

**INGEGNERIA DEL SOFTWARE 2016-2017**

**Test Plan 0.1**

**Fiorazon**

**16/12/2016**

**Sommario**

[**1.** **Introduzione** 3](#_Toc470732167)

[**2.** **Relazione con altri documenti** 3](#_Toc470732168)

[**3.** **Panoramica del sistema** 3](#_Toc470732169)

[**4.** **Funzionalità da testare/ da non testare** 4](#_Toc470732170)

[**5.** **Criteri di successo e di insuccesso** 5](#_Toc470732171)

[**6.** **Approccio** 6](#_Toc470732172)

[**7.** **Sospensione e Ripristino** 6](#_Toc470732173)

[**8.** **Testing materials** 7](#_Toc470732174)

[**9.** **Test Cases** 7](#_Toc470732175)

[**10.** **Testing Schedule** 7](#_Toc470732176)

[**11.** **Team** 7](#_Toc470732177)

[**12.** **Storia delle versioni** 7](#_Toc470732178)

1. **Introduzione**

Il Testing rappresenta una delle più importanti tecniche per verificare la qualità del software, in quanto consente di analizzare, valutare e promuovere il miglioramento della correttezza dell’implementazione con riferimento alle caratteristiche definite in particolare dal modello dei requisiti software. Lo scopo di tale attività è quello di provare il sistema e rilevare problemi; si intende dunque massimizzare il numero dei test effettuati, in modo tale da poter correggere un numero cospicuo di errori. E’ importante definire i casi d’uso su cui verranno testate le funzionalità del sistema. Tale attività va in contrasto con quelle svolte in precedenza: analisi, design, implementazione sono attività costruttive mentre il testing tenta di “rompere il sistema”.

Questo è il test plan document dell’applicazione web “Fiorazon”. L’obiettivo del test è quello di provare il corretto funzionamento dei metodi delle classi di controllo e manager del sistema.  
Si utilizzerà la modalità di test “black box” in quanto, data la semplicità dell’applicazione, si è interessati unicamente al comportamento delle singole classi.  
Si procederà in prima fase con il test di unità, successivamente verrà eseguito anche un test di integrazione per verificare la corretta interdipendenza delle varie classi.

1. **Relazione con altri documenti**

Per la redazione di questo documento si è dovuto far riferimento ai documenti RAD, SDD e ODD; questo per comprendere al meglio il funzionamento delle singole classi e i vincoli di inserimento dei valori (come ad esempio espressi nel RAD).   
Analizzando il documento RAD e in particolare i requisiti funzionali e non funzionali si sono potuti evidenziare i casi di test su tutte le operazioni riguardanti il carrello, il catalogo, l’ordine, l’utente.  
Per quanto riguarda i documenti SDD e ODD, sono serviti ad individuare le relazioni tra le classi e i metodi che le classi espongono.

**2.1 Relazioni con il RAD**  
I test dovranno tenere conto delle specifiche espresse nel RAD. Naturalmente, si darà una maggiore rilevanza ai “casi limite”, cioè tutti quei casi che sono al limite del nostro dominio di dati d’ingresso. Si darà rilevanza anche ai requisiti non funzionali ed ai vari modelli prodotti in fase di analisi dei requisiti.

**2.2 Relazioni con il SSD**  
Il testing dovrà garantire la coerenza tra il software e gli obiettivi di design definiti in fase di System Design, specificati nel SDD. Si analizzeranno, quindi, i conflitti e le inconsistenze presenti tra le componenti del sistema testate e tali obiettivi. Si analizzerà, inoltre, la struttura del sistema al fine di scoprire le differenze presenti tra essa e quella prevista durante la fase di System Design.

**2.3 Relazioni con l’ODD**  
La fase di testing dovrà considerare il contenuto del documento di Object Design, in quanto quest’ultimo rappresenta la base per la realizzazione dell’implementazione, fondamentale per il testing. Pertanto, sarà necessario effettuare il testing delle unità per individuare le differenze tra ciò che è stato stabilito in fase di Object Design ed il sistema effettivo.

1. **Panoramica del sistema**

Fiorazon è un sistema software capace di gestire un negozio di fiori caraterizzato da un’architettura Client/Server. Sono presenti diversi sottositemi, tra i quali quello che gestisce le operazioni dell’utente, quello per la gestione dei prodotti, quello per la gestione degli ordini e quello per la gestione e composizione del carrello. Ognuno di questi sottositemi, dopo essere stato diviso in componenti, sarà testato sia a livello di unità che a livelli più alti e comprensivi di più funzionalità.  
Un test generale del sistema, comprensivo di tutti i sottosistemi e relativo ad ogni interazione tra di essi sarà svolto solo durante il System Testing.   
Le caratteristiche da testare per il controllo del corretto funzionamento di ciascuna funzionalità saranno:   
• robustezza: capacità del sistema di reagire fornendo input non valido per il dominio applicativo;   
• usabilità: analisi di ogni forma di interazione corrisposta da messaggi di aiuto (in caso di errore) o di notifica (in caso di operazioni eseguite con successo) ;   
• sicurezza: verrà testato che un utente col solo interfacciamento grafico non possa intaccare dati non inerenti alla specifica dell’operazione o accedere a dati non consentiti;   
• correttezza: verrà testato che le operazioni vengano eseguite correttamente così come dalla specifica dei requisiti sono stati designati.

1. **Funzionalità da testare/ da non testare**

Si è deciso di testare soltanto alcune delle funzionalità che il sistema offre e che sono state implementate. Tali funzionalità sono quelle definite nelle fasi di Raccolta e di Analisi dei Requisiti e le funzionalità che garantiscono un controllo su queste ultime. Si è stabilito, dunque, di evitare il testing di gran parte delle funzionalità non richieste dal cliente o che servono come semplice supporto a quelle testate

**Funzionalità da testare:**

Le funzionalità da testare sono divisenelle seguenti categorie:

* Gestione utenti;
* Gestione ordini;
* Gestione carrelli;
* Gestione prodotti.

In particolare, per lo Unit Testing si testeranno tutti i metodi che rientrano in queste categorie e che, al tempo stesso, rappresentano funzionalità richieste dal cliente o funzionalità aggiuntive, non visibili all’utente, ma importanti, se non necessarie, per offrire le prime.  
Inerentemente l’Integration Testing, si svolgeranno test che aggregano in vari gruppi tutte le componenti e, quindi, le funzionalità già testate singolarmente. L’integrazione sarà svolta tenendo conto della categoria a cui le funzionalità appartengono e basandosi sulle interazioni tra tali funzionalità.   
Per il System Testing saranno svolti test sull’intero sistema, senza divisioni tra funzionalità diverse. Si testerà, infatti, l’insieme completo di funzionalità.

Per maggiore specificità, vanno testati tutti i metodi delle classi ControlloUtenti.java, UtentiManager.java, ControlloProdotti.java, ProdottiManager.java, ControlloCarrello.java, CarrelloManager.java, ControlloOrdini.java, OrdiniManager.java come esposto nell’ODD.  
Inoltre vanno testate le relazioni tra i metodi di ControlloUtenti.java e UtentiManager.java, tra ControlloProdotti.java e ProdottiManager.java, tra ControlloCarrello.java e CarrelloManager.java, tra ControlloOrdini.java e OrdiniManager.java. In dettaglio vanno testate le relazioni tra le seguenti coppie di metodi:   
ControlloCarrello.controlloAcquistoCarrello() – CarrelloManager.controlloAcquistoCarrello()  
ControlloCarrello.aggiungiProdottoCarrello() – CarrelloManager.aggiungiProdottoCarrello()

ControlloCarrello.rimuoviProdottoCarrello() – CarrelloManager.rimuoviProdottoCarrello()

ControlloProdotti.aggiungiProdotto() – ProdottiManager.aggiungiProdotto()  
ControlloProdotti.caricaProdotti() - ProdottiManager.caricaProdotti()   
ControlloProdotti.cancellaProdotto() - ProdottiManager.cancellaProdotto()   
ControlloProdotti.modificaProdotto() - ProdottiManager.modificaProdotto()   
ControlloProdotti.ricercaProdotti() ProdottiManager.ricercaProdotti()

ControlloUtenti.controlloCancella() - UtentiManager.controlloCancella()

ControlloUtenti.controlloLoginAmministratore() - UtentiManager.controlloLoginAmministratore  
ControlloUtenti.controlloLoginUtente() - UtentiManager.controlloLoginUtente  
ControlloUtenti.controlloModificaDati() - UtentiManager.controlloModificaDati()  
ControlloUtenti.controlloRegistrazione() - UtentiManager.controlloRegistrazione()

ControlloAcquisto.caricamentoOrdini() - AcquistoManager.caricamentoOrdini()  
ControlloAcquisto.controlloSpedizione() - AcquistoManager.controlloSpedizione()

ControlloAcquisto.controlloPagamento() - AcquistoManager.controlloPagamento()

**Funzionalità da non testare:**Le funzionalità che non saranno testate sono divise in più categorie:

− funzionalità appartenenti a componenti prettamente grafiche che non offrono funzionalità di spicco per il corretto funzionamento del sistema ;  
− funzionalità appartenenti a componenti molto semplici, il cui testing è implicito nella loro stessa struttura.

In particolar modo, non vanno testati i metodi delle classi Utente.java, Prodotto.java, Carrello.java, Ordine.java in quanto si tratta, per queste classi, unicamente di metodi “get” e “set”.

1. **Criteri di successo e di insuccesso**

Gli esiti dei test si baseranno su quanto detto nei documenti precedenti, in particolare nel RAD per quanto riguarda i vincoli di inserimento e nell’ODD per quanto riguarda l’output atteso dai metodi delle classi testate.

**5.1 Criteri di successo**

I criteri di successo del sistema software previsti per la fase di testing sono i seguenti:

− uguaglianza tra comportamento atteso e previsto dall’oracolo e comportamento effettivo della componente o dell’insieme di componenti;

− mancanza di errori dovuti a ragioni tecniche;

− mancanza di eccezioni generate a causa di una cattiva scrittura del codice.

**5.2 Criteri di insuccesso**I criteri di insuccesso del sistema software previsti per la fase di testing sono i seguenti:

− differenza, anche minima, tra comportamento atteso e previsto dall’utente e comportamento effettivo della componente o dell’insieme di componenti;

− presenza di errori dovuti a ragioni tecniche;

− presenza di eccezioni generate a causa di una cattiva scrittura del codice.

1. **Approccio**

Le tipologie di testing che si svolgeranno per testare il software LIMS saranno tre:   
• Unit Testing: un testing che coinvolge una singola unità del software e svolto indipendentemente dalle altre unità;

• Integration Testing: un testing che coinvolge più unità che interagiscono tra loro e che sono integrate per essere testate insieme;

• System Testing: un testing che coinvolge l’intero sistema e che quindi testa il software nella sua interezza, analizzando le interazioni tra tutte le unità, i gruppi di unità e le funzionalità .

**6.1 Unit testing**

Sarà effettuato un testing per ogni unità del software individuata. Per ogni componente verranno costruiti uno o più test case, di tipo “Black Box”. I test case “Black Box” saranno svolti analizzando unicamente il comportamento di I/O delle unità in varie situazioni, senza esaminare il codice. Per la scelta degli input dei test case, si definiranno delle classi di equivalenza che ben rappresentino i più comuni valori di input. Si è scelta quindi la strategia “Equivalence Testing”.

**6.2 Integration testing**Dopo aver testato le singole unità del software, ognuna di queste ultime sarà integrata insieme ad altre unità che interagiscono direttamente con essa. Si effettuerà, quindi, un testing che riguarderà gruppi di unità. Tale testing sarà incrementale e sarà di tipo “Top-down”.

Anche nella fase di integration testing verrà utilizzato un approccio “Black box”.

**6.3 System testing**

Dopo aver effettuato il testing d’integrazione, si svolgerà il testing dell’intero sistema software con l’uso di molteplici test case. Si analizzeranno, in particolare, le differenze tra il comportamento effettivo software e quello previsto, dettato dal cliente e descritto nel Documento di Analisi dei Requisiti. Sarà svolto, quindi, un testing funzionale, che riguarderà l’insieme delle funzionalità del software implementato. Anche in tal caso si costruiranno test case di tipologia “Black Box”.

1. **Sospensione e Ripristino**

Nel testing, potrebbero essereci casi in cui debba essere sospeso e/o ripristinato.

**7.1 Sospensione**

Durante la fase di Unit Testing, il testing sarà sospeso se si sono riscontrati errori o fallimenti riguardanti una unità del software e la sospensione riguarderà soltanto il testing di tale unità.

Se il testing di un’unità viene sospeso, si passerà alla correzione dell’unità.  
Cambiamenti ad un’unità del software non hanno influenza sullo Unit Testing delle altre. Durante le altre due fasi di testing, ovvero Integration Testing e System Testing, il testing sarà sospeso quando saranno riscontrati errori o fallimenti per una o più funzionalità testate. In tale caso, si procederà alla correzione del codice.

**7.2 Ripristino**

Il ripristino del testing di una componente o un gruppo di componenti del software sarà svolto quando gli errori riscontrati per tale componente o per tale gruppo di componenti saranno corretti. Inoltre, ad ogni ripristino di testing, saranno effettuati nuovi test su ciò che si è corretto.

1. **Materiale per il testing**

Le risorse software necessarie per il testing sono le seguenti:  
- Eclipse, come ambiente di sviluppo ed esecuzione di applicazioni Java;  
- Microsoft Office Word, come applicazione per la scrittura di test case e plan.

Le risorse hardware saranno i computer portatili dei membri del team.

1. **Test Cases**

Le varie fasi di testing necessiteranno ognuna di test case utili ad individuare errori ed anomalie sia analizzando il codice che provando la sua esecuzione.

Si sono individuate varie classi di equivalenza per ogni input che possa essere immesso per l’utilizzo di una o più componenti. In tal modo, è possibile sviluppare test case con input delle tipologie identificate per testare una o più unità.

Di seguito, sono elencate le classi di equivalenza che saranno prese in considerazione durante Unit Testing, Integration Testing e System Testing per svilupparei test case.

**9.1 Utenti**

**Registrazione e modifica dati profilo**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Classe** | **Valido** | **Classe** | **Non Valido** |
| **Nome** | CE01 | Nome ≠ ‘’  Non contiene numeri | CE02 | Nome = ‘’ Contiene numeri |
| **Cognome** | CE01 | Cognome ≠ ‘ ’  Contiene numeri | CE02 | Cognome =’’  Contiene numeri |
| **e-Mail** | CE03 | e-Mail ≠ ‘ ’ tipo = [indirizzo@dominio.it(com](mailto:indirizzo@dominio.it(com) ecc.) | CE04 | e-Mail = ‘’  tipo ≠ [indirizzo@dominio.it(com](mailto:indirizzo@dominio.it(com) ecc.) |
| **Codice fiscale** | CE05 | CF ≠ ‘ ’ | CE06 | CF = ‘’ |
| **Data di nascita** | CE07 | Data di nascita ≠‘ ’  Data di nascita < Data corrrente Età >= 18 | CE08 | Data di nascita = ‘ ’  Data di nascita ≥ Data corrrente Età < 18 |
| **Città di nascita** | CE09 | Città di nascita ≠‘ ’ | CE10 | Città di nascita = ‘ ’ |
| **Città di residenza** | CE11 | Città di residenza ≠‘ ’ | CE12 | Città di residenza = ‘ ’ |
| **Via** | CE13 | Via ≠‘ ’ | CE14 | Via = ‘’ |
| **Numero Civico** | CE15 | Numero Civico ≠‘ ’  Numero civico > 0 | CE16 | Numero civico = ‘’ Numero civico <= 0 |
| **Provincia** | CE17 | Provincia ≠‘ ’ Provincia esistente | CE18 | Provincia =’’ Provincia non esistente |
| **CAP** | CE19 | CAP≠‘ ’ | CE20 | CAP=’’ |
| **Username** | CE21 | Username≠‘ ’ | CE22 | Username=’’ |
| **Password** | CE23 | Password≠‘ ’ | CE24 | Password=’’ |

**Login utente**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Classe** | **Valido** | **Classe** | **Non valido** |
| **Username** | CE25 | Username≠‘ ’ Username presente nel db | CE26 | Username=’’ Username non presente nel db |
| **Password** | CE25 | Password≠‘ ’  Password presente nel db | CE26 | Password=’’ Password non presente nel db |

**9.2 Prodotti  
Ricerca prodotto**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Classe** | **Valido** | **Classe** | **Non valido** |
| **Nome** | CE27 | Nome non contiene numeri Nome presente nel db | CE28 | Nome contiene numeri Nome non presente nel db |
| **Prezzo** | CE27 | Prezzo >0 | CE28 | Prezzo <=0 |

**Aggiunta prodotto al catalogo e modifica del prodotto**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Classe** | **Valido** | **Classe** | **Non valido** |
| **Nome** | CE29 | Nome ≠ ‘’  Non contiene numeri | CE30 | Nome = ‘’  Contiene numeri |
| **Quantità** | CE31 | Quantità≠ ‘’  Quantità>0 | CE32 | Quantità=’’ Quantità <=0 |
| **Descrizione** | CE33 | Descirizone≠ ‘’ | CE34 | Descrizione=’’ |
| **Prezzo** | CE35 | Prezzo≠ ‘’  Prezzo=0 | CE36 | Prezzo=’’  Prezzo<=0 |

**Cancellazione di un prodotto dal catalogo**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Classe** | **Valido** | **Classe** | **Non valido** |
| **Nome** | CE37 | Nome ≠ ‘’  Non contiene numeri  Nome presente nel database | CE38 | Nome = ‘’  Contiene numeri  Nome non presente nel database |

**9.3 Ordini**

**Pagamento**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Classe** | **Valido** | **Classe** | **Non valido** |
| **Numero di carta di credito** | CE39 | Numero di carta di credtio≠ ‘’  Formato valido | CE40 | Numero di carta di credito=’’ Formato non valido |

1. **Testing Schedule**

**10.1 Gestione dei rischi**I possibili rischi generati dalle attività di testing sono stati minimizzati diminuendo le componenti del sistema da implementare e, quindi, testare. Inoltre, effettuando un testing di tipo funzionale viene limitato lo sviluppo di stub e driver per il testing delle singole componenti e quindi l’introduzione di nuovi errori nel nuovo codice di cui si sarebbero composti.

D’altro canto il testing di funzionalità rallenta l’individuazione di errori qualora un caso di test avesse esito positivo, poiché l’utilizzo di più componenti per il test di una singola funzionalità estranea dall’ipotesi di totale correttezza di ogni componente interessata.

Il risultato atteso è di riscontrare al più un errore parziale per funzionalità: le componenti interagiscono fra loro e la funzionalità viene eseguita ma non in modo completamente corretto.

Qualora la fase di testing evidenziasse un numero di errori maggiore rispetto alla media attesa, viene pianificato un impegno maggiore dei membri del team sulle attività di testing ed in casi estremi l’abbandono delle altre attività finché errori gravi (funzionalità non corretta, risultati errati, modifiche apportate in modo errato) non vengano risolti.

**10.2 Organizzazione delle attività**

Le attività di testing devono svolgersi sulle singole funzionalità divise nei livelli di suddivisione del sistema, rispettando le direttive indicate dal documento di system design.

**10.3 Schedulazione delle attività**  
Le attività di testing sono previste per il seguente periodo:  
- inizio testing: 4 gennaio;  
- fine testing: 13 gennaio.

1. **Team**

|  |  |
| --- | --- |
| **Nome** | **Matricola** |
| Giuseppe Siani | 0512102958 |
| Carmine D’ Alessandro | 0512103402 |
| Aleandro Giuseppe Libano | 0512102976 |

1. **Storia delle versioni**

|  |  |  |  |
| --- | --- | --- | --- |
| DATA | VERSIONE | DESCRIZIONE | AUTORE |
| 16/12/2016 | 0.1 | Parziale inserimento dei punti del documento | Giuseppe Siani, Carmine D’Alessandro |
| 29/12/2016 | 1.0 | Revisione dei precedenti punti e inserimento dei punti 8,9,10. | Aleandro Giuseppe Libano |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |