ITD: Integration Test Document

IFY - Internship For You

Riferimento	
Versione	0.1
Data	15/12/2019
Destinatario	Prof.ssa F. Ferrucci
Presentato da	Roberto Calabrese, Giusy Castaldo, Geremia Cavezza, Benedetta Coccaro, Simone Civale, Carmine Ferrara, Giacomo Izzo, Alessia Natale
Approvato da	Anna Belardo, Rosanna Coccaro

Revision History

Data	Versione	Descrizione	Autori
10/12/2019	0.1	Aggiunta Capitolo 1, Capitolo 2, Capitolo 3 e Capitolo 4	[Tutti]
12/12/2019	0.1	Modifica diagramma del paragrafo Componenti da testare	[Tutti]
15/12/2019	0.1	Modifica componenti da testare	[Tutti]

Sommario

1. Introduzione

Il testing d'integrazione è utile per individuare gli errori all'interno del software e costituisce una delle fasi più rilevanti. Lo scopo del testing è quello di controllare il maggior numero di funzionalità per verificarne il corretto funzionamento, il più frequentemente possibile in modo tale da evidenziare le eventuali irregolarità.

2. Riferimenti

- IFY_RAD_Vers.1.0;
- IFY_SDD_Vers.0.3;
- IFY_ODD_Vers.0.1;
- Materiale di supporto teorico per il testing d'integrazione;
- IF_TP_Vers.0.1.

3. Test di integrazione

3.1 Approccio di Integration Testing

Per il testing di integrazione usiamo la strategia "Bottom-up", quest'ultima prevede che i sottosistemi nei layer del livello più basso vengano testati per primi 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 procederanno, utilizzeremo i test driver al fine di simulare componenti di layer più alti non ancora integrati. Come specificato, nel documento di test plan, in particolare nelle dipendenze con il livello di system design, le attività di testing dovranno essere mirate in particolar modo alla logica di business e tutte le componenti connesse al fine di garantire un corretto funzionamento del sistema stesso. Quindi procederemo in primo luogo con il testing del livello DataAccess per verificare il corretto funzionamento dell'interfaccia con la base dati. In secondo luogo, procederemo con l'integrare le entità prodotte dopo le interrogazioni alla base dati, verificandone la correttezza implementativa e la congruenza con quanto progettato in fase di Object Design. Infine, garantita la correttezza e la congruenza dei dati testati, prevediamo il testing del livello service e della logica di business in modo modulare, come progettato in fase di system design.

N.B: La tecnologia utilizzata in fase di implementazione e i design patterns individuati in fase di Object Design garantiscono una forte concentrazione di carico nella logica di business e riducono la dipendenza

con l'interfaccia di presentazione. Presupponiamo, quindi, che un'attività di testing così progettata garantisca una buona garanzia di affidabilità nel funzionamento dell'intero sistema.

3.2 Componenti da testare

La scelta delle componenti da testare segue la decisione di eseguire la strategia di testing Bottom-up. Per quanto riguarda il primo layer, le componenti da testare sono:

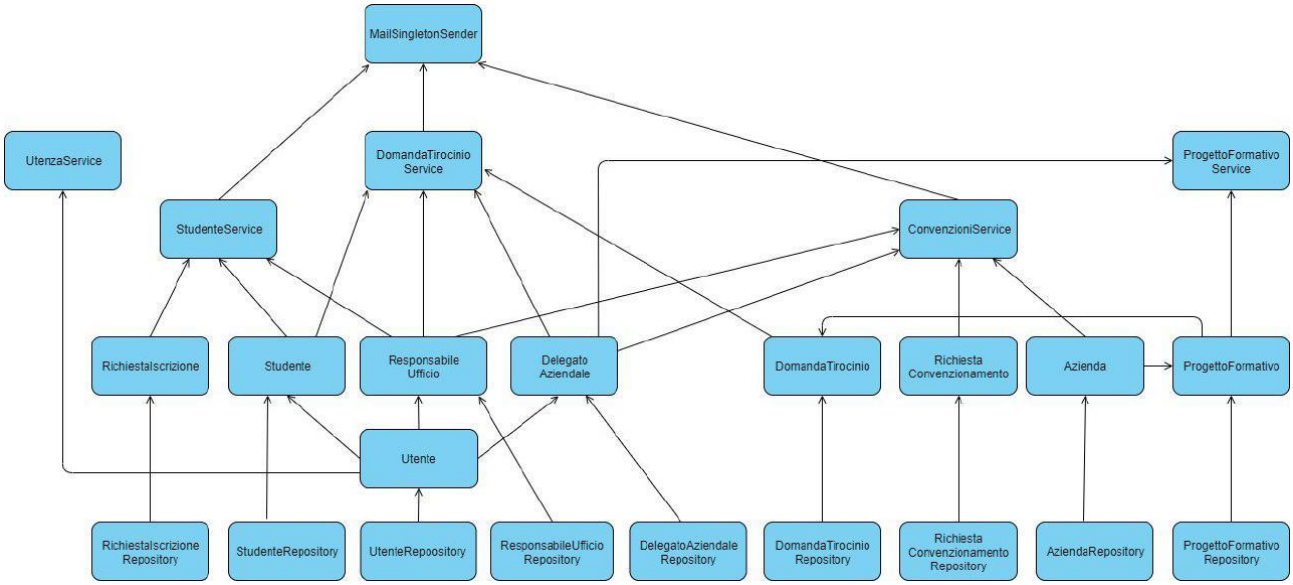
- Utente;
- Studente;
- DelegatoAziendale;
- ResponsabileUfficioTirocini;
- RichiestaConvenzionamento;
- RichiestaIscrizione;
- ProgettoFormativo;
- DomandaTirocinio;
- Azienda;
- UtenteRepository;
- StudenteRepository;
- DelegatoAziendaleRepository;
- ResponsabileUfficioTirociniRepository;
- RichiestaConvenzionamentoRepository;
- RichiestaIscrizioneRepository;
- ProgettoFormativoRepository;
- DomandaTirocinioRepository;
- AziendaRepository.

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

- UtenzaService;
- ConvenzioniService;
- DomandeDiTirocinioService;
- StudenteService;
- ProgettiFormativiService.

Per quanto riguarda il terzo layer, la componente da testare è:

- MailSingletonSender



4. Pass/fail criteri

Il testing ha successo se l'output osservato è diverso da quello atteso, questo sta a significare che parliamo di successo se il test individuerà una failure. In questo caso verrà analizzata e se è legata ad un fault si procederà alla correzione. Infine, sarà iterata la fase di testing per verificare che la modifica non abbia avuto impatto sugli altri componenti del sistema. Invece parliamo di fallimento se il test non riesce ad individuare un errore.