

TP

HAPPY FIELDS

Sommario

[1. Introduzione 2](#_Toc126858022)

[2. Relazione con altri documenti 2](#_Toc126858023)

[3. Panoramica del sistema 3](#_Toc126858024)

[4. Features da testare/non testare 3](#_Toc126858025)

[5. Pass/Fail criteria 3](#_Toc126858026)

[6. Approccio 4](#_Toc126858027)

[7. Sospensione e Ripristino 4](#_Toc126858028)

[8. Materiale di Testing 4](#_Toc126858029)

[9. Test Cases 4](#_Toc126858030)

[10. Testing Schedule 11](#_Toc126858031)

Team members

|  |  |
| --- | --- |
| Nome | Matricola |
| Carlo Tammaro | 0512112173 |
| Antonio Renzullo |  |
| Mattia Garofalo |  |
| Annachiara Varallo |  |

# Introduzione

Il documento che segue ha l’obbiettivo di descrivere e analizzare l’attività del testing per la piattaforma HappyFields. All’interno del documento sono riportate le strategie di testing e le funzionalità che verranno testate.

# 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 e di integrazione, maggiormente legati allo ODD e alla divisione in package del sistema, essi saranno scritti e documentati unicamente all’interno del codice dell’applicativo. Per tale motivo, nel presente documento, non vi saranno riferimenti al loro design.

# Panoramica del sistema

Il sistema proposto è basato sullo stile architetturale Three Tier, implementato utilizzando MVC. Il motivo è che tale architettura è perfetta per lo sviluppo di Web App, come visto e testato nel corso di TSW. In particolare, la separazione dalla logica di presentazione da quella di elaborazione, migliora alcuni aspetti implementativi andando a favorire lo sviluppo del sistema.

Verranno utilizzati HTML5, CSS3, ed eventuali altri Framework per la parte di front-end.

Per la logica applicativa e back-end sarà utilizzato **Java**

Per la gestione del database verrà usato **SQL**

# Features da testare/non testare

Di seguito la lista dei requisiti funzionali che sono stati testati:

* Registrazione Utente (Gestione Utente);
* Ricerca per orario e disponibilità (Gestione Evento);
* Creazione Evento (Gestione Evento);
* Modifica Dati (Gestione Gestore);

# Pass/Fail criteria

Le attività di testing sono mirate a identificare la presenza di errori all’interno del sistema.

L’esito di un test case è valutato mediante un oracolo, che esplicita se il test è andato a buon fine o meno.

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

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

Il test sarà considerato valido se saranno testati almeno **quattro** requisiti funzionali.

# Approccio

Il testing dell’intero sistema si compone di tre fasi: testing di sistema, testing di integrazione e testing di unità. Per lo scarso budget a disposizione non si assicura l’esecuzione dei test di integrazione e di unità.

///////COMPLETARE UN SECONDO

# 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 verrà 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.

# Materiale di Testing

L’hardware necessario per l’attività di testing è un semplice computer, che non necessita nemmeno di una connessione a internet.

# Test Cases

L’approccio per la definizione dei test frame sarà il category partition. Per definire l'output atteso si userà un oracolo umano, per via dell'assenza di specifiche formali/semi-formali.

GESTIONE UTENTE

Registrazione Utente (RF\_GU\_1)

|  |  |
| --- | --- |
| Parametro: Username | |
| Nome Categoria | Scelta per Categoria |
| Lunghezza [LU] | 1. Lunghezza > 30 OR Lunghezza < 1 [errore] 2. Lunghezza <= 30 AND Lunghezza >= 1 [ok] |
| Formato [FU] | 1. Nome.contains(!; ”; ”; £; $; %; &; /; (; ); =; ?; ^; ç; \_; -; @; °; §; [; ]; {; }; #) [errore] 2. Nome [ok] |
| Parametro: Città | |
| Verifica [VC] | 1. CittàInserita == NULL [errore] 2. Città [ok] |
| Parametro: Provincia | |
| Verifica [LP] | 1. Provincia > 2 AND Provincia < 2 [errore] 2. Provincia == 2[ok] |
| Parametro: Via | |
| Lunghezza [LV] | 1. Lunghezza > 30 OR Lunghezza < 1 [errore] 2. Lunghezza <= 30 AND Lunghezza >= 1 [ok] |
| Parametro: Numero di Telefono | |
| Lunghezza [LNT] | 1. Lunghezza < 10 OR Lunghezza > 10 [errore] 2. Lunghezza == 10 [ok] |
| Parametro: Password | |
| Lunghezza [LP] | 1. Lunghezza < 8 [errore] 2. Lunghezza >= 8 [ok] |
| Maiuscola [MP] | 1. !Password.contains(Maiuscola) [errore] 2. Password [ok] |
| Controllo Match [CP] | 1. Match con password = false [errore] 2. Match con password = true [errore] |

|  |  |  |
| --- | --- | --- |
| TEST\_CASE\_ID | Test Frame | Esito |
| TC 1.1\_1 | LU1 | Errato: username troppo lungo |
| TC 1.1\_2 | LU2, FU2 | Errato: formato username errato |
| TC 1.1\_3 | LU2, FU2, VC1 | Errato: il campo città è nullo |
| TC 1.1\_4 | LU2, FU2, VC2, LP1 | Errato: provincia troppo lunga |
| TC 1.1\_5 | LU2, FU2, VC2, LP2, LV1 | Errato: via troppo lunga |
| TC 1.1\_6 | LU2, FU2, VC2, LP2, LV2, LNT1 | Errato: numero di telefono troppo lungo |
| TC 1.1\_7 | LN2, FN2, VC2, LP2, LV2, LNT2, LP1, | Errato: password troppo breve |
| TC 1.1\_8 | LN2, FN2, VC2, LP2, LV2, LNT2, LP2, MP1 | Errato: la password deve contenere almeno una lettera maiuscola |
| TC 1.1\_9 | LN2, FN2, VC2, LP2, LV2, LNT2, LP2, MP2, CP1 | Errato: la password deve combaciare con la seconda password inserita |
| TC 1.1\_10 | LN2, FN2, VC2, LP2, LV2, LNT2, LP2, MP2, CP2 | Corretto: registrazione va a buon fine. |

GESTIONE EVENTO

Ricerca per orario e disponibilità (RF\_GE\_5)

|  |  |
| --- | --- |
| Parametro: Titolo | |
| Nome Categoria | Scelta per Categoria |
| Lunghezza [LT] | 1. Lunghezza > 50 OR Lunghezza < 1 [errore] 2. Lunghezza <= 50 AND Lunghezza >= 1 [ok] |
| Non Esiste [TNE] | 1. TitoloInserito.notExist [errore] 2. TitoloInserito [ok] |

|  |  |  |
| --- | --- | --- |
| TEST\_CASE\_ID | Test Frame | Esito |
| TC 2.1\_1 | LT1 | Errato: titolo troppo lungo |
| TC 2.1\_2 | LT2, TNE1 | Errato: titolo non esistente |
| TC 2.1\_3 | LT2, TNE2 | Corretto: ricerca va a buon fine |

Creazione Evento (RF\_GE\_1)

|  |  |
| --- | --- |
| Parametro: Titolo Evento | |
| Lunghezza [LT] | 1. Lunghezza > 50 OR Lunghezza < 1 [errore] 2. Lunghezza <= 50 AND Lunghezza >= 1 [ok] |
| Parametro: Sport | |
| Scelta Obbligatoria [SOS] | 1. Campo inserito = false [errore] 2. Campo inserito = true [ok] |
| Parametro: Campetto | |
| Scelta Obbligatoria [SOC] | 1. Campo inserito = false [errore] 2. Campo inserito = true [ok] |
| Parametro: Data | |
| Scelta Obbligatoria [SOD] | 1. Campo inserito = false [errore] 2. Campo inserito = true [ok] |
| Parametro: Ora | |
| Scelta Obbligatoria [SOO] | 1. Campo inserito = false [errore] 2. Campo inserito = true [ok] |

|  |  |  |
| --- | --- | --- |
| TEST\_CASE\_ID | Test Frame | Esito |
| TC 2.2\_1 | LT1 | Errato: titolo troppo lungo |
| TC 2.2\_2 | LT2, SOS1 | Errato: campo sport non scelto |
| TC 2.2\_3 | LT2, SOS2, SOC1 | Errato: campo campetto non scelto |
| TC 2.2\_4 | LT2, SOS2, SOC2, SOD1 | Errato: campo data non scelto |
| TC 2.2\_5 | LT2, SOS2, SOC2, SOD2, SOO1 | Errato: campo ora non scelto |
| TC 2.2\_6 | LT2, SOS2, SOC2, SOD2, SOO2 | Corretto: la creazione dell’evento va a buon fine |

GESTIONE GESTORE

Modifica dati campetti (RF\_GG\_3)

|  |  |  |
| --- | --- | --- |
| Parametro: Numero Giocatori | | |
| Nome Categoria | | Scelta per Categoria |
| Massimo [MNG] | | 1. NumeroGiocatori <= 2 AND NumeroGiocatori >= 11 [errore] 2. NumeroGiocatori >= 2 AND NumeroGiocatori <= 11 [ok] |
| Parametro: Città | | |
| Lunghezza [LC] | | 1. Lunghezza <= 2 OR Lunghezza > 20 [errore] 2. Lunghezza > 2 AND Lunghezza <= 20 [ok] |
| Parametro: Provincia | | |
| Lunghezza [LP] | | 1. Lunghezza > 2 OR Lunghezza < 2 [errore] 2. Lunghezza == 2 [OK] |
| Parametro: Via | | |
| Lunghezza [LV] | 1. Lunghezza <= 3 OR Lunghezza > 30 [errore] 2. Lunghezza > 3 AND Lunghezza <= 30 [ok] | |
| Parametro: Costo | | |
| Verifica [VC] | | 1. Costo <= 1 AND Costo >= 300 [errore] 2. Costo >= 1 AND Costo <= 300 [ok] |
| Parametro: Lunghezza | | |
| Lunghezza [LL] | | 1. Lunghezza >= 150 mt AND Lunghezza <= 10mt [errore] 2. Lunghezza <= 150 mt AND Lunghezza >= 10mt [ok] |
| Paramentro: Larghezza | | |
| Lunghezza [LLA] | | 1. Lunghezza >= 50 mt AND Lunghezza <= 5mt [errore] 2. Lunghezza <= 50 mt AND Lunghezza >= 5mt [ok] |

|  |  |  |
| --- | --- | --- |
| TEST\_CASE\_ID | Test Frame | Esito |
| TC 3.1\_1 | MNG1 | Errato: il numero di giocatori è troppo alto. |
| TC 3.1\_2 | MNG2, LC1 | Errato: la lunghezza della città è sbagliata |
| TC 3.1\_3 | MNG2, LC2, LP1 | Errato: la lunghezza della provincia è sbagliata |
| TC 3.1\_4 | MNG2, LC2, LP2, LV1 | Errato: la lunghezza della via è sbagliata |
| TC 3.1\_5 | MNG2, LC2, LP2, LV2, VC1 | Errato: il costo è troppo elevato. |
| TC 3.1\_6 | MNG2, LC2, LP2, LV2, VC1, LL1 | Errato: la lunghezza è troppo elevata. |
| TC 3.1\_7 | MNG2, LC2, LP2, LV2, VC2, LL2, LLA1 | Errato: la larghezza è troppo elevata. |
| TC 3.1\_8 | MNG2, LC2, LP2, LV2, VC2, LL2, LLA2 | Corretto: la modifica va a buon fine. |

# Testing Schedule

Le attività di pianificazione dei testing avverranno dopo la fase di design necessaria per la pianificazione.

La scrittura dei casi di test avverrà durante la scrittura del codice.

L’esecuzione dei casi di test avverrà durante e dopo l’implementazione del sistema.