Introduzione

Il testing di integrazione rappresenta una delle fasi di testing più importanti, in quanto consiste nella verifica delle interazioni tra due o più componenti.

L’obiettivo del testing consiste nella verifica della corretta interazione tra le componenti e il rispetto delle interfacce. Questo documento ha il compito di identificare la strategia di testing di integrazione per il sistema Tirocinio 2.5.

Riferimenti

Per individuare correttamente i test case si terrà conto dei documenti prodotti precedentemente. Infatti la fase di testing è strettamene legata alle fasi precedenti, in quanto saranno un punto di partenza indispensabile per poter effettuare un testing corretto ed adeguato e per verificare che il sistema desiderato sia simile a quello proposto.

**Relazione con RAD**

La relazione tra test plan e RAD riguarda in particolare i requisiti funzionali e non funzionali del sistema poiché i test che saranno eseguiti su ogni funzionalità terranno conto delle specifiche espresse nel RAD.

**Relazione con SDD**

Nel System Design Document abbiamo suddiviso il nostro sistema in sottosistemi e l’architettura in tre livelli: Presentation Layer, Application Layer e Storage Layer. Il test dei vari componenti deve rimanere fedele a queste suddivisioni il più possibile.

**Relazione con ODD**

Il test d’integrazione farà quanto più riferimento possibile alle interfacce delle classi e i package definiti nell’ODD.

Panoramica del sistema

Come stabilito nel System Design Document la struttura del nostro sistema è divisa secondo una architettura “Three Layers” cioè a tre livelli: Presentation Layer, Application Layer, Storage Layer. In questo caso il livello più alto interagisce con il livello applicativo che a sua volta si occuperà di eseguire le operazioni nel database di Tirocinio2.5, cercando di garantire il più possibile basso accoppiamento e alta coesione tra le varie classi. Il sistema inoltre è stato suddiviso in sottosistemi più piccoli, in particolare è stato diviso per gestioni.

Le funzioni previste dalle varie gestione saranno testate nel corso della fase di testing del sistema.

Funzionalità da testare e non

Nella fase di testing saranno testate le funzionalità principali per ogni gestione presente nel sistema:

* **Gestione Utenti**
  + Visualizzazione pagina personale professore e tutor aziendale
  + Login
  + Modifica dei dati personali
* **Gestione Professore\_TutorAziendale**
  + Modifica pagina personale
  + Modifica ore di tirocinio
  + Invio della risposta della domanda di tirocinio
* **Gestione Studente**
  + Aggiunta del curriculum
  + Effettuazione domanda di tirocinio interno
  + Conferma del tirocinio
  + Richiesta accettazione tirocinio effettuato nel passato
* **Gestione Segreteria**
  + Convalida dei tirocini

Le funzionalità escluse dal testing riguardano i requisiti funzionali di bassa priorità o i requisiti di cui non è necessario creare dei casi di test

Pass/Fail Criteria

Lo scopo del testing è quello di dimostrare la presenza di faults (errori) all’interno del sistema. Le attività di testing, infatti, saranno mirate all’identificazione di questi faults e ad un successivo intervento per eliminarne la presenza.

Il testing avrà successo se l’output osservato è diverso dall’output atteso: ciò significa che parliamo di SUCCESSO se il test rileva un failure.

In tal caso questa verrà analizzata e, se legata ad un fault, si procederà alla sua correzione. Sarà infine iterata la fase di testing per verificare che la modifica non abbia impattato su altri componenti del sistema.

Viceversa si parlerà di FALLIMENTO se il test non riesce ad individuare un errore.

Approccio

Per il sistema Tirocinio2.5 il testing si compone di tre fasi. Nella prima fase verranno eseguiti i test di unità dei singoli componenti, in modo da testare nello specifico la correttezza di ciascuna unità andando a constatare il corretto funzionamento di tutte le singole unità di codice. Questa fase verrà effettuata al completamento di ogni unità realizzata per poter individuare tempestivamente gli errori presenti nel codice.

Nella seconda fase verrà fatto il testing di integrazione in cui si andrà a testare l’integrazione dei vari sottosistemi. Questa fase verrà effettuata automaticamente ad ogni commit dal tool di supporto scelto per questa tipologia di testing. Nella sezione dedita al testing di integrazione verrà presentato il tool scelto.

Infine verrà eseguito il testing di sistema, che vedrà come oggetto di testing l’intero sistema assemblato nei suoi componenti. Quest’ultimo servirà soprattutto a verificare che il sistema soddisfi le richieste del committente.

Si andrà ad effettuare lo unit testing su una determinata componente solo dopo che le classi sono “stabili”, cioè sono state realizzate tutte le funzionalità per quella determinata classe.



Testing di unità

Per effettuare il testing di ogni singola componente del sistema verrà utilizzata la tecnica “Black-Box testing” attraverso il framework JUnit. In questa fase saranno analizzate le funzionalità dell’applicazione ed il comportamento delle singole componenti senza tener conto della loro struttura interna. Tali classi serviranno a definire le categorie per poter utilizzare, come testing funzionale, il Category Partition quindi per ogni classe di equivalenza verranno individuati dei vincoli a cui saranno associate delle scelte specifiche. I risultati del testing verranno analizzati e usati per correggere gli errori che causano il fallimento del sistema. Se si verifica un errore con dei risultati inattesi si interviene in maniera tempestiva sulla componente in modo da renderla correttamente funzionante e procedere con le fasi di testing successive.

Per effettuare testing in questa fase le componenti vengono isolate tramite l’utilizzo di Test Stub (implementazione completa o parziale di una componente dalla quale dipende un’altra componente sotto testing) e Test Driver (implementazione completa o parziale di una componente che fa uso di un’altra componente sotto testing) che servono a sostituire le parti mancanti del sistema.

Testing di integrazione

In seguito al test di unità di ogni componente, si procederà con il test di integrazione utilizzando Travis, un servizio di continuous integration. Il tool utilizza un approccio Big Bang per l’integrazione delle componenti.

Testing di sistema

Lo scopo principale di questa fase di testing è quello di dimostrare che l’intero sistema funzioni correttamente e che soddisfi effettivamente i requisiti funzionali e non funzionali descritti nel documento di analisi dei requisiti (RAD). È da considerare l’attività più critica, in quanto può risultare molto complesso andare alla ricerca di eventuali errori, essendo impegnati tutti i sottosistemi. Per questa fase sarà utilizzato il tool Selenium, che simula l’interazione dell’utente con il sistema.

White-box testing

Il testing “white-box” è un testing strutturale poiché utilizza la struttura interna del programma per ricavare i dati di test. In seguito al “black-box testing”, abbiamo, quindi, utilizzato questa tecnica ponendoci come obiettivo quello di raggiungere il 75% di coverage, cercando di testare le varie condizioni di uscita delle istruzioni condizionali del codice.

Sospensione e ripristino

La fase di testing occupa una parte importante nell’intera attività inerente il progetto, essa è anche molto delicata e può essere causa di imprevisti e slittamenti dei tempi a causa di errori e malfunzionamento del sistema.



Criteri di sospensione

Il testing sarà sospeso quando verrà raggiunta una branch coverage di almeno il 75%. Nel caso in cui la schedulazione del progetto fosse rispettata si cercherà di testare tutte le classi selezionate per ogni possibile input e il risultato dovrà essere quello atteso.

Nel momento in cui il testing rivelerà un errore si dovrà passare alla fase di correzione che dovrà essere eseguita da una ripetizione dell’intero processo di testing per rilevare la presenza di eventuali errori introdotti dalle correzioni stesse.

Criteri di risposta

La ripresa del test avviene soltanto quando tutti i problemi relativi alla sospensione dello stesso sono stati risolti. L’attività di testing riprenderà a partire dal test case che ha causato la sospensione.

Strumenti per il testing

Il sistema verrà testato su un single-board computer, configurato dal team stesso. Non sono richieste particolari specifiche hardware, se non quelle adatte a far funzionare il DBMS e il Server Apache Tomcat.