1 Introduction

All’interno del documento sono riportate le strategie di testing adottate, quali funzionalità saranno testate e gli strumenti scelti per la rilevazione degli errori, con lo scopo di presentare al cliente finale una piattaforma priva di malfunzionamenti. Sono state pianificate attività di testing per i seguenti raggruppamenti di funzionalità:

- Account
- Merchandising
- Ordine
- Cart

2 Relationship to other documents

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

Relazioni con il Requirements Analysis Document (RAD)
I test case pianificati nel Test Plan sono elaborati in relazione ai requisiti funzionali e non funzionali presentati nel RAD.

Relazioni con il System Design Document (SDD)
I test case pianificati nel Test Plan devono rispettare la suddivisione in sottosistemi presentata nell’SDD.

Relazioni con il Object Design Document (ODD)
Per ciò che concerne il test di unità e di integrazione, maggiormente legati allo ODD e alla divisione in package del sistema, essi saranno scritti e documentati unicamente all’interno del codice dell’applicativo. Per tale motivo, nel presente documento, non vi saranno riferimenti al loro design.

3 Features to be tested/not to be tested

Saranno delineati i test case con per tutte le funzionalità esposte nel RAD
Abbiamo pianificato di far test di sistema sulle seguenti funzionalità, generando i casi di test:

Account

- Autenticazione Utente
- Autenticazione Gestore
- Registrazione Utente
- Aggiunta Account Gestore
- Rimozione Account Gestore
- Aggiunta indirizzo
- Rimozione indirizzo

- Aggiunta carta di credito
- Rimozione carta di credito

Merchandising
- Aggiunta prodotto
- Rifornisci prodotto
- Rimozione prodotto
- Visualizza catalogo prodotti
- Filtra prodotti nel catalogo

Ordine
- Spedisci Ordine
- Visualizza Ordini da gestire

Cart
- Aggiunta prodotto al carrello.
- Rimozione prodotto dal carrello
- Modifica quantità prodotto nel carrello
- Effettua pagamento ordine

Per questioni relative a costi e tempi abbiamo testato il comportamento del sistema in base ai test definiti solo sulle seguenti funzionalità:

Account
- Registrazione Utente
- Aggiunta indirizzo
- Aggiunta carta di credito

Merchandising
- Aggiunta prodotto
- Rifornisci prodotto
- Rimozione prodotto
- Visualizza catalogo prodotti
- Filtra prodotti nel catalogo

Cart
- Aggiunta prodotto al carrello.
- Rimozione prodotto dal carrello.
- Modifica quantità prodotto nel carrello.

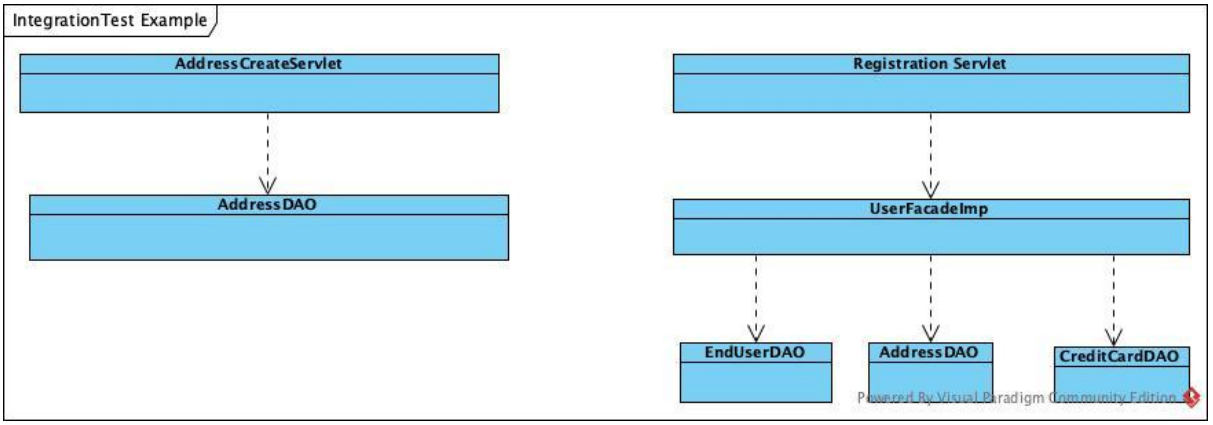
4 Pass/Fail criteria

L'attività di testing è volta all'individuazione di faults all'interno del sistema; i faults eventualmente trovati scateneranno una fase di correzione e ripetizione dei test per i quali si erano generati i faults ed eventualmente verranno ripetuti anche i test per le componenti dipendenti dalle correzioni effettuate.

5.2 Testing di integrazione

È stata impiegata un approccio di integrazione bottom-up, scelta per via della facilità con cui è possibile localizzare eventuali fault.
La definizione dei test case avverrà tramite il framework JUnit mentre verrà usato Mockito per la generazione degli stub.
Il test di integrazione sarà il medesimo per tutte le componenti da testare. Nello specifico, si procederà prima con il test delle classi DAO, successivamente col test su eventuali facade ed infine con il test delle classi Controller.

Un esempio dimostrativo:



Nel caso specifico per la funzionalità di creazione dell'indirizzo, si è proceduti nel seguente modo:

1. Test della componente "AddressDAO" in isolamento in quanto risiede nell'application logic layer (strato più in basso che siamo in grado di controllare e testare).
2. Test della componente control "AddressCreateServlet" in isolamento, utilizzando appositamente degli stub per simulare i comportamenti della componente "AddressDAO"
3. Test del control "AddressCreateServlet" integrando la componente "AddressDAO".

I casi di test generati per il punto 3 sono stati definiti utilizzando un approccio black box, in particolare category partition.
Analogamente abbiamo fatto lo stesso in caso di eventuali facade.

5.3 Testing di unità

Per il testing di unità la strategia prevista consiste nel testare ogni metodo delle classi del sistema. Da esse,sono escluse le classi bean, poiché quest'ultime presentano solo metodi getters e setters. I casi di test seguono un approccio black-box e saranno definiti e documentati direttamente nel codice, attraverso l'uso del framework per il testing di classi Java JUnit.
Le tecnologie utilizzate in tale fase saranno:

- **Mockito**: impiegato per la costruzione degli stub e l'isolamento della componente testata.
- **DBUnit**: impiegato per la fornitura di una connessione ad un database in memory equivalente (in termini di tabelle) a quello realmente utilizzato e per velocizzare e automatizzare la fase di testing sulle componenti DAO.

6 Test cases

Per osservare la generazione dei test cases, fare riferimento al documento “Test Case Specification” presente nella cartella deliverables del progetto.