



---

## Tema 2 - Sablonul arhitectural MVVM

*Proiectare Software*

---

Autor: Mirisan Octavian

Grupa: 30231

FACULTATEA DE AUTOMATICA  
SI CALCULATOARE

8 Aprilie 2025

## Cuprins

<b>1</b>	<b>Enuntul Problemei</b>	<b>2</b>
<b>2</b>	<b>Instrumente utilizate</b>	<b>2</b>
<b>3</b>	<b>Justificarea limbajului de programare</b>	<b>2</b>
<b>4</b>	<b>Descriere diagrame UML</b>	<b>3</b>
4.1	Diagrama de Use Case	3
4.2	Diagramele de activitate	4
4.3	Diagrama de pachete	8
4.4	Diagrama de clase	9
4.5	Diagrama entitate-relatie	10
<b>5</b>	<b>Descriere aplicatie</b>	<b>11</b>
5.1	Interfata utilizatorului	11

## 1 Enuntul Problemei

### **Problema 5**

Dezvoltați o aplicație soft care poate fi utilizată într-o galerie de artă pentru gestiunea operelor de artă expuse. Softul va permite:

- ❖ Adăugarea, ștergerea și actualizarea artiștilor care au expuse opere de artă în galeria de artă;
- ❖ Vizualizarea listei tuturor artiștilor care au expuse opere de artă în galeria de artă (pentru fiecare artist se va afișa numele, data nașterii, locul nașterii, naționalitatea, o fotografie și lista tuturor operelor de artă realizate de artist și expuse în acest muzeu);
- ❖ Căutarea unui artist după nume;
- ❖ Adăugarea, ștergerea și actualizarea operelor de artă expuse în galeria de artă;
- ❖ Vizualizarea listei tuturor operelor de artă expuse în galeria de artă sortate după artist (vizualizarea include și redarea unor imagini cu operele de artă; între 1 și 3 imagini pentru fiecare opera de artă);
- ❖ Căutarea unei opere de artă după titlu;
- ❖ Filtrarea listei operelor de artă după următoarele criterii: artist, preț sau tipul operei de artă;
- ❖ Salvarea listei cu operele de artă expuse în fișiere de tip csv și doc.

Figura 1: Enuntul problemei

## 2 Instrumente utilizate

Pentru dezvoltarea aplicatiei au fost folosite urmatoarele instrumente

- **Visual Studio 2022** mediul principal de dezvoltare, ales pentru suportul solid pentru limbajul C#, design-ul interfețelor WPF, și integrarea nativă cu .NET
- **.NET Framework 4.8** platforma pe care este construită aplicatia, oferind suport pentru WPF, MVVM și bibliotecile necesare pentru binding și UI
- **SQL Server (LocalDB)** motorul de baze de date folosit pentru stocarea informațiilor despre artiști, opere de artă și imagini asociate
- **SQL Server Management Studio** utilizat pentru administrarea bazei de date și rularea manuală a comenzi SQL în etapa de testare
- **WPF XAML Designer** pentru design vizual al interfeței aplicatiei

## 3 Justificarea limbajului de programare

Limbajul C# a fost ales pentru implementarea aplicatiei deoarece:

- Este un limbaj modern, robust, orientat pe obiect, cu suport nativ pentru arhitectura MVVM datorită WPF și binding-ului nativ
- Permite dezvoltarea rapidă a interfețelor grafice prin intermediul XAML, un limbaj de markup foarte expresiv
- Integrarea cu SQL Server este facilă prin biblioteci dedicate (ex: System.Data.SqlClient)
- C# oferă o sintaxă clara și o bună organizare a codului, permitând o implementare eleganta și extensibilă a aplicațiilor de tip CRUD
- Framework-ul .NET include un ecosistem complet care acoperă nevoile unei aplicații desktop: interfață, baze de date, fisiere, rapoarte etc.

## 4 Descriere diagrame UML

### 4.1 Diagrama de Use Case

Diagrama de use case descrie interactiunile utilizatorului cu aplicatia de gestionare a unei galerii de arta. Utilizatorul, reprezentat ca actor, poate efectua actiuni precum adaugarea, stergerea si actualizarea artistilor si operelor de arta, precum si vizualizarea listelor acestora. De asemenea, utilizatorul are posibilitatea de a cauta opere de arta, de a filtra liste si de a salva liste de opere pentru export. Sistemul, denumit "Aplicatie", incapsuleaza toate aceste functionalitati, oferind o interfata centralizata pentru gestionarea datelor galeriei.

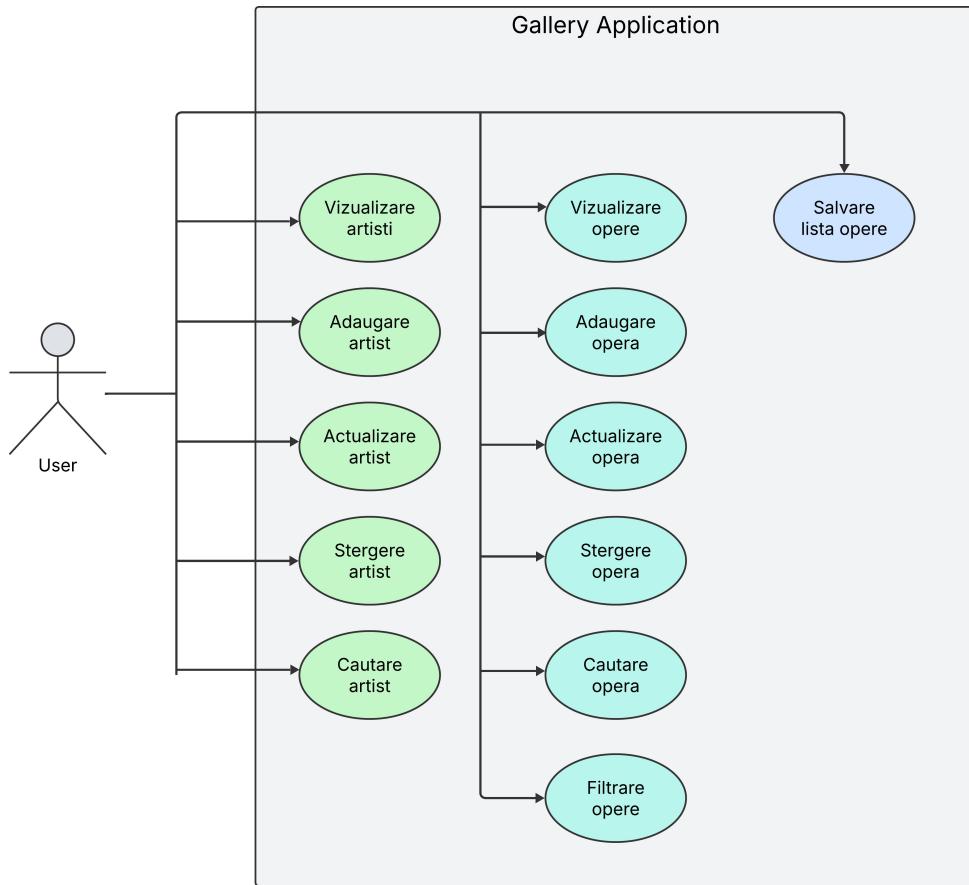


Figura 2: Diagrama de Use Case

## 4.2 Diagramele de activitate

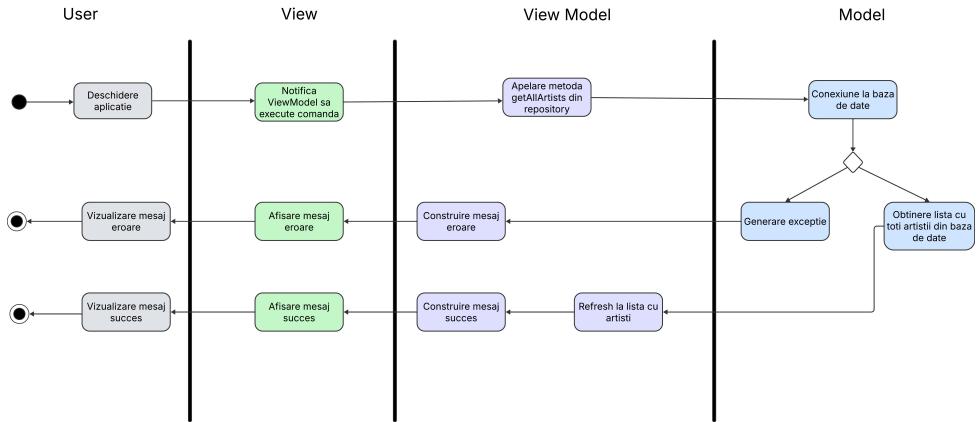


Figura 3: Vizualizare artiști.

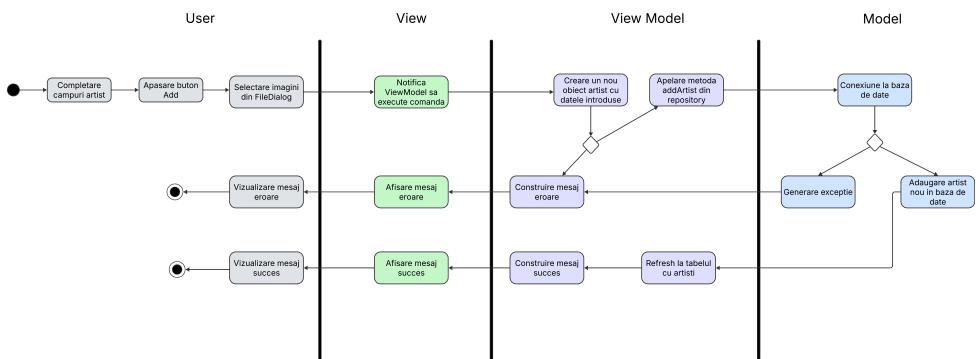


Figura 4: Adaugare artist in baza de date.

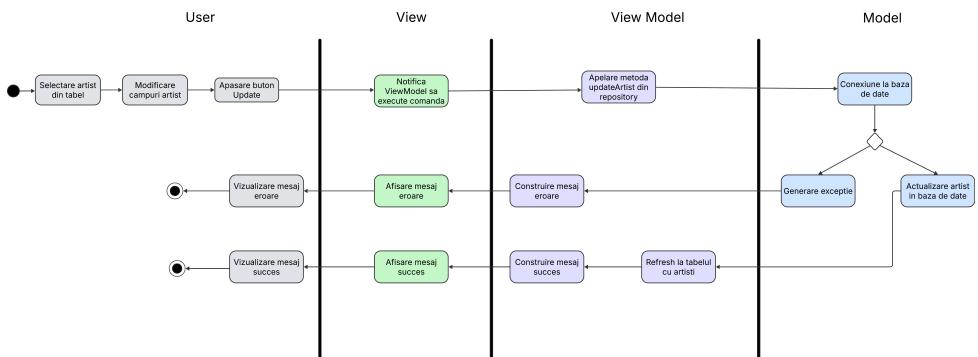


Figura 5: Editare artist in baza de date.

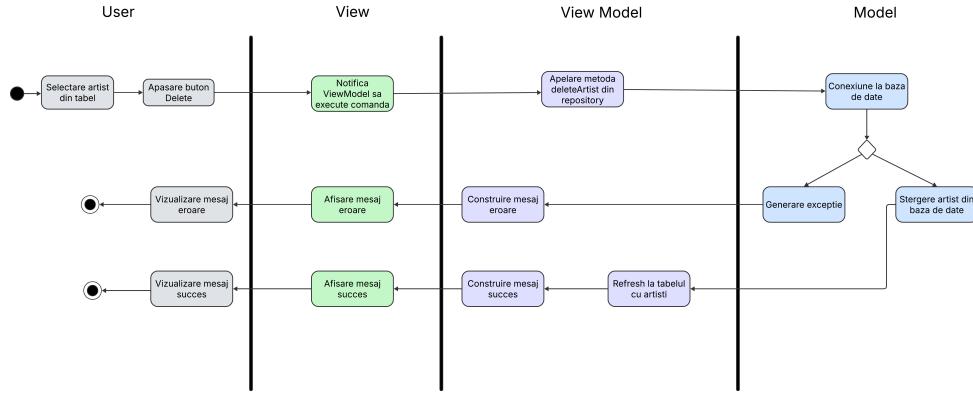


Figura 6: Stergere artist din baza de date.

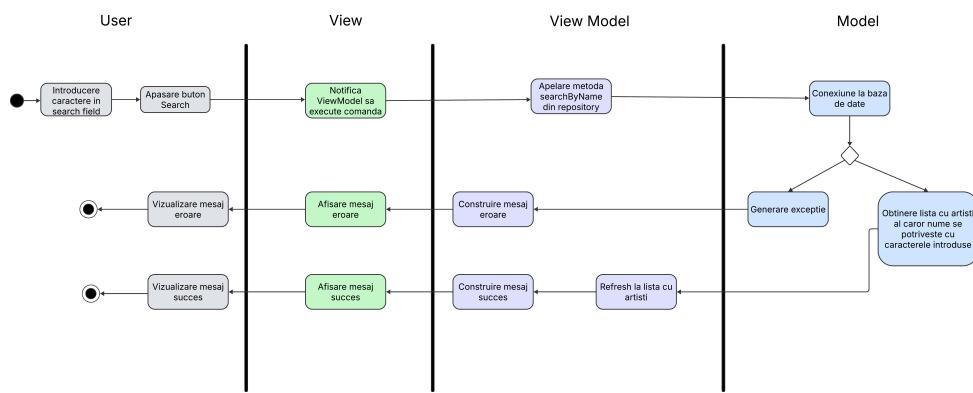


Figura 7: Cautare artist dupa nume.

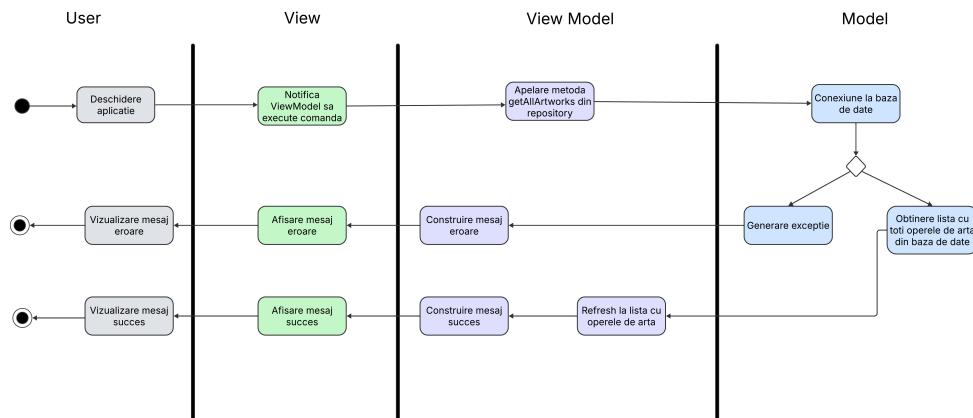


Figura 8: Vizualizare opere de arta.

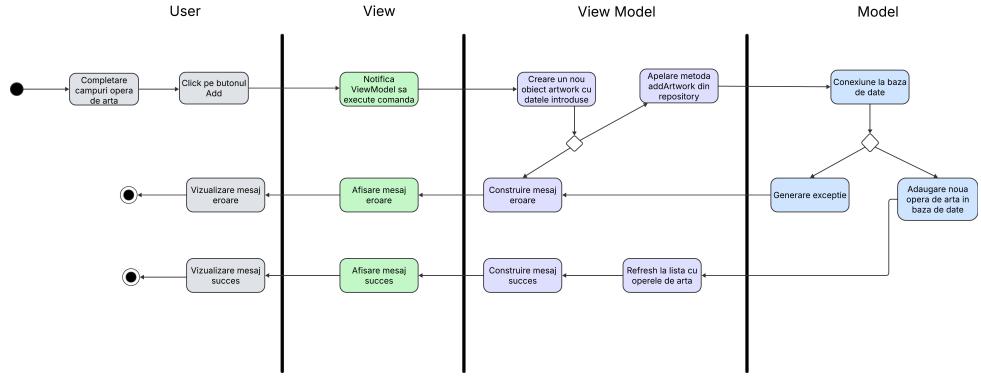


Figura 9: Adaugare opera de arta in baza de date.

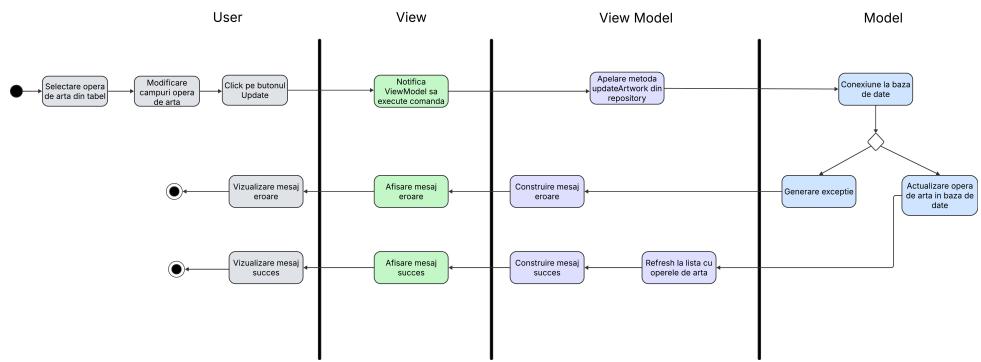


Figura 10: Editare opera de arta in baza de date.

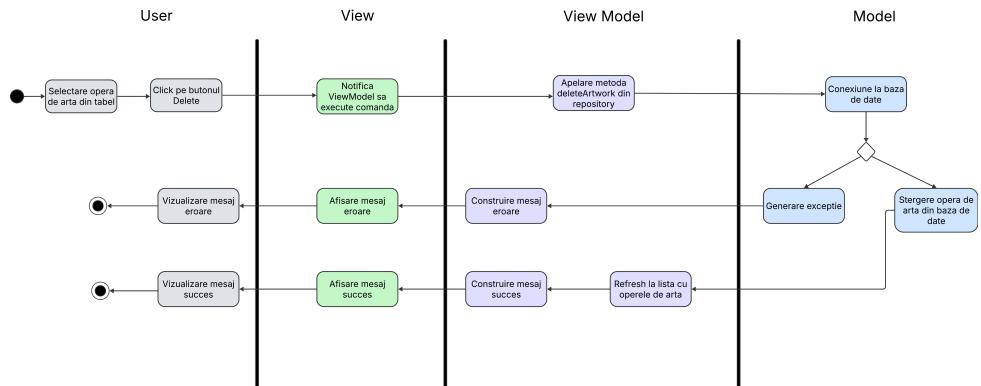


Figura 11: Stergere opera de arta din baza de date.

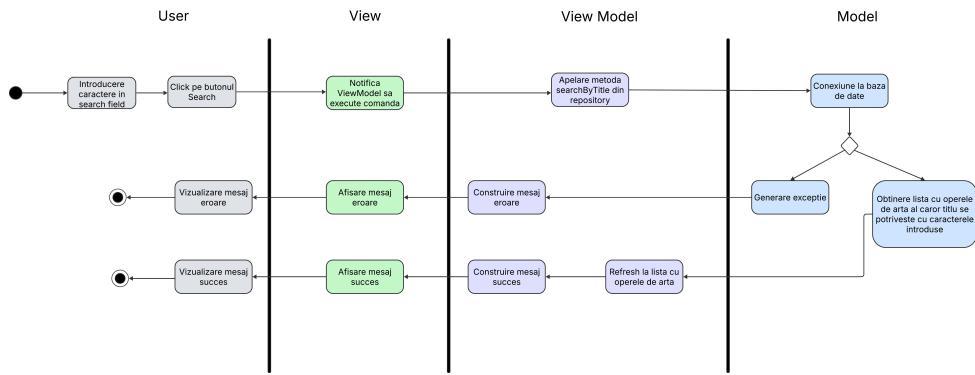


Figura 12: Cautare opera de artă după titlu.

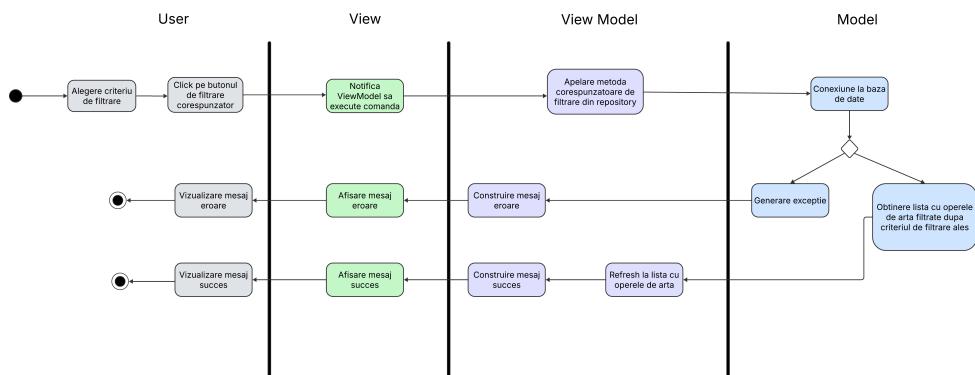


Figura 13: Filtrare opere de artă.

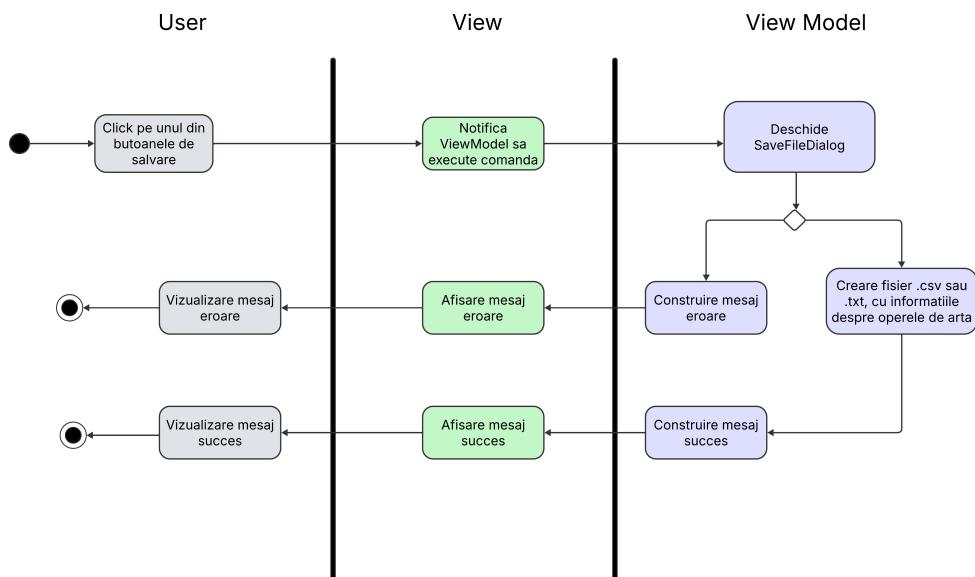


Figura 14: Export opere de artă în format .csv sau .doc

### 4.3 Diagrama de pachete

- **Model** – conține entitățile principale ale aplicației: `Artist`, `Artwork`, `ArtworkImage`, precum și pachetul `Repository`, responsabil de accesul și manipularea datelor persistente din baza de date.
- **ViewModel** – mediază între Model și View. Pachetul conține logica aplicației și starea curentă (prin clasa `GalleryViewModel`). Include și pachetul `Commands`, care definește acțiuni declanșate de interfață (prin `GalleryCommand`).
- **View** – conține elementele vizuale ale aplicației (fișiere XAML și cod-behind), fiind responsabil de afișare și interacțiune cu utilizatorul. Acest pachet este conectat la ViewModel prin data binding.

#### Relații între pachete:

- **ViewModel** depinde de **Model**, apelând metodele repository-urilor și lucrând cu obiectele de tip `Artist`, `Artwork` etc.
- **View** depinde de **ViewModel**, legând UI-ul de proprietățile și comenzi expuse în `GalleryViewModel`.

Această structură promovează separarea responsabilităților, testabilitatea și reutilizarea componentelor.

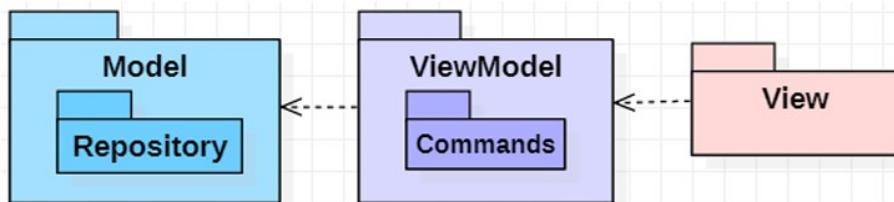


Figura 15: Diagrama de pachete a aplicatiei

## 4.4 Diagrama de clase

O diagramă de clase este o diagramă UML care descrie structura statică a sistemului, evidențiind clasele, atributele, metodele și relațiile dintre ele (asocieri, agregări, compozitii, moșteniri).

În cazul aplicației de față, diagrama de clase reflectă o arhitectură **MVVM**, unde:

- Pachetul **Model** include entitățile **Artist**, **Artwork**, **ArtworkImage** și clasele de tip **repository** care gestionează persistarea lor.
- Clasa **GalleryViewModel** din pachetul **ViewModel** mediază între model și interfață, expunând colecții, proprietăți și comenzi (**ICommand**).
- Clasa **GalleryCommand** implementează acțiuni logice declanșate de interfață (ex: adăugare, stergere, filtrare).
- Pachetul **View** conține clasa **GalleryGUI**, care definește interfața XAML și legăturile către ViewModel.

Această diagramă asigură o separare clară a responsabilităților, favorizând mentenanța și extensibilitatea aplicației.

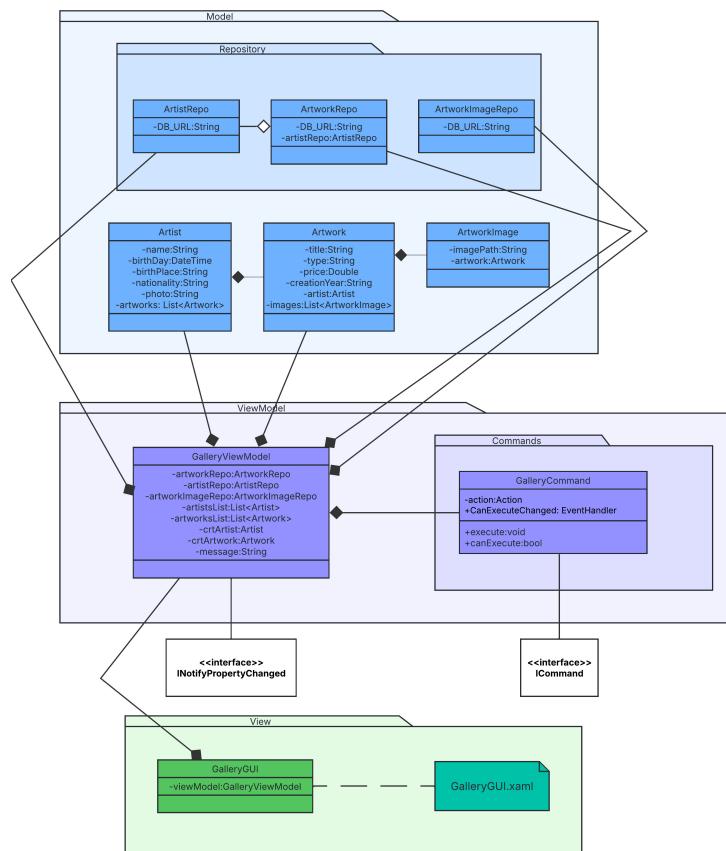


Figura 16: Diagrama de clase

## 4.5 Diagrama entitate-relatie

O diagramă entitate-relație (ER) este o reprezentare grafică a structurii unei baze de date, evidențiind entitățile (tabelele), atributele și relațiile dintre ele.

ER-diagrama reflectă următoarele entități:

- **Artist** – entitate principală, conține detalii despre artist (nume, dată naștere, naționalitate etc.).
- **Artwork** – entitate dependentă, legată de **Artist** printr-o relație de tip **1:N** (un artist are mai multe opere).
- **Artwork\_Images** – entitate ce conține căile către fișierele imagine asociate unei opere. Este legată de **Artwork** printr-o relație **1:N**.

Această structură este normalizată, clară și eficientă pentru o aplicație de tip galerie de artă, în care operele și artiștii sunt gestionati separat.

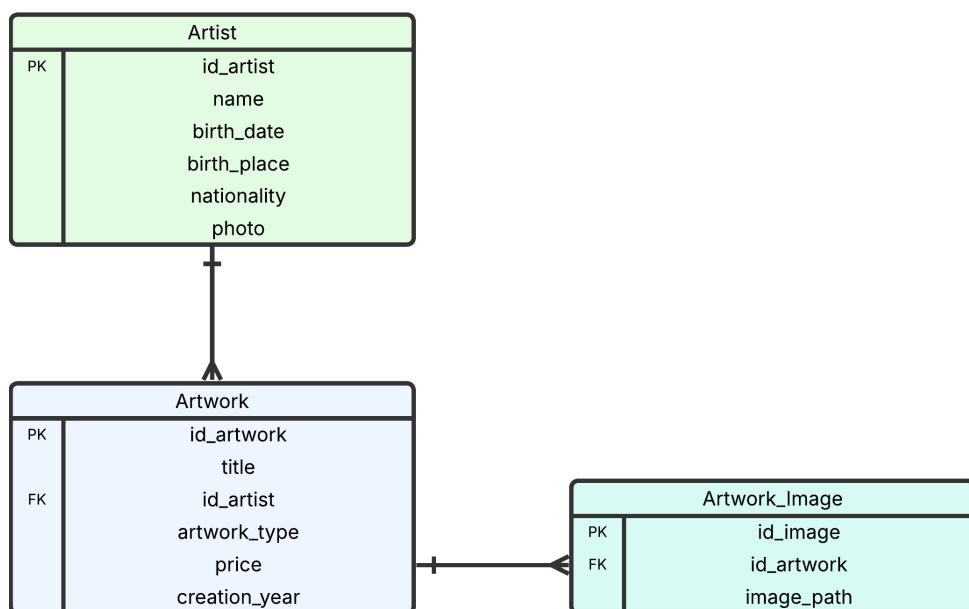


Figura 17: Diagrama entitate-relatie

## 5 Descriere aplicatie

### 5.1 Interfata utilizatorului

Interfața aplicației este organizată pe două secțiuni principale: gestionarea artiștilor și gestionarea operelor de artă. Utilizatorul poate introduce, actualiza, șterge sau căuta artiști și opere, fiecare acțiune fiind declanșată prin butoane dedicate. Datele sunt afișate în două tabele (DataGridView), iar imaginile corespunzătoare artiștilor și operelor sunt vizualizate în timp real. Funcționalitățile de filtrare (după artist, tip și preț) și opțiunile de export în format .csv sau .txt oferă o experiență completă și intuitivă. Interfața este construită în WPF și legată de ViewModel prin binding, respectând arhitectura MVVM.

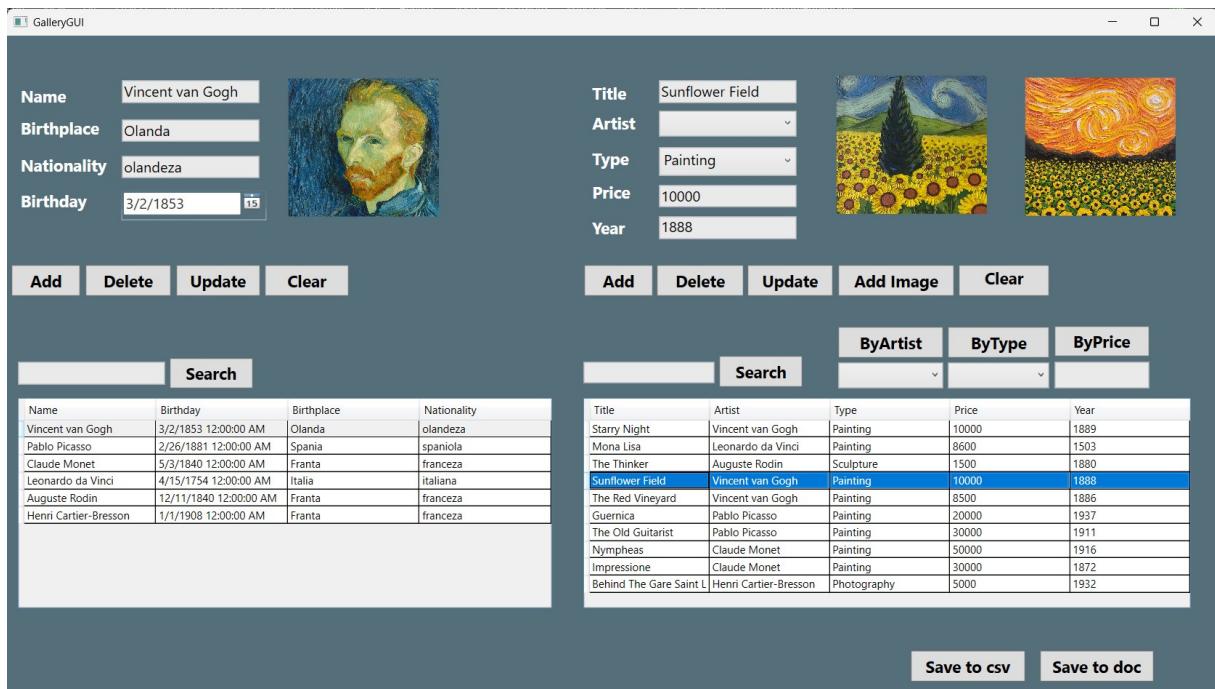


Figura 18: Interfata Utilizatorului