Corso di Ingegneria del Software

Restaurant Management
Test Report Summary
Versione 1.2

Data: 23/01/2019

Partecipanti al progetto

Nome	Matricola
Andrea Cipriano	512104874
Gianmarco Cringoli	512104778
Manuel Flora	512104628

Scritto da:	Andrea Cipriano, Gianmarco Cringoli, Manuel Flora

Revision History

Data	Versione	Descrizione	Autore
21/01/19	1.0	Stesura documento	Andrea Cipriano
22/01/19	1.1	Riempimento tabelle	Gianmarco Cringoli
22/01/19	1.1	Riempimento tabelle	Andrea Cipriano
23/01/19	1.2	Risoluzioni	Andrea Cipriano

Indice

1. Introduzione

Il presente documento fornisce un riepilogo dei risultati di tutti i processi di testing effettuati sul sistema *Restaurant Management* dopo il completamento della fase di implementazione. In particolare, vengono analizzate le attività di testing pianificate prima della realizzazione del sistema, confrontandole con quelle effettivamente eseguite e con i risultati ottenuti.

2. Riepilogo del testing

Così come descritto dal documento *Test Plan - Restaurant Management*, le attività di testing che sono state pianificate durante la fase di progettazione del sistema si dividono in tre diverse categorie: test d'unità, test d'integrazione e test di sistema. Ognuna di queste, focalizzandosi su diversi aspetti del sistema e servendosi di approcci differenti, ha l'obiettivo di verificare che il comportamento del sistema rispetti quello atteso e pianificato durante lo sviluppo. Combinando tra loro questi diversi tipi di testing, si è tentato di garantire un rilevamento degli errori che risultasse quanto più completo possibile.

2.1. Testing d'unità

La fase di testing di unità prevedeva di testare in maniera isolata le diverse componenti del sistema realizzato. Per fare questo, si è pensato di andare a testare i metodi più importanti delle diverse classi che compongono i vari sottosistemi, così come viene specificato dal Test Plan.
Le componenti del layer Model sono state tutte testate così come pianificato, dove è stato analizzato il comportamento di tutti i metodi rilevanti delle classi Bean, DAO, Manager utilizzando, quando necessario, degli stub per sostituire le componenti del livello sovrastante. Lo stesso discorso, tuttavia, non vale per i due layer successivi, ovvero Controller e View. Entrambi i livelli, infatti, sono composti da classi Servlet e Jsp, le quali dipendono fortemente dal server Tomcat sul quale fanno affidamento. Per questo motivo, non è stato possibile testare i metodi delle suddette classi così come indicato dal Test Plan.
Per implementare i casi di test realizzati, ci si è serviti delle librerie Java fornite da JUnit, le quali sono stati utili a richiamare tutti i metodi delle classi interessate e a confrontare i risultati delle loro invocazioni con ciò che i tester si aspettavano.

2.2. Testing d'integrazione

Per quanto riguarda il testing d'integrazione, anche questa attività ha subito gli stessi problemi che hanno riguardato il testing d'unità delle classi Servlet e Jsp. Per testare le diverse integrazioni tra Model e Controller e tra Controller e View, infatti, non è stato sufficiente servirsi dei driver nel primo caso e degli stub nel secondo, poiché la natura stessa del sito web implementato tramite server Tomcat non ha permesso di isolare alcune di queste componenti dalle altre. Per testare ciò che non si è potuto provare qui, ci si è affidati alla fase di testing di sistema.

2.3. Testing di sistema

Il testing di sistema è stato realizzato tenendo presente le diverse funzionalità del sistema che si è deciso di testare, così come specificato dal Test Plan. Questa attività di testing è riuscita in larga parte a coprire le lacune evidenziate in precedenza sia dal testing d'unità che dal testing d'integrazione, in quanto, andando a testare direttamente le funzionalità disponibili, sono stati verificati anche tutti i comportamenti che si desiderava testare durante le altre fasi di testing. Seppur questo non ha garantito l'isolamento delle componenti che quelle fasi di testing richiedevano, questo si è rivelato l'unico modo efficace per testare un sito web realizzato con Java Enterprise e si è riuscito comunque ad avere una buona copertura di quello che si intendeva verificare.
I vari casi di test sono stati codificati tramite le librerie Java messe a disposizione da Selenium, che hanno permesso di verificare, nei casi più comuni, la correttezza dell'esecuzione di tutte le funzionalità offerte dal sistema.

3. Risultati testing

Di seguito, vengono elencati i test effettuati durante la fase di testing di unità.

Classe	Metodo	Esito	Note
UtenteBeanDAO	doSave	Negativo	/
UtenteBeanDAO	doRetrieveByKey	Negativo	/
UtenteBeanDAO	doRetrieveByOneKey	Negativo	/
UtenteBeanDAO	doUpdate	Negativo	/
UtenteBeanDAO	doDelete	Negativo	/
MenùBeanDAO	doSave	Negativo	/
MenùBeanDAO	doRetrieveByKey	Negativo	/
MenùBeanDAO	doRetrieveByAll	Negativo	/
MenùBeanDAO	doUpdate	Negativo	/
MenùBeanDAO	doDelete	Negativo	/
PortataBeanDAO	doSave	Negativo	/
PortataBeanDAO	doRetrieveByKey	Negativo	/
PortataBeanDAO	doRetrieveByAll	Negativo	/
PortataBeanDAO	doRetrieveByCond	Negativo	/
PortataBeanDAO	getIdByNome	Negativo	/
PortataBeanDAO	doDelete	Negativo	/

PrenotazioneBeanDAO	doSave	Negativo	/
PrenotazioneBeanDAO	doDelete	Negativo	/
PrenotazioneBeanDAO	doRetrieveAllByKey	Negativo	/
PrenotazioneBeanDAO	doRetrieveAll	Negativo	/
OrdineBeanDAO	doSave	Negativo	/
OrdineBeanDAO	doRetrieveByDay	Negativo	/
OrdineBeanDAO	doRetrieveByMonth	Negativo	/
OrdineBeanDAO	doRetrieveByYear	Negativo	/
OrdineBeanDAO	doRetrieveAll	Negativo	/
UtenteManager	login	Negativo	/
UtenteManager	registrazione	Negativo	/
UtenteManager	modificaDatiPersonali	Negativo	/
MenùManager	inserimentoMenù	Negativo	/
MenùManager	modificaNomeMenù	Negativo	/
MenùManager	rimozioneMenù	Negativo	/
MenùManager	inserimentoPortata	Negativo	/
MenùManager	modificaPortata	Negativo	/
MenùManager	rimozionePortata	Negativo	/
MenùManager	getPortate	Negativo	/
MenùManager	getMenù	Negativo	/
MenùManager	getIdMenuByNome	Negativo	/
MenùManager	getPortateByMenuTipo	Negativo	/
MenùManager	getPortataByNome	Negativo	/
CamerieреManager	inserimentoCameriere	Negativo	/
CamerieреManager	modificaCameriere	Negativo	/
CamerieреManager	rimozioneCameriere	Negativo	/
ComandaManager	aggiornaListaTavoli	Negativo	/
ComandaManager	creaComanda	Negativo	/
ComandaManager	inserimentoPortataComanda	Negativo	/

ComandaManager	modificaPortataComanda	Negativo	/
ComandaManager	modificaPortataComandaStato	Negativo	/
ComandaManager	rimozionePortataComanda	Negativo	/
ComandaManager	inviaComanda	Negativo	/
ComandaManager	inserimentoOrdine	Negativo	/
ComandaManager	getOrdini	Negativo	/
ComandaManager	getOrdiniByLastDay	Negativo	/
ComandaManager	getOrdiniByLastMonth	Negativo	/
ComandaManager	getOrdiniByLastYear	Negativo	/
PrenotazioneManager	prenotaTavolo	Negativo	/
PrenotazioneManager	rimozionePrenotazione	Negativo	/
PrenotazioneManager	visualizzaPrenotazioni	Negativo	/
AttivitàManager	modificaDatiAttività	Negativo	/

Di seguito, vengono elencati i test effettuati durante la fase di testing di sistema.

Funzionalita	Esito
Login	Negativo
Registrazione	Negativo
Modifica dati personali	Negativo
Inserimento cameriere	Negativo
Modifica cameriere	Negativo
Inserimento menù	**Positivo**
Modifica nome menù	Negativo
Inserimento portata	**Positivo**
Modifica portata	Negativo
Prenota tavolo	Negativo
Inserimento numero persorne	Negativo
Inserimento portata comanda	Negativo

Modifica quantità portata comanda	Negativo
Modifica stato portata comanda	Negativo
Modifica dati attività	Negativo

4. Risoluzioni

Il testing d' unità non rilevato alcun errore, anche rieseguendo molte volte le test suite. Mentre durante l'esecuzione dei test di sistema abbiamo riscontrato errori, nel inserimento cameriere, menù e portata, siccome provavamo ad inserire la stessa entità fittizia. Nel *Incident Report - Restaurant Management* è specificato ogni fallimento.