Student:Savan Catalin

**Group:30235**

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

Realizarea unei aplicati pentru o agentie de turism , aceasta are ca functionalitati , adaugare de client , rezervari si in cazul unei modificari introducerea intr.o tabela istoricul actiunilor. Adminul efectuaza operati CRUD pe useri .

# Functional Requirements

Autentifiare : pentru a utiliza aceasta aplicatie este necesara logarea in interfata de login , in urma tipului de user se vor deschide o anumita interfata : una de user sau una de admin. In urma audentificarii persoana logata poate face actiuni specific tipului de user ce s.a logat.

Interfata externa : In acest proiect sunt prezente 3 interfete : logare , user si admin . In urma logarii se deschide una din cele 2 interfete in functie de tipul utilizatorului . Interfata de user permte adaugarea de date pentru client noi / modificarea datelor pentru client / adaugare de noi rezervari si nu in ultimul rand vizualiarea datelor din baza de date si anume cele de rezervari respective client.

Interfata Admin permine gestionare useri-lor si verificare datelor din istoric unde se afiseaza id.ul userului care a facut modificare cat si ce modificare a facut acesta .

Date istoric:In urma unor modificari facute de useri , intr.o tabela de date se vor retine date despre modificarea adusa bazei de date cum ar fi adaugarea de client / adaugarea de rezervari etc. Aceste date sunt vizibile doar de catre admin care urmeaza sa gestioneze aceste date .

Securitate : Doar in urma logarii cu o parola se vor putea accesa datele din baza de date , asadar doar o persoana cu cont existent sa gestioneze baza de date in functie de tipul utilizatorului.

Specificarea cerintelor/Afisarea de informatii: Afisarea datelor din baza de date se face in tabelele prezente in interfata grafica doar in urma apasarii unui buton dedicate pentru aceasta actiune .

# Non-functional Requirements

Accesabilitate : proiectul poate fi accesat doar de utilizatori prezenti in baza de date .

D[evelopment environment](https://en.wikipedia.org/wiki/Development_environment) : proiectul a fost realizat cu ajutorul limbajului de programare Java in Eclipse, iar baza de date este realizata utilizant MySQL workbrench .

Security: Securiatetea proiectului este data de sistemul de logare cat si de validarea datelor.

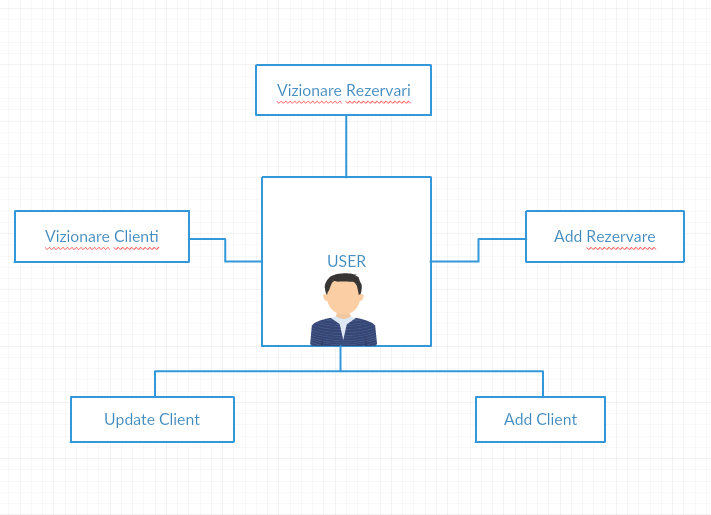
2. Use-Case Model

**Descriere:**

Userul reprezinta angajatul care foloseste aplicatia pentru a gestiona rezervarile , clienti si totodata adaugare si modificare datelor despre clienti , acesta

**Pre-condiţii şi post-condiţii:**

Utilizatorul trebuie doar sa acceseze site.ul astfel reusind sa realizeze comenzi și a analiza produsele disponibile.

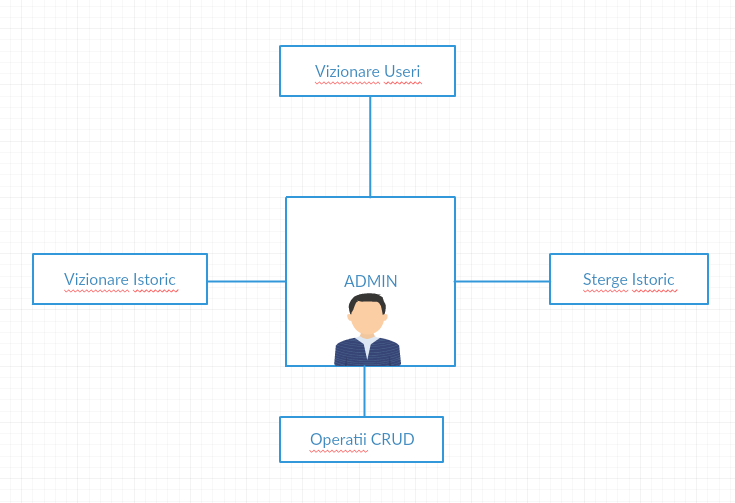


**Descriere:**

Administratorul va avea acces la baza de date și informațiile despre useri si istoric actiunilor facute de acestia , aceste date sunt afisate intr.o tabela unde se vad datele modificarii.

**Pre-condiţii şi post-condiţii:**

Adminul are nevoie de accesul la aplicatie si un cond introdus in baza de date , datorita tipului sau de utilizator va accesa alta interfata grafica fata de user.



3. System Architectural Design

**3.1 Architectural Pattern Description**

Arhitectura folosita la acest proiect este una pe layere acest lucru fiind realizat prin cele 4 nivele mare si anume :

-The presentation layer :este present prin interfata grafica unde utilizatori acceseaza layere urmatoare .

-The application layer : acest layer prezinta controller-ul proiectului care valideaza date care urmeaza sa fie trimizi la urmatorul layer.

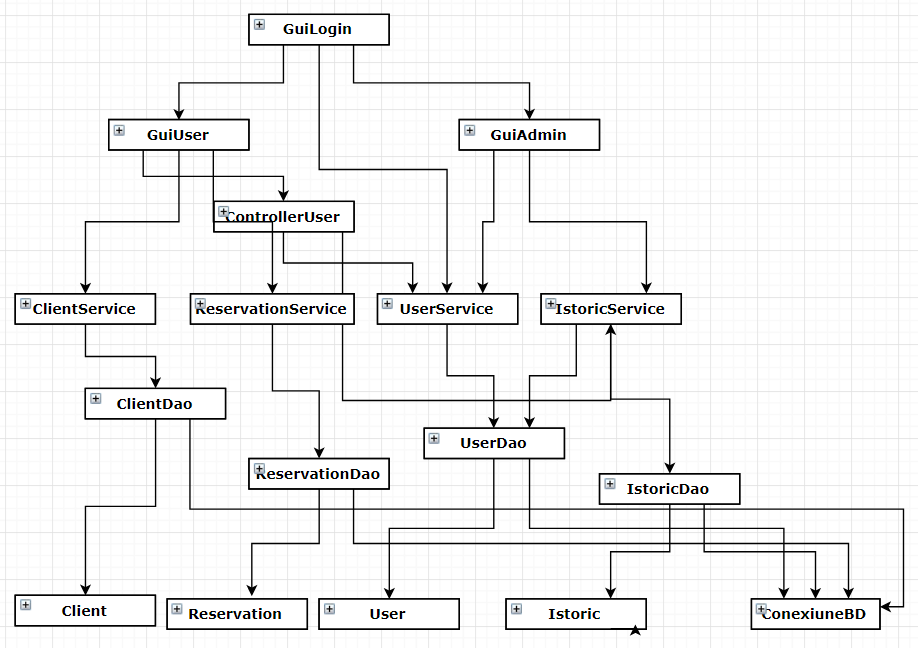
**-The domain layer** :realizat prin clasele de service acest layer reprezinta logica din sptele programului accesand datele de la nivelul urmator .

-The persistence layer: in acest layer sunt accesate datele din baza de date si care urmeaza sa fie procesat la layerele superioare , doar in acest nivel se face conectarea la baza de date si extragerea datelor din aceasta

**3.2 Diagrams**

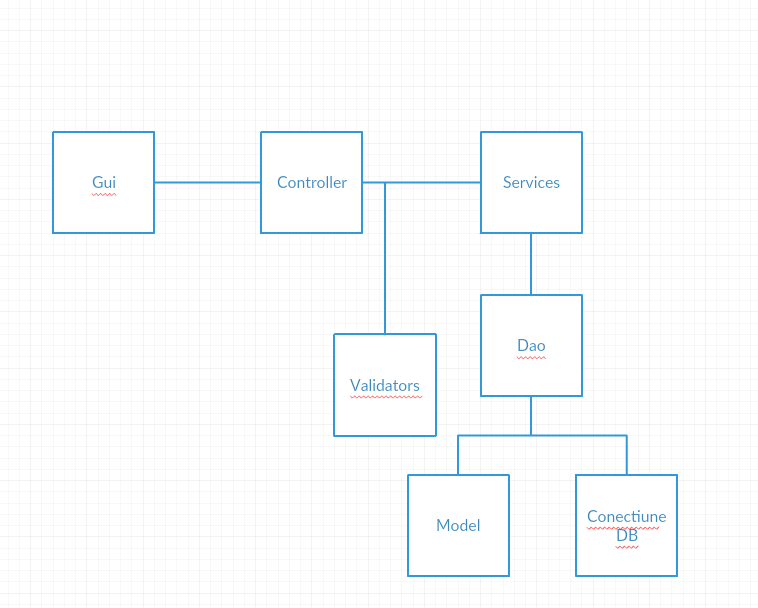
*COMPONENT DIAGRAM*

*DAO-DataAccesOnly*

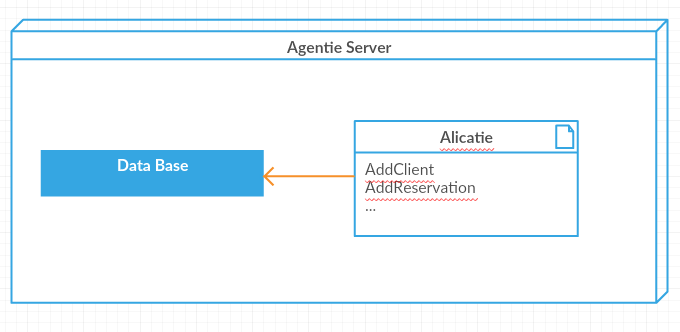
**

*PACKAGE DIAGRAM*

*DAO-DataAccesOnly*

**

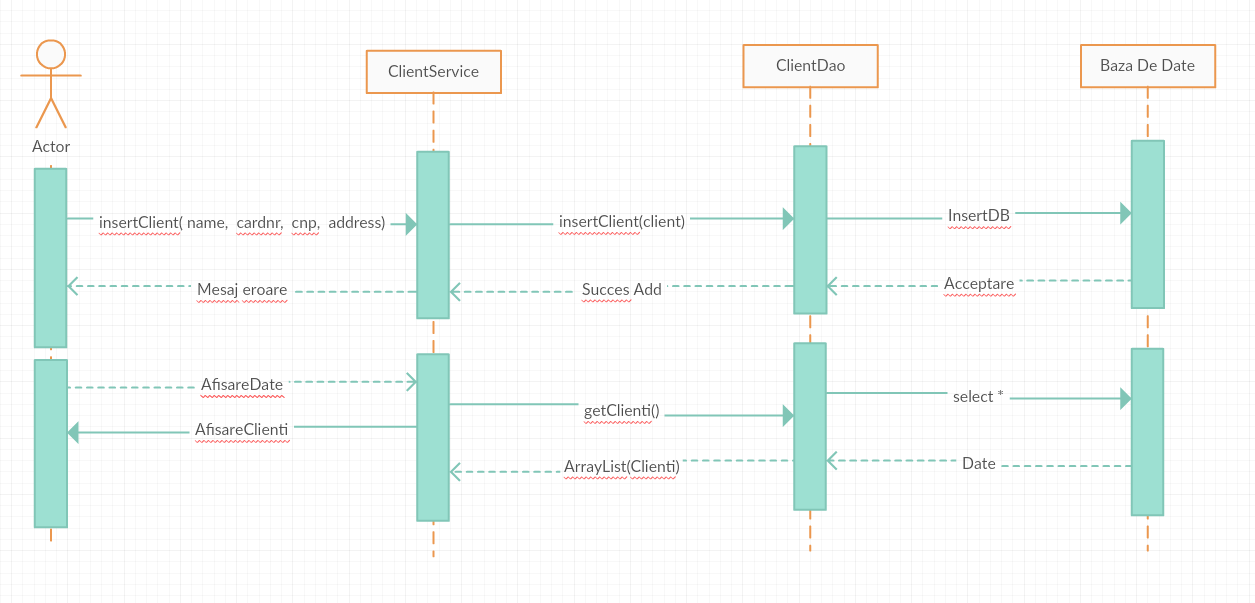
*DEPLOYMENT DIAGRAM*



4. UML Sequence Diagrams

Diagrama de secventa pentru un scenario normal al aplicatiei addClient / addReservation .

*DAO-DataAccesOnly*



5. Class Design

**5.1 Design Patterns Description**

Design pattern-ul folosit pentru realizarea acestui proiect este Domain Model deoarece datele sunt prelucrate chiar de clasele ce definesc obiectul, aceste obiecte sunt instantiate mai departe pentru prelucrarea datelor din baza de date.

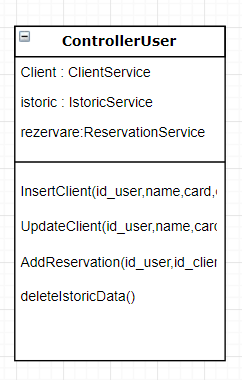
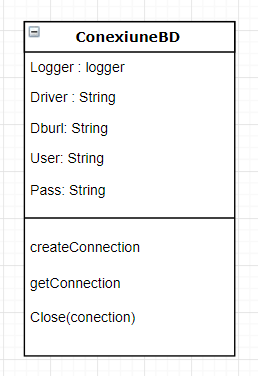
Modelul de date este Table Data gateaway , prin acest pattern am creat clase care se ocupa strict cu preluarea datelor din baza de date acestea si anume clasele *DataAccesOnly(DAO)*

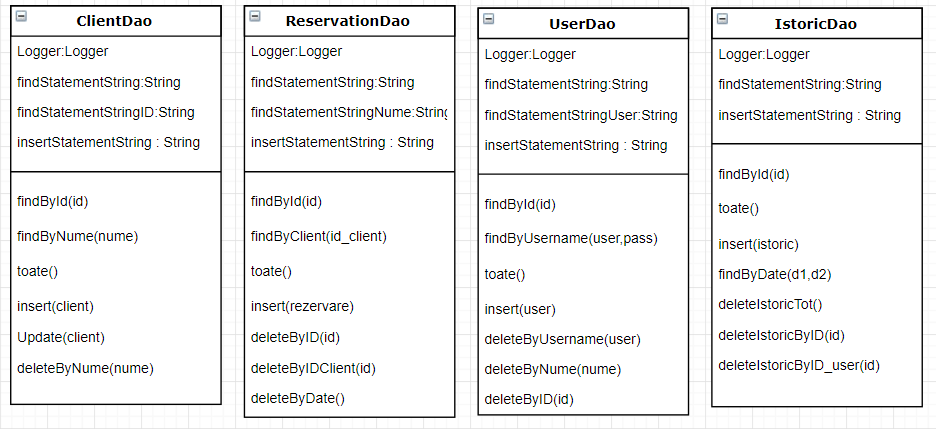
,dar datele fiind procesate mai departe de clasele Service.

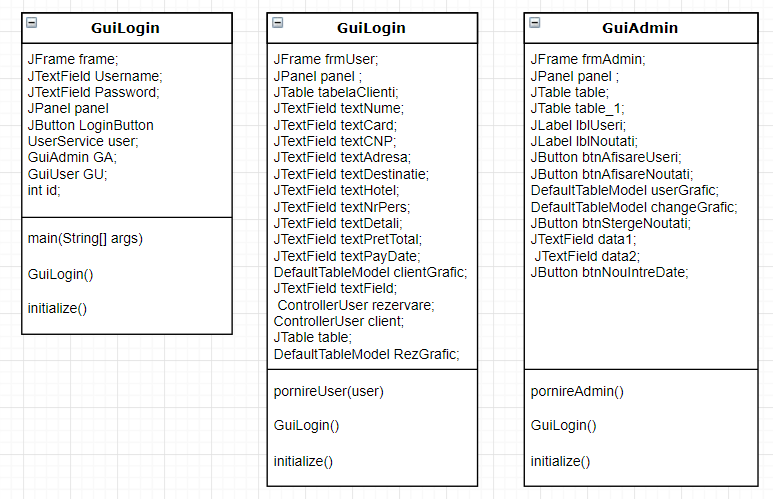
Datele prelucrate sunt dupa aceea afisate pe interfata grafica pentru utilizator inregistrat fiecare avand un anumit drept , acest drept la logare este luat tot din baza de date care difera de la fiecare tip de utilizator.

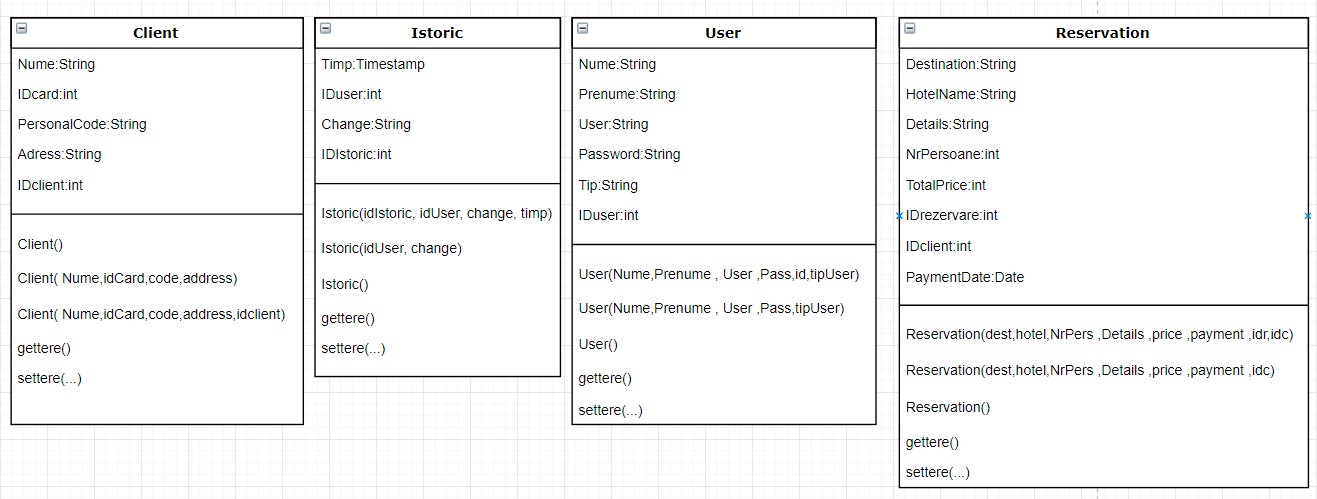
**5.2 UML Class Diagram**

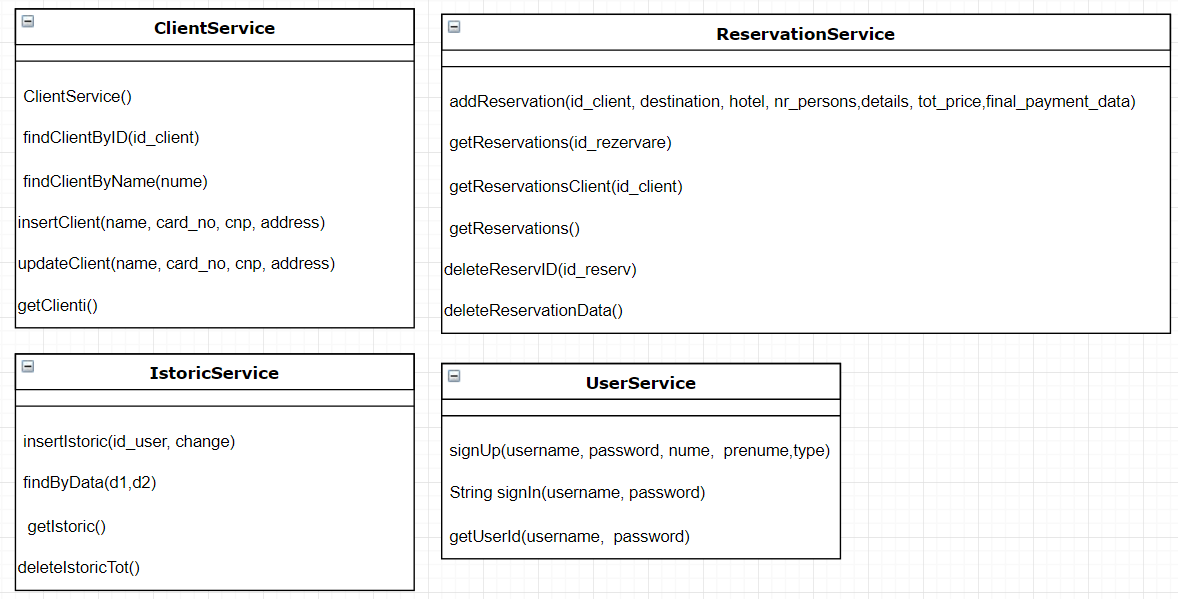
*[Create the UML Class Diagram and highlight and motivate how the design patterns are used.]*

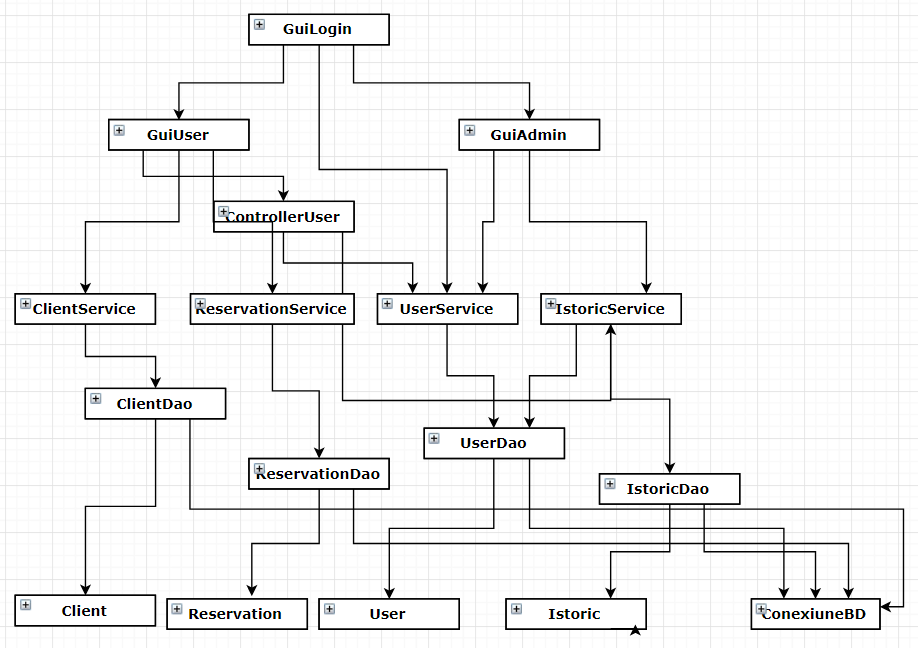
**

**

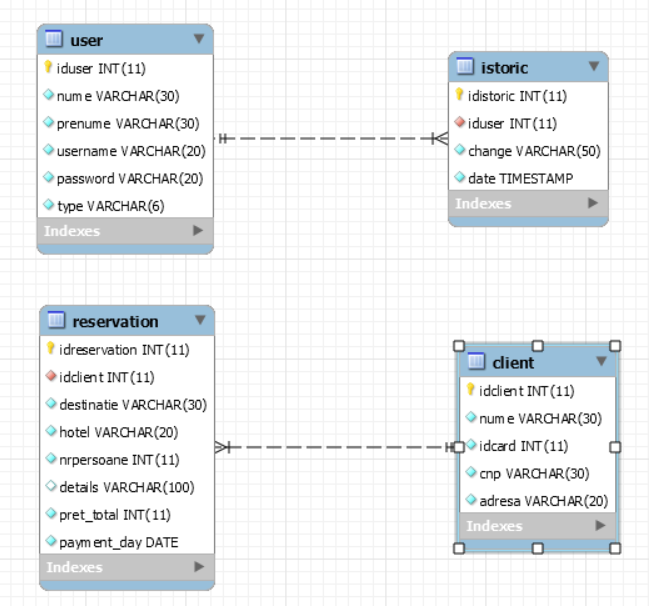
**

**

**

**

6. Data Model

**

7. System Testing

Testul sistemului a fost realizat cu JUnit , un test simplu pentru verificarea clientului , in urma inserari in baza de date a unui client verificam daca datele corespund cu cele introduce in exemplu, din fericire testul a iesit unul pozitiv asa ca am prelucrat datele mai departe ☺ .

Testarea aplicatiei s-a facut prin diferite scenari cum ar fii logarea , adaugarea client , rezervari , modificare si nu in ultimul rand afisarea datelor in urma modificarilor.

8. Bibliography

<http://docs.oracle.com/javase/tutorial/uiswing/>

<http://docs.oracle.com/javase/tutorial/jdbc/basics/index.html>

<https://msdn.microsoft.com/en-us/library/54xbah2z(v=vs.110).aspx>

<https://msdn.microsoft.com/en-us/library/e80y5yhx(v=vs.110).aspx>

<https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm>

<https://en.wikipedia.org/wiki/Data_model>