Università degli studi di Salerno
Corso di Laurea in Informatica

INGEGNERIA DEL SOFTWARE

Test Plan

"UNI-AirLines"

Studenti:

Nome	***Matricola***
Santoro Mario	0512104850
Marino Raffaele	0512104508
Pastore Matteo	0512104724
Fortunato Angelo	0512104532

Anno Accademico: 2018/19

SOMMARIO

1. Introduzione

Il Test Plan del progetto UniAirlines ha l'obiettivo di pianificare le attività di testing sul sistema che si intende realizzare.
In particolare, vengono descritte le funzionalità e le componenti di UniAirlines di cui si ritiene opportuno testare il corretto funzionamento. Nel caso in cui delle attività di testing evidenziassero degli errori che possano causare comportamenti diversi da quelli attesi e che possano compromettere il buon utilizzo del sistema da parte degli utenti, quest'ultimo può essere sottoposto ad un processo di correzione degli errori individuati.

2. Relazioni con altri documenti

Il Test Plan presenta diversi punti di correlazione con i documenti stilati durante le fasi precedenti dello sviluppo di UniAirlines.
In particolare, il presente documento fa riferimento ai requisiti funzionali descritti nel Requirement Analysis Document, ai sottosistemi individuati nel System Design Document ed alle componenti del sistema illustrate nell'Object Design Document.
Ogni qualvolta si fa riferimento a qualunque di questi artefatti, si rimanda alla consultazione della relativa documentazione per ottenere informazioni più dettagliate.

3. Panoramica del sistema

Il sistema UniAirlines fornisce tutte le sue funzionalità attraverso un sito web. Per assicurarsi che ciascuna funzione si comporti come previsto, bisogna quindi assicurarsi di testare ogni funzionalità offerta da ogni schermata che compone il sito. Queste possono essere riassunte nella gestione delle varie entità considerate, tra cui voli, aerei, utenti. L'aggiunta, la modifica e l'eliminazione di questi oggetti rappresentano le principali attività da testare, a cui si va ad aggiungere, naturalmente, il processo di prenotazione di un biglietto aereo. L'obiettivo dei test da compiere, infatti, deve concentrarsi principalmente nel dimostrare che i requisiti di affidabilità descritti nelle varie fasi di sviluppo vengano garantiti. L'importanza di questi requisiti è dettata dall'esigenza di UniAirlines di permettere agli utenti di prenotare correttamente un volo.

4. Funzionalità da testare

Le attività di testing previste per il sistema UniAirlines prevedono di testare il corretto funzionamento della maggior parte delle funzionalità del sistema. Queste sono:

1. Prenotazione di un biglietto aereo.
2. Ricerca di un volo.
3. Aggiunta, eliminazione e modifica di un volo.
4. Aggiunta di un aereo.
5. Aggiunta di un aeroporto.
6. Login al lato amministrativo del sistema.

5. Criteri di successo

Un caso di test ha esito positivo se l'output osservato è differente dal risultato previsto dall'oracolo; al contrario, un caso di test ha esito negativo se l'output osservato coincide con il risultato previsto dall'oracolo. Pertanto, le attività di test hanno successo nei casi in cui riescono ad individuare dei comportamenti anomali nell'esecuzione delle funzionalità del sistema. Nel caso in cui uno o più casi di test riscuotono successo, è possibile attuare un'opportuna procedura di correzione del difetto riscontrato e, successivamente, ricorrere ad un test di regressione per testare nuovamente la funzionalità modificata ed accertarsi che il problema sia stato risolto.

6. Approccio

Le attività di testing da effettuare sul sistema si dividono in tre tipologie:
- Testing di sistema, che si occupa di testare la conformità delle funzionalità del sistema con i rispettivi requisiti funzionali e non funzionali specificati dal Requirement Analysis Document.

- Testing di integrazione, che si occupa di testare l'interoperabilità dei diversi sottosistemi, assicurandosi che il loro comportamento sia conforme a quanto specificato nel System Design Document.

- Testing d'unità, che si occupa di testare il comportamento dei singoli componenti del sistema assicurandosi che questo sia conforme alle specifiche dell'Object Design Document.

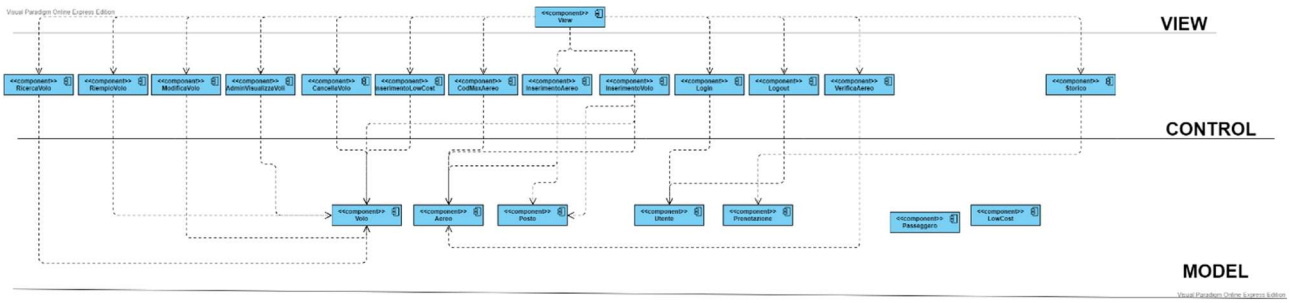
Molte di queste attività di testing si servono di opportune classi di equivalenza per ridurre tutti i possibili input ad un numero contenuto di campioni rappresentativi dei dati.

6.1. Testing di sistema

Le attività di testing di sistema vengono effettuate tramite un approccio black-box per testare la correttezza delle funzionalità definite a partire dai requisiti funzionali e dai casi d'uso esposti dal Requirement Analysis Document. Lo strumento utilizzato per implementare i diversi casi di test è Selenium, il quale fornisce un insieme di librerie Java utili a codificare i vari passaggi necessari a testare il sistema. Nel caso in cui le funzionalità da testare prevedessero degli input da parte dell'utente, è stata utilizzata la strategia del category partition, in cui delle classi di equivalenza partizionano opportunamente l'insieme dei possibili dati di input. I test case così formulati vengono illustrati nel dettaglio nel documento Test Case Specification.

6.2. Testing d'integrazione

La struttura del sistema *UniAirlines* è organizzata in modo gerarchico L'approccio stabilito per il testing d'integrazione del sistema proposto è quello del bottom-up. Questa strategia risulta quella più utile sia perché il sistema proposto è prettamente orientato agli oggetti, sia a causa dell'elevata complessità del layer più basso (contenente lo Storage), che renderebbe l'utilizzo della strategia top-down eccessivamente complicata a causa dell'elevato numero di stub da realizzare.



Le singole componenti del sistema vengono innanzitutto testate singolarmente nell'attività di testing d'unità; in questi test, le componenti si servono, quando necessario, degli opportuni stub e degli opportuni driver. Successivamente, si procede ad integrare le varie componenti seguendo la strategia bottom-up. Pertanto, esse vengono testate nell'ordine seguente:

1. Prenotazione con storico.

2. Utente con Login.
3. Utente con Logout.
4. Posto con Inserimento Volo.
5. Posto con Inserimento Aereo.
6. Aereo con VerificaAereo.
7. Aereo con InserimentoAereo.
8. Aereo con InserimentoVolo.
9. Aereo con CodMaxAereo.
10. Volo con inserimentoLowCost
11. Volo con inserimentoVolo
12. Volo con cancellaVolo
13. Volo con RiempioVolo
14. Volo con ModificaVolo
15. Volo con RicercaVolo
16. Volo con AdminVisualizzaVoli
17. Tutti i Control con View

6.3. Testing d'unità

Anche in questo caso, la metodologia scelta per effettuare i test d'unità è il category partition. I vantaggi forniti da questa tecnica consistono in:

- Esplicitare le relazioni tra le variabili da testare.

- Evitare di selezionare casi di test non utili.

- Selezionare anche casi di test in cui ci siano dei legami tra le diverse variabili.

I test d'unità vengono eseguiti seguendo la suddivisione prevista per il sistema, partendo dal livello *Model* fino ad arrivare a quello *Controller*. Nel package *Model* sono presenti varie classi, quali:

- Classe *DriverManagerConnectionPool*.

- Classi *Bean*.

- Classi *BeanDAO*.

Il testing relativo a questo package prevede di testare la classe *DriverManagerConnectionPool* per garantire la stabilità della connessione e, successivamente, le singole classi *BeanDAO*. Ciascuna di queste ultime utilizza una o più istanze delle corrispondenti classi *Bean* che, tuttavia, vengono considerate come già testate visto che sono composte semplicemente dalle proprie variabili d'istanza e dai relativi metodi *getter* e *setter* associati.

Per ogni classe *DAO*, quindi, vengono testati i seguenti metodi:

- AereoDAO:
  1. doSave
  2. avaibleAir
  3. getCodPosti
  4. maxAereo

- AeroportoDAO
  1. doSave
  2. getAeroporto

- PrenotazioneDAO:
  1. showAll

- UtenteDAO:
  1. doRetrieveByKey
  2. doSave

- VoloDAO
  1. doSave
  2. doUpdate
  3. showAll
  4. showAllLowCost
  5. doRetrieveByCod
  6. isLowCost
  7. doRetrieveByAereoportoeDataSoloAndata

Per testare le classi e i metodi elencati viene utilizzato il framework *JUnit* disponibile per l'ambiente di sviluppo *Eclipse*, già utilizzato per lo sviluppo del sistema.

7. Criteri di sospensione e di ripresa

Le attività di testing pianificate devono portarsi fino a quando tutti i test effettuati non presentino esito negativo. Nel momento in cui un test presenta un esisto positivo evidenziando un potenziale problema, si procede pianificando e mettendo in atto un'opportuna soluzione. Successivamente, una volta eliminato il difetto rilevato, si effettua un test di regressione relativo al test precedente e a quelli strettamente correlati, per assicurarsi che la soluzione adottata abbia effettivamente risolto il problema e non ne abbia causato degli altri. In ogni caso, le attività di testing possono fermarsi se si ritiene che stiano ritardando in maniera eccessiva il completamento del progetto; in questo caso, si può evitare di risolvere completamente i test che presentano una bassa priorità.

8. Materiale di 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 Eclipse, insieme ad altri tool qualora sia necessario.

9. Test case

La specifica dei test case formulati si trova nel documento *Test Case Specification*.