Test Plan Document

KeyMaster

Riferimento	
Versione	2.0
Data	5/12/2021
Destinatario	Prof.ssa F. Ferrucci
Presentato da	Andrea Santaniello, Mattia d'Argenio, Michele Corcione, Daniele Dello Russo
Approvato da	

Revision History

Data	Versione	Descrizione	Autori
18/12/2020	0.1	Prima stesura	Tutti
7/02/2022	0.2	Revisione	Tutti

Sommario

1. Introduzione

Durante la realizzazione di un software, ci si pone sempre l'obiettivo di ottenere un buon prodotto. Al fine di garantire che ciò avvenga, è necessario realizzare un sistema che possa essere considerato affidabile.
Nasce così la necessità di creare un prodotto che sia privo (o quasi) di errori prodotti durante la fase di implementazione per far si che il prodotto diventi uno strumento di cui l'utente finale si possa fidare.
A tal proposito è stato definito il seguente piano di test, il cui obiettivo è quello di analizzare e pianificare le attività di testing relative al sistema proposto.
Visto che è necessario garantire il corretto funzionamento del sistema, sono stati pensati input e casi di test specifici in modo da mettere alla prova le funzionalità offerte dal sistema stesso.
I risultati dei test, che verranno eseguiti successivamente, saranno fondamentali al fine di individuare le aree su cui bisogna intervenire per rimuovere i fault presenti all'interno del sistema.

2. Documenti correlati

2.1. Relazione con il documento di analisi

La progettazione dei casi di test avviene ignorando la struttura interna del sistema e operando solo sulle specifiche. Grazie agli scenari e agli use case prodotti nel documento di analisi otteniamo in maniera dettagliata le specifiche del sistema e per questo motivo è necessario far riferimento a questo documento.

2.2. Relazione con il System Design Document

Il sistema proposto, come si evince dal System Design Document è stato suddiviso in tre sottosistemi:

- GUI
- Business
- Storage

Il livello di business sarà il punto cruciale dei nostri test. Verranno pianificate attività di testing relative alle funzionalità specificate nei sottosistemi business, senza però trascurare quelle del sottosistema storage.

2.3. Relazione con l'Object Design Document

L'Object Design Document verrà utilizzato per la verifica dei contratti e dei comportamenti raffinati all'interno di tale documento.

3. Panoramica del sistema

L'obiettivo di KeyMaster è quello di fornire un sistema di license management (License-as-a-Service, LaaS) snello e facilmente implementabile in prodotti desktop e mobile.
Abbiamo individuato i seguenti sottosistemi per quanto riguarda il livello di business:

4. **Gestione utenza:** fornisce le funzionalità per effettuare l'autenticazione, gestire la propria area utente e richiedere il controllo via Api;
5. **Gestione software:** fornisce le funzionalità per gestire tutta la parte di personalizzazione software e modifica delle informazioni relative ad esso;
6. **Gestione lista software:** fornisce le funzionalità per modificare una lista di software quindi l'aggiunta di un nuovo software o la rimozione di un software già presente nel database.
7. **Gestione pk:** fornisce le funzionalità per generare un codice prodotto per l'attivazione di una nuova licenza;
8. **Gestione licenza:** fornisce le funzionalità per l'attivazione di nuove licenze e la modifica delle stesse.

4. Funzionalità da testare

Per quanto l'ideale sia testare ogni singolo componente del sistema, a causa del budget ridotto non è possibile pensare ad uno scenario del genere. Il team si occuperà pertanto di testare le funzionalità principali del sistema.

Le funzionalità che verranno testate sono riassunte nella seguente tabella:

Sottosistema	Funzionalità
Gestione utente	Registrazione di un nuovo utente alla piattaforma
	Login utente già registrato
	Controllo via API
Gestione lista software	Attivazione licenza mediante pk (utente finale)
	Attivazione licenza mediante assegnamento(utente sviluppatore)
	Inserimento di un nuovo software(utente sviluppatore)
Gestione licenza	Modifica durata licenza
Gestione pk	Generazione codici pk

Non verranno invece testate:

- Sicurezza
- Performance

5. Criteri di Pass/Failed

L'approccio utilizzato per testare il sistema sarà del tipo test to fail. L'obiettivo che ci poniamo è quello di individuare quanti più fault possibili durante le fasi di sviluppo in modo che, una volta rilasciato il software, esso contenga quanti meno fault possibili. L'approccio test to fail ci aiuta in questo senso in quanto spinge la nostra soluzione ai suoi limiti, evitando così che la conoscenza di ciò che abbiamo realizzato inganni la nostra fase di testing. Piuttosto invece incita a provare a riprovare una funzionalità fino a quando un errore non è stato individuato.

Pertanto, il test viene marcato come PASS se il comportamento osservato è diverso da quello atteso. In questo caso analizzeremo la causa dell'errore e verrà risolto. Il test viene marcato come FAILED nel caso in cui non vengano scovati errori nelle componenti.

6. Approccio

Le attività da svolgere per realizzare il testing sono tre. Nella fase iniziale ci occuperemo di individuare gli errori su una singola componente; nella fase successiva invece, testeremo le funzionalità nate dall'integrazione dei vari sottosistemi; infine, andremo a testare l'intero sistema per verificare che le caratteristiche richieste dal nostro committente siano rispettate o meno.

6.1. Testing di unità

Il testing di unità è necessario al fine di evidenziare gli errori delle singole componenti che compongono il nostro sistema. Il testing di unità si focalizza sostanzialmente sul comportamento di una componente e verrà svolto in modalità black-box. Questa scelta strutturerà il testing unitario in un'analisi Input/Ouput delle singole componenti andando ad astrarre la verifica della struttura interna. Il Testing dell'Api per l'autenticazione tra software e sito web verrà invece svolta in modalità white box (branch testing) per testare ogni singola condizione di uscita dai branch if. Per minimizzare i casi di test, gli input verranno divisi in classi di equivalenza e ogni componente avrà un singolo caso di test per ogni classe di equivalenza strutturata. In questa fase, quindi, si avrà particolare attenzione sulla suddivisione delle classi degli input così da poter verificare ogni componente su ogni possibile tipo di input del dominio. La gestione del caso di errore, ossia lo stato in cui il comportamento atteso non è equivalente al comportamento ottenuto, comporterà l'informare gli sviluppatori della presenza di un fault in modo tale da poterlo correggerlo tempestivamente per poi ripassare ad una verifica della correzione.

6.2. Testing di integrazione

Una volta rilevati e sistemati i fault per una singola componente, essa è pronta per essere integrata in un sottosistema più grande andando ad effettuare così il test di integrazione.
Per realizzarlo verrà utilizzata una strategia bottom-up.

6.3. Testing di sistema

Il testing di sistema concluderà la fase di test del prodotto ed il primo ciclo di sviluppo. Per questa tipologia di test, ci affidiamo all'utilizzo di un software ausiliario come Katalon al fine di osservare il comportamento del sistema.

7. Sospensione e ripresa

7.1. Criteri di sospensione

La fase di testing verrà sospesa nel momento in cui almeno il 10% dei casi di test riportano errori: in queste condizioni, il team deve provvedere a correggere i fault prima di procedere con l'implementazione o il testing di nuove funzionalità.

7.2. Criteri di ripresa

Visto che l'approccio per lo sviluppo del software è di tipo incrementale, la fase di testing potrebbe riprendere nel momento in cui verranno introdotti cambiamenti e/o nuove componenti. In questo caso, andranno testati i nuovi elementi introdotti e, tramite regression testing, anche quelli già precedentemente testati. Pertanto, faremo affidamento su un servizio che ci permetta di lavorare in un ambiente di Continous Integration

7.3. Criteri di terminazione

Il test si considera concluso nel momento in cui verrà raggiunta una branch coverage reputata sufficientemente alta, ricordando che il minimo è del 75%.

8. Materiale per il testing

L'esecuzione dei test necessita di un ambiente in cui siano installati Java SE 14 o successivi e MySQL.
Il testing viene effettuato utilizzando i framework JUnit, Mockito e Jacoco, molto affermati in ambiente Java, e il software Katalon.

Mentre JUnit viene utilizzato per il testing d'unità, Mockito viene utilizzato per mascherare le dipendenze tra le componenti. Jacoco verrà utilizzato per generare i report sui test. Katalon viene utilizzato per realizzare i test di sistema.
Come evidenziato prima, si lavorerà in un ambiente di continous integration: ciò è possibile grazie all'utilizzo di Jenkins.

9. Test cases

In questa sezione, per ogni sottosistema verranno mostrate le funzionalità che verranno testate. Ogni funzionalità avrà una tabella per ognuno dei suoi parametri che conterrà i vincoli affinché il valore dell'input sia valido. Infine, vi sarà una tabella che va ad indicare i test case individuati e i relativi esiti attesi dalle combinazioni dei vari vincoli: in caso di errore l'esito sarà uguale a ✗, altrimenti sarà uguale a ✔.

9.1. Gestione utente

9.1.1. Registrazione di un nuovo utente alla piattaforma

Tra le funzionalità da testare per il sottosistema utente è prevista la registrazione di un nuovo utente alla piattaforma. Questa funzionalità prevede che un nuovo utente compili un form di registrazione composto dai seguenti dati:

- Username: stringa con un numero di caratteri compreso tra 4 e 100. Non deve contenere spazi
- E-mail: stringa con un numero di caratteri compreso tra 5 e 150. Contiene una @
- Password: stringa con un numero di caratteri compreso tra 8 e 25. Non deve contenere spazi
- Conferma password

Parametro	**E-Mail**
Formato	^(([^<>()[\]\.,;:\s@\"]+(\.[^<>()[\]\.,;:\s@\"]+)*) | (\".+\"))@(([^<>()[\]\.,;:\s@\"]+\.)+[^<>()[\]\.,;:\s@\"]{2,})$/i
FE - Formato	1. Non rispetta il formato [error] 2. Rispetta il formato [property FE_OK]

Parametro	Username
Formato	^[a-zA-Z]+$
FU - Formato	1. Non rispetta il formato [error] 2. Rispetta il formato [if FE_OK] [property FU_OK]

Parametro	Password
Formato	^[a-zA-Z]+$
LP - Lunghezza	1.< 5 or > 30 [error] 2. >= 5 and <= 30 [property LP_OK]

Parametro	Conferma Password
VCP - Valore	1. Valore diverso da Password [error] 2. Valore uguale a Password[if FP_OK] [property VRP_OK]

Codice	Combinazione	Esito
TC_GU_1:1	FE1	X
TC_GU_1:2	FE2.FU1	X
TC_GU_1:3	FE2.FU2.LP1	X
TC_GU_1:4	FE2.FU2.LP2.VCP1	X
TC_GU_1:5	FE2.FU2.LP2.VCP2	✔

9.1.2. Login di un utente registrato

Un'altra funzionalità è quella del login. Questa funzionalità permette ad un utente registrato di accedere alla piattaforma con il proprio account. In input l'utente dovrà fornire:

- Username
- Password

Parametro	Username
EU - Esistenza	1. L'username non esiste nel db[error] 2. L'username esiste nel db [property EU_OK]

Parametro	Password
EP - Esistenza	1. La password non esiste nel db [error] 2 La possword esiste nel db [if EU_OK] [property LP_OK]

Codice	Combinazione	Esito
TC_GU_2:2	EU2.EP1	X
TC_GU_2:3	EU2.EP2	✔

9.1.3 Controllo via API

Tra le funzionalità più importanti del nostro sistema vi è quella del controllo via Api che permette all'utente di effettuare l'autenticazione.

L'utente non dovrà inserire nessun dato in input in quanto i controlli effettuati dall'Api avvengono nel codice in modalità white box. Tuttavia, nel documento TestCaseSpecification abbiamo riportato un esempio di dovrebbero essere i valori affinché i controlli effettuati dall'Api vadano a buon fine e ritornino ciò che l'utente si aspetta. I parametri su cui vengono effettuati i controlli sono:

- Apikey: stringa identificativa per un determinato software che viene automaticamente generato dal sistema. Deve essere lunga 32 caratteri e non vi è nessun controllo sul formato essendo generata dal sistema
- Utente: deve essere esistente. Qualora non fosse così il metodo ritornerà il valore -1
- HwID: intero identificativo della macchina a cui è collegata la licenza. Può anche essere vuoto per cui non verrà fatto nessun controllo su di esso. Nel caso in cui sia vuoto il sistema setterà automaticamente il sistema.
- UserID: intero identificativo dell'utente
- OwLicID: intero identificativo di una data licenza posseduta dall'utente. La licenza posseduta deve essere attiva per tanto il valore del campo "isActive" deve essere settato a 1

- IsActive: intero che fa riferimento al campo "OwLicID" e sta a significare che quella determinata licenza posseduta è attiva, quando il valore del campo è 1
- LicID: intero identificativo di una licenza connessa all'applicazione con quell'Apikey

Parametro	ApiKey
Formato	^[a-zA-Z]+$
LA - Lunghezza	1. != 32 [error] 2. = 32 [property LA_OK]

Parametro	Utente
Formato	^[a-zA-Z]+$
LU - Login	1.L'utente non è esistente [error] 2 L'utente è loggato correttamente [if LA_OK][property LU_OK]

Parametro	UserID
LUID - Lunghezza	1. = -1 [error] 2. != -1 [if FU_OK] [property FP_OK]

Parametro	OwLicID
FOL - Formato	1. Valore diverso da LicID [error] 2. Valore uguale a LicID [if LUID_OK] [property FP_OK]

Parametro	isActive
FA - Formato	1. != 1 [error] 2. = 1 [if FOL_OK] [property FP_OK]

Parametro	LicID
FL - Formato	1. Valore diverso da OwLicID [error] 2.Valore uguale a OwLicID [if LUID_OK] [property FP_OK]

Codice	Combinazione	Esito

TC_GU_3:1	LA1	X
TC_GU_3:2	LA2.LU1	X
TC_GU_3:3	LA2.LU2.LUID1	X
TC_GU_3:4	LA2.LU2. LUID2.FOL2.FL1	X
TC_GU_3:5	LA2.LU2.LUID2.FOL2.FL2.FA1	X
TC_GU_3:6	LA2.LU2.LUID2.FOL2.FL2.FA2	✔

2.1. Gestione lista software

2.1.1. Attivazione licenza mediante pk

Tra le funzionalità da testare per il sottosistema Gestione lista software vi è l'attivazione di una licenza software mediante pk. Questa funzionalità prevede che un utente finale possa attivare un software mediante un pk. In input l'utente dovrà fornire i seguenti dati:

- Product key: una stringa di 16 cifre che contiene sia numeri che lettere
- Se esiste un product key nel database generato precedentemente e se non è stato già riscattato

Parametro	Product key
EP – Esistenza	1. Il pk non esiste nel db [error] 2. Il pk esiste nel db [property LP_OK]
RP – Riscattato	1. Il pk è stato già riscattato [error] 2. Il pk non è stato ancora riscattato[if EP_OK] [property RP_OK]

Codice	Combinazione	Esito
TC_GLS_1:1	EP1	X
TC_GLS_1:2	EP2.RP1	X
TC_GLS_1:3	EP2.RP2	✔

2.1.2. Attivazione licenza mediante assegnamento

Un'altra importante funzionalità da testare è quella dell'attivazione di una licenza software mediante assegnamento. Qui l'utente sviluppatore dovrà specificare a quale utente vorrà assegnare la licenza quindi verrà indicato l'username. .

Parametro	Username
Formato	^[a-zA-Z]+$
EU - Esistenza	1. L'username non è presente sul db [error] 2. L'username non è presente sul db [if LU_OK] [property FU_OK]

Codice	Combinazione	Esito
TC_GLS_2:1	LU1	X
TC_GLS_2:2	LU2	✔

9.2.3 Inserimento di un nuovo software

Un'altra importante funzionalità da testare è quella dell'inserimento di un nuovo software nella propria lista di software. Qui l'utente sviluppatore dovrà specificare il nome del software che vuole aggiungere, una descrizione del software e il sito web di riferimento.

Parametro	Nome
Formato	^([A-Z ,])+$
LU - Lunghezza	1. <1 or > 200 [error] 2. >= 1 and <= 200 [property LU_OK]

Parametro	Descrizione
LD - Lunghezza	1. <= 0 [error] 2. > 0 [property LD_OK]

Parametro	SitoWeb
FS - Formato	1. Non rispetta il formato [error] 2. Rispetta il formato [if LD_OK] [property FS_OK]

Codice	Combinazione	Esito
TC_GLS_3:1	LU1	X
TC_GLS_3:2	LU2.LD1	X
TC_GLS_3:3	LU2.LD2.FS1	X
TC_GLS_3:4	LU2.LD2.FS2	✔

2.2. Gestione licenza

2.2.1. Modifica durata licenza

Tra le funzionalità da testare per il sottosistema Gestione licenza vi è l'operazione di modifica durata licenza. Questa funzionalità prevede che un utente sviluppatore possa modificare la durata di una licenza, dovrà per questo inserire il nuovo valore della durata.

Parametro	Durata
VD - Valore	1. <= 0 [error] 2. > 0 [property VD_OK]

Codice	Combinazione	Esito
TC_GL_1:1	VD1	X
TC_GL_1:2	VD2	✔

2.3. Generazione product key

Tra le attività del sottosistema Gestione product key vi è l'operazione di generazione codici product key. L'utente sviluppatore dovrà inserire il numero di codici che vuole generare per tanto deve essere un valore > 0.

Parametro	NumCodice
VNC - Valore	1. <= 0 [error] 2. >= 0 and <= 500 [property VNC_OK]

Codice	Combinazione	Esito
TC_GL_2:1	VNC1	X
TC_GL_2:2	VNC2	✔

3. Riferimenti ad altri documenti di test

Documento	Riferimento
Test Case Specification	Combinazioni di input fornite al sistema