PROIECT - INGINERIA SISTEMELOR SOFTWARE

AN UNIVERSITAR 2020-2021, SEMESTRUL 2

Aplicatie

Rezervari locuri teatru

**Nume si Prenume Student: Coordonator:**

Cret Andreea Prof. Ionut-Paul Tirban

Grupa: 222/2

Informatica-romana

1. Prezentarea cerintei:

Teatrul National din Cluj-Napoca doreste sa se modernizeze pentru a face gestiunea mai accesibila pentru urmatoarele generatii.

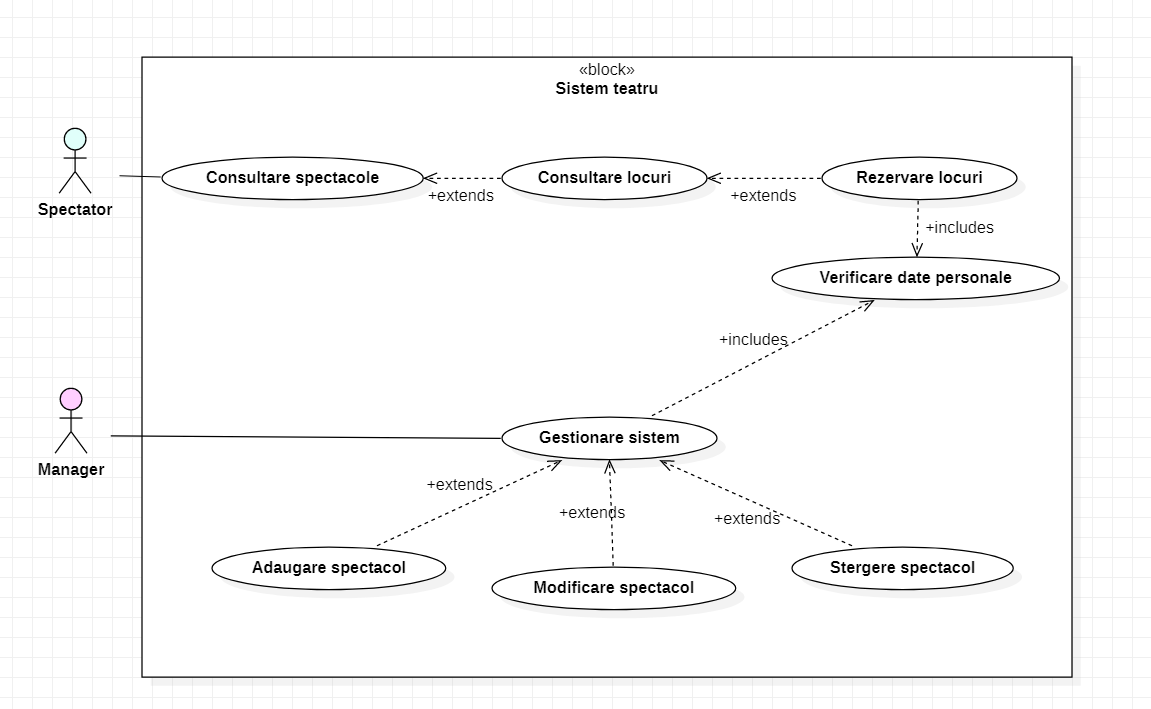
Asadar, aplicatia dezvoltata este o incercare de a simplifica procesul de rezervare a biletelor la spectacole, eliminand pasii care nu sunt necesari, astfel incat aplicația ofera posibilitatea de a rezerva bilete la spectacole printr-o apasare de buton. De asemenea, produsul se dorește a fi un ajutor pentru manageri, care pot urmari cu ușurință rezervarile.

6. REZERVARI LOCURI

O institutie teatrala pune la dispozitia spectatorilor un sistem pentru rezervarea locurilor la spectacole. În fiecare zi, institutia are mai multe reprezentatii, la care spectatorii pot rezerva locuri începand de dimineata. Pentru un loc în sala, sunt memorate urmatoarele informatii: pozitie (rândul x, loja y etc.), numar si pret. Terminalele puse la dispozitia spectatorilor afiseaza întreaga configuratie a salii, precizând pentru fiecare loc pozitia, numarul, pretul si starea (liber sau rezervat). Folosind un astfel de terminal, spectatorul îsi poate introduce datele personale, poate selecta unul sau mai multe locuri si poate declansa un buton pentru rezervarea lor. După fiecare rezervare, toate terminalele vor afisa situația actualizata referitor la ocuparea sălii.

2. Modelul functional:

Diagrama cazuri de utilizare:



Descriere cazuri de utilizare:

|  |  |  |  |
| --- | --- | --- | --- |
| **ID and name** | Rezervare locuri teatru | | |
| **Primary actor** | Spectator | **Secondary actors** |  |
| **Description** | Clientul acceseaza aplicatia teatrului, consulta spectacolele si locurile disponibile in ziua curenta, si rezerva unul sau mai multe bilete. | | |
| **Trigger** | Clientul cauta locuri la un spectacol | | |
| **Preconditions** | PRE-1. Exista cel putin un spectacol disponibil. | | |
| **Postconditions** | POST-1. Sistemul valideaza si retine rezervarea.  POST-2. Sistemul actualizeaza situatia locurilor. | | |
| **Normal flow** | **1.0 Rezervare loc la un spectacol**  1. Clientul consulta spectacolele disponibile, selectand reprezentatia dorita.  2. Sistemul afiseaza configuratia salii.  3. Clientul consulta locurile disponibile, alegand locul dorit (vezi 1.1).  4. Sistemul solicita datele personale.  5. Clientul completeaza datele personale si incheie rezervarea (vezi 1.0.E1, 1.0.E2).  6. Sistemul valideaza rezervarea si actualizeaza situatia locurilor. | | |
| **Alternative flows** | * 1. **Rezervare locuri multiple**   1. Clientul alege mai multe locuri.  2. Revenire la pasul 4 din scenariul normal. | | |
| **Exceptions** | **1.0.E1: locurile selectate au fost rezervate intre timp**  1. Sistemul afiseaza un mesaj de eroare si permite selectarea altor locuri.  2a. Clientul consulta locurile disponibile actualizate si continua de la pasul 3.  2b. Clientul iese din aplicatie.  **1.0.E2: datele introduse sunt incorecte**  1. Sistemul afiseaza un mesaj de eroare si solicita reintroducerea datelor.  2a. Clientul reintroduce datele corect si sare la pasul 6.  2b. Clientul iese din aplicatie. | | |

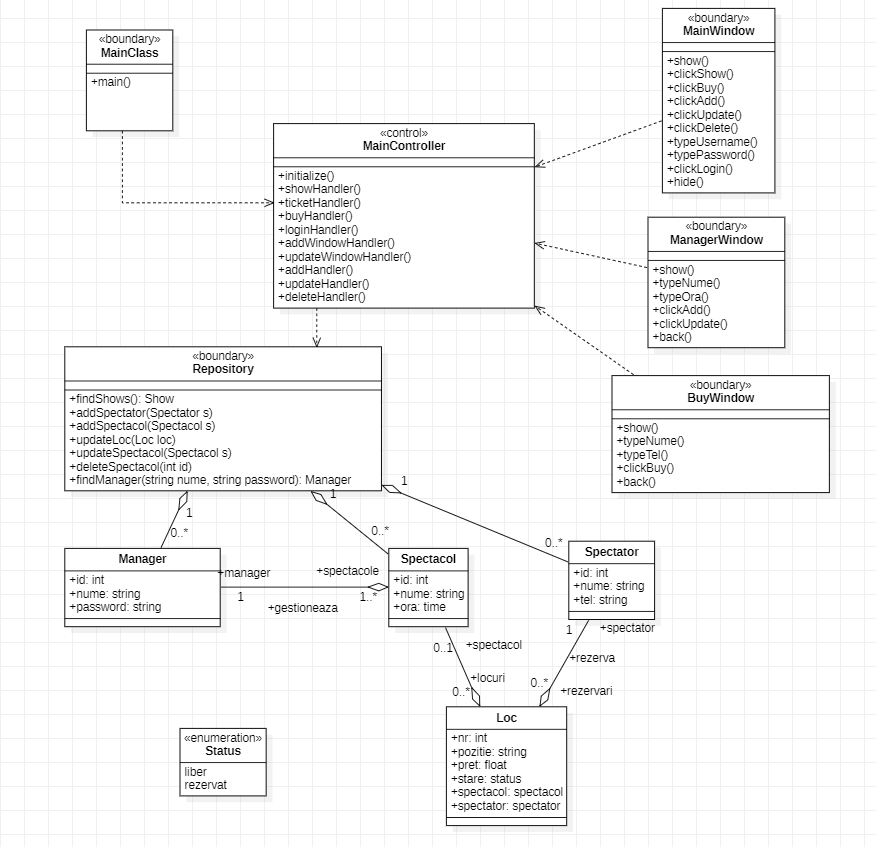
|  |  |  |  |
| --- | --- | --- | --- |
| **ID and name** | Adaugare spectacol teatru | | |
| **Primary actor** | Manager | **Secondary actors** |  |
| **Description** | Managerul se logheaza in aplicatia teatrului si adauga un nou spectacol in ziua curenta | | |
| **Trigger** | Managerul intra in aplicatie | | |
| **Preconditions** | PRE-1. Managerul are un cont in aplicatie | | |
| **Postconditions** | POST-1. Sistemul actualizeaza situatia spectacolelor. | | |
| **Normal flow** | **1.0 Adaugare spectacol**  1. Managerul se logheaza in aplicatie.  2. Sistemul afiseaza configuratia curenta a spectacolelor si locurilor.  3. Managerul solicita adaugarea unui spectacol nou  4. Sistemul solicita informatii despre spectacol.  5. Managerul completeaza informatiile si adauga un nou spectacol (vezi 1.0.E1).  6. Sistemul valideaza adaugarea si actualizeaza situatia spectacolelor. | | |
| **Alternative flows** |  | | |
| **Exceptions** | **1.0.E1: datele introduse sunt incorecte**  1. Sistemul afiseaza un mesaj de eroare si solicita reintroducerea datelor.  2a. Managerul reintroduce datele corect si sare la pasul 6.  2b. Managerul iese din aplicatie. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID and name** | Modificare spectacol teatru | | |
| **Primary actor** | Manager | **Secondary actors** |  |
| **Description** | Managerul se logheaza in aplicatia teatrului si modifica un spectacol din ziua curenta | | |
| **Trigger** | Managerul intra in aplicatie | | |
| **Preconditions** | PRE-1. Managerul are un cont in aplicatie | | |
| **Postconditions** | POST-1. Sistemul actualizeaza situatia spectacolelor. | | |
| **Normal flow** | **1.0 Modificare spectacol**  1. Managerul se logheaza in aplicatie.  2. Sistemul afiseaza configuratia curenta a spectacolelor si locurilor.  3. Managerul solicita modificarea unui spectacol existent  4. Sistemul solicita informatiile noi despre spectacol.  5. Managerul completeaza informatiile si modifica spectacolul (vezi 1.0.E1).  6. Sistemul valideaza adaugarea si actualizeaza situatia spectacolelor. | | |
| **Alternative flows** |  | | |
| **Exceptions** | **1.0.E1: datele introduse sunt incorecte**  1. Sistemul afiseaza un mesaj de eroare si solicita reintroducerea datelor.  2a. Managerul reintroduce datele corect si sare la pasul 6.  2b. Managerul iese din aplicatie. | | |

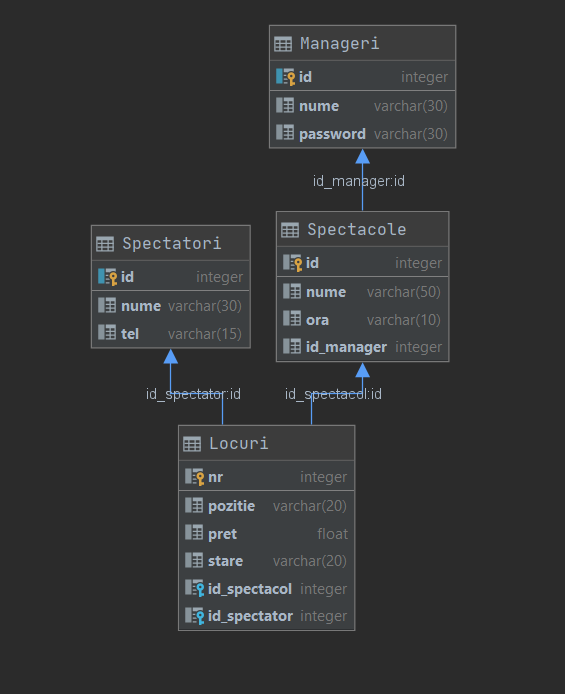
|  |  |  |  |
| --- | --- | --- | --- |
| **ID and name** | Stergere spectacol teatru | | |
| **Primary actor** | Manager | **Secondary actors** |  |
| **Description** | Managerul se logheaza in aplicatia teatrului si sterge un spectacol din ziua curenta | | |
| **Trigger** | Managerul intra in aplicatie | | |
| **Preconditions** | PRE-1. Managerul are un cont in aplicatie | | |
| **Postconditions** | POST-1. Sistemul actualizeaza situatia spectacolelor. | | |
| **Normal flow** | **1.0 Stergere spectacol**  1. Managerul se logheaza in aplicatie.  2. Sistemul afiseaza configuratia curenta a spectacolelor si locurilor.  3. Managerul solicita stergerea unui spectacol existent  4. Sistemul valideaza stergerea si actualizeaza situatia spectacolelor. | | |
| **Alternative flows** |  | | |
| **Exceptions** |  | | |

3. Modelul conceptual:

Diagrama de clase:



4. Diagrama bazei de date:



5. Model dinamic:

Diagrama de secventa a cazului de achizitionare bilete (Iteratia 2):

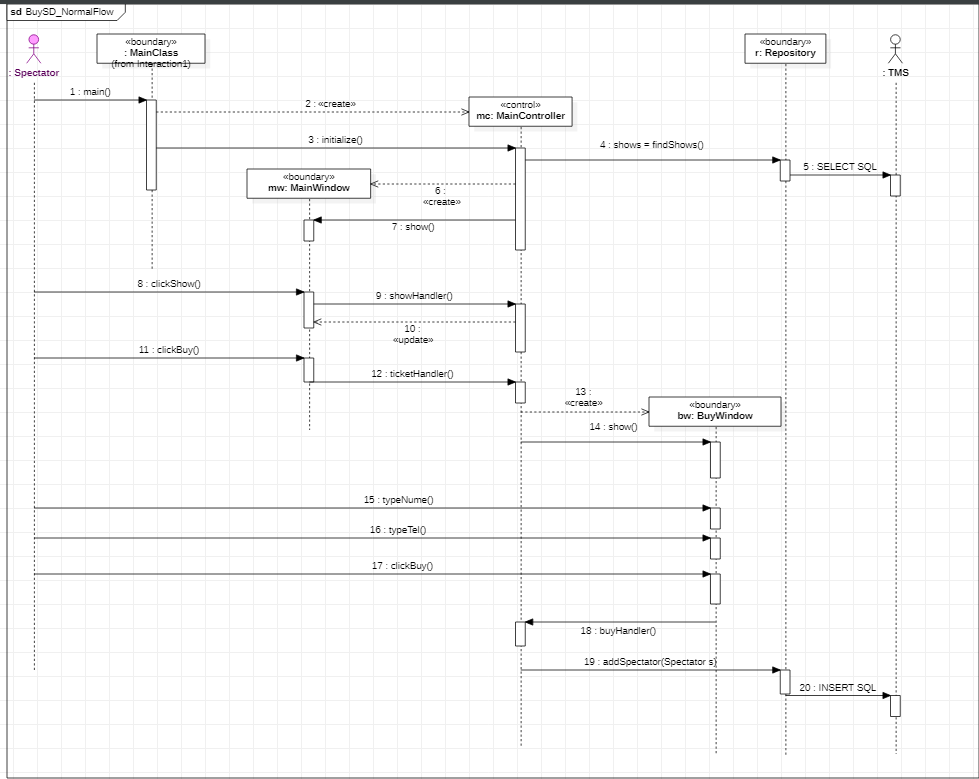


Diagrama de comunicare a cazului de achizitionare bilete (Iteratia 2):

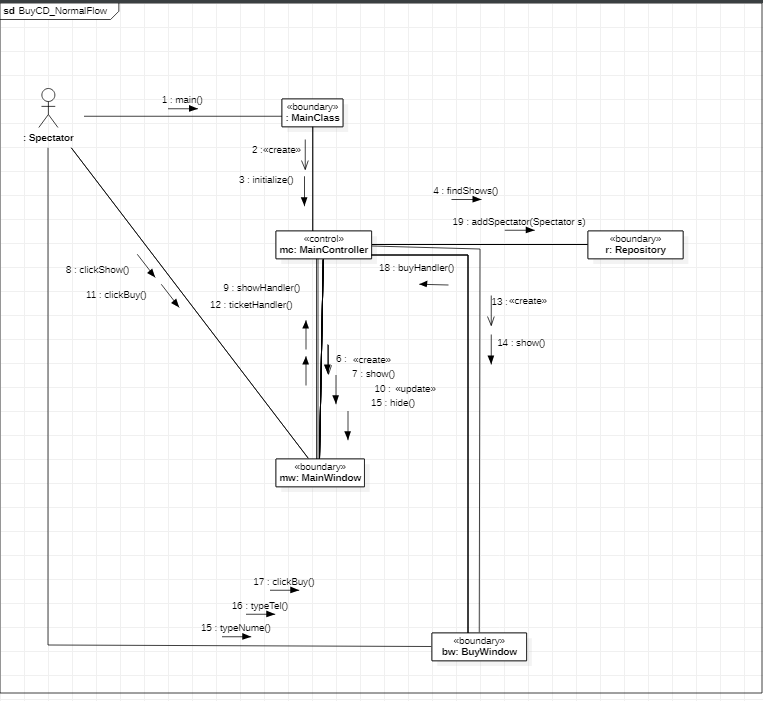


Diagrama de secventa a cazului de adaugare spectacol (Iteratia 3):

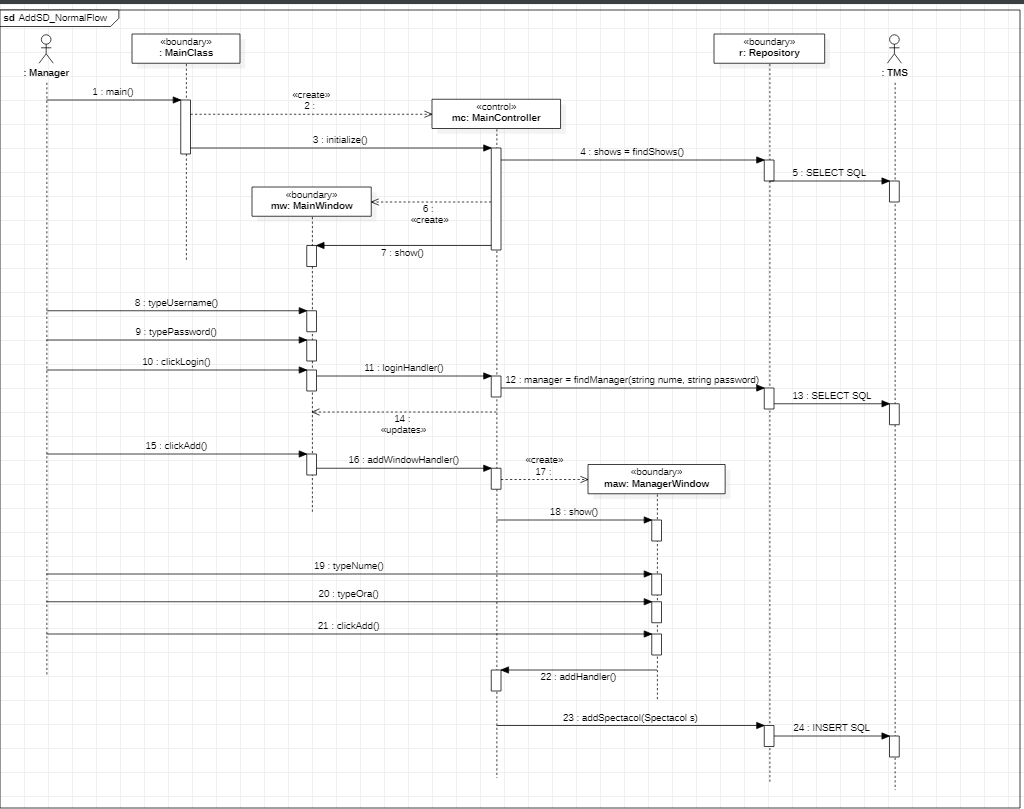


Diagrama de comunicare a cazului de adaugare spectacol (Iteratia 3):

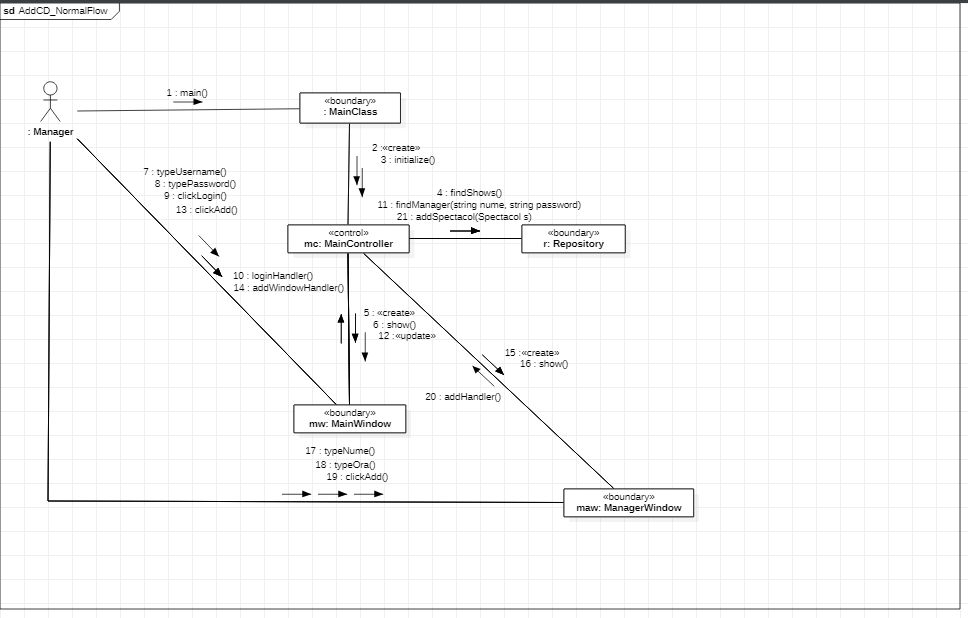


Diagrama de secventa a cazului de modificare a unui spectacol (Iteratia 3):

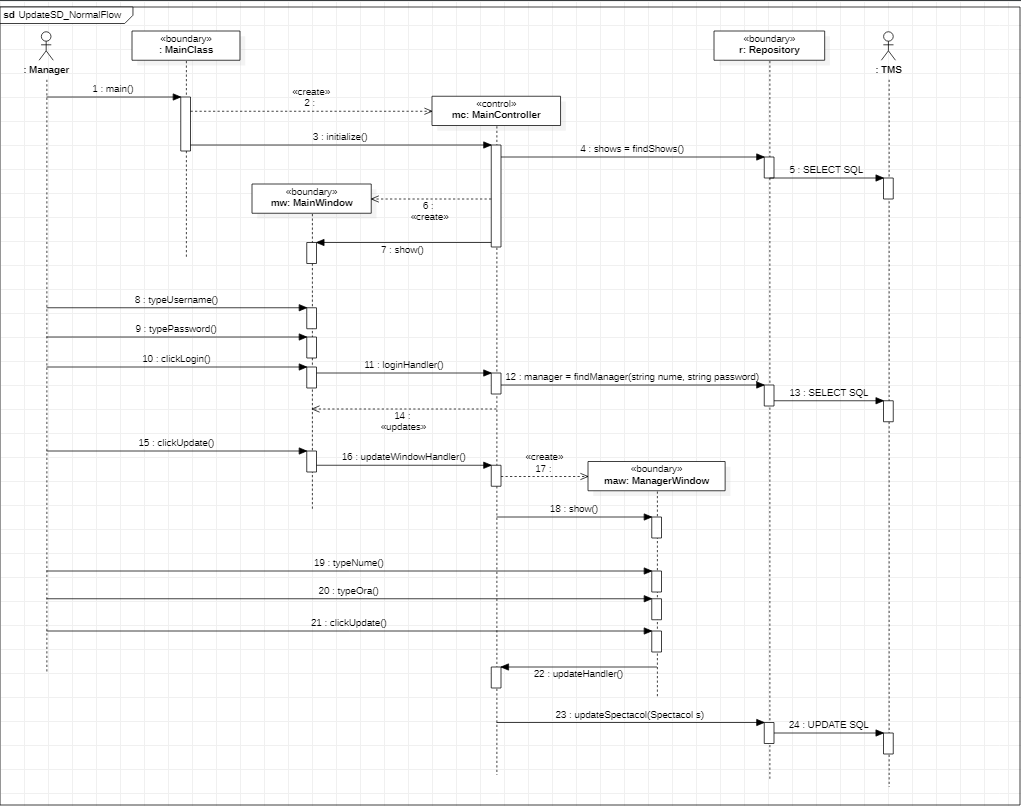


Diagrama de comunicare a cazului de modificare a unui spectacol (Iteratia 3):

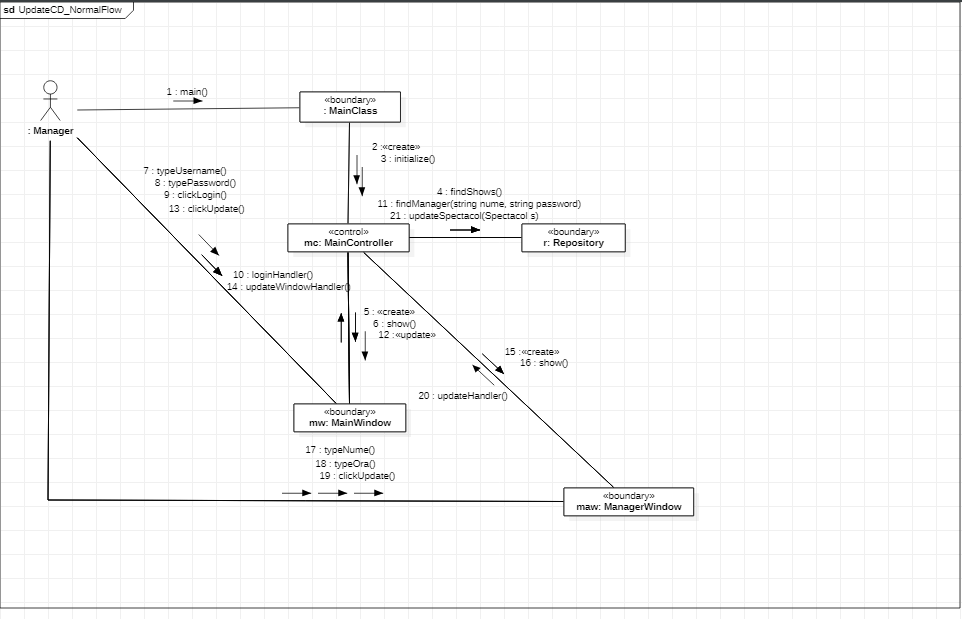


Diagrama de secventa a cazului de stergere a unnui spectacol (Iteratia 3):

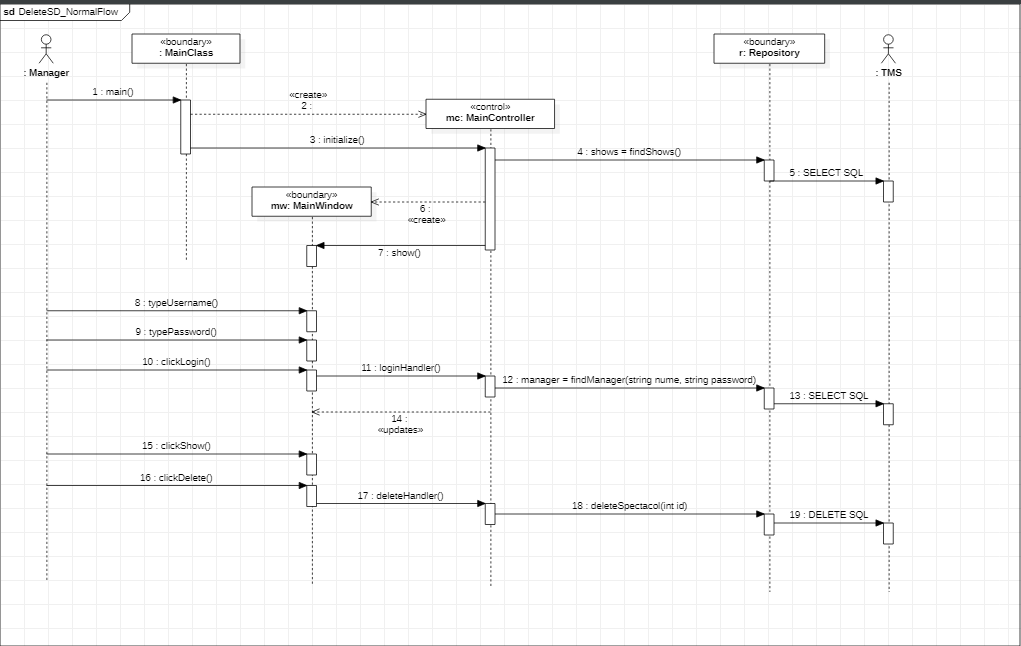
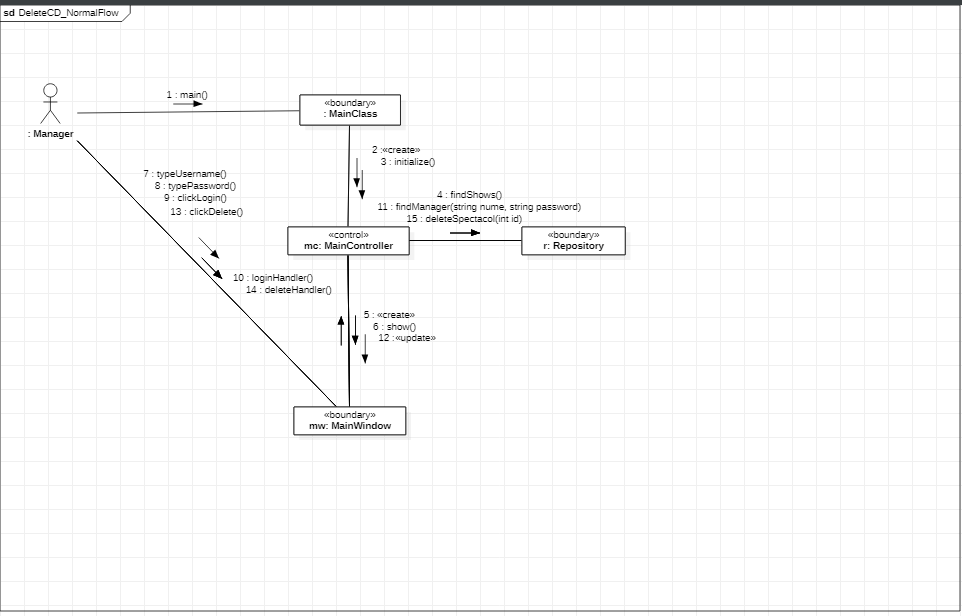


Diagrama de comunicare a cazului de stergere a unui spectacol (Iteratia 3):



6. Documentația tehnică:

Funcționalitățile:

**Client:**

* Intra in aplicatie, aceasta afiseaza spectacolele disponibile, cu ora desfasurarii fiecaruia si pentru fiecare spectacol, locurile disponibile cu pozitia si pretul fiecaruia.
* Alege spectacolul si locurile preferate si apasa butonul rezerva
* Introduce datele personale (numele si numarul de telefon)
* Finalizeaza rezervarea

**Manager:**

* Intra in aplicatie si se logheaza, moment in care apar butoanele pentru functionalitatile accesibile managerilor.
* Poate adauga un nou spectacol, modifica un spectacol existent sau sa stearga un spectacol din lista de spectacole.

Etapele proiectării:

**Analiza cerintelor:**

• Am indentificat cerintele impuse, functionalitatiile (autentificarea managerului, un operator poate să rezerve locuri, un manager poate să adauge/modifice/stearga spectacole) si constrângerile (managerul trebuie să fie autentificat, un loc poate fi rezervat de un singur client la un moment dat).

**Analiza:**

* Cerințe funcționale: Aplicația trebuie să îi permita clientului să rezerve bilete si managerului să adauge/modifice/stearga spectacole.
* Cerințe non-funcționale: Aplicația trebuie să fie scrisă în Java, să fie ușor de utilizat și să folosească un ORM.

**Proiectarea sistemului:**

* Am folosit StarUML pentru a realiza diagrame pentru a planifica dezvoltarea aplicației. Tipurile de diagrame pe care le-am folosit pentru a-mi facilita munca sunt: diagramă de utilizare, diagramă de secvență, diagramă de comunicare, diagram de clase și diagrama bazei de date.

**Implementarea:**

* Am utilizat o abordare top-down, începând cu interfața grafică. Am continuat cu controlerul și baza de date. Am repetat această abordare pentru toate funcționalitățile.

**Testare:**

* Am testat codul în 2 modalități: am rugat o persoană să folosească aplicația și am testat toate cazurile aparute după implementarea codului (white-box testing)

Tehnologii folosite:

**Proiectare:**

* Pentru proiectare și dezvoltarea diagramelor am folosit StarUML, pentru posibilitatea de a grupa diagramele în același proiect și deoarece a fost prezentat atât la această materie cât și la altele, ceea ce mi-a permis să ma familiarizez cu acesta.

**Mediu de dezvoltare:**

* Mediul de dezvoltare pe care l-am folosit este IntelliJ Idea, deoarece este un mediu stabil, în care se pot integra cu ușurință framework-uri. De asemenea acesta oferă posibilitatea de a gestiona baza de date pe care am folosit-o.

**Baza de date:**

* Am utilizat SQLite fiindcă este ușor de folosit și am considerat ca biblioteca nu va avea nevoie de o capacitate mai mare de un milion de înregistrări (capacitate până la care SQLite fucționeaza foarte bine)

**Limbaj de programare:**

* Limbajul de programare ales este Java. Am luat această decizie datorită experienței pe care o am cu acest limbaj.

**GUI (Interfață grafică):**

* Am folosit JavaFX și Scene Builder pentru ușurința de a crea interfețe cu un design plăcut prin drag and drop.

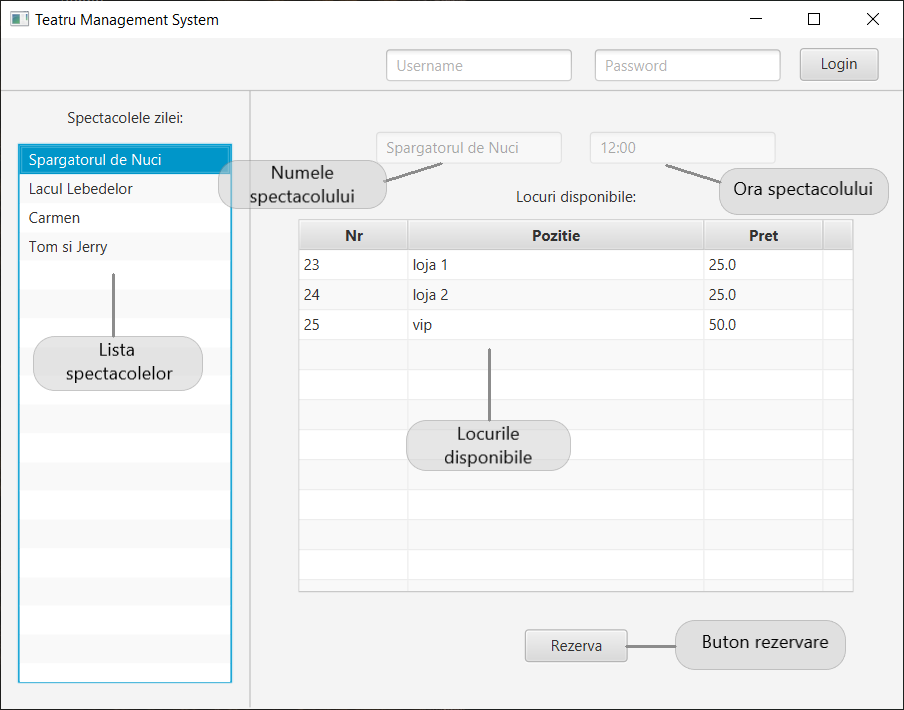
**ORM (Object Relational Mapping)**

* ORM-ul pe care l-am folosit este Hibernate, deoarece am considerat ca este framework-ul despre care pot gasi cea mai multă informație care să mă ajute la implementarea aplicației.

7. Tutorial

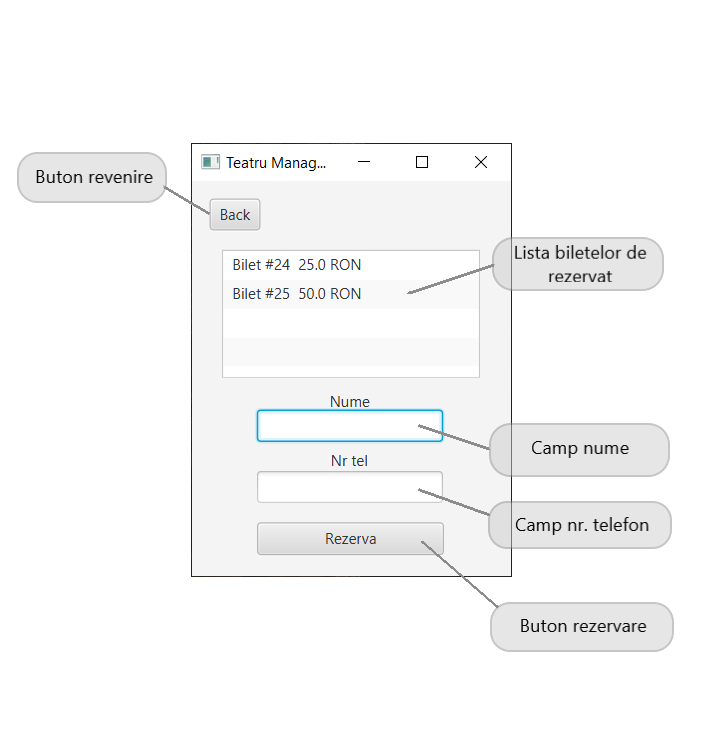
Pagina principala a aplicatiei:

Aceasta este pagina care se va deschide cand un client intra in aplicatie. Aplicatia afiseaza lista de spectacole, iar la selectarea unuia dintre acestea, apar locurile disponibile la spectacolul selectat cu pozitia si pretul acestuia. Dupa selectarea biletului/biletelor alese, se apasa butonul “Rezerva” pentru a trece la pasul urmator.



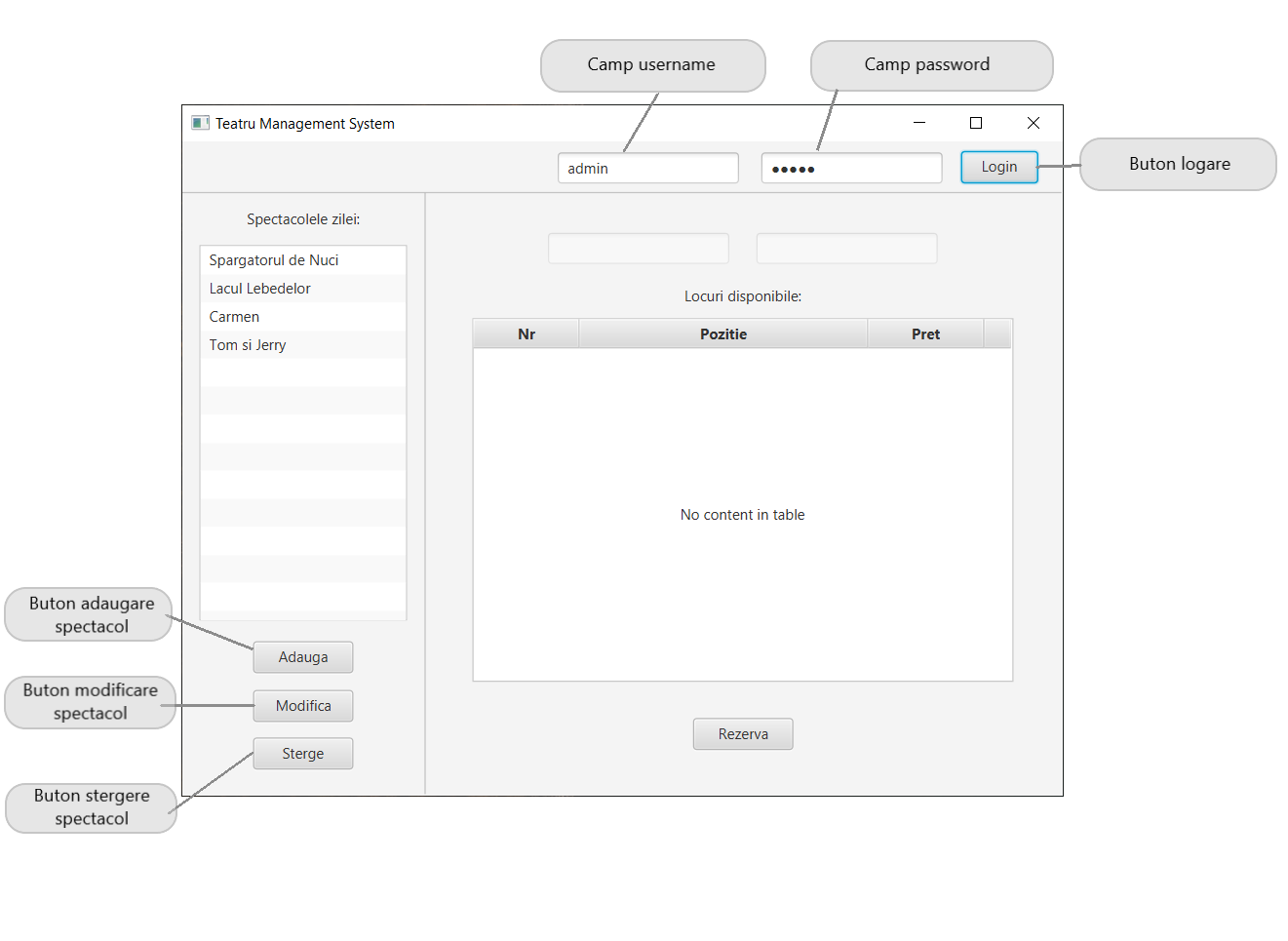
Finalizare achizitionarii biletelor:

Dupa actionarea butonului “Rezerva”, se dechide o noua fereastra, unde apar biletele selectate si campuri pentru introducerea numelui si a numarului de telefon al clientului. Dupa completarea datelor, se pot rezerva biletele selectate.



Pagina principala a aplicatiei pentru un manager logat:

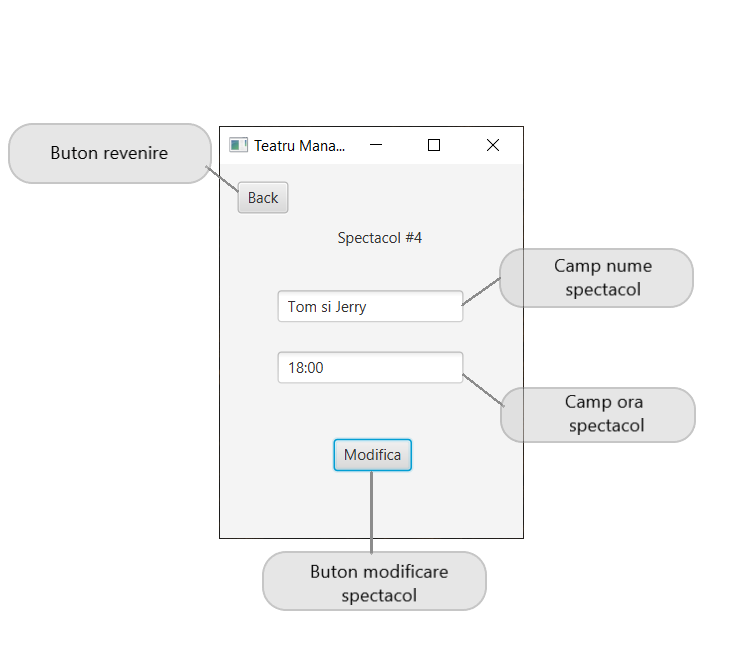
Mangerul se poate loga in campurile din partea de sus a ferestrei, iar apoi, pe langa functionalitatile disponibile operatorului, managerul poate adauga, modifica sau sterge un anumit spectacol dupa selectarea acestuia si actionarea butonului corespunzator. In cazul in care este selectat un spectacol si apasat butonul “Sterge”, spectacolul va fi eliminat.



Modificarea unui spectacol:

Dupa actionarea butonului “Modifica” se dechide o fereastra noua in care apare spectacolul care va fi modificat si campurile in care se pot face modificarile.

Similar, la actionarea butonului “Adauga” se dechide o fereastra noua in care apar campurile pentru numele spectacolului care doreste sa se adauge si ora acestuia. Dupa completarea campurilor, se actioneaza butonul, iar noul spectacol va fi adaugat.



8.Bibliografie

1) <http://www.cs.ubbcluj.ro/~vladi/Teaching/Didactic/ISS%202019-2020/info%20romana/curs/>

2) <https://simaioan.wordpress.com/>

3) <http://www.cs.ubbcluj.ro/~vladi/Teaching/Didactic/ISS%202019-2020/info%20romana/seminar/>

4) <https://stackoverflow.com/>

5) Bruegge, B., Dutoit, A.H., Object-Oriented Software Engineering using UML, Patterns, and Java (3rd edition), Prentice Hall, 2010

6) <https://creately.com/blog/diagrams/class-diagram-relationships/>

7) <http://www.cs.ubbcluj.ro/~grigo/mpp/Curs/>