Text

Description automatically generated

**Facultatea de Automatică și Calculatoare**

**Departamentul calculatoare**

**Lanț hotelier**

Disciplina: Proiectare Software

Student: Tamas Paula-Irina

Specializare: Master Complemente de Știința Calculatoarelor

An școlar: 2023-2024

Cuprins

[Enunțul problemei 3](#_Toc163814148)

[**Partea I –Analiză** 4](#_Toc163814149)

[Introducere 4](#_Toc163814150)

[1.1 Obiective generale 4](#_Toc163814151)

[1.2 Modelul Use-Case 6](#_Toc163814152)

[**Partea II –Proiectare** 8](#_Toc163814153)

[2. Proiectare 8](#_Toc163814154)

[2.1 Arhitectura sistemului 8](#_Toc163814155)

[**Partea III – Implementare** 12](#_Toc163814156)

[3. Implementare 12](#_Toc163814157)

[3.1 Interfata 13](#_Toc163814158)

[4. Concluzii 16](#_Toc163814159)

# Enunțul problemei

Problema 16

Dezvoltați o aplicație care poate fi utilizată de către un lanț hotelier. Aplicația client va avea 3 tipuri de utilizatori: client, angajat și administrator.

Utilizatorii de tip client pot efectua următoarele operații fără autentificare:

❖Vizualizarea listei camerelor dintr-un hotel selectat sortată după locație și număr;

❖Filtrarea listei camerelor după locație, disponibilitate, preț, poziție, facilități.

Utilizatorii de tip angajat pot efectua următoarele operații după autentificare:

❖Toate operațiile permise utilizatorilor de tip client;

❖Operații CRUD în ceea ce privește persistența camerelor din lanțul hotelier;

❖Operații CRUD pentru informațiile legate de utilizatorii de tip client.

Utilizatorii de tip administrator pot efectua următoarele operații după autentificare:

❖Operații CRUD pentru informațiile legate de utilizatorii care necesită autentificare;

❖Vizualizarea listei utilizatorilor care necesită autentificare.

**Interfața grafică a aplicației va fi disponibilă în cel puțin 3 limbi de circulație internațională** (implicit limba română).

# **Partea I –Analiză**

## 

## Introducere

În cadrul acestei teme, sunt prezentate cerințele și specificațiile pentru dezvoltarea unei aplicații destinate unui lanț hotelier.

Lanțul hotelier reprezintă o rețea complexă de unități, fiecare cu propriile sale necesități de gestionare a inventarului, interacțiune cu clienții si de a gestiona accesul angajaților . Această diversitate necesită o abordare adaptabilă și scalabilă în ceea ce privește software-ul utilizat.

Scopul aplicației propuse este de a oferi o platformă centralizată și ușor de utilizat, care să permită angajaților lantului, administratorilor lanțului și clientilor să interacționeze eficient cu sistemul de gestionare a utilizatorilor, angajatilor, respectiv a camerelor dintr-un hotel.

Prin intermediul acestei aplicații, se urmărește creșterea eficienței operaționale, îmbunătățirea experienței utilizatorilor și optimizarea proceselor de gestionare a camerelor. Prin intermediul funcționalităților adaptate fiecărui tip de utilizator, se dorește satisfacerea nevoilor specifice ale fiecărei categorii de utilizatori implicați în sistem.

### 1.1 Obiective generale

Principalele obiective ale acestei aplicații sunt:

* Să ofere utilizatorilor funcționalități specifice în funcție de rolul lor în sistem.
* Să permită vizualizarea și filtrarea eficientă a camerelor disponibile într-un hotel.
* Să faciliteze operațiile CRUD (Create, Read, Update, Delete) pentru gestionarea camerelor și a informațiilor despre utilizatori.

|  |  |
| --- | --- |
| Rol | Obiective |
| Client | Vizualizarea listei tuturor camerelor dintr-un hotel;  Filtrarea listei camerelor dupa locatie, disponibilitate, pret si numar; |
| Angajat | Vizualizarea listei tuturor camerelor dintr-un hotel;  Filtrarea listei camerelor dupa locatie, disponibilitate, pret si numar;  Operații CRUD în ceea ce privește persistența camerelor din lantul hotelier.  Operații CRUD pentru informatiile legate de utilizatorii de tip client. |
| Administrator | Operații CRUD pentru informațiile legate de utilizatorii de tip angajat;  Vizualizarea listei tuturor utilizatorilor de tip angajat. |

**Tabelul 1**. Actorii cazurilor de utilizare

### 1.2 Modelul Use-Case

Diagrama Use-Case realizată în StarUML prezintă interacțiunile dintre diferitele tipuri de utilizatori și funcționalitățile sistemului. Această diagramă Use-Case este concepută pentru a ilustra interacțiunile dintre diferiți tipi de utilizatori și sistemul dezvoltat pentru lanțul hotelier.

Scopul său principal este de a evidenția funcționalitățile disponibile pentru fiecare tip de utilizator și modul în care acestea contribuie la obiectivele sistemului.

Prin intermediul acestei diagrame, se dorește furnizarea unei reprezentări vizuale clare a interacțiunilor dintre actori și funcțiile sistemului, facilitând înțelegerea și comunicarea între membrii echipei de dezvoltare și clienți. Diagrama servește ca instrument de referință pentru definirea și prioritizarea cerințelor funcționale, contribuind la elaborarea unei soluții software eficiente și scalabile. Figura 1.1 prezintă diagrama Use-Case asociată acestui proiect. Figura 1.2 prezinta diagrama ER.

A screenshot of a computer

Description automatically generated

Figura 1.1 Diagrama Use-Case realizata in Star-UML

|  |
| --- |
| Figura 1.2 Diagrama ER |

# **Partea II –Proiectare**

# 2. Proiectare

### 2.1 Arhitectura sistemului

Arhitectura aleasă pentru proiectarea sistemului este MVC (Model View Controller). MVC este un model arhitectural utilizat pentru a separa logica de prezentare a unei aplicații (UI) de logica de afaceri și accesul la date. Implementarea MVC în cadrul aplicației acestui proiect poate furniza o structură bine definită și modulară a codului, ceea ce poate îmbunătăți întreținerea și scalabilitatea sistemului.

* **Modelul:** Această componentă conține logica de afaceri și se ocupă de accesul la date. Responsabilitatea sa principală este de a efectua operațiile CRUD (Create, Read, Update, Delete) asociate cu camerele hotelului și utilizatorii, comunicând cu baza de date sau alte surse de date. Modelul este izolat de interfața utilizatorului, ceea ce facilitează reutilizarea și testarea codului într-un mod eficient.
* **View**: Este componenta care interacționează direct cu utilizatorul (UI), având rolul de a afișa informații și facilita interacțiunea cu acesta. În cadrul aplicației destinate lanțului hotelier, view-urile pot fi pagini web, interfețe mobile sau alte elemente grafice utilizate pentru prezentarea informațiilor utilizatorilor. Acestea sunt lipsite de logica de afaceri, concentrându-se exclusiv pe afișarea datelor și răspunsul la acțiunile întreprinse de utilizator.
* **Controller**: Este elementul intermediar între model și vizualizare, având rolul de a gestiona interacțiunile dintre acestea două. Controlorul conține logica de prezentare și se ocupă de interpretarea acțiunilor utilizatorului din vizualizare, realizând apoi acțiuni corespunzătoare în model. De asemenea, este responsabil pentru actualizarea vizualizării cu datele relevante din model. Utilizarea controlorului permite o separare clară între logica de afișare și cea de afaceri, ceea ce facilitează testarea și întreținerea componentelor

In figura 2.1 este prezentata diagrama UML a arhitecturii MVC realizata pe structura temei.

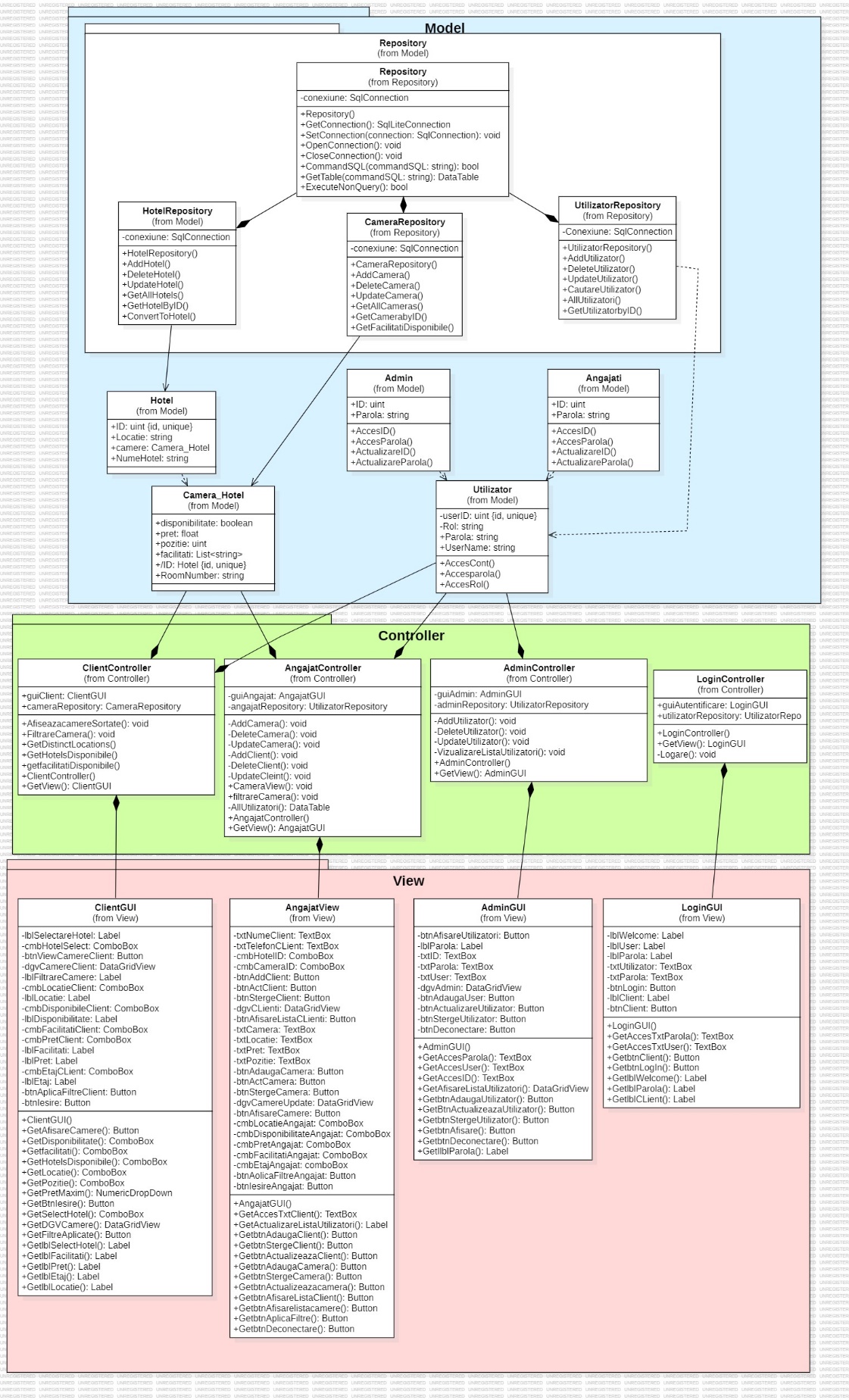


Figura 2.1 Diagrama MVC

A screenshot of a computer

Description automatically generated

Figura 2.2 Model

A screenshot of a computer program

Description automatically generated

Figura 2.3 Controller

A screenshot of a computer program

Description automatically generated

Figura 2.4 View

# **Partea III – Implementare**

# 3. Implementare

Implementarea arhitecturii s-a desfășurat în Visual Studio, utilizând limbajul de programare C#. A fost adăugată o componentă Model, cuprinzând clasele corespunzătoare prezentate în figura 2.1. Clasa Model deține responsabilitatea logicii de afaceri și interacțiunii cu datele. Aceasta poate implica accesul la baza de date sau la alte surse de date, precum și logica pentru operațiunile CRUD (Create, Read, Update, Delete) asociate acestor date. În clasa View au fost create GUI pentru fiecare utilizator și modul lor de afișare, în timp ce în clasa Controller sunt primite acțiunile din View și sunt implementate folosind logica clasei Model, funcționând astfel ca o clasă de legătură.

### 3.1 Interfata

|  |
| --- |
|  |

Figura 3.1 Interfata de autentificare

|  |
| --- |
|  |

Figura 3.2 Interfata client

|  |
| --- |
|  |

Figura 3.3 Interfata angajat

|  |
| --- |
|  |

Figura 3.3 Interfata administratorului

## 4. Concluzii

Scopul acestei teme a fost de a aprofunda cunoștințele și de a elabora o interfață utilizând arhitectura Model-View-Controller (MVC) pentru un lanț hotelier. Prin aplicarea acestei arhitecturi, s-a obținut o aplicatie care să permita o separare clară a responsabilităților între logica de afaceri, interfața cu utilizatorul și accesul la date.