Hilo

Everyone wanna be a Hilo

Test Case Integration Document

Riferimento	
Versione	0.1
Data	17/01/2020
Destinatario	Prof.ssa F. Ferrucci
Presentato da	**H**ermann Senatore, **I**van Carmine Adamo, **L**orenzo Criscuolo, **O**razio Cesarano
Approvato da	

Revision History

DATA	Versione	Cambiamenti	Autori
17/01/2020	0.1	Prima Stesura	[tutti]

Sommario

1. Introduzione

Dopo aver effettuato il test di unità delle singole componenti di un livello più basso all'interno del nostro sistema, è emersa la necessità di verificare come queste componenti vadano ad integrarsi con quelle di un livello superiore.

L'attività di integration testing risponde proprio a questa necessità. Data la grandezza in termini di servizi offerti dal sistema Hilo è stato necessario adottare una strategia ben precisa e strutturata.

2. Test di Integrazione

2.1. Approccio scelto per i test di integrazione

Questo documento fornisce una panoramica sulle scelte effettuate prestando particolare attenzione al concetto di relazione di "utilizzatore-utilizzato" che intercorre tra le varie componenti del sistema.

Si è deciso di adottare un approccio **bottom-up** per questa fase di testing, dal momento che permette di effettuare prima il testing sul funzionamento delle componenti base (tipicamente, quelle afferenti al layer dati (le cosiddette Entity)) e poi, successivamente, andare a controllare come esse si integrano con le componenti di livello superiore che le utilizzano per fornire un servizio.

Questo processo si itera fino al layer di comunicazione con la view e poi alla view stessa, che sarà però interessata dal processo di System Testing (che ha come compito di accertare che il sistema in azione "sul campo" funzioni come previsto.).

2.2. Strumenti utili per i test di integrazione

Dal punto di vista tecnico, in questa fase è stato usato (oltre al consueto framework JUnit per i test di unità) un utile strumento chiamato Mockito, che permette di simulare situazioni altrimenti impossibili con la sola presenza di JUnit (come ad esempio l'arrivo di una richiesta HTTP ad un controller)

3. Componenti presenti nei test

La scelta effettuata per quanto riguarda le componenti da testare si lega a doppio filo con l'approccio bottom up che si è deciso di usare.

È sato prima necessario andare a testare le componenti del model (che si dividono a loro volta in Entity e Manager nel nostro caso), che nello specifico sono:

- Afferire
- Struttura
- HealthWorker
- Pagina
- PaginaDiarioClinico
- Patient
- Radiografia
- Statistica

Che sono usati a loro volta dai loro manager (che prendono il nome dell'entity, quindi ad esempio il Manager di Struttura prenderà il nome StrutturaManager).

I Manager delle entità permettono alle rispettive componenti del layer superiore (Control) di accedere al layer Model. È quindi necessario verificare che questi ultimi si integrino perfettamente con l'interfaccia del layer dati.

Questo discorso si applica chiaramente anche al layer Control stesso, che si occupa di gestire le richieste pervenute al sistema dall'esterno: per fornire l'accesso ai dati, si è detto che il layer Control utilizza i servizi dei Manager che li "mascherano". È quindi necessario verificare la corretta integrazione di questi ultimi col layer Control.

Nello specifico, le componenti Control da testare sono le seguenti:

- AdminController
- HealthController (per la gestione degli operatori sanitari)
- LoginController (per la gestione del login)
- PatientController
- StatisticsController (per la gestione delle statistiche)

Per testare queste componenti è quindi necessario che le componenti Entity e Manager siano già state testate con successo: di conseguenza, le componenti Controller saranno testate nel terzo step di questa fase (dopo i Manager e prima di effettuare il System Testing, dove verranno testate le view e di conseguenza l'interazione con l'utente).

4. Pass / Fail criteria

Se l'output osservato risulta essere diverso dall'output atteso, il testing ha successo. Parleremo, quindi, di **SUCCESSO** se verranno individuate delle failure. In tal caso verranno analizzate e, se legate a dei fault, si procederà con le dovute correzioni. Infine, sarà iterata la fase di testing per verificare che le modifiche apportate non abbiano avuto impatto su altri componenti del sistema. Si parlerà, invece, di **FALLIMENTO** se il test non riesce ad individuare un errore.

Piccola nota tecnica per evitare confusione: questa nomenclatura va a scontrarsi con quella utilizzata dai principali strumenti di Testing (come quelli presenti in IDE come Eclipse o Intellij IDEA) che riportano come "passed" i test che non individuano delle failures e viceversa.