

Test Document

**Versione <1.00>**

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versione** | **Descrizione** | **Autori** |
| 27/02/2018 | 1.0 | Creazione del documento | Domenico Antonio Tropeano  Diego Avella |

# 

# Introduzione

Il testing è una delle tecniche di maggiore importanza per verificare la qualità del software, dato che permette di analizzare,valutare e promuovere il miglioramento della correttezza dell’implementazione facendo riferimento alle caratteristiche definite dal modello dei requisiti software. Lo scopo di questa attività è quello di provare il sistema e rilevare problemi; si cerca quindi di massimizzare e ottimizzare il numero di test effettuati, in modo tale da poter individuare un numero maggiore di errori.

La natura del testing è completamente diversa da quella delle altre attività di sviluppo software. Mentre analisi, design e implementazione sono attività costruttive , col testing si tenta di “rompere” il sistema. Questo documento contiene tutta la pianificazione del testing, contenente una descrizione dell’oggetto, l’approccio generale, le risorse che saranno utilizzate identificando i test item e definendo le attività di testing, i rischi e piani di emergenza e infine saranno definiti i deliverable delle attività di testing.

# 2. Relazione con gli altri documenti

In questo documento si farà stretto riferimento alla documentazione di progettazione. Dato che il progetto è stato fornito senza documentazione si terrà conto della specifica della *change request* e della struttura estratta in fase di analisi del progetto.

# 4. Funzionalità da testare e da non testare

Sarà effettuato un test di integrazione fra i manager ed i database. Verrà testato ogni metodo di ogni manager, controllando la variazioni di stato all’interno del database.

Per il testing si utilizzerà un database secondario, tramite uno script che permette di inizializzarlo in modo tale da poter rendere i test deterministici e ripetibili.

# Approccio

## Test di Integrazione

Per garantire la completa correttezza del layer di storage richiesto dalla change request è necessario effettuare alcuni test che controllino che i cambiamenti di stati interni al database siano corretti. Sarà necessario eseguire tali test su un database diverso da quello che verrà poi utilizzato nella pubblicazione. Inoltre il database deve essere inizializzato in maniera automatica, in modo da poter rendere la procedura di testing deterministica e ripetibile.

Verranno quindi testati tutti i metodi dei manager controllando per ognuno se viene prodotto il giusto cambiamento nel database. Il criterio di testing è quello di individuare delle classi di equivalenza, solo che in questo caso sono state testate solo le classi valide, in quanto i test sulle classi non valide vengono delegati ai test di sistema. Ciò è stato deciso dato che i controlli vengono effettuati nel layer superiore.

## Test di Sistema

Si è deciso di limitarsi allo storage layer in quanto eseguire test di integrazione con le servlet senza utilizzare le jsp aumenta spropositatamente il costo del testing senza porre ulteriori benefici e quindi si è pianificato di effettuare solo i test di sistema.

Tali test hanno il compito di verificare il funzionamento corretto del software oltre a verificare che i controlli nelle servlet adempiono alle specifiche. Si è deciso di effettuare un category partition con criterio weak equivalence.(Uno appartenente all’intervallo, uno non appartenente all’intervallo). Dato che alcune specifiche riguardano sia il formato che la lunghezza dell’input si è deciso di estendere il criterio ad **input corretto**, **input lungo**, **input malformato**

## Regression Test

È il test che viene eseguito prima di effettuare la modifica sul sistema. Lo scopo è quello di controllare che le componenti software non influenzata dalla change request mantengano lo stesso comportamento anche dopo che la modifica viene effettuata. Il regression test normalmente comprende i casi di test contenuti nel sistema da modificare, tali casi molto spesso non esistono (come nel nostro progetto). È necessario quindi effettuare un analisi per controllare e documentare quali funzionalità funzionano correttamente e quali no.

## Test di sistema pre-modifica

Dato che nel sistema oggetto della change request non è presente alcun tipo di test, per validare le funzionalità esistenti abbiamo creato dei test di sistema. Per effettuare la verifica della correttezza delle vecchie funzionalità è stato utilizzato **Selenium,** un framework per l’automazione dell’esecuzione dei test che coinvolgono l’utilizzo del browser. È possibile usare questo framework in due modi:

1. Usare l’IDE messo a disposizione dagli sviluppatori.
2. Usare un approccio code oriented utilizzando l’interfaccia WebDriver. Quest’ultima non è altro che un insieme di specifiche che permettono di interagire con il browser tramite feature OOP. Quest’ultima tecnica è stata scelta perché permette di lavorare facilmente con pagine dinamiche il cui contenuto cambia frequentemente.

### Elenco dei test di sistema eseguiti in pre-modifica:

L’elenco dei test case effettuati sul vecchio sistema sono i seguenti:

* *TestLoginCienteOk*
* *TestLoginGestoreOk*
* *TestLoginClienteFail*
* *TestRicercaCellOk*
* *TestRicercaCellFail*
* *TestInserisciCarrello*
* *TestAcquistaCell*
* *TestAddCellulare*
* *TestEliminaAccount*
* *TestModificaAccount*
* *TestModificaCellulare*
* *TestRicercaAvanzata*
* *TestRimuoviCellulare*
* *TestvisualizzaClienti*
* *TestProdottiEsaurimento*

I dettagli sulla specifica dei test case sopra citati sono inclusi nel documento relativo ai TestCase.

# Sospensione e Ripristino

Si è definito che se il 90% dei test ha successo può essere effettuata una delivery, se la percentuale e fra il 70% e 89%, l’implementazione viene sospesa e viene effettuata una revisione relativa ai casi di test che danno problemi. Se la percentuale è sotto al 69% il delivery viene rifiutato, e viene richiesta una revisione formale sia dell’implementazione sia dei test case generati. Tale percentuale vale solo per i test di sistema. I test di integrazione sul database devono avere tutti successo, in quanto hanno priorità maggiore.

# Materiale per il Testing

Per testare le modifiche dovute alla change request, verranno utilizzati vari tool di supporto al testing. La scelta dei suddetti è stata effettuata basandosi sulla loro stabilità, semplicità e maggiore utilizzo.

* ***JUnit:*** La libreria più utilizzata della tipologia xUnit in ambito Java. La versione presa in considerazione è la 4.12 e serve non solo per eseguire test di unità, ma è di largo utilizzo anche in altre tipologie di testing. Un problema riguardante JUnit è l’impossibilità di eseguire i test seguendo un particolare ordine di esecuzione e per ovviare a questo problema questa versione permette di stabilire un ordine lessicografico per eseguire i test tramite l’annotazione ***@FixMethodOrder***. Il tool è stato usato sia nei test di integrazione che nei test di sistema della nuova e della vecchia versione di quest’ultimo.
* ***Selenium:*** Le caratteristiche di questo framework sono state lungamente messe in evidenza durante il capitolo della fase di testing del vecchio sistema.
* ***Intellij IDEA Coverage Function:*** Questa funzionalità integrata all’interno dell’IDE permette di controllare la percentuale di metodi, classi, linee di codice coperte dai casi test creati. Il tool è stato molto utile perché ha permesso, man mano che i test venivano creati ed eseguiti, di correggere e di individuare immediatamente errori commessi o insiemi di righe di codice mai eseguite.

# Test Case

Si fa riferimento al file Test Case Specification

# Risultati Del Testing

Si fa riferimento ai file “Manager Test Result” e “System Test Result”

## Code Coverage

