
Laurea Triennale in informatica-Università di Salerno
Corso di *Ingegneria del Software- Prof. C.Gravino*

# Testing Plan
# Progetto
# STORYTELLING

# Revision History

| Data | Versione | Descrizione | Autori |
|---|---|---|---|
| 03/12/2021 | 0.1 | Prima stesura | Alessandro Marigliano, Emmanuele Virginio Coppola |
| 04/12/2021 | 0.2 | Aggiunta di Test Cases | Alessandro Marigliano, Emmanuele Virginio Coppola |

# Sommario

# 1 Introduzione

StoryTellling è un software che si impegna nel rendere l'espressione di idee tra i sui utenti in maniera semplice ed efficace.

Il documento di Test Plan ha lo scopo di mettere in chiaro l'organizzazione l'attività di testing per assicurasi che le funzinalità di sistema agiscano nella maniera corretta.

All'interno del documento sono documentate le strategia e gli strumenti impiegati per il testing per assicurarsi che il sistema risulti privo di errori.

Le Attività di testing descritte in questo documento sono dirette verso le seguenti gestioni:

- Gestione Utente
- Gestione Bacheca

# 2 Relazione con altri documenti

Per la corretta individuazione dei Test Case si faccia riferimento agli altri documenti prodotti:

## Relazioni con il Requraments Analisis Document(RAD)

I Test Case pianificati nel Test Plan sono pianificati in base ai requisiti funzionali e non funzionali descritti nel RAD.

## Relazioni con il System Design Document(SDD)

I Test Case Pianificati devono Rispettare la Suddivisione dei sottosistemi descritti all'interno del SDD.

## Relazioni con il Object Design Document(ODD)

Il documento di object design document non è stato ancora formalizzato al tempo di scrittura di qusto documento

# 3 Panoramica Di Sistema

Il Sistema proposto basa la sua architettura sul modello Model View Control.

Verranno usati HTML5, CSS3, AJAX e Javascript per la parte front-end e la generazione delle view.

Per la logica applicativa e quindi il back-end sarà utilizzato Java nella sua versione OpenJdk 15.

Per la gestione del database verrano usati :

- Java Database Connector per il collegamento al database.
- MariaDB come database in fase di produzione e deployment.

# 4 Feature da Testare

Il testing verrà effettuato per le varie gestioni:

- Gestione Account
  - Registrazione Account
  - Login
  - Eliminazione Account
  - Logout
- Gestione Bacheca
  - Inserimento Commento
  - Pubblicazione Storia

Le funzionalità di cui non si andrà a fare il testing sono funzionalità che non prevedono l'input manuale di dati da parte dell'utente come la visualizzazione di dati.

# 5 Criteri di Fallimento/Successo

L'esito di un test case è valutato mediante un'oracolo , inteso come risultato della sua esecuzione in base ai sui requisiti.

Un test ha successo se, dato un input al sistema , l'output ottenuto è diverso dall'output previsto dall'oracolo.

Un test ha fallisce se , dato un input al sistema , l'output ottenuto è uguale all'output atteso dall'oracolo.

Tutto il testing sarà considerato valido se tutti i seguenti vincoli saranno rispettati:

- Testare tutti i requisiti funzionali ad alta priorità;
- Effettuare test di regressione ogni volta che si introducono nuove caratteristiche al sistema o vengono modificate quelle presenti;
- Raggiungere un branch coverage non inferiore al 75%

# 6 Approccio

Il testing dell'intero sistema si compone di 3 fasi : testing di sistema , testing di integrazione e testing di unità. Verrano progettati nell'ordine appena definito e testati nel medesimo ordine.

Prima della fase di implementazione di sistema , avverrà la progettazione dei casi di test di sistema , perfezionati in seguito nella loro fase di esecuzione
; durante la fase implementativa averrò la progettazione dei casi di test di unità.

Durante lo sviluppo saranno eseguite periodiche attività di revisione dul codice del prodotto.

Poichè la progettazione è organizzata seguendo un modello simile al modello a V , il testing di sistema è stato pianificato in seguito alla stesura del documento Requirament Analysis Document, mentre la pianificazione del testing di integrazione avverrà dopo la stesura del System Design Document.

## Testing di Sistema

Per il testing di sistema sarà utilizzato il tool Selenium IDE , che permette di registrare le azioni che un utente può intrapendere sul browser , in modo da poter implementare ed eseguire i test case di sistema. Il server per la fase di testing verrà effettuato il deploy su macchina locale identificata con localhost.

## Performance Testing

A causa del basso budget a disposizione , non si assicura l'esecuzione del performance testing.

## Pilot Testing

A causa del basso budget a disposizione , non si assicura l'esecuzione del pilot testing.

## Acceptance Testing

L'acceptance testing verrà effettuato solo sul functional testing , e il Professore Gravino simulerà la figura del cliente.

## Installation Testing

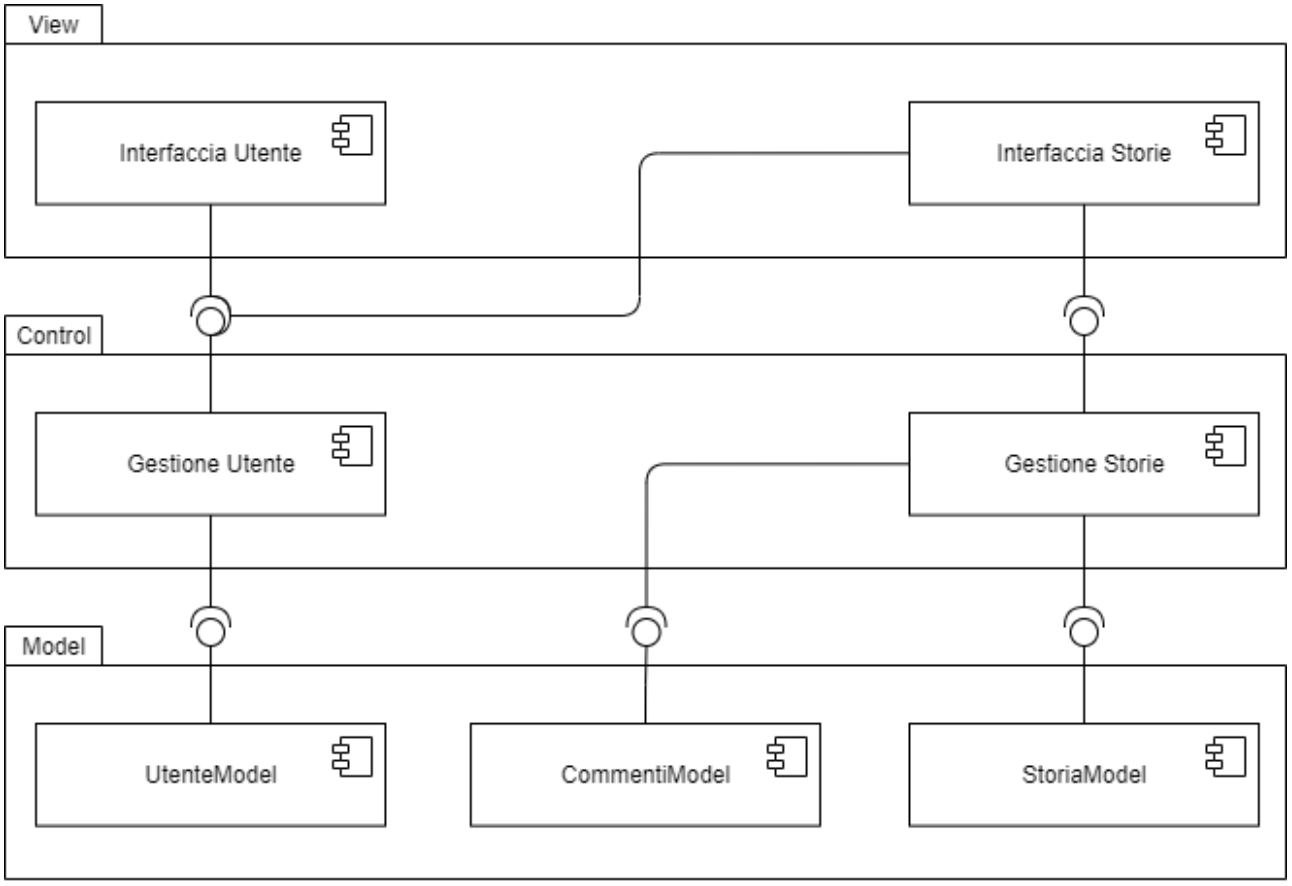
A causa del basso budget a disposizione , non si assicura l'esecuzione dell'installation testing.

## Testing di Integrazione

Verrà utilizzato un approccio bottom-up, metodo ritenuto più adatto per un software basato sul paradigma Object Oriented

Il test di integrazione sarà il medesimo per tutte le componenti da testare. Nello specifico , si procederà prima con il test delle classi appartenenti al package Model, poi dei delle classi appartenetnti al package Control.

Per quanto riguarda il test ecco qui presente uno schema che rappresenta le dipendenze tra i sistemi:




## Testing di Unità

Per il testing di unità la strategia prevista consiste nel testare ogni metodo delle classi del sistema. Da esse, sono escluse le interfacce e le classi entity, poiché quest'ultime presentano solo metodi getters e setters. I casi di test saranno definiti attraverso un approccio black-box e saranno documentati direttamente nel codice, attraverso l'uso del framework per il testing di classi Java JUnit. Per ogni Production Class sarà definita una Test Class che rispetterà il formato NomeProductionClassTest. Tali classi saranno scritte in parallelo alle Production class, per garantire una più facile copertura del codice. Le stesse classi saranno poi revisionate e modificate da sviluppatori differenti. Altre tecnologie usate in tale fase saranno:

- Mockito: per la costruzione degli stub e l'isolamento della componente testata.
- JaCoCo: per il calcolo di metriche tra le quali la Branch Coverage.
- Maven: per la build e l'esecuzione automatica dei tests

# 7 Sospensione e Ripristino

In questa sezione verrano specificati i criteri di sopsenzione del test e le attività di test che dovranno essere ripetute quando si riprende il test.

### Criteri di sospensione

Il testing non verrà sospeso fino alla sua terminazione , anche in caso di rivelazione di una failure. Il testing potrà essere momentaneamente sospeso nel caso venga restituito , al momento di esecuzione , un errore nella definizione di uno dei test stessi.

### Criteri di ripristino

Il testing verrà ripreso In caso di risoluzione di fault precedentementi individuati.

# 8 Materiale di Testing

L'hardware necessario per l'attività di test è un computer per cui non è necessario il collegamento a internet, in quanto il sistema non è stato ancora rilasciato al pubblico.

# 9 Test Cases

## TC_1.1 Registrazione account

| **Parametro: Username**<br>**Formato: {1,20}** | |
|---|---|
| **Categorie** | **Scelte** |
| Lunghezza [lu] | 1. lunghezza < 1 [errore]<br>2. lunghezza >= 1 && lunghezza <= 20 [property lunghezzaLUok]<br>3. lunghezza > 20 [errore] |
| Esistente[eu] | 1. esistente==true[errore] |

| | |
|---|---|
| | 2. esistente==false[property esistenteEUok] |

| **Parametro: email**<br>***Formato: ^[\\w!#$%&'*+/=?`{\|}~^-]+(?:\\.[\\w!#$%&'*+/=?`{\|}~^-]+){1,64}@(?:[a-zA-Z0-9-]+\\.){1,200}[a-zA-Z]{2,6}$*** | |
|---|---|
| **Categorie** | **Scelte** |
| Formato [fe] | 1. Non rispetta il formato [errore]<br>2. Rispetta il formato[property formatoFEok] |
| Esistente[ee] | 1. esistente==true[errore]<br>esistente==false[property esistenteEEok] |

| **Parametro: Password**<br>***Formato: ^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z]).{8,15}$*** | |
|---|---|
| **Categorie** | **Scelte** |
| Formato [fp] | 1. Non rispetta il formato [errore]<br>2. Rispetta il formato [property formatoFPok] |

| **Parametro: Password**<br>***Formato: ^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z]).{8,15}$*** | |
|---|---|
| **Categorie** | **Scelte** |
| Match [mcp] | 1. Match con password = false [error]<br>2. Match con password = true [property MatchMCPok] |

| **Parametro: EULA**<br>***Formato: boolean*** | |
|---|---|
| **Categorie** | **Scelte** |
| Eulachecked [eula] | 1. Not Checked [error]<br>2. Checked [property EulaCheckedEulaok] |

| Codice | Combinazione | Esito |
|---|---|---|
| TC_1.1_01 | lu1 | Errore: Username troppo breve |
| TC_1.1_02 | lu3 | Errore: Username troppo lungo |

| | | |
|---|---|---|
| TC_1.1_03 | lu2.eu1 | Errore: Username già presente |
| TC_1.1_04 | lu2.eu2.fe1 | Errore: Formato email non valido |
| TC_1.1_05 | lu2.eu2.fe2.ee1 | Errore:email già presente |
| TC_1.1_06 | lu2.eu2.fe2.fp1 | Errore: Formato password non valido |
| TC_1.1_07 | lu2.eu2.le2.fp2.mcp1 | Errore: Le due password non corrispondono |
| TC_1.1_08 | lu2.eu2.le2.fp2.mcp2.eula1 | Errore: Il campo eula non è stato accettato |
| TC_1.1_09 | lu2.eu2.le2.fp2.mcp2.eula2 | Successo: Registrazione |

## TC_1.2 Login

| **Parametro: Email** | |
|---|---|
| ***Formato: ^[\\w!#$%&'*+/=?`{\|}~^-]+(?:\\.[\\w!#$%&'*+/=?`{\|}~^-]+){1,64}@(?:[a-zA-Z0-9-]+\\.){1,200}[a-zA-Z]{2,6}$*** | |
| **Categorie** | **Scelte** |

| | |
|---|---|
| Pre-esistente [prem] | 1. Non pre-esistente[errore]<br>2. Pre-esistente [property pre-esitentePREMok] |

| **Parametro: Password**<br>***Formato: ^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z]).{8,15}$*** | |
|---|---|
| **Categorie** | **Scelte** |
| Match [mp] | 1. Password non corretta [if pre-esistentePREMok] [errore]<br>2. Password corretta [if pre-esistentePREMok] [property matchMPok] |

| Codice | Combinazione | Esito |
|---|---|---|
| TC_1.2_01 | prem1 | Errore: Email non presente sulla piattaforma |
| TC_1.2_02 | prem2.mp1 | Errore: Password inserita non corrisponde a quella dell'utente |
| TC_1.2_03 | prem2.mp2 | Successo: Login |

## TC_1.3 Eliminazione Account

| **Parametro: Password**<br>**Formato: ^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z]).{8,15}$** | |
|---|---|
| **Categorie** | **Scelte** |
| Matchp [mpc] | 1. Password non corretta [error]<br>2. Password corretta [property matchpMPCok] |

| **Parametro: Email**<br>***Formato: ^[\\w!#$%&'*+/=?`{\|}~^-]+(?:\\.[\\w!#$%&'*+/=?`{\|}~^-]+){1,64}@(?:[a-zA-Z0-9-]+\\.){1,200}[a-zA-Z]{2,6}$*** | |
|---|---|
| **Categorie** | **Scelte** |
| Matche [mec] | 1. Email non corretta [errore]<br>2. Email corretta [property matchMPok] |

| **Parametro: Password**<br>***Formato: ^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z]).{8,15}$*** |
|---|

| Categorie | Scelte |
|---|---|
| Matchu [muc] | 1. Username non corretto [errore]<br>2. Username corretto [property matchMPok] |

| Codice | Combinazione | Esito |
|---|---|---|
| TC_1.3_01 | mpc1 | Errore: Password non corretta |
| TC_1.3_02 | mpc2.mec1 | Errore: Email non corretta |
| TC_1.3_03 | Mpc2.mec2.muc1 | Errore: Username non corretto |
| TC_1.3_04 | Mpc2.mec2.muc2 | Successo: Cancellazione |

## TC_2.5 Inserimento Commento

| **Parametro: Commento**<br>***Formato: {3,100}*** | |
|---|---|
| **Categorie** | **Scelte** |
| Lunghezza [lc] | 1. lunghezza < 3 [errore]<br>2. lunghezza >= 3 && lunghezza <= 100 [property lunghezzaLCok] |

| | |
|---|---|
| | 3. lunghezza > 100 [errore] |

| Codice | Combinazione | Esito |
|---|---|---|
| TC_2.5_01 | Ic.1 | Errore: Commento troppo breve |
| TC_2.5_02 | Ic.3 | Errore: Commento troppo lungo |
| TC_2.5_03 | Ic.2 | Successo: Inserimento |

## TC_2.6 Pubblicazione Storia

| **Parametro: Storia**<br>***Formato: {1,500}*** | |
|---|---|
| **Categorie** | **Scelte** |
| Lunghezza [ls] | 1. lunghezza < 1 [errore]<br>2. lunghezza >= 1 && lunghezza <= 500 [property lunghezzaLSok]<br>3. lunghezza > 500 [errore] |

| Codice | Combinazione | Esito |
|---|---|---|

| | | |
|---|---|---|
| TC_2.6_01 | Is.1 | Errore: Storia troppo breve |
| TC_2.6_02 | Is.3 | Errore: Storia troppo lunga |
| TC_2.6_03 | Is.2 | Successo: Pubblicazione |

# 10 Testing schedule

Le attività di pianificazione del testing avverranno come definito nei capitoli precedenti , cioè subito dopo la fase di design necessaria per la pianificazione.

La scrittura dei casi di test avverrà in contemporanea con lo sviluppo del codice.

L'esecuzione dei test avverrà sia durante che dopo l'implementazione del sistema. Una volta concluso lo sviluppo, tutti i test saranno rieseguiti per garantirne il corretto funzionamento e produrre i report finali.

Per altre informazioni si rimanda ai documenti di managment sullo schedule.