Student: Alexandra Veres

**Group:30234**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

Scopul acestei teme este crearea unei aplicatii care gestioneaza vanzarea de mobilier pentru diferiti vizitatori. In cadrul acestei aplicatii, vor exista doua tipuri de useri:

* Administratorul, care poate efectua urmatoarele operatii:
* Gestionarea de informatiile despre angajati( creare, citire, update, stergere);
* generare de raporturi despre un anumit angajat, continand activitatile desfasurate pe o anumita perioda de timp de catre un angajat.
* Regular user, care poate efectua urmatoarele operatii:
* Gestionarea de informatii despre comanda( adaugare, update, vizualizare);
* Crearea, stergerea, modificare informatiilor despre un produs;
* Adaugarea produselor la comanda si modicare valorii in functie de stoc.

**Constrangerile aplicatiei**:

* Datele vor fi stocate intr-o baza de date relationala.
* Utilizarea pattern-ului architectural Layer pentru organizarea aplicatiei.

**Cerinte:**

* Crearea documentului de analiza si proiectare.
* Implementarea si testarea aplicatiei.

# Functional Requirements

Aplicatia este destinata tuturor persoanelor care doresc sa vizualizeze anumite noutati legate de mobilier( produse noi, preturi, stoc), precum si celor care doresc sa cumpere mobilier online fara sa efectueze vreo deplasare.

Pentru a se putea testa toate functionalitatile aplicatiei este nevoie ca persoanele care o vor folosi sa aiba un cont(username si parola), pe care il vor folosi pentru a se loga, deoarece pentru a putea cumpara mobilier user-ul trebuie sa se logheze.

Aplicatia are doua tipuri de useri:

1. **Administratorul aplicatie** care se poate loga folosind un username si o parola si poate sa realizeze anumite operatii specifice doar lui precum CRUD pe informatii referitoare la angajati sau generarea de raporturi privind activitatea acestora:

* Introducerea in baza de date a informatiilor referitoare la un anumit angajat
* Poate sa stearga datele referitoare la un anumit angajat
* Poate sa vizualizeze intr-un table toate informatiile stocate despre un anumit angajat
* Poate modifica informatiile referitoare la un anumi

1. **Userul “normal”** care se poate loga folosind tot un username si o parola si care poate realiza urmatoarele operatii:

* Adauga in baza de date informatii despre comanda/ produs
* Modifica in baza de date informatii despre comanda/produs
* Stergere produs
* Vizualizarea intr-un tabel, atat a informatiilor retinute in baza de date despre comanda, cat si despre produs
* Poate modifica informatiile despre produs

# Non-functional Requirements

Aplicatie trebuie să aibă propietatea de a fi portabila, sa poata rula pe calculatoare diferite si pe sisteme de operare diferite.

Aplicatia trebuie să fie stabila, sa poata fi întretinută usor.

Robustness - aplicatia trebuie sa acopere toate cazurile specificate.

2. Use-Case Model

**Diagrama de utilizare** este un tip de diagrama din care reiese modul de utilizare a sistemului informatic - modul în care utilizatorii interactionează cu acesta. Diagrama de utilizare nu este absolut necesara pentru a scrie o specificatie cu use case-uri dar este utila pentru a crea o imagine generala asupra sistemului. Un actor reprezintă un rol jucat de o persoana sau o entitate care interactioneaza cu sistemul. Cum actorii reprezinta utlizatorii sistemului, ei ajuta la delimitarea sistemului si ofera claritate in ceea ce se va intampla in respectivul sistem. Din cerinta problemei deducem ca aplicatia poate fi folosita de catre administratorul aplicatiei care administreaza evenimentele pentru a optimiza vizualizarea datelor de catre client/utilizatori, si de catre useri care acceseaza aplicatia pentru a vizualiza evenimentele viitoare si pentru a cumpara diverse mobiliere.

Use case: login

Level: user-goal-level

Primary actor: administrator, utilizator normal

Main success scenario: Utilizatorul se autentifica cu succes prin furnizarea unui username si a unei parole corecte.

Extensions: Utilizatorul incearca sa se autentifice cu username gresit sau parola gresita.

**Use case:** **Adaugarea unui nou angajat**

**Level:** **Administrator**

**Primary actor:** **Administator**

**Main success scenario:** Pentru a adauga un nou angajat este necesar ca administratorul sa se logheze folosind un username si o parola. Astfel, se va introduce usernaname-ul si parola si se va verifica daca aceste date corespund administratorului, caz in care se va deschide pagina destinata adminstratorului. In aceasta pagina administratorul va alege butonul de adaugare a unui nou angajat. Prin actionarea acestui buton, se va deschide o noua fereastra unde el va trebui sa introduca datele necesare despre angajat( first\_name, last\_name, sex, age). Dupa introducerea acestor date se actioneaza butonul “Adaug”, actiune care va introduce in baza de date noul angajat.

**Extensions:** In cazul in care adminstratorul nu completeaza corect un anumit camp, la apasarea butonului **Adauga** se va semnala o “eroare” specificata prin aparitia unui mesaj specific.

**Use case:** Inregistrare

**Level:** Utilizator nou

**Primary actor:** Utilizator normal

**Main success scenario:** pentru a crea un cont nou, se va selecta butonul de **“Register”**. Se va deschide o noua ferestra, unde sunt disponibile campuri in care se vor introduce detaliile necesare pentru crearea unui cont nou, care va fi stocat in baza de date. Utilizatorul trebuie sa introduca numele, prenumele, un username si o parola.

**Extensions:** un caz in care crearea unui cont nou poate esua este cel in care utilizatorul uita sa completeze un camp sau il completeaza gresit, ceea ce va genera un mesaj de eroare specific. De asemena, e posibil sa nu introduca un camp valid, caz care fi din nou semnalat printr-un mesaj.

**Use case:** Vizualizarea informatiilor

**Level:** Utilizator

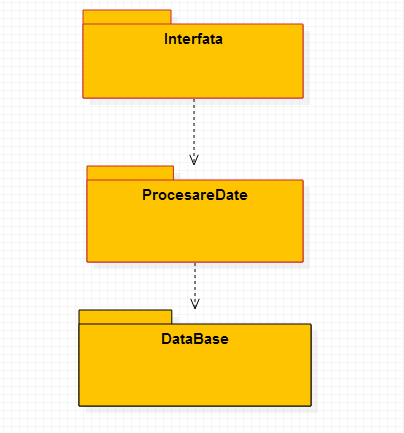
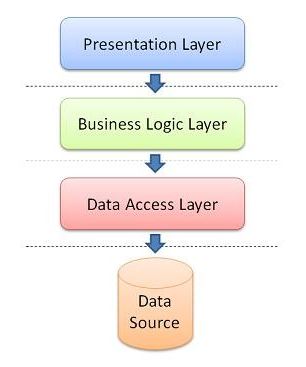
**Primary actor:** Utilizator

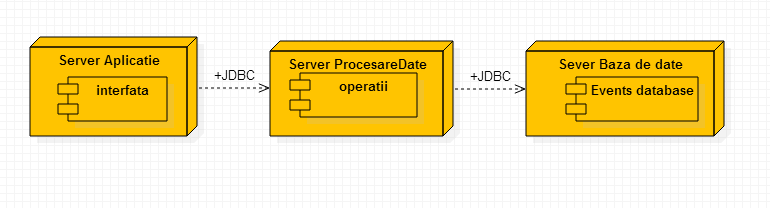
**Main success scenario:** pentru cei care doresc doar sa vada informatiile dorite, in principal exista butonul **“View”** , la apasarea caruia va aparea o lista cu toate informatiile dorite(title, description, color, size, price, stock etc).

3. System Architectural Design

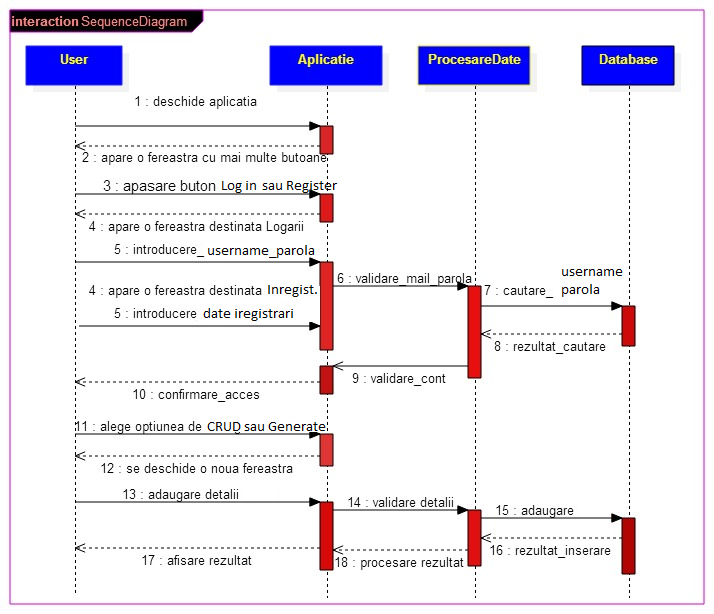
**3.1 Architectural Pattern Description**

**3.2 Diagrams**

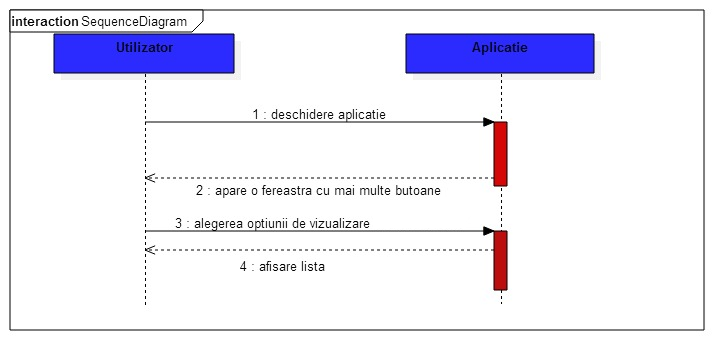
 Deployment diagram

4. UML Sequence Diagrams

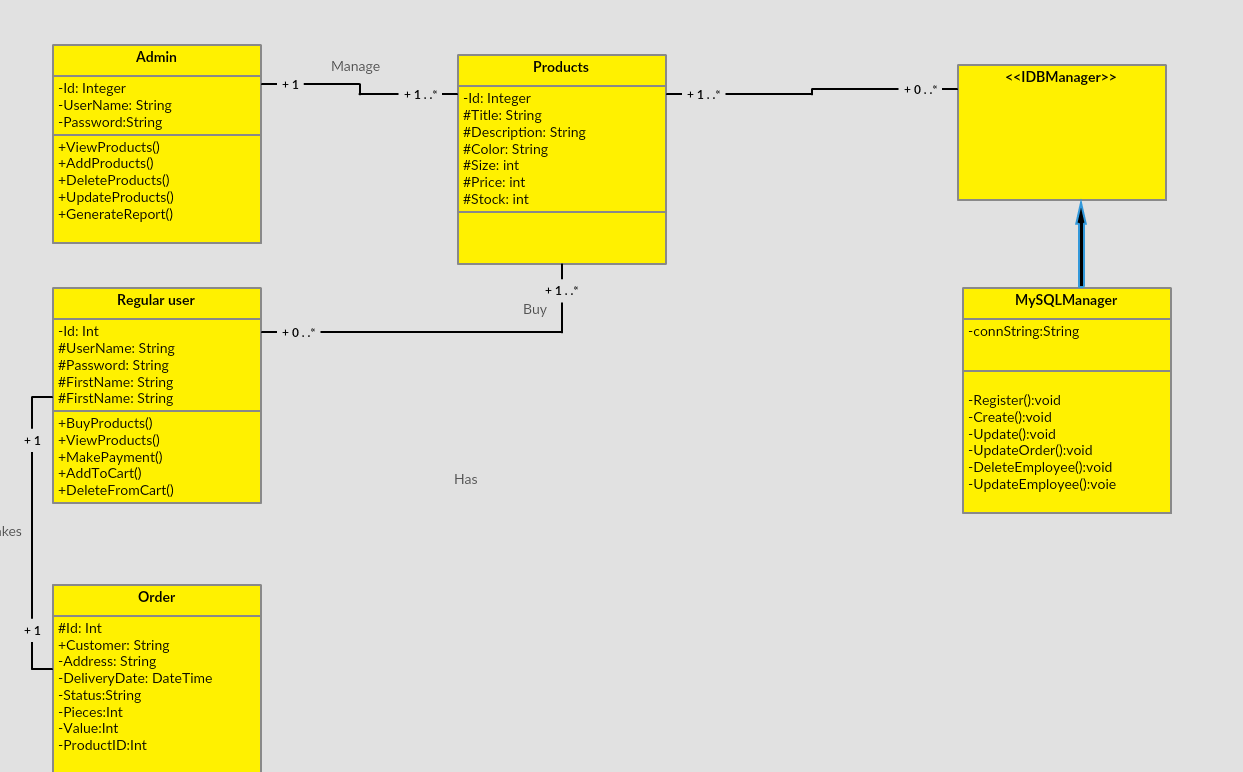
* **Diagrama de secventa corespunzatoare pentru cazul in care se adauga adauga noi obiecte in baza de date:**



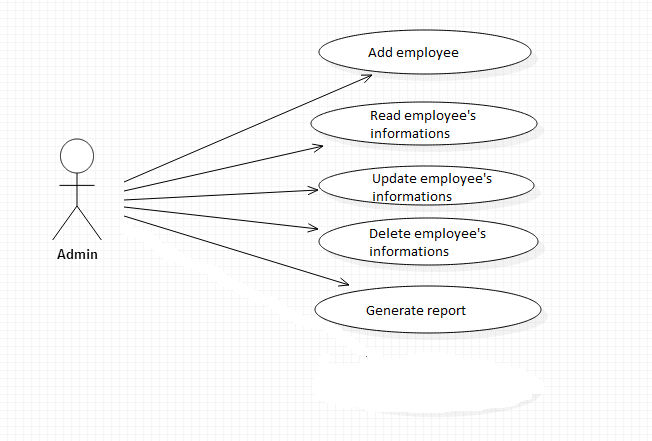
* **Diagrama de secventa corespunzatoare cazului cand user-ul adauga date in baza de date sau vizualizeaza informatii**

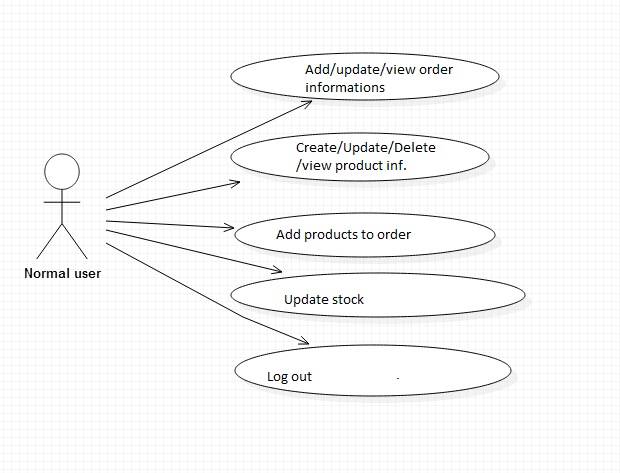


5. Class Design



**5.2 UML Class Diagram**





6. Data Model

Baza de date folosita in aplicatie se numeste **assigment1** si este alcatuita din urmatoarele tabele:

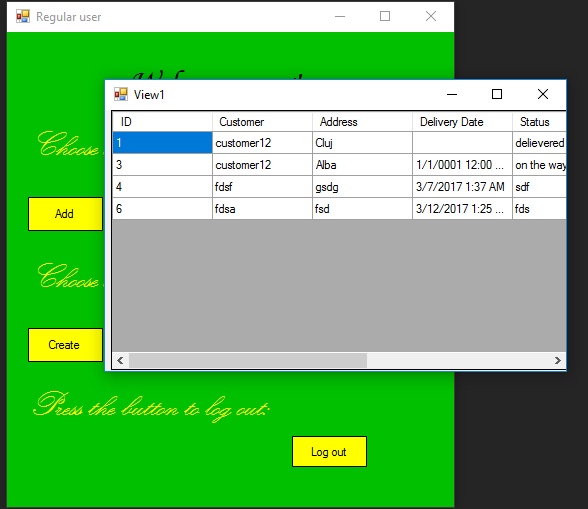
* Tabela **product** care contine informatii despre produsele aplicatiei(**id**, title, description, color, size, price, stock). Id reprezinta cheia primara a tabelei.
* Tabela **user** contine informatii despre client(**idUtilizator**, numeU, prenumeU, parola). idUtilizator reprezinta cheia primara a tabelei.
* Tabela **employees** contine informatii despre angajatii firmei si are urmatoarele campuri***: id, first\_name, last\_name, sex, age***.

7. System Testing

In urma simularii aplicatiei se observa ca simularea decurge fara probleme de implementare. Cand lansam aplicatia utilizatorului ii va aparea urmatoarea fereastra :



Daca se doreste, vizualizaea datelor in tabel, vor fi disponibile urmatoarele ferestre de vizualizare:



8. Bibliography

* Cursul de Software Design, MihaelaDinșoreanu
* Patterns of Enterprise Application Architecture, Martin Fowler
* http://martinfowler.com/eaaCatalog/tableDataGateway.html