**WoodLot**

**Test Execution Report**

**1.0**

****

Data: 05/07/2023

|  |  |
| --- | --- |
| **Coordinatore del progetto:** | |
| **Nome e Cognome:** | **Matricola:** |
| Alessia Ture | 0512107758 |

|  |  |
| --- | --- |
| **Partecipanti:** | |
| **Nome e Cognome:** | **Matricola:** |
| Raffaella Spagnuolo | 0512107794 |
| Salvatore Di Sanza | 0512106066 |

|  |  |
| --- | --- |
| **Scritto da:** | Raffaella Spagnuolo |

|  |  |  |  |
| --- | --- | --- | --- |
| **Revisione History** | | | |
| **Data** | **Versione** | **Descrizione** | **Autore** |
| 03/03/2023 | 0.1 | Creazione e organizzazione del documento | AT, RS |
| 05/07/2023 | 1.0 | Revisione documento | RS, AT |

Sommario

[1. Introduzione 4](#_Toc128747144)

[2. Relazione con altri documenti 4](#_Toc128747145)

[**2.1** **Relazione con RAD** 4](#_Toc128747146)

[**2.2** **Relazione con SSD** 4](#_Toc128747147)

[**2.3** **Relazione con ODD** 4](#_Toc128747148)

[**2.4** **Relazione con TP** 4](#_Toc128747149)

[3. Approccio 4](#_Toc128747150)

[**3.1** **Unit Testing** 5](#_Toc128747151)

[**3.2** **Integration Testing** 5](#_Toc128747152)

[**3.3** **System Testing** 5](#_Toc128747153)

[4. Test Execution 5](#_Toc128747154)

[**4.1** **Testing di unità e di integrazione** 5](#_Toc128747155)

[**4.2** **Testing di sistema** 5](#_Toc128747156)

[**4.2.1** **Gestione Ospite** 6](#_Toc128747157)

[**4.2.2** **Gestione Utente Registrato** 6](#_Toc128747158)

[**4.2.3** **Gestione Contadino** 6](#_Toc128747159)

[**4.2.4** **Gestione Catalogo** 6](#_Toc128747160)

[**4.2.5** **Gestione Ordine** 6](#_Toc128747161)

[**4.2.6** **Gestione Carrello** 6](#_Toc128747162)

# Introduzione

Il testing di unità rappresenta una delle fasi di test più importanti, in quanto consiste nella verifica e il collaudo di singole unità software di un sistema. Tuttavia, queta fase è anche molto delicata e può essere causa di imprevisti e slittamenti dei tempi a causa di errori e malfunzionamenti del sistema. Il testing di integrazione è utile per individuare gli errori all’interno di un software e rappresenta una delle fasi più rilevanti. Lo scopo di questo 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.

Questo documento ha lo scopo di riportare i risultati dell’esecuzione dei test di unità e di integrazione delle varie componenti del sistema software web “WoodLot”. In particolare, sono state testate le classi service e controller con l’utilizzo di JUnit e Mockito.

Il documento ha l’obiettivo di raccogliere le informazioni principali risultanti dall’esecuzione del testing di unità, integrazione e sistema. In particolare, vengono raccolti i dati relativi ai test falliti ed eseguiti con successo, la branch coverage e la line coverage.

# Relazione con altri documenti

Per verificare il corretto funzionamento del sistema l’individuazione delle componenti testate fa riferimento ai documenti prodotti nelle fasi precedenti.

## **Relazione con RAD**

La relazione riguarda in particolare i requisiti funzionali e non funzionali del sistema, poiché i test saranno eseguiti su alcune delle funzionalità.

## **Relazione con SSD**

Nel System Design Document abbiamo suddiviso il nostro sistema in sottosistemi e definito l’architettura. Il test dei vari componenti deve rimanere fedele a queste suddivisioni il più possibile.

## **Relazione con ODD**

Il test di integrazione farà quanto più riferimento possibile alle interfacce delle classi e ai package definiti nell’ODD.

## **Relazione con TP**

Il documento fa riferimento alle attività di testing specificate nel Test Plan.

# Approccio

## **Unit Testing**

Il testing si compone di tre fasi. Inizialmente, verranno eseguiti i test di unità dei singoli componenti, in modo da testare nello specifico la correttezza di ciascuna unità, andando a constatare il corretto funzionamento di tutte le singole unità di codice. Nel corso del progetto, i testi di unità e di integrazione sono stati scritti in due classi distinte: le prime terminano con “UT”, mentre le seconde terminano con “IT”.

## **Integration Testing**

Per il test 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 alti non ancora integrati. Le attività di testing dovranno essere mirate alla logica di business e tutte le componenti connesse, in modo da garantire il corretto funzionamento del sistema. Come stabilito dal Test Plan si prevede di testare sia in unità sia in integrazione le classi Controller e Service. Questa scelta è motivata dal fatto che la logica dell’applicazione risiede principalmente nelle classi service, così come stabilito dal pattern Service Layer individuato nell’ODD.

## **System Testing**

Per effettuare il testing di sistema è stato creato il corrispondente di ogni caso di test utilizzando il tool Selenium IDE. Nello specifico, si è generata una Test Suite per ogni sottosistema. Per i casi di test che prevedono inserimento di un file sono stati allegati i file da utilizzare durante il testing.

# Test Execution

## **Testing di unità e di integrazione**

L’esecuzione dei test di unità non ha rivelato fault nelle componenti del sistema.

|  |  |
| --- | --- |
| Branch Coverage | Line Coverage |
| 80% | 90% |

Nella repository è disponibile il report generato dal tool JaCoCo contenete tutte le informazioni circa la coverage del sistema.

<https://github.com/raffaellaspagnuolo/WoodLot/tree/main/JaCoCo-Coverage-Report>

## **Testing di sistema**

Di seguito i risultati del test di sistema eseguito con il tool Selenium IDE: **il test ha data esito positivo**.

Nella repository è disponibile il resoconto del testing effettuato con il tool Selenium IDE:

<https://github.com/raffaellaspagnuolo/WoodLot/tree/main/systemTestingResources>