RELAZIONE PROGRAMMAZIONE III: GESTIONE MAGAZZINO

MATRICOLA: 0124001729

NOME E COGNOME: RAFFAELE CUSANO

ANNO ACCADEMICO: 2021/22

INDICE

1 Traccia - Negozio on-line

Si vuole simulare un negozio per la vendita di prodotti su internet. Ogni prodotto `e identificato da un codice, da un nome, una descrizione, una quantit`a di scorta, il costo e la categoria. I prodotti sono suddivisi per categorie e sottocategorie (e.g., Casalinghi-Saponi, . . .).

Scrivere un programma per la gestione del negozio. L'accesso pu`o essere effettuato in modalit`a amministratore e in modalità utente.
L'amministratore può effettuare le seguenti operazioni

• inserire un nuovo prodotto con le sue informazioni
• eliminare un prodotto
• visualizzare periodicamente tutti gli acquisti effettuati da un dato utente

L'utente può effettuare le seguenti operazioni

- inserire i prodotti nel carrello della spesa
- eliminare un prodotto precedentemente inserito nel carrello
- effettuare il pagamento. Il pagamento può avvenire secondo le modalit`a: contanti,carta di credito o bancomat.

2 Framework 2.1 Spring MVC

- Il progetto consiste in un'applicazione Web realizzata con framework Spring su plattaforms Java. In particolare l'architettura della Web application è realizzata con Spring MVC, che sfrutta il pattern MVC costituito da 3 componenti:
- Model che si occupa di accedere ai dati necessari alla logica di business ossia oggetti di gestione o classi di accesso al database
- View che si occupa di creare l'interfaccia utilizzabile dall'utente e che espone i dati da esso richiesti. Per questa componente `e utilizzato Thymeleaf, ossia un motore di template Java XML / XHTML / HTML5. Nel caso specifico dell'applicazione i template sono scritti in HTML 5, con Javascript, jQuery, e framework CSS Bulma.
- Controller implementano la logica di controllo, riceve i comandi dell'utente (attraverso lo strato View) e li attua modificando lo stato degli altri due componenti. Nello specifico dell'applicazione i Controller sono rappresentati da classi (chiamate appositamante @Controller) che rimangono "in ascolto" su un determinato URL e, grazie ai Model e alle View, si occupano di gestire la richiesta dell'utente. Anche se i

Controller sono molteplici, le richieste vengono inviate comunque ad unico Handler Mapping che seleziona il Controller adeguato.
- La web application viene eseguita su un web container Apache Tomcat, istanziato da Spring di default.

2.2 Spring Security

- Spring Security è un framework di Spring utilizzato nella Web application per l'autenticazione e le autorizzazioni degli utenti. Queste funzioni sono implementate nella classe WebSecurityConfig.
- Nello specifico l'interfaccia di login che viene sfruttata è quella di base di Spring Security, che è configurata attraverso la funzione configure, la quale fornisce all'AuthenticationManagerBuilder i dettagli dell'utente attraverso la classe UserDetailsServiceImp. Questi dettagli sono caricati direttamente dal database attraverso il DAO e la sua funzione findUsername() che li restitusice in base all'username.
- Per quanto riguarda le autorizzazioni, sempre nella classe WebSecurityConfig, un'altra funzione configure(), gestisce le richieste in base al ruolo degli utenti che può essere ADMIN o CLIENTE.

3 Database

Il database utilizzato è MariaDB Server che è un database relazionale. La connessione al database avviene con la classe Connect attraverso l'URL. E' stato scelto un database su server in modo da rappresentare meglio il concetto di **web application**.

```java
package com.market.DAO;

import ...

public class Connect {

    private Connection connection = null;

    private static final String DATABASE_DRIVER = "org.mariadb.jdbc.Driver";
    private static final String DATABASE_URL = "jdbc:mariadb://localhost:3306/dbmagazzino?user=root&password=";

    private String jdbcDriver;
    private String databaseUrl;

    Connect() {
        Properties properties = new Properties();
        try {
            jdbcDriver = properties.getProperty( key: "driver", DATABASE_DRIVER);
            databaseUrl = properties.getProperty( key: "url", DATABASE_URL);
            connection = DriverManager.getConnection(DATABASE_URL);
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
    }
```

4 Pattern

I pattern utilizzati nella realizzazione della Web application sono:

- **Singleton**: per l'implementazione del DAO. In questo modo è garantito un unico punto di accesso per richiedere operazioni al database.

- **Strategy**: per le diverse metodologie di pagamento. Questa implementazione consente di modificare dinamicamente le istruzioni in base al tipo di pagamento scelto dall'utente.

- **Prototype** : per clonare le istanze di un oggetto ricevuta che in base al parametro di input può diventare una ricevuta di una carta di credito, debito o paypal partendo dalla stessa astrazione "Ricevuta".

- **Factory:** Sono state utilizzate due Factory (categoryFactory, paymentFactory) in modo da creare oggetti diversi partendo da due interfacce base che in base al parametro di input crea l'oggetto desiderato.