A picture containing logo

Description automatically generated**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**CATEDRA CALCULATOARE**

**LANT DE MAGAZINE DE INCALTAMINTE**

**Documentatie**

**Proiectul numarul 4**

**Hitu Octavian**

**Grupa:30236**

Enuntul Problemei

Scopul proiectului este de a dezvolta o aplicatie care poate fi utilizata pentru un lant de magazine de incaltaminte. Aplicatia are doar doua tipuri de utilizatori.Primul este administratorul care poate modifica informatiile legate de ceilalti utilizatori , iar cel de-al doilea este angajatul care poate modifica informatiile despre produsele magazinelor , dar poate si aplica anumite cautari pe ele, urmate de crearea unor statistici in functie de pretul si disponibilitatea produselor din magazine. Un aspect important este acela ca aplicatia este create pe sablonul architectural model-view-viewmodel, folosinf sablonul de proiectare command si designul creational builder.

Instrumente utilizate si justificarea limbajului ales

Ca si prim pas a fost nevoie sa inteleg cum trebuie structurata aplicatia si clasele din ea. Ca sa reusesc acest lucru am realizat diagrama de utilizare , dupa care cea de clase , acestea doua fiind facute cu ajutorul aplciatiei StarUML. Dupa ce am realizat acest prim pas am trecut la implementarea aplicatiei in sine folosind ca si IDE Intellij , programand in limbajul java. Am respectat arhitectula model-view-viewmodel. Am ales sa folosesc java pentru a implementa aplicatia deoarece este un limbaj usor de inteles, cu foarte multe beneficii. Am mai realizat alte aplicatii in acest limbaj pe parcursul anilor de facultate si pot spune ca sunt familiarizat cu el.

Descrierea diagramelor UML

Diagram

Description automatically generatedPoza de mai sus reprezinta diagrama de utilizare. Dupa cum a fost cerut si in problema , diagrama de mai sus are doi utilizatori, un angajat si un administrator. Pentru amandoi prima interactiune cu aplicatie este la fel. Amandoi trebuie sa se logheze cu un username si o parola. Dupa introducerea unor date valide atat administratorul cat si angajatul vor intra in interfata destinate fiecaruia. Daca vreunul din ei introduce date de logare incorecte va fi afiasta o eroare.

Administratorul are controlul absolut asupra tuturor utilizatorilor, cat si asupra lui. El poate sa creeze noi utilziatori , sa ii stearga si sa ii modifice , pe scurt poate face operatii CRUD pentru utilizatori. Toate aceste operatii se fac pe o baza de date mysql.

Angajatul , dupa introducere corecta a datelor, va intra in interfata sa. El are controlul asupra lantului de incaltaminte. El poate modifica orice isi doreste. Poate crea magazine sau incaltaminte noua, poate sa le stearga, editeze si vizualiza. Acesta pentru a putea gasi mai usor incaltaintea are la dispozitie anumite cautari si filtrari in functie de nume, producator, disponibilitate si altele. La fel ca la administrator baza de date este cea facuta in mysql . Angajatul mai poate salva lista de incaltaminte sub forma unul fisier json , csv sau xml, doar la apsarea unui singur buton. O functie noua care a fost adaugata este cea de vizualizare de statistici unde acesta poate vedea anumite interese pe un pie chart sau un bar chart.

In continuare va voi prezenta diagrama de clase care a fost realizata dupa arhitectura model-view-viewmodel.

Diagram, engineering drawing

Description automatically generated

Diagrama Uml pe care o vedeti in poza de mai sus reprezinta diagrama de clase si este realizata dupa arhitectura model-view-viewmodel. Aceasta arhitectura seprata cele 3 componenta ale sale : model, view , viewmodel, comunicarea dintre ele facandu-se prin data binding.

In pachetul Model am realizat clasele Incaltaminte si IncaltaminteDisponibila care au toate atributele pe care pe poate avea un model de incaltaminte din zilele noastre, adica nume, producator , marime , pret , numar exemplare vandute , numar exemplare initiale. Clasa IncaltaminteMagazin defineste o lista de magazine , care la randul lor au in ele o lista de incaltaminte , fiecare cu atributele specifile lor.Avem clasa de user care are atributele unui utilizator: username, password si rol. Putem observa clasele IncaltaminteDao si UserDao. Aceste doua clase fac legatura dintre magazine si baza de date, respectiv dintre aplicatie si lista de utilizatori din baza de date. Cu ajutorul acestor clase se citeste datele din baza de date , se pot insera papuci , useri si se pot sterge de asemenea. Pentru a respecta patterul creational builder care creaza obiecte in mod indirect ave clasel BuilderUser si BuilderIncaltaminte. Clasele de persistenta pentru user si incaltaminte apeleaza funtiile de adaugare , stergere , update pentru baza de date. Clasele CsvSave, JsonSave, XmlSave vor ajuta la salvarea rlistelor de incaltaminte in formatele Csv, Json si Xml. Acestea se vor face simultan la apasarea unui singur buton.

Pachetul View reprezinta interfata grafica a acestei aplicatii. Dupa cum se poate observa avem doua clase care ajuta la afisarea statisticlor despre modelele de incaltaminte din baza de date. Pe langa aceste clase mai avem altele 3 . Clasa de LoginUser reprezinta interfata grafica cu care are de aface fiecare utilizator , iar acolo a fost implemetat butonul de login si metodele care fac posibila comunicarea cu viewmodel-ul. In clasa Administrator view este interafta grafica a administratorului, unde au fost implementate elementele de care acesta dispune pentru a face functiile de crud pe utilizatori si butoanele aferente. AnagajatView este clasa destinate angajatului , iar aici se gasesc implementate butoanele si metodele de care acesta are nevoie , ele comunicand cu viewmodel-ul.

Pachetul ViewModel se ocupa cu implementarea fiecarei functionalitati. Pentru fiecare a fost create cate o clasa separata in pachetul destinat tipului de utilizator pentru care a fost implemetata. Totodata avem aici 3 clase care au rolul de a face data binding si de a crea instante pentru interfata Icommand.

Descrierea aplicatiei

Graphical user interface, application

Description automatically generated

Aplicatia incepe cu aceasta interfata , cea de login. Aici utilizatorul va trebui sa isi introduca datele in campurile de username si password pentru a se putea loga. Daca logarea se va face cu success acesta va fi trimis in interfata special destinate lui, fiind ca si utilizatori doar doua tipuri: angajat, administrator. Aplicatia nu are niciun fel de restrictie pentru numarul de adminstratori, deci acestia se pot modifica cum doresc ei. Cel principal care i-a create pe restul poate si stres ulterior.

Ca si exemplu un administrator doreste sa se logheze. Acesta va fi directionat spre aceasta interfata, unde va putea face operatiile de Crud pe utilizatori. Pentru fiecare operatie acesta are cate un buton . Butonul de afisare trebuie apasat doar la inceput pentru a vizualiza ce dorim sa schimbam, dupa care afisarea se va face automat pentru restul functiiilor.

Graphical user interface, application

Description automatically generated

Daca un angajat se va loga va fi directionat spre aceasta interafata.

Graphical user interface, application, table

Description automatically generated

Aici angajatul va putea face operatiile Crud pe incaltaminte. La fel ca la administrator are cate un buton diferit pentru fiecare tip de operatie. Anagajtul poate aplica diferite filtre. El poate vedea toate produsele doar dintr-un magazin selectat, poate filtra toate produsele mai mici decat un pret selectat , poate cauta dupa nume sau producator un tip de incaltaminte , si poate afisa doar produsele care mai sunt disponibile. Dupa toate aceste functionalitati mai raman inca 2 butoane. Unul este pentru salvarea unor rapoarte in 3 formate diferite , iar celalt pentru a vizualiza statistici aplicate produselor , dupa anumite criterii.

Table

Description automatically generated

In poza de mai sus puteti observa baza de date pentru incaltaminte care este legata la aplicatie.

Aplicatia respecta arhitectura model-view-viewmodel. Aceasta tema m-a ajutat sa inteleg cat de mult poate sa ma ajute o structurare buna a codului.