Proiect Ingineria Sistemelor Soft

Nume echipa: Theta(7)
Membrii echipei:
Berende Teodora-Emilia[221] – Coordonator
Balint Alex-Răzvan[221]
Barac-Antonescu Daniel[221]
Bălaș Tudor-Dan[221]
Brie Andrei-Valentin[221]
Cora Ioan-Radu[222]
Derecichei Denis-Emanuel[232]

Introducere

Proiectul a constat în proiectarea și implementarea unui sistem de management al conferințelor, având cerințe precum adăugarea unei conferințe și afișarea tuturor conferințelor aflate în desfășurare utilizatorilor, gestionarea acestor conferințe, posibilitatea înregistrării noilor utilizatori, înscrierea lucrărilor științifice de către utilizatori la o conferință, acceptarea și respingerea acestor lucrări prin evaluarea acestora de către membrii unui comitet al conferinței respective, etc.

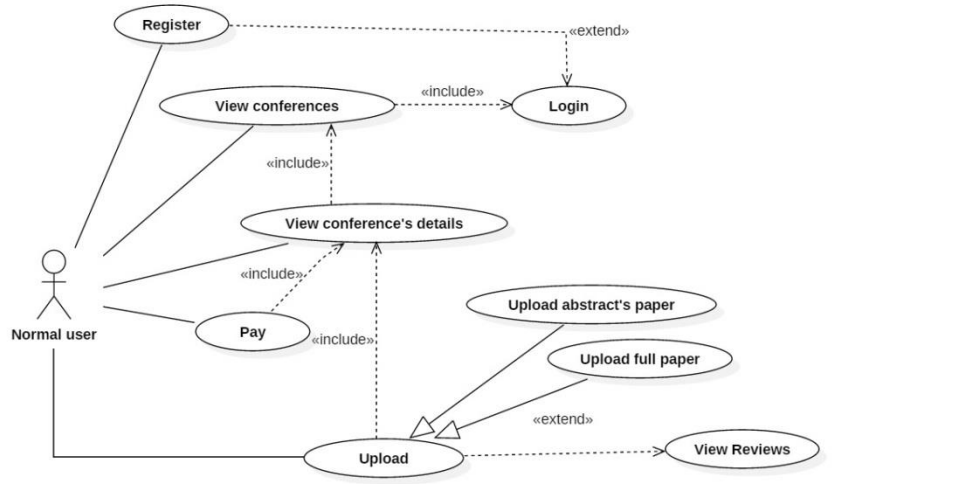
La aceste conferințe, utilizatorii pot avea funcții/roluri diferite, precum participant pasiv(”listener”), participant activ(”speaker”), membru al unui comitet de program (”PC member”) și șef al acestui comitet(”chair/co-chair”). Toate aceste roluri au atribuite un set de drepturi si restricții; un exemplu ar fi faptul că un ”chair” al unei conferințe nu poate înscrie lucrări la respectiva conferință, în schimb acesta poate schimba deadline-urile conferinței respective, cum ar fi deadline-ul ce reprezintă data până când se pot înscrie lucrări, cu condiția ca noua dată să fie mai târzie decât data care a fost modificată. Totodată, un utilizator la o conferință poate fi și ”listener” și ”speaker”, însemnând că acesta participă și își prezintă o lucrare/lucrările în cadrul acestei conferințe sau un utilizator poate fi ”PC member” al unei conferințe si ”speaker”, singura restricție fiind că acesta nu poate vedea numele evaluatorilor (colegii acestuia cu rol de ”PC member”) lucrării acestuia.

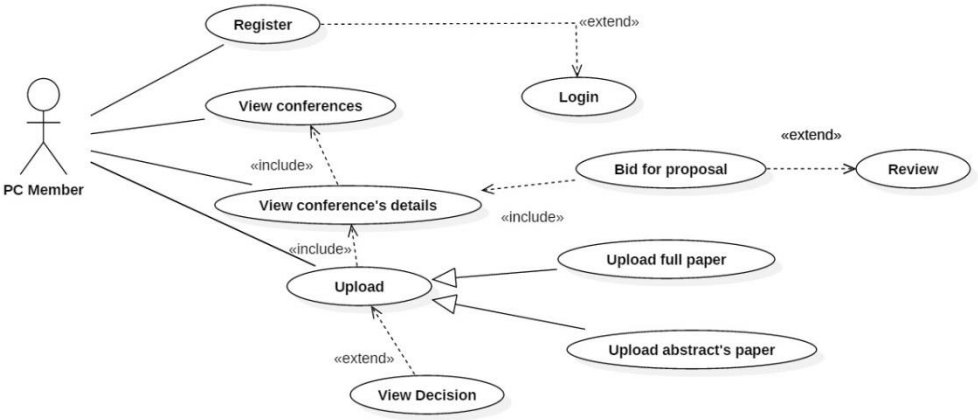
Prin diagrama cazurilor de utilizare sunt prezentate acțiunile posibile pentru fiecare funcție/rol ce pot fi efectuate de către un utilizator în cadrul aplicației.

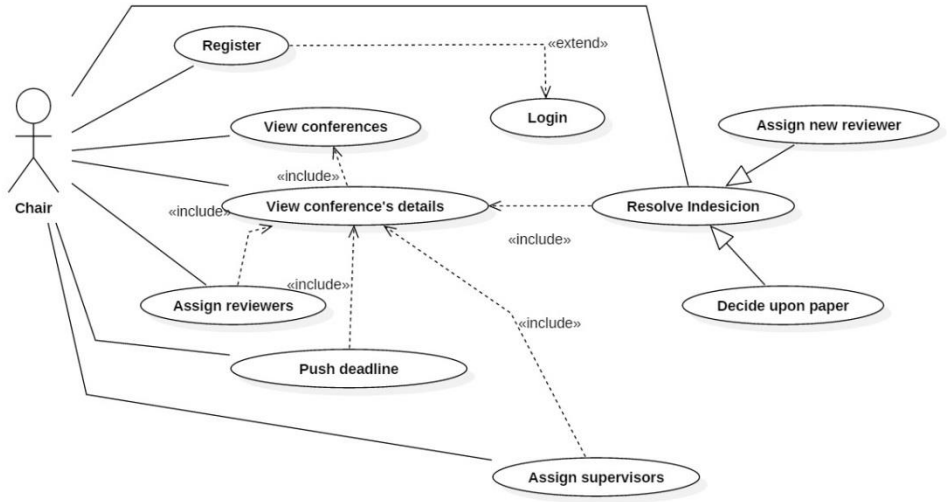
*Notă**:*În mod normal acest proiect este gândit pentru o aplicație web, dar s-a putut alege și varianta desktop, varianta pe care am ales-o si noi.

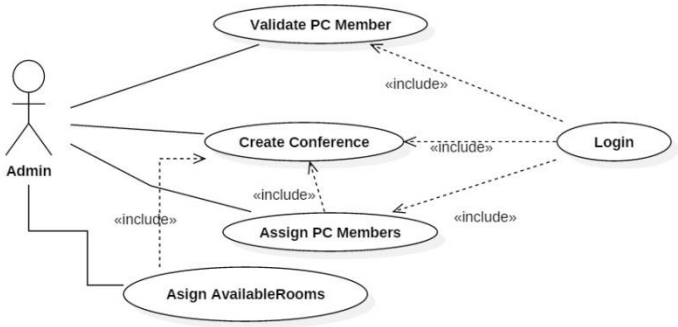
În următoarele pagini se prezintă diagramele efectuate pe parcursul proiectării aplicației.

Diagrama cazurilor de utilizare









Scenarii de utilizare ale cazurilor de utilizare

*Notă**: Aceste scenarii prezintă fluxul de evenimente în cazul de desfășurare fără erori a funționalităților*

I.

- Nume caz de utilizare: Login
- Actorii implicati: Normal user, PC member, Chair, Admin
- Fluxul de evenimente:

1. Utilizatorul porneste aplicatia
2. Aplicatia se deschide afisand form-ul initial
3. Utilizatorul isi introduce username-ul si password-ul in campurile pentru login
4. Utilizatorul apasa butonul de login
5. Aplicatia verifica daca datele introduse sunt valide
6. Aplicatia afiseaza form-ul principal

- Preconditii: Utilizatorul are acces la aplicatie, la form-ul de login
- Postconditii: Utilizatorul este logat in aplicatie

II.

- Nume caz de utilizare: Register
- Actorii implicati: Normal user, PC member, Chair
- Fluxul de evenimente:

1. Utilizatorul porneste aplicatia
2. Aplicatia se deschide afisand form-ul initial
3. Utilizatorul isi introduce datele in campurile pentru register
4. Utilizatorul apasa butonul de register
5. Aplicatia verifica daca datele introduse sunt valide
6. Aplicatia afiseaza mesajul ‘Succes’

- Preconditii: Utilizatorul are acces la aplicatie, la form-ul de login
- Postconditii: Utilizatorul are un cont creat in aplicatie

III.

- Nume caz de utilizare: View conference’s details
- Actorii implicati: Normal User, PcMember, Chair
- Fluxul de evenimente:

1. Utilizatorul porneste aplicatia
2. Aplicatia se deschide afisand form-ul initial
3. Utilizatorul isi introduce username-ul si password-ul in campurile pentru login
4. Utilizatorul apasa butonul de login
5. Aplicatia verifica daca datele introduse sunt valide
6. Aplicatia afiseaza form-ul principal
7. Utilizatorul acceseaza o conferinta
8. Aplicatia afiseaza form-ul cu detaliile conferintei

- Preconditii: Utilizatorul are acces la aplicatie, la form-ul de login si este logat(detine cont)
- Postconditii: Utilizatorul poate vizualiza detaliile conferintei dorite.

IV.

- Nume caz de utilizare: Upload
- Actorii implicati: Normal user, PC member
- Fluxul de evenimente:

1. Utilizatorul porneste aplicatia

2. Aplicatia se deschide afisand form-ul initial

3. Utilizatorul isi introduce username-ul si password-ul in campurile pentru login

4. Utilizatorul apasa butonul de login

5. Aplicatia verifica daca datele introduse sunt valide

6. Aplicatia afiseaza form-ul principal

7. Utilizatorul acceseaza o conferinta

8.Aplicatia afiseaza form-ul cu detaliile conferintei

9. Utilizatorul apasa butonul de Add Abstract

10. Aplicatia afiseaza form-ul de add abstract

11. Utilizatorul incarca rezumatul

12. Aplicatia afiseaza form-ul principal

13. Utilizatorul apasa butonul de Add Paper

14.Aplicatia afiseaza dialogul de Add paper

15. Utilizatorul incarca lucrarea

16. Aplicatia afiseaza form-ul principal

17. Utilizatorul apasa butonul de Save changes

18. Aplicatia afiseaza mesajul upload successful

- Preconditii: Utilizatorul are acces la aplicatie, la form-ul de login si este logat(detine cont), nu a trecut perioada de deadline.
- Postconditii: Utilizatorul a depus lucrarea.

V.

- Nume caz de utilizare: Pay
- Actorii implicati: Normal User
- Fluxul de evenimente:

1. Utilizatorul porneste aplicatia

2. Aplicatia se deschide afisand form-ul initial

3. Utilizatorul isi introduce username-ul si password-ul in campurile pentru login

4. Utilizatorul apasa butonul de login

5. Aplicatia verifica daca datele introduse sunt valide si utilizatorul e Normal user

6.Aplicatia afiseaza form-ul prinicpal unde sunt conferintele

7.Utilizatorul selecteaza o conferinta la care poate participa

8.Utilizatorul apasa pe butonul Pay pentru a participa la conferinta

9.Datele utilizatorului si suma sunt

- Preconditii: Utilizatorul are acces la aplicatie, la form-ul de login
  Actorul are cont.
  Utilizatorul are destui bani pentru a participa.
- Postconditii: Actorul nu are cont => Register.
  Nu mai exista locuri la o conferinta=>eroare

VI.

- Nume caz de utilizare: Create Conference
- Actorii implicati: Initiat de Admin
- Fluxul de evenimente:

1. Utilizatorul porneste aplicatia

2. Aplicatia se deschide afisand form-ul initial

3. Utilizatorul isi introduce username-ul si password-ul in campurile pentru login

4. Utilizatorul apasa butonul de login

5. Aplicatia verifica daca datele introduse sunt valide si utilizatorul e admin

6. Aplicatia afiseaza form-ul prinicpal

7.Admin-ul apasa butonul Create Conference

8.In fereastra afisata introduce datele specifice unei conferinte, precum numele si editia acesteia, orasul in care se desfasoara, data pana cand se pot depune lucrari, etc.

9.Dupa crearea conferintei, trebuie sa aloce conferintei o lista de PC Members

10. Noua conferinta este adaugata in baza de date, spre a fi afisata utilizatorilor

- Pre-conditii: Utilizatorul are acces la aplicatie, la form-ul de login
  Are cont de admin
  Exista PC Members
- Post-Conditii: Noua conferinta este creata.

VII.

- Nume caz de utilizare: Bid for proposal
- Actorii implicati: Initiat de PC Member
- Fluxul de evenimente:

1. Utilizatorul acceseaza fereastra "My conferences"

2. Aplicatia afiseaza lista conferintelor la care utilizatorul este participant

3. Utilizatorul selecteaza conferinta dorita

4. Aplicatia deschide o noua fereastra, cu detaliile conferintei alese si functionalitatile aferente rolului utilizatorului in aceasta, respectiv fazei in care se afla conferinta

5. Utilizatorul selecteaza o lucrare inscrisa

6. Aplicatia va afisa abstractul lucrarii

7. Utilizatorul va alege una din optiunile de bidding: interested/neutral/not interested

8. Aplicatia va salva modificarile

- Preconditii: Utilizatorul are acces la aplicatie si este logat
  Este PC Member la conferinta dorita
  Conferinta se afla in faza de bidding (licitatia lucrarilor)
- Postconditii: Lucrarea va fi insemnata cu interested/not interested din partea utilizatorului sau va ramane cu insemnul default "neutral".

VIII.

- Nume caz de utilizare: Review
- Actorii implicati: Initiat de PC Member
- Fluxul de evenimente:

1. Utilizatorul acceseaza fereastra "My conferences"

2. Aplicatia afiseaza lista conferintelor la care utilizatorul este participant

3. Utilizatorul selecteaza conferinta dorita

4. Aplicatia deschide o noua fereastra, cu detaliile conferintei alese si functionalitatile aferente rolului utilizatorului in aceasta, respectiv fazei in care se afla conferinta

5. Utilizatorul selecteaza o lucrare pentru care a fost asignat

6. Aplicatia afiseaza textul integral al lucrarii

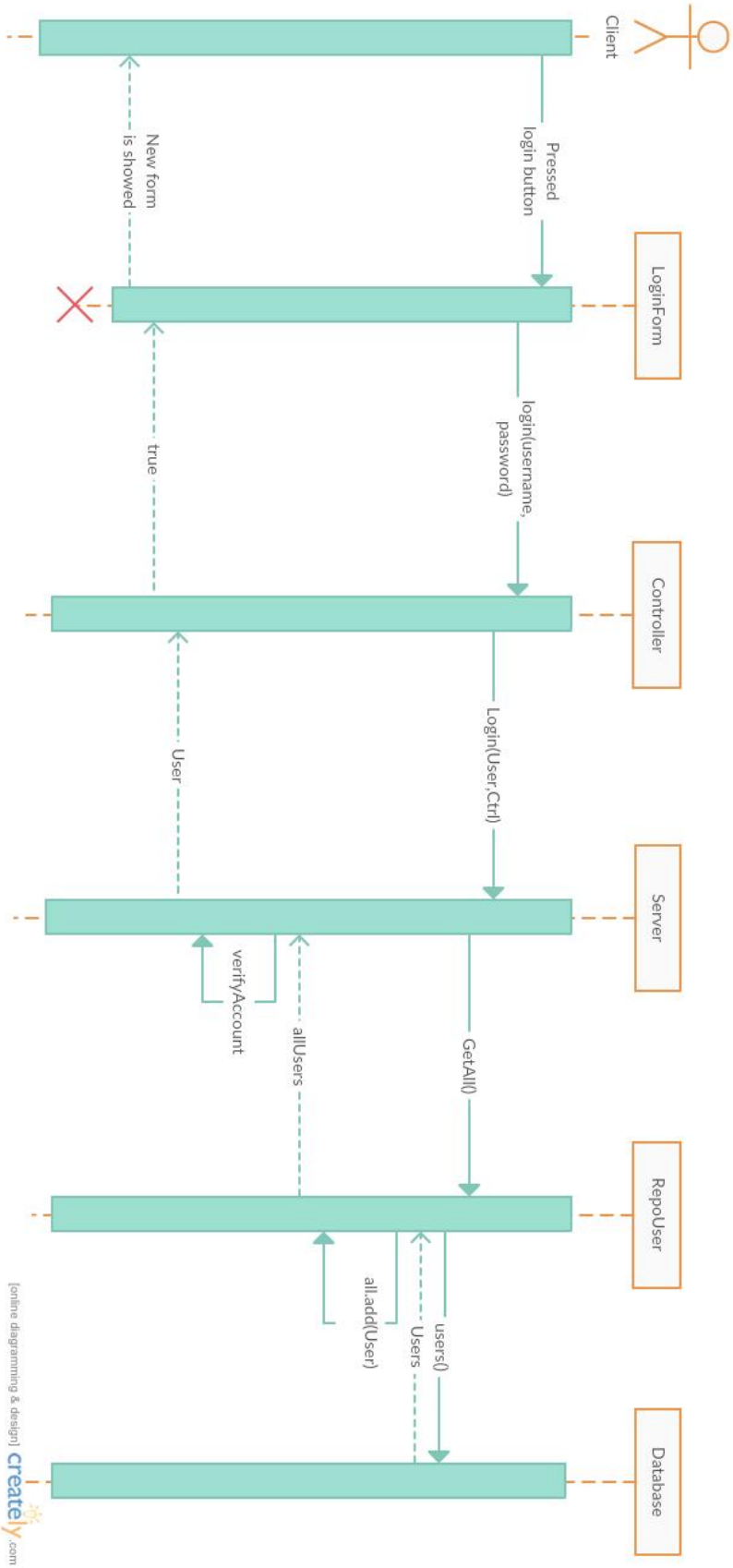
7. Utilizatorul acorda un calificativ lucrarii si scrie un mesaj critic

8. Aplicatia va salva modificarile si va notifica utilizatorul care s-a inscris cu lucrarea respectiva

- Preconditii: Utilizatorul are acces la aplicatie si este logat
  Este PC Member la conferinta dorita
  A fost asignat pentru lucrarea respective
  Conferinta se afla in faza de review(avaluarea lucrarilor)
- Postconditii: Lucrarea a primit evaluarea, iar autorul acesteia este notificat

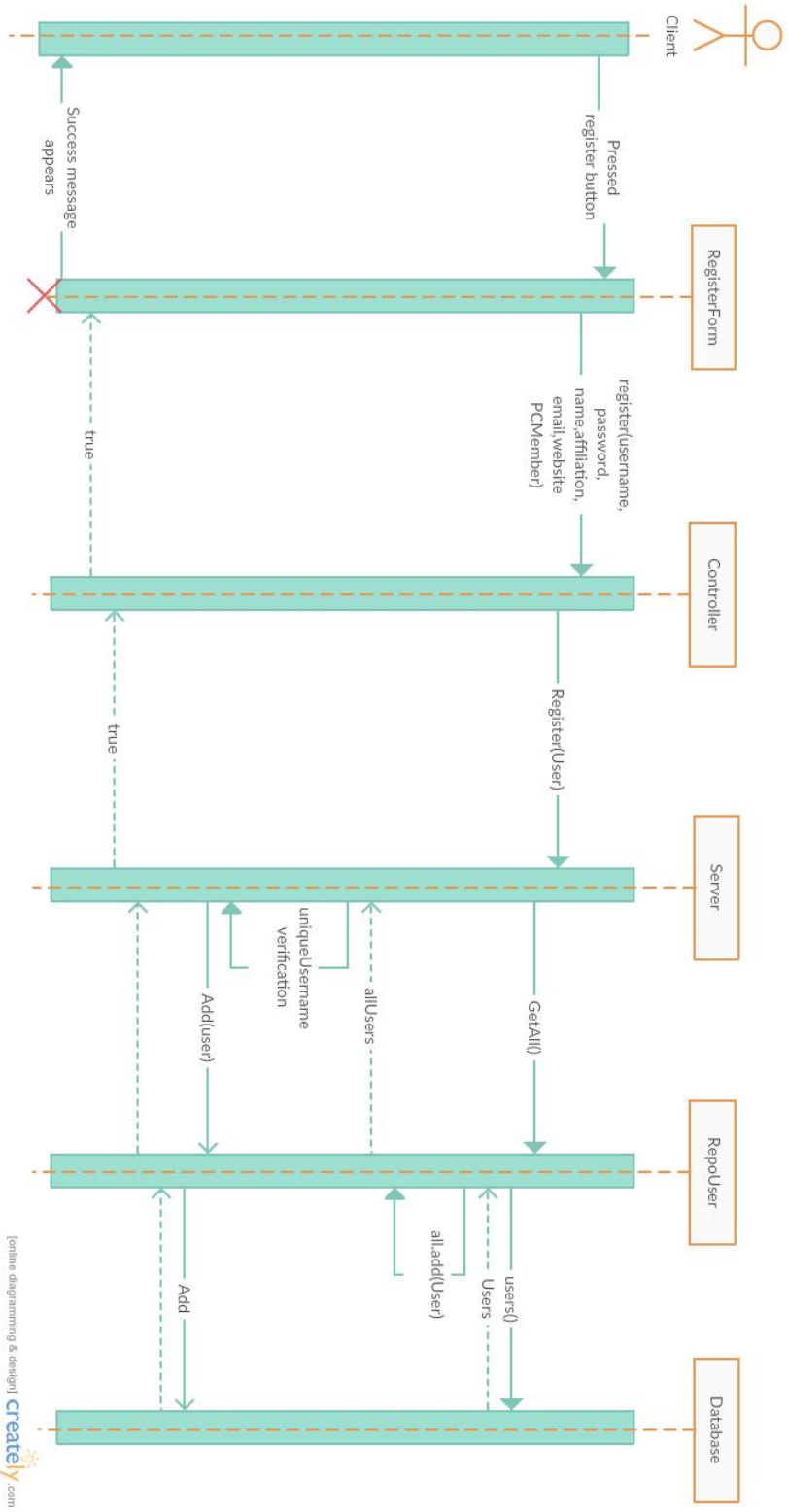
Diagrame de secvență

*Notă**: Aceste diagrame prezintă interacțiunile între entități în cazul de desfășurare fără erori a funționalităților*

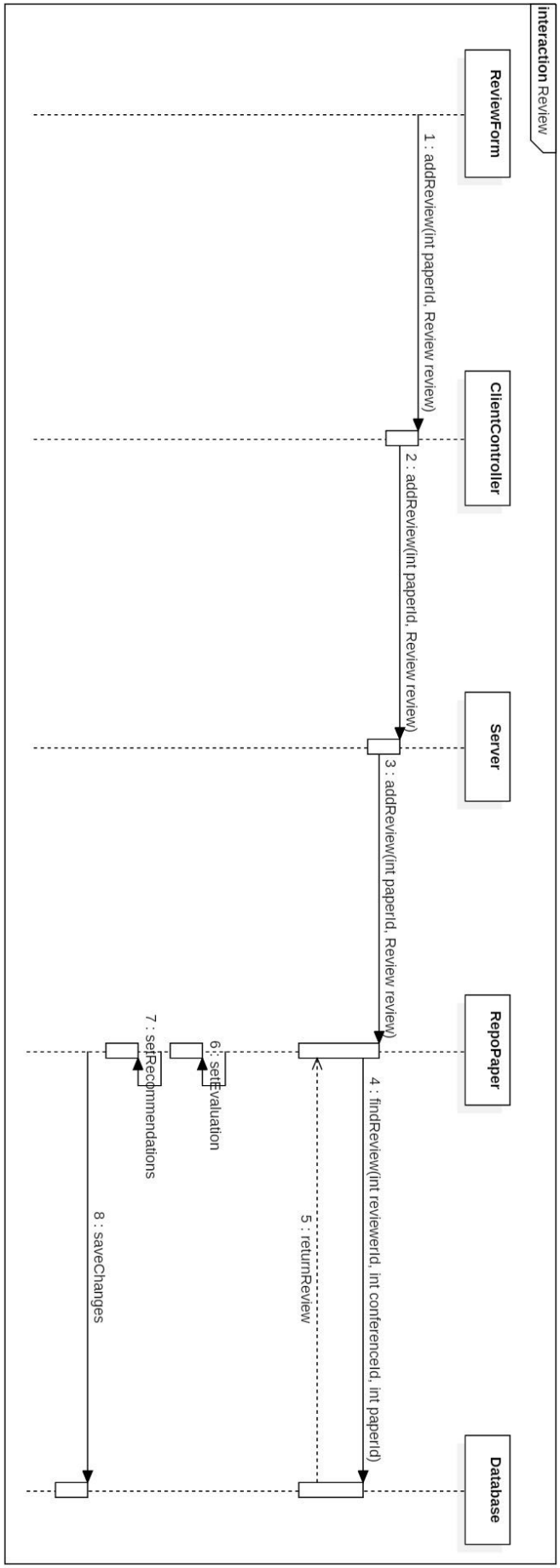


LOGIN

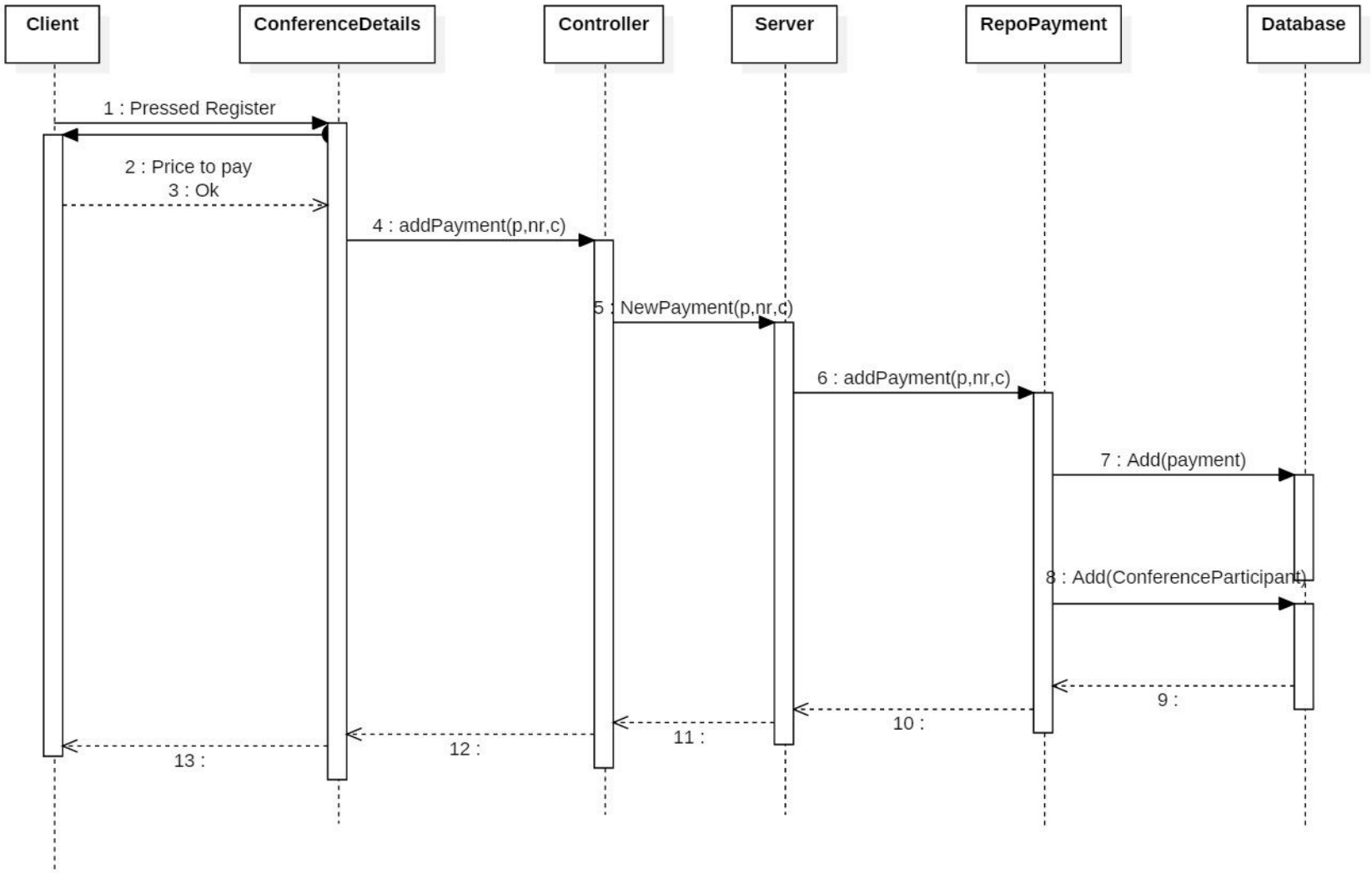
REGISTER


Client
RegisterForm
Controller
Server
RepoUser
Database
Pressed register button
Success message appears
register(username, password, name,affiliation, email,website PCMember)
true
Register(User)
true
uniqueUsername verification
Add(user)
allUsers
GetAll()
all.add(User)
Users
users()
Add
[online diagramming & design] creately.com

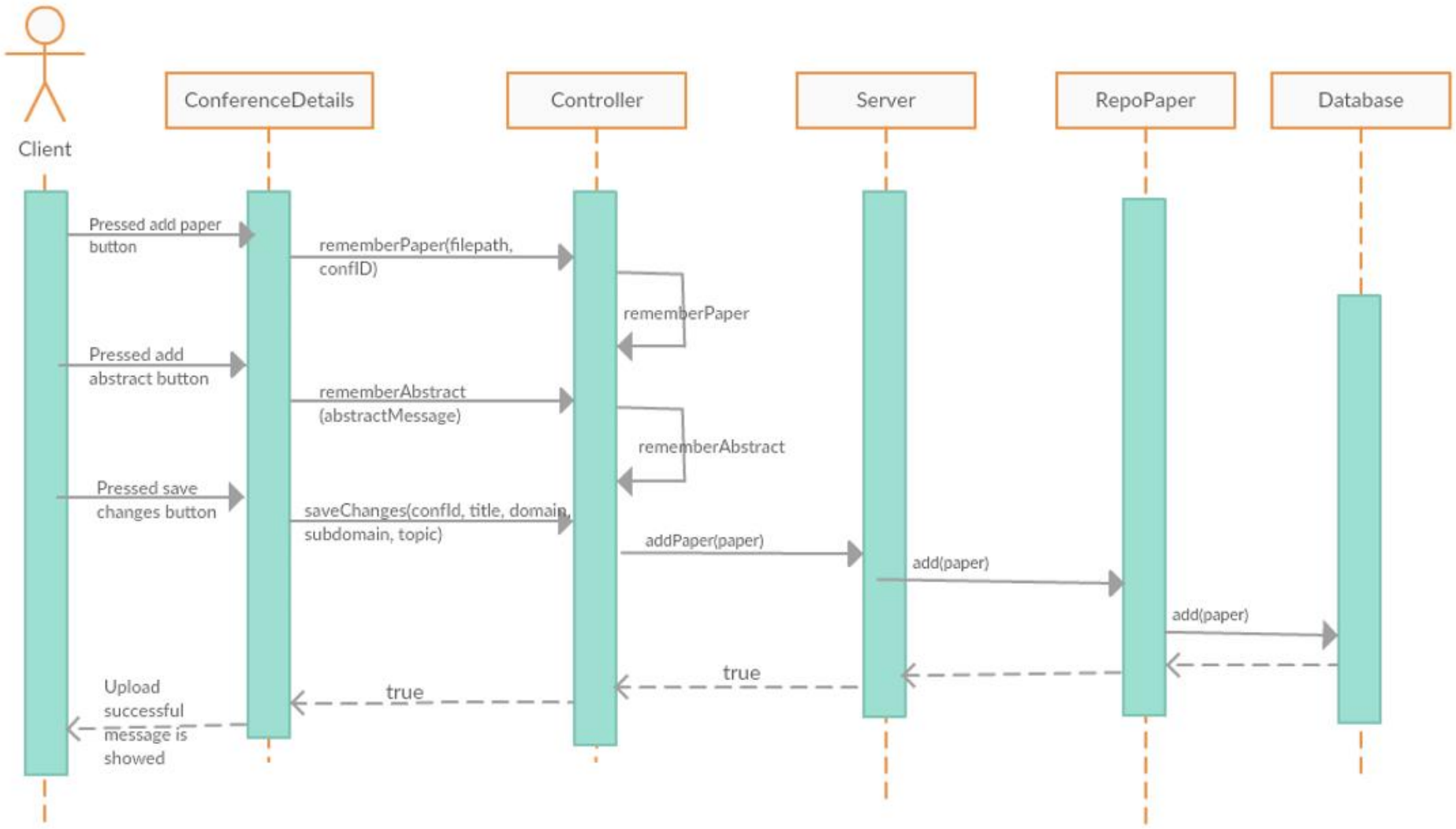
REVIEW


interaction Review
ReviewForm
ClientController
Server
RepoPaper
Database
1 : addReview(int paperId, Review review)
2 : addReview(int paperId, Review review)
3 : addReview(int paperId, Review review)
4 : findReview(int reviewerId, int conferenceId, int paperId)
5 : returnReview
6 : setEvaluation
7 : setRecommendations
8 : saveChanges

PAYMENT


Client
ConferenceDetails
Controller
Server
RepoPayment
Database
1 : Pressed Register
2 : Price to pay
3 : Ok
4 : addPayment(p,nr,c)
5 : NewPayment(p,nr,c)
6 : addPayment(p,nr,c)
7 : Add(payment)
8 : Add(ConferenceParticipant)
9 :
10 :
11 :
12 :
13 :

ADD PAPER


Client
ConferenceDetails
Controller
Server
RepoPaper
Database
Pressed add paper button
rememberPaper(filepath, confID)
rememberPaper
Pressed add abstract button
rememberAbstract (abstractMessage)
rememberAbstract
Pressed save changes button
saveChanges(confId, title, domain, subdomain, topic)
addPaper(paper)
add(paper)
add(paper)
true
true
Upload successful message is showed

Diagramele de clase

Diagrama de clasă înainte de implementare

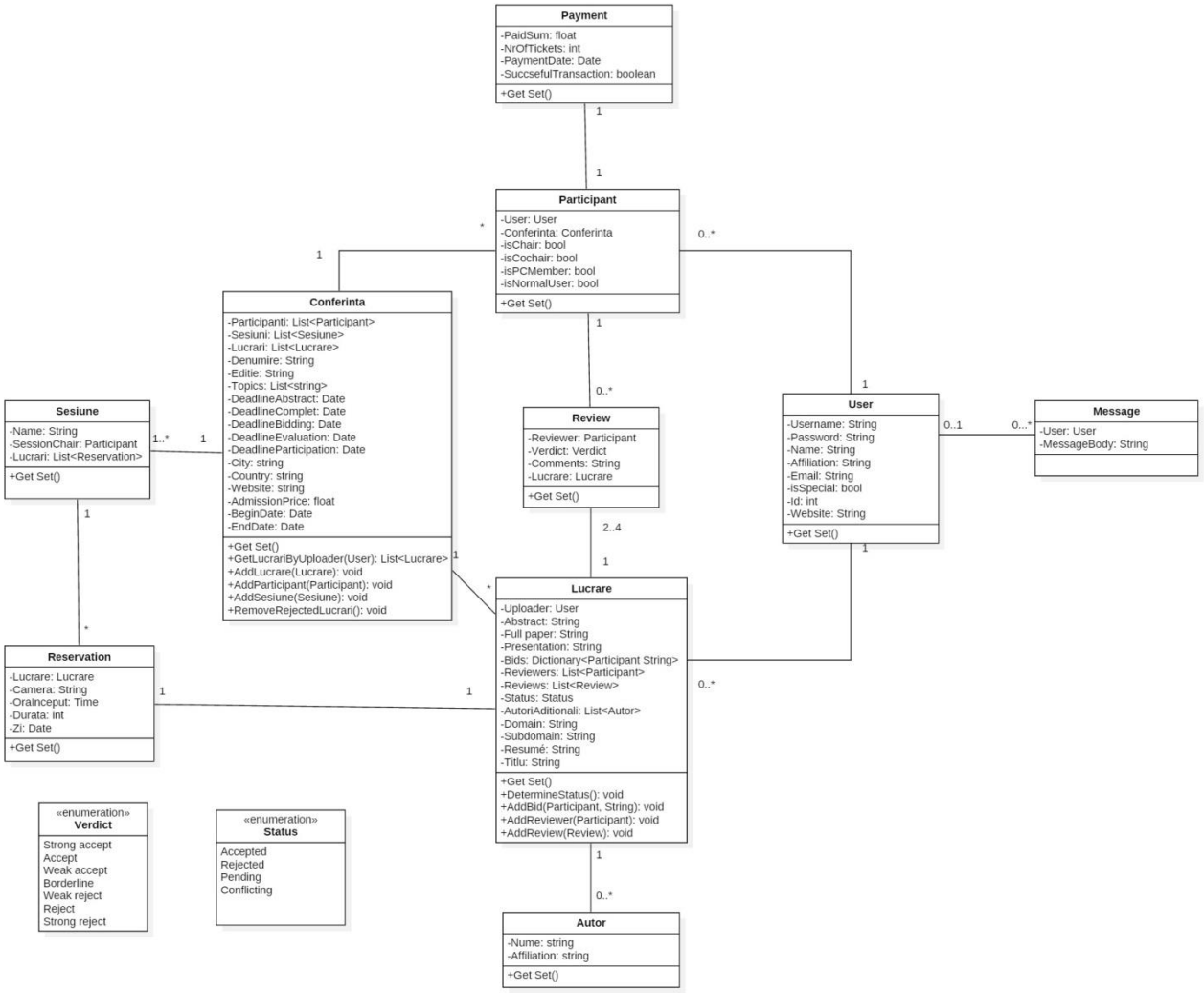


Diagrama de clasă după implementare

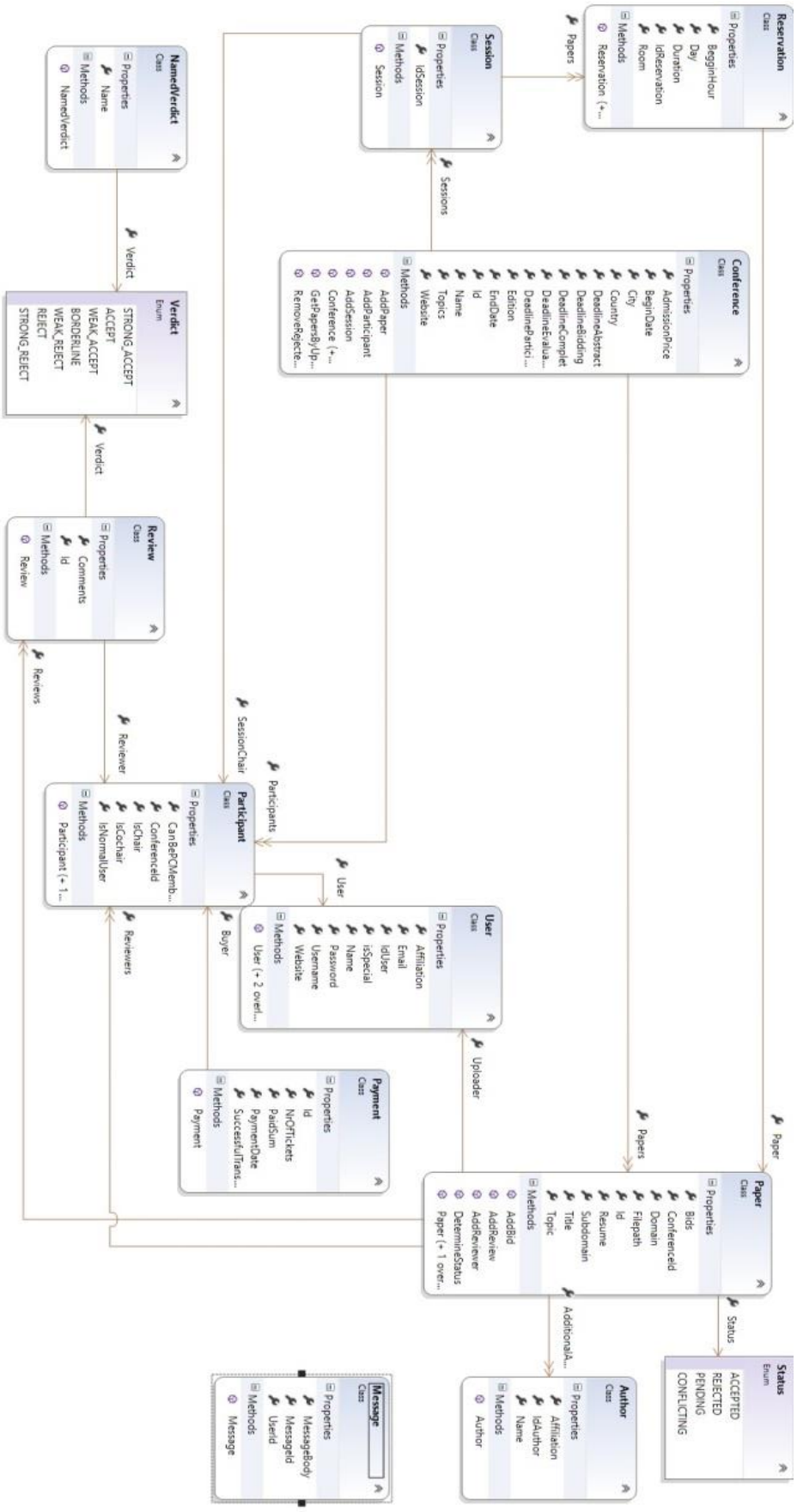
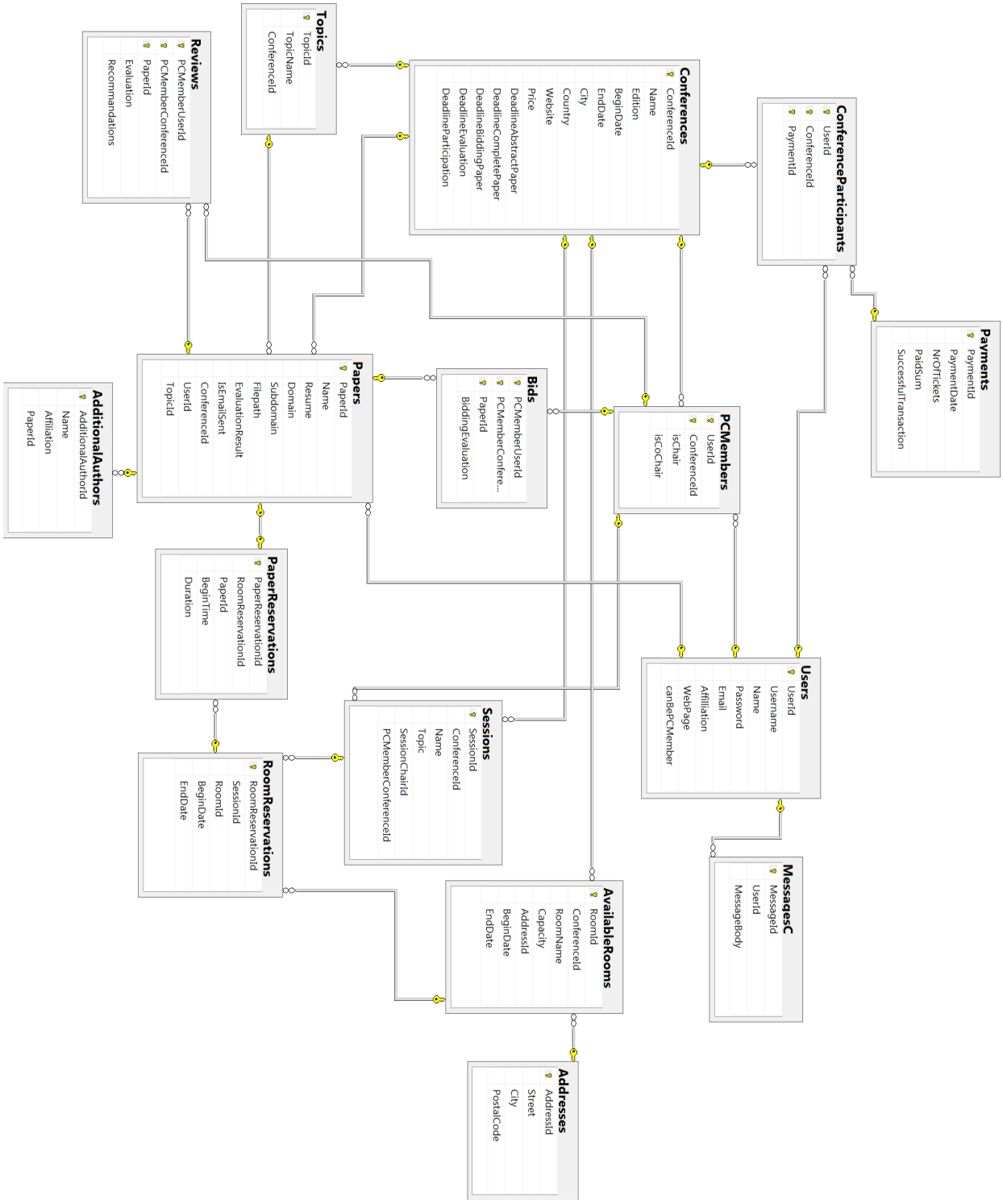


Diagrama bazei de date



Procesul de proiectare și implementare

Pentru implementarea proiectului s-a votat folosirea Microsoft SQL Serverului, ca limbaj de proiectare și gestionare a bazelor de date relaționale, și a C# ca limbaj de programare, atât pe partea de back-end, cât și pentru cea front-end. Pentru crearea diagramelor UML s-au folosit instrumente ce favorizează crearea acestor diagrame, StarUml si Creately(online).

Aplicația este client-server, lăsând astfel posibilitatea accesării simultane a mai multor utilizatori prin intermediul internetul, deoarece serverul împreună cu baze de date se află pe un calculator-server.

Prima etapă a fost analizarea cerințelor problemei primite și clarificarea rolurilor utilizatorilor în cadrul unei conferințe, precum și acțiunile posibile fiecărui rol. Cu aceasta etapă am realizat diagrama cazurilor de utilizare cu ajutorul intrumentului StarUml. Pe urmă s-a proiectat baza de date. Inițial pe foaie ca o schiță, apoi cu ajutorul ORM-ului Entity Framework. Acest ORM/intrument este specific microsoftului pentru a crea legătura între o bază de date, preferabil una de tipul MSQL, și limbajul de programare C#, precum și generarea bazei de date folosind clasele din modelul obiectual, generarea claselor din modelul relațional și generarea ambelor modele, obiectual, cât și relațional cu ajutorul secțiunii Designer. Entity Framework Designer permite crearea tabelelor în mod grafic. Simplu, îți creezi un tabel, în care adaugi atribute și pe aceste atribute poți seta tipul lor(int,Nvarchar,etc), precum și relații între alte tabele. Este reprezentarea bazei de date printr-o diagramă, scutind timp de implementare a scriptului SQL. Din acest instrument, Designer, s-a generat scriptul SQL ce conținea crearea bazei de date si a tabelelor acesteia. Din motive de ușurință a controlului bazei de date și motive de rapiditate, micile modificări ce au apărut pe parcurs și anumite detalii au fost implementate în acest script generat, deoarece la orice mică modificare trebuia regenerat scriptul. Însă acest instrument a fost folositor pentru implementarea bazei de date în mare într-un timp scurt.

Următorul pas a fost proiectarea diagramei de clase influențată de către diagrama bazei de date, clase implementate in C#. Trecându-se la etapa de implementare, 2 persoane au fost distribuite pentru lucrul cu baza de date, o persoană pe front-end și restul de 4 persoane pe back-end. Conexiunea între aplicația C# și baza de date SQL Server s-a realizat prin Entity Framework Model, iar conexiunea în C# între server și clienți s-a realizat prin .net remoting, tehnologie .NET care permite diferitelor procese și componente să interacționeze, stând la baza aplicațiilor distribuite .NET(tehnologie Microsoft).

Funcționalitățile aplicației C# au apelat proceduri de inserare a datelor, view-uri și funcții pentru preluarea datelor, scrise în SQL Server. Totodată, pentru că am considerat că un utilizator, deobicei chairul unei conferințe, nu ar trebui să se ocupe de programarea sălilor disponibile unei conferințe, programarea lucrărilor sau a distribuirii lucrărilor evaluatorilor, s-au implementat în baza de date proceduri de populare ale acestor tabele(generare de date) cu ajutorul datelor aflate în alte tabele, precum, popularea tabelei RoomReservations(după etapa de evaluare a lucrărilor) prin intermediul datelor tabelei AvailableRooms, tabelă ce reține camerele disponibile pentru o conferință introduce de către un administrator. Toate acestea sunt apelate de către un thread in C# ce verifică la fiecare 24 de ore în ce etapă se află fiecare conferință(înscreiere lucrări, evaluare lucrări, etc.).

Etapa de testare

Testarea software reprezintă o investigație dirijată în scopul de a obține informții legate de calitatea produsului software realizat în etapele anterioare.

Pentru testarea produsului nostru am folosit framework-ul Effort, un instrument ajutător pentru testarea aplicațiilor bazate pe Entity Framework. Effort este un provider ADO.NET care execută toate operațiile pe date într-o bază de date din memoria internă, în loc de o bază de date externă obișnuită. Acest framework simulează o bază de date reală, astfel testele pot rula independent de instanța bazei de date folosită în cadrul aplicației. Un alt beneficiu al acestui framework este faptul că fiecare test rulează cu o bază de date goală.

Testele făcute sunt teste unitare de tip white box și au fost realizate pe Repository pentru operatiile CRUD.

```
0 references | Dani Barac, 1 hour ago | 1 author, 1 change
public class UserRepoTest
{
    private ISSEntities2 _context;
    private RepoUserDB _repository;
    [TestInitialize]
    0 references | Dani Barac, 1 hour ago | 1 author, 1 change
    public void MyTestInitialize()
    {
        EffortProviderFactory.ResetDb();
        EntityConnection connection = EntityConnectionFactory.CreateTransient("name=ISSEntities2");
        //var connection = DbConnectionFactory.CreateTransient();
        _context = new ISSEntities2(connection);
        _repository = new RepoUserDB(_context);
    }
    1 reference | 0/1 passing | Dani Barac, 1 hour ago | 1 author, 1 change
    private void PrepareData()
    {
        Model.User userTest = new Model.User(2, "test", "test1", "Test", "Affiliation", "email@yahoo.com", false, "www.website.com");

        _repository.Add(userTest);
    }
    [TestMethod]
    | 0 references | Dani Barac, 1 hour ago | 1 author, 1 change
    public void TestMethod1()
    {
        PrepareData();
        List<Model.User> allUsers = _repository.GetAll();
        foreach (Model.User user in allUsers)
            Assert.AreEqual(user.Name, "Test");
    }
}
```