UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea in Informatica Magistrale

Ingegneria, gestione ed evoluzione del software

M2fas_management

Documento di analisi di manutenzione

Versione 1.0

Docente

Prof.ssa Andrea De Lucia

Studente

Maria Cristina Ricciardi
Matr.: 0522500406

Anno Accademico 2020/2021

Cronologia revisioni

Data	Versione	Descrizione	Autore
23/10/2022	0.1	Prima stesura	M.C.Ricciardi
11/03/2023	0.2	Test di unità	M.C.Ricciardi

Sommario

1. Scopo del documento

In questo documento è riportata l'analisi svolta durante il processo di manutenzione dell'applicazione m2fas management con l'obiettivo di aggiungere un nuovo sottosistema per adattare dell'applicazione sul web.

2. Panoramica del sistema attuale

M2fas management è un software per la gestione di un magazzino per un negozio di abbigliamento. E' progettato per essere utilizzato da due attori principali: l'amministratore del sistema e un operatore addetto alla cassa. Ha un database centralizzato, utilizzato per contenere dati persistenti quali prodotti, fornitori e bilancio del negozio.

Dai documenti forniti insieme al codice sorgente, il software funziona correttamente su sistemi con supporto alla JVM, con interfaccia grafica diversa per l'operatore e l'amministratore.

L'architettura scelta è di tipo repository con tre sottosistemi indipendenti, il repository che funge da database utilizzato dagli altri due sottosistemi per la memorizzazione dei dati: operatore e amministratore.

Il sistema è implementato in Java 2 con l'uso di un database MySQL.

Si è deciso di intervenire su questo sistema con l'obiettivo di aggiungere un nuovo sottosistema per l'accesso dell'applicazione attraverso il web, utilizzando le nuove tecnologie in uso attualmente, in maniera tale da aggiungere nuove funzionalità, come visualizzare la merce del negozio a più utenti possibili.

2.1 Attori e funzionalità

Il sistema prevede due attori principali con le seguenti funzionalità:

- Amministratore
  - Gestione Fornitori
  - Gestione Magazzino
  - Bilancio
- Operatore
  - Gestione cassa

2.2 Design

Dalla documentazione del sistema attuale, possiamo vedere che il sistema è stato concepito per uno stile architetturale del tipo repository composti da tre strati indipendenti:

- Interface layer

  Contiene le componenti delle interfacce grafiche che l'utente usa per poter accedere ai dati

- Logic layer

Sottosistema delle componenti di logica e delle entità del sistema

- Repository layer

Sistema di gestione dei dati, per il recupero e aggiornamento dei dati

2.3 Testing

In questa versione del sistema non è presente una suite test, quindi è stata creata una nuova del sistema corrente.

3. Analisi del sistema

La manutenzione che si intende effettuare sul sistema è di adattare tale sistema sul web, riutilizzando i componenti core e la logica applicativa già progettati, al nuovo modulo web. Inoltre l'intenzione è di connettere il nuovo sistema alla database attuale. La componente grafica o interfaccia grafica verrà totalmente riprogrammata in base alla tecnologia utilizzata per il web, conservando il look and feel invariato.

3.1 Le funzionalità da testare

Durante la fase di analisi sono state prese in considerazione diverse funzionalità, considerate essenziali del sistema m2fas:

- Gestione prodotto

Permette all'amministratore di poter aggiornare, inserire, eliminare e visualizzare i prodotti disponibili nel catalogo del negozio.

- Gestione bilancio

Si tiene traccia delle spese in entrata e uscita

- Gestione spese

Inserimento e eliminazione delle spese relative al negozio e ai fornitori.

- Autenticazione

Sistema di login per l'amministratore e l'operatore

Tali funzionalità saranno riutilizzate dal nuovo modulo con l'intento di migliorare la gestione di autenticazione per la sicurezza degli accessi sia da parte dell'amministratore che dell'operatore.

3.2 Test case

Sono state testate solo quelle classi che hanno un accesso diretto alla logica applicativa e alla base di dati dell'applicazione, evitando quelle classi wrapper e graphic. Di seguito elenchiamo un resoconto dei maggiori errori del sistema, risultati dai test eseguiti sul sistema.

Resoconto:

1. **TC 3.00** La gestione entrate hanno un bug nell'inserimento delle entrate nel database
2. **TC 0.06** Non è possibile aggiornare solo la quantità di un dato prodotto
3. **TC 0.10** Non viene visualizzato nessun messaggio di errore quando si elimina un prodotto non presente nel database e il test viene superato.
4. **TC 1.04** Non viene visualizzato nessun messaggio di errore quando si elimina un fornitore non presente nel database e il test viene superato con successo.
5. **TC 1.08** Non viene visualizzato nessun messaggio di errore quando si aggiorna dati di pagamento a un fornitore con ID non presente nel database e il test viene superato con successo.
6. **TC 1.09** Non è possibile aggiornare solo il campo debito di un fornitore presente nel database.
7. **TC 1.10** Non è possibile aggiornare solo il campo pagato di un fornitore presente nel database.
8. **TC 2.02** Il test è superato e dati inseriti nella tabella uscite nel database, nonostante la data inserita non ha un formato idoneo.
9. **TC 2.05** Il test è superato nonostante la data inserita non ha un formato idoneo.
10. **TC 3.02** Il test è superato nonostante la data inserita non ha un formato idoneo.

Product Management

TC 0.01

Test case ID:	TC 0.01
Test case description:	getProduct
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente registrato ottiene i dati del prodotto inserendo nel form **ID = 10** del prodotto.
Result:	PASSED
Anomalies:	

TC 0.02

Test case ID:	TC 0.02
Test case description:	getProductInputError
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente inserisce nel form **ID = ok** del prodotto.
Result:	FAILED
Anomalies:	Non viene segnalato nessun messaggio di errore nonostante il valore dell'ID non è presente nel database.

TC 0.03

Test case ID:	TC 0.03
Test case description:	insertProduct
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente inserisce un nuovo prodotto indicando marca, ID = 10, modello, categoria, prezzo, quantità.
Result:	PASSED

Anomalies:	Non viene segnalato nessun messaggio di errore.

TC 0.04

Test case ID:	TC 0.04
Test case description:	insertProductWithIDError
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente inserisce un prodotto con ID = 10 già inserito nel database con gli stessi dati.
Result:	FAILED
Anomalies:	Non viene segnalato nessun messaggio di errore.

TC 0.05

Test case ID:	TC 0.05
Test case description:	updateProductPerQtaAndPrz
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente aggiorna un prodotto con ID = 10 già presente nel database, cambiando la quantità e il prezzo
Result:	PASSED
Anomalies:	Non viene segnalato nessun messaggio di errore.

TC 0.08

Test case ID:	TC 0.08
Test case description:	updateProductPerQtaAndPrzWithIDError
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente aggiorna un prodotto con ID = 9 **NON** presente nel database, cambiando la quantità e prezzo
Result:	FAILED
Anomalies:	Non viene segnalato nessun messaggio di errore.

TC 0.06

Test case ID:	TC 0.06
Test case description:	updateProductPerQta
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente aggiorna un prodotto con ID = 10 già presente nel database, cambiando la quantità
Result:	FAILED
Anomalies:	Non viene segnalato nessun messaggio di errore.

TC 0.09

Test case ID:	TC 0.09
Test case description:	updateProductPerQtaWithIDError
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente aggiorna un prodotto con ID = 9 **NON** presente nel database, cambiando la quantità
Result:	FAILED
Anomalies:	Non viene segnalato nessun messaggio di errore.

TC 0.07

Test case ID:	TC 0.07
Test case description:	removeProduct
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente elimina un prodotto con ID = 10 dal database
Result:	PASSED
Anomalies:	Non viene segnalato nessun messaggio di errore.

TC 0.10

Test case ID:	TC 0.10
Test case description:	removeProductWithIDError
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente elimina un prodotto con ID = 9 **NON** presente nel database
Result:	PASSED
Anomalies:	Non viene segnalato nessun messaggio di errore.

SuppliesManagement

TC 1.00

Test case ID:	TC 1.00
Test case description:	getSupplies
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente recupera i dati di un fornitore con Piva **= 1** dal database
Result:	PASSED
Anomalies:	Non viene segnalato nessun messaggio di errore.

TC 1.01

Test case ID:	TC 1.01
Test case description:	insertSupplies

Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente inserisce un nuovo fornitore nel database con Piva = 3, chiave primaria della tabella fornitori
Result:	PASSED
Anomalies:	Non viene segnalato nessun messaggio di errore.

TC 1.02

Test case ID:	TC 1.02
Test case description:	insertSuppliesWidthIDError
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente inserisce di nuovo il fornitore con Piva = 3, ma già presente nel database
Result:	FAILED
Anomalies:	Error. Duplicate entry '3' for key 'fornitori.PRIMARY'

TC 1.03

Test case ID:	TC 1.03
Test case description:	removeSupplies
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente elimina in fornitore con Piva = 3 presente nel database
Result:	PASSED
Anomalies:	Non viene segnalato nessun messaggio di errore.

TC 1.04

Test case ID:	TC 1.04
Test case description:	removeSuppliesWithIDError
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente elimina in fornitore con Piva = 3 **NON** presente nel database
Result:	PASSED
Anomalies:	Non viene segnalato nessun messaggio di errore.

TC 1.05

Test case ID:	TC 1.05
Test case description:	updateSupplies
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente aggiorna i dati del fornitore con Piva = 3 presente nel database

Result:	PASSED
Anomalies:	Non viene segnalato nessun messaggio di errore.

TC 1.06

Test case ID:	TC 1.06
Test case description:	updateSuppliesPerIDError
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente aggiorna i dati del fornitore con Piva = 4 **NON** presente nel database
Result:	PASSED
Anomalies:	Non viene segnalato nessun messaggio di errore.

TC 1.07

Test case ID:	TC 1.07
Test case description:	updateSuppliesPayement
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente aggiorna i campi debito e pagato del fornitore con Piva = 3 presente nel database
Result:	PASSED
Anomalies:	Non viene segnalato nessun messaggio di errore.

TC 1.08

Test case ID:	TC 1.08
Test case description:	updateSuppliesPayementPerIDError
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente aggiorna i campi debito e pagato del fornitore con Piva = 4 **NON** presente nel database
Result:	PASSED
Anomalies:	Non viene segnalato nessun messaggio di errore.

TC 1.09

Test case ID:	TC 1.09
Test case description:	updateSuppliesPeyementPerFieldDebito
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente aggiorna SOLO il campo debito del fornitore con Piva = 3 presente nel database
Result:	FAILED
Anomalies:	Error. Data truncated for column 'pagato' at row 1

TC 1.10

Test case ID:	TC 1.10
Test case description:	updateSuppliesPeyementPerFieldPagato
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente aggiorna SOLO il campo pagato del fornitore con Piva = 3 presente nel database
Result:	FAILED
Anomalies:	Error. Data truncated for column 'debito at row 1

ExitsManagement

TC 2.00

Test case ID:	TC 2.00
Test case description:	addExit
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente aggiunge le spese con data, descrizione, conto, ammontare
Result:	PASSED
Anomalies:	Non viene segnalato nessun messaggio di errore.

TC 2.01

Test case ID:	TC 2.01
Test case description:	addExitWithAmountError
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente aggiunge le spese con data, descrizione, conto, ammontare = ok
Result:	FAILED
Anomalies:	Error. Data truncated for column 'importo' at row 1

TC 2.02

Test case ID:	TC 2.02
Test case description:	addExitWithDateError
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente aggiunge le spese con data giorno = 32 mese = 32 anno = 32, descrizione, conto, ammontare
Result:	PASSED
Anomalies:	Non viene segnalato nessun messaggio di errore.

TC 2.03

Test case ID:	TC 2.03
Test case description:	addExitWithCountError

Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente aggiunge le spese con data, descrizione, conto = ok, ammontare
Result:	FAILED
Anomalies:	Error. Incorrect integer value: 'ok' for column 'conto' at row 1

TC 2.04

Test case ID:	TC 2.04
Test case description:	getExit
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente recupera un elenco di spese dato un certo periodo dal 14/02/2008 al 14/02/2008
Result:	PASSED
Anomalies:	Non viene segnalato nessun messaggio di errore.

TC 2.05

Test case ID:	TC 2.05
Test case description:	getExitWithPeriodError
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente recupera un elenco di spese dato un certo periodo dal 32/32/2008 al 32/32/2008
Result:	PASSED
Anomalies:	Non viene segnalato nessun messaggio di errore. E i dati non sono visualizzati.

EntrancesManagement

TC 3.00

Test case ID:	TC 3.00
Test case description:	addEntrance - Il metodo dovrebbe inserire nuove entrate in base alla vendita di un prodotto riportando l'ID del prodotto da aggiornare, ovvero la quantità del specifico prodotto.
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente inserisce i dati per la gestione entrate con ID = 10, amount, day, month, year
Result:	FAILED
Anomalies:	Error. Column count doesn't match value count at row 1

TC 3.01

Test case ID:	TC 3.01
Test case description:	getEntrance
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente recupera un elenco delle entrate dato un certo periodo dal 23/0/2008 al 23/04/2008
Result:	PASSED
Anomalies:	Dati parziali, non vengono visualizzate i dati del prodotto associato alle entrate

TC 3.02

Test case ID:	TC 3.02
Test case description:	getEntranceWithPeriodError
Pre-condition:	Utente già registrato, come ruolo di amministratore ha effettuato il login.
Flow of events:	L'utente recupera un elenco delle entrate dato un certo periodo dal 32/32/2008 al 32/32/2008
Result:	PASSED
Anomalies:	Non viene segnalato nessun messaggio di errore. E i dati non sono visualizzati.

AdminAccess

TC 4.00

Test case ID:	TC 4.00
Test case description:	verify
Pre-condition:	Nessuna precondizione
Flow of events:	L'utente effettua l'accesso come amministratore
Result:	PASSED
Anomalies:	Non viene segnalato nessun messaggio di errore. E i dati non sono visualizzati.

TC 4.01

Test case ID:	TC 4.01
Test case description:	verifyUserNotAdmin
Pre-condition:	Nessuna precondizione
Flow of events:	L'utente effettua l'accesso come amministratore **NON** presente nel database
Result:	FAILED
Anomalies:	Non viene segnalato nessun messaggio di errore. E i dati non sono visualizzati.

3.3 Code smells detection

Dall'ispezione del codice dell'intero progetto eseguito tramite Intellij IDEA, la maggior parte delle classi hanno riscontrati i seguenti code smell:

Duplicated code fragment

Empty catch clause

> Il blocco nella clausola catch è vuota, quindi viene eseguito questo blocco e non viene lanciata nessuna eccezione, non possiamo sapere cosa non va nel codice.

Long statement

Unused assignment

Magic Number

> Rilevato quando vengono usati numeri che mancano di una corretta semantica, rendendo il codice meno leggibile.
>
> Soluzione:
>
> Si assegna ad ogni Magic number una costante.

Complex Method: cyclomatic complexity of the method

> La complessità ciclomatica è una metrica che indica quanto un programma è complesso e calcola il numero di cammini linearmente indipendenti di un programma.

Per più dettagli consultare il file inspections_results.html