Laurea Magistrale in informatica - Università di Salerno
Corso di *Gestione dei Progetti Software* - Prof.ssa F. Ferrucci

Test Plan

Riferimento	C03_TP
Versione	1.1
Data	18/01/2021
Destinatario	Prof.ssa Filomena Ferrucci
Presentato da	Giosuè Sulipano
Approvato da	Andrea Massaro

Revision History

Data	Versione	Descrizione	Autori
14/12/2020	0.1	Prima stesura	Giosuè Sulipano
15/12/2020	1.0	Aggiunta dei Test Case	Giosuè Sulipano
18/01/2021	1.1	Revisione	Giosuè Sulipano

Team

Nominativo	Ruolo
Giosuè Sulipano	PM
Andrea Massaro	Revisore
Edoardo Iannaccone	Sviluppatore
Giampiero Ferrara	Sviluppatore
Emma Manzo	Sviluppatore
Marco Sica	Sviluppatore
Michele Iannucci	Sviluppatore
Umberto Franzese	Sviluppatore

Sommario

1 Introduzione

myPersonalTrainer si propone di semplificare il lavoro del Personal Trainer, agevolando l'organizzazione degli appuntamenti, la consultazione dello storico dei pagamenti e l'analisi dei progressi di ogni singolo atleta, al fine di aumentare la sua l'efficienza.

Il seguente documento di Test Plan ha l'obiettivo di descrivere ed analizzare le attività di testing per il sistema myPersonalTrainer, al fine di garantire il corretto funzionamento del sistema.

All'interno del documento sono indicate le strategie di testing adottate, le funzionalità da testare e gli strumenti scelti che saranno utilizzati per la rilevazione degli errori, con lo scopo di presentare sul mercato un sistema utilizzabile e privo di malfunzionamenti.

Le attività di testing sono state pianificate per le seguenti gestioni:

- Gestione Utente
- Gestione Scheda di Allenamento
- Gestione Agenda
- Gestione Personal Trainer

1.2 Riferimenti

- Documento di progetto Requirements Analysis Document (RAD).
- Documento di progetto System Design Document (SDD).
- Documento di progetto Object Design Document (ODD).

I documenti sopraelencati sono disponibili nel repository di progetto.

2 Panoramica del sistema

Il sistema proposto è una webapp che basa la sua architettura sul modello MVC. Per il suo sviluppo saranno utilizzati HTML5, CSS3, JavaScript (jQuery) per il front-end e Java per il back-end. Inoltre, si farà utilizzo del framework Bootstrap per lo sviluppo delle View in tempi ridotti e di Firebase per il database NoSQL su cloud.

3 Feature da testare

Le features da testare per ogni gestione saranno le seguenti:

- Gestione Utente
    - Autenticazione Utente
    - Gestione Parametri Fisici
    - Gestione Account
- Gestione Scheda di Allenamento
    - Realizzazione Scheda di Allenamento
- Gestione Agenda
    - Cancellazione Appuntamento
    - Richiesta Appuntamento
    - Inserimento Disponibilità Personal Trainer
- Gestione Personal Trainer
    - Creazione Account Cliente

4 Feature da non testare

Le funzionalità escluse dal testing riguardano i requisiti funzionali di bassa e media priorità o i requisiti per cui non è necessario creare dei casi di test perché non accettano input utente particolari (ad esempio le funzionalità di visualizzazione).

Di seguito sono elencate le funzionalità da non testare:

- Gestione Utente
    - Gestione Abbonamento
    - Visualizzazione Progressi
    - Effettuazione Pagamenti
- Gestione Scheda di Allenamento
    - Visualizzazione Scheda di Allenamento
    - Esportazione Scheda di Allenamento
    - Richiesta Scheda di Allenamento
- Gestione Agenda
    - Consultazione Agenda
    - Verifica Appuntamenti
- Gestione Personal Trainer
    - Visualizzazione Progressi Cliente

  - Visualizzazione Abbonamenti
  - Visualizzazione Informazioni Cliente
- Gestione Avvisi
  - Notifica Scadenza
  - Notifica Cancellazione
  - Notifica Creazione

5 Pass/Fail criteria

Un test avrà successo (pass) se, dato un input al sistema, l'output ottenuto sarà diverso dall'output atteso dall'oracolo, ovvero la sua esecuzione determina una failure. Al contrario, un test fallirà (fail) quando il risultato osservato è uguale a quello atteso, quindi uguale all'oracolo.
Il testing sarà ritenuto valido se saranno rispettati i seguenti vincoli:

- Testare tutti i requisiti ad alta priorità elencati nel capitolo precedente.
- 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 di sistema si compone di tre parti, eseguiti nell'ordine inverso e progettati secondo il seguente ordine:

- Testing di Sistema
- Testing di Integrazione
- Testing di Unità

La progettazione dei casi di test di sistema e di integrazione avverrà prima della fase di implementazione del sistema e, inoltre, verranno raffinati durante la loro esecuzione. La progettazione dei casi di test di unità avverrà durante la fase di implementazione.
La loro esecuzione avverrà durante la stesura del codice, in modo parallelo, così da avere un riscontro immediato degli errori di implementazione. Ogni volta che verrà introdotta una nuova funzionalità verranno eseguiti i test ad essa legata e, inoltre, i test di integrazione che la coinvolgono. Ad ogni modifica delle componenti, verranno eseguiti i test di unità e di integrazione delle componenti direttamente

dipendenti da quelle modificate. Inoltre, verrà effettuato testing di regressione, rilanciando i casi di test delle componenti apparentemente non impattate dalla modifica, per evitare il Ripple Effect.

Testing di Sistema

Per il testing di sistema sarà utilizzato il tool Selenium IDE. Questo permette di registrare le azioni che un utente svolge sul browser, in modo tale da implementare ed eseguire i test case di sistema. Per il testing, il server sarà deployato in localhost.

Testing di Integrazione

L'approccio che verrà utilizzato è il bottom-up, ritenuto il più adatto per i software basati sul paradigma Object Oriented. Si farà utilizzo del framework JUnit per la definizione dei test case in Java, mentre Mockito sarà utilizzato per il mocking delle componenti. Inoltre, per soddisfare i criteri di premialità, si farà utilizzo delle GitHub Actions per realizzare la Continuous Integration. L'esecuzione dei test sarà automatizzata e gestita da Maven, mentre il tool JaCoCo sarà utilizzato per la misurazione e il report della coverage.

Testing di Unità

Per il testing di unità si andranno a testare ogni metodo di ciascuna classe del sistema, fatta eccezione per il testing unitario delle interfacce e delle classi di domain, poiché quest'ultime presentano solo i metodi getter e setter (autogenerati dall'IDE). Si definiranno i casi di test in modo funzionali (black-box). Per la definizione dei casi di test unitari, così come il testing di integrazione, sarà utilizzato il framework JUnit. Verrà definita una classe di test per ciascuna classe sorgente. Se l'esecuzione di tutti i test unitari non porta alla branch coverage minima richiesta (75%), verranno definiti altri casi di test in modo strutturale (white-box). Come per il testing di integrazione, saranno utilizzati Mockito, JaCoCo, Maven e GitHub Actions con gli stessi scopi. Infine, l'esecuzione dei casi di test unitari precede quella dei test di integrazione.

7 Sospensione e ripristino

Per la sospensione del testing, questo non verrà sospeso fino alla sua terminazione, anche in caso di rilevazione di una failure. Inoltre, il testing verrà ripreso dopo aver risolto le failure individuare nell'esecuzione precedente.

8 Materiale di testing

Il necessario per l'attività di testing è un pc con l'ultima versione del sistema ottenuta tramite la repository di progetto.

9 Test cases

Per l'individuazione dei test cases elencati di seguito si è fatto utilizzo della metodica del Category Partition e delle Classi di Equivalenza.

9.1 Gestione Utente

9.1.1 Autenticazione Utente

Parametro: E-mail utente **Formato: \w+([\._\-]?\w+)*@\w+([\.\-]?\w+)*(\.\w+)+$**	
Lunghezza [LE]	1. <7 or >25 [error] 2. >=7 and <=25 [property LE_OK]
Formato [FE]	1. Non match [if LE_OK] [error] 2. Match [if LE_OK] [property FE_OK]

Codice	Test Frame	Esito
TC_1.1_1	LE1	Errore: lunghezza non valida
TC_1.1_2	LE2, FE1	Errore: formato non valido
TC_1.1_3	LE2, FE2	Corretto: tentativo di autenticazione.

9.1.2 Gestione Parametri Fisici

Parametro: Peso **Formato: / ([0-9]){2,3}\\.?[0-9]{0,2}?$/**	
Lunghezza [LP]	1. <2 or >3 [error] 2. >=2 and <=3 [property LP_OK]
Formato [FP]	1. Non match [if LP_OK] [error] 2. Match [if LP_OK] [property FP_OK]

Parametro: Massa magra **Formato: /([0-9]+\%){1,2}$/**	
Lunghezza [LMM]	1. <2 or >3 [error]

	2. >=2 and <=3 [property LMM_OK]
Formato [FMM]	1. Non match [if LMM_OK] [error] 2. Match [if LMM_OK] [property FMM_OK]

Parametro: Massa grassa **Formato: /([0-9]+\%){1,2}$/**	
Lunghezza [LMG]	1. <2 or >3 [error] 2. >=2 and <=3 [property LMG_OK]
Formato [FMG]	1. Non match [if LMG_OK] [error] 2. Match [if LMG_OK] [property FMG_OK]

Codice	Test Frame	Esito
TC_1.2_1	LP1	Errore: lunghezza non valida
TC_1.2_2	LP2, FP1	Errore: formato non valido
TC_1.2_3	LP2, FP2, LMM1	Errore: lunghezza non valida
TC_1.2_4	LP2, FP2, LMM2, FMM1	Errore: formato non valido
TC_1.2_5	LP2, FP2, LMM2, FMM2, LMG1	Errore: lunghezza non valida
TC_1.2_6	LP2, FP2, LMM2, FMM2, LMG2, FMG2	Corretto: inserimento parametri fisici

9.1.3 Gestione Account

Parametro: E-mail utente **Formato: \w+([\._\-]?\w+)*@\w+([\.\-]?\w+)*(\.\w+)+$**	
Lunghezza [LE]	1. <7 or >25 [error] 2. >=7 and <=25 [property LE_OK]
Formato [FE]	1. Non match [if LE_OK] [error] 2. Match [if LE_OK] [property FE_OK]

Parametro: Password utente **Formato: /^[a-zA-Z 0-9 \@\._\!\?\-]{8,}$/**	
Lunghezza [LP]	1. <1 or >30 [error] 2. >=1 and <=30 [property LP_OK]
Formato [FP]	1. Non match [if LP_OK] [error] 2. Match [if LP_OK] [property FP_OK]

Codice	Test Frame	Esito
TC_1.3_1	LE1	Errore: lunghezza non valida
TC_1.3_2	LE2, FE1	Errore: formato non valido
TC_1.3_3	LE2, FE2, LP1	Errore: lunghezza non valida
TC_1.3_4	LE2, FE2, LP2, FP1	Errore: formato non valido
TC_1.3_5	LE2, FE2, LP2, FP2	Corretto: modifica effettuata

9.2 Gestione Scheda di Allenamento

9.2.1 Realizzazione Scheda di Allenamento

Parametro: Nome esercizio **Formato: \s\w+\s**	
Lunghezza [LN]	1. <4 or >25 [error] 2. >=4 and <=25 [property LN_OK]
Formato [FN]	1. Non match [if LN_OK] [error] 2. Match [if LN_OK] [property FN_OK]

Parametro: Numero ripetizioni **Formato: /^[0-9]+$/**	
Lunghezza [LR]	1. <1 or >3 [error] 2. >=1 and <=3 [property LR_OK]
Formato [FR]	1. Non match [if LR_OK] [error] 2. Match [if LR_OK] [property FR_OK]

Parametro: Numero serie **Formato: /^[0-9]+$/**	
Lunghezza [LS]	1. <1 or >3 [error] 2. >=1 and <=3 [property LS_OK]
Formato [FS]	1. Non match [if LS_OK] [error] 2. Match [if LS_OK] [property FS_OK]

Parametro: Tempo recupero in secondi **Formato: /^[0-9]+$/**	
Lunghezza [LT]	1. <1 or >3 [error] 2. >=1 and <=3 [property LT_OK]
Formato [FT]	1. Non match [if LT_OK] [error] 2. Match [if LT_OK] [property FT_OK]

Codice	Test Frame	Esito
TC_2.1_1	LN1	Errore: lunghezza non valida
TC_2.1_2	LN2, FN1	Errore: formato non valido
TC_2.1_3	LN2, FN2, LR1	Errore: lunghezza non valida
TC_2.1_4	LN2, FN2, LR2, FR1	Errore: formato non valido
TC_2.1_5	LN2, FN2, LR2, FR2, LS1	Errore: lunghezza non valida
TC_2.1_6	LN2, FN2, LR2, FR2, LS2, FS1	Errore: formato non valido
TC_2.1_7	LN2, FN2, LR2, FR2, LS2, FS2, LT1	Errore: lunghezza non valida
TC_2.1_8	LN2, FN2, LR2, FR2, LS2, FS2, LT2, FT1	Errore: formato non valido.

TC_2.1_9	LN2, FN2, LR2, FR2, LS2, FS2, LT2, FT2	Corretto: creazione scheda di allenamento

9.3 Gestione Agenda

9.3.1 Cancellazione Appuntamento

Parametro: Appuntamento	
Selezione [SA]	1. Non selezionato [error] 2. Selezionato [property SA_OK]

Codice	Test Frame	Esito
TC_3.1_1	SA1	Errore: appuntamento non selezionato
TC_3.1_2	SA2	Corretto: cancellazione appuntamento

9.3.2 Richiesta Appuntamento

Parametro: Appuntamento	
Disponibilità [DA]	1. Non disponibile [error] 2. Disponibile [property DA_OK]
Selezione [SA]	1. Non selezionato [error] 2. Selezionato [if DA_OK] [property SA_OK]

Codice	Test Frame	Esito
TC_3.2_1	DA1	Errore: appuntamento non disponibile
TC_3.2_2	DA2, SA1	Errore: appuntamento non selezionato
TC_3.2_3	DA2, SA2	Corretto: appuntamento richiesto

9.3.3 Inserimento Disponibilità Personal Trainer

Parametro: Data	
Selezione [SD]	1. Non selezionata [error] 2. Selezionata [property SA_OK]

Parametro: Fascia Oraria	
Selezione [SF]	1. Non selezionata [error] 2. Selezionato [if SD_OK] [property SF_OK]

Codice	Test Frame	Esito
TC_3.3_1	SD1	Errore: data non selezionata
TC_3.3_2	SD2, SF1	Errore: fascia oraria non selezionata
TC_3.3_3	SD2, SF2	Corretto: disponibilità inserita

9.4 Gestione Personal Trainer

9.4.1 Creazione Account Cliente

Parametro: E-mail cliente **Formato: /w+([\._\-]?\w+)*@\w+([\.\-]?\w+)*(\.\w+)+$/**	
Lunghezza [LE]	1. <7 or >25 [error] 2. >=7 and <=25 [property LE_OK]
Formato [FE]	1. Non match [if LE_OK] [error] 2. Match [if LE_OK] [property FE_OK]

Parametro: Data scadenza abbonamento	
Selezione [SD]	1. Non selezionata [error] 2. Selezionata [property LE_OK]

Codice	Test Frame	Esito
TC_4.1_1	LE1	Errore: lunghezza non valida
TC_4.1_2	LE2, FE1	Errore: formato non valido
TC_4.1_3	LE2, FE2, SD1	Errore: data non selezionata
TC_4.1_4	LE2, FE2, SD2	Corretto: nuovo account creato

10 Testing schedule

Per lo schedule del testing si rimanda ai documenti di progetto.