soundrate

Test Plan

1. Introduzione

In questo documento vengono definiti gli approcci e le attività di testing da effettuare sul sistema. Vengono identificati gli elementi da dover testare o meno, insieme agli approcci e gli strumenti usati per le attività di testing. Lo scopo del testing è di rilevare errori presenti nel sistema, infatti si dice che il testing ha avuto successo se si è riusciti ad individuare fault. Questa attività permette di rilevare potenziali problemi prima della messa in atto del sistema in un ambiente di produzione, potendo quindi porne rimedio onde evitare che questi si presentino in fase di utilizzo.

2. Documenti correlati

Il presente documento è in relazione con i documenti prodotti in precedenza, ovvero il *Requirement Analysis Document* (*RAD*), il *System Design Document* (*SDD*) e l'*Object Design Document* (*ODD*).

Nel RAD vengono definiti i requisiti funzionali e non funzionali che il sistema deve supportare, che vanno convalidati nella fase di testing.

Nel SDD viene definita l'architettura del sistema, che segue un'architettura client-server multi-tier basata sul pattern *Model-View-Control* (*MVC*), e i servizi che questo deve esporre ai suoi utenti, le cui funzionalità vanno testate.

Nel ODD vengono definite le interfacce delle componenti del sistema che permettono l'interazione con il sistema sia dal suo interno che dall'esterno, rispettivamente tramite chiamate a metodi di oggetti e richieste HTTP.

3. Panoramica del sistema

Come definito nel SDD e citato in precedenza, il sistema segue un'architettura client-server multi-tier basata sul pattern *Model-View-Control* (*MVC*) che permette ai client di interfacciarsi al sistema attraverso richieste HTTP.

Le richieste HTTP dei client verso i servizi del sistema vengono ricevute su degli appositi endpoint dalle componenti *controller*, le quali convalidano ogni richiesta, dopodiché delegano l'esecuzione delle operazioni richieste alle componenti che formano il *model*, e rispondono ai client restituendo eventuali risultati o avvertendoli di potenziali errori in cui il sistema incorre, o ancora semplicemente indicando che la richiesta è andata a buon fine.

I servizi offerti richiedono l'interazione con dati persistenti; le entità della base di dati utilizzata dal sistema vengono mappate in apposite classi che permettono l'adoperazione di queste in stile *object-oriented*.

Le funzionalità del sistema vengono implementate da tre principali servizi:

- servizio implementante le funzionalità riguardanti l'utilizzo e l'accesso alle informazioni sugli utenti registrati alla piattaforma.
- servizio implementante le funzionalità riguardanti l'utilizzo e l'accesso alle informazioni sulle recensioni pubblicate sulla piattaforma.
- servizio implementante le funzionalità riguardanti l'utilizzo e l'accesso alle informazioni del catalogo della piattaforma.

Un ulteriore servizio si occupa di fornire le funzionalità riguardanti l'autenticazione, de-autenticazione, e registrazione degli utenti sul sistema. Le funzionalità amministrative sono inglobate nei servizi elencati in precedenza.

4. Funzionalità da testare

Vengono testate le principali funzionalità offerte dal sistema. Segue l'elenco delle funzionalità da testare per ogni servizio:

- **servizio autenticazione:**
  - autenticazione al sistema;
  - de-autenticazione dal sistema;
  - registrazione al sistema.
- **servizio utenti:**
  - aggiornamento della password di un utente autenticato;
  - aggiornamento dell'indirizzo e-mail di un utente autenticato.
- **servizio recensioni:**
  - pubblicazione di una recensione;
  - aggiornamento di una recensione;
  - rimozione di una recensione;
  - pubblicazione di un voto ad una recensione;
  - aggiornamento di un voto ad una recensione;
  - rimozione di un voto ad una recensione;
  - segnalazione di una recensione;
  - rimozione delle segnalazioni per una recensione.
- **servizio catalogo:**
  - inserimento di un'entrata in lista d'ascolto di un utente;
  - rimozione di un'entrata dalla lista d'ascolto di un utente.

5. Approccio

L'attività di testing di divide in tre fasi: test di *unità*, test di *integrazione*, e test di *sistema*.

5.1. Test di unità

In questa fase vengono testate in maniera isolata individualmente le componenti del sistema isolandole tra loro.

Viene utilizzato un approccio *black-box*, che si focalizza sull'input da fornire e sull'output prodotto, ignorando il comportamento interno della componente. Vengono testate le funzionalità *CRUD* esposte dalle componenti testate.

Per condurre questa fase di test viene utilizzata la libreria *JUnit* integrata al framework *Arquillian* su un'istanza dell'application server *Apache TomEE Embedded*.

5.2. Test di integrazione

In questa fase vengono testate le componenti che adoperano in combinazione le unità su cui è stato svolto il test nella fase precedente. Di conseguenza, si utilizza un approccio *bottom-up*.

Vengono testate estensivamente le funzionalità implementate sotto gli endpoint esposti dai servizi forniti dal sistema.

Anche per questa fase viene utilizzata la libreria *JUnit* integrata al framework *Arquillian* su un'istanza dell'application server *Apache TomEE Embedded*.

5.3. Test di sistema

Prima della distribuzione in produzione del sistema viene effettuata una fase di test di sistema per dimostrare che i requisiti commissionati dal cliente siano soddisfatti.

In questo tipo di test vengono testate le funzionalità di maggiore rilevanza da presentare al committente. Si effettuerà del *functional testing*.

Trattandosi di un'applicazione web, viene utilizzato lo strumento *Selenium*, che si occupa di automatizzare l'interazione con il sistema dall'esterno tramite browser web, simulando delle interazioni da parte di utenti reali. Viene utilizzata un'istanza dell'application server *Apache TomEE* per eseguire l'applicazione con base di dati in memoria.

6. Criteri di successo

I dati su cui lavorare e da fornire come input durante la fase di test vengono raggruppati in classi di equivalenza, ovvero degli insiemi in cui gli elementi vengono distribuiti in base a delle caratteristiche comuni tra gli stessi, al fine di minimizzare il numero di *test case.*

Si dice che un test ha esito positivo se questo individua una failure, cioè se l'output osservato risulta diverso da quello atteso (detto *oracolo*). Se si incorre in delle failure, queste vengono analizzate e si procede alla correzione dei fault che ne sono causa. Una volta completata la correzione, si ripete, in maniera iterativa, la fase di test, per verificare che la correzione non abbia impatto altrove nel sistema.

Al contrario, si dice che che un test ha esito negativo se l'output osservato corrisponde all'oracolo.

7. Criteri di sospensione e ripresa

La fase di testing viene interrotta quando si raggiunge un compromesso tra la qualità del prodotto e i costi dell'attività di testing. Il testing viene quindi portato avanti nel tempo per un periodo quanto più prolungato possibile, senza dover però rischiare di ritardare i termini per la consegna finale del sistema.

In seguito a eventuali alterazioni al comportamento del sistema che rischiano di introdurre potenziali fault, il sistema viene nuovamente sottoposto ai test case per assicurare una risoluzione definitiva del problema.

8. Risorse necessarie alle attività di test

Per svolgere le attività di testing è necessaria un'installazione del build automation tool *Maven 3* e una connessione ad Internet in quanto il catalogo musicale della piattaforma, su cui vengono eseguiti dei test case, viene fornito remotamente da una API esterna.

9. Schedule di test

Generalmente durante le attività di testing si preferisce che i membri che le conducono non siano coinvolti con lo sviluppo del sistema; tuttavia, dato il numero limitato di componenti del team, questo non è possibile. Le attività di testing vengono quindi svolte da membri che hanno una conoscenza approfondita del funzionamento del sistema.

Tramite le attività di testing si cerca di identificare il maggior numero possibile di fault, per poi correggerle e ri-eseguire le attività per verificare che queste siano state correttamente rimosse e che non se ne siano introdotte di nuove.

Un'attività approfondita di testing è fondamentale per lo sviluppo corretto di software, in quanto la negligenza di questa potrebbe causare il fallimento del sistema in ambito produttivo. Per tale motivo, è fondamentale organizzare le attività di testing.

9.1. Determinazione dei ruoli

Le attività di testing vengono distribuite in maniera equa tra i componenti del team. Le componenti del team parteciperanno ai test di unità, ai test di integrazione e ai test di sistema indistamente.

9.2. Organizzazione delle attività di testing

Le attività di testing vengono suddivise secondo uno schema che effettua una divisione funzionale verticale, che comprende una prima fase di test di unità, una fase di test di integrazione successiva, e infine una fase di test di sistema.
In questo modo, al termine di ogni attività, ci si assicura che le componenti che stanno alla base dei membri correntemente soggetti alle attività di testing funzionino correttamente.