TirocinioSmart

ITD
Integration Test Document

TirocinioSmart

Rosanna Coccaro

Data	Versione	Cambiamenti	Autori
13/08/2020	0.1	Prima stesura	RC
28/08/2020	1.0	Revisione Finale	RC

Sommario

1. Introduzione

Una volta rilevati i bug per una singola componente e riparati, le componenti sono pronte per essere integrate in sottosistemi più grandi. Pertanto, dopo aver testato singolarmente le componenti del sistema, possiamo procedere a testarne le integrazioni.

Durante il testing d'integrazione possiamo "vestire i panni" dell'utente il quale effettivamente utilizza la nostra applicazione aspettandosi un determinato risultato come risposta ad una certa azione. Questo genere di test si occupa, quindi, di verificare non solo il corretto comportamento di ogni singolo oggetto, ma anche le relazioni con gli altri componenti dell'applicazione.

Per effettuare l'integration testing utilizziamo la strategia bottom-up: ciò ci permette di garantire la presenza di fondamenta solide alla base del sistema ma richiede di mettere in campo test driver per simulare le componenti dei layer più in alto che non sono stati ancora integrati.

2. Documenti correlati

2.1 Relazione con il System Design Document

Nel system design document abbiamo definito la suddivisione in sottosistemi relativamente al prodotto che intendiamo presentare.

Il sistema è suddiviso in tre livelli logici: presentazione, business e persistenza. Ogni livello è composto da vari sottosistemi.

In questa fase è importante focalizzare la nostra attenzione sui tre layer; infatti, pianificheremo le attività di testing relative all'integrazione delle componenti appartenenti ai sottosistemi specificati all'interno nel System Design Document relativamente ai tre layer.

2.2 Relazione con l'Object Design Document

Nel documento di Object Design sono state definite le classi che compongono il sistema e le loro mansioni. Faremo riferimento ad esse nel corso del documento per associare i test al codice prodotto.

3. Approccio al testing d'integrazione

Come già anticipato, il testing di integrazione viene eseguito seguendo la strategia "Bottom-up", infatti, sono testati per primi i sottosistemi nei layer del livello più basso per poi essere integrati nei test dei layer di livello superiore. Questo procedimento è ripetuto fino a quando non viene testato il sistema nella sua interezza.

Man mano che i test per livelli procedono, vengono definiti i test driver con l'obiettivo di simulare le componenti ai layer più alti, non ancora integrati.

Come già definito nel documento di Test Plan, le attività di testing sono prevalentemente mirate a collaudare il business layer e tutte le componenti da cui il corretto funzionamento del sistema stesso dipende.

In primo luogo, è testato il layer di persistenza: viene collaudato e verificato il funzionamento del sistema dopo l'integrazione tra DataBase e il sottosistema Data Access che ne definisce le operazioni di accesso, salvataggio e modifica.

Successivamente, il processo di testing si pone come obiettivo primario l'integrazione tra layer di persistenza e layer di business: in questo step, si procede a integrare il sottosistema Data Access con i vari sottosistemi che rappresentano i service all'interno dei quali è definita la vera e propria logica di business.

4. Componenti da testare

Ovviamente, il sistema rilasciato, nella sua prima versione è stato già testato in integrazione integralmente, pertanto, in questo paragrafo sono riportate solo le componenti da testare dopo aver effettuato le modifiche definite all'interno del documento di change request.

Specifichiamo che le componenti da testare in integrazione non derivano dall'implementazione del servizio di mailing: in questo caso, i test di integrazione già definiti e impattati da modifica sono semplicemente rieseguiti durante il testing di regressione.

Come anticipato, ci occupiamo di effettuare il testing di integrazione sul nuovo modulo aggiunto che riguarda la realizzazione dei servizi offerti dall'utilizzo di un Registro di Tirocinio elettronico. Per tale ragione, il primo step nel processo di integration testing comporta l'integrazione tra:

- Il *Database* e la classe *AttivitàRepository* (che definisce le operazioni di accesso, salvataggio e modifica di un'attività di tirocinio sul registro di tirocinio);
- Il *Database* e la classe *RegistroRepository* (che definisce le operazioni di accesso, salvataggio e modifica al Registro di Tirocinio stesso).

Il secondo step realizzato durante il testing di integrazione ha come obiettivo integrare:

- La classe *AttivitàRepository* e la classe *RegistroRepository* con la classe *RegistroTirocinioService* (che definisce i metodi di business che implementano le funzionalità a disposizione dell'utente durante l'utilizzo del Registro di Tirocinio).

Ancora, l'insieme delle componenti da testare è completato con quelle che derivano dall'introduzione di una nuova figura, il Presidente del Consiglio Didattico, all'interno del sistema. In tal caso, ricordiamo che tale utente, ha la possibilità di validare o annullare una domanda di tirocinio pervenuta al Dipartimento di Informatica. Ecco perché, in tal caso, ci occupiamo di effettuare l'integrazione tra:

- Il *Database* e la classe *PresidenteConsiglioDidatticoRepository* (che definisce le operazioni di accesso, salvataggio e modifica dei dati del Presidente del Consiglio Didattico);

Tale change request prevede, come già accennato, la definizione di funzionalità di cui il Presidente del Consiglio Didattico può usufruire, le quali sono definite all'interno del modulo DomandeTirocinio già esistente. Pertanto, il secondo step del testing di integrazione ha come obiettivo integrare:

- La classe *DomandaTirocinioRepository* e le nuove funzionalità fornite dalla classe *DomandeTirocinioService* (che definisce i metodi di business che implementano le nuove funzionalità di validazione e annullamento di una Domanda di Tirocinio da parte del Presidente del Consiglio Didattico).

5. Criteri Pass/Failed

Il test ha successo se il comportamento osservato è diverso dal comportamento specificato nei requisiti funzionali. Ciò significa che raggiungiamo gli obiettivi che ci siamo posti durante questa fase se il test individuerà dei fault nel sistema.

In tal caso analizzeremo i sottosistemi coinvolti nell'errore, ed itereremo la fase di testing per verificare che le modifiche apportate agli stessi non abbiamo avuto impatti negativi su altre componenti del sistema.

Il testing fallirà se non saranno scovati errori causati dall'integrazione delle componenti definite precedentemente.

6. Materiale per il testing

L'esecuzione dei test necessita di un server correttamente configurato su cui siano installati Java e MySQL. La configurazione deve avvenire come da manuale d'installazione.

Il testing è condotto utilizzando alcuni dei framework più famosi ed efficaci in ambienti Java: JUnit, Mockito e Hamcrest. Ad essi viene affiancata tutta la suite di test relativa a Spring, framework adottato per la realizzazione della webapp.

In particolare, per l'esecuzione dell'integration testing si fa uso di JUnit e Hamcrest. Ovviamente, l'utilizzo di Junit è legato all'esigenza di scrivere casi di test che siano ripetibili. Per quanto riguarda Hamcrest, è utilizzato poiché ci consente la scrittura di oggetti cosidetti "matchers" e di definire le regole di "match" in modo dichiarativo.

7. Test cases

Per ogni livello di integrazioni mostriamo i metodi che andremo a testare.

7.1 Integrazione tra Database e AttivitàRepository

Per l'integrazione tra il database e la classe AttivitàRepository è definita una classe ***AttivitàRepositoryIT*** che definisce i seguenti test cases:

- **findAllByStatus** – testa l'interazione con il database per il caricamento della lista di attività che si trovano in uno stato specificato;
- **findById** - testa l'interazione con il database per il singolo caricamento delle attività di tirocinio della lista tramite identificatore;
- **findAllByDomandaTirocinio** - testa l'interazione con il database per il caricamento della lista di attività di tirocinio dato l'id di una domanda;
- **findAllByRegistroTirocinioId** - testa l'interazione con il database per il caricamento della lista di attività di tirocinio dato l'id del registro di tirocinio;
- **findAllByStatusAndDomandaTirocinioId** - testa l'interazione con il database per il caricamento della lista di attività di tirocinio dato l'id di una domanda di tirocinio e lo stato specificato.

7.2 Integrazione tra Database e RegistroRepository

Per l'integrazione tra il database e la classe RegistroRepository è definita una classe ***RegistroTirocinioRepositoryIT*** che definisce i seguenti test cases:

- **findById** - testa l'interazione con il database per il singolo caricamento delle attività di tirocinio della lista tramite identificatore;

7.2 Integrazione tra Repository e RegistroTirocinioService

Per l'integrazione tra le classi AttivitàRepository e Registro Repository con la classe RegistroTirocinioService è definita una classe ***RegistroTirocinioServiceIT*** che definisce i seguenti test cases:

- **aggiungiAttivitaTirocinio** - testa il metodo che permette ad uno studente di inserire un'attività di tirocinio all'interno del registro di tirocinio
- **annullaAttivita** – testa il metodo che permette ad un delegato aziendale di invalidare un'attività di tirocinio all'interno del registro di tirocinio
- **validaAttivita** – testa il metodo che permette ad un delegato aziendale di validare un'attività di tirocinio all'interno del registro di tirocinio
- **elencaAttivita** – esta il metodo che permette di ottenere l'elenco dell'attività di tirocinio relative a un registro di tirocinio
- **chiudiRegistroTirocinio** - testa il metodo che permette ad un delegato aziendale di chiudere un registro di tirocinio
- **validaRegistroTirocinio** - testa il metodo che permette al presidente del consiglio didattico di validare una domanda di tirocinio.

7.3 Integrazione tra Database e PresidenteConsiglioDidatticoRepository

Per l'integrazione tra il Database e la classe PresidenteConsiglioDidatticoRepository è definita una classe ***PresidenteConsiglioDidatticoRepositoryIT*** che definisce i seguenti test cases:

- **findByUsernameAndPassword** – testa l'interazione con il database per il caricamento della lista di presidenti tramite username e password;
- **findaByUsername** - testa l'interazione con il database per il caricamento della lista di presidenti tramite username.

7.4 Integrazione tra DomandeRepository e DomandeTirocinioService

In tal caso, è bene specificare che l'integrazione tra la classe DomandeTirocinioRepository e DomandeTirocinioService è già stata testata nella prima versione del software attraverso la definizione di una classe di test chiamata ***DomandeTirocinioServiceIT.*** L'implementazione dell'aggiunta di una figura all'interno del sistema ha portato a definire due nuovi metodi di business all'interno della classe DomandeTirocinioService per i quali è necessario testarne l'integrazione con il database. Pertanto, alla classe *DomandeTirocinioServiceIT* sono stati aggiunti due nuovi casi di test:

- **validaDomandaTirocinio** - testa il metodo che permette al presidente del consiglio didattico di validare una domanda di tirocinio.
- **annullaDomandaTirocinio** - testa il metodo che permette al presidente del consiglio didattico di annullare una domanda di tirocinio.