Student: Tomuș Alexandra

**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

Utilizați Swing / C # API pentru a proiecta și implementa o cerere pentru managerii de comandă ale unui producător de mobilier. Cererea ar trebui să aibă două tipuri de utilizatori (un utilizator obișnuit reprezentat de comandă manager și un utilizator administrator) care trebuie să furnizeze un nume de utilizator și o parolă, în scopul de a utiliza aplicația.

# Functional Requirements

Utilizatorul regulat poate efectua următoarele operații:

- Adăugați / actualizare / vizualizare informații despre comenzi (client, adresa de livrare, numărul de identificare, data de livrare, starea.).  
- Creare / modificare / ștergere / vizualizare informații despre produs (titlu, descriere, culoare, dimensiune, pret, stoc, etc).  
- Adauga produse la comandă și să actualizeze valoarea comenzii și a stocului în consecință

Utilizatorul administrator poate efectua următoarele operațiuni:

- CRUD privind angajații  
 - Generarea de rapoarte pentru o anumită perioadă care conține activitățile desfășurate de către un angajat.

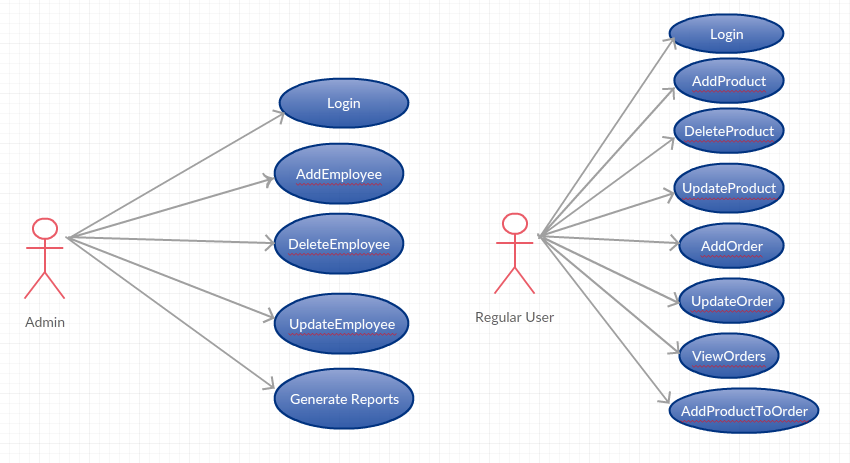
# Non-functional Requirements

Utilizabilitatea este cea mai importanta cerinta non-functionala, de aceea am creat o interfata simpla, care are butoane explicite astfel incat fiecare utilizator sa se poata usor familiariza cu programul.

Disponibilitatea este si ea o cerinta non- functionala importanta, de aceea singura conditie pentru ca programul sa ruleze este ca baza de date sa fie pornita.

2. Use-Case Model

Un model de utilizare constă dintr-un număr de elemente de model. Cele mai importante elemente de model sunt: cazuri de utilizare, actori și relațiile dintre ele.  
 Un actor este reprezentat de administrator, care face operatii CRUD pe informatia angajaților și genereaza rapoarte pentru o anumită perioadă care conține activitățile desfășurate de către un angajat.  
 Un alt actor este utilizatorul obișnuit, care se poate sa faca operatiile de adaugare / actualizare / vizualizare informații despre produse si deasemnea poate sa adauge produse la comanda, sa actualizeze valoarea comenzii si a stocului.

**

3. System Architectural Design

**3.1 Architectural Pattern Description**

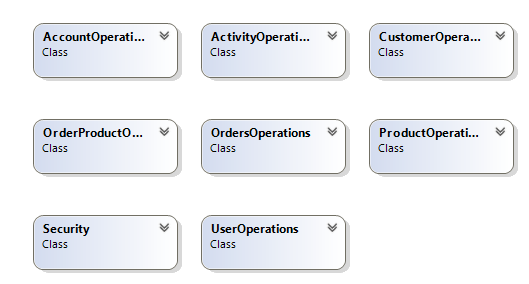
Am utilizat un model de proiectare bazat pe mai multe nivele, separate intre ele. Modelul este bazat pe 3 nivele:

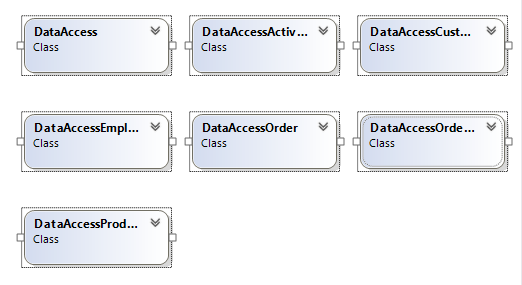
* **Presentation tier**
* **Business Logic tier**
* **Data tier**

**Presentation tier:**

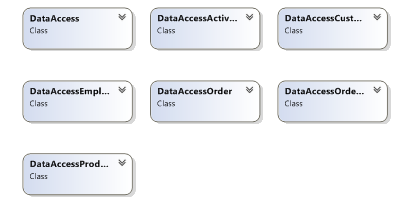
**

**Business Logic tier:**





**Data tier:**

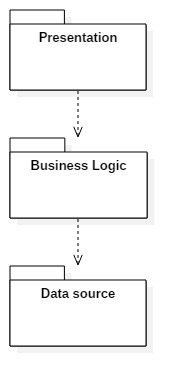
**

**Model:**

****

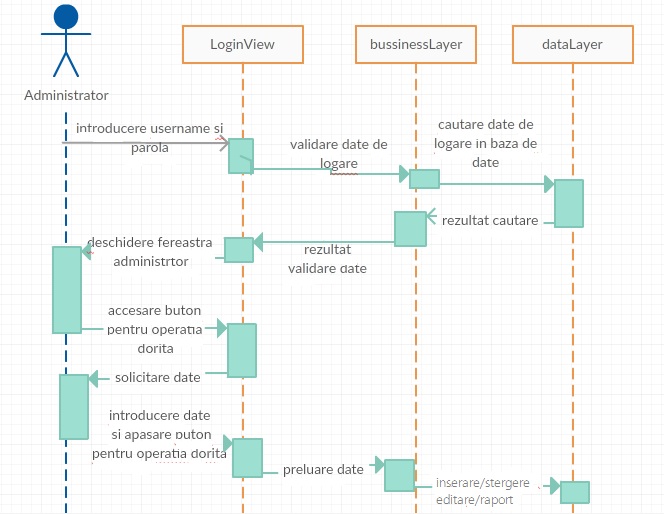
**3.2 Diagrams**

**Package diagram**

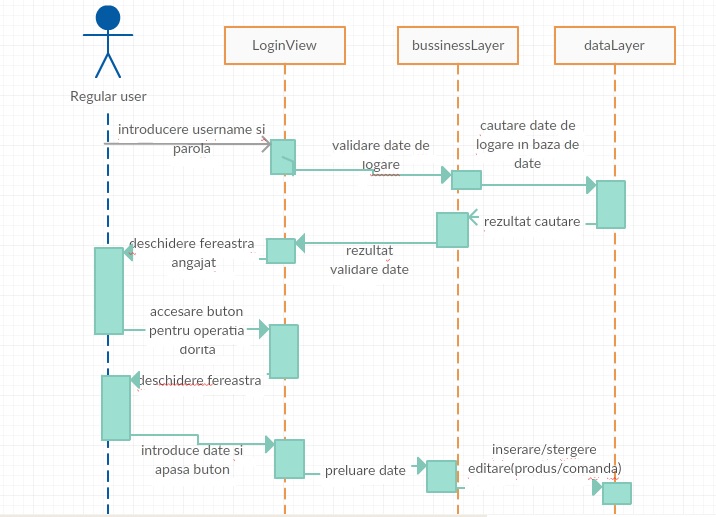
****

4. UML Sequence Diagrams

**Administrator:**



Regular user:



5. Class Design

**5.1 Design Patterns Description**

***Am utilizat 3 nivele de design pattern :***

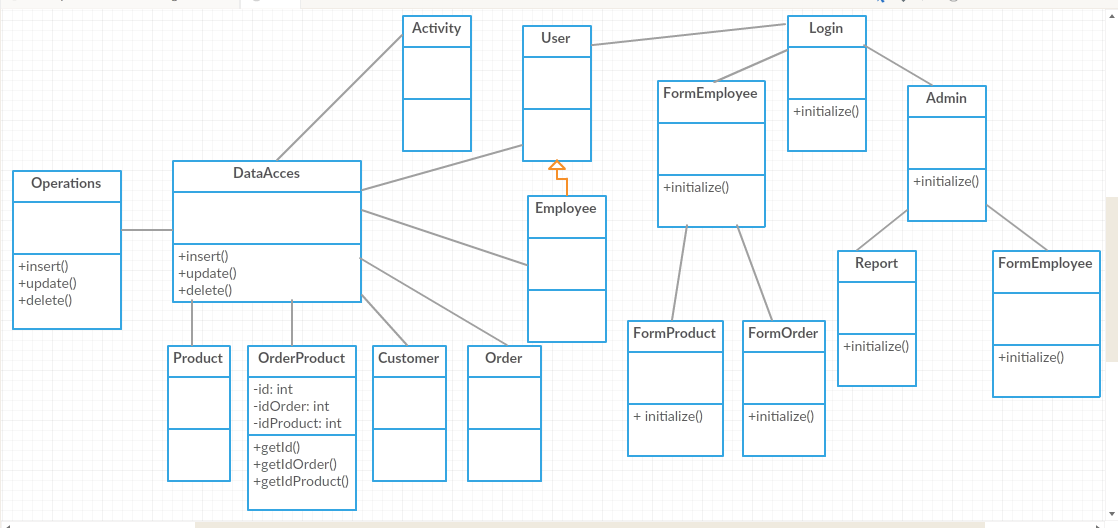
* **Presentation tier:** este reprezentat de interfata utilizator expusa de program.
* **Business Logic tier:** este reprezentat de codul aplicatiei, implementeaza funcționalitatea de baza a sistemului și încapsulează logica.
* **Data tier:** stocheaza datele manipulate de program ( pentru stocare am utilizat baza de date).

DAL (Data access layer) ofera acces simplificat la datele stocate in baza de date. Acest tip de model ar putea fi pus in aplicare prin crearea unei clase de acces la date care face referire in mod direct la un set de proceduri din baza de date.

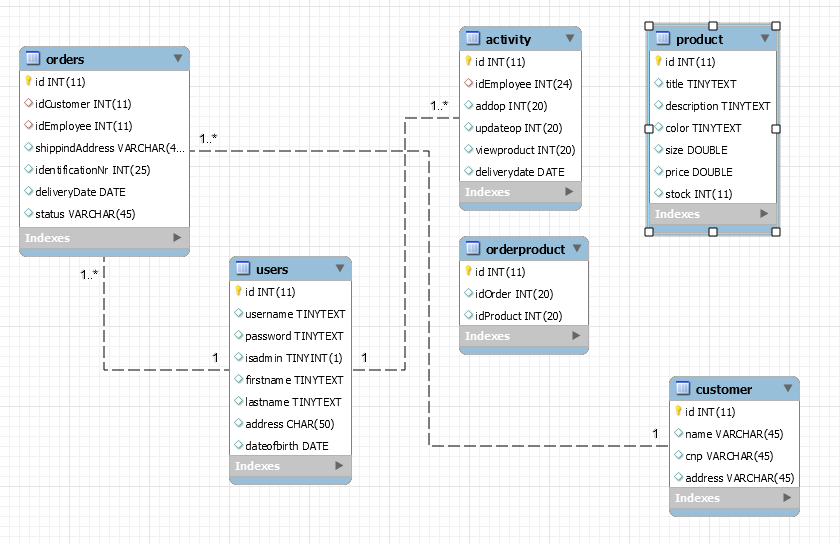
***Domain model design pattern :***

* Modelul reprezinta un obiect
* Poate contine logica de a actualiza controlerului

**5.2 UML Class Diagram**

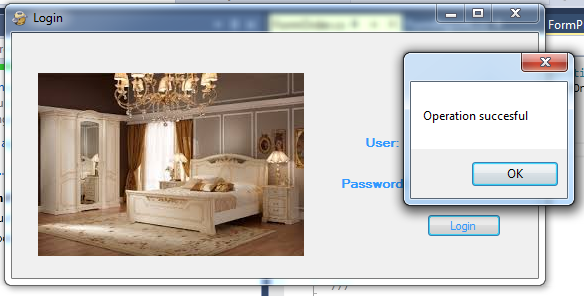


6. Data Model

**

7. System Testing

Pentru a testa aplicatia, am creat dialoguri cu mesaje care vor aparea in interfata cu scopul de a anunta utilizatorul ca operatia efectuata este corecta sau incorecta. Pe langa aceste teste am efectuat UnitTests pentru a verifica corectitudinea metodelor.



8. Bibliography

1. <https://www.codeproject.com/Articles/36847/Three-Layer-Architecture-in-C-NET>
2. <https://www.codeproject.com/Tips/662107/Understand-Tier-Architecture-in-Csharp>
3. <https://en.wikipedia.org/wiki/Data_access_layer>