

Hotel Campus
Test Plan Document
Versione 1.1



Data: 16/12/2024

Coordinatore del progetto:

Nome	Matricola

Partecipanti:

Nome	Matricola
Luca Del Bue	0512116173
Salvatore Di Martino	0512116932

Scritto da:	Team members
--------------------	--------------

Revision History

Data	Versione	Descrizione	Autore
15/12/2024	1.0	Prima stesura	Team members
16/12/2024	1.1	Ristrutturazione documento	Team members

Indice

1.	Introduzione.....	4
2.	Relazione con altri documenti	4
3.	Feature da testare/da non testare.....	4
4.	Pass/Fail Criteria.....	4
5.	Approccio	5
6.	Sospensione e ripristino.....	5
7.	Materiale di testing	5

1. Introduzione

L'obiettivo del test è andare a verificare la correttezza e il grado di affidabilità della funzionalità di prenotazione del soggiorno e di autenticazione. Il processo di finalizzazione della prenotazione inizia con la selezione della camera d'interesse da parte del cliente autenticato, permettendo di aggiungere i servizi extra alla prenotazione. Dopo aver visualizzato un riepilogo del soggiorno da prenotare, il cliente può finalizzare la prenotazione.

2. Relazione con altri documenti

Per la corretta individuazione del test case, si fa riferimento ad altri documenti prodotti:

- Requirements Analysis Document (RAD): il testing fa riferimento al caso d'uso UC16 Finalizza Prenotazione e UC01 Login.
- System Design Document (SDD)
- Object Design Document (ODD)

3. Feature da testare/da non testare

Si effettuerà il testing della feature:

- Finalizza Prenotazione.
- Login

4. Pass/Fail Criteria

Le attività di testing mirano a rilevare errori nel sistema per consentirne la correzione. L'esito di un test è determinato da un oracolo, che rappresenta il risultato atteso basato sui requisiti. Un test ha esito positivo (pass) se l'output ottenuto differisce da quello atteso, mentre fallisce (fail) se l'output coincide con l'atteso.

Il testing sarà considerato valido se verranno testati tutti i requisiti funzionali relativi alla funzionalità trattata. Nel caso in cui una failure venga individuata e risolta, si procede con un regression testing, rieseguendo anche i test precedenti.

5. Approccio

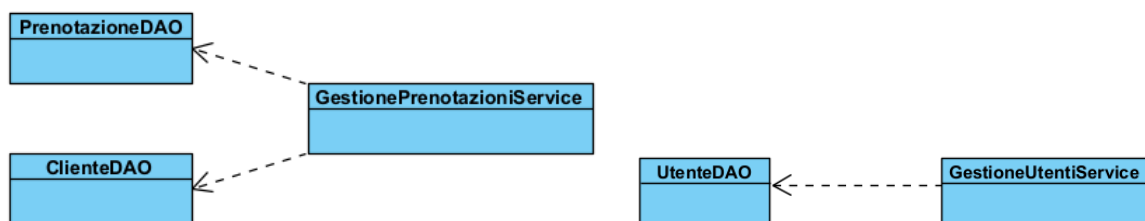
Il testing si compone di tre fasi: testing di unità, testing di integrazione e testing di sistema.

Test di unità

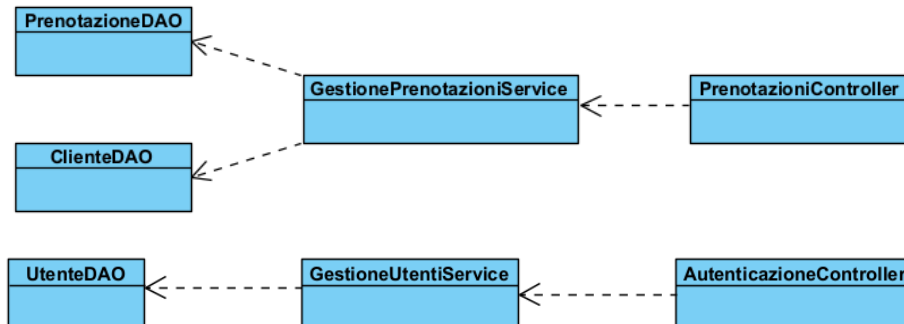
Per il test di unità utilizziamo un approccio black-box, in particolare category partition, ereditato dal test di sistema. Verrà realizzato con JUnit e Mockito per la creazione di stubs e drivers.

Test di integrazione

Il test di integrazione sfrutta una strategia Sandwich, dove il target layer è rappresentato dall'application logic layer, individuato nella decomposizione in sottosistemi (SDD). Dunque il top layer (interface) ed il bottom layer (storage) verranno testati in modo isolato utilizzando stubs e drivers. In seguito, il test di integrazione andrà a testare il target layer sfruttando il bottom layer testato precedentemente.



Infine, per integrare i tre layer, gli stub utilizzati per il testing del top layer saranno sostituiti con il resto delle componenti già testate. Mockito sarà utilizzato per il test di integrazione.



Test di sistema

Per il test di sistema sarà eseguito il functional testing, con l'obiettivo di testare la funzionalità nel suo insieme. I test frame saranno definiti con la strategia category partition e saranno adattati ai singoli componenti per eseguire anche i test di unità.

Sarà utilizzato il tool Selenium IDE, che permette di registrare le azioni che un utente può intraprendere sul browser, in modo da poter implementare ed eseguire i test case di sistema.

6. Sospensione e ripristino

Siccome il test verrà effettuato su singole funzionalità, il test verrà eseguito interamente senza sospensioni.

7. Materiale di testing

L'hardware necessario per l'attività di test prevede un semplice computer.