DODO.NET

TP

Test Plan

|  |  |
| --- | --- |
| Riferimento |  |
| Versione | 2.0 |
| Data | 7/12/2021 |
| Destinatario | Prof.ssa F. Ferrucci |
| Presentato da | Alfonso Cuomo  Simone Farina |
| Approvato da |  |

RevisionHistory

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versione** | **Descrizione** | **Autori** |
| 07/12/2021 | 1.0 | Stesura documento | [Alfonso Cuomo, Simone Farina] |
| 08/12/2021 | 2.0 | Modifiche dei pass/fail criteria | [Alfonso Cuomo,  Simone Farina] |

Sommario

1. Introduzione
2. Relazione con altri documenti

2.1 Relazione con il documento di analisi

2.2 Relazione con il System Design Document

2.3 Relazione con l’Object Design Document

1. Panoramica del sistema

4. Features da testare/da non testare

5. Pass/Fail criteria

6. Approccio

7. Sospensione e ripristino

8. Materiale di testing

9. Test Cases

10. testing schedule

11. Glossario

1. Introduzione

Dodo.net è un e-commerce pensato per semplificare le interazioni tra il bibliotecario e i clienti intenzionati all’acquisto di libri, mediante l’utilizzo di una piattaforma digitale mirata proprio per aiutare tali individui.

Il documento di test Plan ha l’obiettivo di supportare la fase di testing , ossia c’è bisogno di poter rilevare eventuali errori che sono stati prodotti durante la fase di decomposizione di sottosistemi, in modo tale da poter evitare che si verifichino errori nel momento in cui il sistema verrà utilizzato dall’utente finale.

Abbiamo pensato a opportuni casi e dati di input specifici in grado di mettere alla prova ogni singola funzionalità e caratteristica offerta dalla piattaforma.

I risultati dei test che verranno eseguiti saranno il punto cruciale nell’analisi delle failure e delle loro cause (fault) per individuare dove bisognerà intervenire per correggere gli errori o apportare modifiche per il miglioramento dei vari sottosistemi. Lo scopo è quindi stabilire la verità sulla corrispondenza tra comportamento atteso e comportamento osservato dalla piattaforma Dodo.net.

Come linea guida seguiremo gli use case del RAD, per cui le attività di testing saranno effettuate per le seguenti gestioni:

* Gestione Utente
* Gestione Admin

Per ogni use case viene prodotto un test case plan dove vengono individuate le classi di equivalenza di ogni input.

Per ogni input vi sarà una tabella che descriverà le scelte per ogni classe di equivalenza e infine, vi sarà una tabella che riassumerà tutte le possibili combinazioni di errore (e non) per il test case.

2.Relazione con altri documenti

Per la corretta individuazione dei test case, si fa riferimento ad altri documenti prodotti.

**2.1 Relazione con il documento di analisi (RAD)**

La progettazione dei casi di test avviene prescindendo dalla conoscenza della struttura interna del prodotto ed operando solo sulle specifiche. Per questo motivo facciamo riferimento al contenuto del documento di analisi che descrive dettagliatamente le funzionalità del sistema attraverso

scenari e use case, in relazione anche ai requisiti funzionali.

**2.2 Relazione con il System Design Document (SDD)**

Nel system design document abbiamo definito la suddivisione in sottosistemi relativamente al prodotto che intendiamo presentare. Il sistema è suddiviso in tre livelli logici: model, view e control. Ogni livello è composta da vari sottosistemi.

Pianificheremo le attività di testing relative alle funzionalità garantite nei sottosistemi specificati all’interno del System Design Document relativamente al livello control.

**2.3 Relazione con il Object Design Document (ODD)**

Attualmente non è presenta alcun documento di Object Design.

3.Panoramica del sistema

Il sistema proposto basa la sua architettura sul sistema three-tier, in particolare usando MVC e

quindi un sistema Model View Control.

Verranno usati HTML5, CSS3 per la parte di front-end e la generazione delle view.

Per la logica applicativa e quindi il back-end sarà utilizzato JSP.

Per la gestione del database saranno usati:

* JDBC per il collegamento al database.
* Tomcat e SQL per il database su cloud in fase di deployment.

4.Features da testare/da non testare

Di seguito la lista delle features di cui si effettuerà il testing per le varie gestioni:

* Gestione Utente
  + Registrazione
  + Login
* Gestione Admin
  + Inserimento prodotto nel sistema

Le funzionalità di cui non si andrà ad effettuare le attività di testing riguardano requisiti funzionali che non prevedono input manuale da parte dell’utente - ad esempio attività riguardanti esclusivamente visualizzazioni di dati.

5.Pass/Fail criteria

Le attività di testing sono mirate ad identificare la presenza di faults (errori) all’interno del sistema, per effettuarne un successivo intervento di eliminazione.

L’esito di un test case è valutato mediante un oracolo, inteso come il risultato atteso della sua esecuzione, basandosi sui requisiti.

Un test ha successo (pass) se, dato un input al sistema, l’output ottenuto è diverso dall’output atteso dall’oracolo. Un test fallisce (fail) se, dato un input al sistema, l’output ottenuto è uguale all’output atteso dall’oracolo.

Tutto il testing sarà considerato valido se tutti i seguenti vincoli saranno rispettati:

* Testare tutti i requisiti funzionali ad alta priorità;
* Effettuare test di regressione ogni volta che si introducono nuove caratteristiche al sistema o vengono modificate quelle presenti;
* Raggiungere un branch coverage non inferiore al 75%

6.Approccio

Il testing dell’intero sistema si compone di tre fasi: testing di sistema, testing di integrazione e testing di unità. Verranno progettati nell’ordine appena definito, ma verranno eseguiti in ordine inverso.

Prima della fase di implementazione del sistema, avverrà la progettazione dei casi di test di sistema, perfezionati in seguito nella loro fase di esecuzione; durante la fase implementativa avverrà la progettazione dei casi di test di unità.

Durante lo sviluppo saranno eseguite periodiche attività di revisione sul codice prodotto.

Poiché la progettazione è organizzata seguendo un modello simile al modello a V, il testing di sistema è stato pianificato in seguito alla stesura del documento Requirements Analysis Document, mentre la pianificazione del testing di integrazione avverrà dopo la stesura del System Design Document.

**Testing di Sistema**

Per il testing di sistema sarà utilizzato il tool Selenium IDE, che permette di registrare le azioni che un utente può intraprendere sul browser, in modo da poter implementare ed eseguire i test case di sistema. Il server, per la fase di testing, verrà deployato in localhost.

***Functional testing***

Il functional testing ha il fine di validare i requisiti funzionali. Consiste nell’individuare i possibili faults generati dagli input degli utenti.

***Performance Testing***

A causa del basso budget a disposizione, non si assicura l’esecuzione del performance testing.

***Pilot Testing***

A causa del basso budget a disposizione, non si assicura l’esecuzione del pilot testing.

***Acceptance Testing***

L’acceptance testing verrà effettuato solo sul functional testing, ed il Project Manager simulerà la figura del cliente.

***Installation Testing***

A causa del basso budget a disposizione, non si assicura l’esecuzione dell’installation testing.

***Testing di integrazione***

Verrà utilizzato un approccio bottom-up, metodo ritenuto più adatto per un software basato sul

paradigma Object Oriented. La definizione dei test case avverrà tramite il framework JUnit, mentre verrà usato Mockito per il mocking. Verrà valutato l’utilizzo di Github Actions per realizzare la Continuous Integration, in alternativa a Travis CI. L’automatizzazione del run dei test sarà gestita da JUnit, ed infine come tool di misurazione e report coverage sarà utilizzato JUnit. Il test di integrazione sarà il medesimo per tutte le componenti da testare. Nello specifico, si procederà prima con il test delle classi Service, e successivamente con il test delle classi Controller. Durante questa seconda esecuzione, la chiamata al controller sarà mockata usando Mockito. Di seguito viene presentato un esempio grafico di test di integrazione diviso nei due steps.

7.Sospensione e ripristino

In questa sezione verranno specificati i criteri di sospensione del test e le attività di test che dovranno essere ripetute quando si riprende il test.

**Criteri di sospensione**

Il testing non verrà sospeso fino alla sua terminazione, anche in caso di rilevazione di una failure. Il testing potrà essere momentaneamente sospeso nel caso venga restituito, al momento dell’esecuzione, un errore nella definizione di uno dei test stessi.

**Criteri di ripristino**

Il testing verrà ripreso dopo aver risolto i fault individuati.

8.Materiale di testing

L’hardware necessario per l’attività di test è un semplice computer, non necessariamente connesso ad internet, in quanto il sistema non è stato ancora rilasciato.

9. Test Cases

**9.1. TC\_1.1 Registrazione**

|  |  |
| --- | --- |
| **Parametro:** E-mail  **Formato:** [A-z0-9.+-]+@[A-z0-9.-]+.[A-z]{2,6} | |
| categorie | scelte |
| **Lunghezza (ma)** | 1: lunghezza == 0 [ errore]  2: lunghezza > 0 [property lunghezzaMAok] |
| **Formato** **(il)** | 1: rispetta il formato [ if lunghezzaMAok][property formatoILok]    2: non rispetta il formato [ if lunghezzaMAok] [ errore ] |
| **Esiste (es)** | 1.esiste nel db [if lunghezzaMAok and formatoILok] [errore]  2.non esiste nel db [if lunghezzaMAok and formatoILok] [property esisteESok] |

|  |  |
| --- | --- |
| **Parametro:** Nickname  **Formato:** /^[a-z ,.'-]+$/i | |
| categorie | scelte |
| **Lunghezza (ni)** | 1: lunghezza == 0 [ errore]  2: lunghezza > 0 [property lunghezzaNIok] |
| **Formato** **(ck)** | 1: rispetta il formato [ if lunghezzaNIok][property formatoCKok]  2: non rispetta il formato [ if lunghezzaNIok] [ errore ] |

|  |  |
| --- | --- |
| **Parametro:** Password  **Formato:** ^(?=.[A-Za-z])(?=.\d)[A-Za-z\d]{8,}$ | |
| categorie | scelte |
| **Lunghezza (pa)** | 1: lunghezza == 0 [ errore]  2: lunghezza >= 8 [property lunghezzaPAok] |
| **formato (sw)** | 1: rispetta il formato [if lunghezzaPAok]  [property formatoSWok]  2: non rispetta il formato [if lunghezzaPAok] [errore] |

|  |  |  |
| --- | --- | --- |
| **Codice** | **Combinazione** | **Esito** |
| TC\_1.1\_01 | ma1 | errore |
| TC\_1.1\_02 | ma2il2 | errore |
| TC\_1.1\_03 | ma2il1es2 | errore |
| TC\_1.1\_04 | ma2il1es1ni1 | errore |
| TC\_1.1\_05 | ma2il1es1ni2ck2 | errore |
| TC\_1.1\_06 | ma2il1es1ni2ck1pa1 | errore |
| TC\_1.1\_07 | ma2il1es1ni2ck1pa2sw2 | errore |
| TC\_1.1\_08 | ma2il1es1ni2ck1pa2sw1 | registrazione |

**9.2. TC\_16.16 Inserimento prodotto nel sistema**

|  |  |
| --- | --- |
| **Parametro:** ISBN  **Formato:** [ 0-9 ] {13} | |
| categorie | scelte |
| **Lunghezza (is)** | 1: lunghezza == 0 [ errore ]  2: lunghezza == 13 [property lunghezzaISok]  3: lunghezza > 13 [errore] |
| **Formato (bn)** | 1: rispetta il formato [ if lunghezzaISok][property formatoBNok]    2: non rispetta il formato [ if lunghezzaISok] [ errore ] |
| **Esiste (IB)** | 1.esiste nel db [if lunghezzaISok and formatoBNok] [errore]  2.non esiste nel db [if lunghezzaISok and formatoBNok] [property esisteIBok] |

|  |  |
| --- | --- |
| **Parametro:** Titolo  **Formato:** [ 0-9 A-Z a-z ] | |
| categorie | scelte |
| **Lunghezza (ti)** | 1: lunghezza == 0 [ errore]  2: lunghezza >= 1 [property lunghezzaTIok] |
| **Formato (to)** | 1: rispetta il formato [ if lunghezzaTIok][property formatoTOok]    2: non rispetta il formato [ if lunghezzaTIok] [ errore ] |

|  |  |
| --- | --- |
| **Parametro:** Autore  **Formato:** [ A-Z a-z ] | |
| categorie | scelte |
| **Lunghezza (au)** | 1: lunghezza == 0 [ errore]  2: lunghezza >= 1 [property lunghezzaAUok] |
| **Formato (re)** | 1: rispetta il formato [ if lunghezzaAUok][property formatoREok]    2: non rispetta il formato [ if lunghezzaAUok] [ errore ] |

|  |  |
| --- | --- |
| **Parametro:** Genere  **Formato:** [ A-Z a-z ] | |
| categorie | scelte |
| **Lunghezza (ge)** | 1: lunghezza == 0 [ errore]  2: lunghezza >= 1 [property lunghezzaGEok] |
| **Formato (ne)** | 1: rispetta il formato [ if lunghezzaGEok][property formatoNEok]    2: non rispetta il formato [ if lunghezzaGEok] [ errore ] |

|  |  |
| --- | --- |
| **Parametro:** Descrizione  **Formato:** [ 0-9 A-Z a-z ] | |
| categorie | scelte |
| **Lunghezza (de)** | 1: lunghezza == 0 [ errore]  2: lunghezza >= 1 [property lunghezzaDEok] |
| **Formato (sc)** | 1: rispetta il formato [ if lunghezzaDEok][property formatoSCok]    2: non rispetta il formato [ if lunghezzaSCok] [ errore ] |

|  |  |
| --- | --- |
| **Parametro:** Copertina  **Formato:** [ .jpeg png ] {1} | |
| categorie | scelte |
| **Formato (pe)** | 1: rispetta il formato [property formatoPEok]    2: non rispetta il formato [ errore ] |

|  |  |
| --- | --- |
| **Parametro:** Prezzo  **Formato:** [ 0-9 ] | |
| categorie | scelte |
| **Lunghezza (pr)** | 1: lunghezza == 0 [ errore]  2: lunghezza >= 1 [property lunghezzaPRok] |
| **Formato (zo)** | 1: rispetta il formato [ if lunghezzaPRok][property formatoZOok]    2: non rispetta il formato [ if lunghezzaPRok] [ errore ] |

|  |  |  |
| --- | --- | --- |
| **Codice** | **Combinazione** | **Esito** |
| TC\_16.16\_01 | is1 | errore |
| TC\_16.16\_02 | is3 | errore |
| TC\_16.16\_03 | is2 bn2 | errore |
| TC\_16.16\_04 | is2 bn1 ib1 | errore |
| TC\_16.16\_05 | is2 bn1 ib2 ti1 | errore |
| TC\_16.16\_06 | is2 bn1 ib2 ti2 to2 | errore |
| TC\_16.16\_07 | is2 bn1 ib2 ti2 to1 au1 | errore |
| TC\_16.16\_08 | is2 bn1 ib2 ti2 to1 au2 re2 | errore |
| TC\_16.16\_09 | is2 bn1 ib2 ti2 to1 au2 re1 ge1 | errore |
| TC\_16.16\_10 | is2 bn1 ib2 ti2 to1 au2 re1 ge2 ne2 | errore |
| TC\_16.16\_11 | is2 bn1 ib2 ti2 to1 au2 re1 ge2 ne1 de1 | errore |
| TC\_16.16\_12 | is2 bn1 ib2 ti2 to1 au2 re1 ge2 ne1 de2 sc2 | errore |
| TC\_16.16\_13 | is2 bn1 ib2 ti2 to1 au2 re1 ge2 ne1 de2 sc1 pr1 | errore |
| TC\_16.16\_14 | is2 bn1 ib2 ti2 to1 au2 re1 ge2 ne1 de2 sc1 pr2 zo2 | errore |
| TC\_16.16\_15 | is2 bn1 ib2 ti2 to1 au2 re1 ge2 ne1 de2 sc1 pr2 zo1 pe2 | errore |
| TC\_16.16\_16 | is2 bn1 ib2 ti2 to1 au2 re1 ge2 ne1 de2 sc1 pr2 zo1pe1 | aggiornato |

**9.4 TC\_2.2 Login Utente**

|  |  |
| --- | --- |
| **Parametro:** E-mail  **Formato:** [A-z0-9.+-]+@[A-z0-9.-]+.[A-z]{2,6} | |
| categorie | scelte |
| **Lunghezza (ma)** | 1: lunghezza == 0 [errore]  2: lunghezza > 0 [property lunghezzaMAok] |
| **Formato** **(il)** | 1: rispetta il formato [ if lunghezzaMAok][property formatoILok]    2: non rispetta il formato [ if lunghezzaMAok] [ errore ] |
| **Esiste (es)** | 1.esiste nel db [if lunghezzaMAok and formatoILok] [errore]  2.non esiste nel db [if lunghezzaMAok and formatoILok] [property esisteESok] |

|  |  |
| --- | --- |
| **Parametro:** Password  **Formato:** ^(?=.[A-Za-z])(?=.\d)[A-Za-z\d]{8,}$ | |
| categorie | scelte |
| **Lunghezza (pa)** | 1: lunghezza == 0 [ errore]  2: lunghezza >= 8 [property lunghezzaPAok] |
| **Esiste (ep)** | 1. esiste nel DB [if lunghezzaPAok and esisteESok] [property esisteEPok] 2. non esiste nel DB [if lunghezzaLPok and esisteESok] [errore] |
| **Corrisponde (cp)** | 1. corrisponde alla password della mail indicata [if esisteEPok] [property corrispondeCPok] 2. non corrisponde alla password della mail indicato [if esisteEPok][errore] |

|  |  |  |
| --- | --- | --- |
| **Codice** | **Combinazione** | **Esito** |
| TC\_2.2\_01 | ma1 | errore |
| TC\_2.2\_02 | ma2,il2 | errore |
| TC\_2.2\_03 | ma2, il1, es1 | errore |
| TC\_2.2\_04 | ma2, il1, es2, pa1 | errore |
| TC\_2.2\_05 | ma2, il1, es2, pa2, ep2 | errore |
| TC\_2.2\_06 | ma2, il1, es2, pa2, ep1, cp2 | errore |
| TC\_2.2\_07 | ma2, il1, es2, pa2, ep1, cp1 | login |

10.Testing schedule

Le attività di pianificazione del testing avverranno come definito nei capitoli precedenti, cioè subito dopo la fase di design necessaria per la pianificazione.

La scrittura dei casi di test avverrà in contemporanea con lo sviluppo del codice.

L’esecuzione dei test avverrà sia durante che dopo l’implementazione del sistema. Una volta concluso lo sviluppo, tutti i test saranno rieseguiti per garantirne il corretto funzionamento e produrre i report finali.

Per altre informazioni si rimanda ai documenti di management sullo schedule.

11.Glossario e definizioni

**Test Case Name:** test case univoco per distinguere i test case attuali dagli altri. ( la convenzione usata è TC\_[N].[M]\_[O], dove n è il numero dello use case, M quello per lo scenario e O indica il numero progressivo in funzione di N.

**Test Path:** la path del file di test che verrà effettuato.

**Input:** rappresenta la classe di equivalenza che è stata testata.

**Oracle:** sono i dati o comportamenti attesi dal sistema ( lo stato finale della situazione ).

**Log:** il comportamento osservato in seguito all’esecuzione del test.

**Pre-Condition:** è la precondizione dello use case.

**Flow of events:** è il flusso delle interazioni tra sistema e utente. Il flusso non è strettamente collegato allo use case. Bisogna specificare i dati effettivi che si usano e che vengono restituiti.