**Documentatie Assigment 1**

**Student:Popa Ionut**

**Group:30233**

***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 4

3. System Architectural Design 5

4. UML Sequence Diagrams 6

5. Class Design 7

6. Data Model 7

7. System Testing 8

8. Bibliography 8

***1. Requirements Analysis***

1.1 Assignment Specification

Aplicatia implementata este una pentru gestionarea studentilor din cadrul departamentului de Computer Science al facultatii UTCN.Aceasta are doua tipuri de utilizatori,utilizatorii de tip student si utilizatorii de tip profesor.Fiecare dintre acestia trebuie sa utilizeze un user si o parola.

1.2 Functional Requirements

Utilizatorul normal, adica studentul, are acces la urmatoarele operatii :

-Adaugarea, editarea, vizualizarea informatiilor de utilizator ( nume, numar card de identitate, cod numeric personal, adresa, etc)

- Crearea, editarea, stergerea, vizualizarea profilului de student ( informatiile contului : numar de identificare, grupa, inrolari, note)

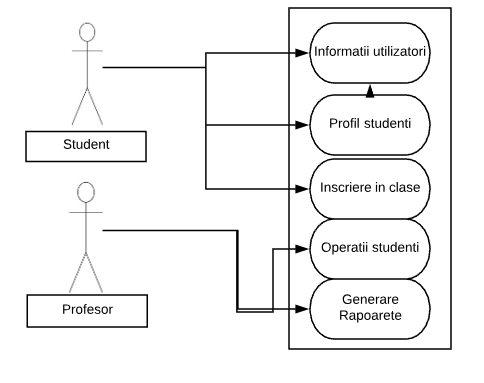
-Inrolarea la clase ( inrolare, examene, note)

1.3 Non-functional requirements

*-Descrie modul de functionare in timp ce functional requirements descrie ce ar trebuii sa faca programul*

*-Printre acestea se numara viteza de executie,securitatea si inregistrarea datelor in baza de date*

***2. Use-Case Model***



**Use case description:**

* Use case: Specificarea fiecarui user ce operatii poate face
* Primary actor: Profeso sau Student
* Main success scenario: Se selecteaza tipul de utilizator,urmat de user si parola.Acesta selecteaza tipul de operatii pe care vrea sa il efectueze.

***3. System Architectural Design***

***3.1 Architectural Pattern Description***

*In creearea aplicatiei am folosit doua tipuri de pattern-uri:*

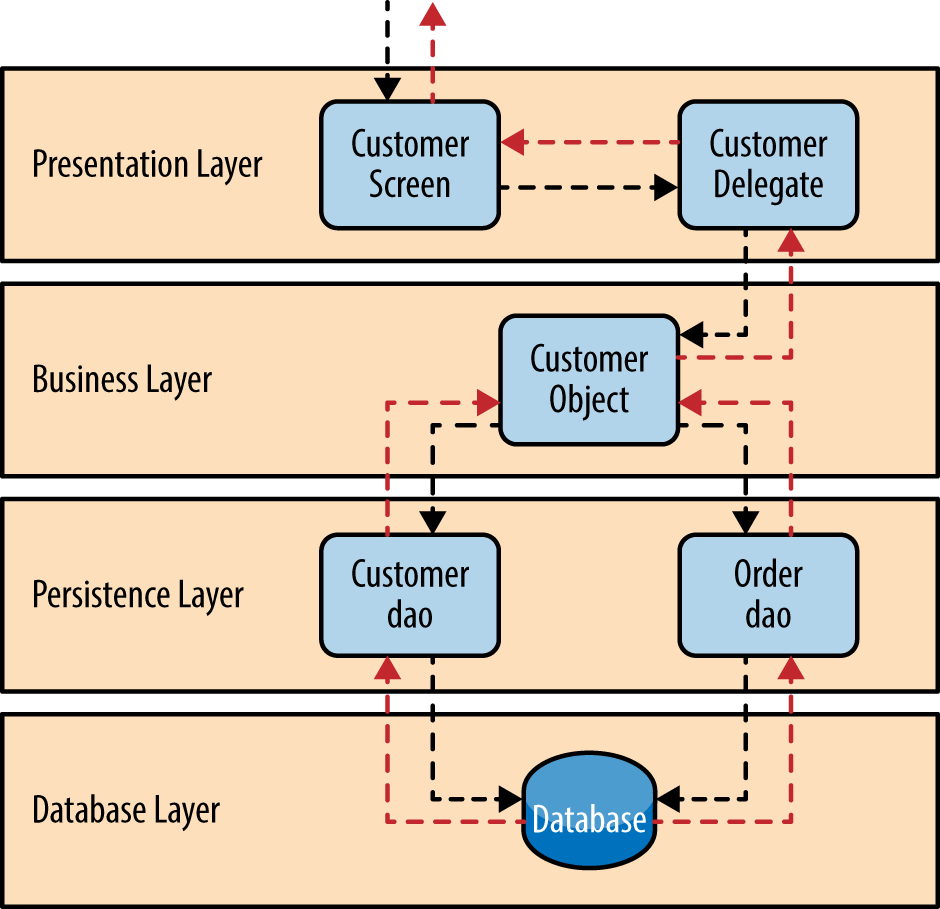
*1.Service layer pattern:Folosit pentru structurarea verticala a programului.Layerele folosite sunt:*

* *Presentation layer*
* *Bussines layer*
* *Persistence layer*
* *Database layer*

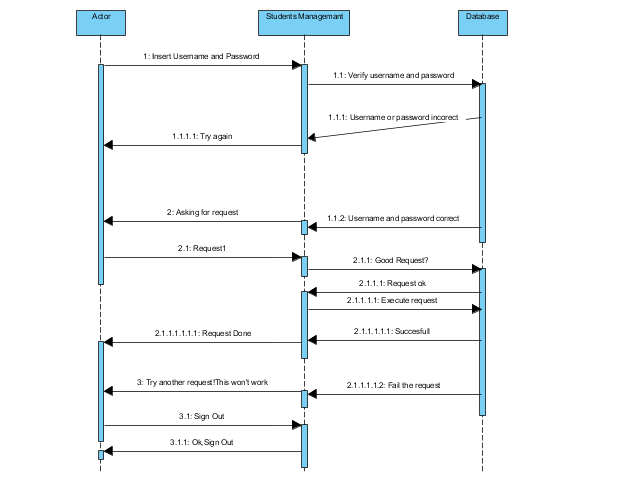
*Cateva atributii ale acestui layer sunt :*

* *Ease to develepment: O aplicatie construita cu layers pattern este usor de implementat,datorita acestui pattern.Este un pattern folosit de multe firme,deoarece proiectul se poate structura pe partea de Presentation,Bussines si Database.*
* *Easy to test*
* *Agility on overall is low*

***3.2 Diagrams***



***4. UML Sequence Diagrams***



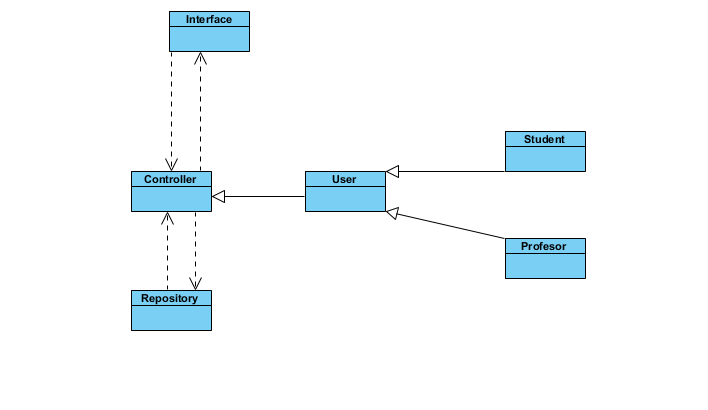
***5. Class Design***

***5.1 Design Patterns Description***

Pentru acest proiect,am ales sa folosesc MVC si Observer Pattern.

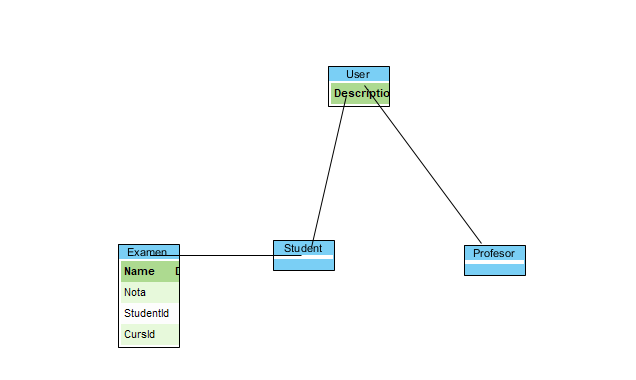
* MVC-ul este folosit pentru a putea impartii aplicatia pe mai multe parti.Partea de Model,reprezentata de obiecte,partea de View-partea de vizualizare de date si partea de controller ,partea in care se executa toate operatiile si modificarile asupra obiectelor din model.
* Observer pattern :Observarea modificarilor obiectelor.Observarea momentelor cand un profesor updateaza nota unui student,sau cand are nevoie de toate notele studentilor pentru a creea un raport al intregii clase

***5.2 UML Class Diagram***



Partea de atribute si metode o voi implemeta in timpul efectuarii proiectului.

***6. Data Model***



***7. System Testing***

***8. Bibliography***

* <https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html>
* <https://www.lucidchart.com/>
* Visual Paradigm