

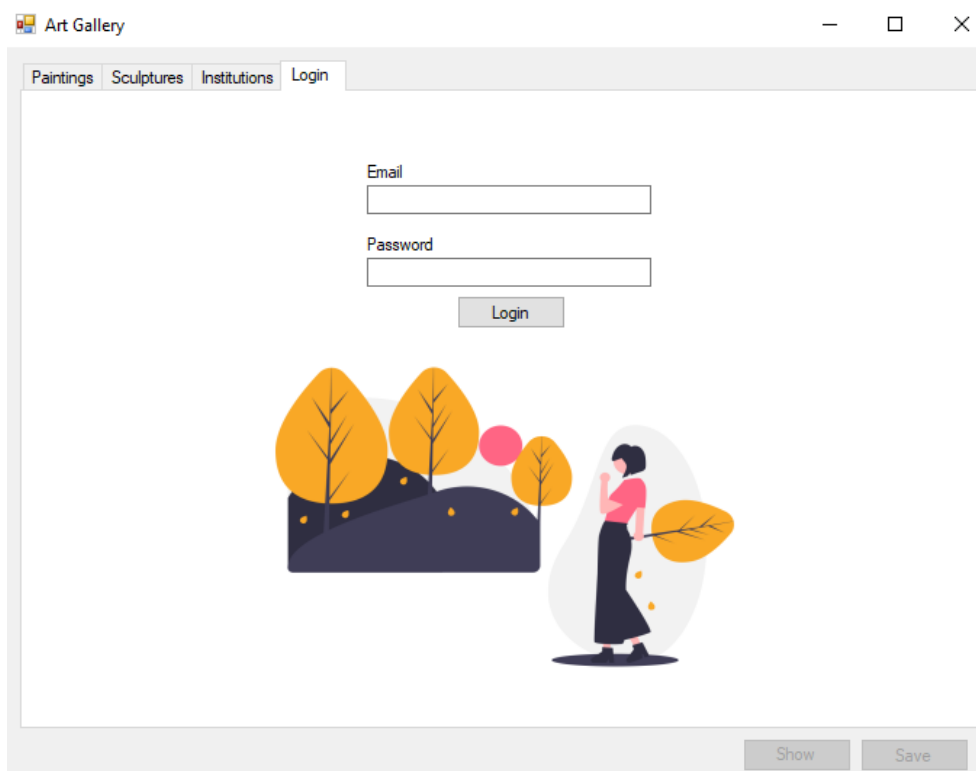


**UNIVERSITATEA TEHNICĂ**  
DIN CLUJ-NAPOCA

Facultatea de Automatică și Calculatoare, CTI

## Documentație

### Galerie de artă – Proiect Proiectare software



Profesor îndrumător:  
Anca Iordan

Student:  
Astaliș Lorena-Maria  
Grupa: 30234

An3, 2022

## Cuprins

1. Introducere
2. Analiză
3. Proiectare
4. Implementare
5. Testare
6. Concluzii

## 1. Introducere

### Cerință proiect

Dezvoltați (analiză, proiectare, implementare) o aplicație desktop care poate fi utilizată în instituții unde pot fi expuse opere de artă: galerii de artă sau muzee. Conceptul de operă de artă plastică este caracterizată de următoarele caracteristici: titlul, numele artistului și anul realizării. Din această clasă se vor deriva clasele corespunzătoare conceptelor tablou și sculptură. Sculptura va avea ca atribut tipul de sculptură (altorelief, basorelief, relief, statuie, statuie ecvestră, etc.), iar tabloul va avea ca attribute genul picturii (peisaj, portret, etc.) și tehnica utilizată (ulei, acuarelă, ceară, frescă, etc.). Aplicația va avea 3 tipuri de utilizatori: vizitator al instituției (galerie/muzeu) de artă, angajat al lanțului de instituții de artă și administrator.

Utilizatorii de tip vizitator pot efectua următoarele operații fără autentificare:

- ❖ Vizualizarea tuturor operelor de artă expuse în aceste instituții de artă;
- ❖ Filtrarea listei operelor de artă plastică după urm. criterii: instituția de artă, artist, tipul operei de artă;
- ❖ Căutarea unei opere de artă după titlu.

Utilizatorii de tip angajat al lanțului de instituții de artă pot efectua următoarele operații după autentificare:

- ❖ Toate operațiile permise utilizatorilor de tip vizitator;
- ❖ Operații CRUD în ceea ce privește persistența operelor de artă expuse;
- ❖ Salvare rapoarte/liste cu operele de artă în mai multe formate: csv, json.

Utilizatorii de tip administrator pot efectua următoarele operații după autentificare:

- ❖ Toate operațiile permise utilizatorilor de tip vizitator;
- ❖ Operații CRUD pentru informațiile legate de utilizatorii care necesită autentificare.

## 2. Analiză

În această etapă de dezvoltare a proiectului se va face diagrama de cazuri. Acestea sunt identificate din cerințele problemei enunțate la capitolul 1.

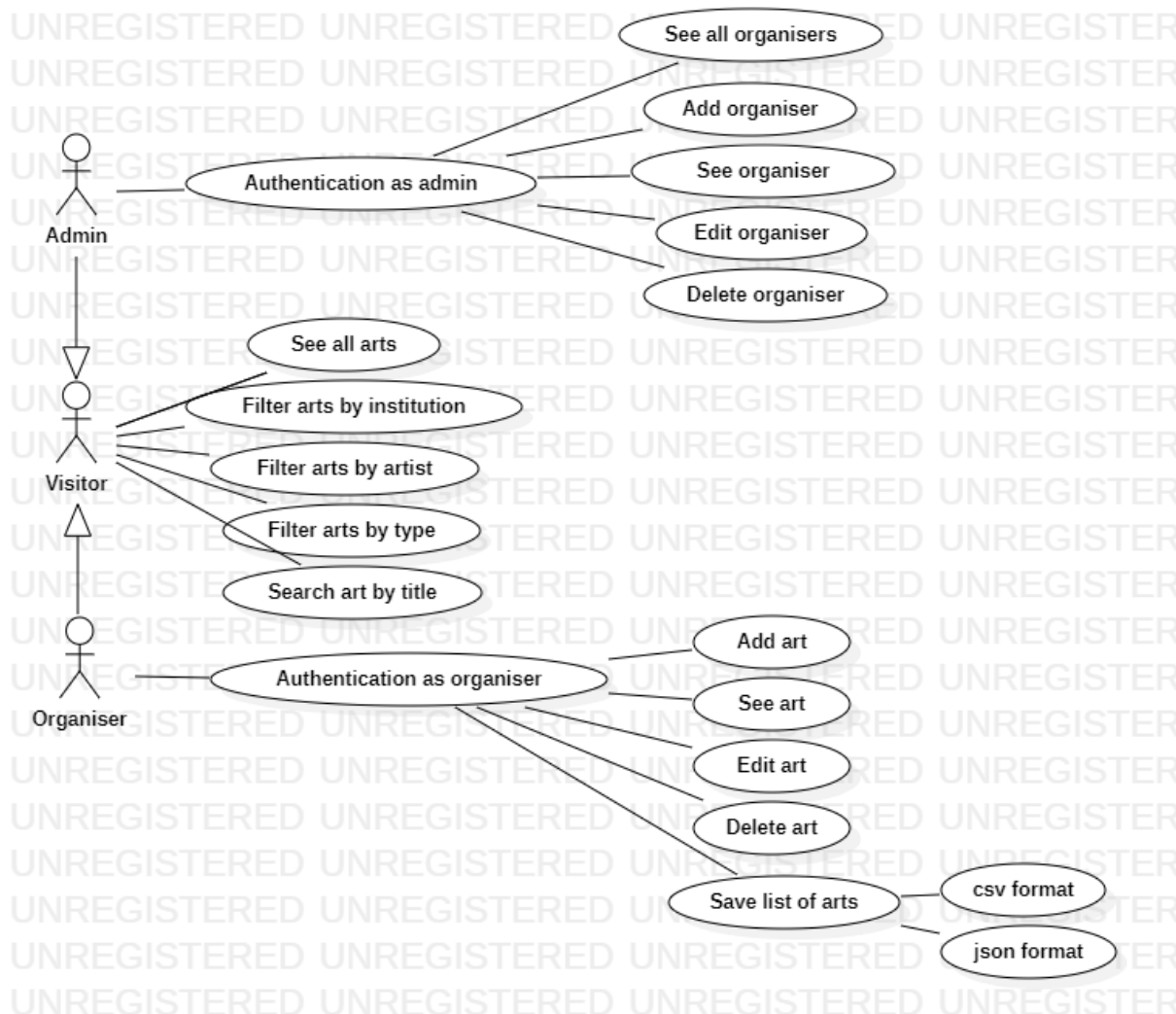


Figura 2.1: Diagrama cazurilor de utilizare

## 3. Proiectare

În această etapă de dezvoltare se identifică după digrama de cazuri, modelele de care este nevoie pentru a modela cerințele din lumea reală în obiecte, după atributele care le descriu. Pe lângă aceasta, se vor adăuga și clasele ce țin de interfața utilizatorului și clasele care vor lega modelele de interfață (controller). Controllerul joacă rolul principal în a lega partea de model cu partea de view. Modelul este de sine stătător, la fel și view-ul, nu se amestecă responsabilitățile. Controllerul se ocupă de a lega partea le model din spate cu ceea ce se poate observa pe view, acesta implementează interfața Observer care este formată dintr-o metodă de tip void care nu primește niciun paramentru, metodă care se numește update. Avem nevoie de aceasta pentru a face re-render la componentele care în spate și-au schimbat valorile. Implementarea pe care am ales-o pentru șablonul de proiectare Model-View-Controller este activ.

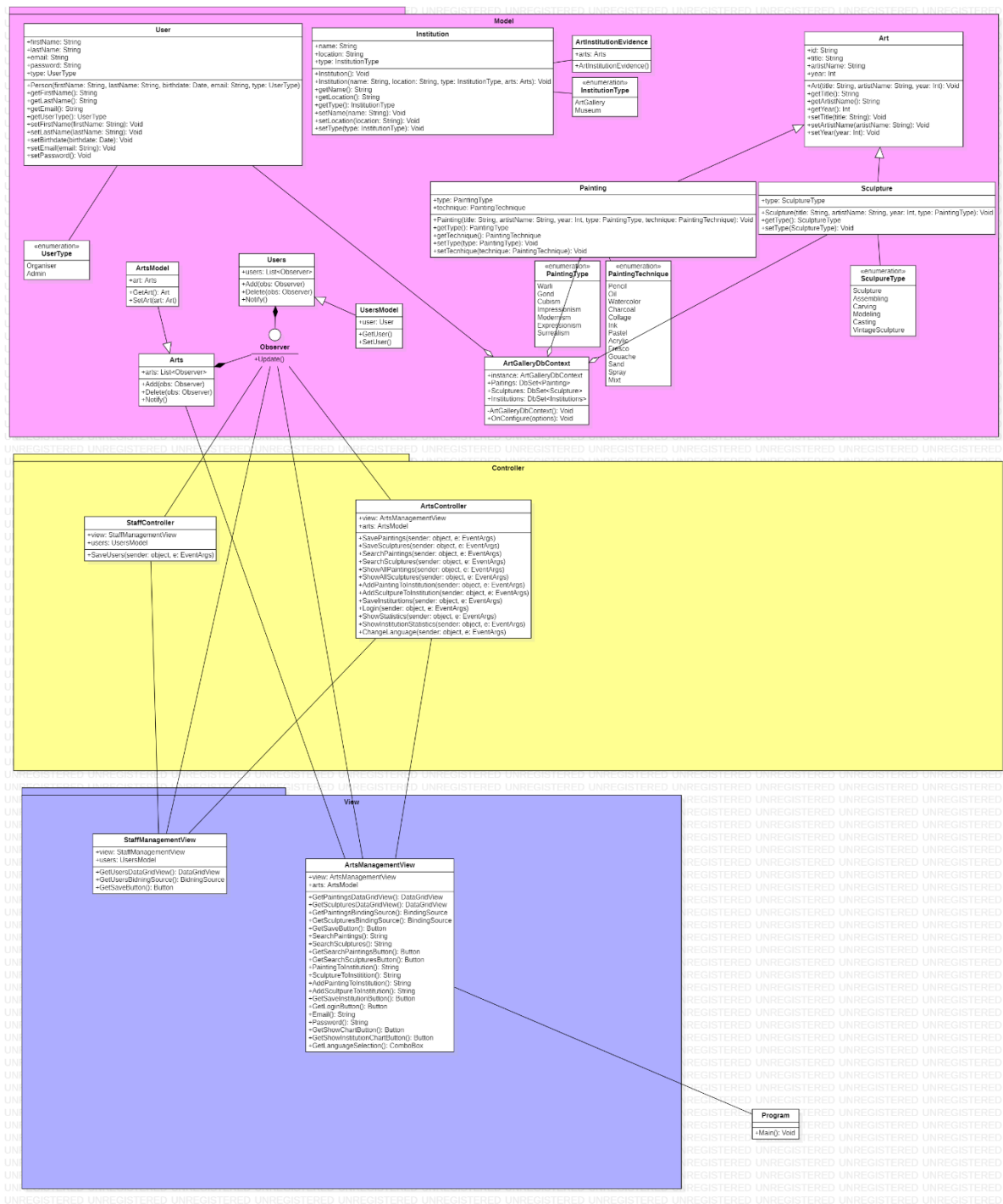


Figura 3.1: Diagrama UML

Pentru clasa care ține conexiunea cu baza de date am ales un design pattern creațional Singleton. Am făcut această alegere din motivul că baza de date este una singură, așa că și conexiunea la aceasta este esențial să fie una singură. Lucrul acesta l-am realizat cu un constructor privat, iar câmpul de instance este static, adică e doar unul și ține de clasă, decât de instanța obiectului.

#### 4. Implementare

S-a ales pentru implementarea acestui limbajul C#, împreună cu framework-ul de .NET din motivele:

- Este ușor de învățat a face interfețe grafice desktop;
- .NET are intergrate multe librării utile care vor ajuta dezvoltările ulterioare ale proiectului;
- Versatil: se pot implementa mai multe lucruri pentru diferite platforme, de exemplu: server, desktop, web, mobile;
- Developer tools: instrumente de debugging performante;
- Standardizarea deprinderilor dobândite;
- .NET Core este un framework open source;

Ca persistență s-a folosit PostgreSQL ca bază de date, cu ajutorul framework-ului EntityFramework care implementează o clasă numită DbContext. În această clasă are loc conexiunea la baza de date.

#### 5. Testare

Cont: Organizator – [lorena@organiser.com](mailto:lorena@organiser.com) și parola: password

Administrator – [lorena@admin.com](mailto:lorena@admin.com) și parola: password

Interfață vizitator:

Paintings Sculptures Institutions Login

	Type	Technique	Id	Title	ArtistName
▶	Modern	Oil	1	Mona	Lisa
	Renininf	Mixt	2	Haha	Lore
*					

Search

Show all



Show

Save

Paintings Sculptures Institutions Login

	Type	Id	Title	ArtistName	Year
▶	Here	2	23432	324	324
	Herewer	1	is	some	4345
	Unknownwer	3	werwer	werwer	1902
*					

Search

Show all

Show

Save



Art Gallery

Paintings

Sculptures

Institutions

Login

	Id	Name	Location	Paintings	Sculptures
▶	0	Scoala de arte	Cluj		
	1	Muzeul de arte	Cluj-Napoca		
*					

Institution id

Painting id

Add

Sculpture id

Add

ShowSave

Interfață logare:

Art Gallery

Paintings

Sculptures

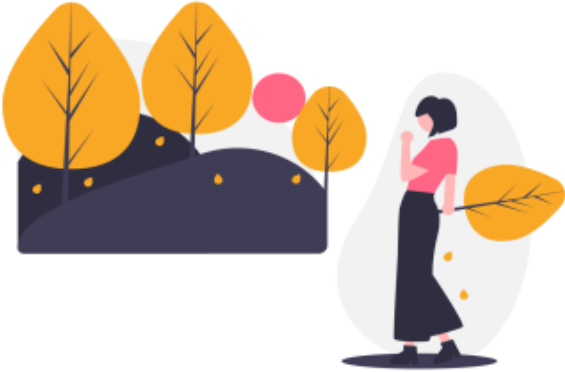
Institutions

Login

Email

Password

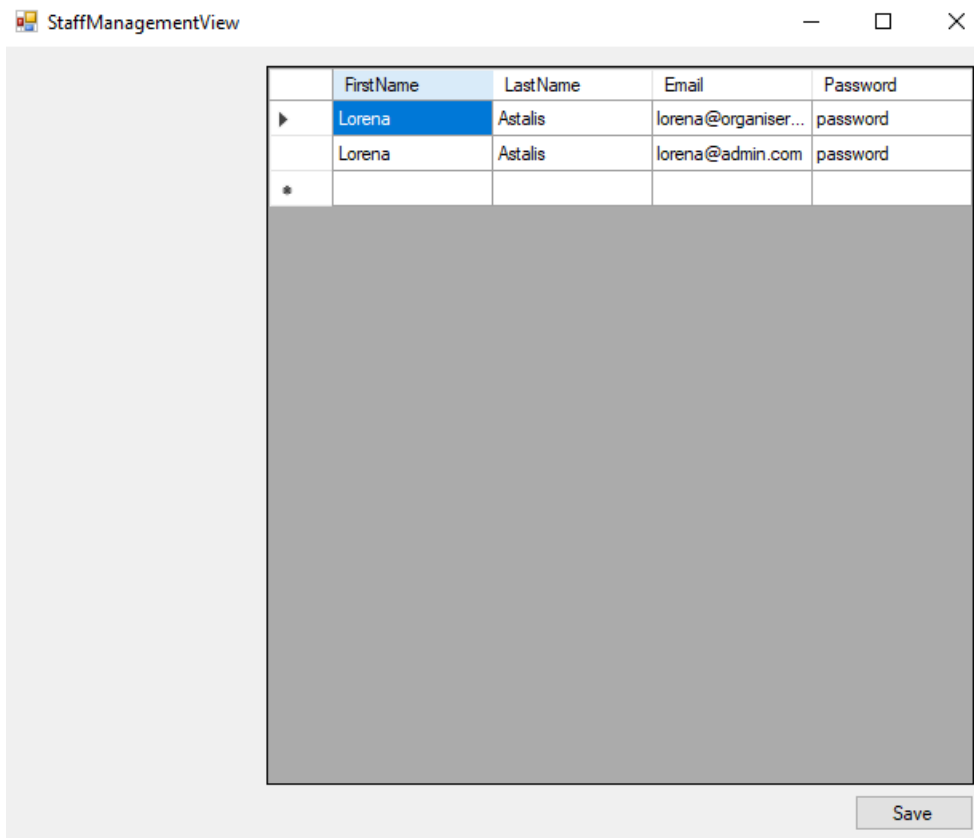
Login



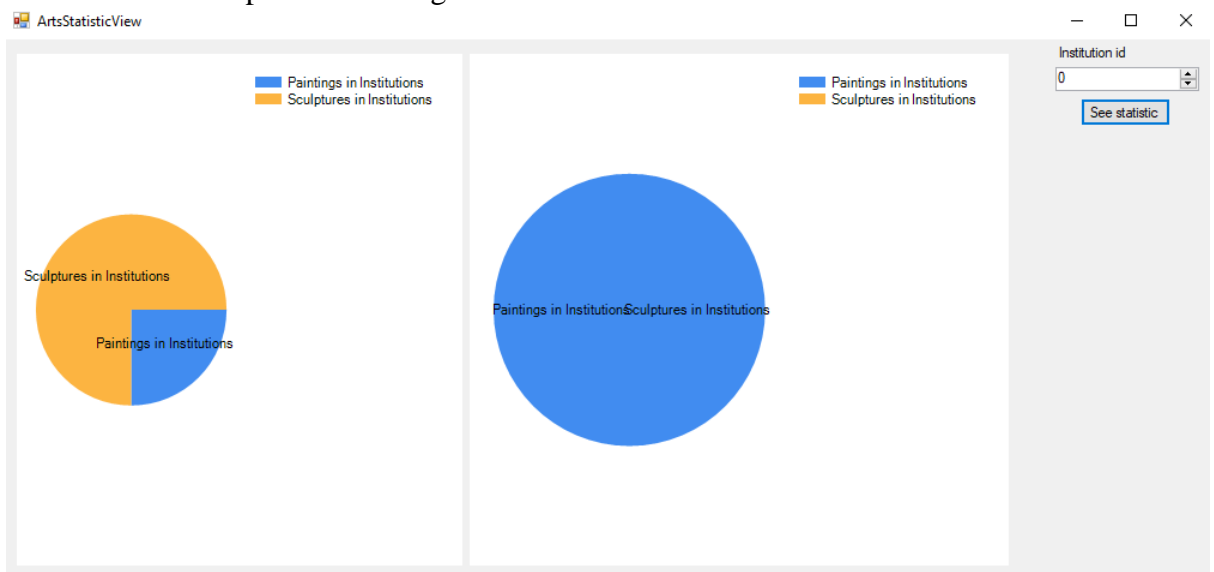
Show

Save

Interfață administrator:



Generare statistici pentru rolul organizator:



## 6. Concluzii

Din acest proiect am învățat MVC pentru o mică aplicație desktop, toată arhitectura care leagă modelul de interfața grafică prin controller cu ajutorul interfeței Observer.