



**Regression test report**

**Partecipanti**:

|  |  |
| --- | --- |
| **Nome** | **Matricola** |
| Valenti Andrea | 0522500403 |
| Papaleo Giuseppe | 0522500465 |
| Tammaro Ruggero | 0522500512 |

|  |  |
| --- | --- |
| **Scritto da:** | Ruggero Tammaro |

# 1. Introduzione

Il seguente documento definisce in che modo avviene il test di regressione. Il test di regressione consiste nel ritestare le parti non modificate, e che quindi rimangono attive, del programma. I test case sono stati rieseguiti con lo scopo di verificare se le parti modificate e le parti aggiunte hanno introdotto o meno nuovi bug.

# 2. Vecchia Documentazione Testing

Il sistema legacy ereditato ha portato con se la documentazione relativa ai test case, in più nel codice era presente un file Junit che testava i vari metodi del programma e le varie featurse.

Come primo passo prima di eseguire i test abbiamo analizzato le due versione del sistema per capire quali parti del sistema sono state modificate, e quali invece sono state cambiate radicalmente. Dopo questa fase siamo passati all’analisi dei test case, per capire quali fossero ancora validi e quali invece avevano bisogno di essere analizzati per adattare il test case ai cambiamenti apportati. Dopo questa attenta analisi abbiamo scelto di non adottare una strategia “test all”, poiché alcune funzionalità del vecchio progetto verranno migrate in un secondo momento. I test Junit ricavati dal vecchio progetto sono stati rieseguiti sul nuovo progetto cosi da verificare che i componenti legacy funzionassero ancora dopo le modifiche apportate, e quindi la manutenzione non ha portato regressioni.

# 3. Failed Riscontrati

Durante la fase di test di regressione ci siamo ci siamo imbattuti in test case che sul vecchio progetto funzionavano mentre, sul nuovo davano un errore. In particolare il tast case con identificativo 10 del vecchio progetto, e il test con identificativo 14 sul nuovo, cioè la visulaizzazzione del credito residuo dopo aver acquistato un offerta. Sul vecchio progetto la visualizzazione era immediata dopo ogni acquisto e non dava nessun errore, mentre invece sul nuovo progetto i soldi scalati dal credito residuo erano soltanto quelli relativi all’ultimo prodotto acquistato, quindi il primo prodotto risultava acquistano ma non pagato.

Un altro errore riscontrato tramite la regressione riguarda il catalogo nel caso in cui risultasse vuoto, sul vecchio progetto non venivano riscontrati errori, mentre invece sul nuovo progetto questa eventualità ha mandato in crash il programma.

Infine effettuando i test Junit, è emerso che l’unica classe che è cambiata notevolmente è la classe Utente, dato che l’accesso avviene tramite una nuova classe managerAutenticazione.java che si interfaccia al DB e controlla i dati e se la login va a buon fine crea delle istanze di queste due classi. Mentre nel vecchio progetto esistevano due metodi di per ogni classe che accedevano direttamente ai file senza passare per nessuno strato di software come accade ora. Invece le altre classi hanno subito lievi modifiche hanno richiesto di conseguenza piccole modifiche ai test Junit ereditati.

# 4. Soluzioni adottate

La soluzione adottata al problema riguardante l’autenticazione dell’utente, è l’inserimento di una classe AutenticazioneServlet.java che va a verificare il tipo di utente che prova ad accedere e tramite il managerAutenticazione.java chiama il metodo login adatto che provvede a fare una chiamata al database e verificare se le credenziali sono corrette.

La soluzione adottata in caso del catalogo vuoto, prevede che il programma va in un pagina che comunica all’utente che attualmente non vi sono offerte presenti nel catalogo e evita casi di inconsistenza.