Università degli Studi di Salerno
Dipartimento di Informatica

GameUp

Test Plan Document

Studenti
Francesco Foglia

Docente
Prof. Andrea De Lucia

Anno Accademico 2020-2021

Cronologia Revisioni

Data	Versione	Descrizione
26/01/2021	1.0	Release iniziale

Indice

Capitolo 1

Introduzione

A causa del tempo ridotto, verranno eseguiti solamente dei test per il rilevamento di errori delle principali funzionalità implementate, ovvero registrazione, login e creazione di una richiesta di pubblicazione. Le ulteriori funzionalità implementate verranno testate nel prossimo processo iterativo di sviluppo, assieme allo sviluppo delle ulteriori funzionalità richieste.

Capitolo 2

Approccio

L'approccio utilizzato sarà di tipo black-box: verranno testate le funzionalità implementate specificate nel RAD senza considerare l'implementazione del codice. Il testing verrà eseguito in modo bottom-up, per evitare l'implementazione di stub ritenuti costosi a causa del paradigma *object-oriented* utilizzato, partendo dalle Repositories che fungono da astrazione per gli oggetti di accesso ai dati permanenti, implementando degli Unit test che testano i singoli metodi in maniera procedurale, poiché tali classi non dipendono da alcuno stato. Proseguendo, verranno verificati i vari Services implementati, senza l'utilizzo di stub poiché sono strettamente correlate alle repositories utilizzate. Terminando, verranno testati gli oggetti Control e le interfacce grafiche (le *view*) che essi ritornano, verificando che l'interfaccia tra la parte front-end e la back-end permetta effettivamente la comunicazione e la chiamata di procedure per effettuare azioni come la visualizzazione o il salvataggio di dati. Gli input per i vari test verranno scelti partizionando l'intero insieme di possibili input in classi di equivalenza, per ridurre il numero di input da utilizzare per la scelta dei vari test cases.

Capitolo 3

Test Case Plan

3.1 Registrazione

Parametro	Username
Esistente [UE]	• L'username esiste già nel database [Error] • L'username è unico [UsernameUnico]

Parametro	Email
Esistente [EE]	• L'email esiste già nel database [Error] • L'email è unica [EmailUnica, if UsernameUnico]

Parametro	Avatar
Presente [PA]	• L'avatar è effettivamente una istanza di UploadedFile [AvatarPresente, if UsernameUnico and EmailUnica] • L'avatar non viene passato ed è quindi null [AvatarAssente, if UsernameUnico and EmailUnica]

Codice	Combinazione	Esito
TC1_1	UE1	Errore: Username esistente
TC1_2	UE2,EE1	Errore: Email esistente
TC1_3	UE2,EE2,PA1	Successo: Utente registrato (con avatar)
TC1_4	UE2,EE2,PA2	Successo: Utente registrato (ma senza avatar)

3.2 Login

Parametro	Username
Esistente [UE]	• L'username non è presente nel database [Error] • L'username esiste nel database [UsernameEsiste]

Parametro	Password
Corretta [PC]	• L'hash della password fornita non coincide con l'hash salvato nel database per l'username fornito [Error] • L'hash della password fornita è uguale all'hash associato all'username fornito [PasswordCorretta]

Codice	Combinazione	Esito
TC2_1	UE1	Errore: Username non esistente
TC1_2	UE2,PC1	Errore: Password non corretta
TC1_3	UE2,PC2	Successo: Password corretta, utente autenticato.

3.3 Richiesta Pubblicazione Videogioco

Parametro	ID Autore
Autore Esistente [AE]	• L'ID dell'autore fornito come argomento non è associato ad alcun utente attualmente nel sistema. [Error] • L'ID dell'autore fornito come argomento è effettivamente associato ad un utente attualmente presente nel database. [AE_OK]

Codice	Combinazione	Esito
TC3_1	AE1	Errore: Utente autore non trovato
TC3_2	AE2	Successo: Richiesta pubblicazione generata

Capitolo 4

Test Case Specification

Identificatore test case	TC1
Locazione	tests/Feature/UtenzaRepositoryRegistrazioneTest
Feature da testare	La funzionalità da testare è il corretto salvataggio dei dati di un utente nel database relazionale.
Criteri di successo/fallimento	Il test passa se viene correttamente inserito nel database un nuovo utente con i dati forniti come input, e in seguito se l'inserimento di un utente con lo stesso username ed email lancia una eccezione.
Metodo di controllo	Il test viene avviato tramite il comando *./vendor/sail/bin artisan test*, il quale esegue il TestCase e PHPUnit.
Dati	I dati di input vengono forniti tramite un dataProvider che ritorna un iteratore di argomenti che verranno passati alla procedura di test, i quali consistono in due set: uno contenente username, email e password, il secondo contenente username, email, password ed un avatar mock.
Procedura di test	Il test viene avviato tramite un comando apposito e la libreria PHPUnit esegue il codice di setup, per poi controllando (tramite l'uso di asserzioni) se i risultati coincidono con l'oracolo proveniente dalle specifiche del requisito in analisi.
Requisiti Speciali	I container Docker devono essere in esecuzione, in particolare quello ospitante il database relazionale MySQL. Tutte le migrazioni devono essere state eseguite tramite il comando *./vendor/sail/bin artisan migrate*.

Identificatore test case	TC2
Locazione	tests/Feature/UtenzaRepositoryLoginTest
Feature da testare	La funzionalità da testare è la corretta autenticazione di un utente al sistema.
Criteri di successo/fallimento	Il test passa se l'utente viene autenticato correttamente all'account associato all'username fornito, se la password associata coincide con quella salvata nel database. Se l'username non è registrato o se la password fornita non è associata all'username fornito, allora viene individuata una failure.
Metodo di controllo	Il test viene avviato tramite il comando *./vendor/sail/bin artisan test*, il quale esegue il TestCase e PHPUnit.
Dati	I dati di input vengono forniti tramite un dataProvider che ritorna un iteratore di argomenti che verranno passati alla procedura di test, i quali consistono in tre set: uno contenente username non esistente e una password, il secondo contenente l'username dell'account iniziale del sistema ('admin') e una password non corretta ('errore') e il terzo contenente le credenziali corrette per l'account amministratore (username 'admin' e password 'root').
Procedura di test	Il test viene avviato tramite un comando apposito e la libreria PHPUnit esegue il codice di setup, per poi controllando (tramite l'uso di asserzioni) se i risultati coincidono con l'oracolo proveniente dalle specifiche del requisito in analisi.
Requisiti Speciali	I container Docker devono essere in esecuzione, in particolare quello ospitante il database relazionale MySQL. Tutte le migrazioni devono essere state eseguite tramite il comando *./vendor/sail/bin artisan migrate*. Il *seeding* iniziale del database deve essere effettuato tramite il comando *./vendor/sail/bin artisan db:seed* per permettere la creazione dell'account amministratore.

Identificatore test case	TC3
Locazione	tests/Feature/ UtenzaRepositoryCreateRichiestaPubblicazioneTest
Feature da testare	La funzionalità da testare è la corretta generazione di una richiesta di pubblicazione di un videogioco.
Criteri di successo/fallimento	Il test passa se .
Metodo di controllo	Il test viene avviato tramite il comando *./vendor/sail/bin artisan test*, il quale esegue il TestCase e PHPUnit.
Dati	I dati di input vengono forniti tramite un dataProvider che ritorna un iteratore di argomenti che verranno passati alla procedura di test, i quali consistono in tre set: uno contenente username non esistente e una password, il secondo contenente l'username dell'account iniziale del sistema ('admin') e una password non corretta ('errore') e il terzo contenente le credenziali corrette per l'account amministratore (username 'admin' e password 'root').
Procedura di test	Il test viene avviato tramite un comando apposito e la libreria PHPUnit esegue il codice di setup, per poi controllando (tramite l'uso di asserzioni) se i risultati coincidono con l'oracolo proveniente dalle specifiche del requisito in analisi.
Requisiti Speciali	I container Docker devono essere in esecuzione, in particolare quello ospitante il database relazionale MySQL. Tutte le migrazioni devono essere state eseguite tramite il comando *./vendor/sail/bin artisan migrate*. Il *seeding* iniziale del database deve essere effettuato tramite il comando *./vendor/sail/bin artisan db:seed* per permettere la creazione dell'account amministratore.