Problema 3

Dezvoltați o aplicație care poate fi utilizată într-un **lanț de magazine de parfumuri**. Aplicația va avea 3 tipuri de utilizatori: angajat al unui magazin de parfumuri, manager al lanțului de magazine de parfumuri și administrator.

Utilizatorii de tip **angajat** al unui magazin de parfumuri pot efectua următoarele operații după autentificare:

1. Filtrarea parfumurilor după următoarele criterii: producător, disponibilitate, preț;
2. Operații CRUD în ceea ce privește persistența parfumurilor din magazinul la care lucrează acel angajat.
3. O imagine care conține text, captură de ecran, diagramă

   Conținutul generat de inteligența artificială poate fi incorect.
4. Pentru dezvoltarea aplicației „PerfumeApp” s-a ales Java ca limbaj principal de programare, împreună cu JavaFX pentru realizarea interfeței grafice. Alegerea acestui limbaj este motivată de următoarele considerente:
5. Portabilitate ridicată
6. Java este un limbaj independent de platformă, datorită principiului „Write Once, Run Anywhere”. Aceasta permite ca aplicația să fie rulată pe diverse sisteme de operare (Windows, Linux, macOS) fără modificări semnificative.
7. Suport pentru programarea orientată pe obiecte (OOP)
8. Aplicația este structurată folosind principiile OOP, ceea ce permite o organizare clară a codului în clase și pachete, separarea responsabilităților și reutilizarea componentelor.
9. Integrare cu baze de date și servicii REST
10. Java oferă suport puternic pentru conexiuni la baze de date (prin JDBC sau ORM-uri precum Hibernate) și pentru implementarea de aplicații conectate la servicii REST, lucru important în scenariul aplicației PerfumeApp.
11. Framework modern pentru GUI – JavaFX
12. JavaFX este un framework modern pentru realizarea interfețelor grafice, care oferă:
13. Componente vizuale avansate și personalizabile;
14. Suport pentru stilizare cu CSS;
15. Integrare facilă cu arhitecturi precum MVP (Model-View-Presenter), asigurând separarea logicii de prezentare de logica de business.
16. Comunitate și resurse bogate
17. Java este unul dintre cele mai utilizate limbaje la nivel global, beneficiind de o comunitate activă, documentație extinsă și o gamă largă de biblioteci și framework-uri.
18. n cadrul aplicației „PerfumeApp”, au fost utilizate mai multe diagrame UML pentru a reprezenta structura și comportamentul sistemului. Aceste diagrame sunt esențiale pentru înțelegerea arhitecturii aplicației și pentru facilitarea procesului de proiectare și implementare.
19. 1. Diagrama de cazuri de utilizare (Use Case Diagram)
20. Această diagramă descrie interacțiunile dintre actorii aplicației și funcționalitățile sistemului. În aplicația PerfumeApp există următorii actori principali:
21. Administrator – gestionează utilizatorii aplicației;
22. Manager – vizualizează statistici, gestionează parfumurile și stocurile;
23. Angajat – efectuează operații CRUD asupra parfumurilor dintr-un magazin.
24. Principalele cazuri de utilizare:
25. Autentificare utilizator;
26. Adăugare / modificare / ștergere parfum;
27. Filtrare și căutare parfumuri;
28. Gestionarea utilizatorilor (doar pentru administrator);
29. Exportul datelor (CSV, TXT, JSON) – pentru manager.
30. Rolul diagramei: oferă o imagine generală asupra funcționalităților aplicației și a actorilor implicați.
31. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
32. 2. Diagrama de clase
33. Diagrama de clase reprezintă structura statică a aplicației, afișând clasele principale, atributele acestora, metodele și relațiile dintre ele. În cazul aplicației PerfumeApp, clasele principale sunt:
34. Model: Perfume, Stock, Store, User;
35. Repository: PerfumeRepository, StockRepository, StoreRepository, UserRepository (pentru acces la date);
36. Presenter: LoginPresenter, EmployeePresenter, ManagerPresenter, AdminPresenter (pentru logica aplicației);
37. View: LoginView, EmployeeView, ManagerView, AdminView (interfața grafică).
38. Rolul diagramei: clarifică arhitectura internă a aplicației și relațiile dintre componente.
39. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
40. 3. Diagrama de secvență
41. Diagrama de secvență arată cum interacționează obiectele în cadrul unui scenariu specific. De exemplu, pentru operația „Adăugare parfum de către angajat”, diagrama arată pașii:
42. Angajatul introduce datele parfumului în interfață (EmployeeView);
43. View-ul trimite datele către EmployeePresenter;
44. Presenterul apelează metoda din PerfumeRepository pentru salvarea în baza de date;
45. Repository-ul confirmă salvarea;
46. Presenterul notifică View-ul pentru actualizarea listei.
47. Rolul diagramei: evidențiază fluxul de mesaje și ordinea execuției între componente.