

Mini-Market Billing System

Progetto di Ingegneria del Software
A.A. 2023/24



Mangini Alessandro - Pesenti Alessandro - Pesenti Luca



Obiettivo

Realizzazione di un'applicazione desktop destinata a piccoli esercizi commerciali che consenta la semplificazione, l'ottimizzazione e la gestione delle attività quotidiane

Le principali funzionalità del sistema sono:

- Emissione scontrini
 - Gestione magazzino
 - Amministrazione personale
 - Monitoraggio prestazioni economiche
- 
- 
-

Difficoltà incontrate

- Familiarizzazione con tecnologie nuove come **Java Swing**, **Hibernate ORM** e **GitHub**
- Implementazione dei principi teorici del **pattern MVC** al fine garantire una corretta separazione delle responsabilità
- Bisogno di comprendere meglio le necessità dei partecipanti per ottimizzare l'allocazione delle competenze e superare difficoltà legate all'organizzazione

Paradigmi di programmazione e modellazione



Linguaggio di programmazione: **Java**
Ha permesso di sfruttare il paradigma della OOP e di integrare in modo efficace il linguaggio di modellazione scelto: **UML**, garantendo una coerenza tra progettazione e implementazione.



Ambiente di sviluppo integrato:
Eclipse IDE
In combinazione con **GitHub Desktop** ha facilitato lo sviluppo del progetto e dinamiche di collaborazione del gruppo.

Strumenti utilizzati



H2 Database

Database embedded per l'archiviazione dei dati in locale leggero e facile da integrare



Hibernate ORM

Framework per la gestione della persistenza dei dati, che mappa gli oggetti Java al database H2



Maven

Strumento per la gestione delle dipendenze e la configurazione del progetto



Dbeaver

Interfaccia grafica utilizzata in locale per esplorare e verificare il database H2

Software configuration management

Ampio utilizzo di **GitHub** come piattaforma principale per il versioning e la gestione collaborativa, che ha permesso al team di organizzare il lavoro, monitorare i progressi e mantenere un repository centralizzato e ben strutturato.

Organizzazione repository

Due cartelle principali:

- `./code/`: contiene il codice sorgente del progetto, aggiornato e pronto per essere eseguito
- `./docs/`: include tutta la documentazione necessaria, come diagrammi UML, il piano di lavoro e altre risorse utili per il coordinamento del team

Workflow collaborativo su GitHub

Il team ha lavorato principalmente in presenza, ma ha utilizzato GitHub per mantenere una sincronizzazione continua e per garantire una gestione ordinata del codice.

Per alcune funzionalità specifiche sono stati creati **branch** dedicati per lo sviluppo in parallelo. Ogni branch è poi stato unito al ramo principale (main) tramite **pull request**.






















Flusso di lavoro effettivo:

- **Collaborazione diretta in presenza:** il gruppo ha programmato insieme, discutendo e implementando le modifiche in tempo reale
 - **Branch dedicati per funzionalità specifiche:** alcuni membri hanno lavorato su branch separati per nuove funzionalità o correzioni di bug
 - **Pull request per revisioni:** le modifiche principali sono state sottoposte a revisione tramite pull request, con discussioni e verifiche del team
-

Gestione delle versioni e issue

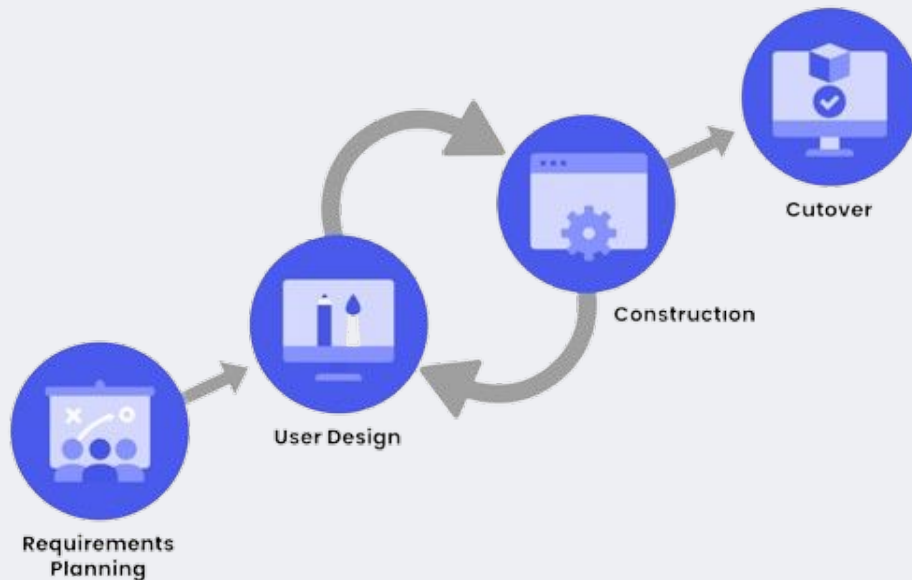
Sono stati effettuati **commit** frequenti e regolari, garantendo una traccia chiara. Il ramo principale (main) è stato mantenuto stabile, accettando solo modifiche verificate.

Inoltre è stata utilizzata la funzionalità degli **issue**, impiegati principalmente per segnalare bug individuati e per definire nuove funzionalità o aggiornamenti da implementare:

<input type="checkbox"/>	<input checked="" type="checkbox"/>	The logged manager does not appear in ModifyUserPanel's JTable and JComboBox along with all the cashiers	 	 2
		#11 by mangini-a was closed 2 weeks ago		
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Unique index or primary key violation when trying to insert a new user into the app_user table	 	 1
		#10 by mangini-a was closed 2 weeks ago		
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Sincronizzazione tra le Tab annidate "Add a new Item" e "Modify an existing Item"	 	 1
		#8 by AlePesenti was closed 3 weeks ago		
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Sviluppo sezione Accounting		 1
		#7 by AlePesenti was closed 2 weeks ago		
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Sviluppo sezione Staff		
		#6 by AlePesenti was closed 2 weeks ago		
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Sviluppo sezione Stock		 1
		#5 by AlePesenti was closed 3 weeks ago		
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Aggiunta pannello scadenze		 1
		#4 by AlePesenti was closed 3 weeks ago		

Software Life Cycle

- Adatto allo sviluppo di software guidato dai requisiti legati alla UI □ **Swing**
- Enfasi sulla prototipazione, che consente di ricevere feedback e suggerimenti dagli utenti finali con costanza □ **Qualità**
- Si focalizza precocemente sulle criticità, adattandosi ad esse grazie alle evidenze empiriche raccolte nella fase embrionale dello sviluppo □ **Mitigazione dei rischi**

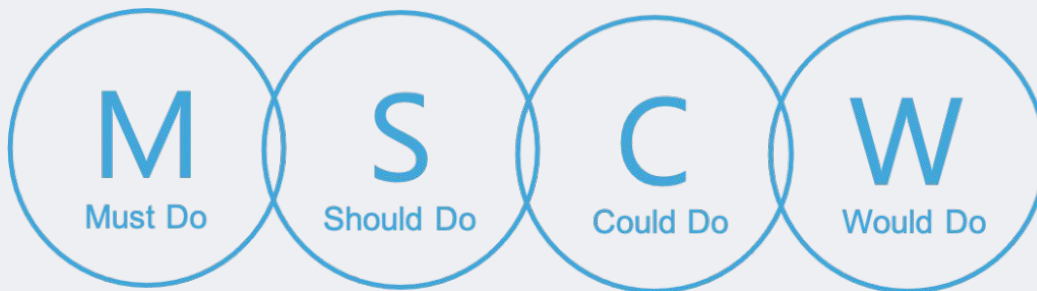


Requisiti

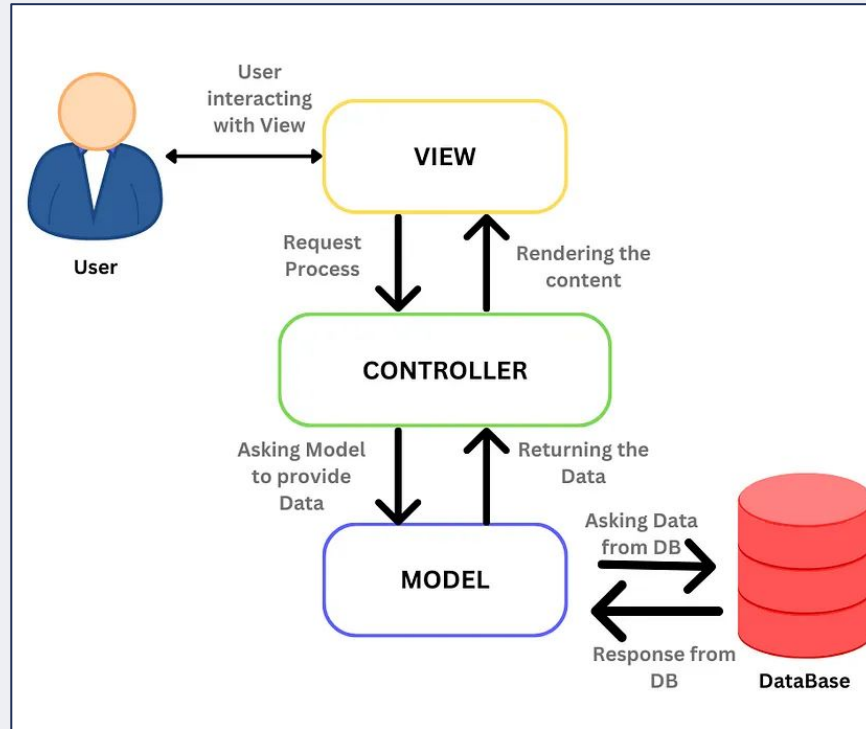
Tecniche di elicitazione dei requisiti impiegate:

- **intervista:** provenendo da piccole frazioni montane, ci siamo interfacciati con piccoli alimentari locali, il cui personale potrebbe trarre beneficio dal programma;
- **analisi dei casi d'uso:** sono state individuate le necessità principali di un negoziante e gli strumenti utili ad ottimizzare le attività di cui si occupa quotidianamente.

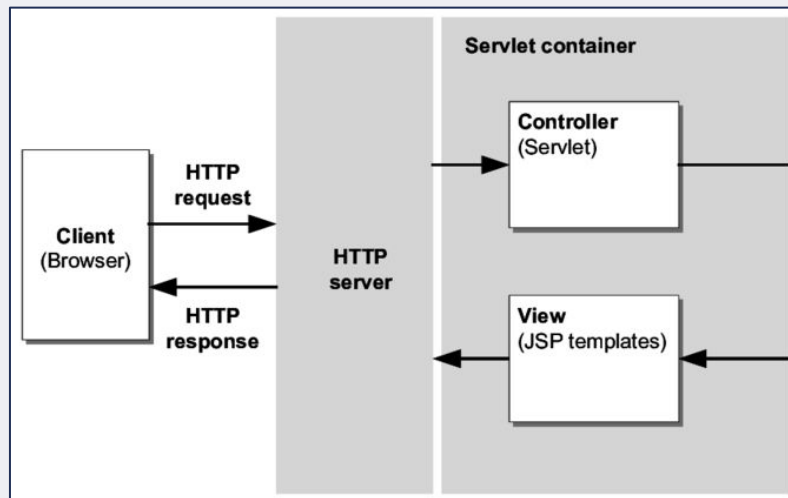
I risultati della fase di analisi sono stati convogliati in un documento di specifica dei requisiti, redatto facendo fede allo standard **IEEE 830** (1998).



Architettura (1)



Architettura (2)



SVILUPPI FUTURI?

Un punto di forza di tale scelta risiede nel fatto che, grazie alla modularità, le componenti sviluppate possano essere riutilizzate nel contesto di applicazioni analoghe: ad esempio, per creare una web app evitando di riprogrammare la logica di controllo e rivedere il livello di accesso ai dati.

Design pattern

Principi di Progettazione

Nel progetto sono stati adottati due principi fondamentali della programmazione orientata agli oggetti:

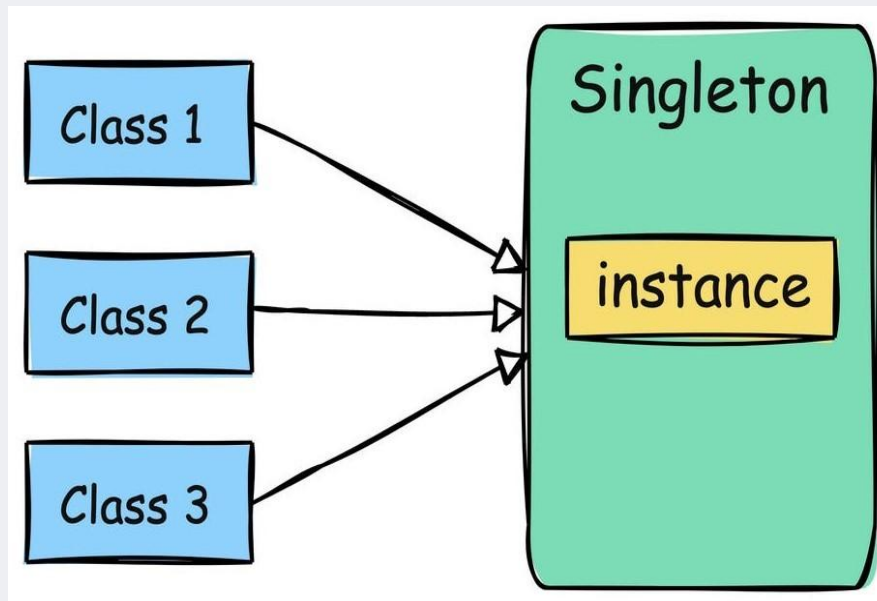
- **Astrazione:** Realizzata separando la struttura applicativa in progetti, package e classi, e utilizzando interfacce per aumentare la modularità. Ad esempio, il package Controller include tre interfacce e tre classi concrete necessarie per operare sul database, ottenendo un grado di astrazione di 1/2.
- **Occultamento delle informazioni:** Implementato per ridurre la visibilità di campi, costruttori e metodi non necessari. Per questo è stato utilizzato il plug-in **UCDetector**, che ha permesso di identificare e ottimizzare gli elementi superflui rendendoli meno accessibili agli altri package.

Design pattern (2)

Il **Singleton Pattern** è stato utilizzato per garantire che alcune classi abbiano una sola istanza all'interno del sistema, semplificando la gestione delle risorse.

Questo pattern è stato applicato in particolare alle classi del package Controller, i cui metodi sono utilizzati da diverse classi del package View. Ciò ha permesso di:

- Evitare duplicazione di risorse.
- Fornire un punto di accesso globale per le operazioni sui dati.



Misurazione del codice

JDepend

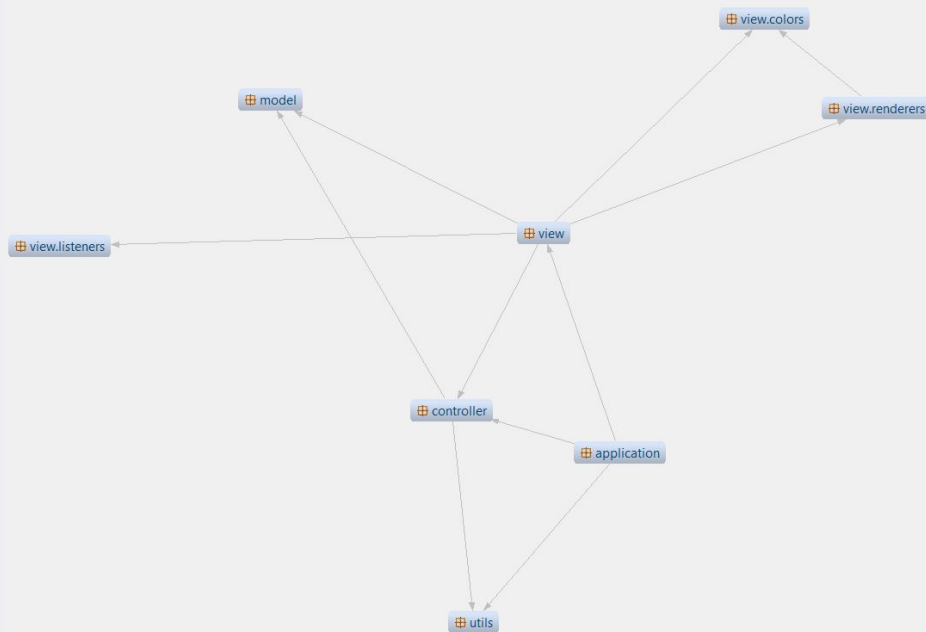
<i>Package</i>	CC	AC	Ca	Ce	Astrazione	Instabilità	Distanza
view.renderers	4	0	1	1	0.00	0.50	0.50
view.listeners	0	1	1	0	1.00	0.00	0.00
view.colors	1	0	2	0	0.00	0.00	1.00
view	16	0	1	10	0.00	0.90	0.09
utils	1	0	2	3	0.00	0.60	0.39
model	5	0	2	1	0.00	0.33	0.66
controller	4	3	2	4	0.42	0.66	0.09
application	1	0	0	6	0.00	1.00	0.00

È stato utilizzato JDepend per l'analisi delle dipendenze, valutando la qualità strutturale del codice e analizzando metriche e parametri utili a fornire una panoramica del sistema e indentificarne punti critici.

Dalla tabella si evince un livello di astrazione generalmente basso questo implica che sia necessario un processo di refactoring atto ad aumentarne il livello.

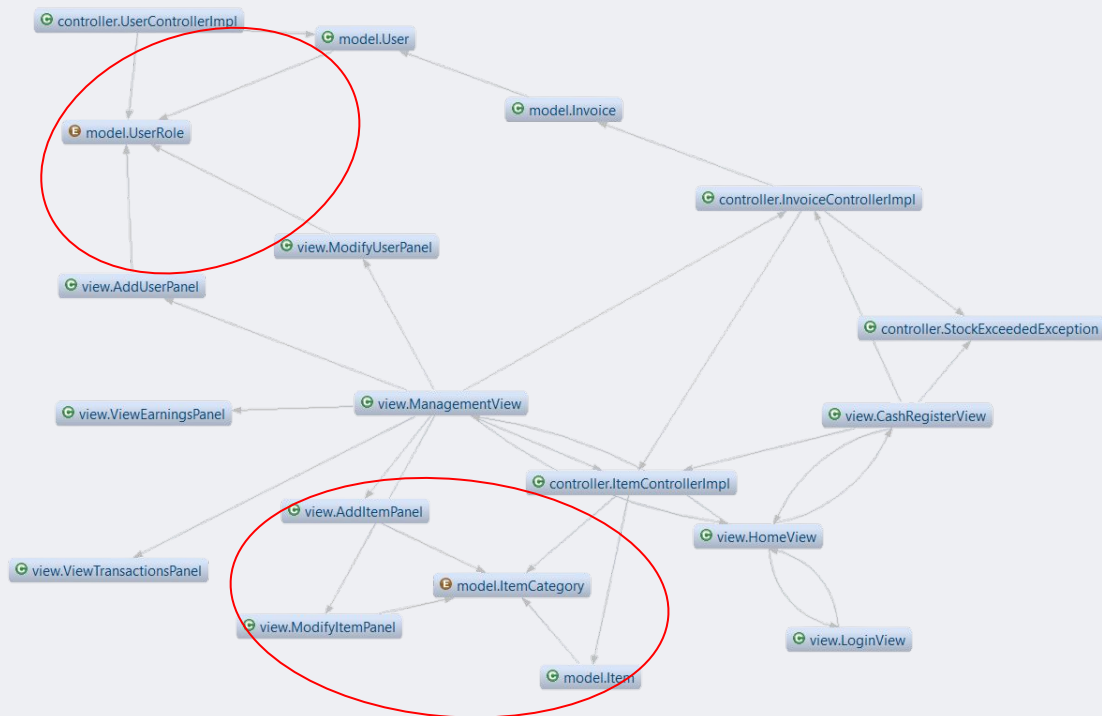
Misurazione del codice (2)

Java Dependency Viewer



Da questo grafico generale dei pacchetti notiamo che esiste una connessione diretta tra controller e model, questo potrebbe suggerire che il principio di separazione delle responsabilità tra View, Model, e Controller non è rispettato perfettamente.

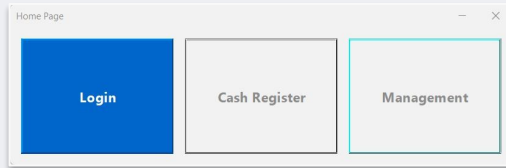
Misurazione del codice (3)



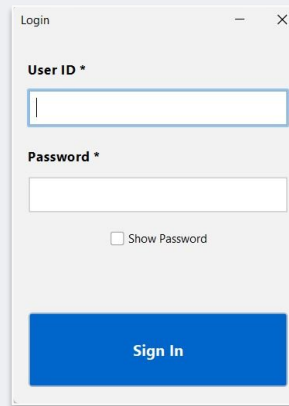
Analizzando più approfonditamente abbiamo notato che le relazioni tra view e model erano dovute alle classi enumerabili. Una soluzione efficace potrebbe essere quella di raggruppare tutte le classi enum in un pacchetto apposito.

Implementazione

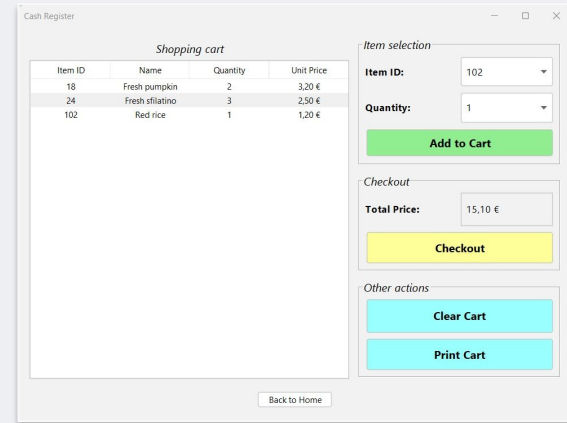
Le funzionalità aggiunte nell'applicazione sono:



Schermata di selezione iniziale

A screenshot of the 'Login' window. It has a title bar with a close button. Below the title bar, there are two input fields: 'User ID *' and 'Password *'. Below the password field is a checkbox labeled 'Show Password'. At the bottom is a large blue button labeled 'Sign In'.

Login utente

A screenshot of the 'Cash Register' window. It has a title bar with standard window controls. The main area is titled 'Shopping cart' and contains a table with columns: Item ID, Name, Quantity, and Unit Price. The table has three rows of data. To the right of the table is an 'Item selection' section with dropdowns for 'Item ID' and 'Quantity', and an 'Add to Cart' button. Below this is a 'Checkout' section with a 'Total Price' display and a 'Checkout' button. At the bottom right is an 'Other actions' section with 'Clear Cart' and 'Print Cart' buttons. A 'Back to Home' button is at the bottom center.

Funzione di cassa:

- creazione del carrello
- emanazione degli scontrini

Implementazione (2)

Le funzionalità aggiunte nell'applicazione sono:

Management
Stock Staff Accounting

Add a new item Modify an existing item's details

Products currently in stock

ID	Name	Quantity	Unit Price	Category
16	Checkpear	20	0.80 €	CANNING
18	Fresh pumpkin	5	3.20 €	VEGETABLES
20	Chocolate bars	17	1.20 €	SNACKS
23	Bottle cleaner	29	2.19 €	CLEANING
24	Fresh vitellino	25	2.50 €	BAKERY
25	Cold drink	10	1.35 €	BEVERAGE
26	Pear jam	20	3.99 €	BIO
102	Red rice	40	1.20 €	GRAIN

Name *

Quantity *

Unit Price *

Category *

BAKERY

Add

Back to Home

Operazioni sull'inventario
Aggiunta, rimozione e modifica

Management
Stock Staff Accounting

Add a new user Modify an existing user's credentials

Staff members you can edit

ID	Name	Surname	Role
1	Tommaso	Vinciguerra	CASHIER
2	Mattéo	Di Falco	MANAGER

User ID *

1

First Name *

Last Name *

Password *

Role *

CASHIER

Remove Update

Back to Home

Operazioni sullo staff
Aggiunta, rimozione e modifica

Management
Stock Staff Accounting

View the past transactions list Quantify the day-to-day profit

Transaction history

Date	Operator ID	Name	Surname	Role	Amount
30-10-2024 09:27:42	2	Mattéo	Di Falco	MANAGER	11.50 €
30-10-2024 10:30:15	2	Mattéo	Di Falco	MANAGER	11.50 €
05-11-2024 17:07:09	4	Eleonora	Antognini	MANAGER	10.60 €
06-11-2024 17:29:07	4	Eleonora	Antognini	MANAGER	4.60 €
07-11-2024 11:25:16	4	Eleonora	Antognini	MANAGER	6.30 €
09-11-2024 11:36:34	4	Eleonora	Antognini	MANAGER	10.00 €

Back to Home

Sezione di accounting

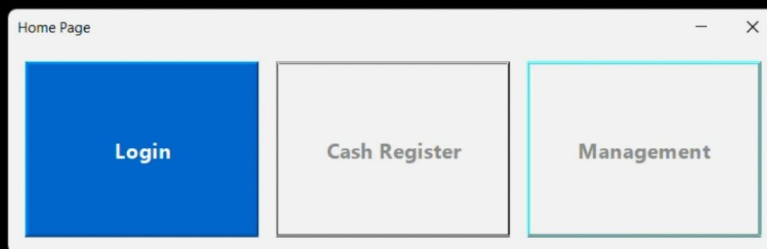
- Visualizzazione degli scontrini
- Report sui guadagni giornalieri

Implementazione (3)

Funzionalità non implementate:

- Incremento delle funzionalità di reportistica, magari generando report con dettagli sulle vendite, profitti e articoli più venduti.
- Sistema di notificazione sul livello delle scorte, in modo da evitare che un articolo rimanga scoperto. Magari ponendo particolare attenzione ai prodotti più acquistati.
- Aggiunta delle varie tassazioni per categoria di prodotto.
- Gestione degli sconti.
- Aggiungere l'opzione per esportare i dati (ad esempio, inventario o storico delle transazioni) in formati standard come CSV o Excel.
- Aggiunta delle date di scadenza per alcuni lotti di prodotti, in modo da renderne più comoda la consultazione.

Demo



Modellazione

Sono stati utilizzati i diagrammi UML per modellare diversi aspetti del sistema realizzato.
Lo strumento utilizzato per la realizzazione di questi diagrammi è StarUML.

Use Case Diagram

Activity Diagram

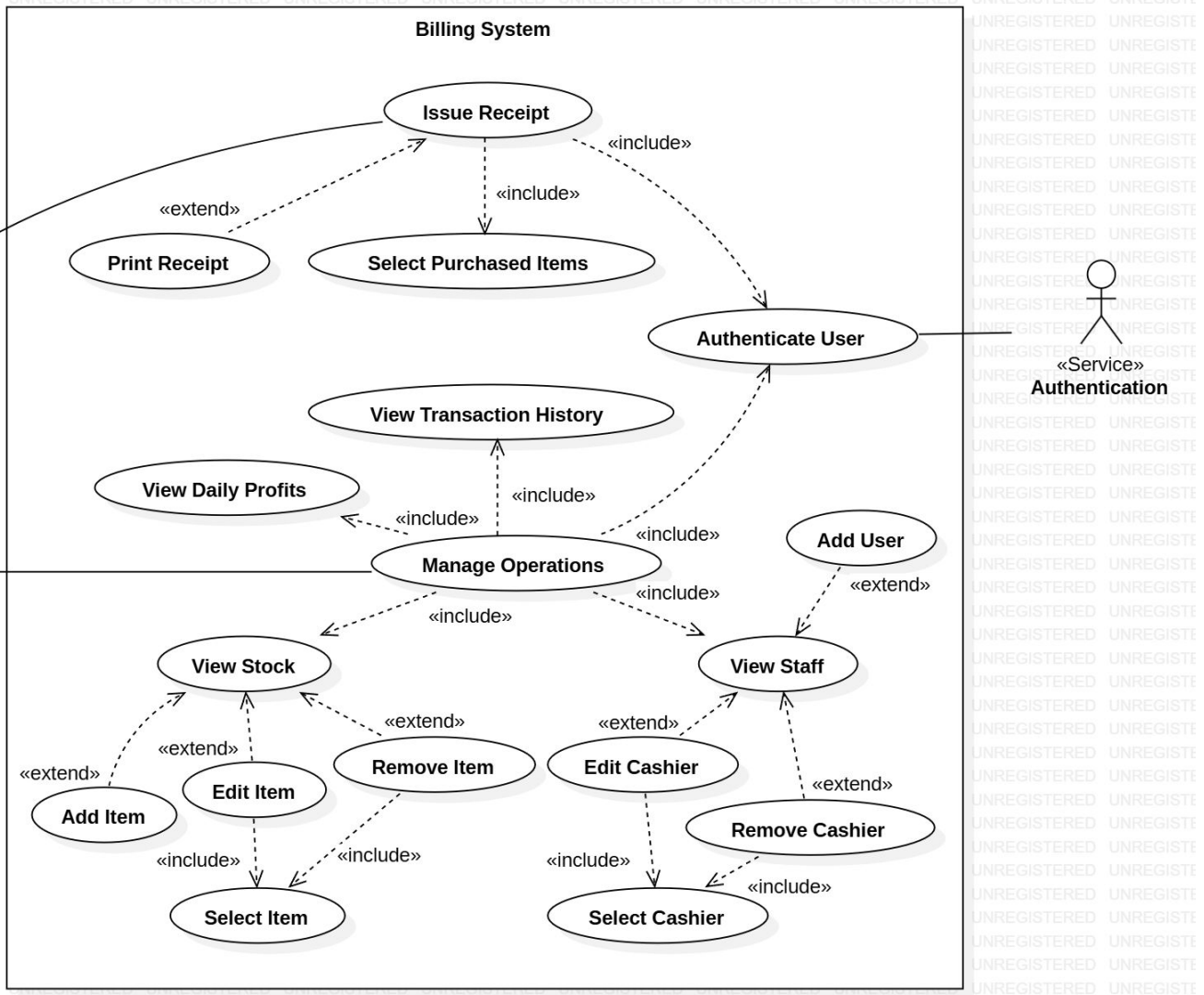
State Chart Diagram

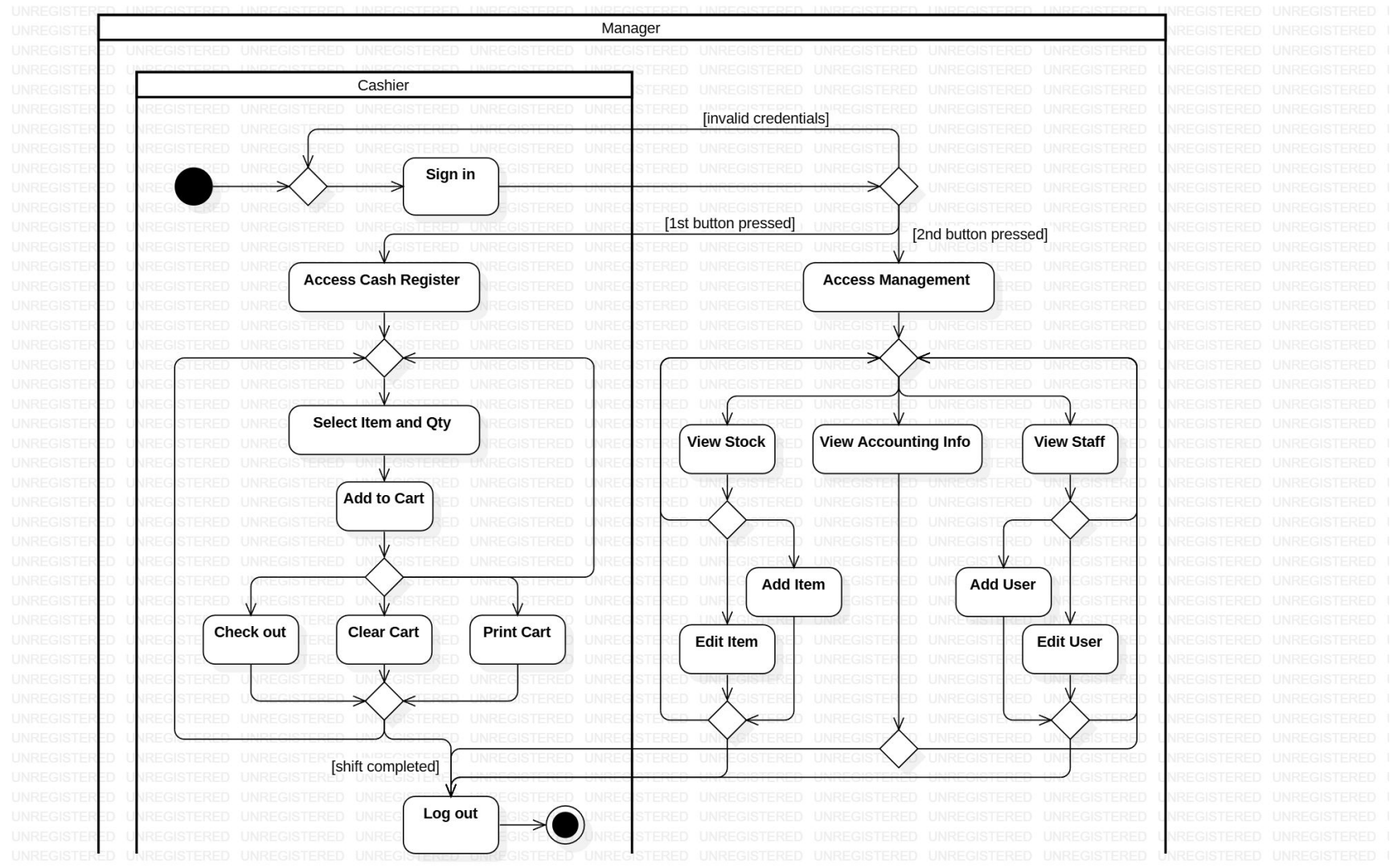
Sequence Diagram

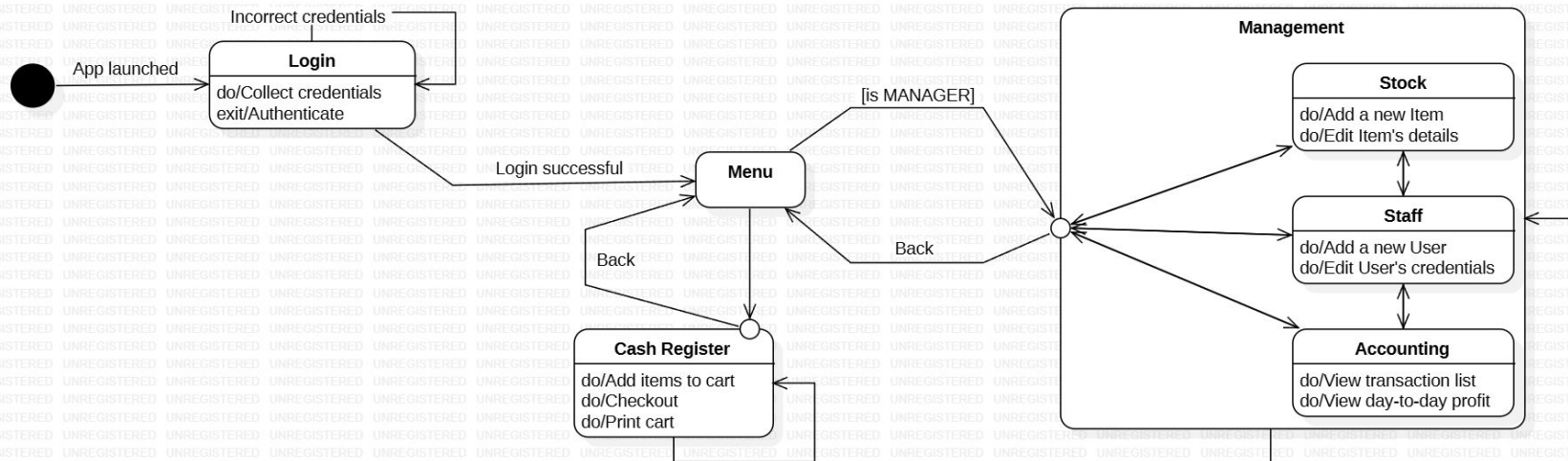
Package Diagram

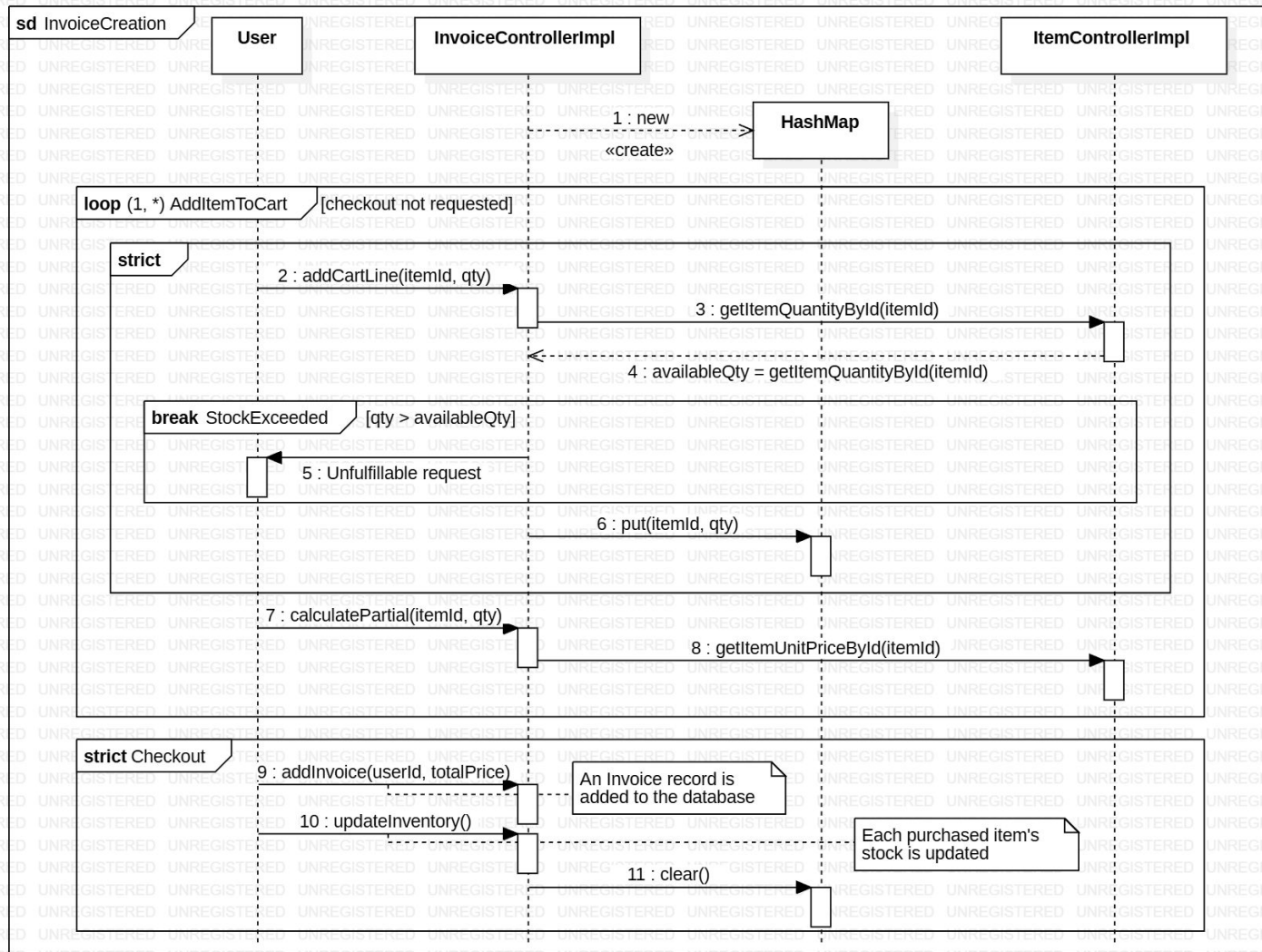
Class Diagram

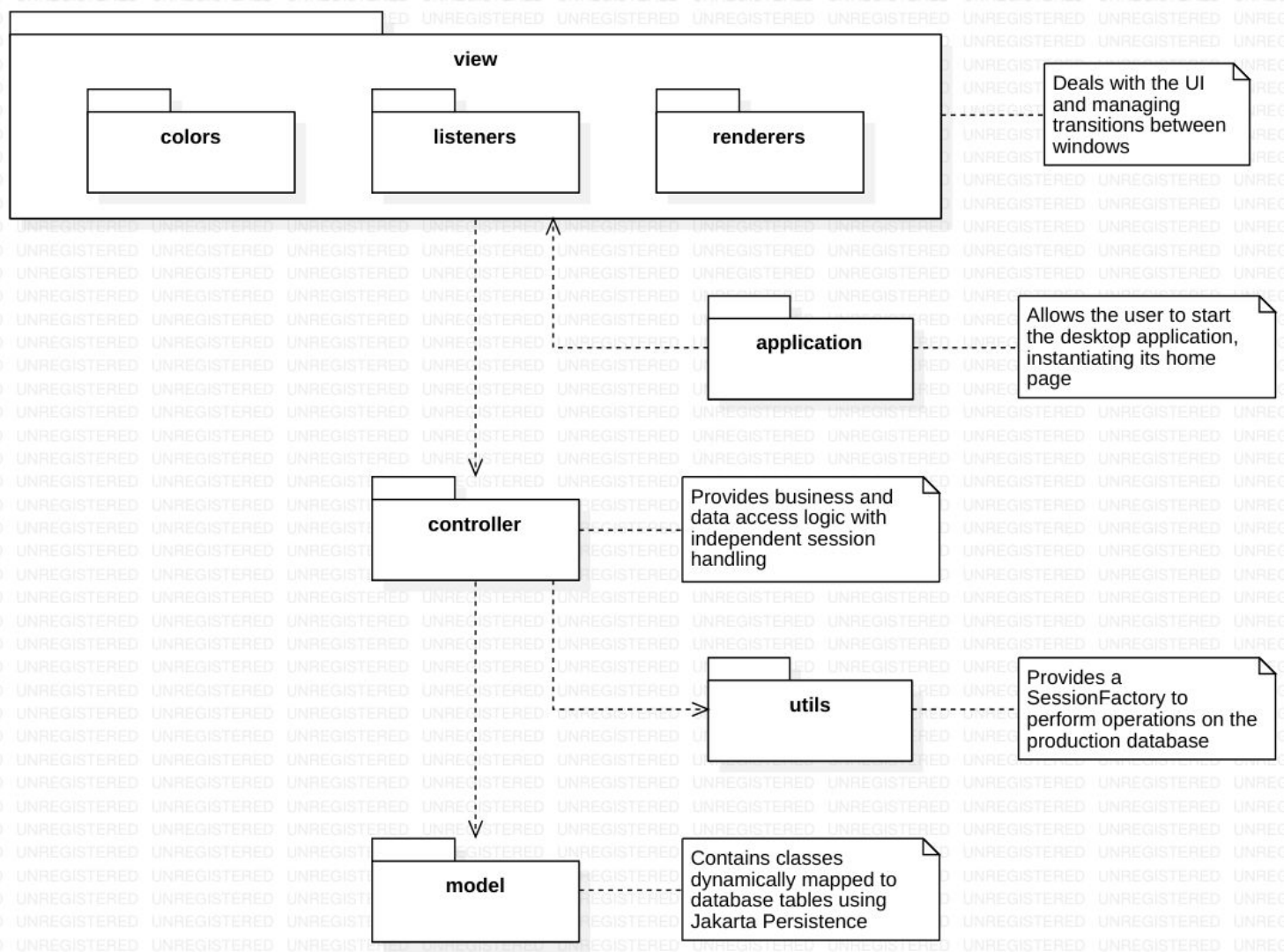


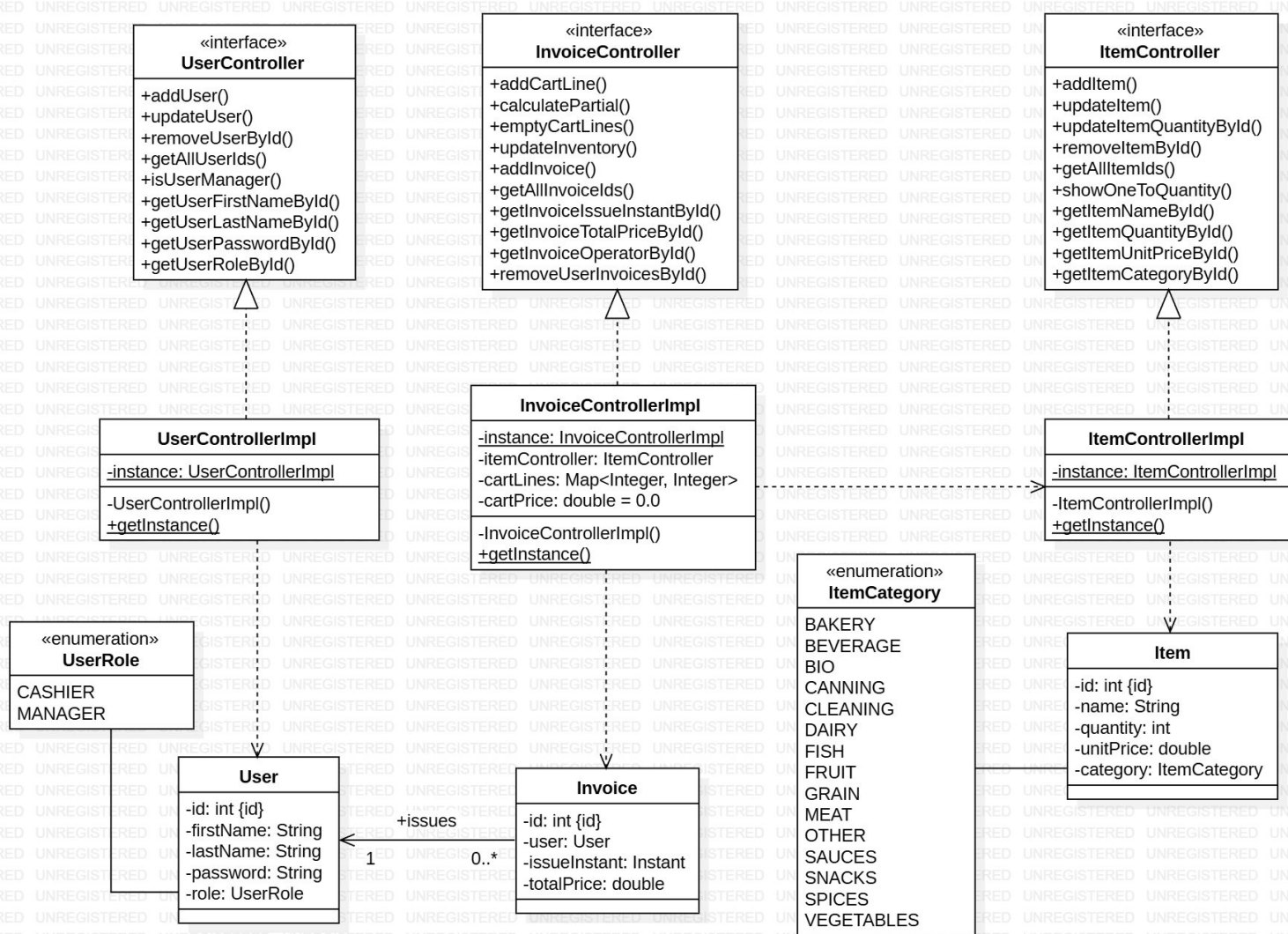




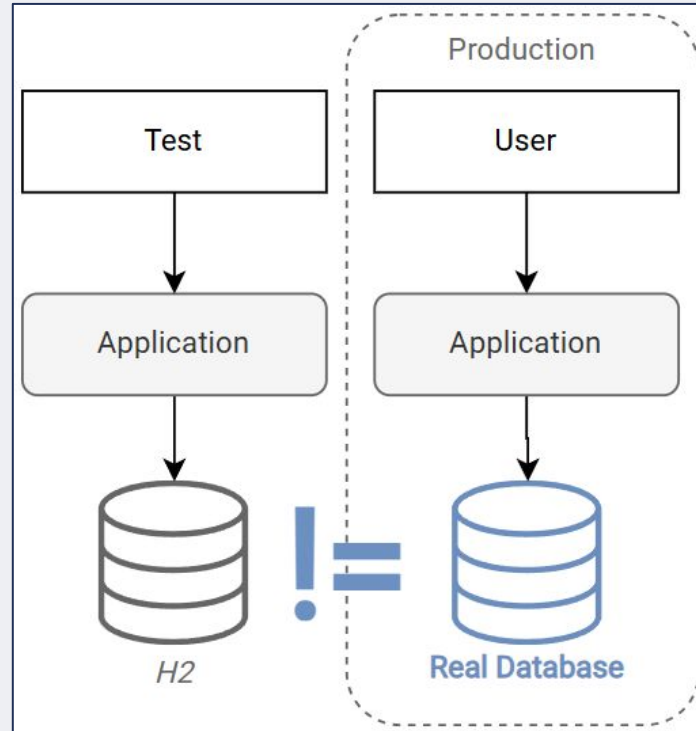








Testing (1)



Testing (2)

```
@Test
public void testIsUserManager_UserIsNotManager() {
    // Arrange
    Integer userId = userController.getAllUserIds().get(0); // Get the ID of the first user

    // Act
    boolean result = userController.isUserManager(userId);

    // Assert
    assertFalse(result, "User should be identified as a manager");
}

@Test
public void testIsUserManager_UserDoesNotExist() {
    // Act
    boolean result = userController.isUserManager(999); // Assuming 999 is a non-existent user ID

    // Assert
    assertFalse(result, "Non-existent user should not be identified as a manager");
}
```



Fine