DOCUMENTAȚIE TEMA 3 - SISTEM DE GESTIUNE PENTRU MUZEE

Craciun Alexandru

Grupa: 30237

1. Enuntul problemei

Dezvoltați o aplicație soft care poate fi utilizată în muzee pentru gestiunea operelor de artă expuse. Softul

va permite:

❖ Adăugarea, ștergerea și actualizarea artiștilor care au expuse opere de artă în muzeu;

❖ Vizualizarea listei tuturor artiștilor care au expuse opere de artă în muzeu (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 muzeu;

❖ Vizualizarea listei tuturor operelor de artă expuse în muzeu sortate după artist (vizualizarea include și redarea unor imagini cu operele de artă; între 1 și 3 imagini pentru fiecare operă de artă);

❖ Căutarea unei opere de artă după titlu;

❖ Filtrarea listei operelor de artă după următoarele criterii: artist sau tipul operei de artă;

❖ Salvarea listei cu operele de artă expuse în fișiere de tip csv și doc.

❖ Vizualizarea unor statistici legate de operele de artă din muzeu utilizând grafice.

1. Instrumente utilizate

Limbaj de programare: Java

IDE: IntelliJ IDEA 2024.3.4.1

Bază de date: SQLite

Biblioteci utilizate:

* Java Swing pentru interfața grafică
* JDBC pentru conexiunea la baza de date
* Instrumente pentru modelarea UML: PlantUML
* JFreeChart pentru implementarea graficelor statistice

1. Justificarea limbajului de programare ales

Aplicația a fost dezvoltată utilizând limbajul de programare Java datorită următoarelor avantaje:

* **Portabilitate:** Java este independent de platformă, permițând aplicației să ruleze pe diverse sisteme de operare fără modificări semnificative.
* **Orientare pe obiecte:** Java este un limbaj complet orientat pe obiecte care facilitează implementarea arhitecturii MVC și respectarea principiilor SOLID.
* **Biblioteci grafice robuste:** Java Swing oferă componente grafice versatile pentru dezvoltarea interfețelor utilizator interactive și responsive.
* **Suport pentru internaționalizare:** Java oferă suport nativ pentru internaționalizare (i18n) prin clasa ResourceBundle, facilitând implementarea interfeței în mai multe limbi.
* **Gestionarea eficientă a resurselor:** Mecanismul automat de garbage collection din Java facilitează gestionarea memoriei și previne memory leaks.
* **Conectivitate cu baze de date:** API-ul JDBC oferă instrumente puternice și standardizate pentru conectarea și interacțiunea cu baza de date MySQL.
* **Documentație cuprinzătoare:** Existența unei documentații bogate și a unei comunități active facilitează rezolvarea problemelor întâmpinate în timpul dezvoltării.
* **Compatibilitate cu arhitectura MVC:** Java se pretează foarte bine la implementarea șablonului arhitectural Model-View-Controller și a șablonului comportamental Observer, permițând o separare clară între logica aplicației, prezentarea datelor și interfața cu utilizatorul.

4. Descrierea diagramelor UML

În cadrul proiectării sistemului de gestiune pentru muzee, au fost dezvoltate următoarele diagrame UML pentru a modela diferitele aspecte ale aplicației:

4.1. Diagrama cazurilor de utilizare (Use Case Diagram)

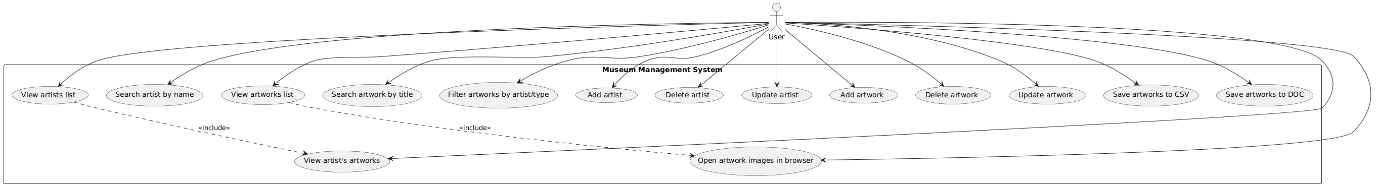


Diagrama cazurilor de utilizare ilustrează interacțiunile dintre utilizator și sistemul de gestiune pentru muzee. Aceasta identifică toate funcționalitățile principale pe care sistemul le oferă utilizatorului:

UC1: Adăugare artist - permite introducerea unui nou artist în sistem

UC2: Ștergere artist - permite eliminarea unui artist existent

UC3: Actualizare artist - permite modificarea datelor unui artist

UC4: Vizualizare listă artiști - afișează toți artiștii din muzeu

UC5: Căutare artist după nume - permite găsirea rapidă a unui artist

UC6: Adăugare operă de artă - permite adăugarea unei noi opere

UC7: Ștergere operă de artă - permite eliminarea unei opere existente

UC8: Actualizare operă de artă - permite modificarea datelor unei opere

UC9: Vizualizare listă opere de artă - afișează toate operele de artă

UC10: Căutare operă după titlu - permite găsirea unei opere după titlu

UC11: Filtrare opere după artist - filtrează lista operelor după artist

UC12: Filtrare opere după tipul operei - filtrează după tipul operei

UC13: Salvare listă opere în fișier CSV - exportă datele în format CSV

UC14: Salvare listă opere în fișier DOC - exportă datele în format DOC

UC15: Vizualizare detalii artist - afișează toate detaliile unui artist

UC16**:** Vizualizare statistici - afișează grafice statistice legate de operele de artă

Diagrama arată de asemenea relații de tip "include" între cazurile de utilizare, evidențiind dependența dintre UC1 și UC14, și dintre UC3 și UC15.

4.2. Diagrama Entitate-Relație (Entity-Relationship Diagram)

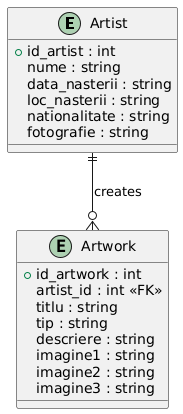


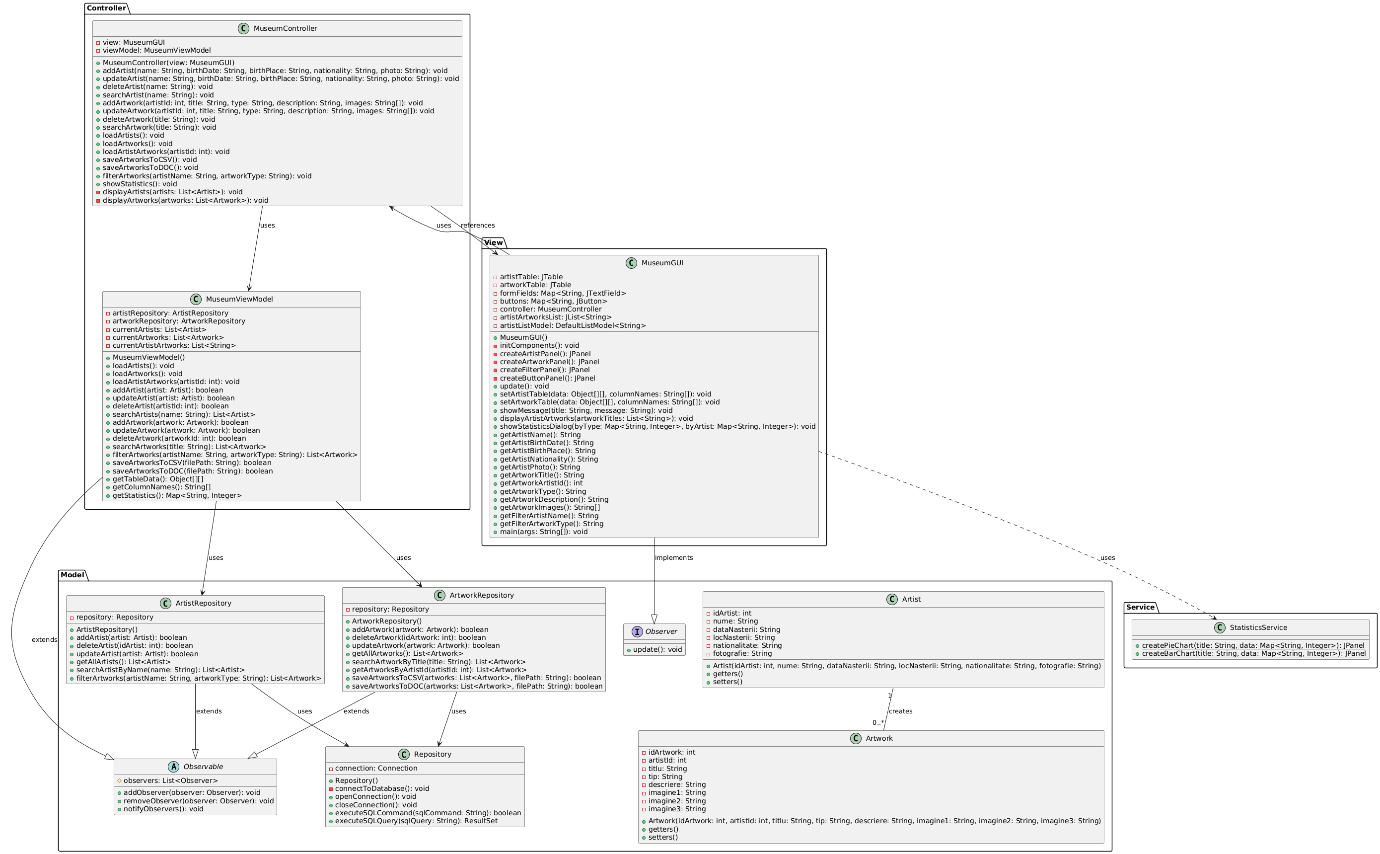
Diagrama Entitate-Relație modelează structura bazei de date a sistemului și ilustrează relația dintre entitățile principale:

Artist: Entitate care stochează informații despre artiști, inclusiv ID-ul unic, numele, data nașterii, locul nașterii, naționalitatea și o referință către fotografia acestuia.

Artwork: Entitate care conține informații despre operele de artă, inclusiv ID-ul unic, titlul, tipul, descrierea, referințe către imagini (până la 3 imagini per operă) și ID-ul artistului care a creat opera.

Relația dintre aceste două entități este de tip one-to-many (1:n), un artist putând crea multiple opere de artă. Această relație este reprezentată prin linia care conectează entitățile și este etichetată cu "creates".

4.3. Diagrama de Clase (Class Diagram)



Această diagramă de clasă reprezintă arhitectura MVC pentru sistemul dumneavoastră de management al muzeului:

Pachet model:

Conține clase de entități (Artist și Artwork)

Include clase de depozit pentru accesul la date (Repository, ArtistRepository, ArtworkRepository)

Implementează modelul Observer cu interfețele Observable și Observer pentru a notifica vizualizările modificărilor datelor

Vezi pachetul:

Conține interfața cu utilizatorul (MuseumGUI) care implementează interfața Observer

Se ocupă de introducerea și afișarea utilizatorului

Pachet controler:

Conține controlerul (MuseumController) care procesează intrarea utilizatorului din vizualizare

Găzduiește ViewModel (MuseumViewModel) care face legătura între Model și View

Se ocupă de logica de afaceri și validarea datelor

Pachet de servicii:

Conține clase de utilitate precum StatisticsService pentru funcționalitatea de vizualizare

MuseumGUI: Implementează interfața grafică folosind Java Swing și comunică cu Presenter-ul

Relațiile între clase reflectă arhitectura MVP și includ:

Relația de asociere între Artist și Artwork (un artist creează mai multe opere)

Relațiile de dependență între repository-uri și clasa de bază Repository

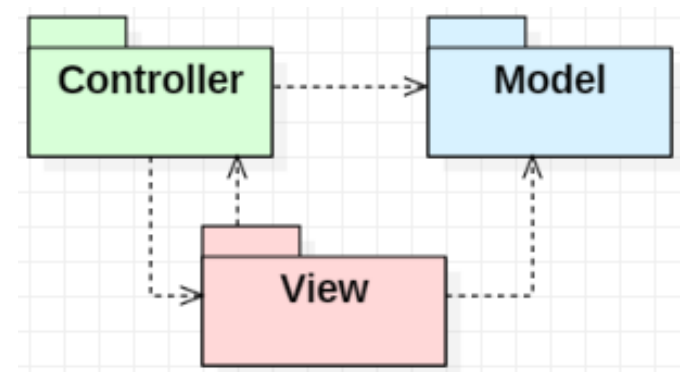
Relațiile de asociere între MuseumPresenter și repository-uri

Relația de implementare între MuseumGUI și interfața IMuseumGUI

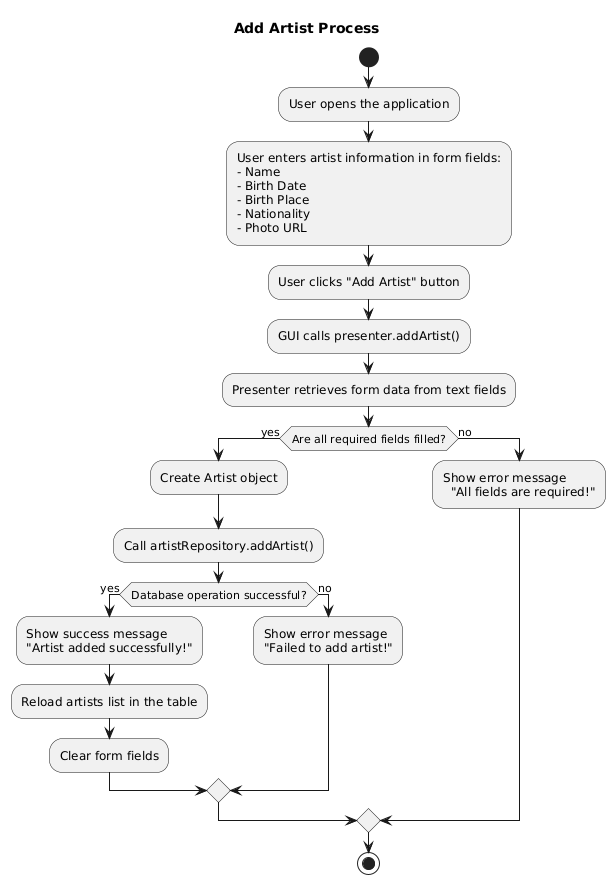
Relația bidirecțională între MuseumPresenter și MuseumGUI

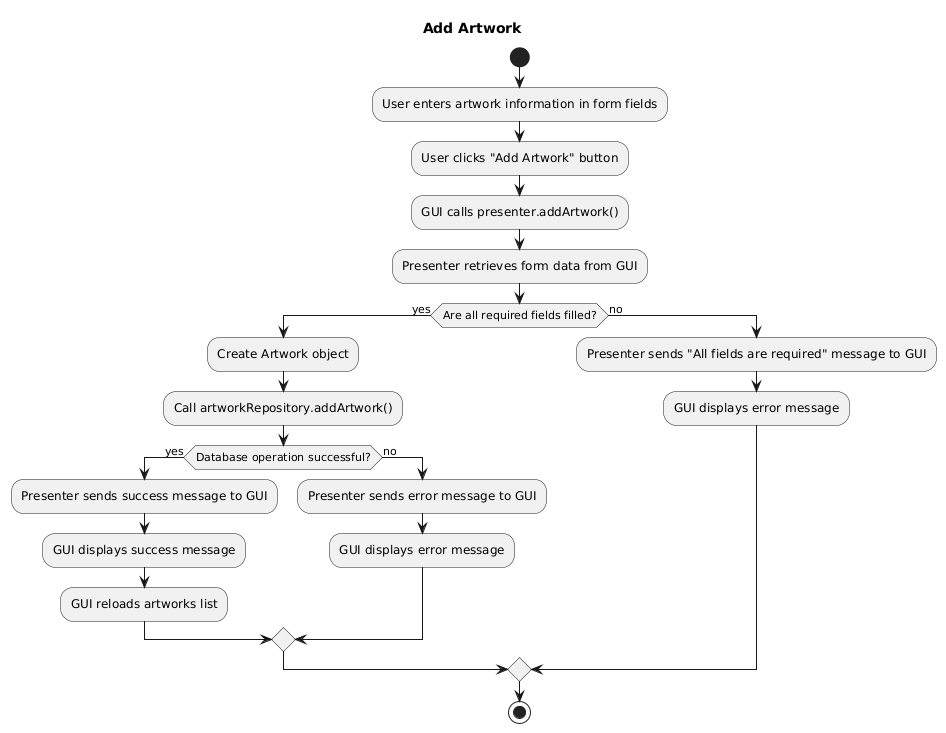
Această organizare pe pachete și relațiile între clase respectă principiile SOLID și faciliteză o separare clară a responsabilităților în cadrul aplicației.

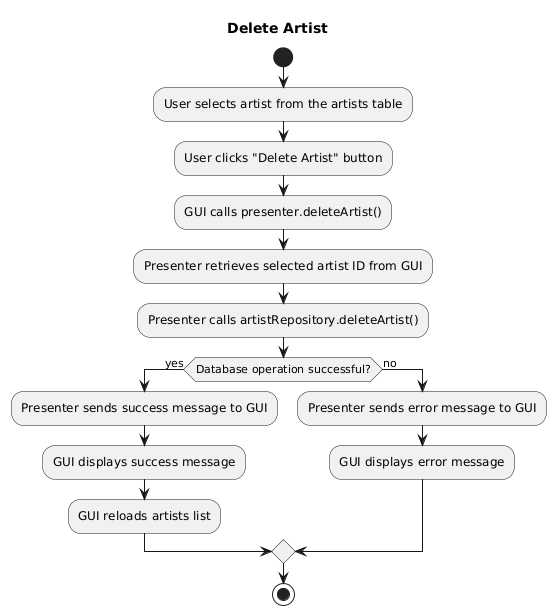
4.4 Diagrama de pachete

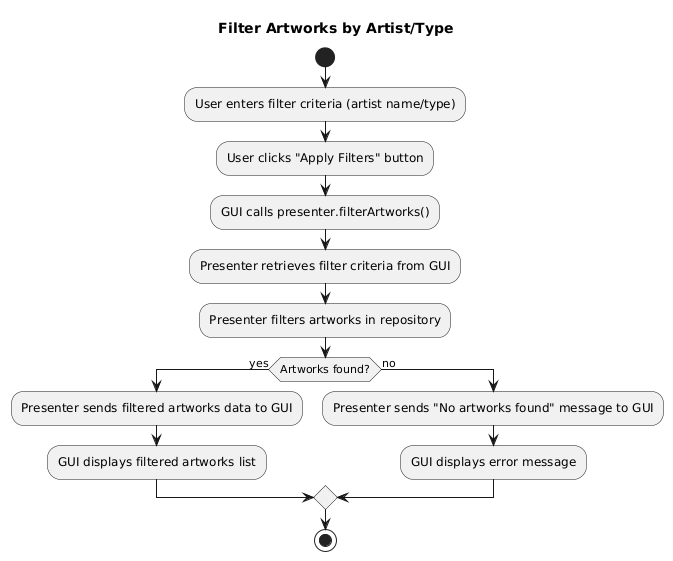
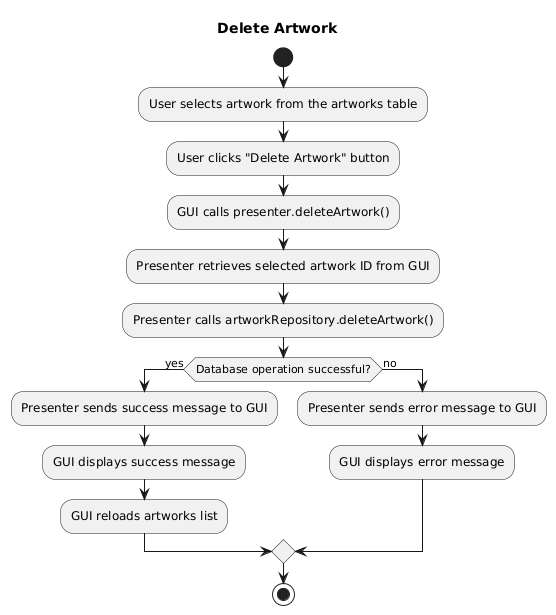


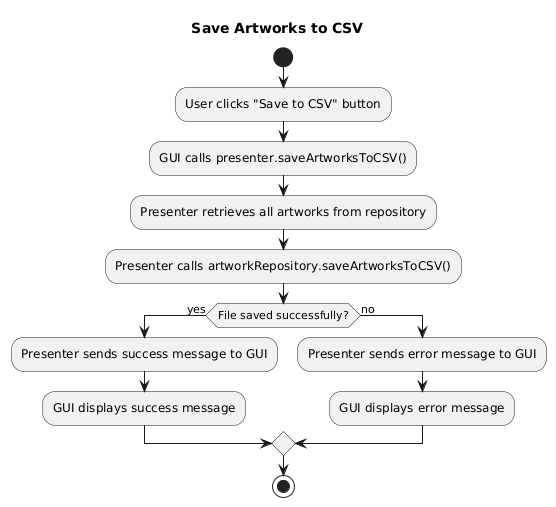
* 1. Diagrame de activitati:

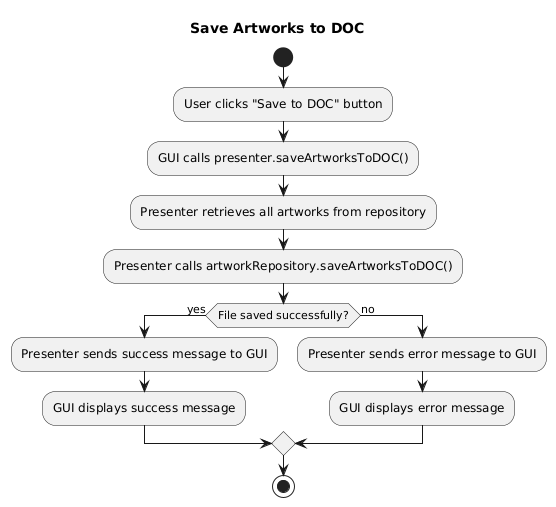


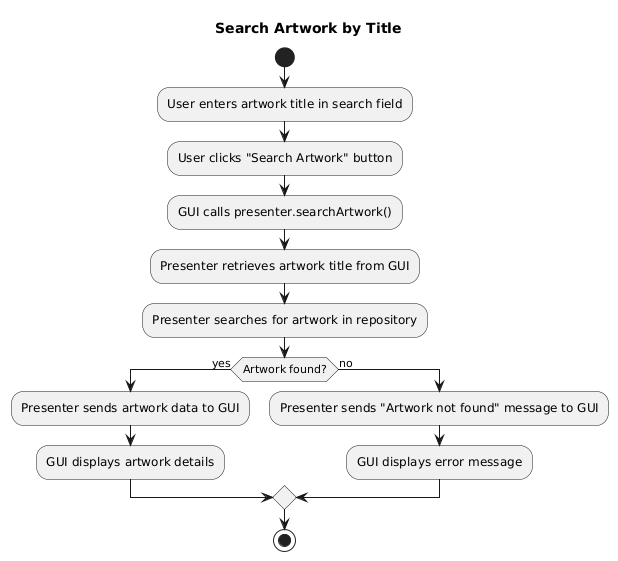


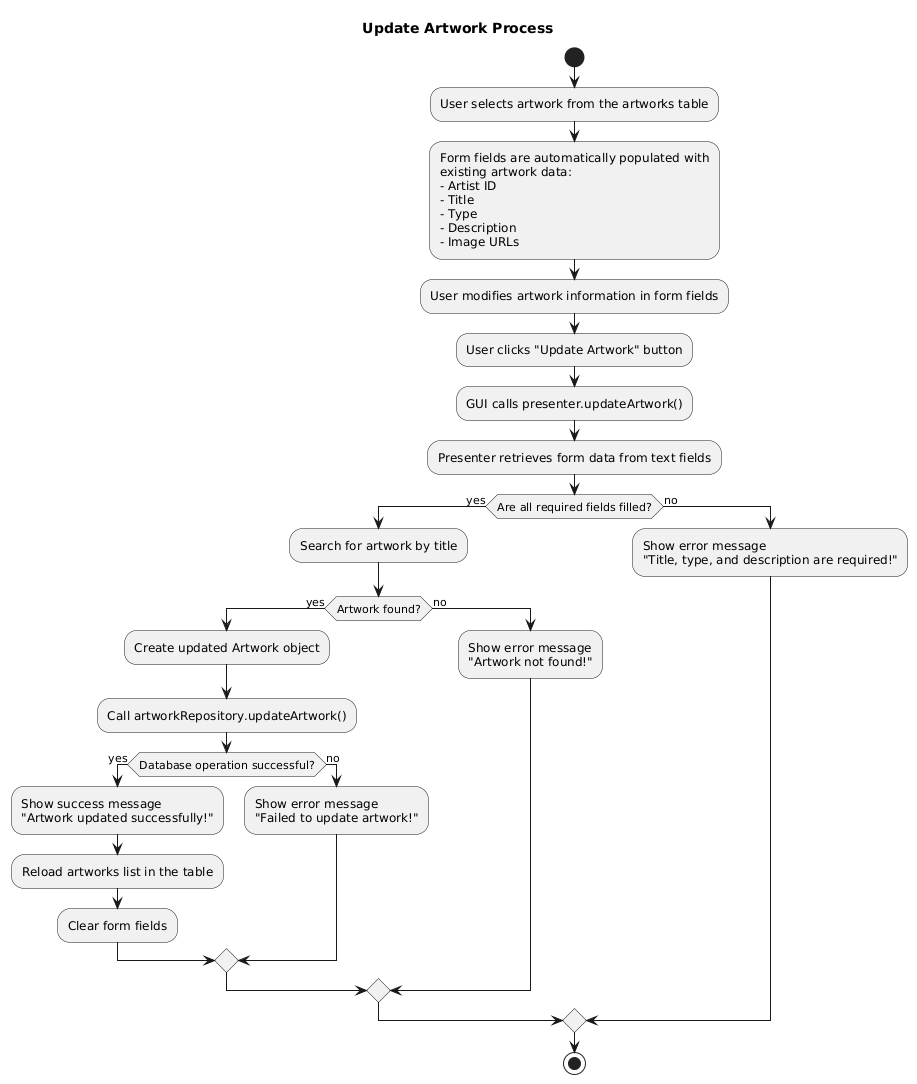
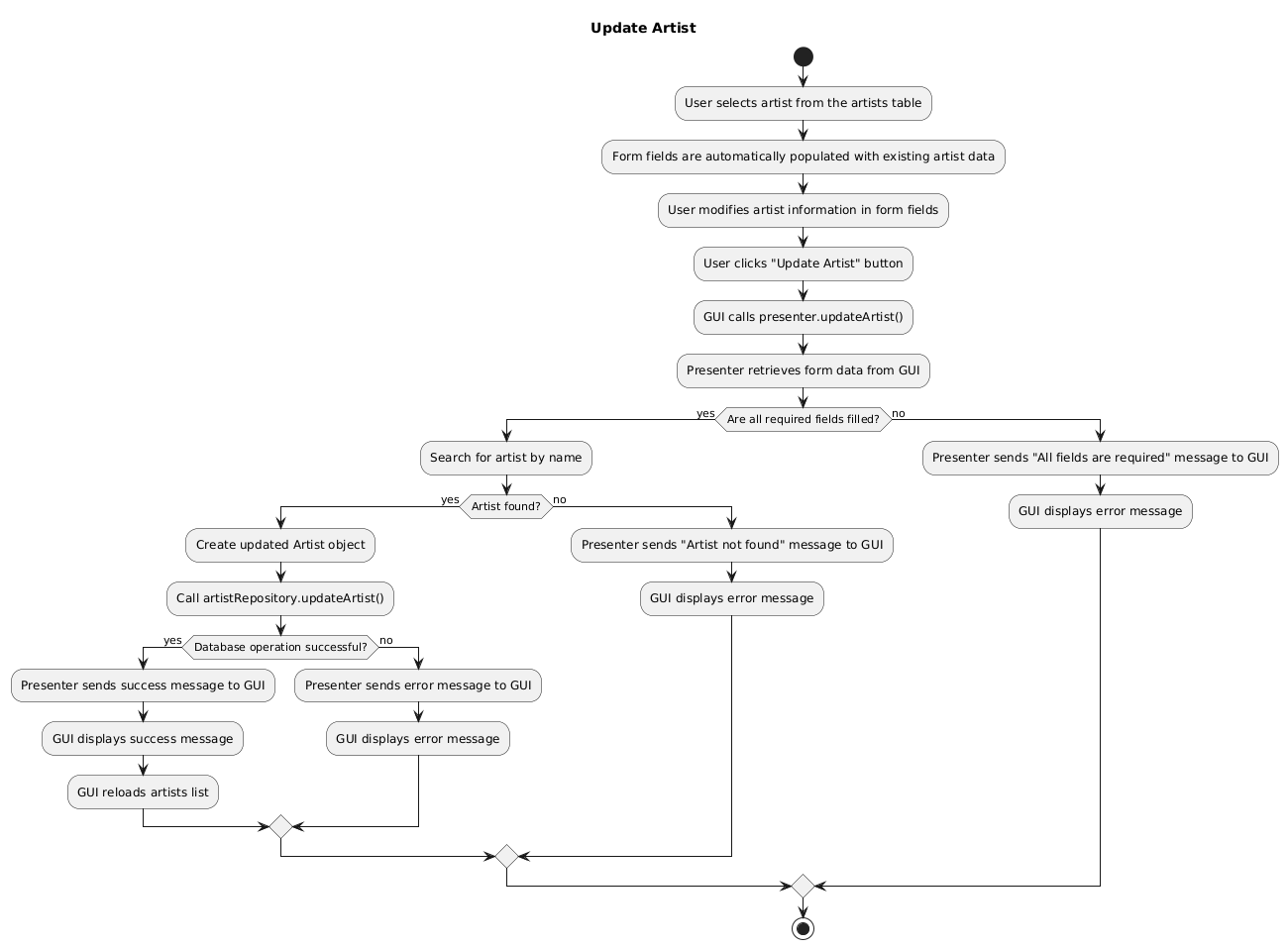


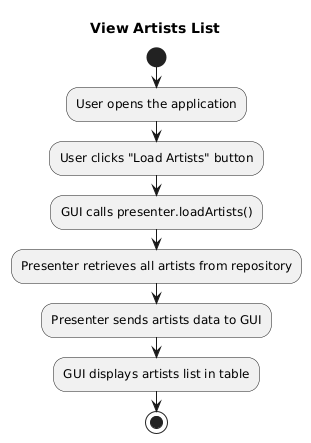


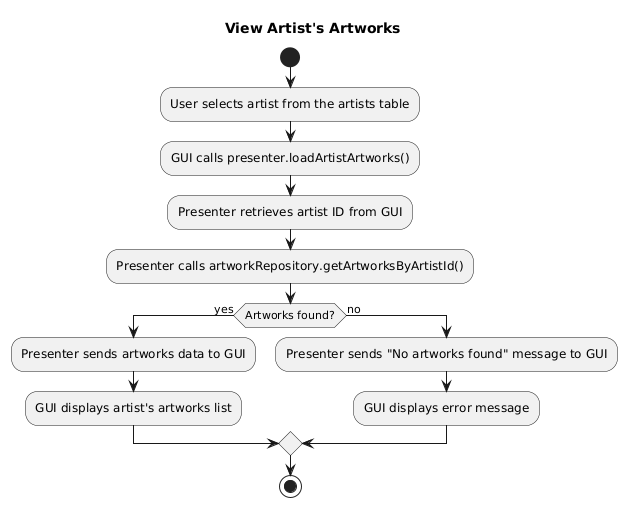


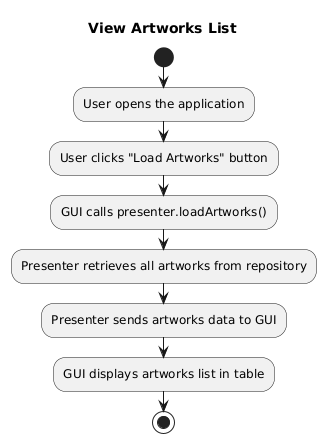




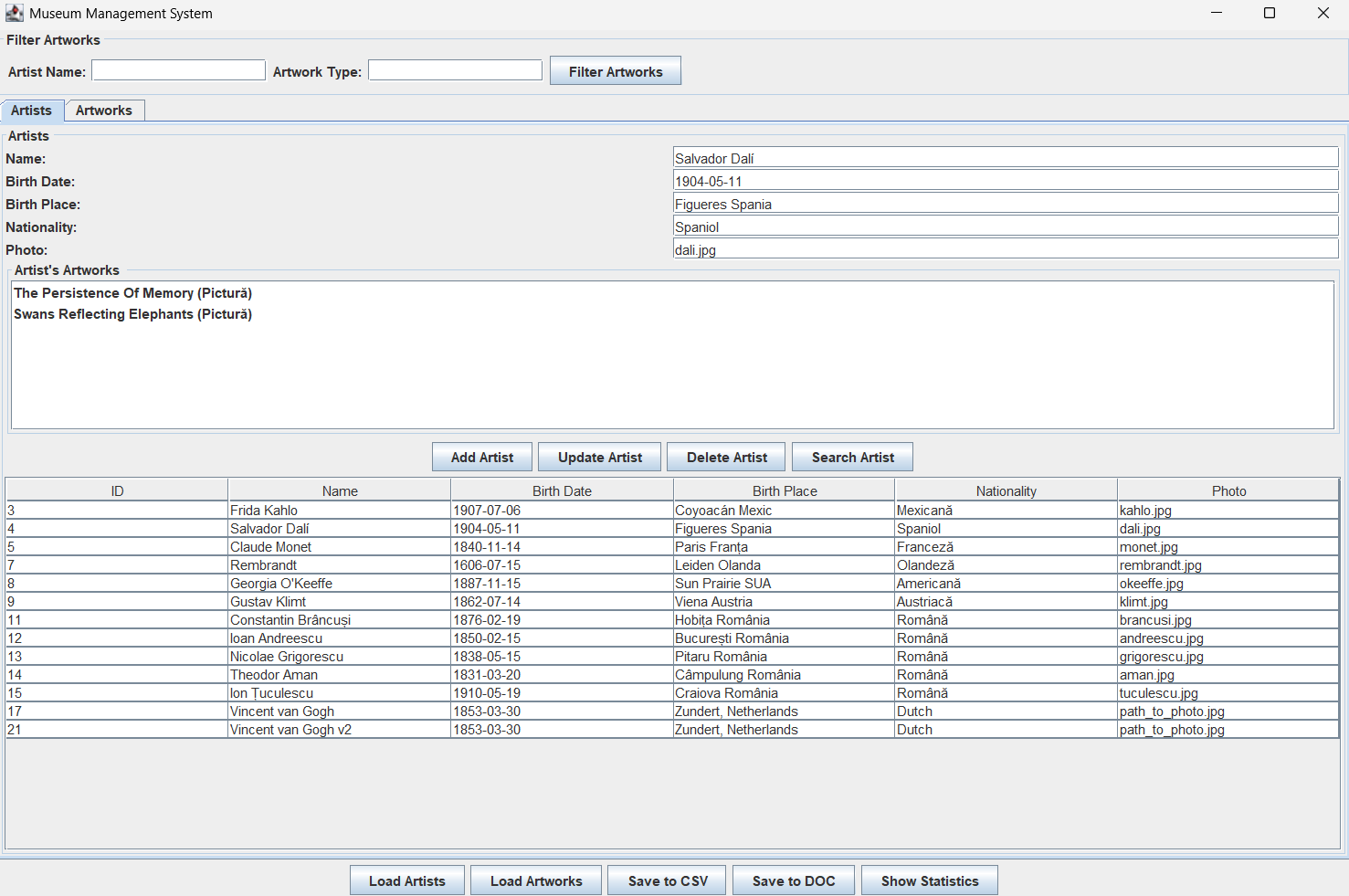








1. Descrierea aplicației



Arhitectura aplicației

Aplicația de gestiune a operelor de artă din muzeu a fost implementată folosind arhitectura Model-View-Controller (MVC), care oferă o separare clară între datele aplicației, logica de business și interfața cu utilizatorul:

Model: Reprezintă structura datelor și logica de acces la baza de date

• Clasele Artist și Artwork definesc entitățile principale ale sistemului

• Clasele Repository (Repository, ArtistRepository, ArtworkRepository) gestionează persistența și interacțiunea cu baza de date SQLite

• Pattern-ul Observer (Observable și Observer) permite notificarea automată a componentelor View când datele sunt modificate

View: Componentele de interfață care permit utilizatorului interacțiunea cu sistemul

• Clasa MuseumGUI implementează interfața grafică folosind Java Swing

• Implementează interfața Observer pentru a primi actualizări când datele se schimbă

• Delegă acțiunile utilizatorului către Controller

Controller: Gestionează logica aplicației și acționează ca intermediar între Model și View

• Clasa MuseumController procesează acțiunile utilizatorului primite de la View

• Implementează validarea datelor și coordonează fluxul de lucru al aplicației

• Folosește MuseumViewModel pentru a transforma datele între Model și View

ViewModel: Componenta mediator dintre Model și View

• Clasa MuseumViewModel extinde Observable și gestionează starea actuală a aplicației

• Expune datele într-un format adecvat pentru afișare și interacțiune în View

• Interacționează cu repository-urile pentru a obține și actualiza datele

Services: Componente utilitare pentru funcționalități specifice

• StatisticsService oferă funcționalități de vizualizare și analiză a datelor

**Funcționalități implementate**

Aplicația implementează toate funcționalitățile cerute în specificații:

Gestiunea artiștilor:

* Adăugare, ștergere și actualizare artiști
* Vizualizarea listei tuturor artiștilor în format tabelar
* Căutare artiști după nume

Gestiunea operelor de artă:

* Adăugare, ștergere și actualizare opere de artă
* Vizualizarea listei tuturor operelor de artă
* Căutare opere de artă după titlu
* Afișarea și vizualizarea imaginilor asociate operelor de artă

Filtrare și sortare:

* Filtrarea operelor de artă după artist sau tipul operei
* Vizualizarea operelor de artă sortate după artist

Exportarea datelor:

* Salvarea listei cu operele de artă în fișier CSV
* Salvarea listei cu operele de artă în fișier DOC

Vizualizare detalii:

* Pentru fiecare artist se pot vizualiza toate operele de artă asociate
* Pentru fiecare operă de artă se pot vizualiza imaginile asociate direct în browser
* Structura bazei de date

Baza de date SQLite conține două tabele principale:

**Tabelul Artist:**

id\_artist (INTEGER, PRIMARY KEY)

nume (TEXT)

data\_nasterii (TEXT)

loc\_nasterii (TEXT)

nationalitate (TEXT)

fotografie (TEXT)

**Tabelul Artwork:**

id\_artwork (INTEGER, PRIMARY KEY)

artist\_id (INTEGER, FOREIGN KEY)

titlu (TEXT)

tip (TEXT)

descriere (TEXT)

imagine1 (TEXT)

imagine2 (TEXT)

imagine3 (TEXT)

Relația dintre cele două tabele este de tip one-to-many, un artist putând avea mai multe opere de artă expuse în muzeu.

Interfața utilizator

Interfața grafică este împărțită în mai multe secțiuni:

Secțiunea pentru gestionarea artiștilor:

* Formular pentru introducerea/editarea datelor despre artiști
* Tabel pentru afișarea listei de artiști
* Butoane pentru adăugare, actualizare, ștergere și căutare artiști
* Vizualizarea operelor create de un artist selectat

Secțiunea pentru gestionarea operelor de artă:

* Formular pentru introducerea/editarea datelor despre operele de artă
* Tabel pentru afișarea listei de opere de artă
* Butoane pentru adăugare, actualizare, ștergere și căutare opere de artă
* Posibilitatea de a vizualiza imaginile asociate operelor de artă

Secțiunea de filtrare:

* Câmpuri pentru filtrarea operelor de artă după artist sau tip

Secțiunea de gestionare a datelor:

* Butoane pentru încărcarea artiștilor și operelor de artă din baza de date
* Butoane pentru salvarea listei de opere de artă în formate CSV și DOC

Caracteristici suplimentare

Validarea datelor - aplicația implementează verificări pentru a evita introducerea de date invalide

Gestionarea erorilor - mesaje de eroare specifice sunt afișate utilizatorului în caz de operațiuni nereușite

Interactivitate - selectarea unui rând din tabelul de artiști populează automat formularul și afișează lista operelor de artă create de artistul respectiv

Vizualizare imagini - aplicația oferă posibilitatea deschiderii imaginilor operelor de artă direct în browser-ul web