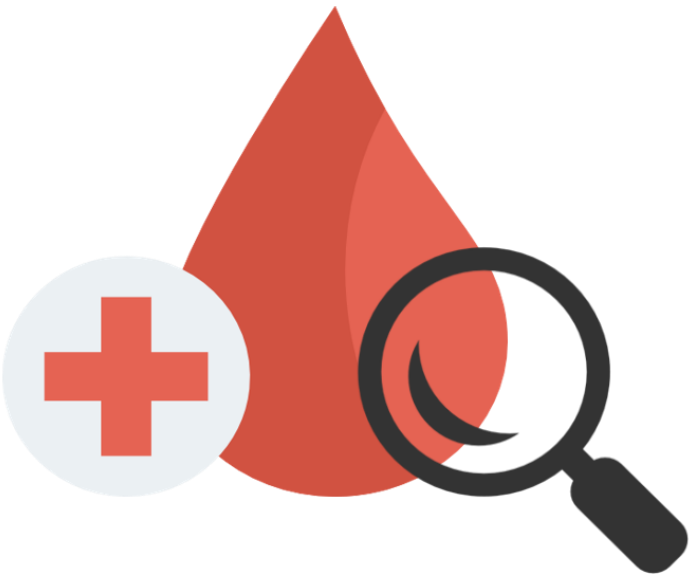


Integration Test Cases Document

Digital Donation

Riferimento	
Versione	1.0
Data	20/12/2021
Destinatario	Prof.ssa F. Ferrucci
Presentato da	Annamaria Basile, Angela De Martino, Elpidio Mazza, Kevin Pacifico, Mattia Sapere, Fabio Siepe, Marika Spagna Zito
Approvato da	Francesco Abate, Carmine Ferrara

Revision History

Data	Versione	Descrizione	Autori
20/12/21	1.0	Aggiunta Capitolo 1, Capitolo 2, Capitolo 3 e Capitolo 4	[Tutti]
20/12/21	1.0	Approvazione Documento	FA, CF

Sommario

1. Introduzione

Il testing d'integrazione è utile per individuare gli errori all'interno di un software e rappresenta una delle fasi più rilevanti. Lo scopo del testing è quello di individuare carenze di correttezza, completezza e affidabilità delle componenti software e controllare il maggior numero di funzionalità per verificarne il corretto funzionamento.

2. Riferimenti

- 2021_RAD_C9_DigitalDonation_Abate-Ferrara_Vers.1.2;
- 2021_SDD_C9_DigitalDonation_Abate-Ferrara_Vers.1.2;
- 2021_ODD_C9_DigitalDonation_Abate-Ferrara_Vers.0.1;
- 2021_TP_C9_DigitalDonation_Abate-Ferrara_Vers.1.0;
- Materiale di supporto teorico per il testing d'integrazione.

3. Test di integrazione

3.1 Approccio di Integration Testing

Per il testing di integrazione abbiamo previsto di utilizzare la strategia "Bottom-up".
Questa strategia prevede che i sottosistemi dei livelli più bassi vengano testati per primi per poi essere compresi nei test di livello più alto.
Utilizzeremo i test driver per simulare componenti di livelli più altri non ancora integrati.
le attività di testing dovranno essere mirate alla logica di Business e tutte le componenti connesse in modo da garantire un corretto funzionamento del Sistema.
Verranno testati con logica Bottom-Up le classi Repository, Entity e Service.
Come stabilito dal Test Plan si prevede di testare sia in unità sia in integrazione le classi controller solo qualora ci dovesse essere budget di tempo sufficiente. Questa scelta è ampliamente motivata dato che la logica dell'applicazione risiede principalmente nelle classi Service così come stabilito dal pattern Service Layer individuato nell'ODD.

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 livello, le componenti da testare sono:

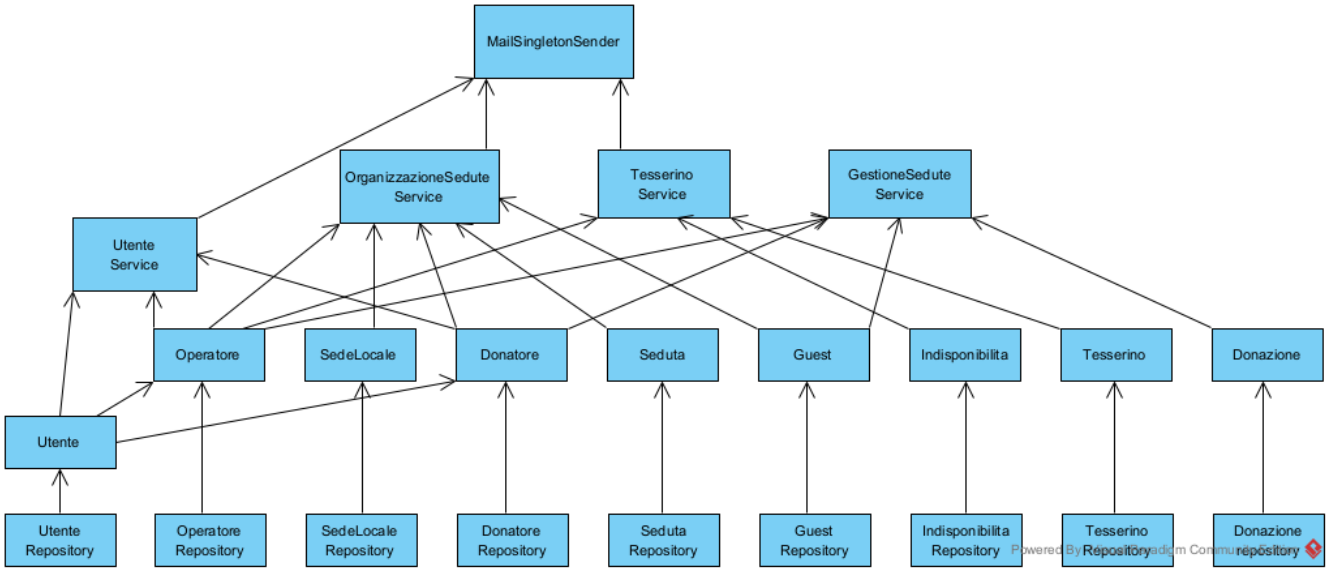
- Utente
- Operatore
- SedeLocale
- Donatore
- Seduta
- Guest
- Indisponibilità
- Tesserino
- Donazione
- UtenteRepository
- OperatoreRepository
- SedeLocaleRepository
- DonatoreRepository
- SedutaRepository
- GuestRepository
- IndisponibilitaRepository
- TesserinoRepository
- DonazioneRepository

Per quanto riguarda il secondo livello, le componenti sono:

- UtenteService
- OrganizzazioneService
- TesserinoService
- GestioneSeduteService

Per quanto riguarda il terzo livello, le componenti da testare sono:

- MailSingletonSender


MailSingletonSender
OrganizzazioneSedute Service
Tesserino Service
GestioneSedute Service
Utente Service
Operatore
SedeLocale
Donatore
Seduta
Guest
Indisponibilita
Tesserino
Donazione
Utente
Utente Repository
Operatore Repository
SedeLocale Repository
Donatore Repository
Seduta Repository
Guest Repository
Indisponibilita Repository
Tesserino Repository
Donazione repository

4. Pass/fail criteri

Il testing ha successo se l'output osservato è diverso da quello atteso, quindi un test ha successo se viene individuata una failure.
In caso di failure quest'ultima 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.