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

# TestPlan Progetto GuardaTV

| | |
|---|---|
| Riferimento | |
| Versione | 1.0 |
| Data | 10/02/2022 |
| Destinatario | Studenti di Ingegneria del Software 2021/22 |
| Presentato da | Gruppo 16 |
| Approvato da | |

# Revision History

| Data | Versione | Descrizione | Autori |
|---|---|---|---|
| 10/12/2021 | 0.1 | Prima stesura | N. Cacace<br>S. Pastore<br>A. Prezioso<br>A. Ricchetti |
| 12/12/2021 | 0.2 | Modifiche e Revisione | N. Cacace<br>S. Pastore<br>A. Prezioso |
| 10/02/2022 | 1.0 | Aggiunta sezioni mancanti | A. Ricchetti<br>S. Pastore |

# Sommario

# 1. Introduzione

Lo scopo di questo documento è quello di analizzare e gestire lo sviluppo e le attività di testing riguardanti il nostro sistema GuardaTv. Lo scopo di questa sessione di lavoro è quello di verificare il corretto funzionamento del sistema sviluppato in diversi casi. Tali casi sono stati studiati per mettere alla prova le varie funzionalità del software. Effettuando tali test saremo in grado di rilevare eventuali errori, bug o incongruenze tra il comportamento desiderato e quello effettivo del sistema. I risultati di questi test saranno utilizzati per capire dove bisognerà intervenire, e quindi correggere eventuali errori o apportare modifiche per il miglioramento dei vari sottosistemi.

# 2. Relazione con altri documenti

Per la corretta individuazione dei test case, si fa riferimento ad altri documenti prodotti.

## Relazioni con il Requirements Analysis Document (RAD)

I test case pianificati nel Test Plan sono elaborati in relazione ai requisiti funzionali e non funzionali presentati nel RAD

## Relazioni con il System Design Document (SDD)

I test case pianificati nel Test Plan devono rispettare la suddivisione in sottosistemi presentata nell'SDD.

## Relazioni con il Object Design Document (ODD)

Per ciò che concerne il test di unità, maggiormente legati allo ODD e alla divisione in package del sistema, essi saranno meglio documentati in altri documenti specifici del testing. Per tale motivo, nel presente documento, non vi saranno riferimenti al loro design.

# 3. Panoramica del sistema

Il sistema proposto basa la sua architettura sul modello three-tier.

Verranno usati HTML5, CSS3, JSTL e JavaScript per la parte di front-end relative al Presentation Layer.

Per la logica applicativa sarà utilizzato Java, in particolare i package Java Servlet e per il testing vengono usate le librerie Mockito, Junit ed il plugin JaCoCo.

Per la gestione del database saranno usati:

JDBC per il collegamento al database.

MySQL come database in fase di produzione.

# 4. Features da testare

Il testing verrà effettuato su tre Use Case :

- UC_NuovaLista_4
- UC_Recensione_10
- UC_AggiuntaContenuto_17

Sono escluse dal testing le funzionalità che non prevedono input manuale da parte dell'utente come la visualizzazione di dati.

# 5. Pass/Fail criteria

Le attività di testing sono mirate ad identificare la presenza di fault all'interno del sistema, per effettuare un successivo intervento di correzione.

L'esito di un test case è valutato mediante un oracolo, inteso come il risultato atteso della sua esecuzione, basandosi sui requisiti.

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

# 6. Approccio

## 6.1. Testing di sistema

Per il testing di sistema sarà utilizzato il tool Selenium IDE, che permette di registrare le azioni compiute da un utente sul browser, in modo da poter implementare ed eseguire i test di sistema in maniera automatica.
Il Server, per la fase di testing, verrà deployato in localhost.

## 6.2. Functional testing

Il functional testing ha il fine di validare i requisiti funzionali, Consiste nell'individuare i possibili faults generati dagli input degli utenti.

## 6.3. Performance testing

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

## 6.4. Pilot testing

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

## 6.5. Acceptance testing

A causa del basso budget non si assicura l'esecuzione del acceptance testing.

## 6.6. Installation testing

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

## 6.7. Testing di integrazione

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

## 6.8. Testing di unità

Per il testing di unità la strategia prevista consiste nel testare le 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, mediante l'uso del framework per il testing di classi Java JUnit.

## 6.9. Ispezione del codice

Sebbene vi sia l'intenzione di eseguire la fase di ispezione del codice, anche se non costante e approfondita, a causa del basso budget non si assicura tale pratica.

# 7. Sospensione e ripristino

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

### Criteri di sospensione

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

### Criteri di ripristino

Il testing verrà ripreso dopo aver risolto i fault individuati.

# 8. Materiale di testing

L'hardware necessario per l'attività di test è un semplice computer, non necessariamente connesso ad internet, in quanto il sistema non è ancora stato rilasciato.

# 9. Test Case di Sistema

## 9.1. Creazione nuova lista

| **Parametro: Nome**<br>**Categorie: Formato, lunghezza** | |
|---|---|
| FORMATO: [A-Za-z0-9] | |
| Lunghezza (ln) | 1. <1[errore]<br>2. >=1 and <=50 [property lunghezzaLNok]<br>3. >50 [errore] |
| Formato (fn) | 1. Rispetta il formato [property formatoFNok]<br>2. Non rispetta il formato [errore] |

| **Parametro: Descrizione**<br>**Categorie: Formato, lunghezza** | |
|---|---|
| FORMATO: [A-Za-z0-9] | |
| Lunghezza (ld) | 1. <=255 [property lunghezzaLDok]<br>2. >255 [errore] |
| Formato (fd) | 1. Rispetta il formato [property formatoFDok]<br>2. Non rispetta il formato [errore] |

| Codice | Combinazione | Esito |
|---|---|---|
| TC_NF_4_01 | ln1 | Errore |
| TC_NF_4_02 | ln3 | Errore |
| TC_NF_4_03 | ln2.fn2 | Errore |
| TC_NF_4_04 | ln2.fn1.ld2 | Errore |
| TC_NF_4_05 | ln2.fn1.ld1.fd2 | Errore |
| TC_NF_4_06 | ln2.fn1.ld1.fd1 | Successo |

## 9.2. Recensione Contenuto

| **Parametro: Punteggio**<br>**Categorie: Formato, lunghezza** | |
|---|---|
| FORMATO: [0-5]{1} | |
| Lunghezza (lp) | 1. <1[errore]<br>2. ==1 [property lunghezzaLPok]<br>3. >1 [errore] |
| Formato (fp) | 1. Rispetta il formato [property formatoFPok]<br>2. Non rispetta il formato [errore] |

| **Parametro: Descrizione**<br>**Categorie: lunghezza** | |
|---|---|
| Lunghezza (ld) | 1. <=255 [property lunghezzaLDok]<br>2. >255 [errore] |

| Codice | Combinazione | Esito |
|---|---|---|
| TC_RW_10_01 | lp1 | Errore |
| TC_RW_10_02 | lp3 | Errore |
| TC_RW_10_03 | lp2.fp2 | Errore |
| TC_RW_10_04 | lp2.fp1.ld2 | Errore |
| TC_RW_10_05 | Lp2.fp1.ld1 | Successo |

## 9.3. Aggiunta Contenuto

| **Parametro: ID**<br>**Categorie: Formato, lunghezza** | |
|---|---|
| FORMATO: [A-Z0-9]{4}-[A-Z0-9]{4}-[A-Z0-9]{4} \| [A-Z0-9]{4}-[A-Z0-9]{4}-[A-Z0-9]{4}-[A-Z0-9]{4}-[A-Z0-9]{1}-[A-Z0-9]{4}-[A-Z0-9]{4}-[A-Z0-9]{1} | |
| Lunghezza (li) | 1. !=14and !=33 [errore]<br>2. ==14 or ==33[property lunghezzaLPok] |
| Formato (fi) | 1. Rispetta il formato [property formatoFPok]<br>2. Non rispetta il formato [errore] |

| **Parametro: Titolo**<br>**Categorie: lunghezza** | |
|---|---|
| Lunghezza (lt) | 1. <1 [errore]<br>2. >=1 and <=50 [property lunghezzaLTok]<br>3. >50 [errore] |

| **Parametro: Descrizione**<br>**Categorie: lunghezza** | |
|---|---|
| Lunghezza (ld) | 1. <1 [errore]<br>2. >=1 and <=255 [property lunghezzaLDok]<br>3. >255 [errore] |

| **Parametro: Genere**<br>**Categorie: Formato, lunghezza** | |
|---|---|
| FORMATO: [A-Za-z] | |
| Lunghezza (lg) | 1. <1 [errore]<br>2. >=1 <=50 [property lunghezzaLRok]<br>3. >50 [errore] |
| Formato (fg) | 1. Rispetta il formato [property formatoFGok]<br>2. Non rispetta il formato [errore] |

| **Parametro: Regista**<br>**Categorie: lunghezza** | |
|---|---|
| Lunghezza (lr) | 1. <1 [errore]<br>2. >=1 and <=50 [property lunghezzaLRok]<br>3. >50 [errore] |

| **Parametro: Durata**<br>**Categorie: Formato, valore** | |
|---|---|
| FORMATO: [0-9] | |
| formato (fd) | 1. Rispetta il formato [property formatoFDok]<br>2. Non rispetta il formato [errore] |
| valore (vd) | 1. Valore < 1 [errore]<br>2. Valore >=1 && Valore <= 9999999999 [property formatoFDok]<br>3. Valore >9999999999 [errore] |

| **Parametro: Data di uscita**<br>**Categorie: Formato** | |
|---|---|
| FORMATO: yyyy-MM-dd | |
| formato (fr) | 1. Rispetta il formato [property formatoFRok]<br>2. Non rispetta il formato [errore] |

| Codice | Combinazione | Esito |
|---|---|---|
| TC_AC_17_01 | li1 | Errore |
| TC_AC_17_02 | li2.fi2 | Errore |
| TC_AC_17_03 | li2.fi1.lt1 | Errore |
| TC_AC_17_04 | li2.fi1.lt3 | Errore |
| TC_AC_17_05 | li2.fi1.lt2.ld1 | Errore |
| TC_AC_17_06 | li2.fi1.lt2.ld3 | Errore |
| TC_AC_17_07 | li2.fi1.lt2.ld2.lg1 | Errore |
| TC_AC_17_08 | li2.fi1.lt2.ld1.lg3 | Errore |
| TC_AC_17_09 | li2.fi1.lt2.ld1.lg2.fg2 | Errore |
| TC_AC_17_10 | li2.fi1.lt2.ld1.lg2.fg1.lr1 | Errore |
| TC_AC_17_11 | li2.fi1.lt2.ld1.lg2.fg1.lr3 | Errore |
| TC_AC_17_12 | li2.fi1.lt2.ld1.lg2.fg1.lr2.fd2 | Errore |
| TC_AC_17_13 | li2.fi1.lt2.ld1.lg2.fg1.lr2.fd1.vd1 | Errore |
| TC_AC_17_14 | li2.fi1.lt2.ld1.lg2.fg1.lr2.fd1.vd3 | Errore |
| TC_AC_17_15 | li2.fi1.lt2.ld1.lg2.fg1.lr2.fd1.vd2.fr2 | Errore |
| TC_AC_17_16 | li2.fi1.lt2.ld1.lg2.fg1.lr2.fd1.vd2.fr1 | Successo |

# 10. Testing schedule

Le attività di pianificazione del testing avverranno, come definito nei capitoli precedenti, 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