Test Plan

EasyExpo

Riferimento	
Versione	1.0
Data	09/12/2020
Destinatario	Prof.ssa Ferrucci Filomena
Presentato da	Gilberto Recupito

Approvato da	

Revision History

Data	Versione	Cambiamenti	Autore
09/12/2020	1.0	Stesura documento	Recupito Gilberto

SOMMARIO

1 Introduzione

1.1 Definizioni, acronimi e abbreviazioni

1.1.1 Definizioni

- Branch Coverage: tecnica adoperata durante la fase di testing, che prevede l'esecuzione di tutti i rami del programma almeno una volta durante la fase di testing.
- Failure: mancata o scorretta azione di un determinato servizio atteso.
- Fault: causa che ha generato una failure.
- Model View Control: è un metodo architetturale che prevede la divisione dell'applicazione di tre parti. Tale divisione viene effettuata per separate la rappresentazione delle informazioni interne del sistema dal meccanismo in cui le informazioni sono presentate all'utente

1.1.2 Acronimi e Abbreviazioni

- RAD: Requirement Analysis Document.
- SDD: System Design Document.
- ODD: Object Design Document.
- SOW: Statement of work.
- TP: Test plan.
- MVC: Model View Controller.
- DB: Database.
- API: Application Programming Interface.
- GUI: Graphical User Interface.

1.1.3 Riferimenti

- Object-Oriented Software Engineering (Using UML, Patterns, and Java) Third Edition. Autori: Bernd Bruegge & Allen H. Dutoit
- PMBOK Guide and Software Extention to the PMBOK Guide, Fifth Ed, Project Management Institute, 2013;
- Documentazione del progetto:
  - C10_RAD_1.0
  - C10_SDD_1.0
  - C10_SOW_1.0

Il seguente documento "RAD" è diviso in sezioni ed ha la seguente composizione:

Sezione di INTRODUZIONE:
Vi sarà presentata una breve descrizione delle esigenze da cui parte l'idea del progetto, viene quindi fornito e descritto il contesto di utilizzo del sistema per poi passare nel fornire gli obiettivi del sistema e i punti di forza/criteri di successo dell'intero progetto. Successivamente vengono citati definizioni, acronimi e abbreviazioni usati per facilitare il lettore a ricordare le parole più usate (con acronimi e abbreviazioni) e/o nel capire parole del gergo tecnico (con le corrispettive definizioni). In seguito, troverete i riferimenti utilizzati come linee guida per lo sviluppo dell'intero progetto dal punto di vista ingegneristico.

Sezione SISTEMA CORRENTE:
Questa sezione è dedicata alla spiegazione di come è la realtà attuale, prima dello sviluppo del sistema. Vengono presentati degli scenari tipici di utilizzo e vengono messi in risalto i punti critici, le difficoltà principali per valorizzare le motivazioni per cui bisogna sviluppare il sistema proposto. Infine, viene mostrato l'Activity Diagram.

Sezione SISTEMA PROPOSTO:
Questa sezione del documento parte con una sottosezione di introduzione nella quale viene fornita una descrizione dell'idea di base di come il sistema dovrebbe essere.
Si procede con la sottosezione dei Requisiti Funzionali del sistema dove vengono identificate le funzionalità che il sistema deve offrire. Infine, viene mostrato l'Activity Diagram.
È bene notare che i requisiti funzionali seguono questo tipo di convenzione:

RF[numero] nomeDelRequisitoFunzionale.

I Requisiti Non Funzionali seguono il modello FURPS+, essi sono:

1. Usabilità,
2. Affidabilità,
3. Prestazioni,
4. Supportabilità,
5. Implementazione,
6. Packaging,
7. Legali.

I Requisiti Non Funzionali, seguono questo tipo di convenzione:

nomeDelRequisitoNonFunzionale.

La composizione del documento segue con i MODELLI DI SISTEMA divisi per:

- Scenari;
- Modello dei casi d'uso dove vengono descritti i casi d'uso e l'use case diagram;
- Modello ad oggetti con una tabella degli oggetti, il Class Diagram e gli Object Diagram;
- Modelli Dinamici con lo Statechart e i Sequence Diagram;
- Mock-ups e Navigational Path.

Il documento "RAD" si conclude con il GLOSSARIO, dove sono specificati i termini utilizzati nel documento per evitare ambiguità.

2 Documenti correlati

Questo documento è correlato a tutti i documenti prodotti fino al rilascio del sistema, quindi verrà modificato in futuro dopo il rilascio di altri documenti non ancora prodotti. I test case sono basati sulle funzionalità del sistema, individuate e raccolte nel RAD.

2.1. Relazione con il documento di raccolta ed analisi dei requisiti (RAD)

La relazione tra questo documento e il RAD riguarda la fase di raccolta dei requisiti funzionali e non funzionali, infatti in questo documento oltre a essere presenti i requisiti si possono trovare anche scenari, casi d'uso, diagrammi di sequenza e mockups.

2.2. Relazione con il System Design Document (SDD)

In questo documento è presente l'architettura del sistema (MVC), la struttura dei dati e i servizi offerti da ogni sottosistema.

2.3. Relazione con l'Object Design Document (ODD)

Nell'ODD sono contenuti i package e le classi del sistema.

2.4. Relazione con lo Statement of Work (SOW)

In questo documento è contenuto un criterio di accettazione per i test case, infatti il Branch Coverage è impostato al 75%, quindi i test case verranno sviluppati in modo da soddisfare questo criterio

3 Panoramica del sistema

Come definito nel System Design Document, il sistema seguirà una pattern architetturale MVC (Model View Controller). La componente fondamentale di questo approccio è il controller che si occuperà della logica esecutiva di ogni sottosistema, nel model verranno indicate le entità che avranno un collegamento persistente nel DB, infine nella view verranno mostrate le interfacce per le diverse tipologie di utente.

3.1 Il livello view è stato decomposto in:

- GUI Admin
- GUI Cliente
- GUI Fornitore

3.2 Il livello control è stato decomposto in:

- Gestione Account
- Gestione Cliente
- Gestione Fornitore
- Gestione Carrello
- Gestione Preventivi
- Gestione Admin

3.3 Il livello model è stato decomposto in:

- DAO
- POJO

4 Funzionalità da testare/non testare

Le attività di testing previste per il sistema EasyExpo prevedono di testare il corretto funzionamento della maggior parte delle funzionalità principale del sistema.
Saranno quindi testate le seguenti funzionalità:

- Gestione Account
  -Form di login
  -Form di Registrazione Cliente
  -Form di Registrazione Fornitore
- Gestione Preventivi
  -Richiesta Preventivo
  -Compila Preventivo
  -Conferma Preventivo
- Gestione Carrello
  -Aggiunta Prodotto al Carrello
- Gestione Fornitore
  -Aggiunta Allestimento
  -Sottoscrizione Abbonamento
  -Rinnovo Abbonamento
  -Eliminazione Prodotto
  -Modifica Prodotto
- Gestione Admin

- Sospensione Cliente
- Riabilitazione Cliente
- Sospensione Fornitore
- Riabilitazione Fornitore

- Gestione Cliente
  - Ricerca Prodotto
  - Ricerca Fornitore

5 Funzionalità da non testare

Le funzionalità che non verranno testate saranno relative a tutti i requisiti funzionali di bassa e media priorità.

- Gestione Account
  - Logout
  - Modifica Profilo
  - Elimina Account
- Gestione Recensione
  - Elimina Recensione
  - Aggiungi Recensione
  - Visualizza Recensione

- Gestione Segnalazioni:
  - Inviare Segnalazioni
  - Visualizzare Segnalazioni

- Gestione Preventivi:
  - Sistema di messaggistica.

6 Approcci utilizzati

6.1 Testing di unità

Lo scopo del testing di unità è quello di testare ogni singola funzione presente all'interno del sistema. Ogni funzione rappresenta quella che viene definita come "unità".

6.1.1 Approccio scelto

L'approccio scelto è di tipo white-box, con tale tipo di approccio quindi andremo a testare il sistema conoscendone il funzionamento interno. Per questa fase verrà utilizzato il tool JUnit.

6.2 Testing di integrazione

Lo scopo del testing di integrazione è quello di "mettere insieme" le componenti testate precedentemente tramite il test di unità per vedere come funzionano una volta integrate tra loro.

6.2.1 Approccio scelto

Per effettuare il testing di integrazione si è scelto di adoperare un approccio bottom-up. Il vantaggio fondamentale di questa tipologia di testing è quello della riusabilità del codice. Questo tipo di approccio prevede però la costruzione driver per simulare l'ambiente chiamante. È stato scelto quindi questo tipo di approccio perché sembra quello più intuitivo e semplice.

6.3 Testing di sistema

Lo scopo del testing di sistema è quello di verificare che i requisiti richiesti dal cliente siano stati

effettivamente rispettati e che il cliente risulti soddisfatto del sistema stesso. In questo tipo di testing si vanno a verificare le funzionalità utilizzate più spesso da parte del cliente e quelle che risultano più "critiche".

6.3.1 Approccio scelto

Per la specifica dei test case è stato adoperata la tecnica del Weak Equivalence Class Testing.
Trattandosi di una piattaforma web il tool scelto per il testing di sistema è "Selenium" che si occuperà di simulare le interazioni con il sistema stesso come se le stesse svolgendo l'utente.

7 Criteri di pass/fail

Dopo aver individuato tutti i dati di input del sistema, quest'ultimi verranno raggruppati insieme in base alle caratteristiche in comune. Questa tecnica ci servirà per poter diminuire il numero di test da dover effettuare. Diremo che la fase di test ha successo se viene individuata effettivamente una failure all'interno del sistema, cioè l'output atteso per quel determinato input non è lo stesso previsto dall'oracolo. Successivamente la failure sarà analizzata e si passerà eventualmente alla sua correzione e verranno eseguiti nuovamente tutti i test necessari per verificare l'impatto che la modifica ha avuto sull'intero sistema. Diremo invece che il testing fallirà se l'output mostrato dal sistema coincide con quello previsto dall'oracolo.

8 Criteri di sospensione e ripresa

8.1 Criteri di sospensione

La fase di testing verrà sospesa nel momento in cui saranno raggiunti i risultati previsti in accordo con quello che è il budget a disposizione.

8.2 Criteri di ripresa

Tutte le attività di testing riprenderanno nel momento in cui verranno effettuate modifiche all'interno del sistema.

9 Test deliverables

I documenti che saranno prodotti durante questa fase sono i seguenti:

- Test Plan;
- Test Case Specification;
- Test Execution Report;
- Test Incident Report;
- Test Summary Report.

10 Materiale per effettuare il testing

Le risorse che vengono utilizzate dalle attività di testing comprendono i documenti di progetto Requirement Analysis Document, System Design Document ed Object Design Document, a partire dai quali vengono individuate le componenti da testare, rispettivamente, nel testing di sistema, nel testing di integrazione e nel testing d'unità. Per l'esecuzione di queste attività, invece, vengono utilizzati gli strumenti Selenium e JUnit su Intellij, insieme ad altri tool qualora sia necessario.

11 Attività di training per lo staff

Nessun team member attualmente conosce Selenium. Sarà, quindi, pianificata una sessione di training da parte del PM.

12 Responsabilità

Ogni team member sarà responsabile della fase di testing. Non è stata prevista una suddivisione tra sviluppatori e tester in quanto ogni team member deve svolgere tutte le mansioni.

13 Glossario

Form: termine utilizzato per riferirsi all'interfaccia di una applicazione che consente all'utente di inserire e inviare dati digitali.
GUI: è un tipo di interfaccia che consente all'utente l'interazione uomo - macchina in maniera visuale.
Testing: procedimento utilizzato per individuare le carenze di correttezza, completezza e affidabilità dei componenti software nel corso dello sviluppo.
Tool: strumento software utilizzato per ottenere un determinato risultato.