Laurea Triennale in informatica-Università di Salerno
Corso di *Ingegneria del Software*- Prof.ssa F.Ferrucci

TP Test Plan

NewDM

Riferimento	
Versione	1.0
Data	3/02/2020
Destinatario	Prof.ssa F. Ferrucci
Presentato da	Cirillo Franco Cirillo Luigi Fusco Ciro Aiello Vincenzo
Approvato da	

Revision History

Data	Versione	Cambiamenti	Autori
17/01/2020	1.0	Prima stesura	Cirillo Franco

Sommario

1. Introduzione

Al fine di consentire ad una catena di supermercati di gestire al meglio i servizi offerti alla clientela, abbiamo messo in campo un sistema in grado di offrire supporto a tutti gli addetti del punto vendita: cassieri, magazzinieri e addetti assistenza.

Nell'affrontare alcuni aspetti fondamentali del sistema NewDM ci siamo posti delle domande:

Come ottenere un buon prodotto? Come essere sicuri che sia un buon prodotto?

Nasce così la necessità di rilevare eventuali errori prodotti durante la fase di implementazione per evitare che essi si presentino nel momento in cui sistema verrà utilizzato dall'utente finale.

A tal proposito definiamo un piano di test il cui obiettivo principale è quello di analizzare e gestire lo sviluppo delle attività di testing relative al sistema proposto.

In questa fase occorre verificare il corretto funzionamento del sistema sotto determinate condizioni.

Abbiamo pensato a opportuni casi e dati di input specifici in grado di mettere alla prova ogni singola funzionalità e caratteristica offerta dalla piattaforma.

I risultati dei test che verranno eseguiti saranno il punto cruciale nell'analisi delle failure e delle loro cause (fault) per individuare dove bisognerà intervenire per correggere gli errori o apportare modifiche per il miglioramento dei vari sottosistemi.

Lo scopo è quindi stabilire la verità sulla corrispondenza tra comportamento atteso e comportamento osservato da NewDM.

2. Documenti correlati

2.1 Relazione con il Requirement Analysis Document

La progettazione dei casi di test avviene prescindendo dalla conoscenza della struttura interna del prodotto ed operando solo sulle specifiche.

Per questo motivo facciamo riferimento al contenuto del documento di analisi che descrive dettagliatamente le funzionalità del sistema attraverso scenari e use case.

2.2 Relazione con il System Design Document

Nel system design document abbiamo definito la suddivisione in sottosistemi relativamente al prodotto che intendiamo presentare.

Il sistema è suddiviso in tre livelli logici: presentazione, business e persistenza. Ogni livello è composta da vari sottosistemi.

In questa fase è importante focalizzare la nostra attenzione sul layer di business. Infatti, pianificheremo le attività di testing relative alle funzionalità garantite nei sottosistemi specificati all'interno nel System Design Document relativamente al livello business.

2.3 Relazione con l'Object Design Document

Nel documento di Object Design sono state definite le classi che compongono il sistema e le loro mansioni. Faremo riferimento ad esse nel corso del documento per associare i test al codice prodotto.

3. Panoramica del sistema

NewDM è una Piattaforma desktop che fornisce un'interfaccia per le sue funzionalità: la gestione dell'utenza, dei ticket, dell'emissione scontrini e del magazzino.

Il sistema che proponiamo prevede tre attori principali:

- Cassiere: potrà scansionare i prodotti selezionati dal cliente, calcolare il totale della spesa e stampare lo scontrino dopo aver ricevuto il pagamento.
- Magazziniere: potrà inserire stock di prodotti appena arrivati e potrà registrare anche un nuovo prodotto. Potrebbe altresì modificare il prezzo di un prodotto oppure fare una richiesta di rifornimento facendosi consigliare da una previsione della quantità di prodotto da acquistare.
- Addetto assistenza: potrà aprire un nuovo ticket per l'assistenza relativa ad un prodotto, validarne degli aspetti ed eventualmente salvarlo.

Nel System Design abbiamo definito l'architettura della piattaforma che si articola in tre layer: presentazione, business e persistenza. I sottosistemi che compongono i vari livelli logici collaborano tra loro cercando però di garantire il più possibile basso accoppiamento ed alta coesione.
Il livello di business è composto da cinque sottosistemi

- Utenza: definisce l'utente (cassiere, magazziniere, addetto assistenza) ed offre i relativi servizi di autenticazione.
- Cassa: modella tutto ciò che riguarda il processo di acquisto dei prodotti ed emissione scontrino
- Assistenza: modella il processo di gestione ticket e tutte le sue operazioni
- Inventario: modella i prodotti nel magazzino e le relative operazioni
- Fornitura: modella il processo di richiesta prodotto

4. Funzionalità da testare

La fase di testing avrà come obiettivo quello di testare interamente la comunicazione tra la piattaforma e il database, così come l'implementazione della logica di business e dei servizi offerti dai sottosistemi del livello centrale.

Il testing funzionale riguarderà nel dettaglio le funzionalità di seguito elencate (in base al sottosistema che le realizza):

- **Cassa**
  - Aggiungi somma versata
  - Inserisci prodotto
- **Inventario**
  - Crea nuovo prodotto
  - Modifica quantità prodotto
- **Assistenza**
  - Crea ticket
- **Utenza**
  - Login

5. Criteri Pass/Failed

Abbiamo determinato un insieme di input possibili che possano aiutarci a trovare errori nel sistema. Pertanto, il test ha successo se il comportamento osservato è diverso dal comportamento specificato nei requisiti funzionali. Ciò significa che raggiungiamo gli obiettivi che ci siamo posti durante questa fase se il test individuerà dei failure nel sistema.

In tal caso analizzeremo i sottosistemi coinvolti nell'errore, ed itereremo la fase di testing per verificare che le modifiche apportate agli stessi non abbiamo avuto impatti negativi su altre componenti del sistema. Il testing fallirà se gli non saranno trovati errori nelle componenti.

Presenza di errori : Pass, assenza di errori: Failed.

6. Approccio

La fase di testing si compone di tre attività: una prima fase si occuperà di trovare errori in una singola componente; la seconda fase, invece, avrà come compito quello di testare le funzionalità nate dall'integrazione dei vari sottosistemi e per ultimo andremo a testare l'intero sistema assemblato al fine di verificare soprattutto che esso soddisfi i desideri del cliente.

Di seguito verranno descritte brevemente le strategie individuate per effettuare il test di unità, d'integrazione, di sistema e di usabilità.

6.1 Testing di unità

Durante la fase di testing è necessario testare singole componenti al fine di evidenziarne gli errori. Il testing di unità infatti, si focalizza sul comportamento di una componente permettendo di eseguire testing in modalità black-box o white-box.

Le nostre componenti saranno testate secondo il metodo white-box. Infatti, durante questa fase poseremo la nostra sulla struttura del codice che realizza le funzionalità fornite dalla componente al fine di individuare errori sia di logica che di implementazione.

Utilizzeremo il metodo branch coverage, questo perché è una metrica facile da comprendere, si misura senza difficoltà, è oggettiva e imparziale, è universale (applicabile a tutti i paradigmi di programmazione).

6.2 Testing di integrazione

Una volta che sono stati rilevati i bug per una singola componente e riparati, le componenti sono pronte per essere integrate in sottosistemi più grandi. Pertanto, dopo aver testato singolarmente le componenti del sistema, possiamo procedere a testarne le integrazioni.

Per effettuare l'integration testing abbiamo pensato di utilizzare la strategia bottom-up: ciò ci permette di garantire la presenza di fondamenta solide alla base del sistema ma richiede di mettere in campo test driver per simulare le componenti dei layer più in alto che non sono stati ancora integrati.

6.3 Testing di sistema

La verifica sulle funzionalità del sistema avviene testando i possibili input degli utenti. La riduzione dei casi di test è attuata tramite l'adozione del category partition.

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 Studio al fine di osservare il comportamento del sistema in presenza di combinazioni di input utente non ammesse.

6.4 Testing di usabilità

Dopo aver realizzato la prima versione del sistema abbiamo deciso di proporla ad utenti reali.

Durante la fase di valutazione dell'usabilità ci soffermiamo, oltre che sugli aspetti funzionali del sistema, anche sul grado di intuizione delle interfacce utente, sulle azioni che si devono compiere

per utilizzare una determinata funzionalità, sull'esperienza che oggetti ed icone dell'interfaccia permettono di vivere all'utente ed il senso che vi suggeriscono.

Adotteremo una tecnica che consiste nel far utilizzare il sistema ad un determinato numero di persone e poi gli verrà somministrato un sondaggio per ricavare il grado di facilità nell'utilizzo del sistema. Le domande del form saranno susseguenti all'assegnazione di alcuni task specificate in esse.

Verrà utilizzato lo strumento Google moduli per il sondaggio e sarà fatta una media tra tutti i risultati per ottenere il grado di usabilità complessivo.

7. Sospensione e ripresa

Tenuto conto delle risorse necessarie impiegate durante la fase di testing, abbiamo stabilito dei criteri in base ai quali le attività di test saranno sospese o riprese.

7.1 Criteri di sospensione

Il test è sospeso se 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

Abbiamo previsto delle modifiche future al sistema dopo il rilascio. Pertanto sarà necessario, dopo aver introdotto cambiamenti, testare le nuove componenti: se esse introducono dei fault che impattano sulle componenti già esistenti, allora verranno testate nuovamente anche quest'ultime.

I test case, quindi, verranno ancora una volta somministrati al sistema per assicurarsi di aver risolto i problemi scorti precedentemente. In questo senso evidenziamo la nostra volontà di effettuare test di regressione ad ogni modifica apportata al sistema, pertanto facciamo affidamento su un servizio che ci permette di lavorare in un ambiente di Continous Integration.

7.3 Criteri di terminazione

Il test si considera terminato quando la totalità dei casi di test somministrati al sistema riporta esito negativo. Per scelte progettuali, la suddetta condizione sussiste solo se il 75% dei branch sviluppati viene ricoperto in questa fase.

8. Materiale per il testing

L'esecuzione dei test necessita di un server correttamente configurato su cui siano installati Java e MySQL. La configurazione deve avvenire come da manuale d'installazione.

Il testing è condotto utilizzando alcuni dei framework più famosi ed efficaci in ambienti Java: JUnit e Katalon e Jacoco per le metriche.

Come detto al punto precedente, i test sono eseguiti ad ogni modifica apportata al sistema, in un ambiente di Continous Integration: ciò è possibile grazie all'utilizzo di Travis CI e Maven.

9. Test cases

Per ogni sottosistema mostriamo le funzionalità che andremo a testare. Inoltre, associamo ad ogni funzionalità una tabella in cui, per ogni parametro, vengono definite le relative categorie, insieme alle possibili scelte.

Nella prima riga troviamo Parametro che sta ad indicare il valore di input e formato che viene indicato solitamente attraverso l'espressione regolare per indicare il pattern che il parametro deve seguire. Nella prima colonna, invece, riportiamo le categorie selezionate, al loro fianco le scelte.

9.1 Assistenza

Per il sottosistema assistenza abbiamo previsto di testarne la funzionalità di creazione nuovo ticket.

Tale funzionalità prevede la possibilità da parte dell'addetto assistenza di compilare un modulo in cui risulta necessario l'inserimento di:

- Nome: stringa con un numero di caratteri compreso tra i 2 ed i 255
- Cognome: stringa con un numero di caratteri compreso tra i 2 ed i 255
- Indirizzo (via, civico, città, CAP): stringa di almeno 2 caratteri fino a 255
- Telefono: stringa contenente 10 o 11 numeri
- Nome prodotto: stringa con un numero di caratteri compreso tra i 2 ed i 255
- Tipo prodotto: stringa con un numero di caratteri compreso tra i 2 ed i 255
- Codice prodotto: stringa di 13 cifre esistente nel Database
- Numero di serie: stringa con un numero di caratteri compreso tra i 2 ed i 255
- Data scontrino antecedente alla data odierna ma non precedente di più di 2 anni.
    - Giorno: intero compreso tra 1 e 31
    - Mese: intero compreso tra 1 e 12
    - Anno: intero compreso tra l'anno corrente -2 e l'anno corrente

- Numero scontrino: intero maggiore di 0 e esistente nel Database per quella data

Parametro	Nome e cognome
LN - Lunghezza	1. < 2 or > 255 [error] 2. ≥ 2 and ≤ 255 [property LN_OK]

Parametro	Indirizzo
LI - Lunghezza	1. < 2 or > 255 [error] 2. ≥ 2 and ≤ 255 [property LI_OK]

Parametro	Telefono
Formato	[0-9]{10,11}
LT - Lunghezza	1. < 10 or > 11 [error] 2. ≥ 10 and ≤ 11 [property LT_OK]
FT - Formato	1. Non rispetta il formato [error] 2. Rispetta il formato [if LT_OK] [property FT_OK]

Parametro	Nome prodotto
LNP - Lunghezza	1. < 2 or > 255 [error] 2. ≥ 2 and ≤ 255 [property LNP_OK]

Parametro	Tipo prodotto
LTP - Lunghezza	1. < 2 or > 255 [error] 2. ≥ 2 and ≤ 255 [property LTP_OK]

Parametro	Codice prodotto
Formato	[0-9]{13}
LCP - Lunghezza	1. < 13 or > 13 [error] 2. =13 [property LT_OK]
FCP - Formato	1. Non rispetta il formato [error] 2. Rispetta il formato [if LCP_OK] [property FCP_OK]

ECP - Esistenza	1. Non esiste nel database [error] 2. Esiste nel database [if LCP_OK] [if FCP_OK] [property ECP_OK]

Parametro	Numero di serie
LNS - Lunghezza	1. < 2 or > 255 [error] 2. ≥ 2 and ≤ 255 [property LNS_OK]

Parametro	Data scontrino
VDS - Valore	1. Non valida or segue la data corrente or precede di 2 anni la data odierna [error] 2. Valida and compresa tra la data corrente e 2 anni prima [property VDS_OK]

Parametro	Numero scontrino
Formato	[0-9]
FNS - Formato	1. Non rispetta il formato [error] 2. Rispetta il formato [property FNS_OK]
VNS - Valore	1. ≤ 0 [error] 2. >0 [if FNS_OK] [property VNS_OK]
ENS - Esistenza	1. Non esiste nel database [error] 2. Esiste nel database[if FNS_OK] [if VNS_OK] [property ENS_OK]

Parametro	Codice fiscale
Formato	^[A-Z]{6}[0-9]{2}[A-Z][0-9]{2}[A-Z][0-9]{3}[A-Z]$
FC - Formato	1. Non rispetta il formato [error] 2. Rispetta il formato[property FC_OK]

Codice	Combinazione	Esito
TC_A_1:1	LN1	X

TC_A_1:2	LN2.LI1	✗
TC_A_1:3	LN2.LI2.LT1	✗
TC_A_1:4	LN2.LI2.LT2.FT1	✗
TC_A_1:5	LN2.LI2.LT2.FT2.LNP1	✗
TC_A_1:6	LN2.LI2.LT2.FT2.LNP2.LTP1	✗
TC_A_1:7	LN2.LI2.LT2.FT2.LNP2.LTP2.LCP1	✗
TC_A_1:8	LN2.LI2.LT2.FT2.LNP2.LTP2.LCP2.FCP1	✗
TC_A_1:9	LN2.LI2.LT2.FT2.LNP2.LTP2.LCP2.FCP2.ECP1	✗
TC_A_1:10	LN2.LI2.LT2.FT2.LNP2.LTP2.LCP2.FCP2.ECP2.LNS1	✗
TC_A_1:11	LN2.LI2.LT2.FT2.LNP2.LTP2.LCP2.FCP2.ECP2.LNS2.VDS1	✗
TC_A_1:12	LN2.LI2.LT2.FT2.LNP2.LTP2.LCP2.FCP2.ECP2.LNS2.VDS2.FNS1	✗
TC_A_1:13	LN2.LI2.LT2.FT2.LNP2.LTP2.LCP2.FCP2.ECP2.LNS2.VDS2.FNS2.VNS1	✗
TC_A_1:14	LN2.LI2.LT2.FT2.LNP2.LTP2.LCP2.FCP2.ECP2.LNS2.VDS2. FNS2.VNS2.ENS1	✗
TC_A_1:15	LN2.LI2.LT2.FT2.LNP2.LTP2.LCP2.FCP2.ECP2.LNS2.VDS2. FNS2.VNS2.ENS2.FC1	✗
TC_A_1:16	LN2.LI2.LT2.FT2.LNP2.LTP2.LCP2.FCP2.ECP2.LNS2.VDS2. FNS2.VNS2.ENS2.FC2	✔

9.2 Inventario

Per il sottosistema inventario abbiamo previsto di testarne la funzionalità di creazione nuovo prodotto, modifica quantità prodotto.

La prima funzionalità prevede la possibilità da parte del magazziniere di compilare un modulo in cui risulta necessario l'inserimento di:

- Nome prodotto: stringa con un numero di caratteri compreso tra i 2 ed i 255
- Codice prodotto: stringa di 13 cifre
- Tipologia: stringa con un numero di caratteri compreso tra i 2 ed i 255
- Scadenza(testuale): stringa con un numero di caratteri compreso tra i 2 ed i 255
- Dimensione confezione(testuale): stringa con un numero di caratteri compreso tra i 2 ed i 255
- Quantità: intero maggiore di 0
- Prezzo: numero decimale maggiore di 0, con fino a 2 cifre decimali

Parametro	Nome prodotto

LNP - Lunghezza	1. < 2 or > 255 [error] 2. ≥ 2 and ≤ 255 [property LNP_OK]

Parametro	Quantità
Formato	[0-9]
FQ - Formato	1. Non rispetta il formato [error] 2. Rispetta il formato [property FQ_OK]
VNS - Valore	1. ≤ 0 [error] 2. >0 [if FQ_OK] [property VQ_OK]

Parametro	Codice prodotto
Formato	[0-9]{13}
LCP - Lunghezza	1. < 13 or > 13 [error] 2. =13 [property LCP_OK]
FCP - Formato	1. Non rispetta il formato [error] 2. Rispetta il formato [if LCP_OK] [property FCP_OK]

Parametro	Prezzo
Formato	^[0-9]+([.,][0-9]{1,2})?$
FP - Formato	1. Non rispetta il formato [error] 2. Rispetta il formato [property FP_OK]
VP - Valore	1. ≤ 0 [error] 2. >0 [if FP_OK] [property VP_OK]

Parametro	Tipologia
LT - Lunghezza	1. < 2 or > 255 [error] 2. ≥ 2 and ≤ 255 [property LT_OK]

Codice	Combinazione	Esito
TC_I_1:1	LNP1	✗
TC_I_1:2	LNP2.FQ1	✗
TC_I_1:3	LNP2.FQ2.VNS1	✗
TC_I_1:4	LNP2.FQ2.VNS2.LCP1	✗
TC_I_1:5	LNP2.FQ2.VNS2.LCP2.FCP1	✗
TC_I_1:6	LNP2.FQ2.VNS2.LCP2.FCP2.FP1	✗
TC_I_1:7	LNP2.FQ2.VNS2.LCP2.FCP2.FP2.VP1	✗
TC_I_1:8	LNP2.FQ2.VNS2.LCP2.FCP2.FP2.VP2.LT1	✗
TC_I_1:9	LNP2.FQ2.VNS2.LCP2.FCP2.FP2.VP2.LT2	✔

La seconda funzionalità prevede la possibilità da parte del magazziniere di compilare un modulo in cui risulta necessario l'inserimento di:

- Codice prodotto: stringa di 13 cifre
- Quantità: intero maggiore di 0

Parametro	Codice prodotto
Formato	[0-9]{13}
LCP - Lunghezza	1. < 13 or > 13 [error] 2. =13 [property LCP_OK]
FCP - Formato	1. Non rispetta il formato [error] 2. Rispetta il formato [if LCP_OK] [property FCP_OK]
ECP - Esistenza	1. Non esiste nel database [error] 2. Esiste nel database [if LCP_OK] [if FCP_OK] [property ECP_OK]

Parametro	Quantità
Formato	[0-9]
FQ - Formato	1. Non rispetta il formato [error] 2. Rispetta il formato [property FQ_OK]
VNS - Valore	1. ≤ 0 [error] 2. >0 [if FQ_OK] [property VQ_OK]

Codice	Combinazione	Esito
TC_I_2:1	LCP1	✗
TC_I_2:2	LCP2.FCP1	✗
TC_I_2:3	LCP2.FCP2.ECP1	✗
TC_I_2:4	LCP2.FCP2.ECP2.FQ1	✗
TC_I_2:5	LCP2.FCP2.ECP2.FQ2.VNS1	✗
TC_I_2:6	LCP2.FCP2.ECP2.FQ2.VNS2	✔

9.3 Cassa

Per il sottosistema cassa abbiamo previsto di testarne la funzionalità di aggiunta somma versata e inserisci prodotto.

La prima funzionalità prevede la possibilità da parte del cassiere di compilare un modulo in cui risulta necessario l'inserimento di:

- Somma versata: numero decimale con fino a 2 cifre decimali, deve essere maggiore dell'importo da versare.

Parametro	Somma versata
Formato	^[0-9]+([.,][0-9]{1,2})?$
FSV - Formato	1. Non rispetta il formato [error] 2. Rispetta il formato [property FSV_OK]
VSV - Valore	1. < somma da versare [error] 2. ≥ somma da versare [if FSV_OK] [property VSV_OK]

Codice	Combinazione	Esito
TC_C_1:1	FSV1	✗
TC_C_1:2	FSV2.VSV1	✗
TC_C_1:3	FSV2.VSV2	✔

La seconda funzionalità prevede la possibilità da parte del cassiere di compilare un modulo in cui risulta necessario l'inserimento di:

- Codice prodotto: stringa di interi esistente nel db

Parametro	Codice prodotto
Formato	[0-9]
FCP - Formato	1. Non rispetta il formato [error] 2. Rispetta il formato [property FCP_OK]
ECP - Esistenza	1. Non esiste nel database [error] 2. Esiste nel database[if FCP_OK] [property ECP_OK]

Codice	Combinazione	Esito
TC_C_2:1	FCP1	✗
TC_C_2:2	FCP2.ECP1	✗
TC_C_2:3	FCP2.ECP2	✔

9.4 Utenza

Per il sottosistema utenza abbiamo previsto di testarne la funzionalità di login.

La suddetta funzionalità prevede la possibilità da parte del dipendente di compilare un modulo in cui risulta necessario l'inserimento di:

- Codice utente: stringa esistente nel db
- Password: stringa corrispondente al nome utente nel db

Parametro	Codice utente
ECU - Esistenza	1. Non esiste nel database [error] 2. Esiste nel database [property ECU_OK]

Parametro	Password
EP - Esistenza	1. Non esiste nel database [error] 2. Esiste nel database [property EP_OK]

Codice	Combinazione	Esito
TC_U_1:1	ECU1	✗
TC_U_1:2	ECU2.EP1	✗
TC_U_1:3	ECU2.EP2	✔

10. Riferimenti ad altri documenti di test

Le combinazioni di input che verranno somministrate al sistema sono definite nel documento di Test Case Specification, mentre sarà nel Test Execution Report che indicheremo i risultati del testing funzionale condotto tramite Katalon Studio.

Eventuali errori rilevati verranno riportati nel Test Incident Report, mentre il riassunto dei riscontri ottenuti in questa fase verrà proposto tramite il documento di Test Summary Report.