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 indormatiile despre produsele magazinelor , dar poate si aplica anumite cautari pe ele. Un aspect important este acela ca aplicatia este create pe sablonul model-view-presenter.

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 folosinf ca si IDE Intellij , programand in limbajul java. Am respectat arhitectula model-view-presenter. 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 generated

Poza de mai sus reprezinta diagrama claselor 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 ce are la abza un fisier Xml ce continue toti utilizatorii.

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 pentru incaltaminte consta intr-un fisier Xml unde sunt stocate produsele. Angajatul mai poate salva lista de incaltaminte sub forma unul fisier json si csv , doar la apsarea unui singur buton.

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

Diagram, engineering drawing

Description automatically generated

Voi incepe sa explic in mare clasele folosite in aplicatie. Ca si structura voi explica pe rand pachetele model, view, presenter. Prima clasa din model este cea de user unde am definit atributele unui user, datele cu care se va loga el la aplicatie. PersistenaUser este clasa care se ocupa de operatiile care se fac pe userii din fisierul Xml. In clasa UserXml am implementat functiile care preiau , sterg , modifica, adauga userii in fisier. In model doar aceeste 3 clase sunt destinate utilizatorilor. Restul se axeaza pe incaltaminte. Clasa incaltaminte are atributele pe care un model de incaltaminte le are , pret , nume si producator. In aceasta clasa avem o subclasa numita INcaltaminteDisponibila in care sunt ateibutele specifice pentru un singur papuc , adica marime , numar de exemplare vandute si numar de exemplare intialiale, pe baza carora se va calcula disponibilitatea. Clasa IncaltaminteMagazin contntine lista de incaltaminte dintr-un magazine , aceasta la randul ei devenind o lista de magazine in clasa IncaltamintePersistenta. Aici se vor face functiile de obtinere, adaugare, stergere, update pentru incaltaminte. In clasa IncaltaminteXml se produc functiile pentru incaltaminte aplicate fisierului xml. Mai avem si cele doua interfete Csvsave si Jsonsave care vor salva lista de incaltaminte in cee doua formate specific.

In pachetul presenter sunt implementate funtiile pentru butoanele din View. Acesta este un pachet de legatura intre model si view. In PresenteUser am implementat functia pentru butonul de login care va face posibila mutarea in celelalte doua interfete. In clasa PresenterAdministrator se implementeaza functiile de afisare , creare, stergere si modificare pentru utilizatori. In clasa PresenterAngajat am implemetat functiile pentru Crud , dar si cele pentru cautare si filtrare pentru incaltamite, folosind functiile din persistenta destinate produselor de incaltaminte in care se fac aceleasi functii direct pe xml. Clasa PresenterFiltru a fost create special pentru implementarea cautarii unui produs care se incardreaza intr-un anumit interval de pret.

View ul se ocupa de inetrfata grafica a aplicatiei. El continue 4 clase: FiltruPret, LoginUser, AdminsitratorView, AngajatView si 4 interfete legate de cele 4 clase. In clasa LoginUser se afiseaza interafat de logare cu campurile de completat si butonul de login. Clasa de AdministratorView afiseaza interfata pentru administrator , iar acoloa dminstratorul va putea face operatiile de crud pe utilizatori. Clasele de AngajatView si FiltruPret sunt destinate angajatului pentru a putea realizata toate operatiile cerute.

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. Aplicatia nu are niciun fel de restrictive pentru numarul de adminstratori, deci acestia se pot modifica cum doresc ei.

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,.

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 with medium confidence

Aici angajatul va putea face operatiile Crud pe incaltaminte. Coloana din stanga este special destinata modificarii unui produs.

Aplicatia implementat a respectat arhitectura MVP , dar si modelul SOLID.