Università degli Studi di Salerno
Corso di Ingegneria del Software

HomeDecore
Test Execution Report
Versione 1.0

Data: 14/02/2025

Coordinatore del progetto:

Nome	Matricola

Partecipanti:

Nome	Matricola
Alfieri Riccardo	0512116533
Cammarota Lucageneroso	0512116941
Fasolino Pietro	0512116473
Marino Michele Graziano	0512117109

Scritto da:	Alfieri Riccardo

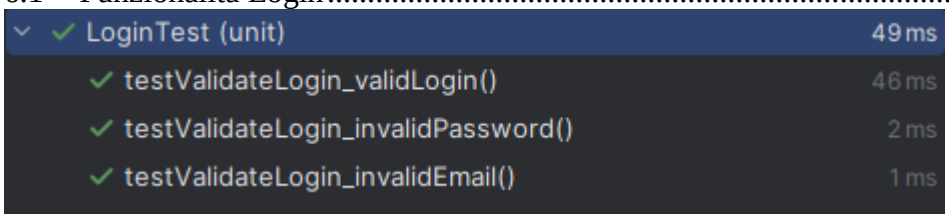
Revision History

Data	Versione	Descrizione	Autore
14/02/2025	1.0	Stestura del documento	Alfieri Riccardo

Indice

Sommario

1. INTRODUZIONE 5
2. RELAZIONI CON ALTRI DOCUMENTI 5
3. FUNZIONALITA' DA TESTARE 5
4. APPROCCIO 5
5. STRUMENTI PER IL TESTING 5
6. UNIT TESTING 6
   6.1 Funzionalità Login 6



..... 6

```
@Test  Riccardoalfieri2003 *
public void testValidateLogin_validLogin() {
    boolean isValid = loginServlet.validateLogin( email: "user@example.com",  password: "correctPassword123");
    assertTrue(isValid);
}
```

..... 6

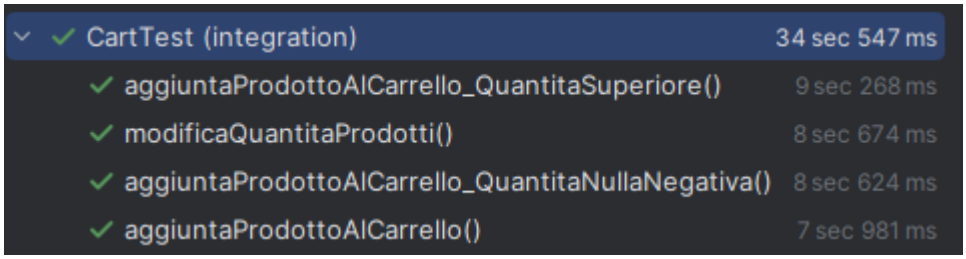
```
@Test  Riccardoalfieri2003 *
public void testValidateLogin_invalidEmail() {
    boolean isValid = loginServlet.validateLogin( email: "user@example",  password: "correctPassword123");
    assertFalse(isValid); // Verifica che un'email non valida renda il login invalido
}
```

..... 6

```
@Test  Riccardoalfieri2003 *
public void testValidateLogin_invalidPassword() {
    boolean isValid = loginServlet.validateLogin( email: "user@example.com",  password: "wrongPassword");
    assertFalse(isValid); // Verifica che una password errata renda il login invalido
}
```

..... 6

   6.2 Funzionalità Registrazione 7
   6.3 Funzionalità Magazziniere 9
7. INTEGRATION TESTING 12
   7.1 Login Test 12
   7.2 Cart Test 12



..... 12

8. RISULTATI FINALI 12
9.VALUTAZIONE FINALE 12

1. INTRODUZIONE

L'obiettivo del documento di Test Execution Report è riportare i risultati dei casi di test stabiliti nel Test Plan, per assicurarci chele funzionalità del sistema siano corrette.

2. RELAZIONI CON ALTRI DOCUMENTI

RAD: si farà riferimento ai requisiti funzionali e non funzionali presenti nel RAD
SDD: si farà riferimento alla struttura del sistema descritta nell'SDD
ODD: si farà riferimento a come la struttura è stata implementata nel sistema descritta nell'ODD

3. FUNZIONALITA' DA TESTARE

Funzionalità dell'utente:

- Validazione di Email nella registrazione
- Validazione di password nella registrazione
- Aggiunta e rimozione di prodotti dal carrello
- Login tramite mail e password

Funzionalità del magazziniere:

- Richiesta di un prodotto
- Rimozione di un prodotto dal catalogo
- Modifica di un prodotto tramite criteri

4. APPROCCIO

L'approccio utilizzato per il testing è di tipo bottom-up tramite tecniche di black-box.
Verranno prima effettuati i test di unità, successivamente test di integrazione e di sistema.

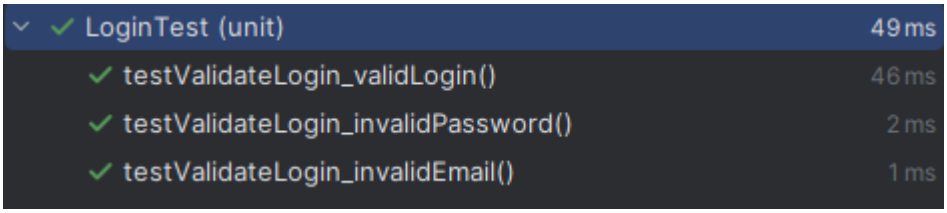
5. STRUMENTI PER IL TESTING

Gli strumenti utilizzati per il testing sono:
- Testing d'unita: JUnit e Mockito
- Test d'integrazione: SeleniumIDE

6. UNIT TESTING

6.1 Funzionalità Login

In questa sezione utilizziamo un’e-mail e password fittizia per simulare il login sul sistema.




```java
@Test  Riccardoalfieri2003 *
public void testValidateLogin_validLogin() {
    boolean isValid = loginServlet.validateLogin( email: "user@example.com",  password: "correctPassword123");
    assertTrue(isValid);
}
```

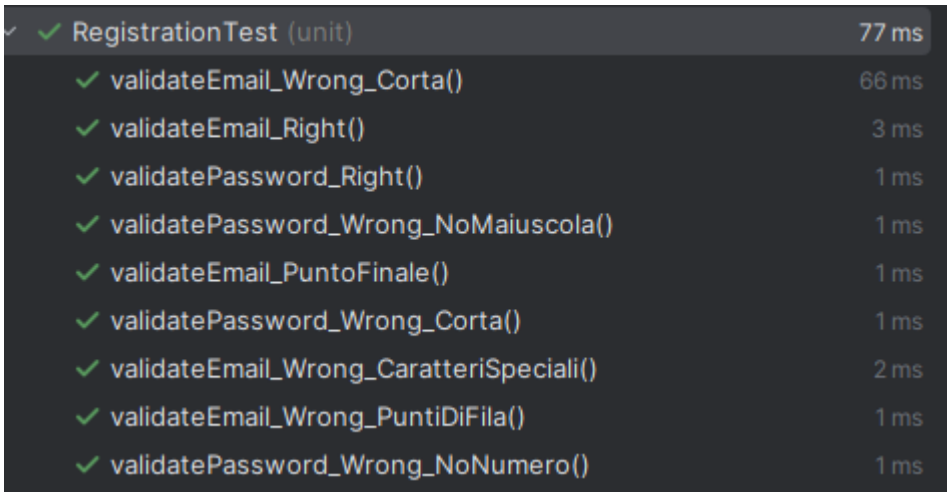
```java
@Test  Riccardoalfieri2003 *
public void testValidateLogin_invalidEmail() {
    boolean isValid = loginServlet.validateLogin( email: "user@example",  password: "correctPassword123");
    assertFalse(isValid); // Verifica che un'email non valida renda il login invalido
}
```

```java
@Test  Riccardoalfieri2003 *
public void testValidateLogin_invalidPassword() {
    boolean isValid = loginServlet.validateLogin( email: "user@example.com",  password: "wrongPassword");
    assertFalse(isValid); // Verifica che una password errata renda il login invalido
}
```

6.2 Funzionalità Registrazione

La validazione delle mail viene effettuata tramite un'espressione regolare, che indica come la mail deve essere strutturata

La validazione delle password viene effettuata tramite un'espressione regolare, che indica come la password deve essere strutturata



Test Validazione Email

```
@Test  Riccardoalfieri2003 *
public void validateEmail_Right() {
    boolean isValid = registerServlet.validateEmail("ric.alfa.2003@gmail.com");
    assertTrue(isValid);
}
@Test  new *
public void validateEmail_PuntoFinale() {
    boolean isValid = registerServlet.validateEmail("ric.alfa.2003@gmail.com.");
    assertFalse(isValid);
}
@Test  Riccardoalfieri2003 *
public void validateEmail_Wrong_CaratteriSpeciali() {
    boolean isValid = registerServlet.validateEmail("ric.al,fa.2003@gmail.com");
    assertFalse(isValid);
}
```

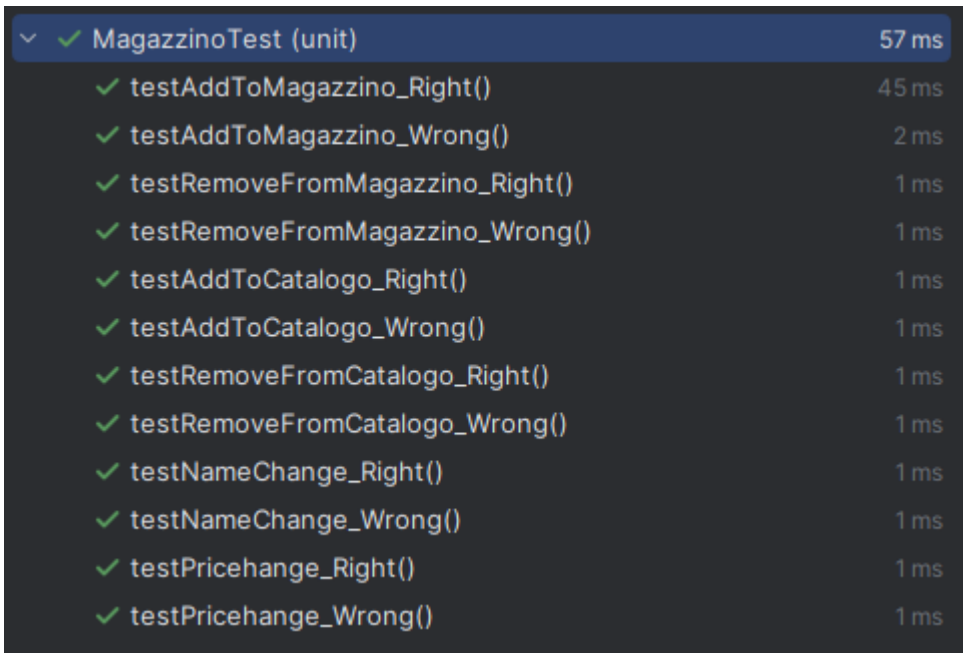
```java
@Test  Riccardoalfieri2003 *
public void validateEmail_Wrong_Corta() {
    boolean isValid = registerServlet.validateEmail("r@g.a");
    assertFalse(isValid);
}
@Test  Riccardoalfieri2003 *
public void validateEmail_Wrong_PuntiDiFila() {
    boolean isValid = registerServlet.validateEmail("ric..alfa.2003@gmail.com");
    assertFalse(isValid);
}
```

Test Validazione Password

```java
@Test  Riccardoalfieri2003
public void validatePassword_Right() {
    boolean isValid = registerServlet.validatePassword("CiaoComeStai2000");
    assertTrue(isValid); // Verifica che il login sia valido
}
@Test  Riccardoalfieri2003 *
public void validatePassword_Wrong_NoMaiuscola() {
    boolean isValid = registerServlet.validatePassword("ciaocomestai200");
    assertFalse(isValid);
}
@Test  Riccardoalfieri2003
public void validatePassword_Wrong_NoNumero() {
    boolean isValid = registerServlet.validatePassword("CiaoComeStai");
    assertFalse(isValid);
}
@Test  Riccardoalfieri2003
public void validatePassword_Wrong_Corta() {
    boolean isValid = registerServlet.validatePassword("C2");
    assertFalse(isValid);
}
```

6.3 Funzionalità Magazziniere

Le modifiche che può apportare il magazziniere sono sul catalogo e sui prodotti in magazzino.
È stato effettato anche il testing sull'aggiunta e rimozione di prodotti dal magazzino per testare il comportamento dell'accettazione delle richieste di prodotti dei fornitori



Cambio Nome

```
// Nome non nullo
@Test public void testNameChange_Right() {  Riccardoalfieri2003
    boolean isValid= catalogo.validateNameChange( nome: "Nuovo prodotto test");
    assertTrue(isValid);
}
```

```
// Nome nullo
@Test public void testNameChange_Wrong() {  Riccardoalfieri2003
    boolean isValid= catalogo.validateNameChange( nome: "");
    assertFalse(isValid);
}
```

Cambio Prezzo

```java
// Prezzo >=0.99
@Test public void testPricehange_Right() {  Riccardoalfieri2003
    boolean isValid= catalogo.validatePriceChange(2.0);
    assertTrue(isValid);
}
```

```java
// Prezzo <=0
@Test public void testPricehange_Wrong() {  Riccardoalfieri2003
    boolean isValid= catalogo.validatePriceChange(-2.0);
    assertFalse(isValid);
}
```

Test addToMagazzino

```java
/* Test su addtoMagazzino  */

//Supponiamo che il prodotto sia in magazzino
@Test public void testAddToMagazzino_Wrong() {  Riccardoalfieri2003
    boolean isValid= catalogo.validateAddToMagazzino(prodotto);
    assertFalse(isValid);
}

//Supponiamo che il prodotto non sia in magazzino
@Test public void testAddToMagazzino_Right() {  Riccardoalfieri2003
    prodotto.setInMagazzino(false); //eliminiamo il prodotto dal magazzino per comodità
    boolean isValid= catalogo.validateAddToMagazzino(prodotto);
    assertTrue(isValid);
}
```

Test addToCatalogo

```java
/* Test su addtoCatalogo  */

//Supponiamo che il prodotto sia in catalogo
@Test public void testAddToCatalogo_Wrong() {  Riccardoalfieri2003
    boolean isValid= catalogo.validateAddToCatalogo(prodotto);
    assertFalse(isValid);
}

//Supponiamo che il prodotto non sia in catalogo
@Test public void testAddToCatalogo_Right() {  Riccardoalfieri2003 *
    prodotto.setInCatalogo(false); //eliminiamo il prodotto dal catalogo per comodità
    boolean isValid= catalogo.validateAddToCatalogo(prodotto);
    assertTrue(isValid);
}
```

Test RemoveFromMagazzino

```java
/* Test su RemoveFromMagazzino  */

//Supponiamo che il prodotto sia in catalogo
@Test public void testRemoveFromMagazzino_Right() {  Riccardoalfieri2003
    boolean isValid= catalogo.validateRemoveFromCatalogo(prodotto);
    assertTrue(isValid);
}

//Supponiamo che il prodotto non sia in catalogo
@Test public void testRemoveFromMagazzino_Wrong() {  Riccardoalfieri2003
    prodotto.setInMagazzino(false); //eliminiamo il prodotto dal magazzino per comodità
    boolean isValid= catalogo.validateRemoveFromMagazzino(prodotto);
    assertFalse(isValid);
}
```

Test RemoveFromCatalogo

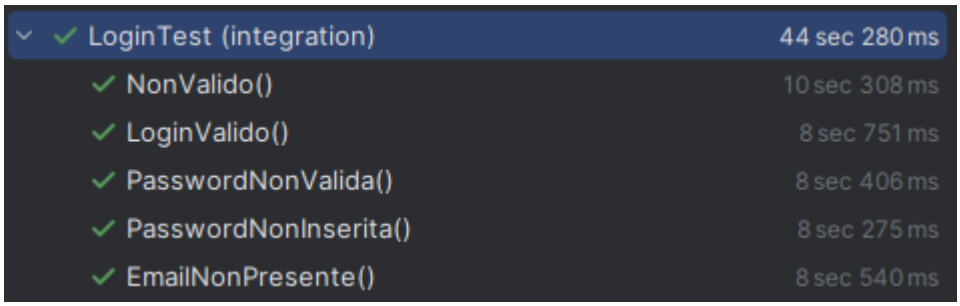
```java
/* Test su RemoveFromCatalogo  */

//Supponiamo che il prodotto sia in catalogo
@Test public void testRemoveFromCatalogo_Right() {  Riccardoalfieri2003
    boolean isValid= catalogo.validateRemoveFromCatalogo(prodotto);
    assertTrue(isValid);
}

//Supponiamo che il prodotto non sia in catalogo
@Test public void testRemoveFromCatalogo_Wrong() {  Riccardoalfieri2003
    prodotto.setInCatalogo(false); //eliminiamo il prodotto dal magazzino per comodità
    boolean isValid= catalogo.validateRemoveFromCatalogo(prodotto);
    assertFalse(isValid);
}
```

7. INTEGRATION TESTING

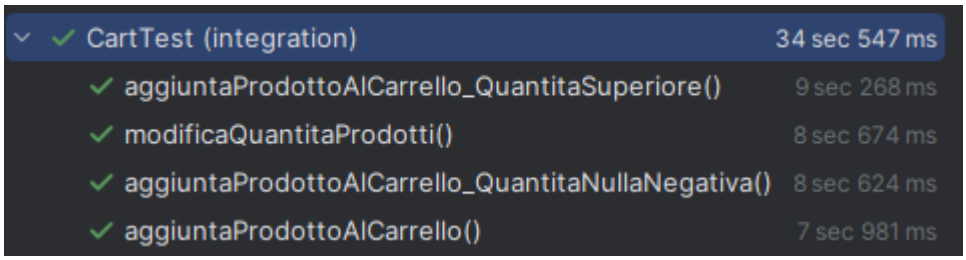
Il testing d'integrazione è effettuato tramite SeleniumIDE, che consente di automatizzare le istruzioni da eseguire sul sistema.

Il metodo di valutazione è l'esecuzione stessa per vedere i risultati ottenuti.

7.1 Login Test



7.2 Cart Test



8. RISULTATI FINALI

Tipo test	Test superati	Test falliti	Test totali
Unità	24	0	24
Integrazione	9	0	9

9.VALUTAZIONE FINALE

Tutti i test, sia d'unità che d'integrazione, restituiscono risultati positivi, senza alcun fallimento. Questo equivale a dire che il sistema offre funzionalità corrette, gestendo errori che porterebbero a fault.