di arrivare fin qui.

Ringrazio i miei datori di lavoro, Giancarlo, Mario e Pasquale, e tutti i miei colleghi per avermi aiutato a consolidare le mie competenze e conoscenze.

Indice

Capitolo 1: Introduzione

1.1. Storia del testing

L'attività di testare software nasce con la prima riga di codice…

- Storicamente la parola testing venne associata per la prima volta in ambito informatico, in un paper del 1949 intitolato *"Checking a Large Routine"* scritto dal celebre padre dell'informatica Alan Turing. Questo documento fu presentato da Turing il 24 giugno 1949 alla conferenza inaugurale del computer EDSAC al Mathematical Laboratory di Cambridge, nel quale propone una prima risposta alla domanda su come si possa verificare la correttezza di una routine.

- Successivamente nel 1950, lo stesso Alan Turing scrisse il più celebre paper "Computing Machinery and Intelligence", introducendo il così detto "Test di Turing".
  Turing nel teorizzarlo si ispirò ad un gioco popolare, chiamato "gioco dell'imitazione", che consisteva nel avere tre partecipanti: un uomo A, una donna B, ed una terza persona C. La persona C, tramite una serie di domande doveva identificare chi fosse l'uomo e chi la donna, ovviamente le domande e le risposte dovevano essere dattiloscritte. La persona A aveva il compito di ingannare C, mentre B di aiutarlo.
  Il test di Turing si basa quindi sul gioco dell'imitazione, con il presupposto che una macchina si sostituisca alla persona A. In questo modo C deve identificare chi fosse l'uomo/donna e chi la macchina.
  Se la percentuale di volte in cui l'individuo C indovina chi è l'uomo e chi la donna è simile alla percentuale di volte in cui indovina chi è la macchina e chi l'essere umano, allora la macchina dovrebbe essere considerata intelligente dato che risulterebbe indistinguibile da un essere umano.

  - Nel 1957 Charles L. Baker ha distinto il testing di un programma dal suo debugging all'interno di una sua recensione del libro "Digital Computer Programming" di Dan McCraken, da allora l'attività del software testing ha assistito ad enormi cambiamenti.

- Il concetto di testing è sempre stato una parte importante e complessa del processo di sviluppo del software. Questo concetto è sempre stato dinamico, evolvendo continuamente ed assumendo sempre più caratteristiche. Una classificazione storica delle varie fasi evolutive del software testing è stata delineata nel 1988 da Dave Gelperin e William C. Hetzel. I ricercatori individuarono cinque distinti periodi storici all'interno dei quali variavano gli obiettivi del testing, distinsero:

- *Periodo orientato al debugging (1956):* i processi di verifica dei programmi, debugging e testing non erano ancora chiaramente differenziati.
- Periodo orientato alla dimostrazione (1957-1958): l'obiettivo del testing era esclusivamente assicurarsi che il software soddisfasse i suoi requisiti.
- Periodo orientato alla distruzione (1979-1982): l'obiettivo principale del testing era l'individuazione dei degli errori di implementazione che causassero il fallimento del software.
- Periodo orientato alla valutazione (1983-1987): l'obiettivo del testing era fornire valutazioni del prodotto durante il ciclo di vita del software.
- Periodo orientato alla prevenzione (dal 1988): l'obiettivo del testing era dimostrare che il software soddisfasse le sue specifiche, per rilevare e prevenire eventuali fallimenti.

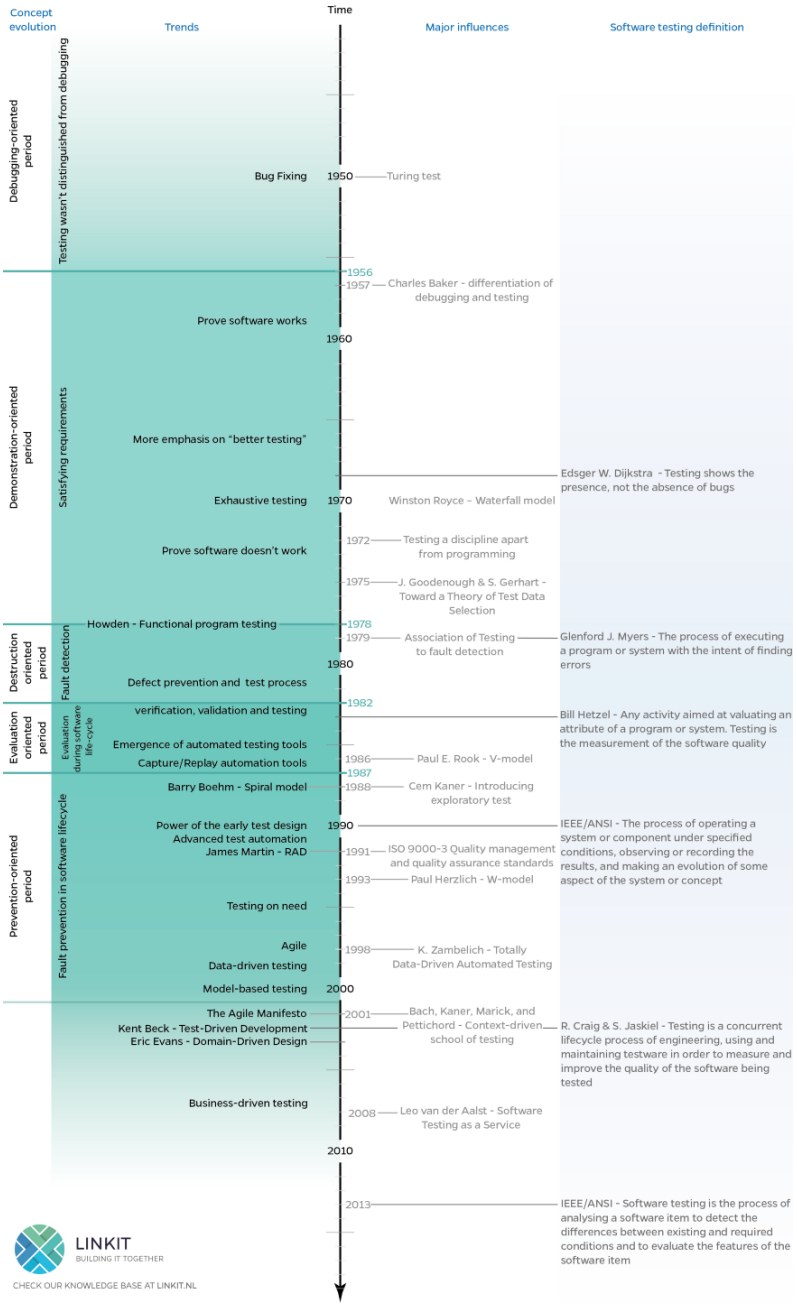


Figura 1.1 - 1: Software testing timeline

Alla luce delle nozioni precedentemente esposte ed a ulteriori citazioni, possiamo andare ulteriormente a definire le attitudini principali del software testing decade per decade:

- Nel corso degli anni 70’ la definizione di testing poteva essere ottimamente espressa da una citazione di Glenford J. Myers che definiva il testing come “il processo di esecuzione di un programma o di un sistema con l’intento di trovare degli errori”. Al tempo si considerava infatti la rilevazione degli errori come unico scopo dell’attività di testing.

- Negli anni 80' la definizione di testing si estese ed andò ad incorporare il concetto di valutazione della qualità del software. Per la prima volta il testing fu considerato come un processo manageriale che dovrebbe essere implementato attraverso l'intero ciclo di vita dello sviluppo software. Negli anni ottanta nacquero anche i primi strumenti di testing automatizzati con l'intento di automatizzare il testing manuale, incrementandone qualità ed efficienza.
- Negli anni 90' l'attività di testing è stata ridefinita come un'attività di pianificazione, progettazione, costruzione, manutenzione ed esecuzione di test all'interno di appositi "environments". In questi anni si è evidenziata l'importanza della progettazione iniziale dei test all'interno del ciclo di vita dei software, incorporando l'idea che un buon testing sia un processo gestito coerentemente con tutti gli altri aspetti dello sviluppo. In questi anni è emerso anche il metodo di testing agile, nato per far fronte ad una crisi di sviluppo software improduttivo causata dalla troppa rigidità delle obsolete precedenti metodologie aziendali.

1.2. Il testing oggi

Sin dall'inizio degli anni 00'sono nate nuove metodologie di sviluppo, come:

- Test-Driven Development (TDD): il TDD, sviluppo guidato dai test, è un modello di sviluppo del software che prevede che la stesura dei test automatici avvenga prima di quella del software che deve essere testato. In tal modo lo sviluppo del software è orientato esclusivamente al superamento dei test predisposti.
- Behaviour-Driven Development (BDD): il BDD, sviluppo guidato dal comportamento, è una metodologia di sviluppo che incoraggia la collaborazione tra developer, tester e rappresentanti di clienti all'interno di un progetto software. Il BDD combina le tecniche di TDD con i principi della DDD e della OOD (Object Oriented Design) per fornire, ai vari team aziendali, strumenti e processi condivisi per collaborare allo sviluppo del software.
- Domain-Driven Design (DDD): il DDD, progettazione guidata dal dominio, è una metodologia secondo la quale la struttura e l'implementazione di un qualsiasi software deve essere guidata dal suo dominio di riferimento. In questo modo le classi, i metodi e

qualsiasi altro elemento (diagrammi, schemi …) dovrà avere dei nomi specifici che siano coerenti con il loro dominio di appartenenza.

- Specification By Example (SBE): la SBE, specifica per esempio, è un approccio collaborativo per la definizione dei requisiti e dei test funzionali dei software. L'approccio è basato sull'utilizzo di esempi realistici anziché di nozioni astratte ed è applicato nel contesto delle metodologie di sviluppo agile.

le quali hanno rivoluzionato lo sviluppo tradizionale e conseguentemente il testing.
L'importanza del testing si è estesa ad ogni fase del ciclo di vita del software.
Sempre dall'inizio degli anni duemila gli strumenti di test automatizzati hanno attirato sempre più l'attenzione del mercato. Si diffusero così velocemente strumenti di automazione come JIRA (2002), Selenium (2004), Soap UI (2005), andando sempre più nella direzione di sostituire i test manuali (anche se ciò non sarà mai completamente realizzabile).

Oggi giorno si ricorre raramente al testing manuale, i test sono progettati in modo tale da essere rieseguibili in maniera automatica. Uno dei framework maggiormente utilizzati è JUnit, un framework per il linguaggio di programmazione Java che è utilizzato per la scrittura dei test di unità. JUnit è considerato di fondamentale importanza per lo sviluppo TDD dato che ci consente di scrivere i test di unità prima ancora che la classe venga implementata.
JUnit fa parte di una famiglia di framework per i test di unità, conosciuta col nome di xUnit che si differenzia nei vari framework in base al linguaggio di programmazione utilizzato.

Un altro approccio oggi molto utilizzato in ambito testing è il "Capture & Replay", una tecnica che consiste nell'utilizzare un tool automatico che registri una serie di operazione effettuate da un utente (fase di capture) e generi in output una sezione di codice (JUnit, o simili…) che sia in grado di replicare automaticamente le azioni rilevate durante la fase di capture.
Un tool di "Capture & Replay" molto utilizzato è "Katalon Recorder", utilizzato come estensione di Chrome è in grado di registrare e trasformare in codice una serie di azioni

che un utente intraprende all'interno di una pagina web. Katalon Recorder è quindi un Selenium IDE che ci consente di ottenere automaticamente script xUnit per automatizzare il testing di pagine web.

Un'altra metodologia di testing sempre più diffusa è il così detto "crowdtesting". Il crowdtesting (o crowdsourcing testing) è una pratica che sfrutta l'efficienza e la diversità del crowdsourcing per i test del software. Software come app mobile, applicazioni desktop native o siti Web in varie fasi di sviluppo vengono distribuiti a un ampio gruppo di persone. Le persone in questo gruppo, i crowdtester, eseguono il software sui propri computer o telefoni cellulari per ispezionarlo alla ricerca di difetti e aree di potenziale miglioramento. Il Crowd (in italiano "folla") Testing permette infatti di coinvolgere un numero molto alto di persone in tutto il mondo, che hanno a disposizione una grande varietà di device, configurazioni, sistemi operativi, reti, etc. In questo modo diventa molto semplice trovare problemi e difetti su di un prodotto software. L'utilizzo del Crowd Testing aiuta a rilevare aspetti funzionali, problemi di usabilità, di performance e di sicurezza, fondamentali per il corretto funzionamento dei prodotti software di una moderna azienda. Grazie al crowdtesting è quindi possibile identificare molto più velocemente difetti critici che sarebbero altrimenti troppo difficili, costosi, e lunghi da trovare con classici test.

1.3. Problematica Affrontata

L'obiettivo del seguente lavoro è quello di andare a realizzare uno strumento automatico, distribuito all'interno di un container Github Actions, che sia in grado di rieseguire dei test *Capture & Replay* su di una generica applicazione web in fase di sviluppo, generando dei report sull'esito delle test suite che si sono susseguite fra le varie versioni del software.

La tecnica di Capture & Replay presentata utilizza degli appositi locatori, detti hooks (ganci), per identificare in maniera più robusta gli elementi di una GUI di una web app. La tecnica di iniezione degli Hooks e di generazione dei test Capture & Replay che tiene conto dell'aggiunta dei locatori, è presa dal lavoro del Dott.Pierantonio Cangianiello (Pierantonio Cangianiello / test-hooks · GitLab).

Il progetto del dottor Cangianiello contiene un file main.js che se opportunamente utilizzato, da linea di comando è in grado di iniettare dei localizzatori (hooks) all'interno del codice sorgente di file freemarker, angular, e di altre tecnologie nelle quali le pagine web seguono un template definito staticamente.
Recandosi nella cartella test-hooks/test-guard e digitando il seguente script in linea di comando:
*node main.js inject-hooks ../frontend/**/*.html --grammar angularjs/freemarker*
vengono iniettati i locatori all'interno del file html specificato.
A questo punto è possibile aggiungere il file attributeHooksLocators.js come estensione di Katalon Recorder, per far in modo che l'IDE sia in grado di generare un codice JUnit che identifichi gli elementi della GUI attraverso i locatori iniettati.

Il primo obiettivo del seguente lavoro è quindi quello di rendere fruibile questo processo di iniezione dei localizzatori all'interno di un Container in Github Actions.
Successivamente un altro obiettivo fondamentale è il deployment, l'esecuzione, ed il testing della web application in fase di sviluppo all'interno del container.
I report generati dall'esecuzione delle test suite saranno poi aggregati ed analizzati durante il ciclo di sviluppo della web app, sia nel caso in cui siano stati iniettati i localizzatori e nel caso in cui non lo siano stati iniettati.
In questo modo al termine della raccolta dei report generati, ci sarà una fase di analisi dei risultati ottenuti confrontando le test suite che tenevano conto degli hooks con quelle che non ne tenevano conto.
Tramite GitHub Actions si vuole rendere automatizzata la generazione dei report, per poi poter verificare l'effettiva maggior robustezza delle test suite tenenti conto dei localizzatori.

Obiettivo finale del seguente lavoro è poi quello di rendere tutto il procedimento più generico possibile, per semplificare al massimo un suo futuro impiego da chiunque ne avesse bisogno durante il ciclo di sviluppo di una applicazione web.

Capitolo 2: Background problema affrontato

In questo capitolo andiamo ad analizzare il background teorico esistente nella letteratura scientifica relativo alla problematica di interesse. Si cita particolarmente il seguente lavoro [1] *"Porfirio Tramontana and Anna Rita Fasolino: A Tool-Supported Technique the Generation of Robust Web Applications E2E Test Cases."* dal quale sono state estrapolate le informazioni riassunte nei paragrafi del corrente capitolo.

2.1. Approcci di testing più utilizzati

Il test funzionale è l'approccio più comunemente utilizzato per assicurare la qualità delle applicazioni Web. Consiste nell'esercitare degli input reali per poter scoprire i possibili guasti di un applicativo, ovvero deviazioni da quanto specificato o dal comportamento previsto.
Il test end-to-end (E2E) è una sorta di test funzionale che viene eseguito dal punto di vista dell'utente finale, testando l'applicazione nel complesso. Di solito viene eseguito stimolando l'applicazione dalla sua interfaccia utente, interagendo con gli elementi che la compongono.
I test E2E possono essere eseguiti sia manualmente dal tester, sia tramite strumenti di automazione dei test in grado di interagire autonomamente con gli elementi dell'interfaccia utente, tramite invio di input ed eventi e verifica dei risultati tramite asserzioni. Gli strumenti di automazione dei test sfruttano script che possono essere prodotti manualmente (utilizzando il cosiddetto "Approccio programmabile al web testing") o mediante "Capture & Replay".
Gli strumenti Capture and Replay (C&R) sono progettati per consentire ad un tester di generare ed eseguire automaticamente script di test partendo da sequenze reali di interazioni dell'utente con l'applicazione sotto test (compresi clic del mouse, inserimenti da tastiera, comandi di digitazione, ecc.). Queste sequenze di interazione sono registrate dallo strumento e tradotte automaticamente in script di test eseguibili.
Tuttavia, le tecniche di C&R non possono essere considerate come "la tecnica definitiva" per il Web testing a causa di diverse limitazioni. Come riportato in [8], presentano la

limitazione dell'incompletezza del test case generato, perché non tutti i comportamenti dell'applicazione possono essere sollecitati da casi di test basati su GUI, sussiste poi il problema della manutenibilità dei casi di test generati, a causa della loro fragilità in seguito ad interventi di manutenzione dell'applicativo.
Ad esempio, gli interventi di manutenzione che incidono sul layout delle pagine Web potrebbero rendere i test case non più rieseguibili, con la conseguente necessità di costose attività di riparazione. Leotta et al. [9] nel 2013 ha svolto un'indagine empirica che mostra come possa l'evoluzione dei casi di test generati dalle tecniche di C&R possa essere più costosa dell'evoluzione dei casi di test scritti manualmente tramite un linguaggio di scripting. Hammoudi, Rothermel e Tonella in [10] definiscono una rottura di un test come l'evento che si verifica quando un test che prima funzionava su un'applicazione web cessa di funzionare in una nuova versione di tale applicazione, non a causa di un cambiamento in una delle specifiche dell'applicazione (che potrebbe rendere un test "obsoleto"), ma piuttosto, a causa di una modifica che impedisce al test di funzionare correttamente. Mentre un tale test si può dire "fallito", le rotture dei test devono essere distinte dai guasti dell'applicazione, che si verificano quando i test funzionano correttamente e rivelano guasti nel sistema sotto esame.
Nello stesso lavoro [10] vengono definite le cause prossimali come le cause "più da vicino collegate" ad ogni rottura che si trova nel codice di test (es. nella GUI HTML codice). Sono diverse dalle cause distali, cioè le modifiche che potrebbero essere apportate nel lato server dell'applicazione Web o nel codice javascript che ha provocato la rottura. Hanno proposto una tassonomia delle cause prossimali della rottura dei test ed i risultati di uno studio empirico sulle applicazioni open source volto a quantificare la frequenza di insorgenza di queste cause. Hanno scoperto che la causa principale delle rotture dei casi di test (73%) sono legate alla rottura dei localizzatori.

2.2. Fragilità dei test – Approcci in letteratura

Il problema della fragilità e delle rotture dei casi di test ottenuti dagli strumenti di C&R è molto sentito nel contesto dei processi di sviluppo iterativo di applicazioni Web, come in quelli incrementali, agili ed evolutivi. In tali processi, le versioni aggiornate dell'applicazione in fase di sviluppo vengono spesso "committate" alla fine di brevi iterazioni temporizzate ed i test di regressione devono essere rieseguiti ad ogni nuovo commit. In un tale contesto, è frequente che i test case esistenti siano soggetti a rotture, e che quindi devano essere riparati.

L'utilizzo di modelli di template per descrivere il layout delle applicazioni Web implementa il principio generale di separazione tra l'interfaccia utente e la logica di business di un applicativo. Questa esigenza è sempre stata fortemente sentita nel contesto delle applicazioni Web, dato che sono sempre state applicazioni dalla forte componente interattiva.

Linguaggi di scripting lato server come JSP, PHP, ASP hanno applicato un modello incentrato sulla pagina in cui le pagine del server sono state scritte utilizzando una versione aumentata di HTML che include tag extra che sono eseguiti da un motore specifico distribuito su di un server web. Il risultato rappresenta le pagine client che sono state inviate al browser nella risposta HTTP.
È emersa quindi la necessità di una separazione completa tra il layout HTML della GUI, i suoi contenuti ed il codice responsabile delle logiche di business, così sono state proposte molte soluzioni [11] compatibili con il pattern architettonico Model View Controller [12]. Più recentemente, sono state però preferite delle soluzioni meno invasive basate sull'arricchimento di tag HTML, consistenti nell'iniezione di nuovi attributi all'interno dei tag HTML, in modo che la pagina risultante rispetti lo standard della sintassi HTML.
In questo modo i comandi espressi all'interno degli attributi possono essere elaborati da leggeri motori javascript che vengono eseguiti sul lato client delle applicazioni. Questa strategia è attualmente adottata da un gran numero di popolari framework (incluso

Angular) che si basano sul pattern MVC o su alcune delle sue evoluzioni, come il modello Model View View Model (MVVM).

L'uso di questi modelli e framework architetturali consente la tecnica di testing che viene proposta in questo lavoro, poichè in questi casi è possibile analizzare staticamente il codice HTML dei modelli sul lato server dell'applicazione e iniettarvi attributi identificativi (ganci). Questi attributi aggiunti non influiranno sull'esecuzione dell'applicazione Web, ma possono essere letti da uno strumento di C&R sulle pagine del client e possono essere utilizzati dai casi di test E2E.

2.3. Una tecnica per la riduzione della fragilità dei test

In questo lavoro si vuole proporre una tecnica per ridurre la fragilità dei casi di test e di conseguenza per ridurre il fenomeno della rottura del test, in base ad un automatico refactoring del codice sorgente delle applicazioni Web basate sui template che inietta identificatori univoci (chiamati hook di seguito) in corrispondenza di ogni elemento dell'applicazione Web che può essere coinvolto in un test case (incluso sia widget che contenitori). Inoltre, un'integrazione in strumenti Capture and Replay (come Katalon Recorder) consente la generazione automatica di robusti localizzatori che sfruttano i ganci iniettati.

Iniziamo a dare delle definizioni per chiarire al meglio il funzionamento della tecnica presentata.

1) Un ***Hook*** è un identificatore univoco permanentemente associato ad un singolo tag di una pagina HTML. Ha un valore intero associato maggiore di zero che non è mai stato associato ad un altro hook generato per la stessa pagina web.
2) L'operazione di ***Hook Injection*** si realizza analizzando il codice sorgente dei modelli HTML dell'applicazione web ed inserendo degli *Hook* univoci in tutti i tag HTML rilevati.

L'Hook Injection è un'operazione abbastanza banale per un codice HTML senza alcun hook precedentemente iniettato. Nel caso di una pagina Web risultante da un'evoluzione di una precedente comprensiva di hook, il codice della versione precedente della pagina (pagina[r-1]) deve essere analizzato per valutare il numero massimo associato ad un gancio *(h max*). Così, i nuovi ganci possono essere iniettati nella nuova versione della pagina (page[r]) dove mancano, utilizzando numeri consecutivi maggiori di *h max*. Nessuno degli hook precedentemente inseriti verrà modificato.
L'algoritmo sotto riportato descrive l'implementazione dell'operazione di hook Injection. Questo algoritmo permette di avere ganci unici per costruzione. Il Web Developer non ha mai bisogno di mantenere manualmente i ganci, ma deve solo fare attenzione per lasciarli invariati durante l'evoluzione di ogni pagina web.

ALGORITHM 1: Hook Injection Algorithm

```
Procedure HookInjection(page[])
    h_max ← 0 ;
    foreach t ∈ page[r − 1].tags do
        if exists(t.hook) then
            if t.hook.number > h_max then
                h_max ← t.hook.number ;
            end
        end
    end
    count ← h_max + 1 ;
    foreach t ∈ page[r].tags do
        if !exists(t.hook) then
            t.hook ← new(Hook) ;
            count ← count + 1 ;
            h.number ← count ;
        end
    end
```

L'iniezione degli hook nei template garantisce che gli stessi ganci possono essere trovati nelle istanze della pagina client. Per questo motivo, uno strumento Capture and Replay è in grado di sfruttare gli hook durante la generazione dei casi di test. A tal fine è possibile definire una strategia per la generazione di locatori basati su hook, che chiameremo strategia *Hook-based Locator* (HL).

- Un Locator HL basato su Hook è un'espressione XPath che include riferimenti agli hook degli elementi che contengono, direttamente o indirettamente, l'elemento che deve essere localizzato.

  Questa strategia è simile a quella considerata dalla tecnica ROBULA [13] e la sua estensione ROBULA+ [14] tenendo conto delle peculiarità introdotte dai Web template. La principale differenza tra il nostro approccio e il ROBULA è che in questo caso il codice sorgente viene preventivamente rifattorizzato al fine di migliorarne la testabilità del codice e di renderne possibile la definizione di un localizzatore più robusto basato su ganci.

2.4. Analisi dello strumento utilizzato

In questa sezione viene descritto lo strumento Test-Hooks che offre due caratteristiche principali (hook injection e locator generation) e diverse utility per il processo di testing. Lo strumento è sviluppato in javascript nel contesto del framework node.js con progettazione basata sui componenti. La sua architettura è mostrata nella figura in basso. Il codice sorgente dello strumento Test Hooks è disponibile gratuitamente su: https://gitlab.com/pcan/test-hooks

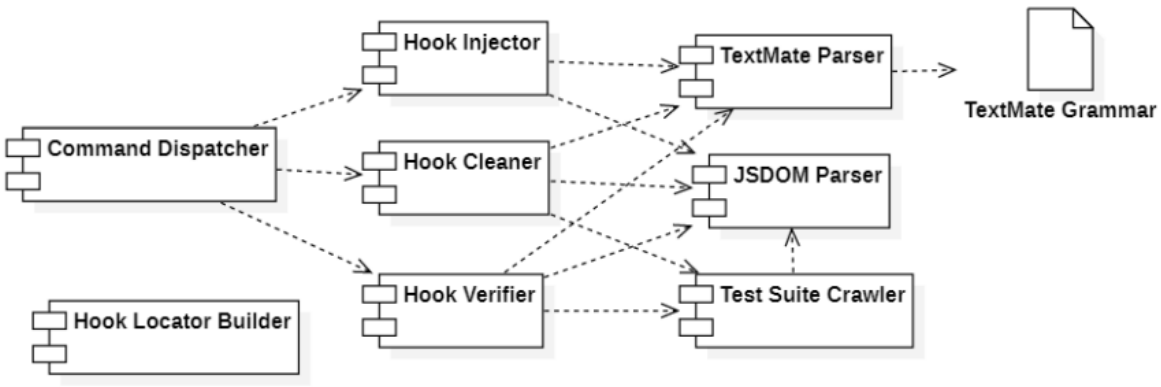


Figura 2.4 – 1: Architettura dello strumento Test-Hooks

Il componente principale di questa architettura è l'*Hook Injector*, che è responsabile dell'analisi del codice sorgente dei template di applicazioni Web ed inietta gli hook all'interno di ogni tag secondo l'algoritmo precedentemente visto. Questo componente è stato costruito in node.js. La versione prototipo di Test-Hooks è in grado di analizzare e iniettare hook su modelli scritti in puro HTML e utilizzando quattro diversi web framework template, ovvero Angular, Freemarker, Twigs e Smarty. Come per quanto riguarda l'HTML puro ed Angular, il componente parser è stato implementato dal componente JSDOM Parser, mentre negli altri casi è stata adottata una soluzione più generale. Questa soluzione utilizza il parser grammaticale TextMate, un parser generale disponibile gratuitamente che può essere utilizzato per diverse lingue e dialetti sulla base delle grammatiche TextMate disponibili.
La libera disponibilità di file di grammatica .cson rendono questa soluzione estendibile a molti altri motori Web template. L'*Hook Injector* può essere chiamato dalla riga di comando tramite il componente *Command Dispatcher*.
Il comportamento dell'Hook Injector è quindi quello descritto dall'algoritmo analizzato: esso rileva in anticipo la presenza di ganci precedentemente iniettati, che non verranno modificati, mentre vengono inseriti nuovi hook con nuovi identificatori univoci.

L'altro componente fondamentale è l'*Hook Locator Builder*. È uno script di estensione che può essere aggiunto come plugin in Katalon Recorder fornendo un aggiuntivo algoritmo di generazione dei localizzatori che può essere impostato come predefinito durante la registrazione dei casi di test. È implementato in javascript ed interagisce direttamente con l'interfaccia *LocatorBuilders* esposta da Katalon Recorder.
Inoltre, in questa versione sono presenti componenti aggiuntivi che possono fornire alcune funzioni di utilità.
Il componente *Hook Cleaner* analizza il codice sorgente dell'applicazione web dopo l'hook injection e la definizione dei casi di test utilizzando i locatori hook-based e pulisce dal codice sorgente tutti gli hook che non sono stati utilizzati nel file di test, rendendo così il codice più chiaro e leggero. Questa operazione aiuta i futuri manutentori per sapere in anticipo se l'evoluzione del codice influenzerà o meno i componenti coinvolti nei casi di test precedentemente registrati. Inoltre, la mancanza di ganci in elementi rilevanti dell'interfaccia può essere interpretata da un tester come una mancanza di copertura della

suite di test suggerendo la necessità di aggiungere nuovi casi di test specifici che coinvolgano quegli elementi.

L'analisi dei file HTML incluse le suite di test generate da Katalon Recorder è ottenuta dal componente *Test Suite Crawler* che utilizza lo stesso componente JSDOM Parser utilizzato da Hook Injector.

Il componente *Verify Hook* ha come input il codice sorgente dell'applicazione Web ed anche la suite di test generata da Katalon Recorder e fornisce un report dei casi di test che sono sicuramente obsoleti dato che non contengono più hook presenti nel codice sorgente. Questa caratteristica permette una rapida identificazione di una parte dei test rotti, senza la necessità di eseguirli. Queste informazioni possono essere utili in diversi scenari. Ad esempio, quando un widget con un hook associato è cancellato dalla GUI, ovviamente i casi di test che lo utilizzano diventeranno obsoleti.

Questi ultimi due componenti possono essere chiamati tramite linea di comando attraverso il *Component Dispatcher*, che fornisce un'interfaccia unica per tutte le funzionalità dello strumento Test-Hooks ad eccezione della generazione dei locatori.

Capitolo 3: Processo Astratto

In questo capitolo si va ad esporre l'aspetto puramente concettuale che si pone di risolvere il seguente lavoro.

L'obiettivo principale è quello di fornire uno strumento automatizzato, che si possa integrare all'interno del ciclo di sviluppo di un applicativo web, in modo da poter fornire un supporto per la generazione ed esecuzione dei casi di test.

Lo strumento realizzato ha quindi l'intento di supportare la generazione di test che risultino più robusti lungo il ciclo di vita di un software, e quello di creare automaticamente una accurata e chiara reportistica dei test eseguiti, in modo da rendere più chiaro ed esente da errori il software da sviluppare.

Il processo dal punto di vista astratto è stato ereditato dai lavori precedenti, (https://gitlab.com/pcan/test-hooks) ma è stato poi dettagliato ed implementato nel contesto di Github.

Il processo astratto consiste quindi in:

1) Creazione di un proprio repository facendo una fork dal lavoro di tesi realizzato.
2) Importazione progetto applicativo web nel repository.
3) Sviluppo nuovo codice.
4) Iniezione dei localizzatori (*hooks*) all'interno del front-end dell'applicazione web.
5) Registrazione dei test tramite tecniche di *capture and replay* applicate sul front-end contenente l'iniezione degli *hooks*.
6) Esportazione dei file di test a partire dai test registrati, convertendoli nel formato desiderato (e.g. Junit).
7) Push dei casi di test, così ottenuti, all'interno del proprio repository Github.
8) Formattazione automatica dei file di test *pushati* nel formato desiderato.
9) Rilascio di una release del software all'interno del repository.
10) Esecuzione automatica dei test di regressione.
11) Generazione automatica dei report contenenti delle statistiche dettagliate sui i test eseguiti.
12) Lettura report dei test eseguiti.

13) Correzione o Eliminazione dei test rotti.
14) *Push* dei test corretti o eliminati all'interno del repository.
15) Formattazione automatica dei file di test *pushati* nel formato desiderato.
16) Torna al passo 3 se non è terminato il ciclo di sviluppo.

Questi step sono stati descritti dal punto di vista metodologico e possono essere applicati alle varie tecnologie di sviluppo web front-end dove la struttura della pagina web segue un template definito staticamente (come *Angular*, *Freemarker*, *Smarty* e *Twigs*).
Concettualmente è possibile iniettare qualsiasi tipo di localizzatore all'interno del front-end della propria applicazione web, ci si potrebbe affidare anche a qualsiasi altro sito di hosting diverso da *Github* purchè basato sulla tecnologia *Git*, ed i file di test generati a partire da tecniche di *capture and replay* possono essere scritti in qualsiasi linguaggio di programmazione.

Descritto quindi il processo astratto alla base del seguente lavoro, si vanno brevemente a delineare le tecnologie utilizzate nel processo pratico sviluppato:

1) Localizzatori *Hooks*, facendo uso del lavoro del Dott. Cangianiello.
2) Sito di Hosting *Github*.
3) *Katalon Recorder*, utilizzato per generare file di test a partire da tecniche "Capture & Replay".
4) *Spring Boot*, framework utilizzato per lo sviluppo di utility.

Andiamo adesso nei paragrafi successivi a descrivere il processo operativo, implementato secondo una metodologia top-down, facendo uso delle particolari tecnologie qui delineate.

Capitolo 4: Background Tecnologico

In questo paragrafo andiamo a delineare dettagliatamente, una per una, le caratteristiche delle tecnologie coinvolte durante la realizzazione del seguente progetto.

4.1. Github & Github Actions

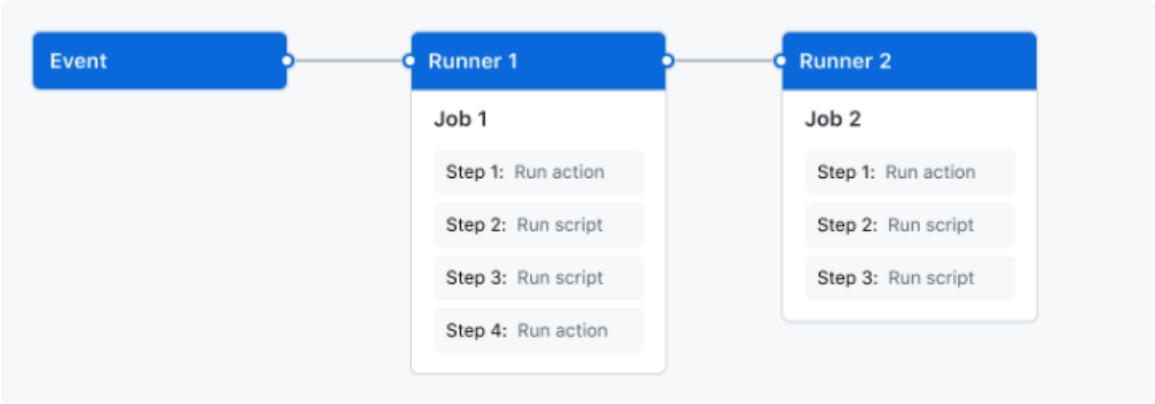
GitHub (https://github.com) è un servizio di hosting per progetti software. Il nome deriva dal fatto che "GitHub" è una implementazione dello strumento di controllo versione distribuito Git.
Git è un version control system open source che è stato avviato da Linus Torvalds, la stessa persona che ha creato Linux. Git è simile ad altri sistemi di controllo della versione: Subversion, CVS e Mercurial per citarne alcuni [15].

I motivi della diffusione di Github stanno quindi sia nell'interfaccia grafica offerta, che permette di eseguire comandi git senza passare da linea di comando, e sia nel servizio di hosting gratuito offerto.
L'interfaccia di Github è estremamente intuitiva e permette facilmente di effettuare i principali comandi git, quali "commit, rollback, push, fork, branch…".

Una dei maggiori punti di forza di Github è però *Github Actions* [16].
Github Actions (GA) è una piattaforma di Continuous Integration e Continuous Delivery (CI/CD), integrata in Github, che permetta di automatizzare il processo di building, testing e deployment del software. GA permette quindi di creare dei workflow che vanno ad effettuare la build ed il testing del proprio repository ad ogni richiesta di pull o più in generale al verificarsi di un evento di interesse specificato.

GA va oltre il semplice DevOps e consente di eseguire interi flussi di lavoro mentre si verificano altri eventi nel repository. Un'altra caratteristica fondamentale è che Github fornisce macchine virtuali Linux, Windows e macOS per eseguire i propri flussi di lavoro.

Dal punto di vista schematico, Github Actions è composto dai seguenti componenti:



Il flusso di lavoro (*workflow*) contiene uno o più *work* che possono essere eseguiti in ordine sequenziale o in parallelo. Ogni processo verrà eseguito all'interno del proprio *runner della* macchina virtuale o all'interno di un contenitore e include uno o più *step* che eseguono uno script definito dall'utente o una *action*, che è un'estensione riutilizzabile che può semplificare il flusso di lavoro.

Diamo adesso delle definizioni dei componenti fondamentali di *Github Actions*:

- **Workflows**: è un configurabile processo automatizzato che esegue uno o più job. I workflow vengono definiti in dei file YAML, i quali vengono archiviati nella repository e vengono eseguiti quando c'è il verificarsi di un evento specificato nel file.
- **Events**: è un'attività specifica in un repository che attiva un workflow. Esempi di eventi possono essere una pull request o un push.
- **Jobs**: è un insieme di step all'interno di un flusso di lavoro che vengono eseguiti dallo stesso runner. Ogni step è uno script di shell che verrà eseguito o un'azione che verrà eseguita. Gli step vengono eseguiti in ordine e dipendono l'uno dall'altro e dato che questi vengono eseguiti in uno stesso runner possono condividere dati fra di loro.
- **Actions**: è un'applicazione personalizzata per la piattaforma GitHub Actions che esegue un compito complesso, ma ripetuto frequentemente.
- **Runners**: è il server che esegue il flusso di esecuzione quando viene triggerato.

4.2. Katalon Recorder

Katalon Recorder [17] è una estensione Chrome utilizzata per la registrazione e riproduzione di eventi su di un web browser. Katalon è una perfetta alternativa al *Selenium IDE* per la registrazione e l'export degli script Selenium.

A differenza di *Selenium IDE*, la capacità di registrazione di Katalon Recorder è potente sui principali browser Web: Chrome, Firefox e IE. Questa estensione è stata il progetto campione del concorso "Katalon Studio Hackathons". *Katalon Automation Recorder* è stato sviluppato per supportare gli utenti che non erano più in grado di registrare e riprodurre i test di automazione utilizzando il vecchio Selenium IDE. Katalon Recorder può essere integrato con altre soluzioni Katalon (Katalon Studio e Katalon TestOps) per esigenze avanzate di generazione di test, reporting e orchestrazione dei test.

Sintetizziamo in questo elenco le principali caratteristiche di Katalon:

- Automatizza le attività ripetitive sui browser, come la generazione di report, la compilazione di moduli, l'automazione di giochi, ecc.
- Usa i comandi originali Selenium IDE (Selenese), oltre alle istruzioni di blocco if...elseIf...else...endIf e while. ..endWhile. È supportato il test del controllo dell'input file.
- Utilizza più tipi di localizzatore tra cui XPath e CSS.
- Importa i dati dei test da file CSV per i test basati sui dati.
- Importa script da Selenium IDE al progetto esistente in Katalon Recorder
- Genera dei report condivisibili tramite dashboard e grafici, con dati storici e analisi effettuate tramite la soluzione integrata Katalon TestOps.
- Supporto dei comandi e degli script di estensione dell'IDE Selenium legacy (anche conosciuto come *user-extensions.js*) per lo sviluppo di builder e azioni di localizzatore personalizzati.
- Esportazione di script Selenium WebDriver in vari formati e framework.

È possibile esportare i test registrati nei seguenti linguaggi e framework supportati: Node New Relic Synthetics, Python App Dynamics, Java JUnit, Java TestNG, Python unittest,

C# MSTest, C# NUnit, Java WebDriver RC +JUnit, JSON Dynatrace Synthetics, Typescript Potractor, Framework robot, Ruby Rspec, JavaScript WebDriver.io, XML, JavasScript Puppeteer, JSON Puppeteer.

4.3. Progetto test-hooks

Test-hooks è uno strumento software ([2] Dott. Pierantonio Cangianiello – progetto https://gitlab.com/pcan/test-hooks) in grado di iniettare un tipo di localizzatore (hooks) all'interno del codice html di applicazioni web che caricano il front-end in maniera statica (Angular, Freemarker, Smarty e Twigs).
L'obiettivo fondamentale del progetto è quello di ridurre la *fragilità dei test*, intesa come la necessità di aggiornamento di test di regressione, i quali possono non risultare più eseguibili durante l'evolversi dello sviluppo di un applicativo web.
A tale scopo, il seguente progetto offre quindi una tecnica automatica (*hook-injection*) di instrumentazione del codice sorgente di applicazioni web template-based, unita ad una tecnica di generazione semiautomatica di casi di test tramite approccio *Capture & Replay* che sfrutti questa instrumentazione.

L'obiettivo finale dello strumento è quindi quello di misurare l'aumento della robustezza dei casi di test, ottenuta passando da una soluzione basata su generatore di test di default ad una basata sulla generazione di casi di test ottenuti a partire dal codice instrumentato.

Il seguente software si basa a sua volta su altri strumenti tecnologici, che risultano essere quindi dei prerequisiti tecnici per il suo utilizzo.
Le operazioni da eseguire per l'utilizzazione sono:

- Effettuare il download del progetto dal seguente repository https://gitlab.com/pcan/test-hooks
- Installazione di **node.js** versione 8 o successiva da https://nodejs.org/en/download/
- Installazione di tutte le librerie di supporto di node.js:

  Ci si posizioni nella cartella test-hooks/test-guard del codice scaricato e si eseguano le seguenti operazioni (potrebbero essere necessari permessi da amministratore):

  - **npm install** (per installare le librerie di base)
  - **npm install --global windows-build-tools** (solo in ambiente windows)
- Installazione di **Katalon Recorder** all'interno del browser di test.
- Installazione dello script personalizzato di generazione dei casi di test
  - E' necessario aprire da browser *Katalon Recorder*, poi la scheda **Extension Script** ed aggiungere lo script **attributeHooksLocator** localizzato nella cartella *selenium* del codice scaricato. Successivamente è necessario riavviare il browser per completare l'installazione.

4.4. Spring Boot

Nel corso del seguente lavoro verrà ampiamente utilizzato il framework Spring Boot per la realizzazione di applicativi sotto forma di file jar (Java ARchive), i quali serviranno per l'esecuzione automatica e per l'autogenerazione dei report inerenti ai i casi di test eseguiti.

Java Spring Framework (Spring Framework) [18] è un popolare framework open source utilizzato a livello aziendale per la creazione di applicazioni standalone che vengono eseguite su Java Virtual Machine (JVM).
Java Spring Boot (Spring Boot) è uno strumento che rende più semplice e veloce lo sviluppo di applicazioni Web e di microservizi con Spring Framework attraverso tre funzionalità principali:

1) Autoconfigurazione
2) Opinionated approach (approccio supponente)
3) La possibilità di creare applicazioni standalone

Queste funzionalità interagiscono per fornire uno strumento che consente di configurare un'applicazione basata su Spring con una configurazione e un'impostazione minimale.

Spring Framework offre una *funzione di dependency injection* che consente agli oggetti di definire le proprie dipendenze che il contenitore Spring in seguito inserisce in essi. Ciò

consente agli sviluppatori di creare applicazioni modulari costituite da componenti fortemente disaccopiati ideali per microservizi e applicazioni di rete distribuite. Spring Framework offre anche il supporto integrato per le attività tipiche che un'applicazione deve eseguire, come data binding, conversione del tipo, convalida, gestione delle eccezioni, gestione di risorse ed eventi, internazionalizzazione e altro ancora. Si integra con varie tecnologie Java EE come RMI (Remote Method Invocation), AMQP (Advanced Message Queuing Protocol), Java Web Services e altri. In sintesi, Spring Framework fornisce agli sviluppatori tutti gli strumenti e le funzionalità necessarie per creare applicazioni Java EE multipiattaforma liberamente accoppiate che vengono eseguite in qualsiasi ambiente.

Per quanto completo sia Spring Framework, richiede comunque tempo e conoscenze significative per configurare, impostare e distribuire le applicazioni Spring. Spring Boot mitiga quindi questo sforzo con le tre già citate importanti funzionalità.

1) **Autoconfigurazione**

Le applicazioni vengono inizializzate con dipendenze preimpostate che non è necessario configurare manualmente. Poiché Java Spring Boot è dotato di funzionalità di configurazione automatica integrate, configura automaticamente sia il framework Spring sottostante che i pacchetti di terze parti in base alle impostazioni (e in base alle migliori pratiche, il che aiuta a evitare errori). Anche se si possono ignorare queste impostazioni predefinite una volta completata l'inizializzazione, la funzione di configurazione automatica di Java Spring Boot consente di iniziare a sviluppare rapidamente le applicazioni basate su Spring e riduce la possibilità di errori umani.

2) **Approccio supponente**

Spring Boot utilizza un approccio supponente per aggiungere e configurare le dipendenze di avviamento, in base alle esigenze del progetto. Seguendo il proprio giudizio, Spring Boot sceglie quali pacchetti installare e quali valori predefiniti utilizzare, invece di richiedere all'utente di prendere tutte quelle decisioni da solo e impostare tutto manualmente.

Si possono definire le esigenze del proprio progetto durante il processo di inizializzazione, durante il quale si possono scegliere tra più dipendenze di avvio,
denominate *Spring* Starter, che coprono casi d'uso tipici. Eseguendo Spring Boot Initializr compilando un semplice modulo web, senza alcuna codifica.
Spring Boot include oltre 50 Spring Starter e sono disponibili molti altri avviatori di terze parti.

3) **Applicazioni autonome**

Spring Boot aiuta gli sviluppatori a creare applicazioni che vengono *eseguite*. In particolare, consente di creare applicazioni autonome che funzionano da sole, senza fare affidamento su un server Web esterno, incorporando un server Web come Tomcat o Netty nelle applicazioni durante il processo di inizializzazione. Di conseguenza, si può avviare la propria applicazione su qualsiasi piattaforma semplicemente premendo il comando *Esegui*.

Capitolo 5: Processo operativo

In questo capitolo andiamo a descrivere formalmente la struttura del processo operativo centrale del seguente lavoro.
Su grandi linee i principali passi effettuati per la realizzazione del seguente lavoro di tesi, sono stati:

- Implementazione di una applicazione web, sulla quale applicare l'automazione, derivante dal lavoro successivamente descritto, nel suo ciclo di sviluppo.
- Importazione del progetto *test-hooks*, implementato dal Dott. Pierantonio Cangianiello (Pierantonio Cangianiello / test-hooks · GitLab).
- Implementazione di vari Java ARchive (.jar), che serviranno come programmi di supporto nell'automazione del processo operativo.
- Implementazione files di Continuos Integration (CI) in YAML, che automatizzino l'intero processo operativo. Per quanto riguarda l'implementazione dei file YAML, è stato utilizzato Github Actions per avere un supporto contemporaneamente sul versioning e sulla CI nel ciclo di sviluppo del software.

Il processo operativo è gestito in modo tale da essere facilmente generalizzabile per il ciclo di sviluppo di una generica applicazione web.

5.1. Descrizione Top Down

In questo paragrafo, seguendo un approccio descrittivo top-down, si va a descrivere formalmente passo per passo quali step deve seguire un generico utilizzatore del seguente lavoro.

Le macro-attività principali possono essere schematizzate come segue:

1) Creazione del repository, facendo una fork a partire dal repository del seguente lavoro.
2) Importazione del progetto di una propria applicazione web all'interno del repository.
3) Settaggio *variabili di ambiente* in Github.
4) Sviluppo nuovo codice applicazione web.
5) Registrazione casi di test con tecniche *Capture & Replay* tramite *Katalon Recorder*.
6) Esportazione test registrati nel formato *WebDriver+JUnit*.
7) Push dei casi di test, così ottenuti, all'interno del proprio repository Github.
8) Esecuzione file di CI per formattare le test suite nel formato desiderato.
9) Generazione prima release in Github.
10) Esecuzione file di CI per l'esecuzione dei test di regressione e generazione della reportistica associata, il quale contiene il nucleo del processo operativo implementato nel seguente lavoro. Assume quindi l'onere di andare ad iniettare i locatori, eseguire automaticamente i file di test, generare dei report accurati e salvarli in apposite directory all'interno della release appena rilasciata.
11) Lettura report dei test eseguiti.
12) Correzione o Eliminazione dei test rotti.
13) Push in Github dei test corretti o eliminati.
14) Riesecuzione file di CI per formattare le test suite nel formato desiderato.
15) Se non è terminato il ciclo di sviluppo, torna al passo 4.

Per averne una descrizione più formale, vediamo a pagina seguente l'*Activity Diagram* che descrive con massima astrazione questo processo operativo.

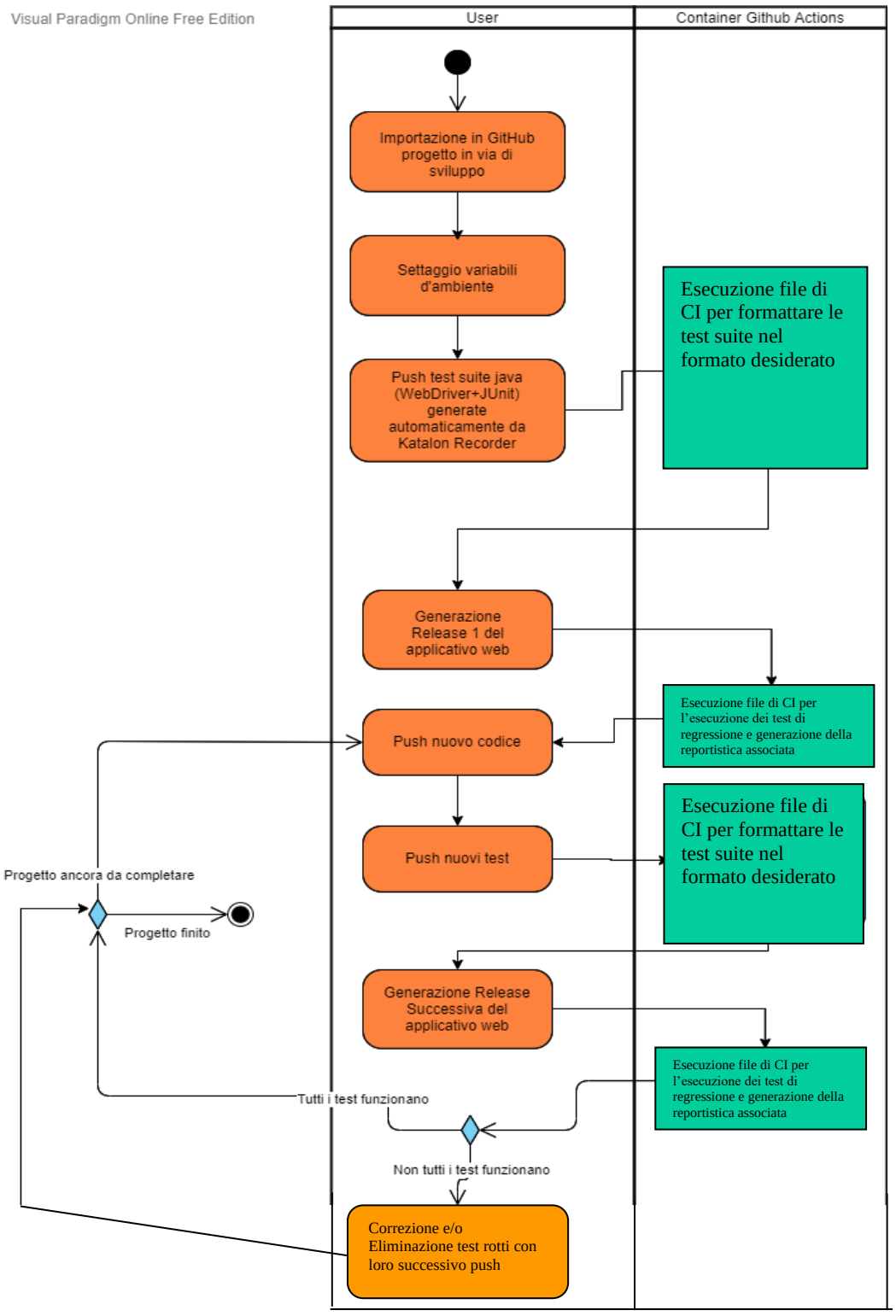


Figura 5.1 – 1: Activity Diagram – "Procedimento Operativo"

5.2. Importazione progetto

Il primo passo operativo consiste nell'importare un generico progetto di una web app in via di sviluppo.
Un generico applicativo per poter essere importato deve rispettare il seguente requisito:

- Il lato front-end deve essere realizzato secondo una delle tecnologie compatibili con il lavoro precedentemente citato del Dott. Cangianiello. Quindi deve essere realizzato in modo tale che la struttura delle pagine web segua un template definito staticamente. Le tecnologie compatibili sono *Angular, Freemarker, Smarty e Twigs*.

5.3. Settaggio variabili d'ambiente

Dopo aver importato il proprio progetto contenente lo sviluppo di una generica web application, la prima cosa da fare è il settaggio delle variabili d'ambiente.

Questo step è reso necessario dal fatto, che tutti i software realizzati (sia i *jar* che i file *yml*) si basano su delle variabili di ambiente esternamente definibili. Grazie a questo metodo è infatti possibile generalizzare la strategia operativa per qualsiasi progetto compatibile con il lavoro del Dott. Cangianiello.

Le variabili d'ambiente sono state salvate all'interno di un *enviroment* del repository Github, in modo tale che possano essere utilizzate ed aggiornate da un qualsiasi utente. Queste hanno inoltre il vantaggio di poter essere utilizzate senza essere visualizzate in chiaro, ciò è molto importante in quanto è possibile inserirvi al loro interno i valori di dati sensibili come le credenziali dell'account personale github per poter effettuare una push.

In totale ci sono solamente 6 *environment variable* da definire:

EMAIL_ACCOUNT_GITHUB
NOME_ACCOUNT_GITHUB
PASSWORD_ACCOUNT_GITHUB
FE_EXTENSION_TYPE
GRAMMAR_TYPE
DIR_FILE_FE

Tabella 5.3 – 1: Variabili d'ambiente

Analizziamone adesso il significato una per una:

1) *EMAIL_ACCOUNT_GITHUB:* e-mail associata all'account github personale dell'utente.
2) *NOME_ACCOUNT_GITHUB:* nome associato all'account github personale dell'utente.
3) *PASSWORD_ACCOUNT_GITHUB:* password associata all'account github personale dell'utente.
4) *FE_EXTENSION_TYPE:* estensione dei file di *frontend* sui quali si vuole andare ad effettuare la *hook-injection* (e.g. *.html*, *.ftl*).
   I file di frontend possono essere realizzati tramite *Angular*, *Freemarker*, *Smarty o Twigs*, ovvero di una delle tecnologie supportate dal progetto *test-hooks.*
5) *GRAMMAR_TYPE:* valore del flag *--grammar*, serve per specificare di che tipo è il file di front-end sul quale si vuole effettuare la hook-injection (freemarker, angularjs…)
6) *DIR_FILE_FE:* directory root all'interno della quale (comprese sottocartelle) sono memorizzati tutti i file di frontend.

NB: I file di test devono essere memorizzati nella directory *./project-test-headless/src/test/java/com/example/TesiIntegrazioneProgettoEsterno/*. Si è intrapresa la scelta di far inserire i casi di test in una directory apposita, posta all'interno del progetto di utility *project-test-headless* per fare in modo che i casi di test esportati da Katalon siano completamente indipendenti da qualsiasi tecnologia utilizzata dalla applicazione da testare.

Prima di andare avanti si riporta un esempio di configurazione delle variabili d'ambiente per fornire una maggiore chiarezza:

```
EMAIL_ACCOUNT_GITHUB: t*********@gmail.com
NOME_ACCOUNT_GITHUB: g*********
PASSWORD_ACCOUNT_GITHUB: *********
FE_EXTENSION_TYPE: .html
GRAMMAR_TYPE: angularjs
DIR_FILE_FE: /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale/insert-here-your-web-app/root-web-app/frontend
```

5.4. File mainOnPush.yml

Il file *mainOnPush.yml* è un file YAML che viene eseguito su di una macchina ubuntu in cloud ad ogni push all'interno del repository.

Il suddetto file ha un duplice fine:

- Iniettare gli *Hooks* all'interno dei file di FrontEnd dell'applicazione web da testare.
- Correggere automaticamente il formato dei test esportati da *Katalon Recorder*, per fare in modo che possano essere eseguiti in modalità *headless* all'interno di un container Ubuntu.

Vediamo l'activity diagram riportante le attività che esegue sequenzialmente il file *mainOnPush.yml* ogni volta che viene effettuato un push nel repository.

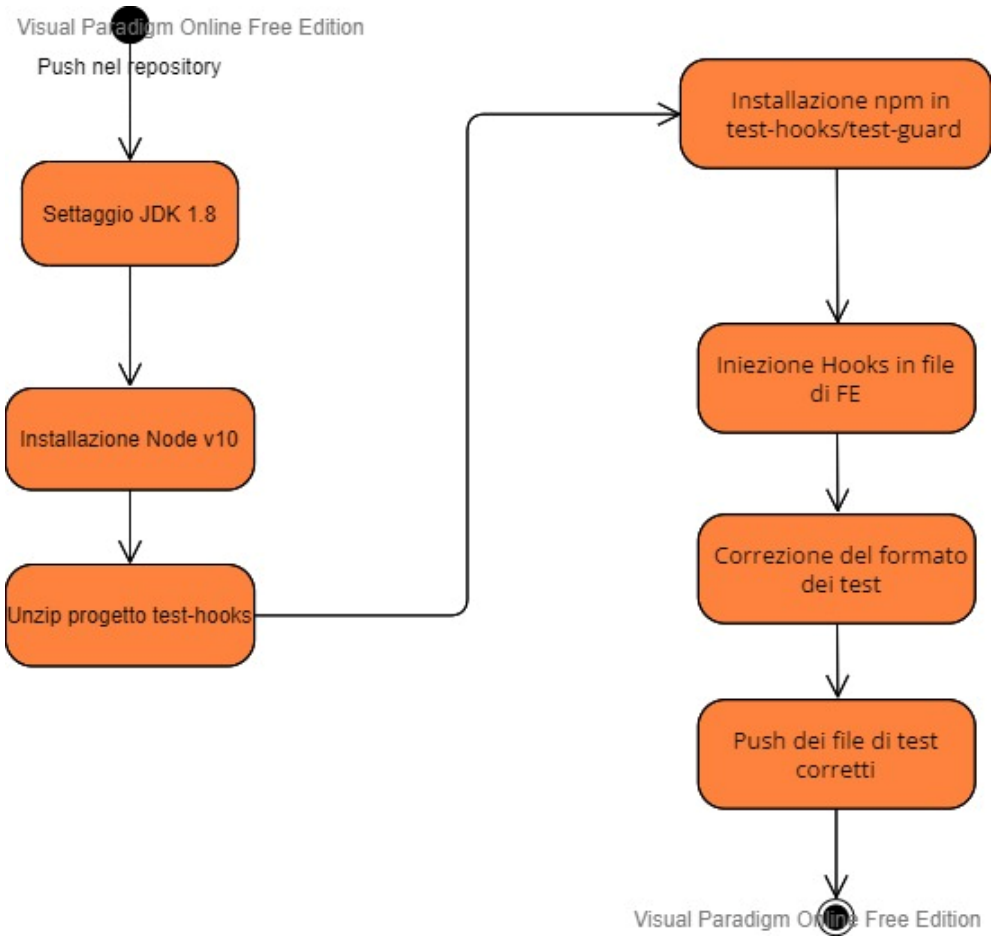


Figura 5.4 – 1: Activity Diagram *mainOnPush.yml*

Si vuole far notare che i passi specificati si riferiscono a ciò che avviene all'interno del container, durando quindi per l'intera esecuzione del file *mainOnPush.yml*. Quindi tutti i passi relativi alle installazioni (JDK 1.8, Node v.10, Installazione npm in *test-hooks/test-guard*) sono effettuati totalmente all'interno delle directory del container che viene istanziato e non all'interno delle omonime cartelle presenti all'interno del repository.

5.4.1. Implementazione mainOnPush.yml

A questo punto, dopo aver mostrato l'*activity diagram*, possiamo visualizzare l'implementazione del file *mainOnPush.yml*:

```
# This is a basic workflow to help you get started with Actions

name: Github Actions - mainOnPush

# Controls when the workflow will run
on:
  push:
    branches:
      - '**'      # matches every branches



  # Allows you to run this workflow manually from the Actions tab
  workflow_dispatch:

# A workflow run is made up of one or more jobs that can run sequentially or in parallel
jobs:
  # This workflow contains a single job called "build"
  build:
    # The type of runner that the job will run on
    runs-on: ubuntu-latest
    environment:
      name: envForGithubActions


   # Steps represent a sequence of tasks that will be executed as part of the job
    steps:
      # Checks-out your repository under $GITHUB_WORKSPACE, so your job can access it
      - name: Step 1 - Checkout main branch from GitHub
        uses: actions/checkout@v2

      # Runs a single command using the runners shell
      - name: Step 2 - Set up JDK 1.8
        uses: actions/setup-java@v1
        with:
          java-version: 1.8
```

```
- name: Step 3 - Installare Node versione 10 in Ubuntu
  run: |
    sudo apt update
    sudo apt -y install curl dirmngr apt-transport-https lsb-release ca-certificates
    curl -sL https://deb.nodesource.com/setup_10.x | sudo bash
    sudo apt install nodejs
    sudo npm cache clean -f
    sudo npm install -g n
    sudo n stable
    sudo n 10.18.0
    echo "Versione di node: "
    node -v
    echo "Versione di npm: "
    npm -v
    npm install
    echo "Provo ad INSTALLARE bcrypt"
    npm install bcrypt
    npm fund
    echo "vedi se ha fatto bene o no"

- name: Step 4 - Unzip test-hooks ed installazione dentro test-guard di npm
  run: |
    ls -a
    unzip test-hooks.zip
    cd test-hooks
    ls -a
    cd test-guard
    ls -a
    echo "Versione di node: "
    node -v
    echo "Versione di npm: "
    npm -v
    echo "Provo ad installare con npm install"
    npm install
    ls -a
    npm audit fix
    echo "Provo ad INSTALLARE bcrypt"
    npm install bcrypt
    npm fund
    echo "vedi se ha fatto bene o no"

- name: Step 5 - Instrumenta l'applicazione iniettando gli Hooks
  run: |
      cd test-hooks
      cd test-guard
      ls -a
      echo "Prima dell injection! "
      echo "Visualizza versione di node"
      node -v
      echo "... Cancellazione e successivo aggiornamento di node_modules ..."
      rm -rf node_modules/
```

```
          npm update
          echo "INJECTION adesso..."
          echo "Visualizza versione di node"
          node -v
          cd /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale/Tesi-injector-plugin
          mvn clean install
          cd target
          sudo bash -c 'java -jar Tesi-injector-plugin-1.0-SNAPSHOT.jar ${{ secrets.FE_EXTENSION_TYPE }} ${{ secrets.GRAMMAR_TYPE }} ${{ secrets.DIR_FILE_FE }} correct nomeTag'
          echo "Dopo l'injection!!!"
          git config --global user.email "${{ secrets.EMAIL_ACCOUNT_GITHUB }}"
          git config --global user.name "${{ secrets.NOME_ACCOUNT_GITHUB }}"
          git config --global user.password "${{ secrets.PASSWORD_ACCOUNT_GITHUB }}"
          git status
          echo "Andiamo ad effettuare il push dei file iniettati"
          git add /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale/insert-here-your-web-app/*
          cd /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale
          echo "`date +%Y-%m-%d_%H-%M-%S`" > timeCommit.txt
          git add timeCommit.txt
          echo "File aggiunti!"
          git commit -m "Commit dal file yml dei file iniettati"
          git branch -M ${GITHUB_REF#refs/heads/}
          git push -u origin ${GITHUB_REF#refs/heads/} --force
          echo "Push effettuato"

      - name: Step 6 - Esecuzione progetto correzioneFormatoTest
        run: |
          cd correzioneFormatoTest
          mvn clean install
          cd target
          echo "Vediamo contenuto cartella target"
          ls -a
          java -jar correzioneFormatoTest-0.0.1-jarCorrezioneFormatoTest.jar /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale/project-test-headless/src/test/java/com/example/TesiIntegrazioneProgettoEsterno /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale/chromedriver_v94_linux64/chromedriver com.example.TesiIntegrazioneProgettoEsterno

      - name: Step 7 - Push dei file di test corretti
        run: |
          echo "Vediamo quali cartelle ci sono nella directory di partenza"
          ls -a
          git status
          git config --global user.email "${{ secrets.EMAIL_ACCOUNT_GITHUB }}"
          git config --global user.name "${{ secrets.NOME_ACCOUNT_GITHUB }}"
          git config --global user.password "${{ secrets.PASSWORD_ACCOUNT_GITHUB }}"
          echo "Andiamo ad aggiungere la cartella dei report"
          git add /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale/project-test-headless/src/test/java/com/example/TesiIntegrazioneProgettoEsterno
```

```
echo "Cartella aggiunta!"
cd /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale
echo "`date +%Y-%m-%d_%H-%M-%S`" > timeCommit.txt
git add timeCommit.txt
git commit -m "push automatico per la correzione del formato dei test"
git branch -M ${GITHUB_REF#refs/heads/}
git push -u origin ${GITHUB_REF#refs/heads/}
```

Tabella 5.4.1 – 1: file mainOnPush.yml

Riassumendo brevemente ciò che accade, ad ogni push viene istanziata una macchina ubuntu in un container, dove:

- *Step 1:* si effettua il checkout dal branch in uso del repository github.
- *Step 2:* viene settata la JDK versione 1.8.
- *Step 3:* viene installata la versione 10.18.0 di Node nel container Ubuntu.
- *Step 4:* viene decompresso il file *test-hooks.zip* e successivamente viene installato npm all'interno della directory test-hooks/test-guard. Questi sono passi necessari per poter utilizzare l'hook-injection offerta dal progetto test-hooks del dottor Cangianiello.
- *Step 5:* vengono iniettati gli hooks all'interno dei file di FE della propria applciazione web da testare. L'iniezione viene fatta tramite il progetto di utility *Tesi-injector-plugin*, il quale utilizzando il progetto *test-hooks*, si occupa di andare ad iniettare gli hooks nei file di FE. Dopo l'iniezione, vengono *pushate* le modifiche nel repository.
- *Step 6:* viene eseguito il progetto di utility *correzioneFormatoTest* passandogli tre argomenti da linea di comando (il path dei test JUnit, il path di dove è memorizzato il chromedriver, il nome del package dei test JUnit nel progetto).
- *Step 7:* dopo aver inserito le credenziali per l'accesso al proprio account github, viene fatto il push nel corrente branch dei file di test aggiornati.

Proseguiamo andando quindi a definire più dettagliatamente il comportamento dei due progetti di utility qui utilizzati.

5.4.2. Progetto di utility *Tesi-injector-plugin*

Il progetto di utility *Tesi-injector-plugin* è un progetto Maven implementato in Java, che è stato realizzato ad-hoc per poter effettuare automaticamente l'injection degli hooks all'interno di tutte le sottodirectory della directory specificata nella variabile d'ambiente DIR_FILE_FE.

Analizziamo quindi l'Activity Diagram che esplica il comportamento del seguente progetto di utility:

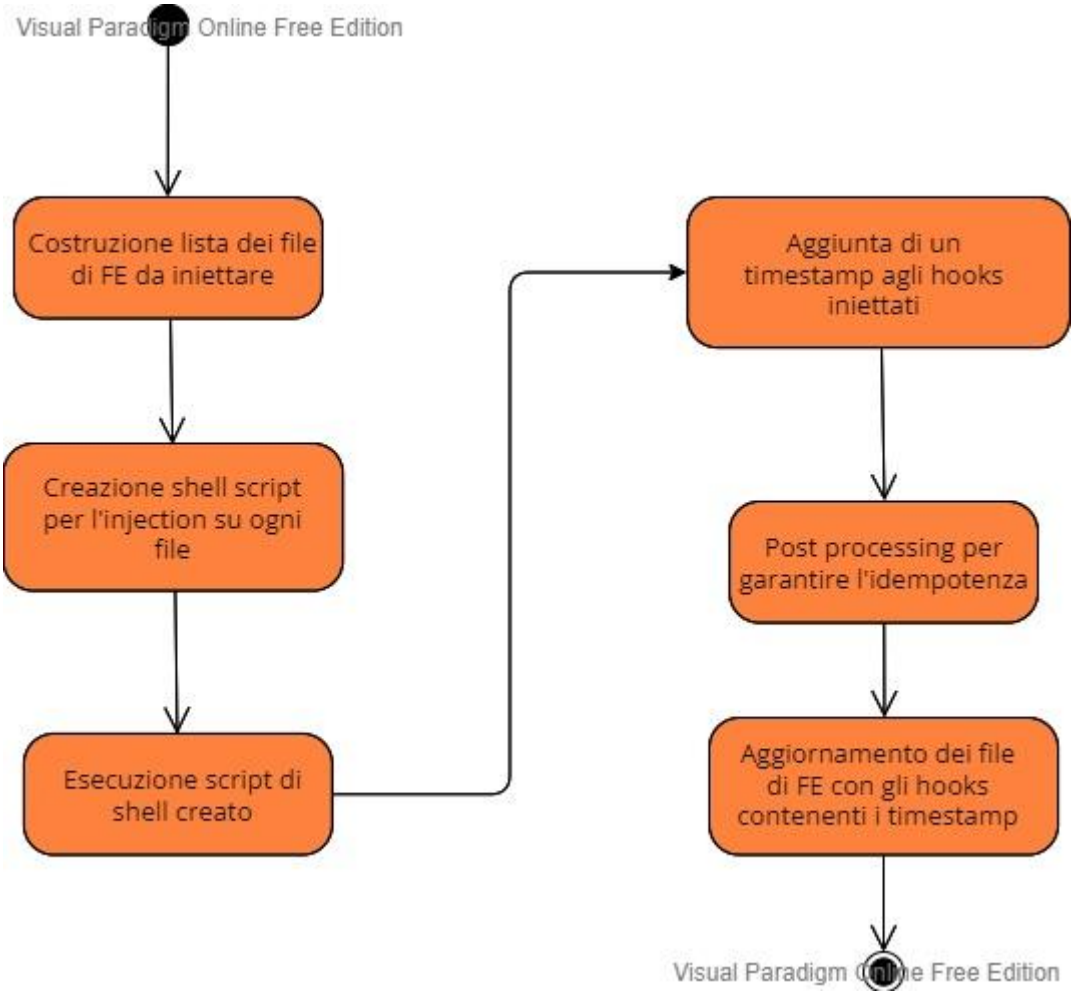


Figura 5.4.2 – 1: ActivityDiagram *Tesi-injector-plugin*

Quindi andando a dettagliare le fasi elencate nell'Activity Diagram:

1) Viene calcolata la lista contenente i path di tutti i file con estensione FE_EXTENSION_TYPE (variabile d'ambiente) presenti in tutte le sottodirectory DIR_FILE_FE (variabile d'ambiente).

2) A partire dalla lista di tutti i path, viene costruito un file *.sh* che contiene tutti i comandi necessari per iniettare gli hooks all'interno dei file di FE utilizzando il progetto test-hooks.

3) Viene eseguito il file .sh precedentemente creato per fare l'iniezione degli hooks.

4) Vengono modificati gli hooks iniettati, facendo in modo che venga anteposto un timestamp (con precisione fino ai nanosecondi) agli hooks iniettati. Questa modifica degli Hooks è necessaria per garantire che gli hooks iniettati siano univocamente differenti su tutti i file di FE appartenenti al progetto.

5) A questo punto viene effettuato un post processing, per fare in modo che ogni tag presente in ogni file di FE abbia effettivamente un solo hook iniettato e che questo hook rimanga inalterato dopo successive esecuzioni della seguente utility.

6) Infine si occupa di aggiornare i valori dei file di FE a valle dell'aggiunta degli hooks contenenti i timestamp.

5.4.3. Progetto di utility *correzioneFormatoTest*

Andiamo adesso a capire il funzionamento del progetto di utility realizzato ad-hoc *correzioneFormatoTest*.

Il suo scopo è quello di prendere in ingresso file di test (con estensione *.java*) di tipo *JUnit+WebDriver* generati con *Katalon Recorder* e di adattarli all'esecuzione *headless* utilizzando i driver di chrome in modalità headless.

L'obiettivo del suddetto progetto è quello di andare a correggere automaticamente il formato dei test junit, facendo in modo che questi possano essere eseguiti in modalità headless.

Gli step concettuali sono i seguenti:

1) L'utente sviluppatore registra un caso di test con la tecnica *Capture & Replay* tramite *Katalon Recorder*.
2) L'utente esporta il caso di test registrato tramite *Katalon Recorder* nel formato *WebDriver+JUnit*.
3) L'utente carica il file di test generato all'interno della directory di test presente nel repository github.
4) Al push del nuovo file di test inserito viene eseguito il file *correzioneFormatoTest.jar*
5) A questo punto l'applicativo, con l'ausilio di tecniche di parserizzazione, va a modificare il codice java del file di test caricato in modo da renderlo automaticamente compatibile per l'esecuzione headless (deve eseguire in modalità headless dato che dovrà essere eseguito in un container ubuntu non dotato di interfaccia grafica).
6) All'interno del file di test appena autocorretto viene inserita dall'applicativo anche una riga commentata che funge da marcatore, per poter identificare il file di test come un test che è già stato corretto. Quindi grazie all'aggiunta di questo marker, il file correzioneFormatoTest.jar ad ogni push di nuovo codice, andrà a correggere solamente i file di test che non contengono ancora il marker (che quindi non sono ancora stati corretti).

Analizziamo quindi l'*Activity Diagram* ad alto livello di astrazione relativo al progetto di utility correzioneFormatoTest, in modo da capirne meglio il funzionamento logico.

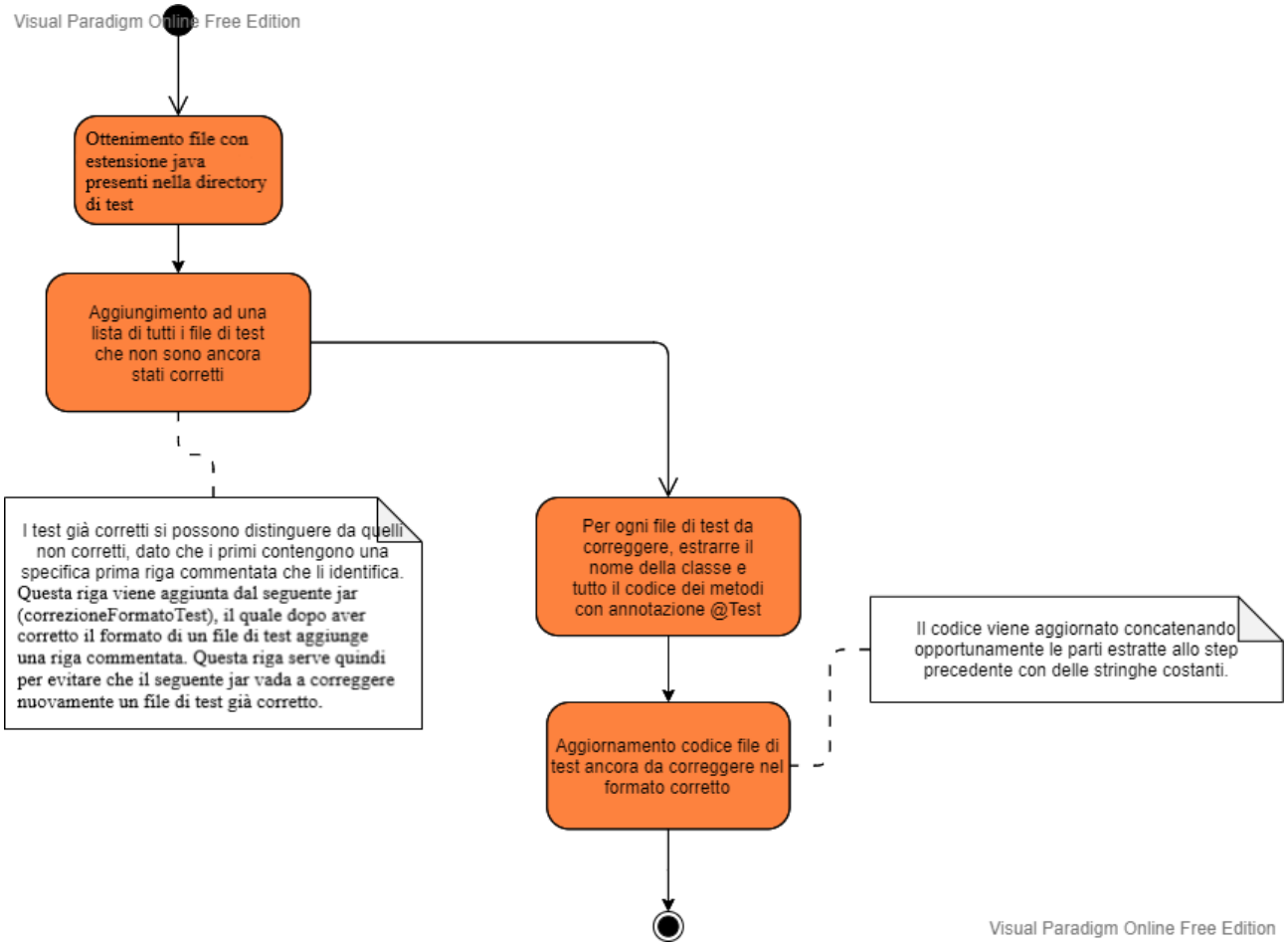


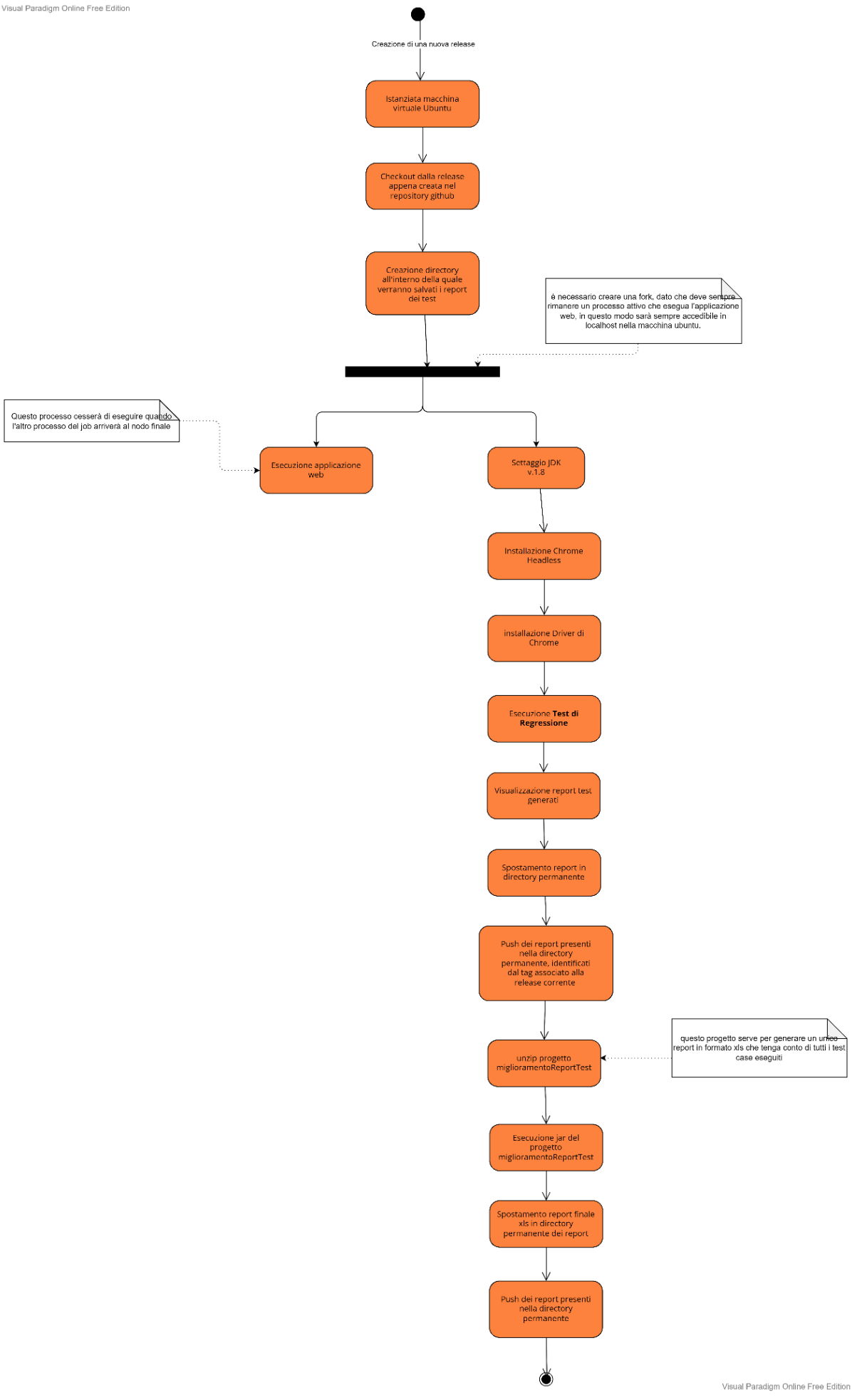
Figura 5.4.3 – 1: ActivityDiagram *correzioneFormatoTest*

5.5. Creazione nuova release

Quando si raggiunge una versione stabile dell'applicativo, è fondamentale creare una nuova release (identificata da un tag) all'interno del repository Github.

Questo procedimento di creazione di una release innesta un trigger di attivazione per il file YAML *main.yml*. Il file *main.yml* è un processo complesso di CI/CD, suddiviso in molte fasi. Per poter capire al meglio il suo funzionamento, anche in questo caso andiamo a rappresentarne la logica tramite un apposito Activity Diagram:

Vediamo nella pagina successiva l'*activity diagram* inerente al file *main.yml*.


Visual Paradigm Online Free Edition
Creazione di una nuova release
Istanziata macchina virtuale Ubuntu
Checkout dalla release appena creata nel repository github
Creazione directory all'interno della quale verranno salvati i report dei test
è necessario creare una fork, dato che deve sempre rimanere un processo attivo che esegua l'applicazione web, in questo modo sarà sempre accedibile in localhost nella macchina ubuntu.
Questo processo cesserà di eseguire quando l'altro processo del job arriverà al nodo finale
Esecuzione applicazione web
Settaggio JDK v.1.8
Installazione Chrome Headless
installazione Driver di Chrome
Esecuzione Test di Regressione
Visualizzazione report test generati
Spostamento report in directory permanente
Push dei report presenti nella directory permanente, identificati dal tag associato alla release corrente
unzip progetto miglioramentoReportTest
questo progetto serve per generare un unico report in formato xls che tenga conto di tutti i test case eseguiti
Esecuzione jar del progetto miglioramentoReportTest
Spostamento report finale xls in directory permanente dei report
Push dei report presenti nella directory permanente
Visual Paradigm Online Free Edition

Una volta visto l'activity diagram, possiamo andare a vedere con maggior consapevolezza il codice sorgente del file *main.yml*:

```
        # This is a basic workflow to help you get started with Actions

name: Github Actions - main (On Release)

# Controls when the workflow will run
on:
  release:
    types: [published]



  # Allows you to run this workflow manually from the Actions tab
  workflow_dispatch:

# A workflow run is made up of one or more jobs that can run sequentially or in parallel
jobs:
  # This workflow contains a single job called "build"
  build:
    # The type of runner that the job will run on
    runs-on: ubuntu-latest
    environment:
      name: envForGithubActions


    # Steps represent a sequence of tasks that will be executed as part of the job
    steps:
      # Checks-out your repository under $GITHUB_WORKSPACE, so your job can
access it
      - name: Step 1 - Checkout main branch from GitHub
        uses: actions/checkout@v2

      - name: Step 2 - Visualizzare tag corrente
        run: |
          echo "Vediamo il tag corrente:"
          echo "${GITHUB_REF#refs/tags/}"
          echo "${GITHUB_REF#refs/heads/}"
          echo "Vediamo se si potrebbe capire automaticamente che si usano gli hooks"
          echo "Stampa a video degli utlimi 5 caratteri del tag della release corrente"
          varTemp=$(echo "${GITHUB_REF#refs/tags/}" | tail -c 6)
          echo "$varTemp"
          echo "Vediamo qual è l'id dell'attuale commit"
          echo "$GITHUB_SHA"

      - name: Step 3 - Creazione directory Test Suite
        run: |
          ls -a
          if [ ! -d "/home/runner/work/Tesi-StrumentoGenerale/Tesi-
StrumentoGenerale/TestSuite" ]; then
```

```
          echo "Vedi se sono dentro l'if numero 1"
          sudo mkdir TestSuite
        fi
        echo "Vediamo se ha creato la cartella"
        ls -a

    - name: Step 4 - SEZIONE BACK-END
      run: |
          cd /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale
          bash startBackEnd.sh

    - name: Step 5 - SEZIONE FRONT-END
      run: |
          cd /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale
          bash startFrontEnd.sh

    - name: Step 6 - Set up JDK 1.8 per progetti di utility
      uses: actions/setup-java@v1
      with:
        java-version: 1.8

    - name: Step 7 - Install Chrome Headless
      run: |
        sudo apt-get update
        sudo apt-get upgrade
        sudo apt-get -u dist-upgrade
        sudo apt-get install -y libappindicator1 fonts-liberation
        sudo apt-get -y install dbus-x11 xfonts-base xfonts-100dpi xfonts-75dpi xfonts-
scalable
        wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
        sudo apt-get -f install
        google-chrome-stable --headless --disable-gpu

    - name: Step 8 - Install Chrome-Driver for test junit
      run: |
          wget https://chromedriver.storage.googleapis.com/2.41/chromedriver_linux64.zip
          echo "Vedo se si trova il punto zip"
          ls -a
          echo "Provo ad unzippare chormedriver_linux64.zip"
          unzip chromedriver_linux64.zip
          sudo mv chromedriver /home/runner/work/Tesi-StrumentoGenerale/Tesi-
StrumentoGenerale/chromedriver_v94_linux64/chromedriver
          sudo chown root:root /home/runner/work/Tesi-StrumentoGenerale/Tesi-
StrumentoGenerale/chromedriver_v94_linux64/chromedriver
          sudo chmod +x /home/runner/work/Tesi-StrumentoGenerale/Tesi-
StrumentoGenerale/chromedriver_v94_linux64/chromedriver
          echo "Vediamo il nome della cartella unzippata"
          ls -a
          echo "Vediamo se ha scaricato il driver"
          cd chromedriver_v94_linux64
          echo "Contenuto della cartella chromedriver_v94_linux64"
          ls -a
```

```
    - name: Step 9 - Eseguire test maven
      run:  |
        cd /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale/project-test-headless
        sudo bash -c 'mvn test site' || echo "Uno o più test sono falliti."

    - name: Step 10 - Visualizzare report test
      run: |
        cd /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale/project-test-headless/target/surefire-reports
        ls -a
        echo "Visualizziamo adesso tutti i report txt ottenuti dalle nostre test suite"
        cat *.txt
        echo "Fine visualizzazione report!!!"

    - name: Step 11 - Spostamento report in directory corretta
      run: |
       cd /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale/project-test-headless/target
       echo "Visualizziamo se c'è la cartella surefire-reports"
       ls -a
       echo "Rinominare la cartella surefire-reports in nome del tag della release creata"
       sudo mv surefire-reports ${GITHUB_REF#refs/tags/}
       echo "Vediamo se la ha rinominata:"
       ls -a
       sudo cp -r ${GITHUB_REF#refs/tags/} /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale/TestSuite

    - name: Step 12 - Push dei report
      run: |
       echo "Vediamo quali cartelle ci sono nella directory di partenza"
       ls -a
       git config --global user.email "${{ secrets.EMAIL_ACCOUNT_GITHUB }}"
       git config --global user.name "${{ secrets.NOME_ACCOUNT_GITHUB }}"
       git config --global user.password "${{ secrets.PASSWORD_ACCOUNT_GITHUB }}"
       git status
       echo "Andiamo ad aggiungere la cartella dei report"
       git add /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale/TestSuite
       echo "Cartella aggiunta!"
       git commit -m "Eseguendo un primo commit dal file yml"
       git tag --list
       echo "Andiamo ad aggiungere, se non ci era già, il tag corrente"
       git tag --force ${GITHUB_REF#refs/tags/}
       echo "Vediamo la nuova lista di tag:"
       git tag --list
       git push -u origin ${GITHUB_REF#refs/tags/} --force
       echo "Push effettuato"

    - name: Step 13 - Unzip progetto "Miglioramento report test"
```

```
      run: |
        ls
        echo "Unzip del progetto"
        unzip miglioramentoReportTest.zip
        echo "Progetto zippato"
        ls

    - name: Step 14 - Esecuzione progetto "Miglioramento report test"
      run: |
        echo "Andiamo nella directory dove si trova il file jar eseguibile"
        cd miglioramentoReportTest/target
        ls -a
        echo "Proviamo ad eseguire il file jar, passandogli come args[0] la directory dei
report da analizzare e come args[1] il nome del report complessivo xls da generare"
        java -jar miglioramentoReportTest-0.0.1-jarReportTest.jar
/home/runner/work/Tesi-StrumentoGenerale/Tesi-
StrumentoGenerale/TestSuite/${GITHUB_REF#refs/tags/} tabellaReportTest
        echo "File jar eseguito!!!"
        echo "Vediamo il file xls di report unificato se c'è"
        ls -a
        echo "Vediamo il contenuto del file xls di report unificato!!!"
        sudo mv tabellaReportTest.xls ${GITHUB_REF#refs/tags/}.xls

    - name: Step 15 - Spostamento report xls in cartella dei report txt
      run: |
        cd miglioramentoReportTest/target
        sudo cp ${GITHUB_REF#refs/tags/}.xls /home/runner/work/Tesi-
StrumentoGenerale/Tesi-StrumentoGenerale/TestSuite/${GITHUB_REF#refs/tags/}

    - name: Step 16 - Commit del report xls
      run: |
        echo "Vediamo quali cartelle ci sono nella directory di partenza"
        ls -a
        git config --global user.email "${{ secrets.EMAIL_ACCOUNT_GITHUB }}"
        git config --global user.name "${{ secrets.NOME_ACCOUNT_GITHUB }}"
        git config --global user.password "${{
secrets.PASSWORD_ACCOUNT_GITHUB }}"
        git status
        echo "Andiamo ad aggiungere la cartella dei report"
        git add /home/runner/work/Tesi-StrumentoGenerale/Tesi-
StrumentoGenerale/TestSuite
        echo "Cartella aggiunta!"
        cd /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale
        echo "`date +%Y-%m-%d_%H-%M-%S`" > timeCommit.txt
        git add timeCommit.txt
        git commit -m "Commit dal file main.yml"
        git tag --list
        echo "Andiamo ad aggiungere, se non ci era già, il tag corrente"
        git tag --force ${GITHUB_REF#refs/tags/}
        echo "Vediamo la nuova lista di tag:"
        git tag --list
        git push -u origin ${GITHUB_REF#refs/tags/} --force
```

```
        echo "Push effettuato"
```

Tabella 5.5 – 1: file *main.yml*

Per dare la maggior chiarezza possibile, andiamo a vedere una descrizione testuale di ciò che avviene in ogni step:

- *Step 1:* si effettua il checkout dal tag specificato del repository github.
- *Step 2:* Visualizzazione del nome del tag appena generato.
- *Step 3:* creazione directory all'interno della quale verranno successivamente salvati i report dei test.
- *Step 4:* si esegue il file *startBackEnd.sh*.

  Nota Bene: Vi è la necessità di avviare un nuovo flusso esecutivo per fare in modo che il primo dei due flussi mantenga sempre in esecuzione il lato back-end dell'applicativo web, mentre il secondo proceda nell'esecuzione dei passi successivi del file YAML. Il file *startBackEnd.sh* va ovviamente customizzato dall'utente in base alla tipologia di back-end della propria applicazione web da testare. Tenuto conto di ciò, dopo che l'utente ha scritto i comandi per l'installazione del software necessario per l'esecuzione del *BackEnd* (in *startBackEnd.sh*), il comando dell'esecuzione dovrà essere scritto seguito da una "e-commerciale (&)" che serve per creare un nuovo flusso di esecuzione in un terminale Ubuntu.
  Per esempio, il comando per l'esecuzione di un file jar potrebbe essere:
  java -jar nome-applicativo-be.jar &
- *Step 5:* si esegue il file *startFrontEnd.sh* all'interno di un nuovo flusso di esecuzione.

  Nota Bene: Vi è la necessità di avviare un nuovo flusso esecutivo per fare in modo che il primo dei due flussi mantenga sempre in esecuzione il lato front-end dell'applicativo web, mentre il secondo proceda nell'esecuzione dei passi successivi del file YAML. Il file *startFrontkEnd.sh* va ovviamente customizzato dall'utente in base alla tipologia di front-end della propria applicazione web da testare. Tenuto conto di ciò, dopo che l'utente ha scritto i comandi per l'installazione del software necessario per l'esecuzione del *Front-End* (in *startFrontEnd.sh*), il comando dell'esecuzione dovrà essere scritto seguito da una "e-commerciale (&)" che serve per creare un nuovo flusso di esecuzione in un terminale Ubuntu.
  Per esempio, il comando per l'esecuzione di un FE Angular potrebbe essere:

npm start &

- *Step 6:* settaggio JDK versione 1.8 (verrà utilizzata dai progetti di utility).
- *Step 7:* installazione Chrome Headless nel container ubuntu.
- *Step 8:* installazione dei driver di Chrome nella macchina ubuntu.
- *Step 9:* esecuzione dei test JUnit desiderati.
- *Step 10:* si visualizzano tutti i report ottenuti dalle test suite eseguite. L'esecuzione di ogni test suite ha comportato la creazione di due report distinti, uno in formato *txt* e l'altro in *xml*.
- *Step 11:* spostamento report ottenuti all'interno di una directory coerente, il cui nome cambia a seconda del tag di release che ha attivato il file *main.yml*.
- *Step 12:* si effettua il push dei report generati dalle test suite eseguite all'interno di una directory identificata dal tag che ha innescato il trigger di attivazione del file yaml.
- *Step 13:* unzip del progetto "*miglioramentoReportTest*".
- *Step 14:* ci si reca nella directory *miglioramentoReportTest/target* e ne si esegue il file *jar* passandogli in ingresso da linea di comando il *path* dei report generati dalle test suite eseguite ed il nome della tabella di report finale che deve generare. In questo modo al termine della sua esecuzione, il file *jar* avrà creato un file *excel (xls)* che racchiuda al suo interno tutte le informazioni rilevanti estratte dagli *n* report delle test suite eseguite.
- *Step 15:* si sposta il report finale complessivo, generato dal progetto *miglioramentoReportTest*, nella directory dove si trovano salvati tutti i singoli report generati nella corrente release.
- *Step 16:* si effettua il push del report complessivo (memorizzato nel file *nomeTag.xls*) all'interno della corrente release del repository.

A questo punto dopo aver descritto nel dettaglio il funzionamento di ciò che avviene alla creazione di ogni nuova release, rimane da definire il funzionamento del progetto "*miglioramentoReportTest*" utilizzato dal file *main.yml*. Prima di procedere con la sua definizione, nel prossimo paragrafo andiamo a delineare il comportamento e l'architettura in toto dello strumento realizzato nel corso del seguente lavoro.

5.5.1. Comportamento dello strumento realizzato

Riepilogando, lo strumento realizzato segue le seguenti dinamiche, a seconda che si stia effettuando una push nel repository o che si stia creando un nuovo tag:

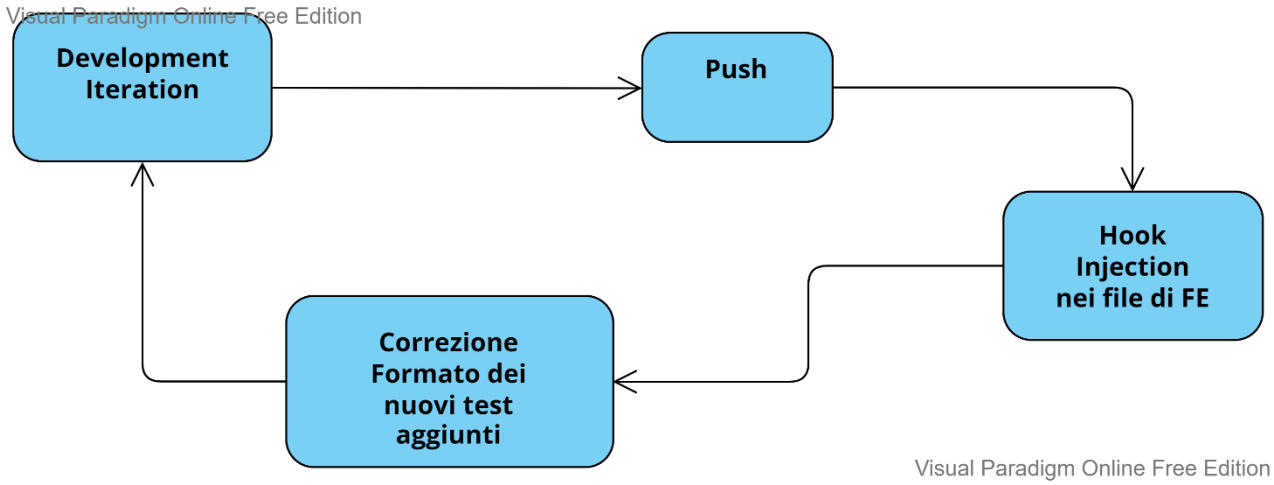


Figura 5.5.1 – 1: Comportamento strumento dopo un push

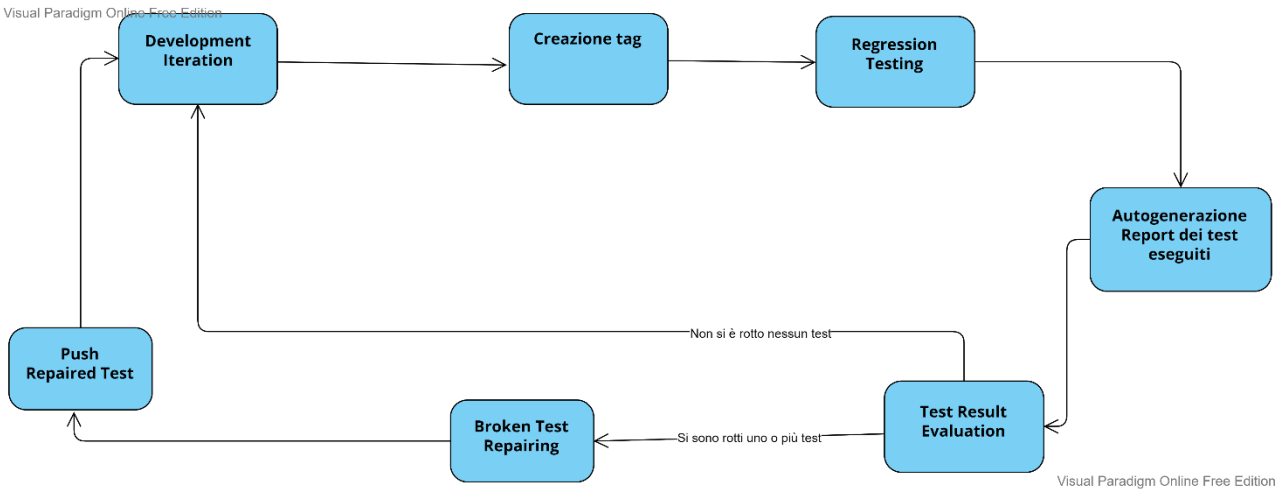


Figura 5.5.1 – 2: Comportamento strumento dopo creazione tag.

Quindi:

- Dopo un push effettuato nel repository, vengono iniettati gli hooks all'interno dei file di FE e successivamente vengono corretti i formati dei casi di test in modo da renderli eseguibili in modalità headless all'interno di un container.

Nel caso in cui non vi è nessun nuovo test da correggerne il formato, non accade nulla.

- Alla creazione di un tag, vengono eseguiti i test di regressione e vengono autogenerati dei report dettagliati sui test eseguiti. A questo punto si valuta se ci sono dei test rotti, se ve ne sono si deve procedere manualmente a correggerli per poi poterne fare il push. Al push avviene ciò che è stato spiegato nel punto precedente, ovvero viene effettuata la correzione del formato dei test, se necessaria. A questo punto inizia una nuova iterazione del ciclo di sviluppo, tornando nell'attività "*Development Iteration*", la quale comprende la scrittura di nuovo codice e la creazione di nuovi test.

5.5.2. Architettura dello strumento realizzato

In questo paragrafo andiamo ad analizzare invece l'aspetto architetturale dello strumento realizzato, tramite l'esposizione di un *Component Diagram* e di *Deployment Diagram*.

Analizziamo prima il Component Diagram:

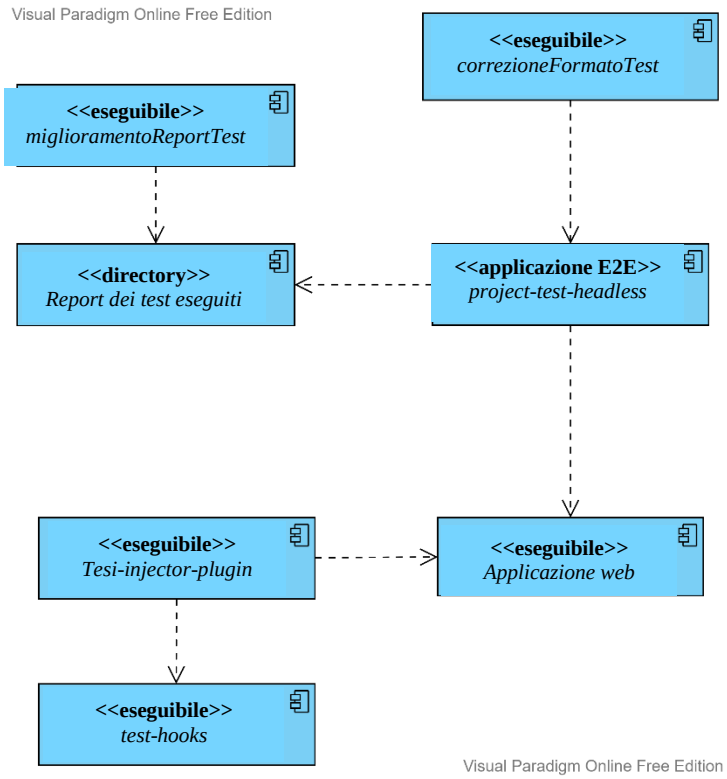


Figura 5.2.2 – 1: *Component Diagram*

Nel diagramma compaiono i seguenti sette componenti:

- **correzioneFormatoTest:** il seguente componente è rappresentato da un applicativo in formato jar (Java Archive) che si occupa di adattare i file di test, rendendoli idonei per l'esecuzione headless all'interno di un container.
- **project-test-headless:** il seguente componente rappresenta il progetto che contiene i file di test da eseguire.
- **applicazione web:** il seguente componente è l'unico che deve essere customizzato dall'utente sviluppatore, in quanto rappresenta l'applicativo web in via di sviluppo che si intende testare.
- **test-hooks:** il seguente componente è composto dall'intero progetto sviluppato dal Dott. Cangianiello [2].
- **Tesi-injector-plugin:** il seguente componente si occupa di iniettare gli Hooks modificati con i timestamp all'interno del FE dell'applicazione web.
- **miglioramentoReportTest:** il seguente componente è rappresentato da un applicativo in formato jar (Java Archive), che prendendo in ingresso dei report (in formato txt e xml) generati a valle dell'esecuzione dei test di regressione, ne va ad effettuare una parserizzazione al fine di generare dei report riassuntivi e significativi (in formato xls) circa lo stato dei test eseguiti.
- **directory reportTest eseguiti:** il seguente componente rappresenta invece la directory presente all'interno del repository all'interno della quale verrano *pushate* tutte le tipologie dei test di regressione eseguiti.

A questo punto per poter comprendere al meglio l'architettura realizzata, è opportuno riportare dei *Deployment Diagram* al fine di poter meglio specificare in che modo vengono distribuiti i componenti, e quali sono le dipendenze fra di loro.

In prima analisi si riporta il *Deployment Diagram*, correlato all'attività di creazione di una nuova release.
Si riporta il diagramma nella pagina successiva.

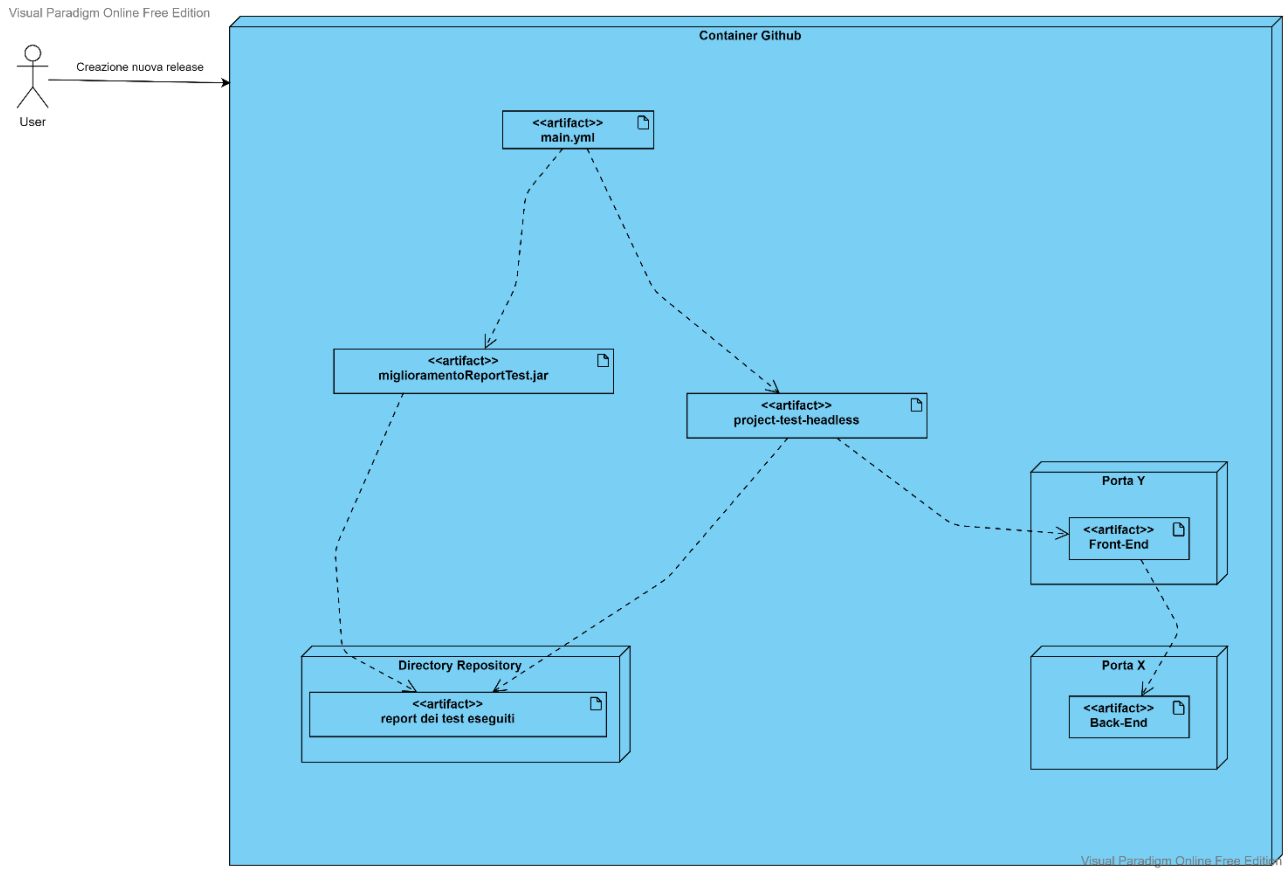


Figura 5.2.2 – 2: *Deployment Diagram – on relase*

Analizzando il diagramma soprastante ciò che si evince è che alla creazione di una nuova release da parte di un utente, l'artefatto *main.yml* (presente all'interno del repository) va ad innescare un container all'interno del quale verranno eseguiti gli artefatti *miglioramentoReportTest* e *project-test-headless.*

I loro funzionamenti sono già stati dettagliati precedentemente, qui infatti si mette enfasi sul mero aspetto della distribuzione dei vari artefatti di cui è composta l'architettura.

Altra caratteristica fondamentale che si evince è che l'*applicazione web* viene *deployata* all'interno del container, andando ad eseguire su delle porte differenti il lato front-end ed il lato back-end dell'applicativo.

Grazie a questa astrazione è infatti possibile che i casi di test memorizzati nell'artefatto *project-test-headless* possano andare a testare un applicativo web (facendo richieste a localhost) il cui codice sorgente risiede in un altro artefatto.

Si mostra infine il Deployment Diagram relativo alle dipendenze dei componenti che entrano in gioco a seguito di un'attività di push da parte di un utente.

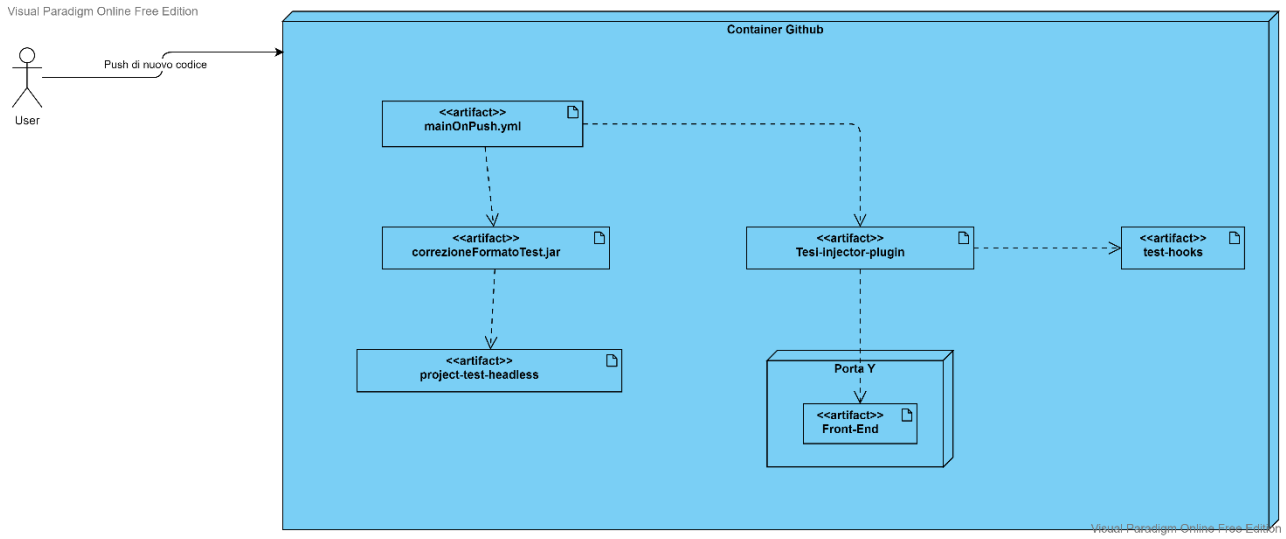


Figura 5.2.2 – 3: *Deployment Diagram – on push*

Al push di nuovo codice si esegue il file *mainOnPush.yml*, il quale va prima ad eseguire il progetto di utility *Tesi-injector-plugin* che si occupa di iniettare gli hooks contenenti i timestamp (utilizzando il progetto *test-hooks*) all'interno del FE dell'applicazione web da testare.

Successivamente il file *mainOnPush.yml* va ad eseguire il file *jar* per la correzione del formato dei test in modo da renderli eseguibili nel container in modalità headless.

I file che vengono corretti appartengono all'artefatto *project-test-headless*.

5.5.3. Progetto miglioramentoReportTest

L'artefatto rimanente da descrivere è il jar "*miglioramentoReportTest*", il quale esegue prendendo in ingresso:

- la directory dei report generati durante l'esecuzione delle test suite nell'ultima release. Ogni test suite eseguita ha generato due report, uno in formato testuale (txt) e l'altro in formato xml.
- Il nome del file *xls* che deve generare.

Quindi partendo da *n* file *txt* ed *n* file *xml*, deve generare in output *un solo* file *xls* che tenga conto complessivamente di tutte le informazioni rilevanti contenute nei *2n* file di report.

Prima di capire il funzionamento logico del progetto è necessario vedere la struttura degli *n* report testuali e degli *n* xml, vediamo quindi i due seguenti report come esempio:

```
-------------------------------------------------------------------------------
Test set: it.catalogo.test.TestSuiteErrorFailureSuccessProva1
-------------------------------------------------------------------------------
Tests run: 3, Failures: 1, Errors: 1, Skipped: 0, Time elapsed: 42.936 s <<< FAILURE! - in it.catalogo.test.TestSuiteErrorFailureSuccessProva1
testLocalHost2_loc_Hooks_release_1_2  Time elapsed: 1.964 s  <<< FAILURE!
org.opentest4j.AssertionFailedError: expected: <Violoncello> but was: <Batteria>
        at it.catalogo.test.TestSuiteErrorFailureSuccessProva1.testLocalHost2_loc_Hooks_release_1_2(TestSuiteErrorFailureSuccessProva1.java:83)

provaSuWikipedia_release_1_0  Time elapsed: 30.72 s  <<< ERROR!
org.openqa.selenium.NoSuchElementException:
no such element: Unable to locate element: {"method":"link text","selector":"Jakarta (Parte in parentesi aggiunta per far fallire test) quServlets"}
  (Session info: headless chrome=94.0.4606.61)
For documentation on this error, please visit: https://www.seleniumhq.org/exceptions/no_such_element.html
Build info: version: '3.141.59', revision: 'e82be7d358', time: '2018-11-14T08:17:03'
System info: host: 'LAPTOP-E23N3NJF', ip: '192.168.56.1', os.name: 'Windows 10', os.arch: 'amd64', os.version: '10.0', java.version: '1.8.0_302'
Driver info: org.openqa.selenium.chrome.ChromeDriver
Capabilities {acceptInsecureCerts: false, browserName: chrome, browserVersion: 94.0.4606.61, chrome: {chromedriverVersion: 93.0.4577.63 (ff5c0da2ec0ad..., userDataDir: C:\Users\talit\AppData\Loca...}, goog:chromeOptions: {debuggerAddress: localhost:50526}, javascriptEnabled: true, networkConnectionEnabled: false, pageLoadStrategy: normal, platform: WINDOWS, platformName: WINDOWS, proxy: Proxy(), setWindowRect: true, strictFileInteractability: false, timeouts: {implicit: 0, pageLoad: 300000, script: 30000}, unhandledPromptBehavior: dismiss and notify, webauthn:extension:credBlob: true, webauthn:extension:largeBlob: true, webauthn:virtualAuthenticators: true}
Session ID: ea0c60dbf463d5d036a3b6aca16f47fa
```

```
*** Element info: {Using=link text, value=Jakarta (Parte in parentesi aggiunta per far fallire test) quServlets}
        at
it.catalogo.test.TestSuiteErrorFailureSuccessProva1.provaSuWikipedia_release_1_0(TestSuiteErrorFailureSuccessProva1.java:49)
```

Tabella 4.5.3 – 1: report *it.catalogo.test.TestSuiteErrorFailureSuccessProva1.txt*

```
<?xml version="1.0" encoding="UTF-8"?>
<testsuite xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="https://maven.apache.org/surefire/maven-surefire-plugin/xsd/surefire-test-report.xsd" name="it.catalogo.test.TestSuiteErrorFailureSuccessProva1" time="42.936" tests="3" errors="1" skipped="0" failures="1">

…….. (parte centrale tagliata per non occupare  troppo spazio)……
  <testcase name="testLocalHost2_loc_Hooks_release_1_2" classname="it.catalogo.test.TestSuiteErrorFailureSuccessProva1" time="1.964">
    <failure message="expected: &lt;Violoncello&gt; but was: &lt;Batteria&gt;" type="org.opentest4j.AssertionFailedError"><![CDATA[org.opentest4j.AssertionFailedError: expected: <Violoncello> but was: <Batteria>
        at
it.catalogo.test.TestSuiteErrorFailureSuccessProva1.testLocalHost2_loc_Hooks_release_1_2(TestSuiteErrorFailureSuccessProva1.java:83)
]]></failure>
    …..(parte tagliata)……
  </testcase>
  <testcase name="provaSuWikipedia_release_1_0" classname="it.catalogo.test.TestSuiteErrorFailureSuccessProva1" time="30.72">
    <error message="no such element: Unable to locate element: {&quot;method&quot;:&quot;link text&quot;,&quot;selector&quot;:&quot;Jakarta (Parte in
it.catalogo.test.TestSuiteErrorFailureSuccessProva1.provaSuWikipedia_release_1_0(TestSuiteErrorFailureSuccessProva1.java:49)
</error>
    <system-out><![CDATA[La suit di Test è iniziata
]]></system-out>
  </testcase>
  <testcase name="testProva_loc_genericoLocatore_release_1_2" classname="it.catalogo.test.TestSuiteErrorFailureSuccessProva1" time="7.713"/>
</testsuite>
```

Tabella 4.5.3 – 2: report *TEST_it.catalogo.test.TestSuiteErrorFailureSuccessProva1.xml*

Una volta vista la struttura dei singoli report generati da ogni test suite, si va a delineare il funzionamento logico dell'applicativo "miglioramentoReportTest" tramite un opportuno *Activity Diagram*:

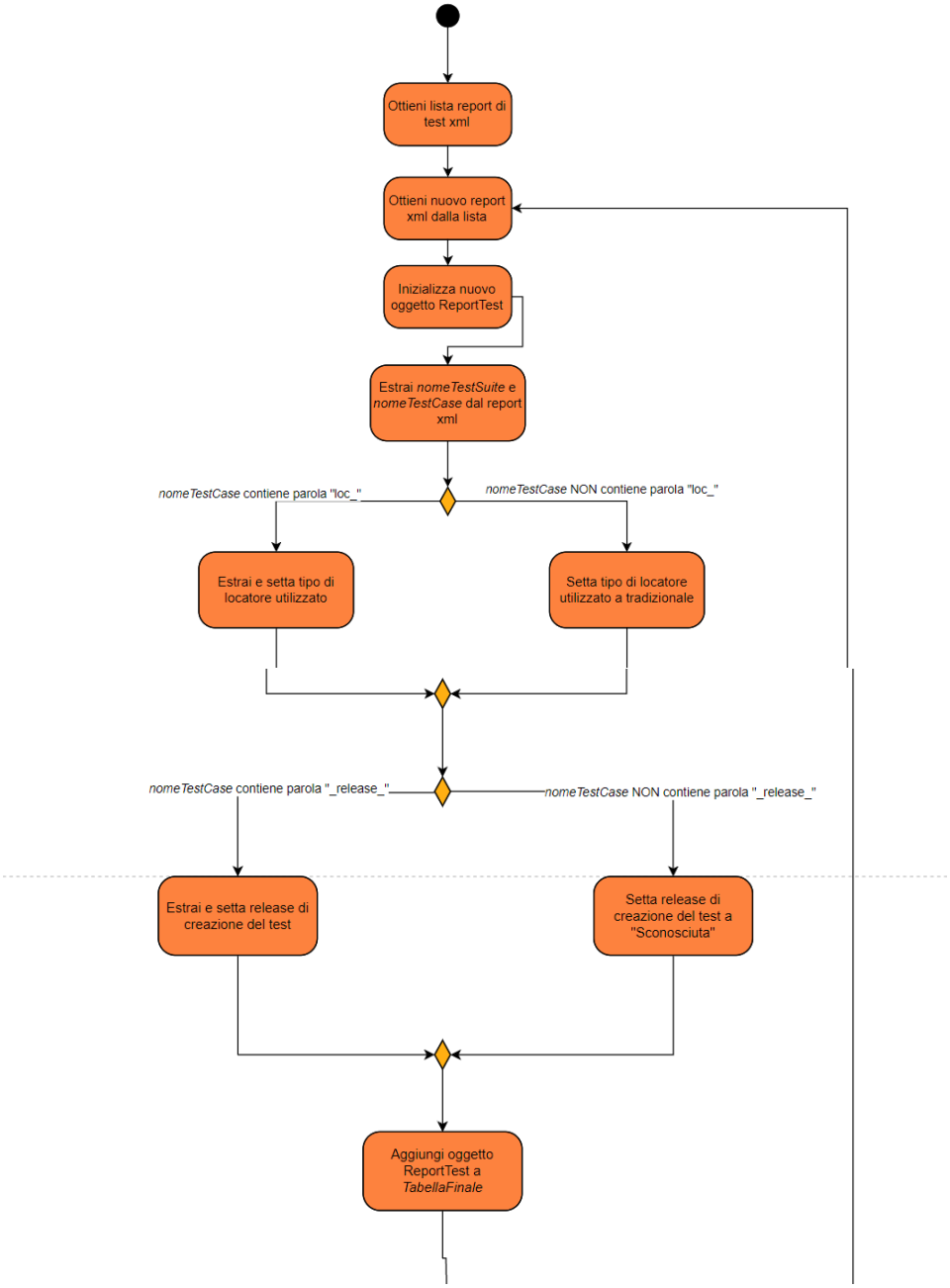


Figura 5.5.3 – 1: *miglioramentoReportTest* - Activity Diagram Parte (1)

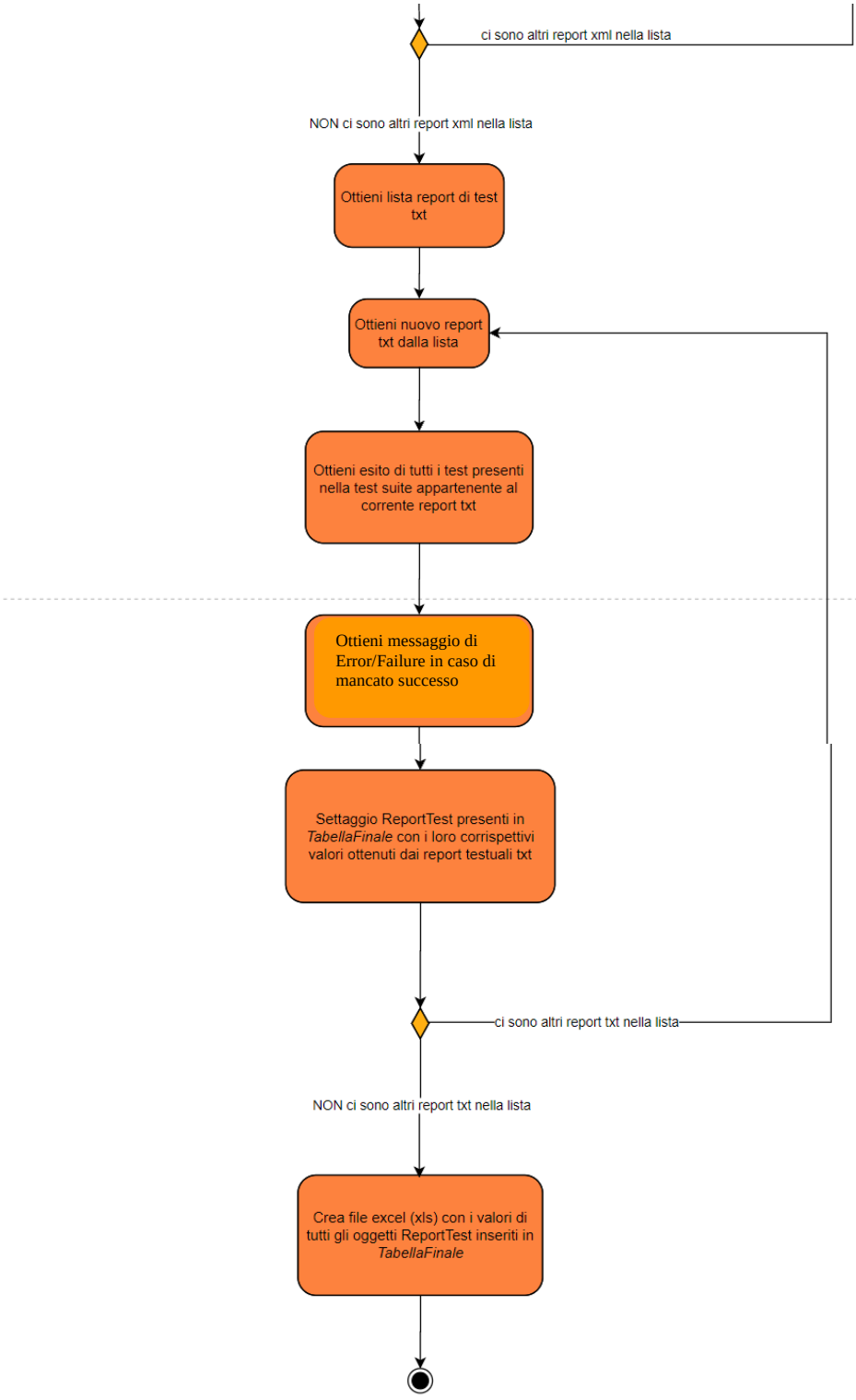


Figura 5.5.3 – 1: *miglioramentoReportTest* - Activity Diagram Parte (2)

Dopo aver visto l'*Activity Diagram*, si riporta il *Class Diagram* riportante la struttura delle entità coinvolte nel processo di miglioramento dei report.

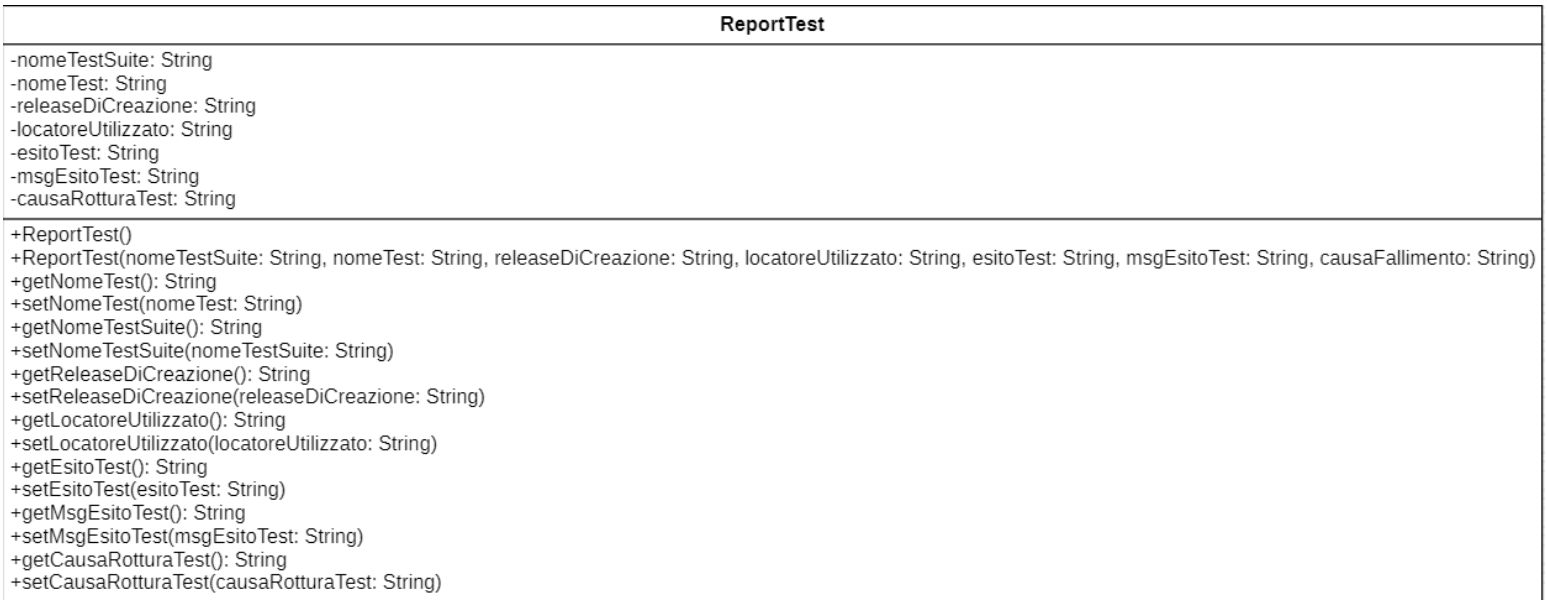


Figura 5.5.3 – 3: Class Diagram - "ReportTest"

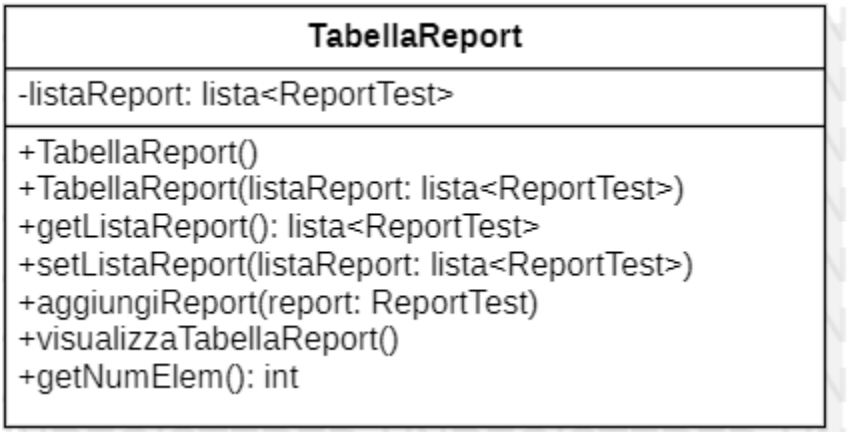


Figura 5.5.3 – 4: Class Diagram - "TabellaReport"

NB: Per non rendere troppo onerosa la trattazione, si è preferito non inserire l'implementazione java del file "*MiglioramentoReportTestApplication.java*" a causa della sua eccessiva lunghezza.

A valle dei diagrammi UML analizzati si è compreso il funzionamento logico del programma *miglioramentoReportTest*, che partendo da vari report separati in formato txt e xml ne genera un unico report complessivo in formato *xls*.

N.B. Per quanto riguarda le convenzioni utilizzate dal progetto, vi sono degli standard da rispettare nel dare i nomi ai test-case:

- Nel caso si voglia specificare la release di creazione di un test-case, è necessario inserire al termine del nome del caso di test la sottostringa "_release_" seguita dal numero della release di creazione (e.g. 1_0). Se non viene specificata esplicitamente la release di creazione di un test, questa viene automaticamente posta a "Sconosciuta".
- Nel caso si voglia specificare il tipo di locatore utilizzato da un test-case, è necessario che il nome del caso di test contenga la sottostringa "loc_" subito prima della sottostringa "_release_". In questo modo la sottostringa compresa tra "loc_" e "_release_" verrà settata come locatore utilizzato. Nel caso in cui la sottostringa inserita abbia come valore "Hooks", questa verrà automaticamente tradotta nell'alias "test-hooks". Se non viene specificato alcun tipo di locatore in un test, questo viene automaticamente settato a "tradizionale".

Vediamo infine un esempio di file di report finale in formato excel (xls), generato dal programma "*miglioramentoReportTest*", attraverso una serie di screenshot:

	A	B
1	TEST SUITE	TEST CASE
2	it.catalogo.test.TestSuiteErrorFailureSuccessProva1	testLocalHost2_loc_Hooks_release_1_2
3	it.catalogo.test.TestSuiteErrorFailureSuccessProva1	provaSuWikipedia_release_1_0
4	it.catalogo.test.TestSuiteErrorFailureSuccessProva1	testProva_loc_genericoLocatore_release_1_2
5	it.catalogo.test.TestSuiteProva2	testLocalHost_loc_Hooks_release_1_0
6	it.catalogo.test.TestSuiteProva2	provaSuWikipedia_release_1_1
7	it.catalogo.test.TestSuiteProva3	provaSuWikipedia_loc_GancioAlternativoProva_release_1_4
8	it.catalogo.test.TestSuiteProva3	testSuLocalHostProva1_release_1_0
9		

Figura 5.5.3 – 5: TabellaFinale.xls – Prime due colonne

C	D
RELEASE DI CREAZIONE	LOCATORE
1_2	test-hooks
1_0	tradizionale
1_2	genericoLocatore
1_0	test-hooks
1_1	tradizionale
1_4	GancioAlternativoProva
1_0	tradizionale

Figura 5.5.3 – 6: TabellaFinale.xls – Terza e quarta colonna

E	F	G
ESITO TEST	MESSAGGIO ESITO TEST	CAUSA ROTTURA TEST
Presenta Failures	org.opentest4j.AssertionFailedError: expected: <Violoncello> but was: <Batteria>	Causa di fallimento obsolescenza/fragilità (specificare a mano)
Presenta un Error	org.openqa.selenium.NoSuchElementException:	Causa di errore obsolescenza/fragilità (specificare a mano)
Superato	-	-
Superato	-	-
Superato	-	-
Superato	-	-
Superato	-	-

Figura 5.5.3 – 7: TabellaFinale.xls – Ultime tre colonne

5.6. Correzione/Eliminazione test rotti

Ricordando lo schema del procedimento operativo riportato nel primo *Activity Diagram*:

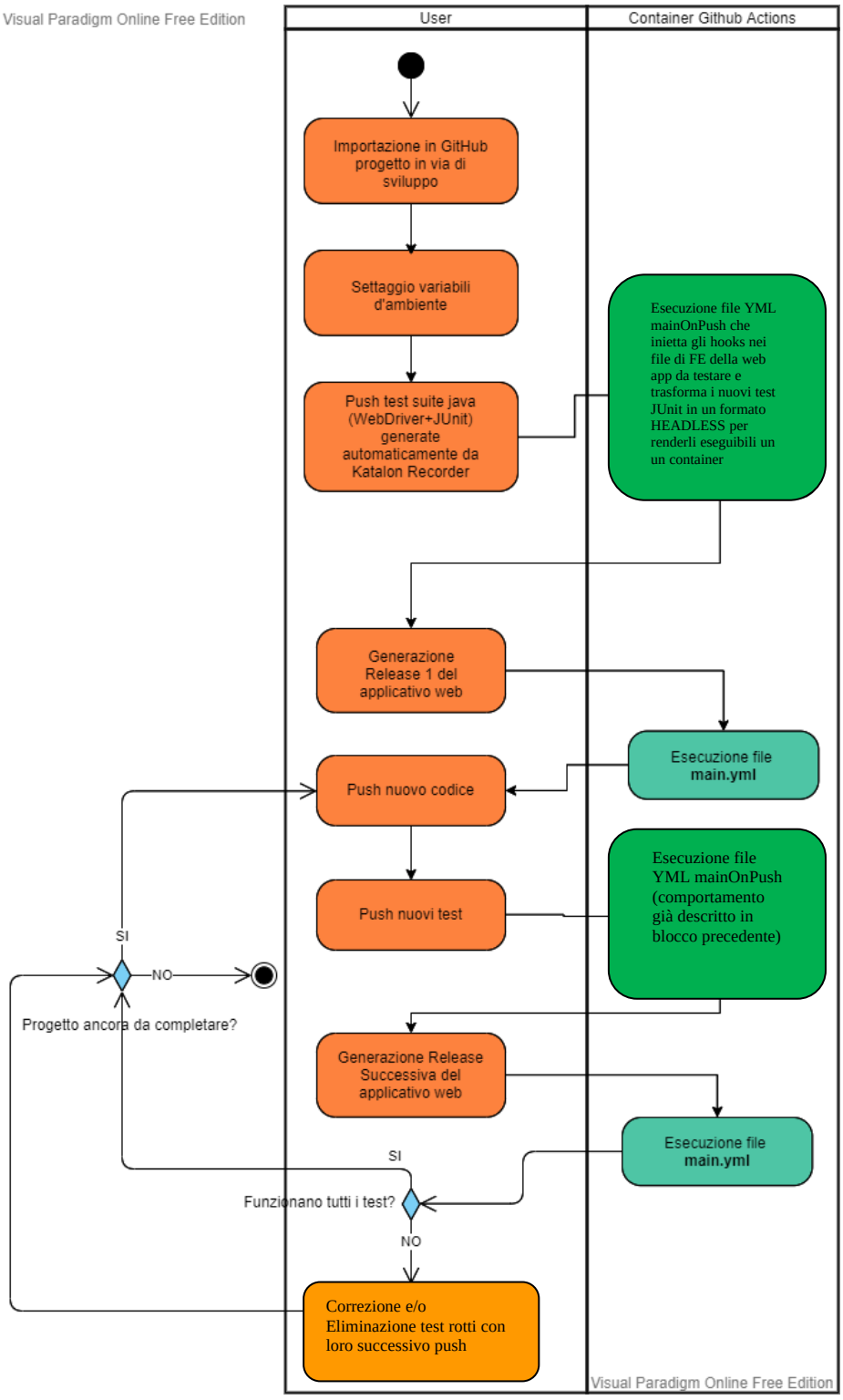


Figura 5.6 – 1: Activity Diagram "*Procedimento Operativo*"

Notiamo che dopo l'esecuzione del file *main.yml*, nel caso in cui non funzionino tutti i testi si giunge nell'activity "*Correzione o Eliminazione test rotti*".

Per poter verificare se ci siano test rotti, il modo più efficace è andare a consultare il report excel (xls) che viene generato e pushato al termine dell'esecuzione del job in *main.yml*.

Quindi ogni volta che viene rilasciata una nuova release, l'esecuzione automatica del file *main.yml* va a generare in una cartella associata al tag identificativo della release il report complessivo, contenente l'esito di tutti i file di test.

A questo punto, basta semplicemente andare a controllare nella colonna *ESITO TEST*, quali test hanno la dicitura "Presenta Failures" o "Presenta un Error".

A questo punto leggendo i valori della colonna *MESSAGGIO ESITO TEST* per tutti i test rotti, è molto più semplice capire se la rottura del test è stata causata da obsolescenza oppure fragilità, andando ad inserire manualmente la causa all'interno della colonna *CAUSA ROTTURA TEST*.

NB: Si ricorda che un test si rompe per obsolescenza quando la funzionalità che va a testare non è più presente nella nuova versione del software, mentre si rompe per fragilità se non riesce più a riconoscere i corretti elementi della UI con cui interagire a causa di cambiamenti dell'interfaccia. A questi andrebbero aggiunti anche i test che falliscono semplicemente a causa di un bug del codice del programma, ma si suppone che le correzioni del codice siano parte integrante delle attività di commit tra una release e l'altra.

A questo punto prima di proseguire con il push di nuovo codice, è necessario che tutti i test diventino funzionanti, andando a correggere oppure (dove non fosse possibile) ad eliminare i test rotti.

5.7. Considerazione sul processo Sperimentale

Sperimentalmente l'obiettivo di questa tesi di laurea è quello di utilizzare gli strumenti sviluppati per effettuare dei confronti tra la robustezza di diversi localizzatori.
Ciò può essere effettuato andando ad implementare, durante il ciclo di sviluppo dell'applicazione web di riferimento, per ogni release del progetto due insiemi di test suite perfettamente identici a meno dei locatori utilizzati.
Il primo insieme di test suite conterrà tutti test case che utilizzino gli "hooks" come localizzatori, il secondo insieme che conterrà invece tutte test suite utilizzanti localizzatori tradizionali.

Tenendo traccia del numero di test rotti in entrambe le versioni, distinguendoli in test rotti per obsolescenza e rotti per fragilità, è quindi possibile fare delle statistiche sulla quantità di test che si rompono a causa della loro fragilità in entrambi i casi.

Quindi a valle di tutto il processo sperimentale sarà possibile stabilire empiricamente quale dei localizzatori utilizzati è più robusto, basandosi sull'assunto che un localizzatore è più robusto se ha un tasso di test rotti per fragilità più basso rispetto ad un altro localizzatore.

Capitolo 6: Processo di sviluppo applicativo web

In questo capitolo si va a descrivere un caso di studio di applicazione della tecnica e degli strumenti realizzati, nel contesto di un processo di sviluppo di un'applicazione web di esempio.

Anche se da un punto di vista astratto la metodologia è applicabile su qualsiasi applicativo che abbia un FE realizzato in Angluar, Freemarker, Smarty, o Twigs, per motivi pratici ci si è dovuto basare su strumenti ben precisi, quali:

- Spring Boot (*back-end*)
- Freemarker (*front-end*)
- Maven (tool di *build-automation*)
- Github (servizio di hosting basato su *Git*)
- Github Actions (*CI/CD* platform)

6.1. Implementazione Versione 1

La descrizione di questa prima versione sarà quella più prolissa, dato che in questa sezione verrà spiegata la struttura base dell'applicativo software che rimarrà l'ossatura di tutte le successive versioni, a meno di qualche piccolo cambiamento grafico.

6.1.1. Package Model

All'interno del package model si trovano le classi rappresentati gli oggetti di interesse all'interno dell'applicativo.

In questa prima versione, vi si trova solamente la classe Prodotto, costituita da quattro attributi:

Id: un intero che rappresenta in modo univoco una singola entità di tipo Prodotto

Nome: una stringa, rappresentate il nome di un prodotto

Descrizione: una stringa, che dia una descrizione più accurata sulle caratteristiche del prodotto

Prezzo: un intero, rappresentante il prezzo di vendita di un generico prodotto.

Vediamone l'implementazione java, contenuta all'interno del file Prodotto.java:

```java
package it.catalogo.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Prodotto {

	@Id
	@GeneratedValue(strategy= GenerationType.IDENTITY)
	private Integer id;

	@Column(name = "nome")
	private String nome;

	@Column(name = "descrizione")
	private String descrizione;

	@Column(name = "prezzo")
	private Integer prezzo;




	public Prodotto() {
		super();
	}
	public Prodotto(String nome, String descrizione, Integer prezzo) {
		super();
		this.nome = nome;
		this.descrizione = descrizione;
		this.prezzo = prezzo;
	}
	public Integer getId() {
		return id;
	}
	public void setId(Integer id) {
		this.id = id;
	}
	public String getNome() {
		return nome;
	}
	public void setNome(String nome) {
		this.nome = nome;
	}
	public String getDescrizione() {
		return descrizione;
	}
	public void setDescrizione(String descrizione) {
		this.descrizione = descrizione;
```

```
	}
	public Integer getPrezzo() {
		return prezzo;
	}
	public void setPrezzo(Integer prezzo) {
		this.prezzo = prezzo;
	}

}
```

Tabella 6.1.1 - 1: file Prodotto.java

6.1.2. File Properties

All'interno del file "application.properties" del progetto Spring-Boot è specificata la tipologia di database nel quale vengono memorizzate le istanze del model, indicando anche url, username e password per l'accesso al db.

Inoltre sempre nel suddetto file viene specificato che come front-end viene utilizzato il template engine *freemarker*.

V

```
spring.datasource.url = jdbc:mysql://bcburz79apide5jfo804-mysql.services.clever-cloud.com:3306/bcburz79apide5jfo804
spring.datasource.username = **********
spring.datasource.password = **********
spring.datasource.driver-class-name = com.mysql.cj.jdbc.Driver

spring.freemarker.prefix=
spring.freemarker.suffix=.ftl
spring.freemarker.template-loader-path=classpath:/templates/
```

e

d

i

a

m

o

quindi la sua implementazione:

Tabella 6.1.2 – 1: file application.properties

6.1.3 Package Repository

Il package Repository viene utilizzato per utilizzare le API della CrudRepository, dove CRUD sta per le operazioni di base su database:

Create – Read – Update – Delete

Infatti andando a definire delle apposite interfacce che vadano ad estendere la classe CrudRepository, è possibile utilizzare dei metodi di CRUD per le nostre classi del model senza andare ad implementarli.

Dato che in questa prima versione del software abbiamo una sola classe all'interno del model, si è definita una sola interfaccia all'interno del package Repository, che ci permetta di utilizzare le API di CRUD per gli oggetti di tipo prodotto.

Vediamo quindi l'implementazione dell'interfaccia ProdottoRepository:

```
package it.catalogo.repository;

import org.springframework.data.repository.CrudRepository;

import it.catalogo.model.Prodotto;

public interface ProdottoRepository extends CrudRepository<Prodotto, Integer>{
        }
```

Tabella 6.1.3 – 1: file ProdottoRepository.java

Da come si può notare, l'interfaccia prende in ingresso tra le parentesi angolari due argomenti:

- La classe Prodotto, ovvero la classe per la quale si vanno ad utilizzare le API di CRUD della CrudRepository.
- Il tipo Integer, che sta ad indicare che i Prodotti sono identificati univocamente tramite una chiave Integer, che in realtà corrisponde all'attributo Id.

In questo modo è possibile accedere alle API senza implementarle, potendo utilizzare metodi quali:

- save(Prodotto): serve per creare o aggiornare un oggetto, in questo caso di tipo prodotto, presente nel repository/database
- findByAttribute(Integer): implementazione di default di una select che va a filtrare per lo specifico attributo, in mase al nome dato al metodo. Prende in ingresso il valore dell'attributo da cercare.
- deleteByAttribute(Integer): implementazione di default di una delete, che va a cancellare l'istanza che presenta il valore in ingresso dell'attributo specificato nel nome del metodo.

Andando ad estendere la classe *CrudRepository* è quindi possibile utilizzare metodi di CRUD su tutti gli attributi degli oggetti di interesse, cambiando semplicemente il nome

dei metodi utilizzati secondo una appropriata sintassi, senza andare veramente ad implementarli.

6.1.4. Package Controller

All'interno del package Controller sono implementati i metodi che vanno a mappare le funzionalità dell'applicativo sul protocollo http in specifici url.

Andiamo prima a vedere l'implementazione della classe ProdottoController, per poi andare a spiegarne ogni metodo.

```
package it.catalogo.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;

import it.catalogo.repository.*;
import it.catalogo.model.*;

@Controller
@RequestMapping("/")
public class ProdottoController {

	@Autowired
	private ProdottoRepository repo;

	//serve localhost:8080/lista
	@ResponseBody
	@GetMapping("/lista")
	public Iterable<Prodotto> list() {

		Iterable<Prodotto> prods = repo.findAll();

		//Dovrebbe essere una funzione inline (lambda function)
		prods.forEach((Prodotto p) -> {
			System.out.println(p.getNome());
		});

		return prods;
	}
```

```
		//serve localhost:8080/prodotti
		@GetMapping("/prodotti")
		public ModelAndView index(@RequestParam(required = false) String id,
ModelMap mm) {

			if(id != null) {
				Prodotto prod = new Prodotto();
				prod = repo.findById(Integer.parseInt(id)).get();
				mm.addAttribute("prodottoDaModificare",  prod);
			}

			Iterable<Prodotto> prods = repo.findAll();

			return new ModelAndView("indexProdotti", "listaProdotti", prods);
		}

		@PostMapping("/add")
		public String add(@ModelAttribute("datiProdotto") Prodotto p) {
			repo.save(p);
			return "redirect:/prodotti";
		}


		@PostMapping("/update")
		public String update(@ModelAttribute("datiProdotto") Prodotto p) {
			repo.save(p);
			return "redirect:/prodotti";
		}


		@GetMapping("/delete")
		public String delete(@RequestParam("id") String idProdotto) {
			if(idProdotto != null)
				repo.deleteById(Integer.parseInt(idProdotto));

			return "redirect:/prodotti";
		}
}
```

Tabella 6.1.4. – 1: file ProdottoController.java

I metodi all'interno del seguente controller sono:

- *@ResponseBody @GetMapping("/lista")* public Iterable<Prodotto> list():

Il metodo *list* serve le richieste all'url

localhost:8080/lista

Quando viene fatta una richiesta all'url, allora stampa a video nella console i nomi di tutti i prodotti presenti nel repository, e ritorna nella web page un file JSON contenente la lista di tutti i prodotti presenti nel database con i valori di tutti i loro attributi.

- *@GetMapping("/prodotti")* public ModelAndView index(@RequestParam(required = false) String id, ModelMap mm):
  Il metodo index mappa le richieste all'url
  localhost:8080/prodotti
  Quando viene fatta una richiesta all'url, ritorna il file freemarker *indexProdotti.ftl* contenente la struttura e la logica della pagina principale dell'applicativo web.

  Se invece la richiesta è effettuata con il parametro id:
  localhost:8080/prodotti?id=numero_intero
  Allora viene posto a *true* l'attributo *prodottoDaModificare*, e la pagina principale sarà caricata in modo tale che si possano modificare i valori del prodotto del quale si era specificato l'identificativo.

- *@PostMapping("/add")* public String add(@ModelAttribute("datiProdotto") Prodotto p):
  Il metodo add può servire solamente richieste di tipo Post, quindi si attiva solamente alla richiesta di uno specifico form di registrazione prodotto.
  Quando viene invocato, tutto ciò che fa è prendere i valori del prodotto da aggiungere dagli appositi form, aggiungerlo al repository ed infine fare il redirect alla pagina principale
  localhost:8080/prodotti

- *@PostMapping("/update")* public String update(@ModelAttribute("datiProdotto") Prodotto p):
  Anche il metodo di update serve solamente richieste di tipo Post, attivandosi solamente alla richiesta di uno specifico form.
  Una volta invocato, ciò che fa è aggiornare i valori del prodotto da aggiornare con i valori dei campi di appositi form, infine fa il redirect alla pagina principale
  localhost:8080/prodotti

- *@GetMapping("/delete")* public String delete(@RequestParam("id") String idProdotto):
  Quando viene invocato lo specifico url
  localhost:8080/delete?id=numero_id

Allora viene eliminato dal repository il prodotto avente identificativo *numero_id*, il metodo è solitamente chiamato automaticamente alla pressione di un button di cancellazione nella sezione del file freemarker mostrante la lista dei prodotti presenti nel catalogo.

6.1.5. File Freemarker

Il file freemarker della pagina principale, che viene mappato a localhost:8080/prodotti, è intitolato indexProdotti.ftl.

Di questo file ne esistono però due versioni differenti, ma semanticamente identiche, dove in una delle due sono stati iniettati degli hooks come localizzatori.

Quindi anche se questi due file avranno una implementazione apparentemente diversa, hanno in realtà lo stesso contenuto informativo, dato che ciò che cambia sono solo gli identificatori degli elementi grafici dei file ftl.

Vediamo prima l'implementazione del file *indexProdottiOriginaleNoHooks.ftl*, contenente la versione con localizzatori tradizionali:

```
<!DOCTYPE html>
<html>
		<head>
			<title>Catalogo prodotti</title>
			<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.min.css"
integrity="sha384-B0vP5xmATw1+K9KRQjQERJvTumQW0nPEzvF6L/Z6nronJ3oUOFUFpCjEUQouq2+l"
crossorigin="anonymous">
			<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" crossorigin="anonymous"></script>
			<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.bundle.min.js"
integrity="sha384-Piv4xVNRyMGpqkS2by6br4gNJ7DXjqk09RmUpJ8jgGtD7zP9yug3goQfGII0yAns"
crossorigin="anonymous"></script>
		</head>
		<body>
			<h1>Catalogo prodotti</h1>


			<#if prodottoDaModificare??>
			<h2>Modifica del prodotto - ${prodottoDaModificare.nome}</h2>
				<div style="margin 20px;">
					<form method="POST" action="update" id="datiProdotto">
						<input type="hidden" name="id"
value="${prodottoDaModificare.id}" />
						<div>
							<label for="nome">Nome</label>
```

```
                                                    <input type="text" name="nome" id="nome"
value="${prodottoDaModificare.nome}" />
                                                </div>
                                                <div>
                                                    <label for="descrizione">Descrizione</label>
                                                    <input type="text" name="descrizione"
id="descrizione" value="${prodottoDaModificare.descrizione}" />
                                                </div>
                                                <div>
                                                    <label for="prezzo">Prezzo</label>
                                                    <input type="number" name="prezzo" id="prezzo"
value="${prodottoDaModificare.prezzo}" />
                                                </div>
                                                <div>
                                                    <input type="submit" name="invia" value="Salva
modifiche" />
                                                </div>


                                            </form>
                                        </div>
                                    <#else>
                                    <h2>Nuovo prodotto</h2>
                                        <div style="margin 20px;">
                                            <form method="POST" action="add" id="datiProdotto">
                                                <div>
                                                    <label for="nome">Nome</label>
                                                    <input type="text" name="nome" id="nome"
value="" />
                                                </div>
                                                <div>
                                                    <label for="descrizione">Descrizione</label>
                                                    <input type="text" name="descrizione"
id="descrizione" value="" />
                                                </div>
                                                <div>
                                                    <label for="prezzo">Prezzo</label>
                                                    <input type="number" name="prezzo" id="prezzo"
value="" />
                                                </div>
                                                <div>
                                                    <input type="submit" name="invia"
value="Aggiungi" />
                                                </div>


                                            </form>
                                        </div>

                                    </#if>


                                    <hr>
```

```
                    <h2>Lista prodotti</h2>
                    <div>
                        <table>
                            <thead>
                                <tr>
                                    <th>Nome</th>
                                    <th>Descrizione</th>
                                    <th>Prezzo</th>
                                    <th>Azioni</th>
                                </tr>
                            </thead>
                            <tbody>
                                <#list listaProdotti as prodotto>
                                <tr>
                                    <td>${prodotto.nome}</td>
                                    <td>${prodotto.descrizione}</td>
                                    <td>${prodotto.prezzo}</td>
                                    <td>
                                        <a
href="delete?id=${prodotto.id}">Elimina</a>
                                        <a
href="prodotti?id=${prodotto.id}">Modifica</a>
                                    </td>
                                </tr>
                                </#list>
                            </tbody>
                        </table>
                    </div>
                </body>
</html>
```

Tabella 6.1.5 - 1: file indexProdottiOriginaleNoHooks.ftl

Successivamente vediamo l'implementazione contenente l'iniezione dei ganci nel codice, salvata nel file *indexProdotti.ftl*:

```
<!DOCTYPE html>
<html>
        <head>
            <title x230953393589-x-test-tpl-1>Catalogo prodotti</title>
            <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.min.css" integrity="sha384-
B0vP5xmATw1+K9KRQjQERJvTumQW0nPEzvF6L/Z6nronJ3oUOFUFpCjEUQouq2+l"
crossorigin="anonymous" x230953393589-x-test-tpl-2>
            <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>
            <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-
Piv4xVNRyMGpqkS2by6br4gNJ7DXjqk09RmUpJ8jgGtD7zP9yug3goQfGII0yAns"
crossorigin="anonymous"></script>
        </head>
        <body>
            <h1 x230953393589-x-test-tpl-3>Catalogo prodotti</h1>
```

```
                    <#if prodottoDaModificare??>
                    <h2 x230953393589-x-test-tpl-4>Modifica del prodotto -
${prodottoDaModificare.nome}</h2>
                        <div style="margin 20px;" x230953393589-x-test-tpl-5>
                            <form method="POST" action="update" id="datiProdotto"
x230953393589-x-test-hook-6>
                                <input type="hidden" name="id"
value="${prodottoDaModificare.id}" x230953393589-x-test-hook-7 />
                                <div x230953393589-x-test-hook-8>
                                    <label for="nome" x230953393589-x-
test-hook-9>Nome</label>
                                    <input type="text" name="nome"
id="nome" value="${prodottoDaModificare.nome}" x230953393589-x-test-hook-10 />
                                </div>
                                <div x230953393589-x-test-hook-11>
                                    <label for="descrizione"
x230953393589-x-test-hook-12>Descrizione</label>
                                    <input type="text" name="descrizione"
id="descrizione" value="${prodottoDaModificare.descrizione}" x230953393589-x-test-hook-13 />
                                </div>
                                <div x230953393589-x-test-hook-14>
                                    <label for="prezzo" x230953393589-x-
test-hook-15>Prezzo</label>
                                    <input type="number" name="prezzo"
id="prezzo" value="${prodottoDaModificare.prezzo}" x230953393589-x-test-hook-16 />
                                </div>
                                <div x230953393589-x-test-hook-17>
                                    <input type="submit" name="invia"
value="Salva modifiche" x230953393589-x-test-hook-18 />
                                </div>


                            </form>
                        </div>
                    <#else>
                    <h2 x230953393589-x-test-tpl-19>Nuovo prodotto</h2>
                        <div style="margin 20px;" x230953393589-x-test-tpl-20>
                            <form method="POST" action="add" id="datiProdotto"
x230953393589-x-test-hook-21>
                                <div x230953393589-x-test-hook-22>
                                    <label for="nome" x230953393589-x-
test-hook-23>Nome</label>
                                    <input type="text" name="nome"
id="nome" value="" x230953393589-x-test-hook-24 />
                                </div>
                                <div x230953393589-x-test-hook-25>
                                    <label for="descrizione"
x230953393589-x-test-hook-26>Descrizione</label>
                                    <input type="text" name="descrizione"
id="descrizione" value="" x230953393589-x-test-hook-27 />
                                </div>
                                <div x230953393589-x-test-hook-28>
                                    <label for="prezzo" x230953393589-x-
```

```
test-hook-29>Prezzo</label>
                                                    <input type="number" name="prezzo"
id="prezzo" value="" x230953393589-x-test-hook-30 />
                                                </div>
                                                <div x230953393589-x-test-hook-31>
                                                    <input type="submit" name="invia"
value="Aggiungi" x230953393589-x-test-hook-32 />
                                                </div>


                                            </form>
                                        </div>

                                    </#if>


                                    <hr x230953393589-x-test-tpl-33>

                                    <h2 x230953393589-x-test-tpl-34>Lista prodotti</h2>
                                    <div x230953393589-x-test-tpl-35>
                                        <table x230953393589-x-test-hook-36>
                                            <thead x230953393589-x-test-hook-37>
                                                <tr x230953393589-x-test-hook-38>
                                                    <th x230953393589-x-test-hook-
39>Nome</th>
                                                    <th x230953393589-x-test-hook-
40>Descrizione</th>
                                                    <th x230953393589-x-test-hook-
41>Prezzo</th>
                                                    <th x230953393589-x-test-hook-
42>Azioni</th>
                                                </tr>
                                            </thead>
                                            <tbody x230953393589-x-test-hook-43>
                                                <#list listaProdotti as prodotto>
                                                <tr x230953393589-x-test-hook-44>
                                                    <td x230953393589-x-test-hook-
45>${prodotto.nome}</td>
                                                    <td x230953393589-x-test-hook-
46>${prodotto.descrizione}</td>
                                                    <td x230953393589-x-test-hook-
47>${prodotto.prezzo}</td>
                                                    <td x230953393589-x-test-hook-48>
                                                        <a
href="delete?id=${prodotto.id}" x230953393589-x-test-hook-49>Elimina</a>
                                                        <a
href="prodotti?id=${prodotto.id}" x230953393589-x-test-hook-50>Modifica</a>
                                                    </td>
                                                </tr>
                                                </#list>
                                            </tbody>
                                        </table>
                                    </div>
                                </body>
```

```
</html>
```

Da come si può notare dalle due versioni appena viste, l'unica differenza lessicale esistente è la presenza nel secondo file di una stringa aggiuntiva (hooks) all'interno di ogni tag del file *ftl*.

L'importanza semantica di questa aggiunta assume però rilevanza solamente quando si va a valutare la robustezza di suite di test, le quali possono meglio identificare gli elementi della UI anche a valle di cambiamenti di stile di interfaccia.

Infatti l'obiettivo del seguente lavoro è proprio quello di confrontare test suite che utilizzano localizzatori con hooks con test suite che utilizzano localizzatori tradizionali.

6.1.6. Avvio Applicazione

L'applicativo Spring Boot viene avviato all'interno della classe "CatalogoProdottiBootApplication", tramite le semplici righe di codice:

```
package it.catalogo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

@SpringBootApplication
@EnableJpaRepositories(basePackages="it.catalogo.repository")
public class CatalogoProdottiBootApplication {
	public static void main(String[] args) {
	SpringApplication.run(CatalogoProdottiBootApplication.class, args);
	}
}
```

Una volta avviata l'applicazione, tramite un qualsiasi browser è possibile collegarsi al seguente url:

localhost:8080/prodotti

visualizzando la web page principale, costituita dal file freemarker *indexProdotti.ftl*.

6.1.7. Istruzioni operative di configurazione

Per poter configurare opportunamente l'ambiente di sviluppo, vediamo l'insieme completo di istruzioni operative da utilizzare per installare ed avviare l'applicazione web.

Innanzitutto, la prima cosa da fare è creare e configurare opportunamente le 6 variabili d'ambiente all'interno dell'environment *envForGithubActions*. Si riportano qui i valori reali, che sono stati inseriti una volta all'inizio del processo:

```
EMAIL_ACCOUNT_GITHUB: t*********@gmail.com
NOME_ACCOUNT_GITHUB: g*********
PASSWORD_ACCOUNT_GITHUB: *********
FE_EXTENSION_TYPE: .ftl
GRAMMAR_TYPE: freemarker
DIR_FILE_FE: /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale/insert-here-your-web-app/catalogoProdottiBoot
```

Per poter poi utilizzare gli strumenti automatici di iniezione degli hooks è necessario installare node (versione 10) e npm, in ubuntu (ubuntu 20.04.03, LTS in data 01/02/2022) è possibile farlo tramite i seguenti comandi:

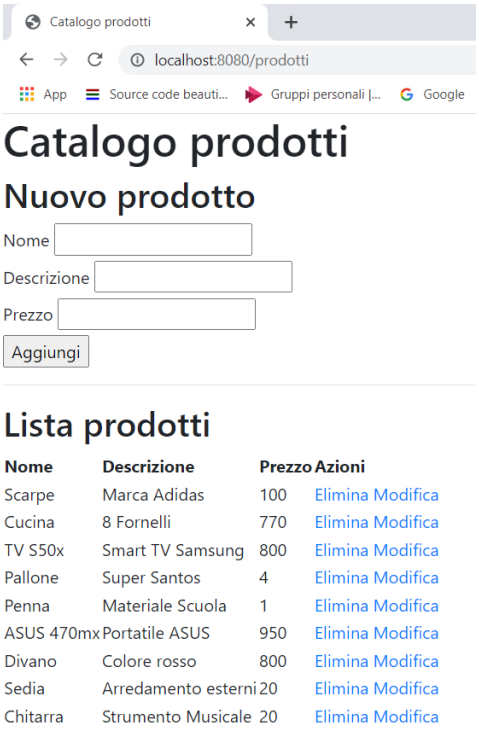
```
sudo apt update
sudo apt -y install curl dirmngr apt-transport-https lsb-release ca-certificates
curl -sL https://deb.nodesource.com/setup_10.x | sudo bash
sudo apt install nodejs
sudo npm cache clean -f
sudo npm install -g n
sudo n stable
sudo n 10.18.0

npm install
npm install bcrypt
npm fund
```

Invece per eseguire l'applicativo web, dopo averne effettuato il packaging, è necessario recarsi nella sotto directory *target* del progetto ed eseguire il seguente comando:

```
java -jar NOME_JAR_WEBAPR.jar
```

6.1.8. Interfaccia applicativo

Dopo l'esecuzione dell'applicativo, andando tramite browser all'url *localhost:8080/prodotti* si avrà davanti la seguente interfaccia grafica:



Nella sezione in alto è possibile compilare i tre form (*Nome*, *Descrizione*, *Prezzo*) per poter aggiungere un nuovo prodotto al catalogo (l'*id* è settato automaticamente secondo una strategia incrementale).

Nella sezione in basso è invece visibile la lista dei prodotti attualmente presenti nel catalogo, e quindi memorizzati nel database *CleverCloud*.

Da come è possibile notare, affianco ogni elemento della lista sono presenti due button: "*Elimina*" e "*Modifica*".

Premendo su "*Elimina*", verrà servita la richiesta del metodo *delete* al path *localhost:8080/delete?id=x*, dove al posto della *x* avremo l'*id* del prodotto della lista che si vuole eliminare.

Premendo invece su "*Modifica*", verrà servita la richiesta del path *localhost:8080/prodotti?id=x*, dove al posto della *x* si avrà l'*id* del prodotto che si intende modificare. A questo punto si caricheranno automaticamente i valori del suddetto prodotto all'interno dei tre form in alto (*Nome*, *Descrizione*, *Prezzo*), cambiandone il valore e premendo su aggiungi avverà quindi l'effettivo *update* del prodotto desiderato.

6.2. Release v1.0

Dato che stiamo lavorando contemporaneamente con due progetti identici, dei quali però solamente uno presenta gli hooks iniettati all'interno del file freemarker, sono state rilasciate due differenti versioni 1.0.

Precisamente abbiamo:

- *v1.0-Hooks*
- *v1.0-Tradizionale*

Dato che entrambe le release sono identiche a livello di funzionamento, l'unica cosa che cambierà è che le test suite utilizzate della release *v1.0-Hooks* utilizzeranno i locatori *hooks*.

Le test suite implementate per questa prima release, sono state:

- Una test suite che testa la creazione e la successiva eliminazione dei prodotti creati.

- Una test suite che testa l'update dei prodotti ed il loro successivo ritorno al valore iniziale.

NB: Vale il duplice assunto secondo il quale, lo stato iniziale è comune a tutti i test case ed ogni test case è idempotente. Se un test modifica lo stato iniziale, prima di terminare lo va a ripristinare.

6.2.1. Release v1.0-Hooks

I due test case implementati vanno sotto il nome di:

- testCreateDelete_loc_Hooks_release_1_0
- testUpdate_loc_Hooks_release_1_0

Da come si può notare entrambe rispettano lo standard utilizzato dal programma "*miglioramentoReportTest*", andando a specificare correttamente quale *locatore* utilizzano e qual è la loro *release di creazione*.

Una volta quindi generato il codice dei file di test in maniera automatica (a partire da tecniche di *Capture&Replay*) da *Katalon Recorder* (nel formato *JUnit+WebDriver*), dopo aver dato un nome appropriato ai test case, si vanno ad importare i file di test così ottenuti all'interno del repository github.

A questo punto, dato che nel repository è avvenuto un push, si avvia automaticamente il file yml "*correzioneFormatoTest*".
Il file YAML, da come spiegato nel capitolo precedente, utilizzando un file *jar* implementato ad hoc, va a modificare i file di test JUnit generati automaticamente da Katalon Recorder per trasformarli in un formato idoneo all'esecuzione headless all'interno di un Container Ubuntu.

Quindi dopo aver fatto il push dei test, questi saranno immediatamente trasformati in un formato eseguibile nel container.

Ora non rimane altro che creare una nuova release, identificata dal tag:
v1.0-Hooks
Alla creazione della release si avvia il già precedentemente descritto file *main.yml*, il quale al termine della sua esecuzione va a creare ed a fare il push di un file di report complessivo (in formato xls) di tutti i test eseguiti nella corrente release.

6.2.1.1. File JUnit generati da Katalon Recorder

In questo sottoparagrafo andiamo a visualizzare i file di test che sono stati generati automaticamente da Katalon Recorder (dopo aver impostato manualmente un nome per i test che sia adatto al progetto *miglioramentoReportTest*):

```
package com.example.tests;

import java.util.regex.Pattern;
import java.util.concurrent.TimeUnit;
import org.junit.*;
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.*;
import org.openqa.selenium.*;
import org.openqa.selenium.firefox.FirefoxDriver;
```

```
import org.openqa.selenium.support.ui.Select;

public class SuiteCreateDeleteLocHooks {
  private WebDriver driver;
  private String baseUrl;
  private boolean acceptNextAlert = true;
  private StringBuffer verificationErrors = new StringBuffer();

  @Before
  public void setUp() throws Exception {
    driver = new FirefoxDriver();
    baseUrl = "https://www.google.com/";
    driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
  }

  @Test
  public void testSuiteCreateDeleteLocHooks_loc_Hooks_release_1_0() throws Exception {
    driver.get("http://localhost:8080/prodotti");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
24]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
24]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
24]")).sendKeys("Cellulare");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
27]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
27]")).sendKeys("Prodotto Samsung");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
30]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
30]")).sendKeys("400");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
32]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
24]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
24]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
24]")).sendKeys("Lavagna");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
27]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
27]")).sendKeys("Prodotto Scolastico");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
30]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
30]")).sendKeys("60");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
32]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
24]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
24]")).clear();
```

```
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).sendKeys("Dipinto");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).sendKeys("Dimensioni 60cmx90cm");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).sendKeys("25");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-32]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][12]//*[@x230953393589-x-test-hook-49]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][11]//*[@x230953393589-x-test-hook-49]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][10]//*[@x230953393589-x-test-hook-49]")).click();
  }

  @After
  public void tearDown() throws Exception {
    driver.quit();
    String verificationErrorString = verificationErrors.toString();
    if (!"".equals(verificationErrorString)) {
      fail(verificationErrorString);
    }
  }

  private boolean isElementPresent(By by) {
    try {
      driver.findElement(by);
      return true;
    } catch (NoSuchElementException e) {
      return false;
    }
  }

  private boolean isAlertPresent() {
    try {
      driver.switchTo().alert();
      return true;
    } catch (NoAlertPresentException e) {
      return false;
    }
  }

  private String closeAlertAndGetItsText() {
    try {
      Alert alert = driver.switchTo().alert();
      String alertText = alert.getText();
      if (acceptNextAlert) {
```

```
      alert.accept();
    } else {
      alert.dismiss();
    }
    return alertText;
  } finally {
    acceptNextAlert = true;
  }
 }
}
```

Tabella 6.2.1.1 – 1: file SuiteCreateDeleteLocHooks.java

```
package com.example.tests;

import java.util.regex.Pattern;
import java.util.concurrent.TimeUnit;
import org.junit.*;
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.*;
import org.openqa.selenium.*;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.support.ui.Select;

public class SuiteUpdateLocHooks {
  private WebDriver driver;
  private String baseUrl;
  private boolean acceptNextAlert = true;
  private StringBuffer verificationErrors = new StringBuffer();

  @Before
  public void setUp() throws Exception {
    driver = new FirefoxDriver();
    baseUrl = "https://www.google.com/";
    driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
  }

  @Test
  public void testSuiteUpdateLocHooks_loc_Hooks_release_1_0() throws Exception {
    driver.get("http://localhost:8080/prodotti");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][1]//*[@x230953393589-x-test-hook-50]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-13]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-13]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-13]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-13]")).clear();
```

```
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-13]")).sendKeys("Marca Nike");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).sendKeys("95");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-18]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][2]//*[@x230953393589-x-test-hook-50]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-13]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-13]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-13]")).sendKeys("4 Fornelli");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).sendKeys("380");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-18]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][3]//*[@x230953393589-x-test-hook-50]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-10]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-10]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-10]")).sendKeys("TV S150x");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).sendKeys("900");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-18]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][3]//*[@x230953393589-x-test-hook-50]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-10]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-10]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-10]")).sendKeys("TV S50x");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).clear();
```

```
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).sendKeys("800");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-18]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][2]//*[@x230953393589-x-test-hook-50]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-13]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-13]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-13]")).sendKeys("8 Fornelli");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).sendKeys("770");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-18]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][1]//*[@x230953393589-x-test-hook-50]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-13]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-13]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-13]")).sendKeys("Marca Adidas");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).clear();
    drier.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).sendKeys("100");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-18]")).click();
  }

  @After
  public void tearDown() throws Exception {
    driver.quit();
    String verificationErrorString = verificationErrors.toString();
    if (!"".equals(verificationErrorString)) {
      fail(verificationErrorString);
    }
  }


  private boolean isElementPresent(By by) {
    try {
      driver.findElement(by);
      return true;
    } catch (NoSuchElementException e) {
      return false;
    }
  }
```

```
 private boolean isAlertPresent() {
  try {
   driver.switchTo().alert();
   return true;
  } catch (NoAlertPresentException e) {
   return false;
  }
 }

 private String closeAlertAndGetItsText() {
  try {
   Alert alert = driver.switchTo().alert();
   String alertText = alert.getText();
   if (acceptNextAlert) {
    alert.accept();
   } else {
    alert.dismiss();
   }
   return alertText;
  } finally {
   acceptNextAlert = true;
  }
 }
}
```

Figura 6.2.1.1 – 2: file SuiteUpdateLocHooks.java

6.2.1.2. File JUnit corretti da file *mainOnPush.yml*

In quest'altro sottoparagrafo vediamo invece in che modo sono stati modificati i file di test generati da *Katalon Recorder* da parte del file *mainOnPush.yml*.

Vediamo quindi qui i file di test adattati per l'esecuzione *headless* in un container:

```
//File risulta attualmente aggiornato per webdriver chrome headless!
package it.catalogo.test;

import static org.junit.Assert.fail;
import static org.junit.jupiter.api.Assertions.*;

import java.util.concurrent.TimeUnit;

import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;
```

```
import org.openqa.selenium.Alert;
import org.openqa.selenium.By;
import org.openqa.selenium.NoAlertPresentException;
import org.openqa.selenium.NoSuchElementException;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;

class SuiteCreateDeleteLocHooks {
private static WebDriver driver;
private boolean acceptNextAlert = true;
private static StringBuffer verificationErrors = new StringBuffer();

        @BeforeAll
        public static void setUp() throws Exception {

                // Init chromedriver
                String chromeDriverPath =
"/home/runner/work/WebAppTesi/WebAppTesi/chromedriver_v94_linux64/chromedriver";
                System.setProperty("webdriver.chrome.driver", chromeDriverPath);
                ChromeOptions options = new ChromeOptions();
                options.addArguments("--headless", "--disable-gpu", "--window-
size=1920,1200","--ignore-certificate-errors");
                driver = new ChromeDriver(options);



          driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
        }
  @Test
  public void testSuiteCreateDeleteLocHooks_loc_Hooks_release_1_0() throws Exception {
    driver.get("http://localhost:8080/prodotti");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-
test-hook-24]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-
test-hook-24]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-
test-hook-24]")).sendKeys("Cellulare");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-
test-hook-27]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-
test-hook-27]")).sendKeys("Prodotto Samsung");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-
test-hook-30]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-
test-hook-30]")).sendKeys("400");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-
test-hook-32]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-
test-hook-24]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-
test-hook-24]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-
test-hook-24]")).sendKeys("Lavagna");
```

```
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).sendKeys("Prodotto Scolastico");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).sendKeys("60");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-32]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).sendKeys("Dipinto");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).sendKeys("Dimensioni 60cmx90cm");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).sendKeys("25");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-32]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][12]//*[@x230953393589-x-test-hook-49]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][11]//*[@x230953393589-x-test-hook-49]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][10]//*[@x230953393589-x-test-hook-49]")).click();
  }


 @AfterAll
        public static void tearDown() throws Exception {
          driver.quit();
          String verificationErrorString = verificationErrors.toString();
          if (!"".equals(verificationErrorString)) {
            fail(verificationErrorString);
          }
        }

        private boolean isElementPresent(By by) {
          try {
            driver.findElement(by);
            return true;
          } catch (NoSuchElementException e) {
            return false;
          }
        }
```

```
        private boolean isAlertPresent() {
          try {
            driver.switchTo().alert();
            return true;
          } catch (NoAlertPresentException e) {
            return false;
          }
        }

        private String closeAlertAndGetItsText() {
          try {
            Alert alert = driver.switchTo().alert();
            String alertText = alert.getText();
            if (acceptNextAlert) {
              alert.accept();
            } else {
              alert.dismiss();
            }
            return alertText;
          } finally {
            acceptNextAlert = true;
          }
        }
}
```

Tabella 6.2.1.2 – 1: file SuiteCreateDeleteLocHooks.java adattato per i test Headless

```
//File risulta attualmente aggiornato per webdriver chrome headless!
package it.catalogo.test;

import static org.junit.Assert.fail;
import static org.junit.jupiter.api.Assertions.*;

import java.util.concurrent.TimeUnit;

import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;
import org.openqa.selenium.Alert;
import org.openqa.selenium.By;
import org.openqa.selenium.NoAlertPresentException;
import org.openqa.selenium.NoSuchElementException;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;

class SuiteUpdateLocHooks {
private static WebDriver driver;
private boolean acceptNextAlert = true;
private static StringBuffer verificationErrors = new StringBuffer();

        @BeforeAll
```

```
        public static void setUp() throws Exception {

                // Init chromedriver
                String chromeDriverPath =
"/home/runner/work/WebAppTesi/WebAppTesi/chromedriver_v94_linux64/chromedriver";
                System.setProperty("webdriver.chrome.driver", chromeDriverPath);
                ChromeOptions options = new ChromeOptions();
                options.addArguments("--headless", "--disable-gpu", "--window-
size=1920,1200","--ignore-certificate-errors");
                driver = new ChromeDriver(options);



          driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
        }
  @Test
  public void testSuiteUpdateLocHooks_loc_Hooks_release_1_0() throws Exception {
    driver.get("http://localhost:8080/prodotti");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-
test-hook-44][1]//*[@x230953393589-x-test-hook-50]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-
test-hook-13]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-
test-hook-13]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-
test-hook-13]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-
test-hook-13]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-
test-hook-13]")).sendKeys("Marca Nike");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-
test-hook-16]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-
test-hook-16]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-
test-hook-16]")).sendKeys("95");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-
test-hook-18]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-
test-hook-44][2]//*[@x230953393589-x-test-hook-50]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-
test-hook-13]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-
test-hook-13]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-
test-hook-13]")).sendKeys("4 Fornelli");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-
test-hook-16]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-
test-hook-16]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-
test-hook-16]")).sendKeys("380");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-
test-hook-18]")).click();
```

```
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][3]//*[@x230953393589-x-test-hook-50]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-10]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-10]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-10]")).sendKeys("TV S150x");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).sendKeys("900");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-18]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][3]//*[@x230953393589-x-test-hook-50]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-10]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-10]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-10]")).sendKeys("TV S50x");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).sendKeys("800");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-18]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][2]//*[@x230953393589-x-test-hook-50]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-13]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-13]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-13]")).sendKeys("8 Fornelli");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).sendKeys("770");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-18]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][1]//*[@x230953393589-x-test-hook-50]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-13]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-13]")).clear();
```

```
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-13]")).sendKeys("Marca Adidas");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).sendKeys("100");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-18]")).click();
  }


 @AfterAll
        public static void tearDown() throws Exception {
          driver.quit();
          String verificationErrorString = verificationErrors.toString();
          if (!"".equals(verificationErrorString)) {
            fail(verificationErrorString);
          }
        }

        private boolean isElementPresent(By by) {
          try {
            driver.findElement(by);
            return true;
          } catch (NoSuchElementException e) {
            return false;
          }
        }

        private boolean isAlertPresent() {
          try {
            driver.switchTo().alert();
            return true;
          } catch (NoAlertPresentException e) {
            return false;
          }
        }
        private String closeAlertAndGetItsText() {
          try {
            Alert alert = driver.switchTo().alert();
            String alertText = alert.getText();
            if (acceptNextAlert) {
              alert.accept();
            } else {
              alert.dismiss();
            }
            return alertText;
          } finally {
            acceptNextAlert = true;
          }
        }
}
```

Tabella 6.2.1.2 – 2: file SuiteUpdateLocHooks.java adattato per i test Headless

6.2.1.3. Report complessivo – Release v1.0-Hooks

In quest'ultimo sottoparagrafo vediamo invece il report complessivo generato dal file *main.yml*:

TEST SUITE	TEST CASE	RELEASE DI CREAZIONE	LOCATORE
it.catalogo.test.SuiteCreateDeleteLocHooks	testSuiteCreateDeleteLocHooks_loc_Hooks_release_1_0	1_0	test-hooks
it.catalogo.test.SuiteUpdateLocHooks	testSuiteUpdateLocHooks_loc_Hooks_release_1_0	1_0	test-hooks

LOCATORE	ESITO TEST	MESSAGGIO ESITO TEST	CAUSA ROTTURA TEST
test-hooks	Superato	-	-
test-hooks	Superato	-	-

Figura 6.2.1.3 -1: file tabellaReportTest.xls

Da come si può facilmente notare dal report complessivo, entrambi i test sono stati superati (dato che siamo nella prima release non potevano non esserlo), la loro release di creazione è ovviamente la 1_0 ed entrambi risultano utilizzare gli Hooks come localizzatori.

6.2.2. Release v1.0-Tradizionale

Nel seguente paragrafo analizziamo la release 1.0 che non utilizza gli Hooks come localizzatori.

Le uniche differenze con la release v1.0-Hooks saranno quindi che i test JUnit eseguiti non useranno gli Hooks, i loro nomi saranno quindi:

- *testCreateDelete_release_1_0*
- *testUpdate_release_1_0*

Procedendo esattamente come nel paragrafo precedente, quando verranno pushati i test JUnit generati da Katalon Recorder, questi verranno automaticamente adattati in test eseguibili in modalità headless all'interno di un container.

Vediamo l'implementazione delle test suite headless che non utilizzano hooks:

```
//File risulta attualmente aggiornato per webdriver chrome headless!
package it.catalogo.test;

import static org.junit.Assert.fail;
import static org.junit.jupiter.api.Assertions.*;
```

```
import java.util.concurrent.TimeUnit;

import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;
import org.openqa.selenium.Alert;
import org.openqa.selenium.By;
import org.openqa.selenium.NoAlertPresentException;
import org.openqa.selenium.NoSuchElementException;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;

class SuiteCreateDelete {
private static WebDriver driver;
private boolean acceptNextAlert = true;
private static StringBuffer verificationErrors = new StringBuffer();

        @BeforeAll
        public static void setUp() throws Exception {

                // Init chromedriver
                String chromeDriverPath =
"/home/runner/work/WebAppTesi/WebAppTesi/chromedriver_v94_linux64/chromedriver";
                System.setProperty("webdriver.chrome.driver", chromeDriverPath);
                ChromeOptions options = new ChromeOptions();
                options.addArguments("--headless", "--disable-gpu", "--window-size=1920,1200","--
ignore-certificate-errors");
                driver = new ChromeDriver(options);



          driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
        }
 @Test
 public void testSuiteCreateDelete_release_1_0() throws Exception {
   driver.get("http://localhost:8080/prodotti");
   driver.findElement(By.id("nome")).click();
   driver.findElement(By.id("nome")).clear();
   driver.findElement(By.id("nome")).sendKeys("Cellulare");
   driver.findElement(By.id("descrizione")).clear();
   driver.findElement(By.id("descrizione")).sendKeys("Prodotto Samsung");
   driver.findElement(By.id("prezzo")).clear();
   driver.findElement(By.id("prezzo")).sendKeys("400");
   driver.findElement(By.name("invia")).click();
   driver.findElement(By.id("nome")).click();
   driver.findElement(By.id("nome")).clear();
   driver.findElement(By.id("nome")).sendKeys("Lavagna");
   driver.findElement(By.id("descrizione")).clear();
   driver.findElement(By.id("descrizione")).sendKeys("Prodotto Scolastico");
   driver.findElement(By.id("prezzo")).clear();
   driver.findElement(By.id("prezzo")).sendKeys("60");
   driver.findElement(By.name("invia")).click();
```

```
    driver.findElement(By.id("nome")).click();
    driver.findElement(By.id("nome")).clear();
    driver.findElement(By.id("nome")).sendKeys("Dipinto");
    driver.findElement(By.id("descrizione")).clear();
    driver.findElement(By.id("descrizione")).sendKeys("Dimensioni 60cmx90cm");
    driver.findElement(By.id("prezzo")).click();
    driver.findElement(By.id("prezzo")).clear();
    driver.findElement(By.id("prezzo")).sendKeys("25");
    driver.findElement(By.name("invia")).click();
    driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-
space(.)='Modifica'])[12]/preceding::a[1]")).click();
    driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-
space(.)='Modifica'])[11]/preceding::a[1]")).click();
    driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-
space(.)='Modifica'])[10]/preceding::a[1]")).click();
  }


 @AfterAll
        public static void tearDown() throws Exception {
          driver.quit();
          String verificationErrorString = verificationErrors.toString();
          if (!"".equals(verificationErrorString)) {
            fail(verificationErrorString);
          }
        }

        private boolean isElementPresent(By by) {
          try {
            driver.findElement(by);
            return true;
          } catch (NoSuchElementException e) {
            return false;
          }
        }

        private boolean isAlertPresent() {
          try {
            driver.switchTo().alert();
            return true;
          } catch (NoAlertPresentException e) {
            return false;
          }
        }

        private String closeAlertAndGetItsText() {
          try {
            Alert alert = driver.switchTo().alert();
            String alertText = alert.getText();
            if (acceptNextAlert) {
              alert.accept();
            } else {
              alert.dismiss();
            }
```

```
        return alertText;
      } finally {
        acceptNextAlert = true;
      }
    }

}
```

Tabella 6.2.2 – 1: SuiteCreateDelete.java *headless*

```
//File risulta attualmente aggiornato per webdriver chrome headless!
package it.catalogo.test;

import static org.junit.Assert.fail;
import static org.junit.jupiter.api.Assertions.*;

import java.util.concurrent.TimeUnit;

import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;
import org.openqa.selenium.Alert;
import org.openqa.selenium.By;
import org.openqa.selenium.NoAlertPresentException;
import org.openqa.selenium.NoSuchElementException;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;

class SuiteUpdate {
private static WebDriver driver;
private boolean acceptNextAlert = true;
private static StringBuffer verificationErrors = new StringBuffer();

        @BeforeAll
        public static void setUp() throws Exception {

                // Init chromedriver
                String chromeDriverPath =
"/home/runner/work/WebAppTesi/WebAppTesi/chromedriver_v94_linux64/chromedriver";
                System.setProperty("webdriver.chrome.driver", chromeDriverPath);
                ChromeOptions options = new ChromeOptions();
                options.addArguments("--headless", "--disable-gpu", "--window-
size=1920,1200","--ignore-certificate-errors");
                driver = new ChromeDriver(options);



          driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
        }
  @Test
  public void testSuiteUpdate_release_1_0() throws Exception {
    driver.get("http://localhost:8080/prodotti");
```

```
    driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-
space(.)='Elimina'])[2]/preceding::a[1]")).click();
    driver.findElement(By.id("descrizione")).click();
    driver.findElement(By.id("descrizione")).click();
    driver.findElement(By.id("descrizione")).click();
    driver.findElement(By.id("descrizione")).clear();
    driver.findElement(By.id("descrizione")).sendKeys("Marca Nike");
    driver.findElement(By.id("prezzo")).click();
    driver.findElement(By.id("prezzo")).clear();
    driver.findElement(By.id("prezzo")).sendKeys("95");
    driver.findElement(By.name("invia")).click();
    driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-
space(.)='Elimina'])[2]/following::a[1]")).click();
    driver.findElement(By.id("descrizione")).click();
    driver.findElement(By.id("descrizione")).clear();
    driver.findElement(By.id("descrizione")).sendKeys("4 Fornelli");
    driver.findElement(By.id("prezzo")).click();
    driver.findElement(By.id("prezzo")).clear();
    driver.findElement(By.id("prezzo")).sendKeys("380");
    driver.findElement(By.name("invia")).click();
    driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-
space(.)='Elimina'])[3]/following::a[1]")).click();
    driver.findElement(By.id("nome")).click();
    driver.findElement(By.id("nome")).clear();
    driver.findElement(By.id("nome")).sendKeys("TV S150x");
    driver.findElement(By.id("prezzo")).click();
    driver.findElement(By.id("prezzo")).clear();
    driver.findElement(By.id("prezzo")).sendKeys("900");
    driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-
space(.)='Elimina'])[3]/following::a[1]")).click();
    driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-
space(.)='Elimina'])[3]/following::a[1]")).click();
    driver.findElement(By.id("nome")).click();
    driver.findElement(By.id("nome")).clear();
    driver.findElement(By.id("nome")).sendKeys("TV S50x");
    driver.findElement(By.id("prezzo")).click();
    driver.findElement(By.id("prezzo")).clear();
    driver.findElement(By.id("prezzo")).sendKeys("800");
    driver.findElement(By.name("invia")).click();
    driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-
space(.)='Elimina'])[2]/following::a[1]")).click();
    driver.findElement(By.id("descrizione")).click();
    driver.findElement(By.id("descrizione")).clear();
    driver.findElement(By.id("descrizione")).sendKeys("8 Fornelli");
    driver.findElement(By.id("prezzo")).click();
    driver.findElement(By.id("prezzo")).clear();
    driver.findElement(By.id("prezzo")).sendKeys("770");
    driver.findElement(By.name("invia")).click();
    driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-
space(.)='Elimina'])[1]/following::a[1]")).click();
    driver.findElement(By.id("descrizione")).click();
    driver.findElement(By.id("descrizione")).clear();
    driver.findElement(By.id("descrizione")).sendKeys("Marca Adidas");
    driver.findElement(By.id("prezzo")).clear();
```

```
    driver.findElement(By.id("prezzo")).sendKeys("100");
    driver.findElement(By.name("invia")).click();
  }


  @AfterAll
  public static void tearDown() throws Exception {
    driver.quit();
    String verificationErrorString = verificationErrors.toString();
    if (!"".equals(verificationErrorString)) {
      fail(verificationErrorString);
    }
  }

  private boolean isElementPresent(By by) {
    try {
      driver.findElement(by);
      return true;
    } catch (NoSuchElementException e) {
      return false;
    }
  }

  private boolean isAlertPresent() {
    try {
      driver.switchTo().alert();
      return true;
    } catch (NoAlertPresentException e) {
      return false;
    }
  }

  private String closeAlertAndGetItsText() {
    try {
      Alert alert = driver.switchTo().alert();
      String alertText = alert.getText();
      if (acceptNextAlert) {
        alert.accept();
      } else {
        alert.dismiss();
      }
      return alertText;
    } finally {
      acceptNextAlert = true;
    }
  }

}
```

Tabella 6.2.2 – 2: SuiteUpdate.java *headless*

Al rilascio della release anche qui si avvia il file *main.yml* che genera un *report xls* complessivo dei test eseguiti.

Visualizziamo degli screenshot del report ottenuto:

TEST SUITE	TEST CASE	RELEASE DI CREAZIONE	LOCATORE	ESITO TEST	MESSAGGIO ESITO	CAUSA ROTTURA TEST
it.catalogo.test.SuiteCreateDelete	testSuiteCreateDelete_release_1_0	1_0	tradizionale	Superato	-	-
it.catalogo.test.SuiteUpdate	testSuiteUpdate_release_1_0	1_0	tradizionale	Superato	-	-

Figura 6.2.2 – 1: tabellaReportTest.xls

Entrambi i test utilizzano un locatore tradizionale, sono stati creati nella release 1.0 e dato che questo è il primo rilascio risultano ovviamento con esito "*Superato*".

6.3. Release v1.1

Il secondo rilascio delle due versioni della applicativo web è stato realizzato col tag *v1.1-(Tradizionale/Hooks)* dato che dal punto di vista logico non vi è alcuna novità introdotta. Infatti ciò che contraddistingue la versione *1.1* dalla *1.0* è solamente un cambiamento nell'interfaccia grafica, che adesso si presenta così:

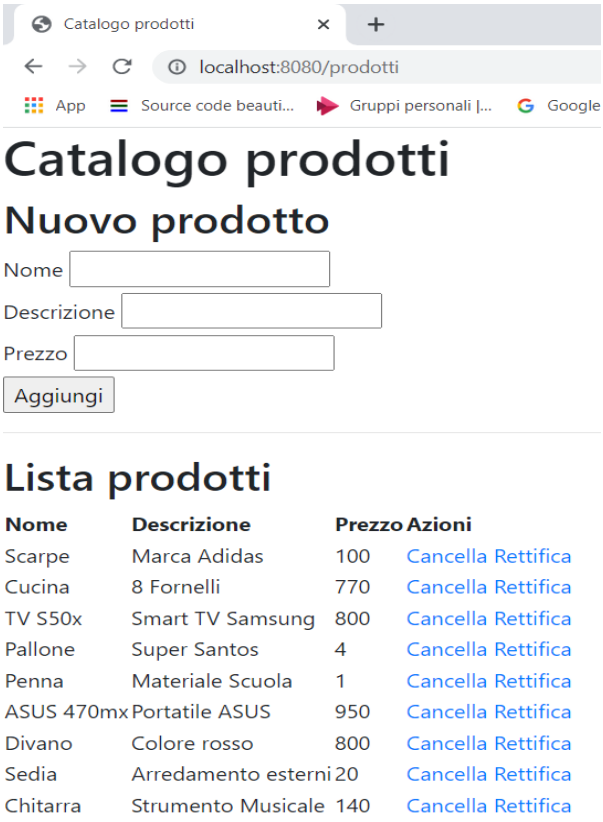


Figura 6.3. – 1: UI applicativo web v1.1

È quindi facile accorgersi che gli unici cambiamenti nella UI riguardano il nome dei Button "*Elimina*" e "*Modifica*", i quali adesso hanno assunto il nome rispettivamente di "*Cancella*" e "*Rettifica*".
Si ricorda che scopo ultimo del seguente lavoro è quello di mostrare la robustezza dei vari tipi di test durante il ciclo di sviluppo del software.

6.3.1. v1.1-Hooks

In questo paragrafo andiamo ad analizzare la versione 1.1 dell'applicativo all'interno della quale sono stati iniettati gli Hooks.
Per non appesantire troppo la trattazione si emette il codice sorgente del file di front-end aggiornato, *indexProdotti.ftl*.

Secondo lo schema operativo precedentemente visto, quando si arriva ad una nuova versione del software, bisogna eseguire i nuovi test appena implementati e rieseguire i test della versione precedente.
A questo punto avremo che i nuovi test saranno sicuramente funzionanti, essendo stati sviluppati sulla versione corrente, mentre alcuni dei test vecchi potranno "rompersi".
I cambiamenti tra una versione e l'altra del software possono infatti far nascere problemi di obsolescenza e/o di fragilità all'interno dei test scritti per le versioni precedenti.
Infatti, se un test vecchio andava a testare una funzionalità che attualmente non è più presente, questo diverrà un test rotto per obsolescenza; invece se un test vecchio testa una funzionalità ancora presente, ma attualmente non funziona più, allora questo si è rotto per fragilità (non riuscendo più ad identificare i componenti della UI con cui interagire).

Per questa nuova versione 1.1 dell'applicativo web, si è implementato uno nuovo test, presente all'interno del file *SuiteCreateAndAssertHooks.java*, il quale dopo essere stato pushato nel repository, come gli altri file di test è stato portato nel formato idoneo per i test headless per opera del progetto *correzioneFormatoTest*.

Vediamo adesso l'implementazione di questo nuovo file di test, anch'esso utilizzante i locatori hooks.

```
//File risulta attualmente aggiornato per webdriver chrome headless!
package it.catalogo.test;

import static org.junit.Assert.fail;
import static org.junit.jupiter.api.Assertions.*;
import java.util.concurrent.TimeUnit;
import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;
import org.openqa.selenium.Alert;
import org.openqa.selenium.By;
import org.openqa.selenium.NoAlertPresentException;
import org.openqa.selenium.NoSuchElementException;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;

class SuiteCreateAndAssertHooks {
private static WebDriver driver;
private boolean acceptNextAlert = true;
private static StringBuffer verificationErrors = new StringBuffer();
		@BeforeAll
		public static void setUp() throws Exception {
			// Init chromedriver
			String chromeDriverPath =
"/home/runner/work/WebAppTesi/WebAppTesi/chromedriver_v94_linux64/chromedriver";
			System.setProperty("webdriver.chrome.driver", chromeDriverPath);
			ChromeOptions options = new ChromeOptions();
			options.addArguments("--headless", "--disable-gpu", "--window-
size=1920,1200","--no-sandbox","--ignore-certificate-errors");
			driver = new ChromeDriver(options);
		  driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
		}
 @Test
 public void testSuiteCreateAndAssertHooks_loc_Hooks_release_1_1() throws Exception {
   driver.get("http://localhost:8080/prodotti");
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
24]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
24]")).clear();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
24]")).sendKeys("Il ritratto di Dorian Gray");
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
27]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
27]")).clear();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
27]")).sendKeys("Libro");
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
```

```
30]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).clear();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).sendKeys("10");
   assertEquals("Il ritratto di Dorian Gray", driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).getAttribute("value"));
            assertEquals("Libro", driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).getAttribute("value"));
            assertEquals("10", driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).getAttribute("value"));
            driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-32]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][10]//*[@x230953393589-x-test-hook-49]")).click();
  }
 @AfterAll
            public static void tearDown() throws Exception {
           //……..parte tagliata per non appesantire troppo la trattazione…….
 }
}
```

Tabella 6.3.1 – 1: Implementazione *SuiteCreateAndAssertHooks.java*

Quindi in toto, al rilascio della versione *v1.1-Hooks* vengono eseguiti i seguenti tre file di test:

- *SuiteCreateDeleteLocHooks.java*
- *SuiteUpdateLocHooks.java*
- *SuiteCreateAndAssertHooks.java*

Vediamo adesso il report complessivo dei test eseguiti, ottenuto al termine dell'esecuzione del file Main.yml.

TEST SUITE	TEST CASE	RELEASE DI CREAZIONE	LOCATORE	ESITO TEST	MESSAGGIO ESITO TEST	CAUSA ROTTURA TEST
it.catalogo.test.SuiteCreateAndAssertHooks	testSuiteCreateAndAssertHooks_loc_Hooks_release_1_1	1_1	test-hooks	Superato	-	-
it.catalogo.test.SuiteCreateDeleteLocHooks	testSuiteCreateDeleteLocHooks_loc_Hooks_release_1_0	1_0	test-hooks	Superato	-	-
it.catalogo.test.SuiteUpdateLocHooks	testSuiteUpdateLocHooks_loc_Hooks_release_1_0	1_0	test-hooks	Superato	-	-

Figura 5.3.1 – 1: Screenshot file *tabellaReportTest.xls*

Come possiamo notare dal report complessivo ottenuto:

- Il test *testSuiteCreateAndAssertHooks_loc_Hooks_release_1_1* è stato creato nella corrente release (1.1) ed ovviamente risulta superato.

- I test *testSuiteCreateDeleteLocHooks_loc_Hooks_release_1_0, testSuiteUpdateHooks_loc_Hooks_release_1_0,* sono stati creati nella precedente release (1.0) e risultano ancora funzionanti.
- Tutti i test case utilizzano come locatori gli *Hooks*.

Dato che nessun test si è rotto e tutti risultano quindi funzionanti la release *v1.1-Hooks* è considerata completa, e può continuare il ciclo di vita del software andando a sviluppare la successiva versione, per esempio la release *v2.0-Hooks*.

6.3.2. v1.1-Tradizionale

La versione v1.1-Tradizionale è ovviamente identica alla v1.1-Hooks, con l'unica differenza che non utilizza i localizzatori hooks.
La nuova test suite introdotta è quindi *SuiteCreateAndAssert.java*:

```
//File risulta attualmente aggiornato per webdriver chrome headless!
package it.catalogo.test;

import static org.junit.Assert.fail;
import static org.junit.jupiter.api.Assertions.*;

import java.util.concurrent.TimeUnit;

.....
import org.openqa.selenium.chrome.ChromeOptions;

class SuiteCreateAndAssert {
private static WebDriver driver;
private boolean acceptNextAlert = true;
private static StringBuffer verificationErrors = new StringBuffer();

	@BeforeAll
	public static void setUp() throws Exception {

		// Init chromedriver
		String chromeDriverPath =
"/home/runner/work/WebAppTesi/WebAppTesi/chromedriver_v94_linux64/chromedriver";
		System.setProperty("webdriver.chrome.driver", chromeDriverPath);
		ChromeOptions options = new ChromeOptions();
		options.addArguments("--headless", "--disable-gpu", "--window-
size=1920,1200","--no-sandbox","--ignore-certificate-errors");
		driver = new ChromeDriver(options);
```

```
                driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
            }
 @Test
 public void testSuiteCreateAndAssert_release_1_1() throws Exception {
  driver.get("http://localhost:8080/prodotti");
  driver.findElement(By.id("nome")).click();
  driver.findElement(By.id("nome")).clear();
  driver.findElement(By.id("nome")).sendKeys("Il ritratto di Dorian Gray");
  driver.findElement(By.id("descrizione")).click();
  driver.findElement(By.id("descrizione")).clear();
  driver.findElement(By.id("descrizione")).sendKeys("Libro");
  driver.findElement(By.id("prezzo")).click();
  driver.findElement(By.id("prezzo")).clear();
  driver.findElement(By.id("prezzo")).sendKeys("10");
  assertEquals("Il ritratto di Dorian Gray",
driver.findElement(By.id("nome")).getAttribute("value"));
            assertEquals("Libro",
driver.findElement(By.id("descrizione")).getAttribute("value"));
            assertEquals("10", driver.findElement(By.id("prezzo")).getAttribute("value"));
            driver.findElement(By.name("invia")).click();
  driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-
space(.)='Rettifica'])[10]/preceding::a[1]")).click();
 }

 ..........................................................................
            }

}
```

Tabella 6.3.2 – 1: Implementazione *SuiteCreateAndAssert.java*

Quindi in toto, al rilascio della versione *v1.1-Tradizionale*, eseguendo vecchi e nuovi test abbiamo i tre seguenti file di test:

- *SuiteCreateDeleteLocHooks.java*
- *SuiteUpdateLocHooks.java*
- *SuiteCreateAndAssertHooks.java*

Vediamo adesso il report complessivo dei test eseguiti, ottenuto al termine dell'esecuzione del file *main.yml*.

TEST SUITE	TEST CASE	RELEASE DI CREAZIONE	LOCATORE
it.catalogo.test.SuiteCreateDelete	testSuiteCreateDelete_release_1_0	1_0	tradizionale
it.catalogo.test.SuiteUpdate	testSuiteUpdate_release_1_0	1_0	tradizionale
it.catalogo.test.SuiteCreateAndAssert	testSuiteCreateAndAssert_release_1_1	1_1	tradizionale

Figura 6.3.2 – 1: Screenshot file *tabellaReportTest.xls - Parte1*

Figura 6.3.2 – 2: Screenshot file *tabellaReportTest.xls – Parte2*

Come possiamo notare dal report complessivo ottenuto:

ESITO TEST	MESSAGGIO ESITO TEST	CAUSA ROTTURA TEST
Presenta un Error	org.openqa.selenium.NoSuchElementException:	Causa di errore obsolescenza/fragilità (specificare a mano)
Presenta un Error	org.openqa.selenium.NoSuchElementException:	Causa di errore obsolescenza/fragilità (specificare a mano)
Superato	-	-

- Il test *testSuiteCreateAndAssert_release_1_1* è stato creato nella corrente release (1.1) ed ovviamente risulta superato.
- I test *testSuiteCreateDelete_release_1_0, testSuiteUpdate_release_1_0,* sono stati creati nella precedente release (1.0) e risultano rotti. Come esito infatti hanno la dicitura "*Presenta un Error"*, dal messaggio di esito test si evince che l'errore è dovuto al fatto che non riescono ad identificare un elemento nel file di front-end.
- Tutti i test case utilizzano locatori *Tradizionali*.

Dato che entrambi i test vecchi risultano rotti, presentando un errore dovuto alla mancata identificazione di un elemento della UI, dobbiamo andare ad analizzarli per capire se l'errore è dovuto ad un problema di obsolescenza o di fragilità.

Se si vanno anche ad analizzare i report singoli delle test suite, è ancora più chiaro che l'errore è stato causato dalla mancata identificazione di un elemento della UI che adesso è cambiato.

Infatti passando dalla versione v1.0-Tradizionale alla v1.1-Tradizionale ciò che è cambiato è stato il nome dei Button "Elimina" e "Modifica", diventando "Cancella" e "Rettifica".
Si può facilmente capire che l'errore era stato casuato da "*fragilità"*, in quanto cambiando il nome dei button i test non riescono più ad identificarli.

Si va quindi ad inserire la stringa "*fragilità"* nella colonna "CAUSA ROTTURA TEST" dei due test in errore.

6.3.2.1. Correzione Test rotti

Prima di procedere nello sviluppo del software è necessario andare a correggere o ad eliminare i test rotti.

Dato che i due test in esame si erano rotti per motivi di fragilità e non di obsolescenza, ciò che bisogna fare è andare a correggerli.

È possibile banalmente correggerli andando a modificare le stringhe “Modifica” ed “Elimina” con “Rettifica” e “Cancella” all’interno del codice dei test rotti.
Aggiornando infatti nel seguente modo il codice dei due test rotti:

```
@Test
public void testSuiteCreateDelete_release_1_1() throws Exception {
  System.out.println("Test iniziato!");

  driver.get("http://localhost:8080/prodotti");
                driver.findElement(By.id("nome")).click();
                driver.findElement(By.id("nome")).clear();
                driver.findElement(By.id("nome")).sendKeys("Cellulare");
                driver.findElement(By.id("descrizione")).clear();
                driver.findElement(By.id("descrizione")).sendKeys("Prodotto Samsung");
                driver.findElement(By.id("prezzo")).clear();
                driver.findElement(By.id("prezzo")).sendKeys("400");
                driver.findElement(By.name("invia")).click();
                driver.findElement(By.id("nome")).click();
                driver.findElement(By.id("nome")).clear();
                driver.findElement(By.id("nome")).sendKeys("Lavagna");
                driver.findElement(By.id("descrizione")).clear();
                driver.findElement(By.id("descrizione")).sendKeys("Prodotto Scolastico");
                driver.findElement(By.id("prezzo")).clear();
                driver.findElement(By.id("prezzo")).sendKeys("60");
                driver.findElement(By.name("invia")).click();
                driver.findElement(By.id("nome")).click();
                driver.findElement(By.id("nome")).clear();
                driver.findElement(By.id("nome")).sendKeys("Dipinto");
                driver.findElement(By.id("descrizione")).clear();
                driver.findElement(By.id("descrizione")).sendKeys("Dimensioni 60cmx90cm");
                driver.findElement(By.id("prezzo")).click();
                driver.findElement(By.id("prezzo")).clear();
                driver.findElement(By.id("prezzo")).sendKeys("25");
                driver.findElement(By.name("invia")).click();
                driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-
space(.)='Rettifica'])[12]/preceding::a[1]")).click();
                driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-
space(.)='Rettifica'])[11]/preceding::a[1]")).click();
                driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-
space(.)='Rettifica'])[10]/preceding::a[1]")).click();
```

```
        System.out.println("Test finito!");
    }
```

Tabella 6.3.2.1 – 1: test-case testSuiteCreateDelete_release_1_1

```
@Test
public void testSuiteUpdate_release_1_1() throws Exception {
driver.get("http://localhost:8080/prodotti");
driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-
space(.)='Cancella'])[2]/preceding::a[1]")).click();
driver.findElement(By.id("descrizione")).click();
driver.findElement(By.id("descrizione")).click();
driver.findElement(By.id("descrizione")).click();
driver.findElement(By.id("descrizione")).clear();
driver.findElement(By.id("descrizione")).sendKeys("Marca Nike");
driver.findElement(By.id("prezzo")).click();
driver.findElement(By.id("prezzo")).clear();
driver.findElement(By.id("prezzo")).sendKeys("95");
driver.findElement(By.name("invia")).click();
driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-
space(.)='Cancella'])[2]/following::a[1]")).click();
driver.findElement(By.id("descrizione")).click();
driver.findElement(By.id("descrizione")).clear();
driver.findElement(By.id("descrizione")).sendKeys("4 Fornelli");
driver.findElement(By.id("prezzo")).click();
driver.findElement(By.id("prezzo")).clear();
driver.findElement(By.id("prezzo")).sendKeys("380");
driver.findElement(By.name("invia")).click();
driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-
space(.)='Cancella'])[3]/following::a[1]")).click();
driver.findElement(By.id("nome")).click();
driver.findElement(By.id("nome")).clear();
driver.findElement(By.id("nome")).sendKeys("TV S150x");
driver.findElement(By.id("prezzo")).click();
driver.findElement(By.id("prezzo")).clear();
driver.findElement(By.id("prezzo")).sendKeys("900");
driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-
space(.)='Cancella'])[3]/following::a[1]")).click();
driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-
space(.)='Cancella'])[3]/following::a[1]")).click();
driver.findElement(By.id("nome")).click();
driver.findElement(By.id("nome")).clear();
driver.findElement(By.id("nome")).sendKeys("TV S50x");
driver.findElement(By.id("prezzo")).click();
driver.findElement(By.id("prezzo")).clear();
driver.findElement(By.id("prezzo")).sendKeys("800");
driver.findElement(By.name("invia")).click();
driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-
space(.)='Cancella'])[2]/following::a[1]")).click();
driver.findElement(By.id("descrizione")).click();
driver.findElement(By.id("descrizione")).clear();
driver.findElement(By.id("descrizione")).sendKeys("8 Fornelli");
```

```
driver.findElement(By.id("prezzo")).click();
driver.findElement(By.id("prezzo")).clear();
driver.findElement(By.id("prezzo")).sendKeys("770");
driver.findElement(By.name("invia")).click();
driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-
space(.)='Cancella'])[1]/following::a[1]")).click();
driver.findElement(By.id("descrizione")).click();
driver.findElement(By.id("descrizione")).clear();
driver.findElement(By.id("descrizione")).sendKeys("Marca Adidas");
driver.findElement(By.id("prezzo")).clear();
driver.findElement(By.id("prezzo")).sendKeys("100");
driver.findElement(By.name("invia")).click();
}
```

Tabella 6.3.2.1 – 2: test-case testSuiteUpdate_release_1_1

Otteniamo quindi all'interno dei file *SuiteCreateDelete.java, SuiteUpdate.java* due nuovi test-case perfettamente funzionanti.
Ovviamente si è dovuto cambiare manualmente la release di creazione dei test-case dato che essendo stati modificati la loro release di creazione passa dalla *1.0* alla *1.1*.

Adesso l'ultima cosa da fare prima di proseguire con lo sviluppo di nuovo codice, è far partire automaticamente l'esecuzione dei test correnti dopo la correzione dei test rotti.
Questo procedimento ha lo scopo di poter verificare automaticamente se effettivamente, prima di procedere con lo sviluppo abbiamo tutti i test funzionanti (per vedere se non abbiamo commesso errori nella correzione dei test rotti).

Quindi con al push dei test corretti si attiva automaticamente lo script "*eseguiTest.yml*" che fa le stesse cose dello script "*main.yml*" senza però andare ad effettuare nessun commit e/o push verso il repository, esegue quindi solamente i test.

Se al termine dell'esecuzione dello script tutti i test funzionano, vuol dire che si può continuare con lo sviluppo di una nuova versione dell'applicativo.
Invece se alcuni test non funzionano, significa che non abbiamo corretto e/o eliminato bene i test rotti, bisogna quindi riandare a modificarli.

Aggiornando i due test rotti, nel modo analizzato a pagina precedente, al termine dello scirpt *eseguiTest.yml* risulta che tutti i test eseguiti sono funzionanti, quindi la correzione effettuata era corretta.

6.3.3. Confronto test-hooks/test-tradizionali in v1.1

Nel passaggio dalla versione 1.0 alla 1.1 abbiamo avuto che:

- Nella versione utilizzante gli Hooks tutti e tre i test hanno eseguito correttamente.
- Nella versione tradizionale ben due test su tre hanno rilevato errori durante la loro esecuzione.

I due test rotti sono risultati in errore a causa della loro fragilità dovuta al fatto che localizzano gli elementi della UI in una maniera poco robusta affidandosi al valore di stringhe che vengono visualizzate nella UI.
I test che utilizzano gli hooks come localizzatori non hanno infatti fallito a valle del cambiamento dei nomi dei *button* da "*Modifica*" ed "*Elimina*" a "*Rettifica*" e "*Cancella*".

Già in questo primo confronto effettuato tra test-case tradizionali e test-case con localizzatori hooks è emersa una maggiore robustezza dei secondi.
Obiettivo dei successivi capitoli sarà quello di rendere ancora più chiara e statisticamente evidente questo maggior livello di robustezza dei localizzatori *Hooks*.

6.4. Release v2.0

Nella realizzazione della versione successiva dell'applicativo, ovvero la 2.0, si è deciso di ampliarne le funzionalità.
Adesso oltre ad essere disponibile il catalogo dei prodotti, vi sarà un apposito url dal quale è possibile visualizzare e gestire la lista dei clienti di un ipotetico store online.

Dal punto di vista implementativo è stata prima implementata la classe *Cliente* nel package *model*:

```
package it.catalogo.model;
import javax.persistence.Column;
```

```
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Cliente {
	@Id
	@GeneratedValue(strategy= GenerationType.IDENTITY)
	private Integer id;
	@Column(name = "nome")
	private String nome;
	@Column(name = "cognome")
	private String cognome;
	@Column(name = "storicospesa")
	private Integer storicospesa;		//numero di soldi mai spesi sul sito
	public Cliente() {
		super();
	}
	public Cliente(Integer id, String nome, String cognome, Integer storicospesa) {
		super();
		this.id = id;
		this.nome = nome;
		this.cognome = cognome;
		this.storicospesa = storicospesa;
	}
	public Integer getId() {
		return id;
	}
	public void setId(Integer id) {
		this.id = id;
	}
	public String getNome() {
		return nome;
	}
	public void setNome(String nome) {
		this.nome = nome;
	}
	public String getCognome() {
		return cognome;
	}
	public void setCognome(String cognome) {
		this.cognome = cognome;
	}
	public Integer getStoricospesa() {
		return storicospesa;
	}
	public void setStoricospesa(Integer storicospesa) {
		this.storicospesa = storicospesa;
	}
}
```

Tabella 6.4 – 1: implementazione file Cliente.java

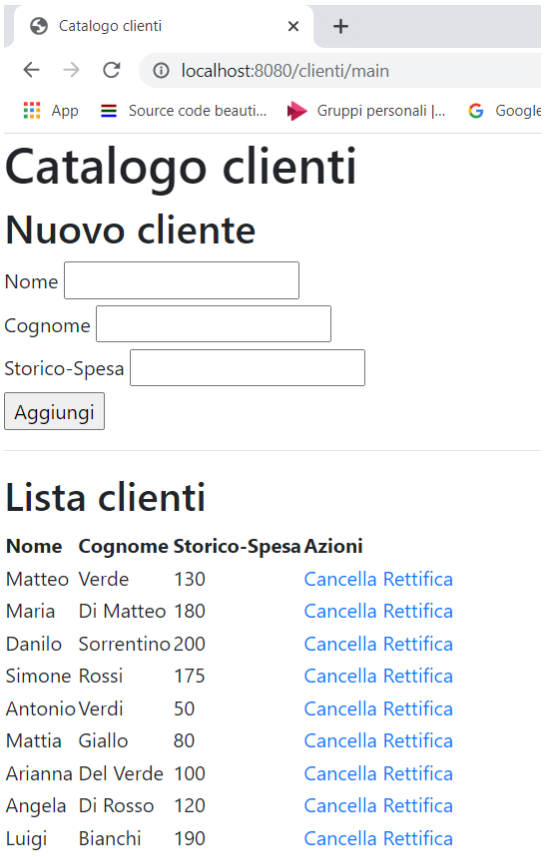
Successivamente è stata definita una interfaccia *ClienteRepository* che va ad estendere *CrudRepository<Cliente,Integer>*, per fare in modo che possano essere utilizzate le API della CRUD Repository nelle operazioni di lettura e scrittura su database per gli oggetti di tipo *Cliente*.

```
package it.catalogo.repository;
import org.springframework.data.repository.CrudRepository;
import it.catalogo.model.Cliente;

public interface ClienteRepository extends CrudRepository<Cliente, Integer>{

}
```

Tabella 6.4 – 2: implementazione file ClienteRepository.java

Successivamente si è andato a definire un nuovo file freemarker per il frontend, *indexClienti.ftl*, definito in maniera strutturalmente simile al precedente file *indexProdotti.ftl*.

Infine è stata creata una nuova classe *ClienteController* all'interno del package *Controller*. Il nuovo controller si occupa quindi di gestire richieste provenienti da specifici url, andando a costruire opportunamente la nuova sezione di gestione e visualizzazione dei clienti appena sviluppata.



Quindi durante l'esecuzione dell'applicativo, andando all'url:

localhost:8080/clienti/main

si giunge nella seguente web che possaimo visualizzare qui a sinistra.
Abbiamo un catalogo dei clienti di un'ipotetico store online, sul quale è possibile fare le fondamentali operazioni di CRUD per la gestione dei clienti.

6.4.1. v2.0-Hooks

Recandosi nella directory *test-hooks/test-guard* del progetto del Dott. Cangianiello e digitando il seguente script da linea di comando:

node main.js inject-hooks …\pathCompleto\...\indexClienti.ftl --grammar freemarker

Si vanno ad iniettare gli hooks all'interno del file freemarker appena sviluppato, relativo al catalogo clienti.

A questo punto mandando in esecuzione l'applicativo in locale, tramite Katalon Recorder è possibile generare dei nuovi casi di test che abbiano come "Release di Creazione" la *2_0*.

Le test suite generate con Katalon Recorder:

- *SuiteCreateClienteHooks*: con test case *testSuiteCreateClienteHooks_loc_Hooks_release_2_0*
- *SuiteUpdateClienteHooks:* con test case *testSuiteUpdateClienteHooks_loc_Hooks_release_2_0*
- *SuiteCreateAndAssertClienteHooks:* con test case *testSuiteCreateAndAssertClienteHooks_loc_Hooks_release_2_0*

Dove manualmente si è dovuto solamente aggiungere ai nomi dei test case, la dicitura _loc_*Hooks* per specificare che stiamo utilizzando gli *Hooks* come locatori, la dicitura *_release_2_0* per specificare che la *2.0* è la release di creazione dei test.

NB: Si è generato un solo test case per test suite poiché nel processo operativo del seguente lavoro, si vanno a salvare i file di test java automaticamente generati da Katalon e tramite lo script *correzioneFormatoTest* li si trasformano automaticamente in test eseguibili in modalità headless all'interno di un Container. Katalon Recorder è infatti in grado di generare codice di test java automaticamente soltanto per test suite composte da un solo test case.

Vediamo qui l'implementazione dei file di test in esame, ottenuta dopo la loro correzione da parte del file *mainOnPush.yml*:

```
//File risulta attualmente aggiornato per webdriver chrome headless!
package it.catalogo.test;

import static org.junit.Assert.fail;
import static org.junit.jupiter.api.Assertions.*;

import java.util.concurrent.TimeUnit;

import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;
import org.openqa.selenium.Alert;
import org.openqa.selenium.By;
import org.openqa.selenium.NoAlertPresentException;
import org.openqa.selenium.NoSuchElementException;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;

class SuiteCreateClienteHooks {
private static WebDriver driver;
private boolean acceptNextAlert = true;
private static StringBuffer verificationErrors = new StringBuffer();

	@BeforeAll
	public static void setUp() throws Exception {

		// Init chromedriver
		String chromeDriverPath = "/home/runner/work/WebAppTesi/WebAppTesi/chromedriver_v94_linux64/chromedriver";
		System.setProperty("webdriver.chrome.driver", chromeDriverPath);
		ChromeOptions options = new ChromeOptions();
		options.addArguments("--headless", "--disable-gpu", "--window-size=1920,1200","--no-sandbox","--ignore-certificate-errors");
		driver = new ChromeDriver(options);

	  driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
	}
 @Test
 public void testSuiteCreateClienteHooks_loc_Hooks_release_2_0() throws Exception {
   driver.get("http://localhost:8080/clienti/main");
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).clear();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).sendKeys("Sergio");
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-
```

```
27]")).clear();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).sendKeys("Viola");
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).clear();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).sendKeys("150");
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-32]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).clear();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).sendKeys("Filippo");
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).clear();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).sendKeys("Verdi");
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).clear();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).sendKeys("50");
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-32]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][11]//*[@x230953393589-x-test-hook-49]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][10]//*[@x230953393589-x-test-hook-49]")).click();
  }
 @AfterAll
                public static void tearDown() throws Exception {
                  driver.quit();
                  String verificationErrorString = verificationErrors.toString();
                  if (!"".equals(verificationErrorString)) {
                    fail(verificationErrorString);
                  }
                }
……parte finale file tagliata per non appesantire troppo la trattazione……
```

Tabella 6.4.1 – 1: file *SuiteCreateClienteHooks.java* corretto.

```
@Test
  public void testSuiteUpdateClienteHooks_loc_Hooks_release_2_0() throws Exception {
    driver.get("http://localhost:8080/clienti/main");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-
```

```
test-hook-44][9]//*[@x230953393589-x-test-hook-50]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-14]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).clear();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).sendKeys("280");
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-18]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][9]//*[@x230953393589-x-test-hook-50]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-14]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).clear();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).sendKeys("170");
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-18]")).click();
  }
```

Tabella 6.4.1 – 2: file *SuiteUpdateClienteHooks.java – parte rilevante*

```
@Test
 public void testSuiteCreateAndAssertClienteHooks_loc_Hooks_release_2_0() throws Exception {
   driver.get("http://localhost:8080/clienti/main");
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).clear();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).sendKeys("Aurelio");
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).clear();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).sendKeys("Rossi");
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).clear();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).sendKeys("500");
   assertEquals("Aurelio", driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).getAttribute("value"));
            assertEquals("Rossi", driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).getAttribute("value"));
            assertEquals("500", driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).getAttribute("value"));
            driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-32]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][10]//*[@x230953393589-x-test-hook-49]")).click();
```

```
}
```

Tabella 6.4.1–3: *SuiteCreateAndAssertClienteHooks.java –parte rilevante*

Dovendone eseguire sia vecchi che nuovi, al rilascio della versione *v2.0-Hooks* vengono eseguiti complessivamente i seguenti 6 file di test:

- *SuiteCreateDeleteLocHooks.java*
- *SuiteUpdateLocHooks.java*
- *SuiteCreateAndAssertHooks.java*
- *SuiteCreateClienteHooks.java*
- *SuiteUpdateClienteHooks.java*
- *SuiteCreateAndAssertClienteHooks.java*

Vediamo adesso il report complessivo dei test eseguiti, ottenuto al termine dell'esecuzione del file *main.yml*.

	TEST SUITE	TEST CASE	RELEASE DI CREAZIONE	LOCATORE	ESITO TEST	MESSAGGIO ESITO TEST	CAUSA ROTTURA TEST
2	it.catalogo.test.SuiteCreateAndAssertHooks	testSuiteCreateAndAssertHooks_loc_Hooks_release_1_1	1_1	test-hooks	Superato	-	-
3	it.catalogo.test.SuiteUpdateClienteHooks	testSuiteUpdateClienteHooks_loc_Hooks_release_2_0	2_0	test-hooks	Superato	-	-
4	it.catalogo.test.SuiteCreateDeleteLocHooks	testSuiteCreateDeleteLocHooks_loc_Hooks_release_1_0	1_0	test-hooks	Superato	-	-
5	it.catalogo.test.SuiteUpdateLocHooks	testSuiteUpdateLocHooks_loc_Hooks_release_1_0	1_0	test-hooks	Superato	-	-
6	it.catalogo.test.SuiteCreateClienteHooks	testSuiteCreateClienteHooks_loc_Hooks_release_2_0	2_0	test-hooks	Superato	-	-
7	it.catalogo.test.SuiteCreateAndAssertClienteHooks	testSuiteCreateAndAssertClienteHooks_loc_Hooks_release_2_0	2_0	test-hooks	Superato	-	-

Figura 6.4.1 – 1: Screenshot file *tabellaReportTest.xls*

Come possiamo notare dal report complessivo ottenuto:

- I test case *testSuiteUpdateClienteHooks_loc_Hooks_release_2_0*, *testSuiteCreateClienteHooks_loc_Hooks_release_2_0*, *testSuiteCreateAndAssertClienteHooks_loc_Hooks_release_2_0* sono stati creati nella corrente release (2.0) ed ovviamente risultano superati.

- I test *testSuiteCreateDeleteLocHooks_loc_Hooks_release_1_0*, *testSuiteUpdateHooks_loc_Hooks_release_1_0* e *testSuiteCreateAndAssertHooks_loc_Hooks_release_1_1* , sono stati creati nella precedenti release e risultano ancora funzionanti.

- Tutti i test case utilizzano come locatori gli *Hooks*.

Dato che nessun test si è rotto e tutti risultano quindi funzionanti la release *v2.0-Hooks* è considerata completa, e può continuare il ciclo di vita del software andando a sviluppare la successiva versione, la release *v2.1-Hooks*.

6.4.2. v2.0-Tradizionale

Le test suite generato con Katalon Recorder sono le stesse della versione v2.0-Hooks, utilizzando però dei locatori tradizionali invece che gli Hooks.

Le test suite create in questa versione sono:

- *SuiteCreateCliente*: con test case *testSuiteCreateCliente_release_2_0*
- *SuiteUpdateCliente:* con test case *testSuiteUpdateClienteHooks_release_2_0*
- *SuiteCreateAndAssertCliente:* con test case *testSuiteCreateAndAssertCliente_release_2_0*

Dove manualmente si è dovuto solamente aggiungere ai nomi dei test case, la dicitura *_release_2_0* per specificare che la 2.0 è la release di creazione dei test.

Vediamo quindi il codice dei file di test ottenuti, dopo che siano stati corretti dallo script *mainOnPush.yml*.

```
//File risulta attualmente aggiornato per webdriver chrome headless!
package it.catalogo.test;

import static org.junit.Assert.fail;
……..
class SuiteCreateCliente {
private static WebDriver driver;
private boolean acceptNextAlert = true;
private static StringBuffer verificationErrors = new StringBuffer();

        @BeforeAll
        public static void setUp() throws Exception {
```

```
                    // Init chromedriver
                    String chromeDriverPath = "/home/runner/work/WebAppTesi/WebAppTesi/chromedriver_v94_linux64/chromedriver";
                    System.setProperty("webdriver.chrome.driver", chromeDriverPath);
                    System.setProperty("webdriver.chrome.whitelistedIps", "");
                    ChromeOptions options = new ChromeOptions();
                    options.addArguments("--headless", "--disable-gpu", "--window-size=1920,1200","--no-sandbox","--ignore-certificate-errors");
                    driver = new ChromeDriver(options);



                driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
              }
 @Test
 public void testSuiteCreateCliente_release_2_0() throws Exception {
   driver.get("http://localhost:8080/clienti/main");
   driver.findElement(By.id("nome")).click();
   driver.findElement(By.id("nome")).clear();
   driver.findElement(By.id("nome")).sendKeys("Sergio");
   driver.findElement(By.id("cognome")).clear();
   driver.findElement(By.id("cognome")).sendKeys("Viola");
   driver.findElement(By.id("storicospesa")).clear();
   driver.findElement(By.id("storicospesa")).sendKeys("150");
   driver.findElement(By.name("invia")).click();
   driver.findElement(By.id("nome")).click();
   driver.findElement(By.id("nome")).clear();
   driver.findElement(By.id("nome")).sendKeys("Filippo");
   driver.findElement(By.id("cognome")).click();
   driver.findElement(By.id("cognome")).clear();
   driver.findElement(By.id("cognome")).sendKeys("Verdi");
   driver.findElement(By.id("storicospesa")).click();
   driver.findElement(By.id("storicospesa")).clear();
   driver.findElement(By.id("storicospesa")).sendKeys("50");
   driver.findElement(By.name("invia")).click();
   driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-space(.)='Rettifica'])[11]/preceding::a[1]")).click();
   driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-space(.)='Rettifica'])[10]/preceding::a[1]")).click();
  }

 @AfterAll
              public static void tearDown() throws Exception {

 (NoSuchElementException e) {
                  return false;
……………………………
             }
…….
}
```

Tabella 6.4.2 – 1: *SuiteCreateCliente.java – parte rilevante*

NB: Dove nel codice si trovano i puntini (……) si intende che è stata omessa una parte di codice, per non appesantire troppo la trattazione.

```
…….
@Test
  public void testSuiteUpdateCliente_release_2_0() throws Exception {
    driver.get("http://localhost:8080/clienti/main");
    driver.findElement(By.xpath("(.//*[normalize-space(text())  and  normalize-
space(.)='Cancella'])[9]/following::a[1]")).click();
    driver.findElement(By.xpath("(.//*[normalize-space(text())  and  normalize-
space(.)='Cognome'])[1]/following::div[1]")).click();
    driver.findElement(By.id("storicospesa")).clear();
    driver.findElement(By.id("storicospesa")).sendKeys("280");
    driver.findElement(By.name("invia")).click();
    driver.findElement(By.xpath("(.//*[normalize-space(text())  and  normalize-
space(.)='Cancella'])[9]/following::a[1]")).click();
    driver.findElement(By.xpath("(.//*[normalize-space(text())  and  normalize-
space(.)='Cognome'])[1]/following::div[1]")).click();
    driver.findElement(By.id("storicospesa")).clear();
    driver.findElement(By.id("storicospesa")).sendKeys("170");
    driver.findElement(By.name("invia")).click();
  }
……..
```

Tabella 6.4.2 – 2: *SuiteUpdateCliente.java – parte rilevante*

```
……..
@Test
  public void testSuiteCreateAndAssertCliente_release_2_0() throws Exception {
    driver.get("http://localhost:8080/clienti/main");
    driver.findElement(By.id("nome")).click();
    driver.findElement(By.id("nome")).clear();
    driver.findElement(By.id("nome")).sendKeys("Aurelio");
    driver.findElement(By.id("cognome")).click();
    driver.findElement(By.id("cognome")).clear();
    driver.findElement(By.id("cognome")).sendKeys("Rossi");
    driver.findElement(By.id("storicospesa")).click();
    driver.findElement(By.id("storicospesa")).clear();
    driver.findElement(By.id("storicospesa")).sendKeys("500");
              assertEquals("Aurelio", driver.findElement(By.id("nome")).getAttribute("value"));
              assertEquals("Rossi", driver.findElement(By.id("cognome")).getAttribute("value"));
              assertEquals("500",
driver.findElement(By.id("storicospesa")).getAttribute("value"));
    driver.findElement(By.name("invia")).click();
    driver.findElement(By.xpath("(.//*[normalize-space(text())  and  normalize-
space(.)='Rettifica'])[10]/preceding::a[1]")).click();
  }
……..
```

Tabella 6.4.2 – 3: *SuiteCreatAndAssertCliente.java – parte rilevante*

Dovendone eseguire sia vecchi che nuovi, al rilascio della versione *v2.0-Tradizionale* vengono eseguiti complessivamente i seguenti 6 file di test:

- *SuiteCreateDelete.java*
- *SuiteUpdate.java*
- *SuiteCreateAndAssert.java*
- *SuiteCreateCliente.java*
- *SuiteUpdateCliente.java*
- *SuiteCreateAndAssertCliente.java*

Al termine dell'esecuzione del file *Main.yml* otteniamo il seguente report dei test complessivo:

	TEST SUITE	TEST CASE	RELEASE DI CREAZIONE	LOCATORE	ESITO TEST	MESSAGGIO ESITO TEST	CAUSA ROTTURA TEST
2	it.catalogo.test.SuiteCreateAndAssertCliente	testSuiteCreateAndAssertCliente_release_2_0	2_0	tradizionale	Superato	-	-
3	it.catalogo.test.SuiteCreateDelete	testSuiteCreateDelete_release_1_1	1_1	tradizionale	Superato	-	-
4	it.catalogo.test.SuiteCreateCliente	testSuiteCreateCliente_release_2_0	2_0	tradizionale	Superato	-	-
5	it.catalogo.test.SuiteUpdate	testSuiteUpdate_release_1_1	1_1	tradizionale	Superato	-	-
6	it.catalogo.test.SuiteCreateAndAssert	testSuiteCreateAndAssert_release_1_1	1_1	tradizionale	Superato	-	-
7	it.catalogo.test.SuiteUpdateCliente	testSuiteUpdateCliente_release_2_0	2_0	tradizionale	Superato	-	-

Figura 6.4.2 – 1: Screenshot file *tabellaReportTest.xls*

Come possiamo notare dal report complessivo ottenuto:

- I test case *testSuiteUpdateCliente_release_2_0, testSuiteCreateCliente_release_2_0, testSuiteCreateAndAssertCliente_release_2_0* sono stati creati nella corrente release (2.0) ed ovviamente risultano superati.

- I test *testSuiteCreateDelete_release_1_1, testSuiteUpdate_release_1_1* e *testSuiteCreateAndAssert_release_1_1* , risultano ancora funzionanti. Questi test erano stati creati in realtà nella release 1.0, ma dato che nella versione 1.1 si sono rotti, si è dovuto correggerli, quindi la loro nuova release di creazione è divenuta la versione 1.1.

- Tutti i test case utilizzano locatori tradizionali.

Dato che nessun test si è rotto e tutti risultano quindi funzionanti la release *v2.0-Tradizionale* è considerata completa, e può continuare il ciclo di vita del software andando a sviluppare la successiva versione, ovvero la release *v2.1-Tradizionale*.

6.5. Release v2.1

Nella versione 2.1 non vi è nessuna aggiunta o modifica all'aspetto puramente funzionale dell'applicativo, ma vengono modificati solamente alcuni dettagli.

Come prima modifica, all'interno del file *ClienteController* viene cambiato il path al quale viene associato il file *indexClienti.ftl*, passando da *localhost:8080/clienti/main* a *localhost:8080/clienti/catalogoclienti*.

Vediamo quindi la nuova implementazione del file *ClienteController.java*:

```
package it.catalogo.controller;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.ModelAndView;
import it.catalogo.repository.*;
import it.catalogo.model.*;

@Controller
@RequestMapping("/clienti")
public class ClienteController {

@Autowired
private ClienteRepository repoC;

//serve localhost:8080/clienti/lista
@ResponseBody
@GetMapping("/lista")
public Iterable<Cliente> list() {
Iterable<Cliente> clienti = repoC.findAll();

//Dovrebbe essere una funzione inline (lambda function)
clienti.forEach((Cliente c) -> {
```

```
System.out.println(c.getNome());
});
return clienti;
}
//serve localhost:8080/clienti/catalogoclienti
@GetMapping("/catalogoclienti")
public ModelAndView indexClienti(@RequestParam(required = false) String id, ModelMap mm) {
if(id != null) {
Cliente cliente = new Cliente();
cliente = repoC.findById(Integer.parseInt(id)).get();
mm.addAttribute("clienteDaModificare",  cliente);
}
Iterable<Cliente> clienti = repoC.findAll();
return new ModelAndView("indexClienti", "listaClienti", clienti);
}
@PostMapping("/add")
public String add(@ModelAttribute("datiCliente") Cliente c) {
repoC.save(c);
return "redirect:/clienti/catalogoclienti";
}
@PostMapping("/update")
public String update(@ModelAttribute("datiCliente") Cliente c) {
repoC.save(c);
return "redirect:/clienti/catalogoclienti";
}
@GetMapping("/delete")
public String delete(@RequestParam("id") String idCliente) {
if(idCliente != null)
repoC.deleteById(Integer.parseInt(idCliente));
return "redirect:/clienti/catalogoclienti";
}
}
```

Tabella 6.5 – 1: implementazione file *ClienteController.java*

Inoltre nel file indexClienti.ftl si va ad aggiornare la riga

```
<a href="main?id=${cliente.id}">Rettifica</a>
```

Con

```
<a href="catalogoclienti?id=${cliente.id}">Rettifica</a>
```

Come seconda modifica invece è stato modificato il file

indexProdotti.ftl, sostituendo le seguenti righe di codice:

```
<div>
<input type="submit" name="invia" value="Salva modifiche" />
</div>
```

```
<div>
<input type="submit" name="invia" value="Aggiungi" />
</div>
```

Con

```
<div>
<input type="submit" name="inviodatiprodotto" value="Salva modifiche" />
</div>
```

```
<div>
<input type="submit" name="inviodatiprodotto" value="Aggiungi" />
</div>
```

Quindi ciò che è stato fatto è stata una semplice rinominazione dell'identificativo di un button della UI, per meglio esplicitare la sua funzionalità.

6.5.1. v2.1-Hooks

Dato che in questo rilascio non sono state aggiunte modifiche alla logica del programma si è sviluppato un solo nuovo caso di test.
Tramite Katalon Recorder si è quindi creato il seguente file:

- SuiteCreateAndUpdateHooks: con test case *testSuiteCreateAndUpdate_loc_Hooks_release_2_1*.

Dove manualmente si è dovuto solamente aggiungere ai nomi del test case, la dicitura _loc_*Hooks* per specificare che si sta utilizzando gli Hooks come locatori, la dicitura *_release_2_1* per specificare che la 2.1 è la release di creazione dei test.

Al push del nuovo file di test, questo viene automaticamente convertito nel formato headless dal file *correzione FormatoTest.yml*, ottenendo:

```
//File risulta attualmente aggiornato per webdriver chrome headless!
package it.catalogo.test;
import static org.junit.Assert.fail;
import static org.junit.jupiter.api.Assertions.*;
import java.util.concurrent.TimeUnit;
import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;
import org.openqa.selenium.Alert;
import org.openqa.selenium.By;
import org.openqa.selenium.NoAlertPresentException;
import org.openqa.selenium.NoSuchElementException;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
```

```
class SuiteCreateAndUpdateHooks_loc_Hooks_release_2_1 {
private static WebDriver driver;
private boolean acceptNextAlert = true;
private static StringBuffer verificationErrors = new StringBuffer();

        @BeforeAll
        public static void setUp() throws Exception {
                // Init chromedriver
                String chromeDriverPath =
"/home/runner/work/WebAppTesi/WebAppTesi/chromedriver_v94_linux64/chromedriver";
                System.setProperty("webdriver.chrome.driver", chromeDriverPath);
                System.setProperty("webdriver.chrome.whitelistedIps", "");
                ChromeOptions options = new ChromeOptions();
                options.addArguments("--headless", "--disable-gpu", "--window-
size=1920,1200","--no-sandbox","--ignore-certificate-errors");
                driver = new ChromeDriver(options);
          driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
        }
  @Test
  public void testSuiteCreateAndUpdateHooks() throws Exception {
    driver.get("http://localhost:8080/prodotti");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-
test-hook-24]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-
test-hook-24]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-
test-hook-24]")).sendKeys("Tavolo");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-
test-hook-27]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-
test-hook-27]")).sendKeys("Arredamento salotto");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-
test-hook-30]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-
test-hook-30]")).sendKeys("400");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-
test-hook-32]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-
test-hook-44][10]//*[@x230953393589-x-test-hook-50]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-
test-hook-14]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-
test-hook-16]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-
test-hook-16]")).sendKeys("550");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-
test-hook-18]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-
test-hook-44][10]//*[@x230953393589-x-test-hook-49]")).click();
  }
 @AfterAll
        public static void tearDown() throws Exception {
          driver.quit();
```

```
                ……
                }
               }
}
```

Tabella 6.5.1 – 1: file di test SuiteCreateAndUpdateHooks_loc_Hooks_release_2_1.java

Dovendo adesso eseguire sia vecchi che nuovi test, al rilascio della versione *v2.1-Hooks* vengono eseguiti complessivamente i seguenti 7 file di test:

- *SuiteCreateDeleteLocHooks.java*
- *SuiteUpdateLocHooks.java*
- *SuiteCreateAndAssertHooks.java*
- *SuiteCreateClienteHooks.java*
- *SuiteUpdateClienteHooks.java*
- *SuiteCreateAndAssertClienteHooks.java*
- *SuiteCreateAndUpdateHooks.java*

Vediamo adesso il report complessivo dei test eseguiti, ottenuto al termine dell'esecuzione del file Main.yml.

	TEST SUITE	TEST CASE	RELEASE DI CREAZIONE	LOCATORE
2	it.catalogo.test.SuiteCreateAndAssertHooks	testSuiteCreateAndAssertHooks_loc_Hooks_release_1_1	1_1	test-hooks
3	it.catalogo.test.SuiteUpdateClienteHooks	testSuiteUpdateClienteHooks_loc_Hooks_release_2_0	2_0	test-hooks
4	it.catalogo.test.SuiteCreateDeleteLocHooks	testSuiteCreateDeleteLocHooks_loc_Hooks_release_1_0	1_0	test-hooks
5	it.catalogo.test.SuiteUpdateLocHooks	testSuiteUpdateLocHooks_loc_Hooks_release_1_0	1_0	test-hooks
6	it.catalogo.test.SuiteCreateAndUpdateHooks	testSuiteCreateAndUpdateHooks_loc_Hooks_release_2_1	2_1	test-hooks
7	it.catalogo.test.SuiteCreateClienteHooks	testSuiteCreateClienteHooks_loc_Hooks_release_2_0	2_0	test-hooks
8	it.catalogo.test.SuiteCreateAndAssertClienteHooks	testSuiteCreateAndAssertClienteHooks_loc_Hooks_release_2_0	2_0	test-hooks

Figura 6.5.1 – 1: Screenshot file *tabellaReportTest.xls* – Parte (1)

ESITO TEST	MESSAGGIO ESITO TEST	CAUSA ROTTURA TEST
Superato	-	-
Presenta un Error	org.openqa.selenium.NoSuchElementException:	Causa di errore obsolescenza/fragilità (specificare a mano)
Superato	-	-
Superato	-	-
Superato	-	-
Presenta un Error	org.openqa.selenium.NoSuchElementException:	Causa di errore obsolescenza/fragilità (specificare a mano)
Presenta un Error	org.openqa.selenium.NoSuchElementException:	Causa di errore obsolescenza/fragilità (specificare a mano)

Figura 6.5.1 – 2: Screenshot file *tabellaReportTest.xls* – Parte (2)

Come possiamo notare dal report complessivo ottenuto:

- Il test case *testSuiteCreateAndUpdateHooks_loc_Hooks_release_2_1*, è stato creato nella corrente release (2.1) ed infatti risulta funzionante.

- I test case *testSuiteUpdateClienteHooks_loc_Hooks_release_2_0, testSuiteCreateClienteHooks_loc_Hooks_release_2_0, testSuiteCreateAndAssertClienteHooks_loc_Hooks_release_2_0* che sono stati creati nella corrente release (2.0) risultano rotti.

- I test *testSuiteCreateDeleteLocHooks_loc_Hooks_release_1_0, testSuiteUpdateHooks_loc_Hooks_release_1_0* e *testSuiteCreateAndAssertHooks_loc_Hooks_release_1_1* , sono stati creati nella precedenti release e risultano ancora funzionanti.

- Tutti i test case utilizzano come locatori gli *Hooks*.

Ciò che rimane da capire è come mai i tre test sviluppati nella release 2.0 si siano rotti presentando un errore.
Andando ad effettuare un analisi più accurata, ci si può rendere conto che nella nuova release l'url http://localhost/clienti/main non corrisponde più a nessuna risorsa, quindi la funzionalità offerta da quell'url non esiste più.
Possiamo quindi affermare che i tre test sono andati in errore per un problema di obsolescenza dato che facevano richieste ad un url che non è più valido, ovvero che è diventato obsoleto.

In questo caso i test non sono quindi falliti per problemi di fragilità attribuibili alla debolezza dei locatori utilizzati.

6.5.1.1. Correzione Test rotti

Prima di procedere nello sviluppo del software è quindi necessario andare a correggere o ad eliminare i test rotti.

I test si erano rotti per motivi di obsolescenza, quindi se si vuole correggerli bisogna sostituire l'url obsoleto (http://localhost/clienti/main) con quello aggiornato (http://localhost/clienti/catalogoclienti).

Effettuando questa sostituzione all'interno dei tre test rotti, questi verranno corretti.

Aggiornando infatti nel seguente modo il codice dei tre test rotti:

```
@Test
  public void testSuiteCreateClienteHooks_loc_Hooks_release_2_1() throws Exception {
    driver.get("http://localhost:8080/clienti/catalogoclienti");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).sendKeys("Sergio");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).sendKeys("Viola");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).sendKeys("150");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-32]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).sendKeys("Filippo");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).sendKeys("Verdi");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).sendKeys("50");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-32]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][11]//*[@x230953393589-x-test-hook-49]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][10]//*[@x230953393589-x-test-hook-49]")).click();
  }
```

Tabella 6.5.1.1 – 1: test-case testSuiteCreateClienteHooks_loc_Hooks_release_2_1

```
@Test
 public void testSuiteCreateAndAssertClienteHooks_loc_Hooks_release_2_1() throws Exception {
   driver.get("http://localhost:8080/clienti/catalogoclienti");
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).clear();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).sendKeys("Aurelio");
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).clear();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).sendKeys("Rossi");
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).clear();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).sendKeys("500");
   assertEquals("Aurelio", driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).getAttribute("value"));
            assertEquals("Rossi", driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).getAttribute("value"));
            assertEquals("500", driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).getAttribute("value"));
            driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-32]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][10]//*[@x230953393589-x-test-hook-49]")).click();
  }
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][10]//*[@x230953393589-x-test-hook-49]")).click();
  }
```

Tabella 6.5.1.1 – 2: test-case testSuiteCreateAndAssertClienteHooks_loc_Hooks_release_2_1

```
@Test
 public void testSuiteUpdateClienteHooks_loc_Hooks_release_2_1() throws Exception {
   driver.get("http://localhost:8080/clienti/catalogoclienti");
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][9]//*[@x230953393589-x-test-hook-50]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-14]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).clear();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).sendKeys("280");
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-18]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-
```

```
test-hook-44][9]//*[@x230953393589-x-test-hook-50]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-14]")).click();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).clear();
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-16]")).sendKeys("170");
   driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-5]//*[@x230953393589-x-test-hook-18]")).click();
  }
```

Tabella 6.5.1.1 – 3: test-case testSuiteUpdateClienteHooks_loc_Hooks_release_2_1

Otteniamo quindi all'interno dei file *SuiteCreateClienteHooks.java, SuiteCreateAndAssertClienteHooks.java,SuiteUpdateClienteHooks.java* tre nuovi test-case perfettamente funzionanti.
Ovviamente si è dovuto cambiare manualmente la release di creazione dei test-case dato che essendo stati modificati la loro release di creazione passa dalla 2.0 alla 2.1.

Adesso l'ultima cosa da fare prima di proseguire con lo sviluppo di nuovo codice, è far partire automaticamente l'esecuzione dei test correnti dopo la correzione dei test rotti, per verificare se effettivamente tutti i test siano funzionanti.

Quindi con al push dei test corretti si attiva automaticamente lo script "*eseguiTest.yml*" che fa le stesse cose dello script "*Main.yml*" senza però andare ad effettuare nessun commit e/o push verso il repository, esegue quindi solamente i test.

Eseguendo lo script *eseguiTest.yml* risulta che tutti i test eseguiti sono funzionanti, quindi la correzione effettuata era corretta.

6.5.2. v2.1-Tradizionale

La test suite generata nella versione v2.1-Tradizionale, identica a quella generata nella versione v2.1-Hooks a meno dei locatori utilizzati, è:

- *SuiteCreateAndUpdate*: con test case *testSuiteCreateAndUpdate_release_2_1*

Dove manualmente si è dovuto solamente aggiungere al nome del test case, la dicitura *_release_2_1* per specificare che la 2.1 è la release di creazione del test.

Vediamo quindi il codice del file di test ottenuto, dopo che sia stato corretto dallo script *mainOnPush.yml*.

```
//File risulta attualmente aggiornato per webdriver chrome headless!
package it.catalogo.test;
import static org.junit.Assert.fail;
import static org.junit.jupiter.api.Assertions.*;
import java.util.concurrent.TimeUnit;
import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;
import org.openqa.selenium.Alert;
import org.openqa.selenium.By;
import org.openqa.selenium.NoAlertPresentException;
import org.openqa.selenium.NoSuchElementException;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
class SuiteCreateAndUpdate {
private static WebDriver driver;
private boolean acceptNextAlert = true;
private static StringBuffer verificationErrors = new StringBuffer();
	@BeforeAll
	public static void setUp() throws Exception {
		// Init chromedriver
		String chromeDriverPath = "/home/runner/work/WebAppTesi/WebAppTesi/chromedriver_v94_linux64/chromedriver";
		System.setProperty("webdriver.chrome.driver", chromeDriverPath);
		System.setProperty("webdriver.chrome.whitelistedIps", "");
		ChromeOptions options = new ChromeOptions();
		options.addArguments("--headless", "--disable-gpu", "--window-size=1920,1200","--no-sandbox","--ignore-certificate-errors");
		driver = new ChromeDriver(options);
	  driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
	}
  @Test
  public void testSuiteCreateAndUpdate_release_2_1() throws Exception {
    driver.get("http://localhost:8080/prodotti");driver.findElement(By.id("nome")).click();
    driver.findElement(By.id("nome")).clear();
driver.findElement(By.id("nome")).sendKeys("Tavolo");
    driver.findElement(By.id("descrizione")).clear();
    driver.findElement(By.id("descrizione")).sendKeys("Arredamento salotto");
    driver.findElement(By.id("prezzo")).clear();
    driver.findElement(By.id("prezzo")).sendKeys("400");
    driver.findElement(By.name("inviodatiprodotto")).click();
    driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-space(.)='Cancella'])[10]/following::a[1]")).click();
    driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-space(.)='Descrizione'])[1]/following::div[1]")).click();
```

```
    driver.findElement(By.id("prezzo")).clear();
    driver.findElement(By.id("prezzo")).sendKeys("550");
    driver.findElement(By.name("inviodatiprodotto")).click();
    driver.findElement(By.xpath("(.//*[normalize-space(text())          and          normalize-
space(.)='Rettifica'])[10]/preceding::a[1]")).click();
  }
 @AfterAll
                public static void tearDown() throws Exception {
………………………………………………………………………………..
                }
}
```

Tabella 6.5.2 – 1: *SuiteCreateAndUpdate.java – parte rilevante*

Dovendone eseguire sia vecchi che nuovi, al rilascio della versione *v2.1-Tradizionale* vengono eseguiti complessivamente i seguenti 7 file di test:

- *SuiteCreateDelete.java*
- *SuiteUpdate.java*
- *SuiteCreateAndAssert.java*
- *SuiteCreateCliente.java*
- *SuiteUpdateCliente.java*
- *SuiteCreateAndAssertCliente.java*
- *SuiteCreateAndUpdate.java*

Al termine dell'esecuzione del file *main.yml* otteniamo il seguente report dei test complessivo:

1	TEST SUITE	TEST CASE	RELEASE DI CREAZIONE	LOCATORE
2	it.catalogo.test.SuiteCreateAndUpdate	testSuiteCreateAndUpdate_release_2_1	2_1	tradizionale
3	it.catalogo.test.SuiteCreateAndAssertCliente	testSuiteCreateAndAssertCliente_release_2_0	2_0	tradizionale
4	it.catalogo.test.SuiteCreateDelete	testSuiteCreateDelete_release_1_1	1_1	tradizionale
5	it.catalogo.test.SuiteCreateCliente	testSuiteCreateCliente_release_2_0	2_0	tradizionale
6	it.catalogo.test.SuiteUpdate	testSuiteUpdate_release_1_1	1_1	tradizionale
7	it.catalogo.test.SuiteCreateAndAssert	testSuiteCreateAndAssert_release_1_1	1_1	tradizionale
8	it.catalogo.test.SuiteUpdateCliente	testSuiteUpdateCliente_release_2_0	2_0	tradizionale

Figura 6.5.2 – 1: Screenshot file *tabellaReportTest.xls* – Parte (1)

ESITO TEST	MESSAGGIO ESITO TEST	CAUSA ROTTURA TEST
Superato	-	-
Presenta un Error	org.openqa.selenium.NoSuchElementException:	Causa di errore obsolescenza/fragilità (specificare a mano)
Presenta un Error	org.openqa.selenium.NoSuchElementException:	Causa di errore obsolescenza/fragilità (specificare a mano)
Presenta un Error	org.openqa.selenium.NoSuchElementException:	Causa di errore obsolescenza/fragilità (specificare a mano)
Presenta un Error	org.openqa.selenium.NoSuchElementException:	Causa di errore obsolescenza/fragilità (specificare a mano)
Presenta un Error	org.openqa.selenium.NoSuchElementException:	Causa di errore obsolescenza/fragilità (specificare a mano)
Presenta un Error	org.openqa.selenium.NoSuchElementException:	Causa di errore obsolescenza/fragilità (specificare a mano)

Figura 6.5.2 – 2: Screenshot file *tabellaReportTest.xls* – Parte (2)

Come possiamo notare dal report complessivo ottenuto:

- I test case *testSuiteUpdateCliente_release_2_0, testSuiteCreateCliente_release_2_0, testSuiteCreateAndAssertCliente_release_2_0* creati nella release (2.0) risultano in errore. Da un'attenta analisi si capisce che anche in questo caso l'errore è dovuto all'obsolescenza, dato che l'url http://localhost:8080/clienti/main non corrisponde più a nessuna risorsa.
  Quindi come nel paragrafo precedente (v2.1-Hooks) è possibile correggere facilmente questi test, sostituendo l'url non funzionante con l'url aggiornato http://localhost:8080/clienti/catalogoclienti all'interno dei file di test.

- I test *testSuiteCreateDelete_release_1_1, testSuiteUpdate_release_1_1* e *testSuiteCreateAndAssert_release_1_1,* risultano anch'essi in errori.
  Se si va ad analizzare i file di report notiamo però che stavolta l'errore è dovuto alla fragilità dei test.
  Il motivo del fallimento dei test è il cambiamento dell'identificativo associato al button di invio dati della parte "catalogo prodotti" dell'applicativo web.
  Infatti i test in esame identificavano il button dal nome "invia", che passando dalla release 2.0 alla 2.1 si è tramutato in "inviodatiprodotto", non essendo più riconoscibile da questi test tradizionali che non utilizzano appositi locatori.

  In questo caso quindi per correggere questi 3 test rotti è sufficiente sostituire, ove compare, la stringa "*invia*" con la stringa "*inviodatiprodotto*". Così facendo si correggeranno anche questi altri test rotti per motivi di fragilità.

- Il test case *testSuiteCreateAndUpdate_release_2_1* risulta ovviamente funzionante, essendo stato sviluppato nella corrente release.

- Tutti i test case utilizzano localizzatori tradizionali

Dopo aver corretto opportunamente i 6 test rotti (di cui 3 rotti per motivi di obsolescenza e 3 per motivi di fragilità), al push della versione corretta dei test partirà lo script *eseguiTest.yml* che andrà ad eseguire nuovamente tutti i test col solo scopo di poterne verificare l'effettivo funzionamento.

Al termine dello script tutti i test sono risultati funzionanti, la release *v2.1-Tradizionale* è considerata completa, e può continuare lo sviluppo del software con la release v3.0-Tradizionale.

NB: Bisogna ricordarsi di cambiare le release di creazione di tutti i test case correnti con il valore della corrente release.

6.5.3. Confronto test delle versioni 2.1

Riassumendo nella versione *2.1-Hooks* sono stati eseguiti 7 casi di test, di cui:

4 casi di test funzionanti
3 casi di test rotti → in errore → causa obsolescenza

Mentre nella versione *2.1-Tradizionale* sono stati eseguiti sempre 7 casi di test, ma con:

1 caso di test funzionante
3 casi di test rotti → in errore → causa obsolescenza
3 casi di test rotti → in errore → causa fragilità

Infatti nel secondo caso oltre a rompersi i tre casi di test a causa dell'obsolescenza del url: http://localhost:8080/clienti/main

si sono rotti anche altri tre casi di test a causa del cambiamento del nome di un *button* da "*invia*" ad "*inviodatiprodotto*".

Questo cambiamento di nome si è rivelato un motivo di rottura solamente per i test tradizionali, mentre i test con localizzatori hooks sono stati lo stesso in grado di identificare il button a valle del cambiamento di identificativo.

Quindi anche dall'analisi concorrente effettuata fra le versioni 2.1 con Hooks e Tradizionale viene fuori l'effettiva maggiore robustezza dei test che fanno uso dei localizzatori hooks iniettati nel codice front-end dell'applicativo web.

6.6. Release v2.2

La versione 2.2 è l'ultima versione analizzata all'interno del seguente lavoro, ritenendo che un confronto tra i casi di test effettuato attraverso 5 versioni di un software in via di sviluppo sia sufficiente per stabilire empiricamente la maggior robustezza dei localizzatori basati su Hooks.

Nell'ultima versione realizzata si ipotizza che vi siano verificati dei cambiamenti all'interno dell'interfaccia grafica della parte *catalogo clienti* dell'applicativo web.
Si introduce quindi un cambiamento grafico, con i button "*Rettifica*" e "*Cancella*" che cambiano nomi in "*Aggiorna dati cliente*" e "*Rimuovi dati cliente*".

Ottenendo quindi la seguente interfaccia grafica:

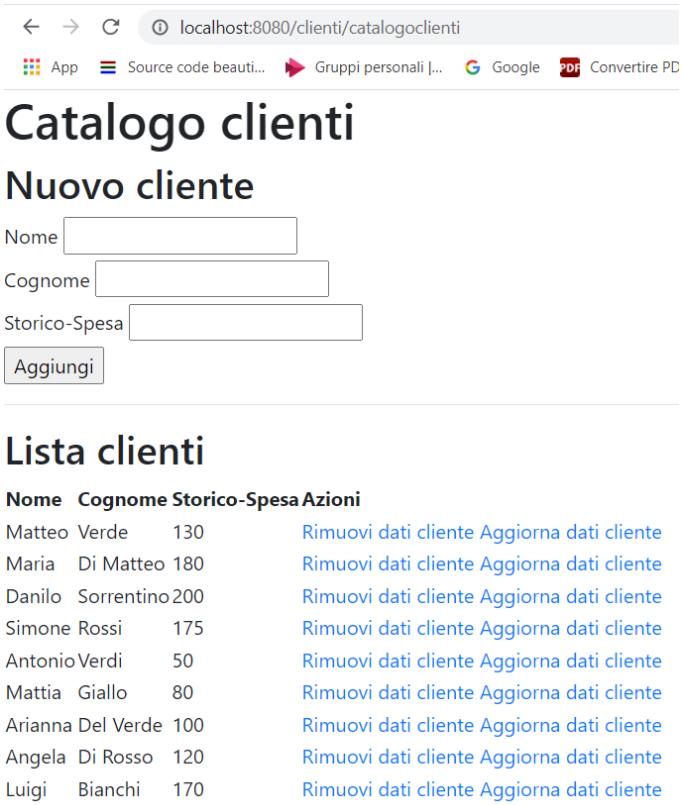


Figura 6.6 – 1: Screenshot UI catalogo clienti

6.6.1. v2.2-Hooks

Nell'ultima versione si è aggiunto un ultimo caso di test, presente all'interno del file *SuiteTwoLinkHooks.java*, il quale dopo essere stato pushato all'interno del repository è stato portato in un formato idoneo all'esecuzione headless dal file *mainOnPush.yml*.

Il codice del test case appena creato è il seguente:

```
@Test
  public void testSuiteTwoLinkHooks_loc_Hooks_release_2_2() throws Exception {
    driver.get("http://localhost:8080/clienti/catalogoclienti");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).sendKeys("Mario");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).sendKeys("Rossi");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).sendKeys("200");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-32]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-test-hook-44][10]//*[@x230953393589-x-test-hook-49]")).click();
driver.get("http://localhost:8080/prodotti");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-24]")).sendKeys("Batteria");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-27]")).sendKeys("Strumento Musicale");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).clear();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-30]")).sendKeys("400");
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-20]//*[@x230953393589-x-test-hook-32]")).click();
    driver.findElement(By.xpath("//*[@x230953393589-x-test-tpl-35]//*[@x230953393589-x-
```

```
test-hook-44][10]//*[@x230953393589-x-test-hook-49]")).click();
  }
```

Tabella 6.6.1 – 1: test case appena creato

Dove al nome del test case *testSuiteTwoLinkHooks* generato da Katalon Recorder, come di consueto, è stata aggiunta la dicitura *loc_Hooks* e *_release_2_2* per specificare che il test utilizza gli *hooks* come locatori e che è stato creato nella release *2.2*.

Dovendo adesso eseguire sia vecchi che nuovi test, al rilascio della versione *v2.2-Hooks* vengono eseguiti complessivamente i seguenti 8 file di test:

- *SuiteCreateDeleteLocHooks.java*
- *SuiteUpdateLocHooks.java*
- *SuiteCreateAndAssertHooks.java*
- *SuiteCreateClienteHooks.java*
- *SuiteUpdateClienteHooks.java*
- *SuiteCreateAndAssertClienteHooks.java*
- *SuiteCreateAndUpdateHooks.java*
- *SuiteTwoLinkHooks.java*

Eseguendo quindi il file Main.yml al momento del rilascio della corrente release, si ottiene il seguente file di report complessivo:

	TEST SUITE	TEST CASE	RELEASE DI CREAZIONE	LOCATOR	ESITO TE	MESSAGGIO ESITO TEST	CAUSA ROTTURA TEST
2	it.catalogo.test.SuiteCreateAndAssertHooks	testSuiteCreateAndAssertHooks_loc_Hooks_release_1_1	1_1	test-hooks	Superato	-	-
3	it.catalogo.test.SuiteUpdateClienteHooks	testSuiteUpdateClienteHooks_loc_Hooks_release_2_1	2_1	test-hooks	Superato	-	-
4	it.catalogo.test.SuiteCreateDeleteLocHooks	testSuiteCreateDeleteLocHooks_loc_Hooks_release_1_0	1_0	test-hooks	Superato	-	-
5	it.catalogo.test.SuiteUpdateLocHooks	testSuiteUpdateLocHooks_loc_Hooks_release_1_0	1_0	test-hooks	Superato	-	-
6	it.catalogo.test.SuiteCreateAndUpdateHooks	testSuiteCreateAndUpdateHooks_loc_Hooks_release_2_1	2_1	test-hooks	Superato	-	-
7	it.catalogo.test.SuiteCreateClienteHooks	testSuiteCreateClienteHooks_loc_Hooks_release_2_1	2_1	test-hooks	Superato	-	-
8	it.catalogo.test.SuiteCreateAndAssertClienteHooks	testSuiteCreateAndAssertClienteHooks_loc_Hooks_release_2_1	2_1	test-hooks	Superato	-	-
9	it.catalogo.test.SuiteTwoLinkHooks	testSuiteTwoLinkHooks_loc_Hooks_release_2_2	2_2	test-hooks	Superato	-	-

Figura 6.6.1 – 1: Screenshot file *tabellaReportTest.xls*

Come possiamo notare dal report complessivo ottenuto:

- Tutti i test case utilizzano come locatori gli *Hooks*.

- Il test case *testSuiteTwoLinkHooks_loc_Hooks_release_2_2*, è stato creato nella corrente release (2.2) ed infatti risulta funzionante.

- Tutti i vecchi test risultano ancora perfettamente funzionanti.

  Dato che tutti e 8 i test risultano perfettamente funzionanti quest'ultima release con localizzatori Hooks può considerarsi conclusa.

6.6.2. v2.2-Tradizionale

Anche in questo caso la test suite generata nella versione v2.2-Tradizionale è ovviamente identica a quella generata nella versione v2.2-Hooks a meno dei locatori utilizzati.
La test suite creata è quindi:

- *SuiteTwoLink*: con test case *testSuiteTwoLink_release_2_2*

Dove manualmente si è dovuto solamente aggiungere al nomw del test case, la dicitura *_release_2_2* per specificare che la 2.2 è la release di creazione del test.

Vediamo quindi il codice del file di test ottenuto, dopo che sia stato corretto dallo script *mainOnPush.yml*.

```
//File risulta attualmente aggiornato per webdriver chrome headless!
package it.catalogo.test;
import static org.junit.Assert.fail;
import static org.junit.jupiter.api.Assertions.*;
import java.util.concurrent.TimeUnit;
import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;
import org.openqa.selenium.Alert;
import org.openqa.selenium.By;
import org.openqa.selenium.NoAlertPresentException;
import org.openqa.selenium.NoSuchElementException;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;

class SuiteCreateAndUpdate {
private static WebDriver driver;
private boolean acceptNextAlert = true;
private static StringBuffer verificationErrors = new StringBuffer();
```

```
        @BeforeAll
        public static void setUp() throws Exception {
                // Init chromedriver
                String chromeDriverPath =
"/home/runner/work/WebAppTesi/WebAppTesi/chromedriver_v94_linux64/chromedriver";
                System.setProperty("webdriver.chrome.driver", chromeDriverPath);
                System.setProperty("webdriver.chrome.whitelistedIps", "");
                ChromeOptions options = new ChromeOptions();
                options.addArguments("--headless", "--disable-gpu", "--window-
size=1920,1200","--no-sandbox","--ignore-certificate-errors");
                driver = new ChromeDriver(options);
          driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
        }
@Test
  public void testSuiteTwoLink_release_2_2() throws Exception {
    driver.get("http://localhost:8080/clienti/catalogoclienti");
    driver.findElement(By.id("nome")).click();
    driver.findElement(By.id("nome")).clear();
    driver.findElement(By.id("nome")).sendKeys("Mario");
    driver.findElement(By.id("cognome")).clear();
    driver.findElement(By.id("cognome")).sendKeys("Rossi");
    driver.findElement(By.id("storicospesa")).clear();
    driver.findElement(By.id("storicospesa")).sendKeys("200");
    driver.findElement(By.name("invia")).click();
    driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-
space(.)='Aggiorna dati cliente'])[10]/preceding::a[1]")).click();
    driver.get("http://localhost:8080/prodotti");
    driver.findElement(By.id("nome")).click();
    driver.findElement(By.id("nome")).clear();
    driver.findElement(By.id("nome")).sendKeys("Batteria");
    driver.findElement(By.id("descrizione")).click();
    driver.findElement(By.id("descrizione")).clear();
    driver.findElement(By.id("descrizione")).sendKeys("Strumento Musicale");
    driver.findElement(By.id("prezzo")).click();
    driver.findElement(By.id("prezzo")).clear();
    driver.findElement(By.id("prezzo")).sendKeys("400");
    driver.findElement(By.name("inviodatiprodotto")).click();
    driver.findElement(By.xpath("(.//*[normalize-space(text()) and normalize-
space(.)='Rettifica'])[10]/preceding::a[1]")).click();
  }

 @AfterAll
        public static void tearDown() throws Exception {
……..
        }
}
```

Tabella 6.6.2 – 1: *SuiteTwoLink.java – parte rilevante*

Adesso dovendo eseguire sia vecchi che nuovi test, al rilascio della versione *v2.2-Tradizionale* vengono eseguiti complessivamente i seguenti 8 file di test:

- *SuiteCreateDelete.java*
- *SuiteUpdate.java*

- *SuiteCreateAndAssert.java*
- *SuiteCreateCliente.java*
- *SuiteUpdateCliente.java*
- *SuiteCreateAndAssertCliente.java*
- *SuiteCreateAndUpdate.java*
- *SuiteTwoLink.java*

Al momento del rilascio della corrente release viene quindi eseguito il file Main.yml, il quale genera in output il seguente report complessivo dei test:

	TEST SUITE	TEST CASE	RELEASE DI CREAZIONE	LOCATORE
2	it.catalogo.test.SuiteCreateAndUpdate	testSuiteCreateAndUpdate_release_2_1	2_1	tradizionale
3	it.catalogo.test.SuiteCreateAndAssertCliente	testSuiteCreateAndAssertCliente_release_2_1	2_1	tradizionale
4	it.catalogo.test.SuiteCreateDelete	testSuiteCreateDelete_release_2_1	2_1	tradizionale
5	it.catalogo.test.SuiteCreateCliente	testSuiteCreateCliente_release_2_1	2_1	tradizionale
6	it.catalogo.test.SuiteUpdate	testSuiteUpdate_release_2_1	2_1	tradizionale
7	it.catalogo.test.SuiteCreateAndAssert	testSuiteCreateAndAssert_release_2_1	2_1	tradizionale
8	it.catalogo.test.SuiteUpdateCliente	testSuiteUpdateCliente_release_2_1	2_1	tradizionale
9	it.catalogo.test.SuiteTwoLink	testSuiteTwoLink_release_2_2	2_2	tradizionale

Figura 6.6.2 – 1: Screenshot file *tabellaReportTest.xls* – Parte (1)

ESITO TEST	MESSAGGIO ESITO TEST	CAUSA ROTTURA TEST
Superato	-	-
Presenta un Error	org.openqa.selenium.NoSuchElementException:	Causa di errore obsolescenza/fragilità (specificare a mano)
Superato	-	-
Presenta un Error	org.openqa.selenium.NoSuchElementException:	Causa di errore obsolescenza/fragilità (specificare a mano)
Superato	-	-
Superato	-	-
Presenta un Error	org.openqa.selenium.NoSuchElementException:	Causa di errore obsolescenza/fragilità (specificare a mano)
Superato	-	-

Figura 6.6.2 – 2: Screenshot file *tabellaReportTest.xls* – Parte (2)

Come possiamo notare dal report complessivo ottenuto:

- I test case *testSuiteUpdateCliente_release_2_1*, *testSuiteCreateCliente_release_2_1*, *testSuiteCreateAndAssertCliente_release_2_1* creati nella release (2.1) risultano in errore. Da un'attenta analisi si capisce che ciò è causato da un problema di fragilità dei locatori tradizionali, infatti i test falliscono quando cercano di riconoscere uno dei button

"Rimuovi dati cliente", "Aggiorna dati cliente". Dato che nella precedente release questi due button erano identificati con nomi diversi, "Cancella" e "Rettifica", adesso i casi di test tradizionali non sono più in grado di riconoscerli, rompendosi.

Una soluzione per correggere questi test può essere quella di cambiare i codici dei test case in modo che dove compaia l'identificativo "Cancella" questo venga sostituito da "Rimuovi dati cliente" e dove compare "Rettifica" da "Aggiorna dati cliente".
Così facendo si potrà quindi correggere il codice dei tre casi di test, ovviamente bisogna esplicitare nel loro nome la nuova release di creazione (2.2).
NB: Si omette la trascrizione dei codici sorgente corretti per non appesantire troppo la trattazione.

- I test *testSuiteCreateDelete_release_2_1*, *testSuiteUpdate_release_2_1* e *testSuiteCreateAndAssert_release_2_1*, *testSuiteCreateAndUpdate_release_2_1*, risultano invece superati. Infatti questi vanno a testare il file *indexProdotti.ftl* che non è minimamente cambiato dalla release 2.1 alla 2.2.

- Il test case *testSuiteTwoLink_release_2_2* risulta ovviamente funzionante, essendo stato sviluppato nella corrente release.

- Tutti i test case utilizzano localizzatori tradizionali.

6.6.3. Confronto test delle versioni 2.2

Riassumendo nella versione *2.2-Hooks* sono stati eseguiti 8 casi di test, di cui:

8 casi di test funzionanti

Mentre nella versione *2.2-Tradizionale* sono stati eseguiti sempre 8 casi di test, ma con:

5 casi di test funzionante
3 casi di test rotti → in errore → causa fragilità

Infatti nel secondo caso ci sono tre casi di test che si rompono a causa della loro fragilità, non riconoscendo più dei button a valle di un loro semplice cambiamento di nome.
Questo cambiamento si è rivelato un motivo di rottura solamente per i test tradizionale, dato che i test con localizzatori hooks sono stati lo stesso in grado di identificare i button.

Anche qui dalle analisi concorrenti effettuate fra le versioni 2.2 con Hooks e Tradizionale viene fuori l'effettiva maggiore robustezza dei test che fanno uso dei localizzatori hooks iniettati nel codice front-end dell'applicativo web.

6.7. Risultati sperimentali ottenuti

In questo ultimo paragrafo si vanno a riportare i risultati empirici complessivi ottenuti lungo il corso dell'intero capitolo.

Si riporta nella seguente tabella l'esito dei test eseguiti lungo il susseguirsi delle versioni rilasciate:

RELEASE	vers. HOOKS	vers. TRADIZIONALE
v.1.0	Test eseguiti: 2 Test funzionanti: 2 Test rotti: 0 Test fragili: 0 Test obsoleti: 0	Test eseguiti: 2 Test funzionanti: 2 Test rotti: 0 Test fragili: 0 Test obsoleti: 0
v1.1	Test eseguiti: 3 Test funzionanti: 3 Test rotti: 0 Test fragili: 0 Test obsoleti: 0	Test eseguiti: 3 Test funzionanti: 1 Test rotti: 2 Test fragili: 2 Test obsoleti: 0
v2.0	Test eseguiti: 6 Test funzionanti: 6 Test rotti: 0 Test fragili: 0 Test obsoleti: 0	Test eseguiti: 6 Test funzionanti: 6 Test rotti: 0 Test fragili: 0 Test obsoleti: 0
v2.1	Test eseguiti: 7 Test funzionanti: 4 Test rotti: 3 Test fragili: 0 Test obsoleti: 3	Test eseguiti: 7 Test funzionanti: 1 Test rotti: 6 Test fragili: 3 Test obsoleti: 3
v2.2	Test eseguiti: 8 Test funzionanti: 8 Test rotti: 0 Test fragili: 0 Test obsoleti: 0	Test eseguiti: 8 Test funzionanti: 5 Test rotti: 3 Test fragili: 3 Test obsoleti: 0

Volendo dare dei valori complessivi, risulta:

	Test-Hooks	Test-Tradizionali
Totale test Eseguiti	2+3+6+7+8= 26	2+3+6+7+8= 26
Totale test Funzionanti	2+3+6+4+8= 23	2+1+6+1+5= 15
Totale test Rotti	0+0+0+3+0= 3	0+2+0+6+3= 11
Totale test Fragili	0+0+0+0+0= 0	0+2+0+3+3= 8
Totale test Obsoleti	0+0+0+3+0= 3	0+0+0+3+0= 3

Quindi dai risultati empirici ottenuti si ricava che:

	Test-Hooks	Test-Tradizionali
Percentuale test funzionanti	(23/26) * 100 = 88,461 %	(15/26) * 100 = 57,692 %
Percentuale test rotti	(3/26) * 100 = 11,539 %	(11/26) * 100 = 42,308 %
Percentuale test fragili	(0/26) * 100 = 0,000 %	(8/26) * 100 = 30,769 %
Percentuale test obsoleti	(3/26) * 100 = 11,539 %	(3/26) * 100 = 11,539 %

Guardando le statistiche complessive, si può notare facilmente che l'utilizzo dei locatori ha portato, in questo caso, ad azzerare completamente la percentuale dei test fragili, portandola dal 30,769% al 0,000%.

I risultati empirici qui presentati, ovviamente sono specifici del caso sperimentale analizzato, tuttavia rendono chiaro un concetto generale, ovvero che nella maggior parte dei casi pratici i test che utilizzano gli *hooks* come locatori sono molto più **robusti** di quelli tradizionali.

Il motivo è dovuto al fatto che questi si basano su dei *ganci* iniettati appositamente all'interno del codice sorgente dei file di front-end, in modo da poter identificare i componenti dell'interfaccia grafica anche a valle di un loro cambiamento di nome e/o posizione.

Ovviamente l'utilizzo dei ganci non può risolvere tutti i tipi di errori che possono riscontrare i casi di test; infatti, possiamo notare che in entrambi i casi la percentuale dei test obsoleti è rimasta costante al 11,539%. Ciò è dovuto al fatto che i test obsoleti vanno a testare funzionalità che non sono più previste nella versione attuale; quindi, il loro crash non è dovuto ad una mancata identificazione dei componenti, in quanto questi non esistono più.

Capitolo 7: Visualizzazione complessiva report finali

Per ottenere una più chiara visualizzazione dei report complessivi ottenuti attraverso tutto il ciclo di sviluppo dell'*Application Under Testing* (AUT), si è deciso di sviluppare un ulteriore strumento presente all'interno del seguente repository github:
https://github.com/gianlucat97/Tesi-ReportFinali-StrumentoGenerale.git

All'interno della cartella "Report-Separati" bisogna memorizzare i report complessivi (in formato *xls*) ottenuti ad ogni release dell'AUT.

Nel repository si trova poi il file unisciReportExcel.zip contenente un *jar* in grado di unire i report di tutto lo storico delle versioni del software rilasciate.

Vi si trova poi il file YAML *generaReportFinale.yml* all'interno della sottodirectory ./.github/workflows, il quale ad ogni push di nuovo codice va a generare il report complessivo contenente l'esito dei test eseguiti ad ogni release del software che è stata rilasciata.
Vediamone l'implementazione:

```
# This is a basic workflow to help you get started with Actions

name: On Push - Generazione report complessivo (xls) di tutte le versioni.

# Controls when the workflow will run
on:
  # Triggers the workflow on push or pull request events but only for the master branch
  push:
    branches: [ master ]


  # Allows you to run this workflow manually from the Actions tab
  workflow_dispatch:

# A workflow run is made up of one or more jobs that can run sequentially or in parallel
jobs:
  # This workflow contains a single job called "build"
  build:
    # The type of runner that the job will run on
    runs-on: ubuntu-latest
    environment:
      name: envForGithubActions
```

```
  # Steps represent a sequence of tasks that will be executed as part of the job
   steps:
     # Checks-out your repository under $GITHUB_WORKSPACE, so your job can access it
     - name: Step 1 - Checkout main branch from GitHub
       uses: actions/checkout@v2

     # Runs a single command using the runners shell
     - name: Step 2 - Set up JDK 1.8
       uses: actions/setup-java@v1
       with:
         java-version: 1.8


     - name: Step 3 - Unzip ed esecuzione progetto unisciReportExcel
       run: |
         echo "Proviamo ad unzippare"
         unzip unisciReportExcel
         echo "Unzippato"
         ls -a
         cd unisciReportExcel/target
         echo "Vediamo contenuto cartella target"
         ls -a
         sudo bash -c 'java -jar unisciReportExcel-0.0.1-jarReportTest.jar
/home/runner/work/Tesi-ReportFinali-StrumentoGenerale/Tesi-ReportFinali-
StrumentoGenerale/Report-Separati reportComplessivo'

     - name: Step 4 - Sposta file xls generati
       run: |
         ls -a
         cd unisciReportExcel/target
         echo "Vediamo contenuto cartella target"
         ls -a
         sudo cp reportComplessivo.xls /home/runner/work/Tesi-ReportFinali-
StrumentoGenerale/Tesi-ReportFinali-StrumentoGenerale

     - name: Step 5 - Push dei file dei report ottenuti
       run: |
         echo "Vediamo quali cartelle ci sono nella directory di partenza"
         ls -a
         git status
         git config --global user.email "${{ secrets.EMAIL_ACCOUNT_GITHUB }}"
         git config --global user.name "${{ secrets.NOME_ACCOUNT_GITHUB }}"
         git config --global user.password "${{ secrets.PASSWORD_ACCOUNT_GITHUB }}"
         echo "Andiamo ad aggiungere i file xls"
         git add *.xls
         echo "File xls aggiunti!"
         echo "`date +%Y-%m-%d_%H-%M-%S`" > timeCommit.txt
         git add timeCommit.txt
         git commit -m "Aggiunta dei report complessivi finali"
         git branch -M master
         git push -u origin master
```

Capitolo 7 – Tabella 1: file *generaReportFinale.yml*

Quindi il file di report finale viene generato a partire dai file xls caricati all'interno della directory "Report-Separati".

Anche per questo progetto è stato utilizzato un virtual environment "*envForGithubActions*".

Vediamo qui la lista delle variabili d'ambiente presenti all'interno dell'environment definito:

```
EMAIL_ACCOUNT_GITHUB
NOME_ACCOUNT_GITHUB
PASSWORD_ACCOUNT_GITHUB
```

In questo progetto le variabili di ambiente devono quindi essere customizzate per permettere il push del report complessivo all'interno del repository di proprietà dell'utente che ne ha fatto la fork.

N.B. Le seguenti variabili d'ambiente sono ovviamente diverse da quelle descritte precedenti capitoli, dato che queste si riferiscono allo strumento:
https://github.com/gianlucat97/Tesi-ReportFinali-StrumentoGenerale.git
mentre le precedenti si riferivano al lavoro principale:
https://github.com/gianlucat97/Tesi-StrumentoGenerale.git

7.1. Report finali caso di studio

In questo paragrafo andiamo quindi a customizzare la repository precedente, facendone una fork e creando una nuova repository:

https://github.com/gianlucat97/Tesi-ReportFinali.git

Ciò che bisogna fare per la customizzazione è semplicemente dare dei nomi idonei alle variabili d'ambiente:

```
EMAIL_ACCOUNT_GITHUB: t*********@gmail.com
NOME_ACCOUNT_GITHUB: ********
PASSWORD_ACCOUNT_GITHUB: **********
```

Andando quindi ad aggiungere i report *xls* ottenuti dal progetto principale all'interno della cartella *Report-Separati* e facendone il push nel repository si avvierà l'esecuzione del file *generaReportFinale.yml*.

Al termine dell'esecuzione del file YAML si ottiene il seguente file *reportComplessivo.xls*:

TEST SUITE	TEST CASE	RELEASE	LOCATORE	ESITO TE	MESSAGG	CAUSA ROTTURA TEST
File: /home/runner/work/Tesi-ReportFinali/Tesi-ReportFinali/Hooks/tabellaReportTest_v2_2.xls					-	-
it.catalogo.test.SuiteCreateAndAssertHooks	testSuiteCreateAndAssertHooks_loc_Hooks_release_1_1	1_1	test-hooks	Superato	-	-
it.catalogo.test.SuiteUpdateClienteHooks	testSuiteUpdateClienteHooks_loc_Hooks_release_2_1	2_1	test-hooks	Superato	-	-
it.catalogo.test.SuiteCreateDeleteLocHooks	testSuiteCreateDeleteLocHooks_loc_Hooks_release_1_0	1_0	test-hooks	Superato	-	-
it.catalogo.test.SuiteTwoLinkHooks	testSuiteTwoLinkHooks_loc_Hooks_release_2_2	2_2	test-hooks	Superato	-	-
it.catalogo.test.SuiteUpdateLocHooks	testSuiteUpdateLocHooks_loc_Hooks_release_1_0	1_0	test-hooks	Superato	-	-
it.catalogo.test.SuiteCreateAndUpdateHooks	testSuiteCreateAndUpdateHooks_loc_Hooks_release_2_1	2_1	test-hooks	Superato	-	-
it.catalogo.test.SuiteCreateClienteHooks	testSuiteCreateClienteHooks_loc_Hooks_release_2_1	2_1	test-hooks	Superato	-	-
it.catalogo.test.SuiteCreateAndAssertClienteHooks	testSuiteCreateAndAssertClienteHooks_loc_Hooks_release_2_1	2_1	test-hooks	Superato	-	-
File: /home/runner/work/Tesi-ReportFinali/Tesi-ReportFinali/Hooks/tabellaReportTest_v1_0.xls					-	-
it.catalogo.test.SuiteCreateDeleteLocHooks	testSuiteCreateDeleteLocHooks_loc_Hooks_release_1_0	1_0	test-hooks	Superato	-	-
it.catalogo.test.SuiteUpdateLocHooks	testSuiteUpdateLocHooks_loc_Hooks_release_1_0	1_0	test-hooks	Superato	-	-
File: /home/runner/work/Tesi-ReportFinali/Tesi-ReportFinali/Hooks/tabellaReportTest_v2_1.xls					-	-
it.catalogo.test.SuiteCreateAndAssertHooks	testSuiteCreateAndAssertHooks_loc_Hooks_release_1_1	1_1	test-hooks	Superato	-	-
it.catalogo.test.SuiteUpdateClienteHooks	testSuiteUpdateClienteHooks_loc_Hooks_release_2_0	2_0	test-hooks	Presenta u	org.openq	Causa di errore obsolescenza/fragilità (specific
it.catalogo.test.SuiteCreateDeleteLocHooks	testSuiteCreateDeleteLocHooks_loc_Hooks_release_1_0	1_0	test-hooks	Superato	-	-
it.catalogo.test.SuiteUpdateLocHooks	testSuiteUpdateLocHooks_loc_Hooks_release_1_0	1_0	test-hooks	Superato	-	-
it.catalogo.test.SuiteCreateAndUpdateHooks	testSuiteCreateAndUpdateHooks_loc_Hooks_release_2_1	2_1	test-hooks	Superato	-	-
it.catalogo.test.SuiteCreateClienteHooks	testSuiteCreateClienteHooks_loc_Hooks_release_2_0	2_0	test-hooks	Presenta u	org.openq	Causa di errore obsolescenza/fragilità (specific
it.catalogo.test.SuiteCreateAndAssertClienteHooks	testSuiteCreateAndAssertClienteHooks_loc_Hooks_release_2_0	2_0	test-hooks	Presenta u	org.openq	Causa di errore obsolescenza/fragilità (specific
File: /home/runner/work/Tesi-ReportFinali/Tesi-ReportFinali/Hooks/tabellaReportTest_v1_1.xls					-	-
it.catalogo.test.SuiteCreateAndAssertHooks	testSuiteCreateAndAssertHooks_loc_Hooks_release_1_1	1_1	test-hooks	Superato	-	-
it.catalogo.test.SuiteCreateDeleteLocHooks	testSuiteCreateDeleteLocHooks_loc_Hooks_release_1_0	1_0	test-hooks	Superato	-	-
it.catalogo.test.SuiteUpdateLocHooks	testSuiteUpdateLocHooks_loc_Hooks_release_1_0	1_0	test-hooks	Superato	-	-
File: /home/runner/work/Tesi-ReportFinali/Tesi-ReportFinali/Hooks/tabellaReportTest_v2_0.xls					-	-
it.catalogo.test.SuiteCreateAndAssertHooks	testSuiteCreateAndAssertHooks_loc_Hooks_release_1_1	1_1	test-hooks	Superato	-	-
it.catalogo.test.SuiteUpdateClienteHooks	testSuiteUpdateClienteHooks_loc_Hooks_release_2_0	2_0	test-hooks	Superato	-	-
it.catalogo.test.SuiteCreateDeleteLocHooks	testSuiteCreateDeleteLocHooks_loc_Hooks_release_1_0	1_0	test-hooks	Superato	-	-
it.catalogo.test.SuiteUpdateLocHooks	testSuiteUpdateLocHooks_loc_Hooks_release_1_0	1_0	test-hooks	Superato	-	-
it.catalogo.test.SuiteCreateClienteHooks	testSuiteCreateClienteHooks_loc_Hooks_release_2_0	2_0	test-hooks	Superato	-	-
it.catalogo.test.SuiteCreateAndAssertClienteHooks	testSuiteCreateAndAssertClienteHooks_loc_Hooks_release_2_0	2_0	test-hooks	Superato	-	-

File: /home/runner/work/Tesi-ReportFinali/Tesi-ReportFinali/Tradizionale/tabellaReportTest_v2_2.xls					-	-
it.catalogo.test.SuiteCreateAndUpdate	testSuiteCreateAndUpdate_release_2_1	2_1	tradizionale	Superato	-	-
it.catalogo.test.SuiteCreateAndAssertCliente	testSuiteCreateAndAssertCliente_release_2_1	2_1	tradizionale	Presenta un Error	org.openqa.s	Causa di errore obsolescenza/fragilità (spec
it.catalogo.test.SuiteCreateDelete	testSuiteCreateDelete_release_2_1	2_1	tradizionale	Superato	-	-
it.catalogo.test.SuiteCreateCliente	testSuiteCreateCliente_release_2_1	2_1	tradizionale	Presenta un Error	org.openqa.s	Causa di errore obsolescenza/fragilità (spec
it.catalogo.test.SuiteUpdate	testSuiteUpdate_release_2_1	2_1	tradizionale	Superato	-	-
it.catalogo.test.SuiteCreateAndAssert	testSuiteCreateAndAssert_release_2_1	2_1	tradizionale	Superato	-	-
it.catalogo.test.SuiteTwoLink	testSuiteTwoLink_release_2_2	2_2	tradizionale	Superato	-	-
it.catalogo.test.SuiteUpdateCliente	testSuiteUpdateCliente_release_2_1	2_1	tradizionale	Presenta un Error	org.openqa.s	Causa di errore obsolescenza/fragilità (spec
File: /home/runner/work/Tesi-ReportFinali/Tesi-ReportFinali/Tradizionale/tabellaReportTest_v1_0.xls					-	-
it.catalogo.test.SuiteCreateDelete	testSuiteCreateDelete_release_1_0	1_0	tradizionale	Superato	-	-
it.catalogo.test.SuiteUpdate	testSuiteUpdate_release_1_0	1_0	tradizionale	Superato	-	-
File: /home/runner/work/Tesi-ReportFinali/Tesi-ReportFinali/Tradizionale/tabellaReportTest_v2_1.xls					-	-
it.catalogo.test.SuiteCreateAndUpdate	testSuiteCreateAndUpdate_release_2_1	2_1	tradizionale	Superato	-	-
it.catalogo.test.SuiteCreateAndAssertCliente	testSuiteCreateAndAssertCliente_release_2_0	2_0	tradizionale	Presenta un Error	org.openqa.s	Causa di errore obsolescenza/fragilità (spec
it.catalogo.test.SuiteCreateDelete	testSuiteCreateDelete_release_1_1	1_1	tradizionale	Presenta un Error	org.openqa.s	Causa di errore obsolescenza/fragilità (spec
it.catalogo.test.SuiteCreateCliente	testSuiteCreateCliente_release_2_0	2_0	tradizionale	Presenta un Error	org.openqa.s	Causa di errore obsolescenza/fragilità (spec
it.catalogo.test.SuiteUpdate	testSuiteUpdate_release_1_1	1_1	tradizionale	Presenta un Error	org.openqa.s	Causa di errore obsolescenza/fragilità (spec
it.catalogo.test.SuiteCreateAndAssert	testSuiteCreateAndAssert_release_1_1	1_1	tradizionale	Presenta un Error	org.openqa.s	Causa di errore obsolescenza/fragilità (spec
it.catalogo.test.SuiteUpdateCliente	testSuiteUpdateCliente_release_2_0	2_0	tradizionale	Presenta un Error	org.openqa.s	Causa di errore obsolescenza/fragilità (spec
File: /home/runner/work/Tesi-ReportFinali/Tesi-ReportFinali/Tradizionale/tabellaReportTest_v1_1.xls					-	-
it.catalogo.test.SuiteCreateDelete	testSuiteCreateDelete_release_1_0	1_0	tradizionale	Presenta un Error	org.openqa.s	Causa di errore obsolescenza/fragilità (spec
it.catalogo.test.SuiteUpdate	testSuiteUpdate_release_1_0	1_0	tradizionale	Presenta un Error	org.openqa.s	Causa di errore obsolescenza/fragilità (spec
it.catalogo.test.SuiteCreateAndAssert	testSuiteCreateAndAssert_release_1_1	1_1	tradizionale	Superato	-	-
File: /home/runner/work/Tesi-ReportFinali/Tesi-ReportFinali/Tradizionale/tabellaReportTest_v2_0.xls					-	-
it.catalogo.test.SuiteCreateAndAssertCliente	testSuiteCreateAndAssertCliente_release_2_0	2_0	tradizionale	Superato	-	-
it.catalogo.test.SuiteCreateDelete	testSuiteCreateDelete_release_1_1	1_1	tradizionale	Superato	-	-
it.catalogo.test.SuiteCreateCliente	testSuiteCreateCliente_release_2_0	2_0	tradizionale	Superato	-	-
it.catalogo.test.SuiteUpdate	testSuiteUpdate_release_1_1	1_1	tradizionale	Superato	-	-
it.catalogo.test.SuiteCreateAndAssert	testSuiteCreateAndAssert_release_1_1	1_1	tradizionale	Superato	-	-
it.catalogo.test.SuiteUpdateCliente	testSuiteUpdateCliente_release_2_0	2_0	tradizionale	Superato	-	-

Capitolo 8: Utilizzo progetto con un'altra AUT

Video tutorial per l'utilizzo dello strumento disponibile al seguente link:
https://www.youtube.com/watch?v=qkWTLIPaWEA

Da come già specificato, il seguente lavoro è composto dalle seguenti parti:

- Progetto "*test-hooks*" che permette di iniettare i localizzatori all'interno del codice front-end.
- Progetti di utility: *Tesi-injector-plugin, correzioneFormatoTest, miglioramentoReportTest*.
- File YAML di CI/CD (*main.yml*, *mainOnPush.yml*) per l'automazione dell'intero processo analizzato.

Se si vuole quindi utilizzare questo progetto su una propria AUT, bisogna tener conto delle seguenti considerazioni:

- Il *Front-End* deve essere sviluppato in *Angular, Freemarker, Twigs o Smarty*, ovvero le tecnologie supportate dal progetto *test-hooks*, dove il codice delle pagine web viene generato staticamente. Naturalmente è necessario andare ad editare i comandi per l'esecuzione del front-end, all'interno del file *startFrontEnd.sh* (che viene utilizzato dal file *main.yml*), a seconda della tecnologia utilizzata.
- Il *Back-End* può essere realizzato con qualsiasi tecnologia. Anche in questo caso è necessario andare a editare i comandi per l'esecuzione del back-end, all'interno del file *startBackEnd.sh* (che viene utilizzato dal file *main.yml*), a seconda della tecnologia utilizzata.

Tenendo conto delle precedenti considerazioni, ciò che rimane da fare è configurare opportunamente le 9 *Environment Variables* in Github.

Per dimostrare il corretto funzionamento del seguente lavoro di tesi di laurea si è deciso di testarlo con un *applicativo open-source* basato su un front-end (FE) *Angular* sviluppato da terzi, reperibile su github al seguente collegamento:
https://github.com/bbachi/angular-java-example

Una volta scelta la web app sotto test, ciò che bisogna fare può essere riassunto nei seguenti step:

1) Fare la fork del seguente repository: https://github.com/gianlucat97/Tesi-StrumentoGenerale.git

2) Dopo aver creato il proprio repository (tramite la fork), fare la clone in locale del repository creato.

3) Inserire all'interno della directory "insert-here-yuor-web-app" la cartella contenente il progetto della propria applicazione web.

4) Creare il virtual environment "*envForGithubActions*".
Per creare un *environments*, bisogna:

- recarsi nel proprio repository github;
- andare in *"Settings → Environments → New Environments"*;
- inserire *"envForGithubActions"* all'interno del form per la scelta del nome dell'environment;
- premere il button *"Configure environment"*;

5) Inserire le seguenti 6 variabili d'ambiente, customizzando i valori di esempio qui riportati in base al proprio caso d'uso:

```
EMAIL_ACCOUNT_GITHUB: t*********@gmail.com
NOME_ACCOUNT_GITHUB: g*********
PASSWORD_ACCOUNT_GITHUB: *********
FE_EXTENSION_TYPE: .html
GRAMMAR_TYPE: angularjs
DIR_FILE_FE: /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale/insert-here-your-web-app/root-frontend-web-app
```

NB: I GRAMMAR_TYPE consentiti sono: ['angularjs', 'html', 'php', 'smarty', 'twig', 'freemarker']

6) Customizzare i file startBackEnd.sh e startFrontEnd.sh, in base al proprio caso d'uso. Vediamo degli esempi,

startBackEnd.sh:

```
echo "Inizio comandi installazione precondizioni"
sudo apt update
sudo apt install openjdk-11-jdk openjdk-11-jre
echo "Installata versione di java numero"
java -version

echo "Inizio comandi esecuzione Backend"
cd /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale/insert-here-your-web-app/root/backend
mvn clean install
cd /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale/insert-here-your-web-app/root/backend/target
echo "Vediamo quali file jar si trovano in cartella target"
ls -a
java -jar backend-0.0.1-SNAPSHOT.jar &
```

startFrontEnd.sh:

```
echo "Inizio comandi installazione precondizioni"
cd /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale
curl -sL https://deb.nodesource.com/setup_12.x -o nodesource_setup.sh
cat nodesource_setup.sh
sudo bash nodesource_setup.sh
sudo apt install nodejs
echo "versione di node installata"
node -v
echo "Installazione di npm"
sudo apt install npm
echo "Fine comandi installazione Node"

echo "Inizio comandi esecuzione Frontend"
cd /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale/insert-here-your-web-app/root/frontend
echo "Siamo nella directory FE, proviamo a lanciarlo in esecuzione"
npm install
echo "Installazione npm effettuata, prossimo comando: npm start"
npm start &
```

7) Andare nel tab "Actions" ed attivare i "Workflows" cliccando sul button "I understand my workflows, go ahead and enable them".

8) Fare una modifica ad un file txt e pushare (e.g. aggiungere un carattere al readme.txt), in modo da triggerare l'esecuzione del file mainOnPush.yml

9) Attendere il termine dell'esecuzione del file mainOnPush.yml, il quale si occupa di fare la hook-injection all'interno dei file di FE.

10) Fare la pull in locale, in modo da poter avere anche sulla propria macchina i file di FE con i localizzatori iniettati.

11) Eseguire l'applicazione web in locale

12) Aprire Katalon Recorder e nella sezione "Extension script" aggiungere il file "attributeHooksLocators.js" reperibile nel corrente repository.

13) Registrare casi di test con Katalon Recorder ed esportarli in modalita (JUnit + WebDriver)

14) Si otterrà un file zip, estrarre il contenuto del file e pushare all'interno della directory *./project-test-headless/src/test/java/com/example/TesiIntegrazioneProgettoEsterno/* solamente i file di test (con estensione .java) estratti

15) Al push verrà triggerata l'esecuzione del file *mainOnPush.yml* che si occuperà di correggere il formato dei file di test pushati (in modo da renderli eseguibili all'interno di un container), attendere quindi il termine della sua esecuzione.

16) Creare una nuova release, triggerando quindi l'esecuzione del file *main.yml*

17) Al termine dell'esecuzione del file *main.yml*, all'interno del tag creato, nella directory *./TestSuite/nomeTagCreato* si troveranno tutti i report autogenerati inerenti ai test di regressione eseguiti.

Dopo aver seguito questi 17 passi il progetto sarà perfettamente configurato; quindi, ad ogni push eseguirà il file *mainOnPush.yml* che si occuperà di effettuare l'injection degli hooks e la correzione del formato dei file di test.
Ad ogni rilascio di una nuova release eseguirà invece il file *main.yml* che genererà i report dei test (in formato excel) andandoli a *pushare* in un'apposita directory, proprio come faceva il progetto originale.

Nell'esempio effettuato di riutilizzo del progetto su una seconda AUT, sono state create le seguenti variabili d'ambiente in *envForGithubActions*:

EMAIL_ACCOUNT_GITHUB: t*********@gmail.com
NOME_ACCOUNT_GITHUB: g*********
PASSWORD_ACCOUNT_GITHUB: *********
FE_EXTENSION_TYPE: .html
GRAMMAR_TYPE: angularjs
DIR_FILE_FE: /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale/insert-here-your-web-app/angular-java-example-master

Tabella 8. - 1: variabili d'ambiente per utilizzo progetto con altra AUT

Dopo aver configurato correttamente il nuovo progetto ed averlo inserito nel repository github:
https://github.com/gianlucat973/Tesi-StrumentoGenerale.git
si è proceduto con l'inserimento dei test Junit, autogenerati da Katalon Recorder, all'interno della directory:

/home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale/project-test-headless/src/test/java/com/example/TesiIntegrazioneProgettoEsterno.

Per poter eseguire l'applicativo web all'interno del container ubuntu in github actions, si è andato quindi a configurare gli script per l'esecuzione del *back-end* e del *front-end*:

```
echo "inserisci qui i comandi per lo starting del Back-End della tua applciazione web"
cd /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale/insert-here-your-web-app/angular-java-example-master
mvn clean install
cd /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale/insert-here-your-web-app/angular-java-example-master/target
echo "Vediamo quali file jar si trovano in cartella target"
ls -a
echo "Proviamo ad eseguire ${{ secrets.NOME_JAR_WEBAPP }}.jar"
java -jar users-0.0.1-SNAPSHOT.jar
```

Tabella 8. - 2: file *startBackEnd.sh*

```
echo "inserisci qui i comandi per lo starting del Front-End della tua applciazione web"
cd /home/runner/work/Tesi-StrumentoGenerale/Tesi-StrumentoGenerale/insert-here-your-web-app/angular-java-example-master/src/main/ui
echo "Siamo nella directory FE, proviamo a lanciarlo in esecuzione"
npm install
echo "Installazione npm effettuata, prossimo comando: npm start"
npm start
```

Tabella 8. - 3: file *startFrontEnd.sh*

In questo modo avremo in esecuzione il back-end sulla porta 8080 ed il front-end sulla porta 4200 del container; i file di test andranno così a testare la UI dell'applicativo web facendo richieste sulla porta 4200.

Al termine di queste configurazioni, il file "*main.yml*" è pronto per la generazione dei report alla creazione di un nuovo tag all'interno del repository.

A questo punto creando una nuova release chiamata "*versione_1_0*" viene eseguito lo script *main.yml* che va a generare correttamente tutti i report dei file di test inseriti in questo secondo progetto di esempio.

Una volta configurato il repository e creata la prima release, fino a quando non termina il ciclo di sviluppo dell'applicazione web sotto test bisogna seguire i seguenti passi:

1) Fare una pull, per avere sempre allineato il progetto in locale con il remote.

2) Scrivere nuovo codice web app.

3) Pushare le modifiche effettuate.

4) Attendere che il file mainOnPush termini la sua esecuzione (si occupa di iniettare nuovamente gli hooks a valle delle modifiche).

5) Eseguire la pull (per avere anche in locale i file di FE con gli hooks aggiornati).

6) Rieseguire applicazione web in locale.

7) Registrare nuovi casi di test con Katalon Recorder.

8) Esportare i nuovi casi di test (in Katalon in formato WebDriver+JUnit) e pushare i file (.java) ottenuti nella cartella ./project-test-headless/src/test/java/com/example/TesiIntegrazioneProgettoEsterno/

9) Attendere che il file mainOnPush.yml termini la sua esecuzione (si occupa di rendere i nuovi file di test eseguibili in modalità headless).

10) Creazione nuova release.

11) Attendere esecuzione file main.yml (che va ad eseguire ed autogenerare la reportistica dei test di regressione).

12) Nella directory ./TestSuite/nomeTagCreato si troveranno tutti i report autogenerati inerenti ai test di regressione eseguiti.
Analizzare il report in formato ".xls" ed andare a correggere e/o eliminare i test rotti.

13) Tornare al punto 1, se non è ancora terminato il ciclo di sviluppo dell'applicativo web.

Si ribadisce che è possibile visionare i risultati ottenuti in questo utilizzo del lavoro di tesi al seguente link:
https://github.com/gianlucat973/Tesi-StrumentoGenerale.git

In questo capitolo è stato quindi mostrato l'utilizzo del seguente lavoro di tesi di laurea su di un generico applicativo basato su una tecnologia (Angular) diversa da quella precedentemente testata (Freemarker).

Conclusioni

Nel corso del seguente lavoro si è mostrato uno strumento in grado di integrare ed automatizzare un robusto processo di testing, con generazione di una accurata reportistica, all'interno del ciclo di sviluppo di un applicativo web.

La corrente soluzione propone dunque uno strumento generale, che integrato con il proprio applicativo web, supporta lo sviluppo esonerando il programmatore dall'onere di andare ad eseguire i test di regressione volta per volta, ed aiutandolo nel verificare i loro esiti tramite la generazione automatica della reportistica.

Lo strumento provvede inoltre ad iniettare automaticamente dei localizzatori (*hooks*) all'interno del front-end dell'applicazione web, in modo da poter utilizzare dei casi di test che siano più robusti ed esenti da rotture dovute alla fragilità dei test.
L'obiettivo del seguente lavoro è stato quindi quello di supportare uno sviluppatore web lungo il ciclo di sviluppo del software, rendendolo più veloce, robusto ed efficiente.

L'intero lavoro sviluppato opera, tra l'altro, in un ambiente in cloud, ciò significa che l'integrazione del seguente progetto non comporterebbe alcun overhead sulla macchina dell'utente sviluppatore che decidesse di utilizzarlo.

Data la generalità del seguente lavoro di tesi, questo potrà essere integrato con qualunque tecnologia di back-end e con le tecnologie di front-end che supportano l'iniezione degli hooks (Freemarker, Angular, Smarty, Twigs).

A valle delle seguenti considerazioni, si conclude il seguente elaborato, consigliandone il suo utilizzo qualora si dovesse sviluppare un applicativo web e si necessiti di dover eseguire ripetutamente molti test di regressione, dovendo tenere traccia dei loro esiti lungo tutto il ciclo di sviluppo. È infatti in questi scenari che si manifesta la maggiore efficacia in termini di tempi e costi di sviluppo, che apporta l'integrazione del seguente lavoro.

Sviluppi futuri

Il corrente lavoro potrebbe consentire di automatizzare il processo di testing anche con un generico tipo di locatore, diverso dagli hooks e dai tradizionali.
Tenuto conto di ciò, un possibile futuro sviluppo di questo progetto potrebbe essere quello di integrarlo con nuovi software che si occupino di iniettare nuovi tipi di locatori. Si potrebbe quindi dare all'utilizzatore la scelta di effettuare l'iniezione con il locatore che desidera tra tutti quelli messi a disposizione.

Un altro possibile sviluppo potrebbe essere quello di fornire un supporto a casi di test scritti in diversi linguaggi di programmazione e quindi non utilizzanti esclusivamente il framework *Junit*. Si potrebbe poi pensare di migrare questo processo di *CI/CD* implementato su *Github* verso altre piattaforme di hosting basate sull'ecosistema *Git*.

Un'altra idea potrebbe poi essere quella di rendere compatibile lo strumento per piattaforme mobile, andando ad utilizzare quindi tecnologie alternative che si addicano allo sviluppo mobile.

Si potrebbe poi valutare l'idea di un utilizzo misto tra *hooks* e *XPath minimizzati* (e.g. tecnica *ROBULA* o similari) per tentare di abbassare ulteriormente l'impatto sui codici sorgenti delle pagine web.

Un'altra idea potrebbe essere quella di inglobare il processo di *Building Automation* implementato in *Github Actions* direttamente all'interno di un *IDE*.
Sarebbe infatti un'ottimizzazione dal punto di vista della velocità all'interno del ciclo di vita del software se fosse possibile far partire gli script YAML di automazione effettuando delle push o creando dei tag direttamente durante lo sviluppo all'interno del proprio IDE preferito.

Infine, un'altra possibile strada da intraprendere è quella di estendere lo strumento di reportistica implementato, rendendolo in grado di fornire informazioni ancora più accurate sullo stato dell'esecuzione dei casi di test effettati. Infatti, se si integrasse questo

strumento con altri già esistenti, si potrebbero ottenere delle informazioni importanti sul software testato, per esempio sarebbe molto utile poter avere le informazioni relative alla *code coverage* delle suite di test eseguite.

References

[1] Porfirio Tramontana and Anna Rita Fasolino: A Tool-Supported Technique the Generation of Robust Web Applications E2E Test Cases.

[2] Dott. Pierantonio Cangianiello – progetto https://gitlab.com/pcan/test-hooks

[3] http://www.testingreferences.com/testingtimeline.php

[4] http://www.testingreferences.com/testinghistory.php

[5] http://www.turingarchive.org/browse.php/b/8

[6] https://itnext.io/concept-evolution-of-software-testing-6fb1401b6df0

[7] https://itnext.io/concept-evolution-of-software-testing-part-2-229cf9309c49

[8] Rafi, D.M., Moses, K.R.K., Petersen, K., M¨antyl¨a, M.V.: Benefits and limitations of automated software testing: Systematic literature review and practitioner survey. In: 2012 7th International Workshop on Automation of Software Test (AST). pp. 36–42 (2012). https://doi.org/10.1109/IWAST.2012.6228988

[9] Leotta, M., Clerissi, D., Ricca, F., Tonella, P.: Capture-replay vs. programmable web testing: An empirical assessment during test case evolution. pp. 272–281 (2013). https://doi.org/10.1109/WCRE.2013.6671302

[10] Hammoudi, M., Rothermel, G., Tonella, P.: Why do record/replay tests of web applications break? In: 2016 IEEE International Conference on Software Testing, Verification and Validation (ICST). pp. 180–190 (April 2016). https://doi.org/10.1109/ICST.2016.16

[11] Parr, T.J.: Enforcing strict model-view separation in template engines. In: Proceedings of the 13th International Conference on World Wide Web. p. 224–233. WWW '04, Association for Computing Machinery, New York, NY, USA (2004). https://doi.org/10.1145/988672.988703, https://doi.org/10.1145/ 988672.988703

[12] Krasner, G.E., Pope, S.T.: A cookbook for using the model-view controller user interface paradigm in smalltalk-80. J. Object Oriented Program. 1(3), 26–49 (Aug 1988)

[13] Leotta, M., Stocco, A., Ricca, F., Tonella, P.: Reducing web test cases aging by means of robust xpath locators. pp. 449–454 (2014). https://doi.org/10.1109/ISSREW.2014.17

[14] Leotta, M., Stocco, A., Ricca, F., Tonella, P.: Robula+: An algorithm for generating robust xpath locators for web testing. Journal of Software: Evolution and Process 28(3), 177–204 (2016). https://doi.org/10.1002/smr.1771

[15] https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/

[16] https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions

[17] https://chrome.google.com/webstore/detail/katalon-recorder-selenium/ljdobmomdgdljniojadhoplhkpialdid

[18] https://www.ibm.com/cloud/learn/java-spring-boot

[19]https://blog.app-quality.com/it/crowd-testing-it/4-ragioni-per-integrare-il-crowd-testing-nelle-regolari-attivita-di-sviluppo-software

[20] https://www.mrw.it/java/cercare-sottostringhe-java_7271.html

[21] https://medium.com/bb-tutorials-and-thoughts/how-to-develop-and-build-angular-app-with-java-backend-87fb603c6e17

[22] https://github.com/bbachi/angular-java-example

[23] https://stackoverflow.com/questions/42007826/downgrade-gradle-from-3-3-to-2-14-1

Link *repository Github* del **lavoro di tesi**:

- https://github.com/gianlucat97/Tesi-StrumentoGenerale.git
- https://github.com/gianlucat97/Tesi-ReportFinali-StrumentoGenerale.git

Link video tutorial:

- https://www.youtube.com/watch?v=qkWTLIPaWEA

Link *repository Github* progetto **demo** sviluppato:

- https://github.com/gianlucat97/WebAppTesi.git
- https://github.com/gianlucat97/Tesi-ReportFinali.git

Link repository Github utilizzo progetto con un'altra **AUT**:

- https://github.com/gianlucat973/Tesi-StrumentoGenerale.git
- https://github.com/gianlucat973/Tesi-StrumentoGenerale.git