Easy Fut5al

Test Plan Document

Sommario

1. 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.

2. Riferimenti

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

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 testeremo solamente il sottosistema ***Calendario.***

La funzione Calendario si occupa di fornire in output le partite da giocare in un campetto scelto in una data giornata. Il client Android fa richiesta di suddetta funzione spedendo dei parametri (grazie all'interfaccia ConnectionTask) verso la componente ***Middleware****, sul lato Server.*
La risposta è servita dalle Servlet ***CercaPartiteServlet*** *e* ***getInfoServlet*** *all'interno di quest'ultima.*
Le Servlet a loro volta fanno uso della componente ***Storage****, che è incaricata di restituire valori di record del database.*

4. Funzionalità da testare

Sottosistema di appartenenza	Nome funzione	Descrizione
Middleware	GetInfoServlet.DoGet	Risponde al client Android con una lista info di oggetti
Middleware	CercaPartiteServlet.Doget	Risponde al client Android con una lista di partite, selezionate in base ai valori forniti dalla request.
Storage	storageFacade.getLista	Restituisce una lista di un tipo di oggetti che risiedono del database.
Storage	storage.getOggetto	Restituisce un oggetto identificato da un ID prelevato dal Database di un certo tipo.

5. Funzionalità da non testare

Registrazione
Crea Nuova Partita
Invita giocatori e partecipazione a partite
Visualizza Profilo
Valutazione Campetto
Visualizza Inviti
Ricerca per città

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.

In particolare, con quest'approccio abbiamo previsto due differenti tipi di testing:

Unit Testing: che agisce sui singoli componenti.
Integration Testing: che va a testare l'integrazione dei vari sottosistemi.

Per testare l'intera funzionalità procederemo con l'effettuare dei Test di Unità sui metodi di StorageFacade (del package Storage), dopodiché effettueremo l'Integration Testing con le servlets. Questa tecnica viene definita come "**Bottom up**" poiché i sottosistemi nel layer più basso della gerarchia sono testati individualmente, e quelli del livello superiore chiamano i componenti sottostanti già testati in precedenza. Per tal motivo non è necessario creare stub dall'implementazione onerosa, ecco perché è molto utile per la programmazione ad oggetti.

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:

COMPONENTE STORAGE

Classi di equivalenza per Storage.GetLista e Storage.GetOggetto:

I1: 1<= ID <= NumeroDiRecord
I2: ID< 1
I3: ID > NumeroDiRecord

S1: "storage.Atleta"
S2: "storage.Gestore"
S3: "storage.Gioca"
S4: "storage.Campetto"
S5: "storage.Partita"
S6: "Albicocca" //Non valido

GetLista

Parametro: Nome lista

Codice TestCase	Combinazione	Esito
GetLista_Atleta	S1	Pass
GetLista _Gestore	S2	Pass
GetLista _Gioca	S3	Pass
GetLista _Campetto	S4	Pass
GetLista _Partita	S5	Pass
GetLista _ERR	S6	Error //poi corretto

GetOggetto

Parametri: Stringa tipo oggetto, ID oggetto

Codice TestCase	Combinazione	Esito
Get_Oggetto_AtletaValido	I1.S1	Error //poi corretto
Get_Oggetto_GestoreNonValido	I2.S2	Pass
Get_OggettoGiocaNonValido	I3.S3	Error //poi corretto
Get_Oggetto_CampettoValido	I1.S4	Pass
Get_OggettoPartitaNonValido	I2.S5	Pass
Get_OggettoNullNonValido	I3.S6	Pass

COMPONENTE MIDDLEWARE

Classi di equivalenza per GetInfoServlet

T1: "Atleta"
T2: "Campetto"
T3: "Albicocca" // Non Valido

GetInfoServlet

Parametri: Tipo Oggetto

Codice TestCase	Combinazione	Esito
InfoDoGet_Atleta	T1	Pass
InfoDoGet_Campetto	T2	Pass
InfoDoGet_ERR	T3	Pass

Classi di equivalenza per CercaPartiteServlet

P1: "si"
P2: "no"
P3: "indifferente"
P4: "Albicocca" // Fuori dominio

M1: "DaGiocare"
M2: "InSospeso"
M3: "Terminata"
M4: "Albicocca" //Fuori dominio

E1: Classe Email valide
E2: Classe Email non valide

CercaPartiteServlet

Parametri: Tipo partecipazione, Tipo partite, Email

Codice TestCase	Combinazione	Esito
Mia_DaGiocare_Valid	P1.M1.E1	Pass
InSos_NonValid	P2.M2.E2	Pass
Terminata_Valid	P3.M3.E1	Pass
Error_NonValid	P4.M4.E2	Pass