***Easy Fut5al***

**Test Plan Document**

Sommario

[**1.** **Introduzione** 1](#_Toc534479393)

[**2.** **Riferimenti** 4](#_Toc534479399)

[**3.** **Obiettivi del testing** 5](#_Toc534479400)

[**4.** **Funzionalità da testare** 9](#_Toc534479406)

[**1.** **Funzionalità da non testare** 1](#_Toc534479393)

[**2.** **Approccio** 4](#_Toc534479399)

[**3.** **Criteri di sospensione e di uscita** 5](#_Toc534479400)

[**4.** **Test Cases** 9](#_Toc534479406)

[**4.** **Scheduling** 9](#_Toc534479406)

# 

# **Introduzione**

Il Test plan si focalizza sugli aspetti manageriali del testing, in particolare sullo scopo, l’approccio e lo scheduling di tutte le attività del testing del progetto Easy-Fut5al.

# **Riferimenti**

* Bernd Bruegge & Allen H. Dutoit, *Object-Oriented Software;*
* RAD-EasyFut5al;

# **3. Obiettivo del testing**

L’obiettivo del testing che vogliamo effettuare è quello di verificare le funzionalità individuate per il progetto EasyFut5al, in particolare si vogliono individuare le differenze tra il comportamento atteso specificato nel modello di sistema e il comportamento osservato con l’immissione di alcuni input selezionati.

Nello specifico, ci è stato richiesto di sviluppare la documentazione di pianificazione ed esecuzione del testing per almeno uno dei sottosistemi. Per motivi di tempistica ne testeremo soltanto uno:

***Storage,***  il quale ci risulta il più agevole per effettuare testing black box non avendo a che fare con UI a eventi o Servlets.

# **4. Funzionalità da testare**

|  |  |  |
| --- | --- | --- |
| **Sottosistema di appartenenza** | **Nome funzionalità** | **Descrizione** |
| **Storage** | Salva | Salva un oggetto Bean nel database. |
| **Storage** | GetLista | Torna una lista di Bean del tipo specificato. |
| **Storage** | GetOggetto | Torna un oggetto identificato da un ID. |
| **Storage** | Aggiorna | Sostiuisce un record del db . |
| **Storage** | Elimina | Elimina un record nel db che corrisponde ai valori del bean dato in input. |

# **5. Funzionalità da non testare**

**Tutte le funzioni relative ai package:**

**Middleware**

**Prenotazioni Management**

**Conferma Campetto**

**Login/Logout**

**Registrazione Atleta/ Registrazione Gestore**

**Activity Atleta (comprende Sezione Home, FragmentNuovaPartita, Inviti)**

**Activity Gestore**

**Activity Autenticazione**

**Profilo**

**Calendario**

**ConnectionTask**

# **6. Approccio**

Nella sessione di testing del sistema verrà utilizzato un approccio di tipo "BLACK BOX", che prevede che i test vengano effettuati in maniera da non scendere nei dettagli del codice, ma basandoci sulle specifiche delle funzionalità da testare.

# **7. Criteri di sospensione e di uscita**

**Criteri di sospensione**

Abbiamo stabilito che se i membri del team riportano che più del 20% dei test case sono falliti, il testing dovrà essere sospeso fino a quando non verranno individuati e risolti i bug che hanno causato il fallimento.

**Criteria di uscita**

Specifica dei criteri che denotano il successo della fase di testing:

- La percentuale di successo è dell’80% ed è la percentuale minima per il completamento del testing.

# **8. Test cases**

Per sviluppare i test case sarà utilizzato il Metodo **WECT:**

Inizieremo a partizionare il dominio di input in diverse classi di equivalenza.

Nel caso di interi (es: ID), si crea una classe di equivalenza VALIDA per i valori nell’intervallo accettato e classi NON VALIDE per valori minori o maggiori che sforano l’intervallo.

***I1: 1<= ID <= NumeroDiRecord***

***I2: ID< 1***

***I3: ID > NumeroDiRecord***

Se trattasi di input di un elemento di un insieme discreto, crea una classe per ogni valore VALIDO e una classe che rappresenta I valori non ammissibili

***S1: “storage.Atleta”***

***S2: “storage.Campetto”***

***S3: “storage.Gestore”***

***S4: “storage.Gioca”***

***S5: “storage.Partita”***

***S6: “Albicocca” //Non valido***

Dopodiché si sceglie un valore per una variabile da ogni classe.

|  |  |  |
| --- | --- | --- |
| **Test Case** | **I** | **S** |
| WE1 | I1 | S1 |
| WE2 | I2 | S2 |
| WE3 | I3 | S3 |
| WE4 | I1 | S4 |
| WE5 | I2 | S5 |
| WE6 | I3 | S6 |

Con queste premesse, sviluppiamo I test per le funzionalità indicate in precedenza:

***Salva***

Parametro: Bean da salvare

|  |  |  |
| --- | --- | --- |
| **Codice TestCase** | **Combinazione** | **Esito** |
| Salva\_Atleta | S1 | Pass |
| Salva\_Gestore | S2 | Pass |
| Salva\_Campetto | S3 | Pass |
| Salva\_Gioca | S4 | Pass |
| Salva\_Partita | S5 | Pass |
| Salva\_ERR | S6 | Pass |

***GetLista***

Parametro: Nome lista

|  |  |  |
| --- | --- | --- |
| **Codice TestCase** | **Combinazione** | **Esito** |
| GetLista\_Atleta | S1 | Pass |
| GetLista \_Gestore | S2 | Pass |
| GetLista \_Campetto | S3 | Pass |
| GetLista \_Gioca | S4 | Pass |
| GetLista \_Partita | S5 | Pass |
| GetLista \_ERR | S6 | Pass |

***GetOggetto***

Parametri: Stringa tipo oggetto, ID oggetto

|  |  |  |
| --- | --- | --- |
| **Codice TestCase** | **Combinazione** | **Esito** |
| Get\_Oggetto\_AtletaValido | I1.S1 | Pass |
| Get\_Oggetto\_GestoreNonValido | I2.S2 | Pass |
| Get\_OggettoCampettoNonValido | I3.S3 | Pass |
| Get\_Oggetto\_GiocaValido | I1.S4 | Pass |
| Get\_OggettoPartitaNonValido | I2.S5 | Pass |
| Get\_OggettoNullNonValido | I3.S6 | Pass |

***Aggiorna***

Parametri: Bean da rimpiazzare

|  |  |  |
| --- | --- | --- |
| **Codice TestCase** | **Combinazione** | **Esito** |
| Aggiorna\_Atleta | S1 | Pass |
| Aggiorna\_Gestore | S2 | Pass |
| Aggiorna\_Campetto | S3 | Pass |
| Aggiorna\_Gioca | S4 | Pass |
| Aggiorna\_Partita | S5 | Pass |
| Aggiorna\_ERR | S6 | Pass |

***Elimina***

Parametri: Bean da eliminare

//E’ stata implementata solo la funzione Elimina per il Bean Gioca

|  |  |  |
| --- | --- | --- |
| **Codice TestCase** | **Combinazione** | **Esito** |
| Elimina\_Gioca | S1 | Pass |