Test Plan

PocketMuseum

Riferimento	TP_1.0
Versione	1.0
Data	13/06/2021
Destinatario	Andrea De Lucia
Presentato da	Simone Avolicino
Approvato da	

Revision History

Data	Versione	Cambiamenti	Autore
25/05/21	0.1	Stesura capitoli	Avolicino Simone
13/06/21	1.0	Revisione	Avolicino Simone

SOMMARIO

1 Introduzione

Il documento ha lo scopo di descrivere la pianificazione delle attività di testing del sistema al fine di verificare se esistono differenze tra il comportamento atteso e il comportamento reale del sistema. Sarà necessario verificare che tutte le funzionalità del sistema siano eseguite correttamente.

2 Funzionalità da testare

Le attività di testing previste per il sistema Pocket Museum prevedono di testare il corretto funzionamento della maggior parte delle funzionalità del sistema.
Per quanto riguarda il layer Model, le componenti da testare sono:

- AcquistaBean
- AcquistaModelDM
- BigliettoBean
- BigliettoModelDM
- CodeGenerator
- EventoBean
- EventoModelDM
- OperaBean
- OperaModelDM
- RecensioneBean
- RecensioneModelDM
- UtenteBean
- UtenteModelDM
- Utility

Per quanto riguarda il layer Control, le componenti da testare sono:

- AcquistaBigliettoServlet
- ControlloCodice
- ControlloDataServlet
- PageBigliettiServlet
- RecuperaBiglietti
- EventiHomepageServlet
- InserisciEventoServlet
- ModificaEventoServlet
- ModificaEventoPageServlet
- MostraEventiServlet
- AggiungiRecensione
- CercaOpera
- CercaOperaById
- GestioneOperaServlet
- GuidaPageServlet
- RecuperaImmagini
- RecuperaOpere
- RicercaRecensione
- RimuoviOpera

- LoginServlet
- LogoutServlet
- RegistrazioneServlet
- ModificaDati
- RecuperaPassword
- ControlloPassword
- UtenteDuplicatoControllo

3 Funzionalità da non testare

Per ottenere un risultato ottimale ho deciso di non lasciare alcuna funzionalità, per tanto, dovranno essere testate tutte.

4 Approcci utilizzati

4.1 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".

4.1.1.1 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.

4.2 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.

4.2.1.1 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.

4.3 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".

4.3.1.1 Approccio scelto

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.