A picture containing logo

Description automatically generated

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**CATEDRA CALCULATOARE**

Aplicatie lant de galerii de arta  
Client - Server

Documentatie

NUME STUDENT: Turcu Cezar Simeon

GRUPA: 30235

# CUPRINS

[1. Enuntul problemei 3](#_Toc99311586)

[2. Instrumente utilizate 3](#_Toc99311587)

[3. Justificare limbaj de programare ales 4](#_Toc99311588)

[4. Descrierea diagramelor UML 4](#_Toc99311589)

[5. Descrierea aplicatiei 7](#_Toc99311590)

1. Enuntul problemei

Dezvoltați o aplicațieclient/servercare poate fi utilizată într-un lanț de galerii de artă. Aplicația va avea 3 tipuri de utilizatori: vizitator al galeriei de artă, angajat al lanțului de galerii de artă și administrator.

Utilizatorii de tip vizitator pot efectua următoarele operații fără autentificare:  
❖Vizualizarea listei tuturor operelor de artă expuse în galeriile de artă sortată după anul realizării;  
❖Filtrarea listei operelor de artă plastică după următoarele criterii: artist, tipul operei de artă;   
❖Căutarea unei opere de artă după titlu.

Utilizatorii de tip angajat al unei galerii 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;  
❖Vânzarea unei opere de artă din galeria de artă la care lucrează acel angajat;   
❖Salvare liste cu situația operelor de artă în mai multe formate: csv, json, xml, txt.  
❖Vizualizarea unor statistici legate de operele de artă utilizând grafice (structură radială, structură inelară, de tip coloană, etc.).

Utilizatorii de tip administratorpot efectua următoarele operații după autentificare:   
❖Toate operațiile permise utilizatorilor de tip vizitator;  
❖Operații CRUD pentru informațiile legate de utilizatorii care necesită autentificare;  
❖Vizualizarea listei utilizatorilor care necesită autentificareși filtrarea acesteia după tipul utilizatorilor;   
❖Notificarea fiecărui utilizator care necesită autentificare prin cel puțin 2 variante (email, SMS, WhatsApp, Skype, etc.) la orice modificare a informațiilor de autentificare aferente acelui utilizator. Interfața grafică a aplicației client va fi disponibilă în cel puțin 3 limbi de circulație internațională.

2. Instrumente utilizate

**Limbajul de programare** - Java - Limbaj de programare de nivel inalt, orientat pe obiecte, potrivindu-se astfel cerintelor problemei.

**Biblioteci** – Java Swing –Biblioteca grafica pentru limbajul de programare Java, care ofera un set de componente pentru crearea de interfete grafice.  
-JFreeChart –Bazata pe Java Swing fiind folosita la creearea de grafice  
-FasterXml – Faciliteaza lucrul cu fisiere de tip xml si adauga functionalitati pentru maparea obiectelor in Java  
-Twilio – Biblioteca pentru trimiterea de mesaje SMS

**IDE** - IntelliJ IDEA Ultimate - Mediu integrat de dezvoltare (IDE) pentru limbajul de programare Java, ce ofera instrumente pentru depanare si testare, ajuta utilizatorul in scrierea codului si include si tool-uri pentru generarea automata a diagramelor UML.

**Baza de date** - PostgreSQL

**JDBC (Java Database Connectivity)** - Este o interfata standard pentru conectarea la bazele de date relaționale din limbajul de programare Java.

**Maven** - Este un sistem de gestionare a dependențelor pentru proiecte Java, care faciliteaza integrarea cu alte proiecte și biblioteci Java si a fost folosit pentru gestionarea dependintelor pentru PostgreSQL si a testarii cu JUnit.

**Version Control** - Git – Ajuta la gestionarea modificarilor si permite dezvoltarea unor feature-uri diferite in paralel.

3. Justificare limbaj de programare ales

Am ales ca limbaj de programare Java datorita caracteristicilor acestora de limbaj de programare orientat pe obiecte, permitand astfel modelarea entitatilor constituente pentru dezvoltarea aplicatiei.  
 De asemenea, Java are suport pentru un numar foarte mare de librarii ce usureaza treaba programatorului in dezvoltarea aplicatiei.  
 Un alt motiv pentru care am ales sa folosesc Java este suportul pe care acest limbaj de programare il ofera in dezvoltarea aplicatiilor cu o arhitectura layered, asa cum este si arhitectura MVC.

4. Descrierea diagramelor UML

4.1. Diagrama use case

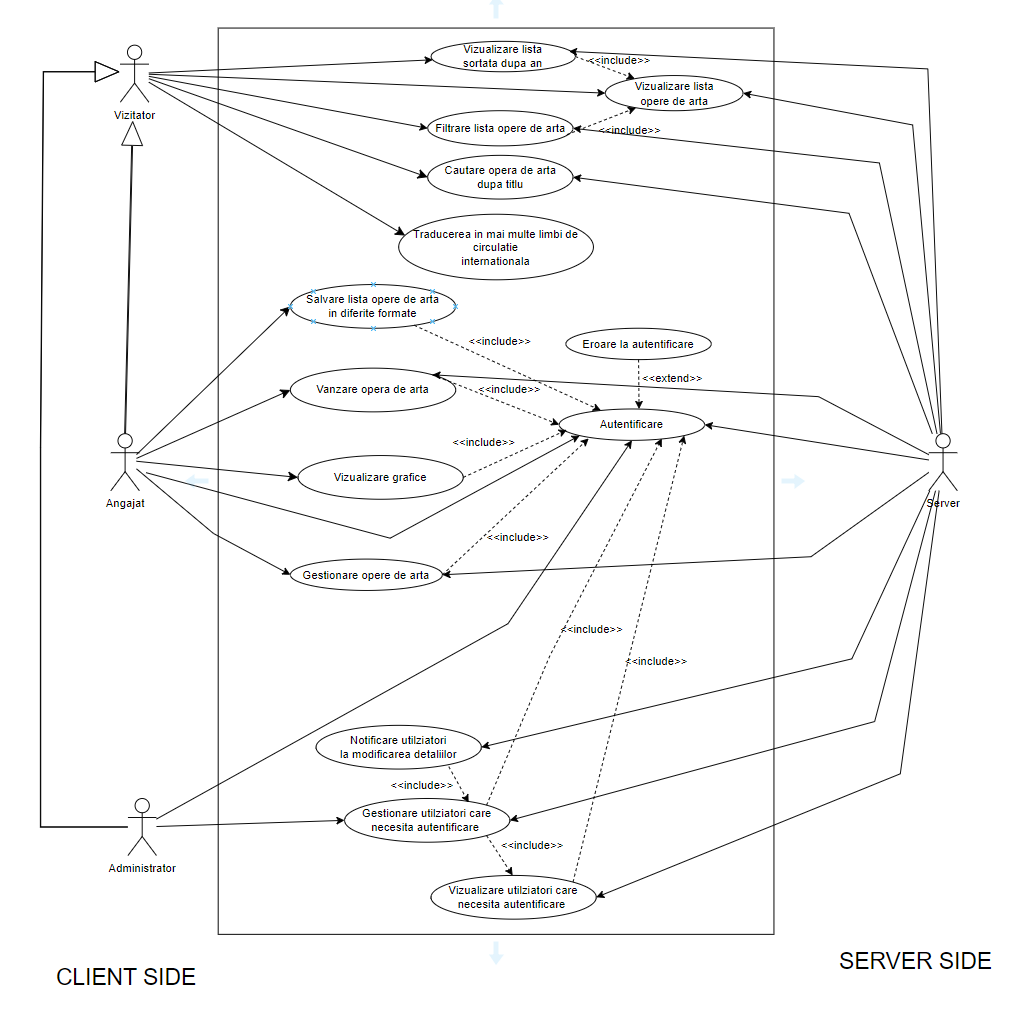


Figure 1 -Diagrama Use Case

Utilizatorii de tip Vizitator nu necesita autentificate si pot vizualiza lista operelor de arta, sa filtreze aceasta lista dupa numele artistului si tipul operei de arta, sa caute o opera de arta dupa titlu si sa sorteze operele de arta dupa anul realizarii acestora. De asemenea din pagina utilizatorilor de tip Vizitator este selectata limba de circulatie internationala care va fi aplicata tuturor paginilor.

Utilizatorii de tip Angajat necesita autentificare, care poate genera o eroare in cazul in care user-ul nu exista in baza de date, daca acesta nu a fost enabled de un administrator sau daca acesta a gresit parola. De asemenea, angajatii pot realiza toate operatiile pe care le realizeaza vizitatorii, operatii CRUD asupra operelor de arta dupa autentificare, vanzarea unei opere de arta si salvarea listei cu operele de arta in mai multe formate. Tot prin intermediul interfetei Angajat se pot genera grafice in formate diferite ce prezinta statistici legate de operele de arta.

Utilizatorii de tip administrator pot, asemenea angajatilor sa se autentifice, pot realiza aceleasii operatii ca si vizitatorii si pot realiza operatii CRUD asupra utilzatorilor care necesita autentificare. Cand datele de autentificare ale unui utilizator vor fi modificate acesta va fi notificat prin intermediul unui SMS si al unui email.

4.2. Diagrama entity relationship

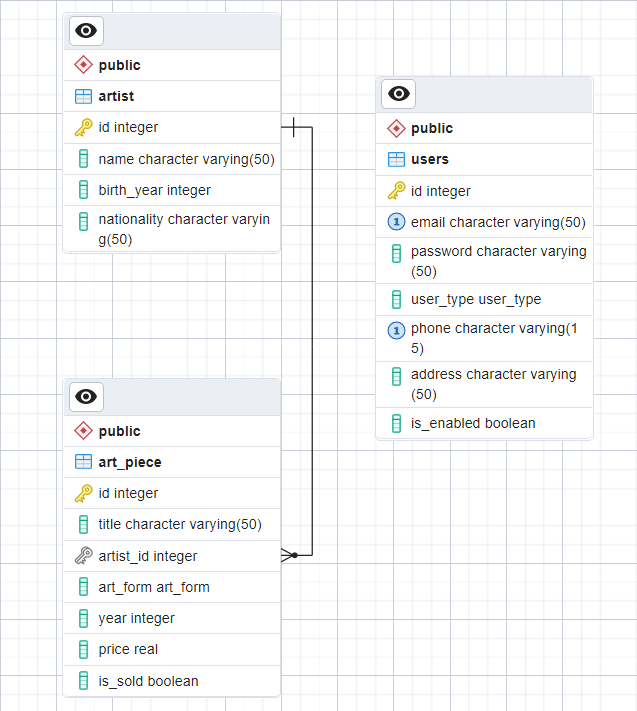


Figure 2 Diagrama Entity Relationship

Tabela Users modeleaza utilizatorii aplicatiei si contine datele necesare pentru identificarea unica a acestora, avand un id ca si cheie primara, si field-uri pentru email(unic), parola, tipul de utilzator, numarul de telefon, adresa si statusul de enabled a acestuia.

Tabela Artist contine un id unic, numele artistului, anul nasterii si nationalitatea acestuia.   
 Tavela Art\_piece contine field-uri pentru id, titlul operei de arta, un foreign key cu id-ul artistului ce referentiaza field-ul id din tabela artist, cat si field-uri pentru identificarea tipului de arta, a anului realizarii, a pretului si a informatiei legata de daca aceasta a fost vandut

4.3. Diagrame de activitati

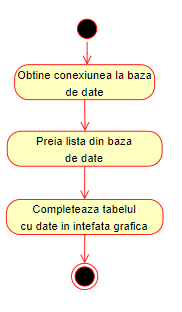


Figure 3 Diagrama vizualizare lista opere de arta

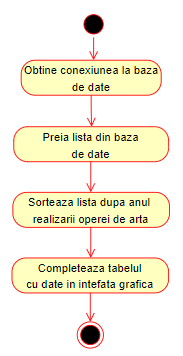


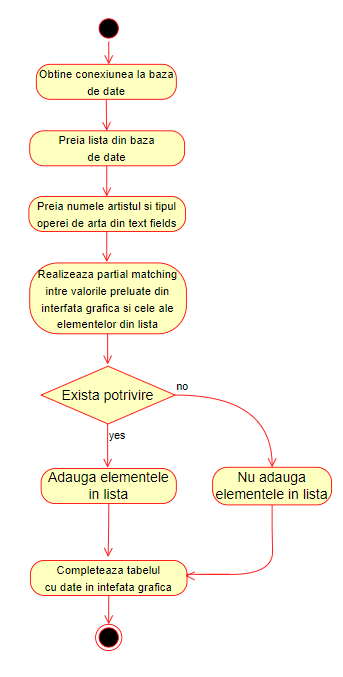
Figure 4 Diagrama sortare lista opere de arta dupa anul realizarii  


Figure 5 Diagrama filtrare lista opere de arta dupa nume artist si tipul operei de arta

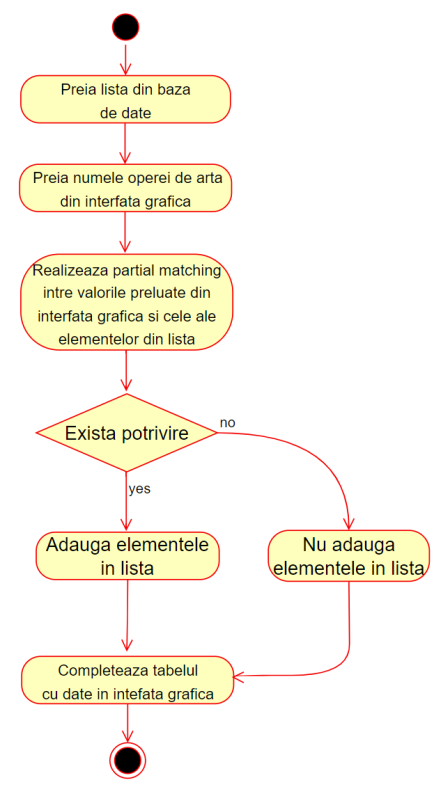


Figure 6 Diagrama cautare opere de arta dupa numele operei de arta

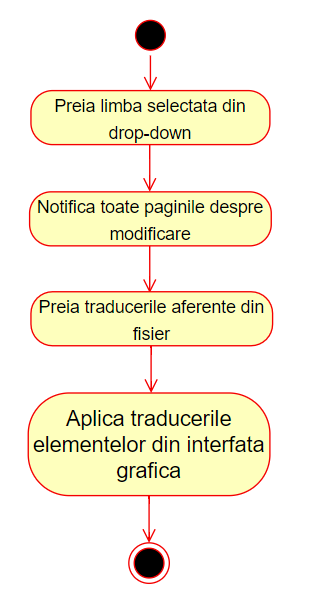


Figure 7 Diagrama traducere interafata grafica

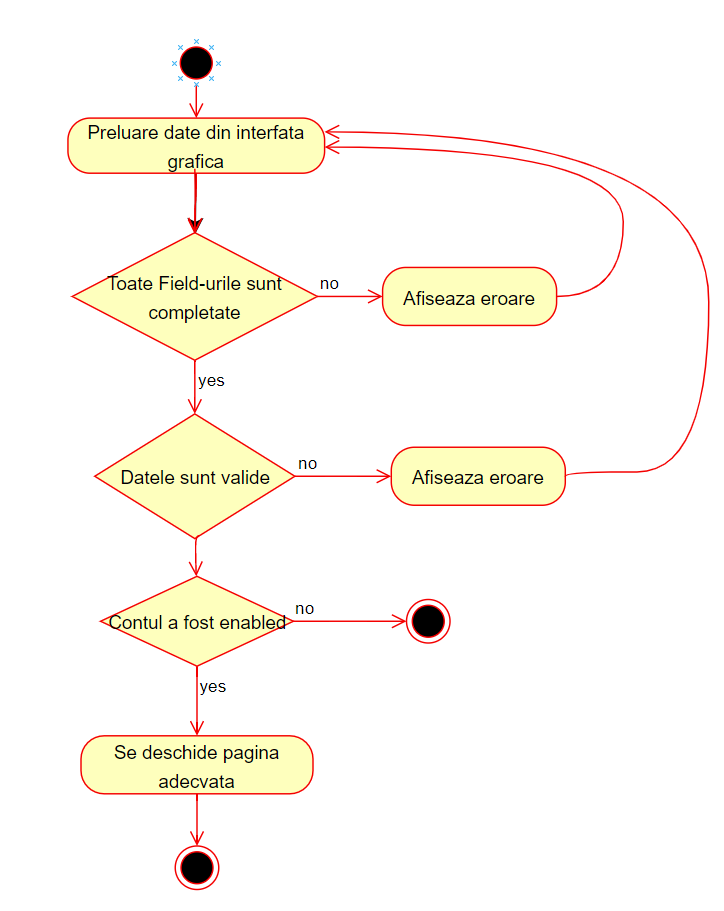


Figure 8 Diagrama Log In

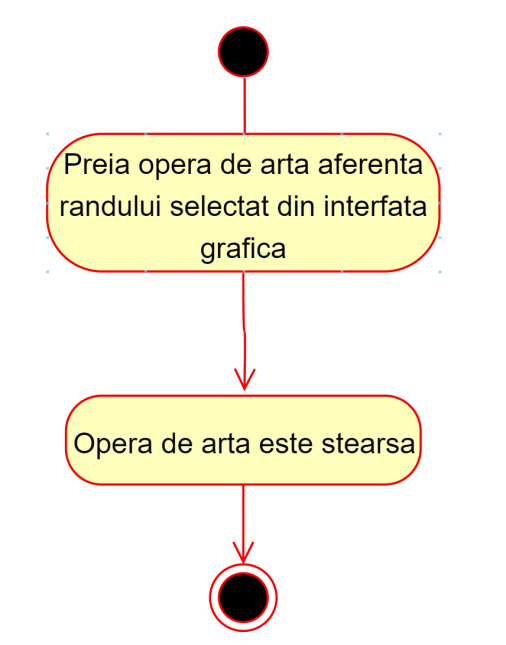


Figure 9 Diagrama stergere opera de arta

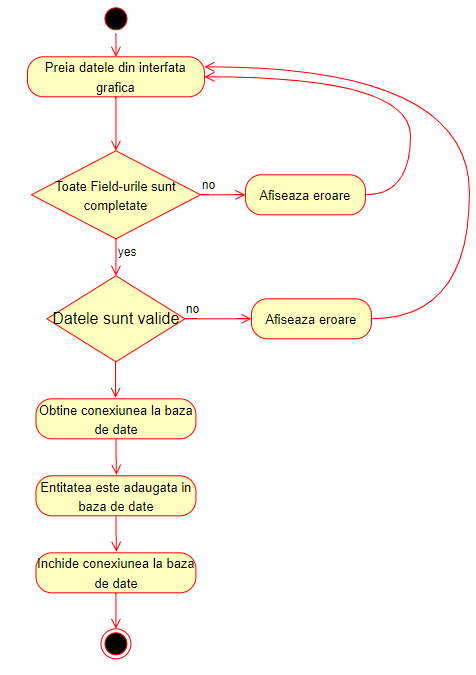


Figure 10 Diagrama adaugare opera de arta

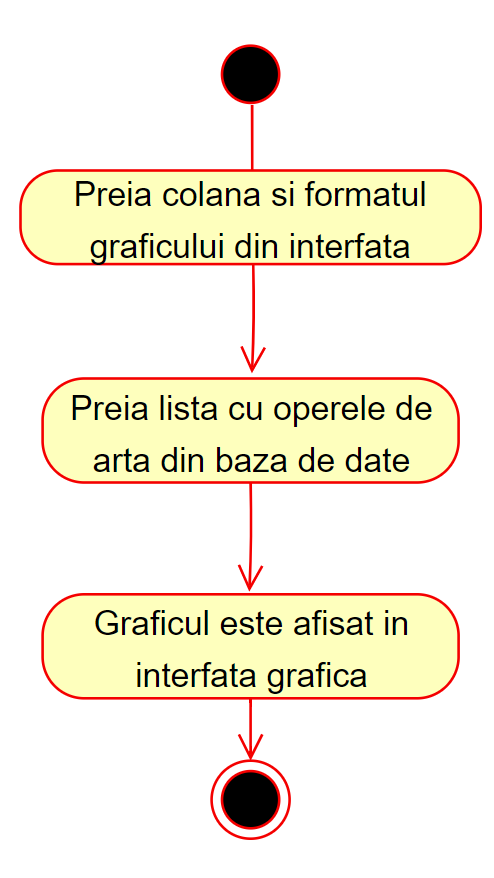


Figure 11 Diagrama vizualizare chart-uri

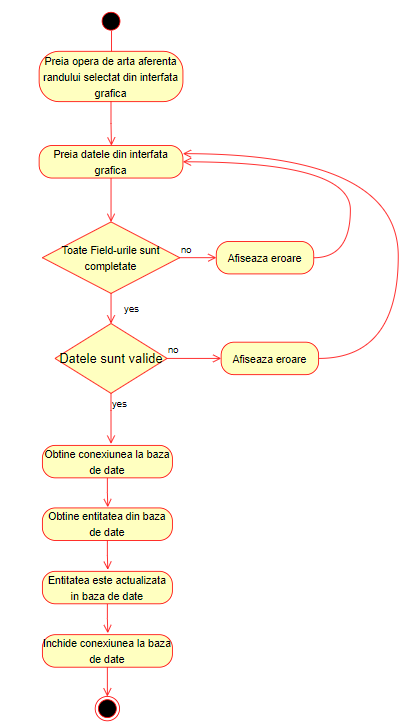


Figure 12 Diagrama actualizare opera de arta

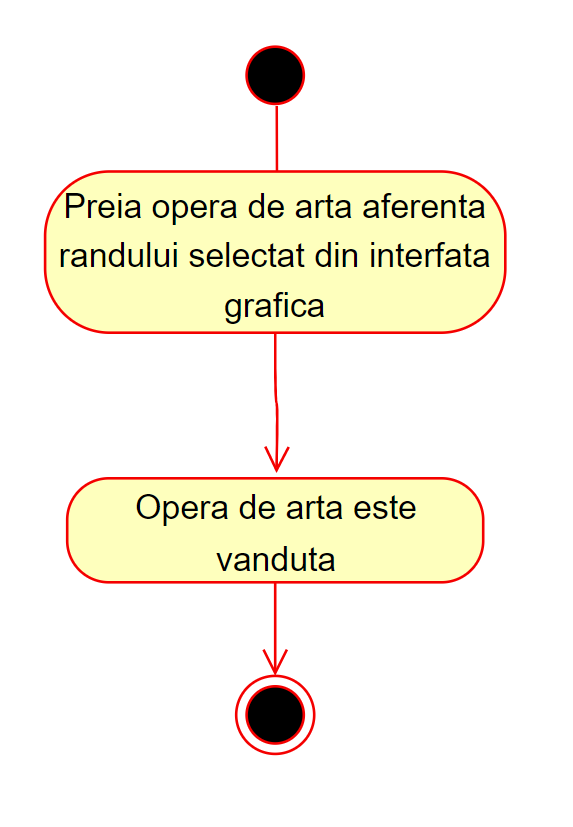


Figure 13 Diagrama vanzare opera de arta

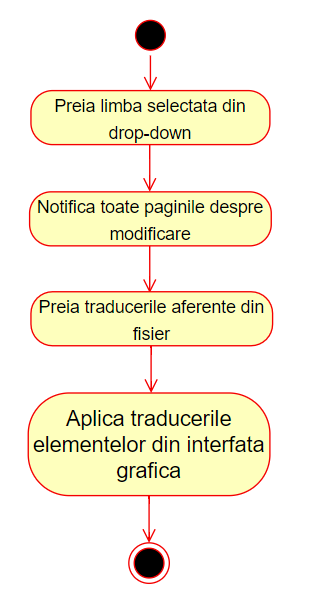


Figure 14 Diagrama exportare situatie operele de arta in diferite formate

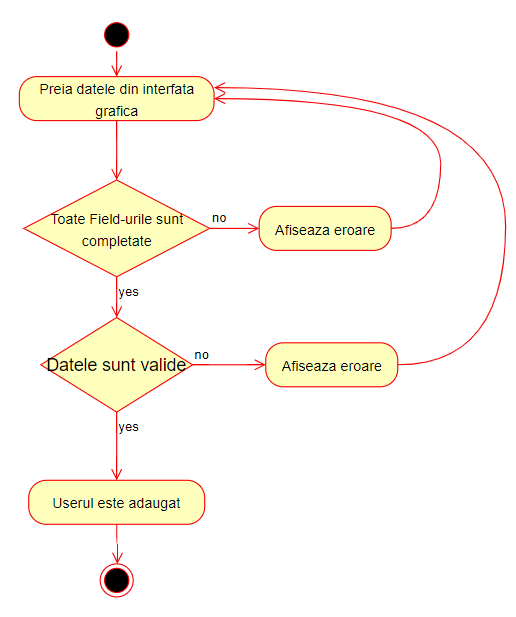


Figure 15 Diagrama adaugare user

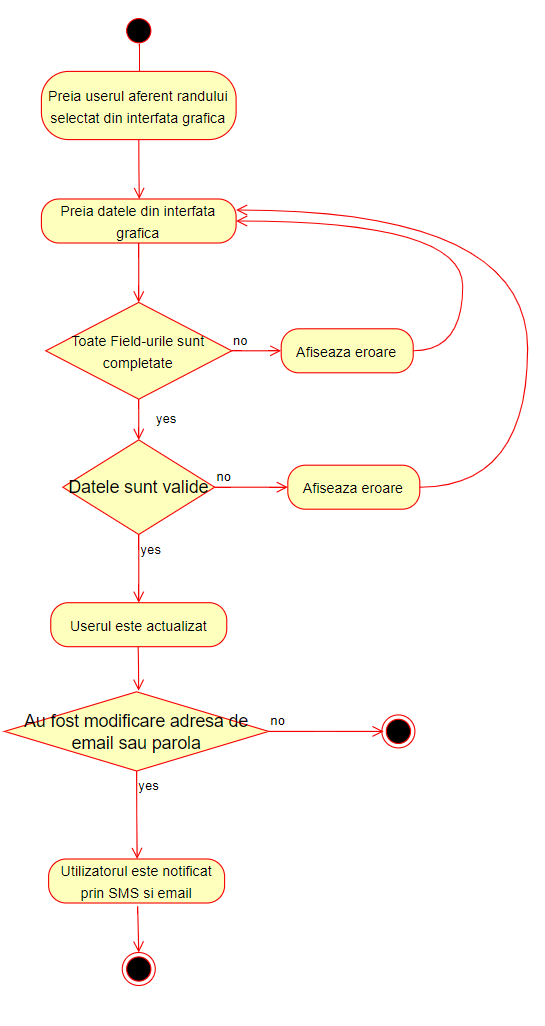


Figure 16 Diagrama actualizare user

4.4. Diagrame de pachete

4.4.1 Client

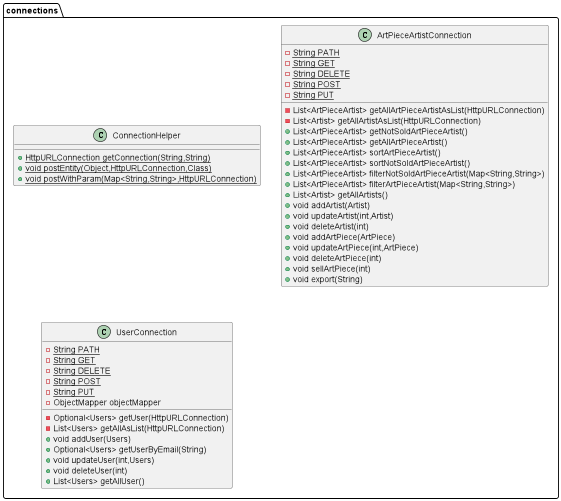


Figure 17 Diagrama pachetului connections

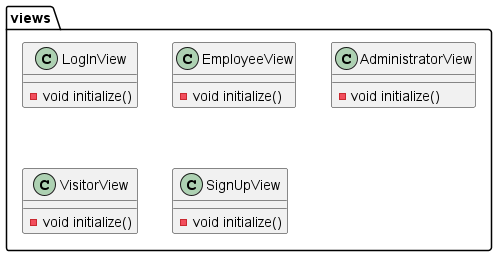


Figure 18 Diagrama pachetului views

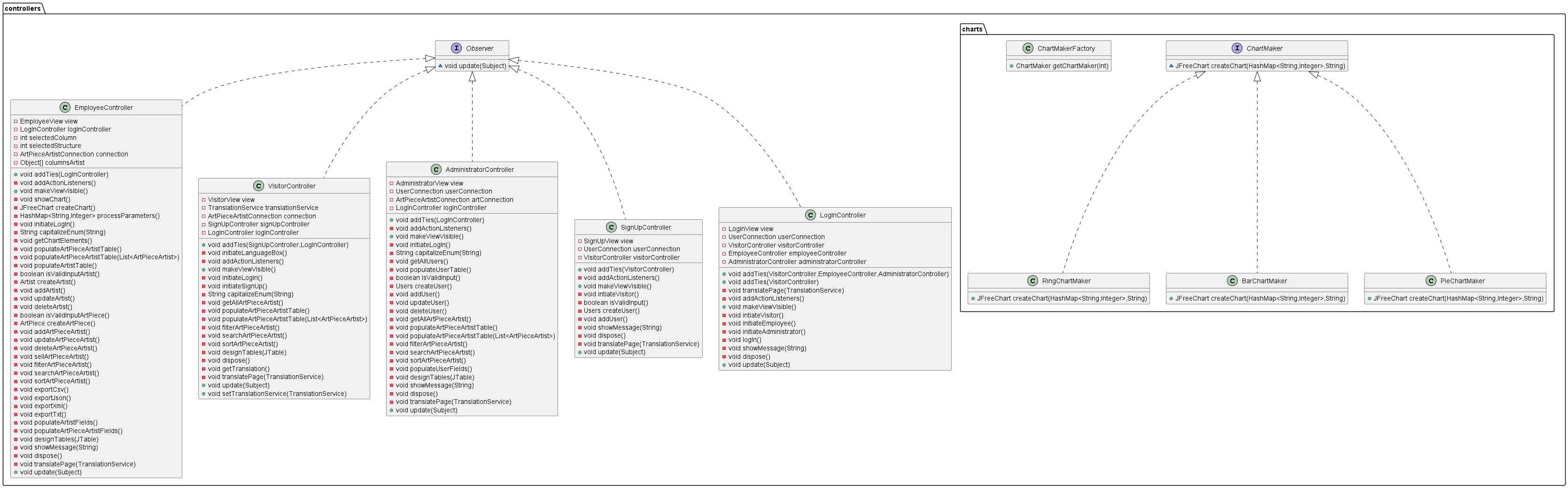


Figure 19 Diagrama pachetului controllers

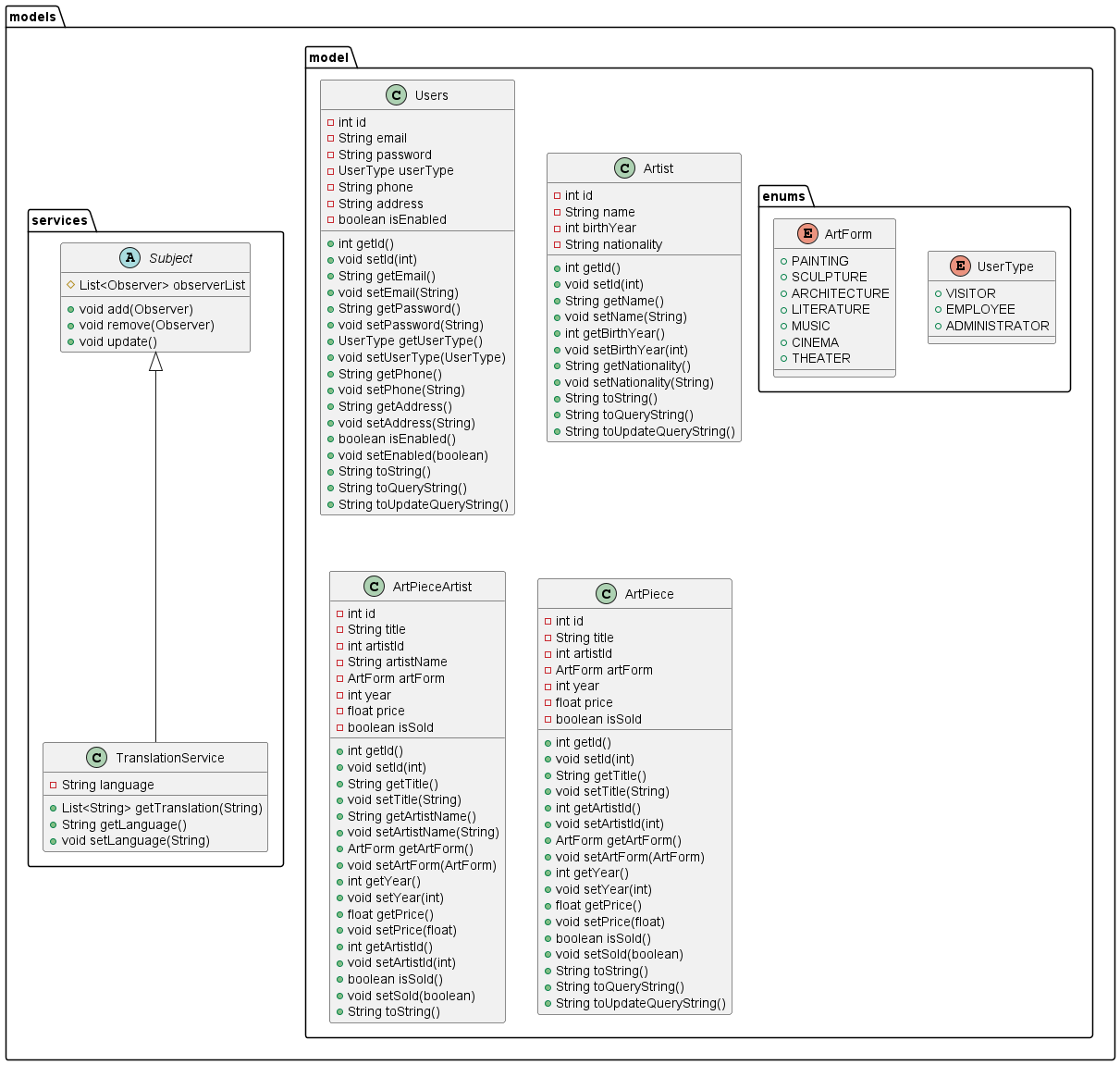


Figure 20 Diagrama pachetului models

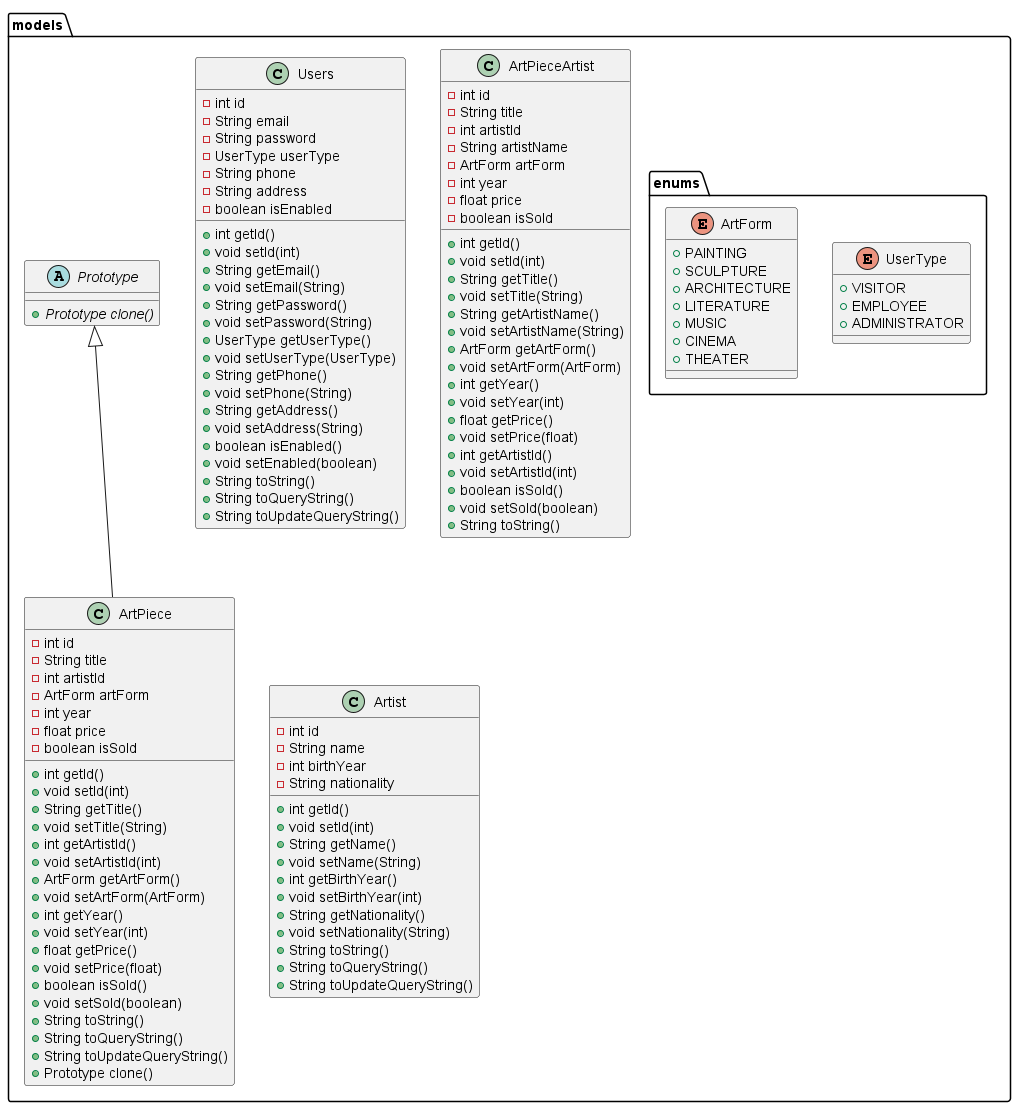
4.4.2 Server

Figure 21 Diagrama pachetului models

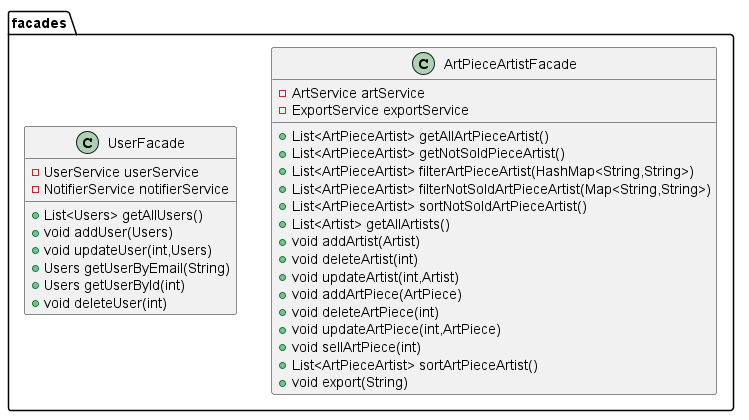


Figure 22 Diagrama pachetului facades

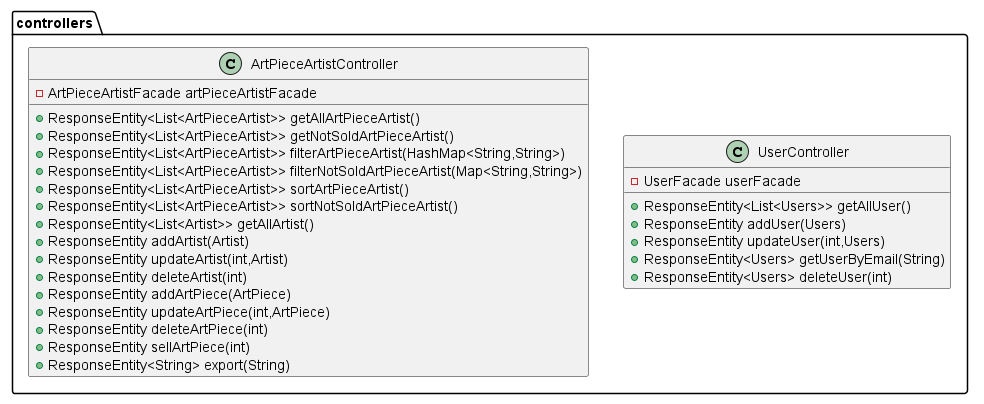


Figure 23 Diagrama pachetului controllers

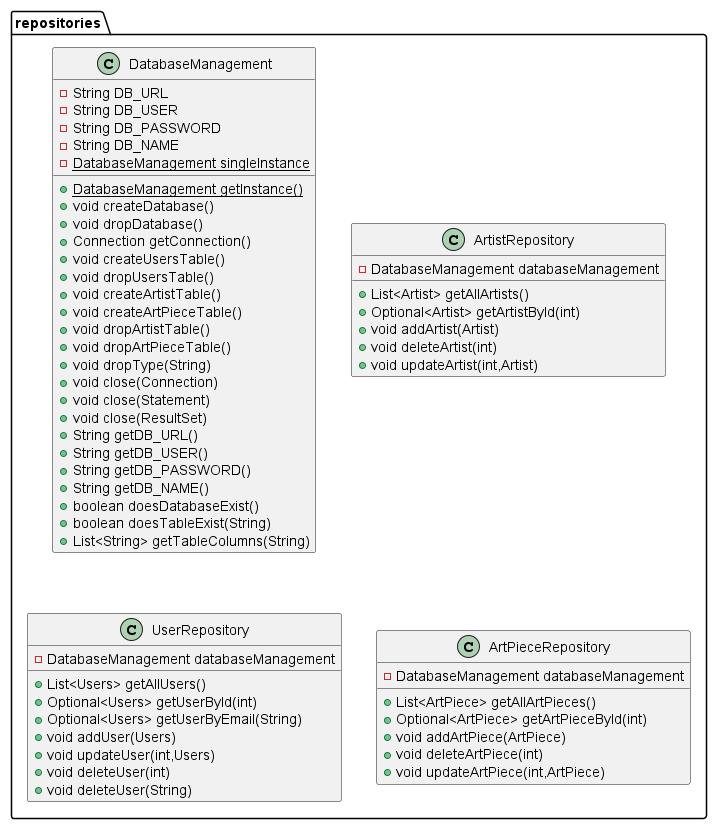


Figure 24 Diagrama pachetului repositories

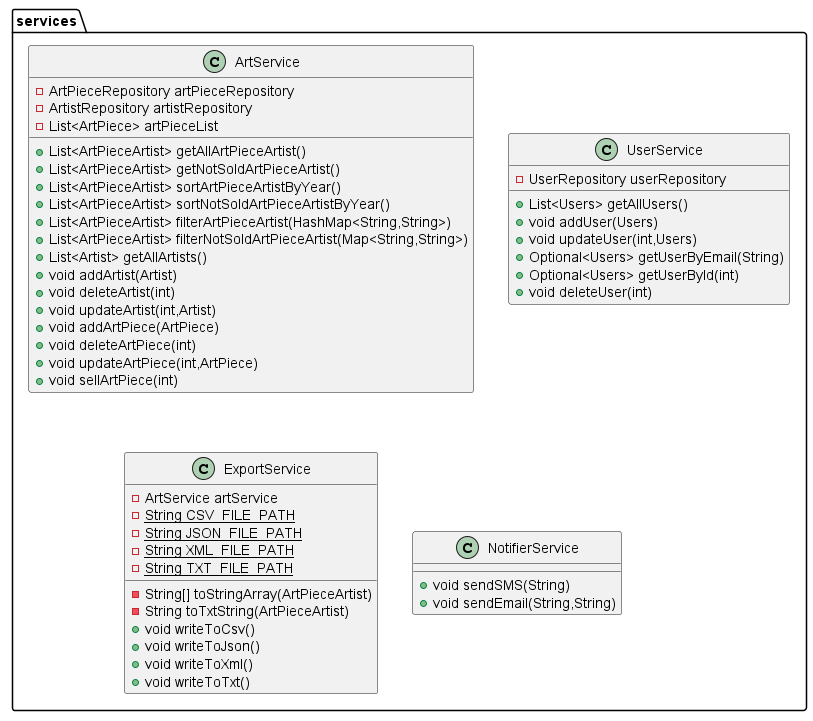


Figure 25 Diagrama pachetului services

Arhitectura client-server reprezinta un model de organizare a sistemelor informatice in care functionalitatea este distribuita intre doua componente principale: clientul si serverul. Clientul este aplicatia sau dispozitivul utilizatorului final, care solicita si primeste servicii sau resurse de la server. Serverul, pe de alta parte, este responsabil de gestionarea cererilor primite de la clienti si de furnizarea de raspunsuri sau de resurse solicitate. Comunicarea intre client si server se realizeaza prin intermediul unei retele, cum ar fi internetul. Aceasta arhitectura ofera numeroase avantaje, inclusiv scalabilitate, securitate si usurinta in dezvoltarea aplicatiilor distribuite. Prin separarea responsabilitatilor intre client si server, arhitectura client-server permite dezvoltarea aplicatiilor complexe si eficiente care pot fi accesate de la distanta, oferind utilizatorilor o experienta interactiva si personalizata

Pentru a implementa arhitectura client-server cu Spring Boot, am utilizat Spring MVC (Model-View-Controller) pentru a gestiona cererile primite de la client și pentru a returna răspunsuri adecvate. Astfel, am putut defini endpoint-uri (rute) pentru diferitele cereri HTTP și le-am asociat cu metode dintr-un controller. Aceasta mi-a oferit flexibilitatea necesară pentru a gestiona logica aplicației și a asigura interacțiunea corectă între client și server.  
Intr-o aplicatie client-server, serverul poate avea functionalitati si subsisteme complexe care pot fi dificil de utilizat direct de catre client. Utilizand pattern-ul Facade, pot defini o interfata simplificata si coerenta care sa ofere o abstractie a functionalitatilor complexe ale serverului.

Facade actioneaza ca un strat intermediar intre client si server, oferind un set de metode simplificate si bine definite pe care clientul le poate utiliza pentru a interactiona cu serverul. Facade preia responsabilitatea de a coordona interactiunile si operatiunile complexe necesare pentru a obtine rezultatele dorite de client.

4.5 Diagrame de secventa



Figure 26 Diagrama de secventa pentru obtinerea tuturor operelor de arta



Figure 27 Diagrama de secventa pentru filtrarea operelor de arta



Figure 28 Diagrama de secventa pentru sortarea operelor de arta dupa anul realizarii



Figure 29 Diagrama de secventa pentru cautearea unei opre de arta dupa titlu

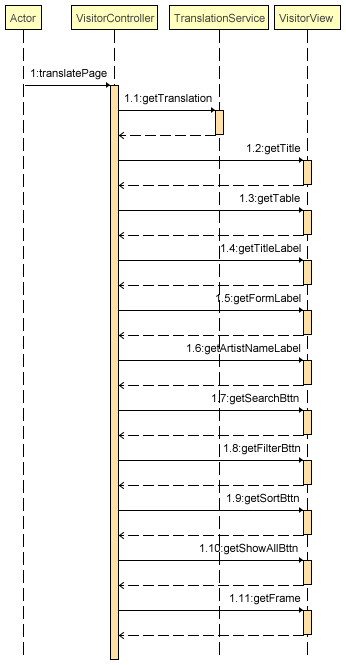


Figure 30 Diagrama de secventa pentru traducerea unei pagini

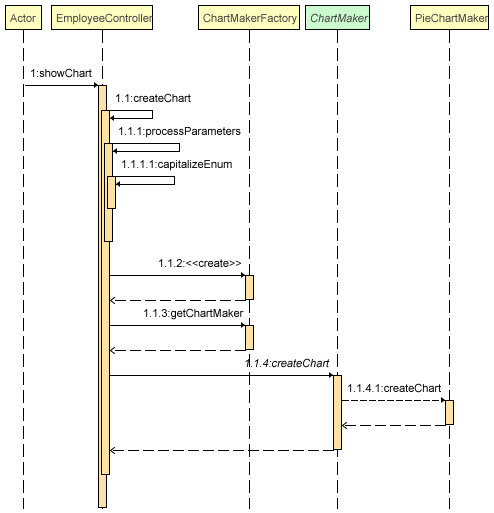


Figure 31 Diagrama de secventa pentru afisarea graficelor cu statistici



Figure 32 Diagrama de secventa pentru adaugarea unei noi opere de arta



Figure 33 Diagrama de secventa pentru actualizarea unei opere de arta

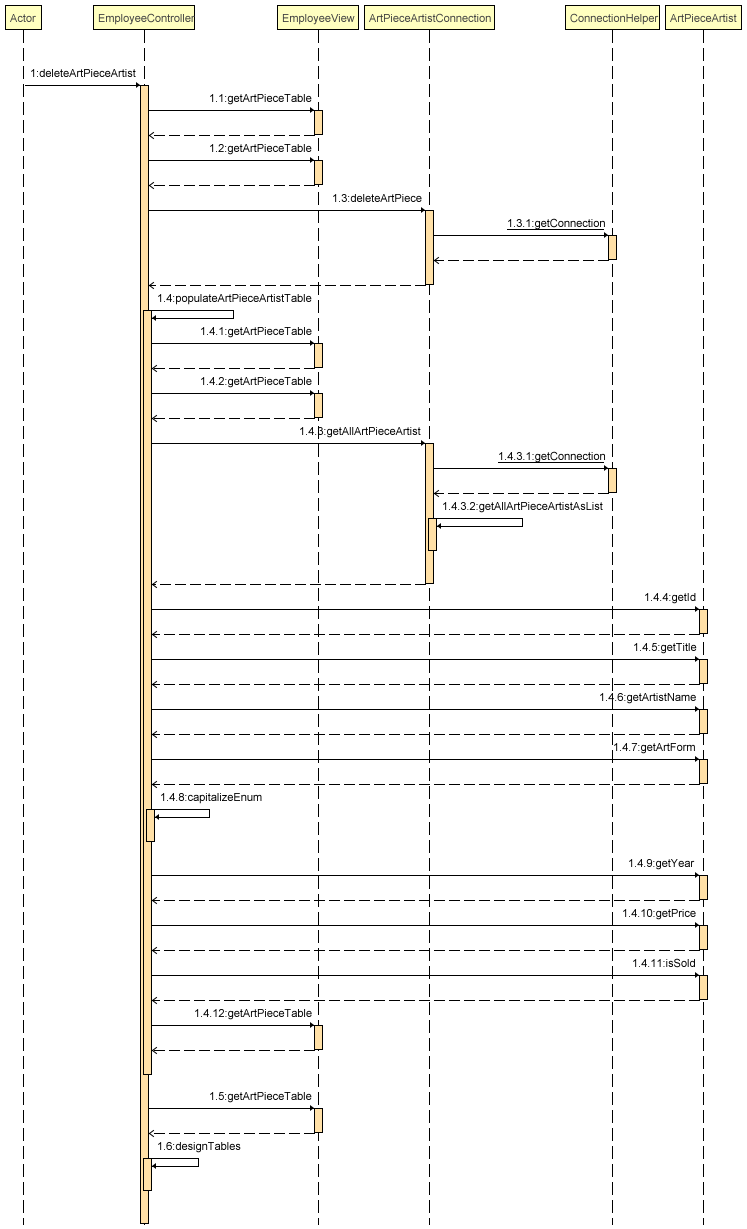


Figure 34 Diagrama de secventa pentru stergerea unei opere de arta

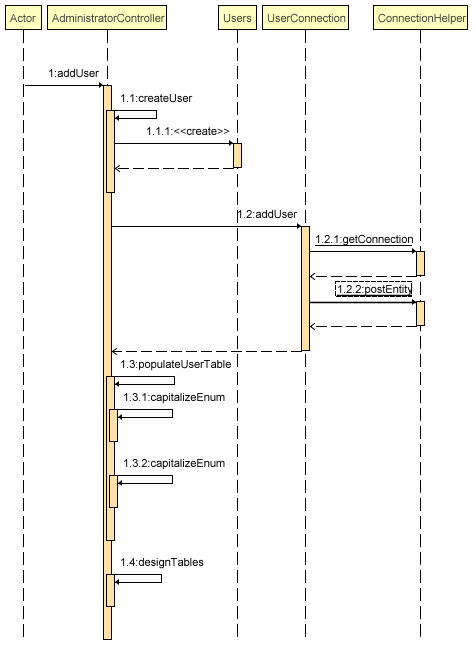


Figure 35 Diagrama de secventa pentru adaugarea unui nou utilizator

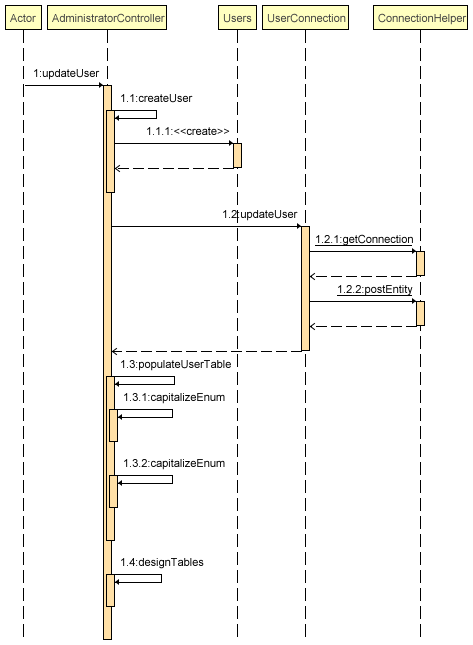


Figure 36 Diagrama de secventa pentru actualizarea unui utilizator

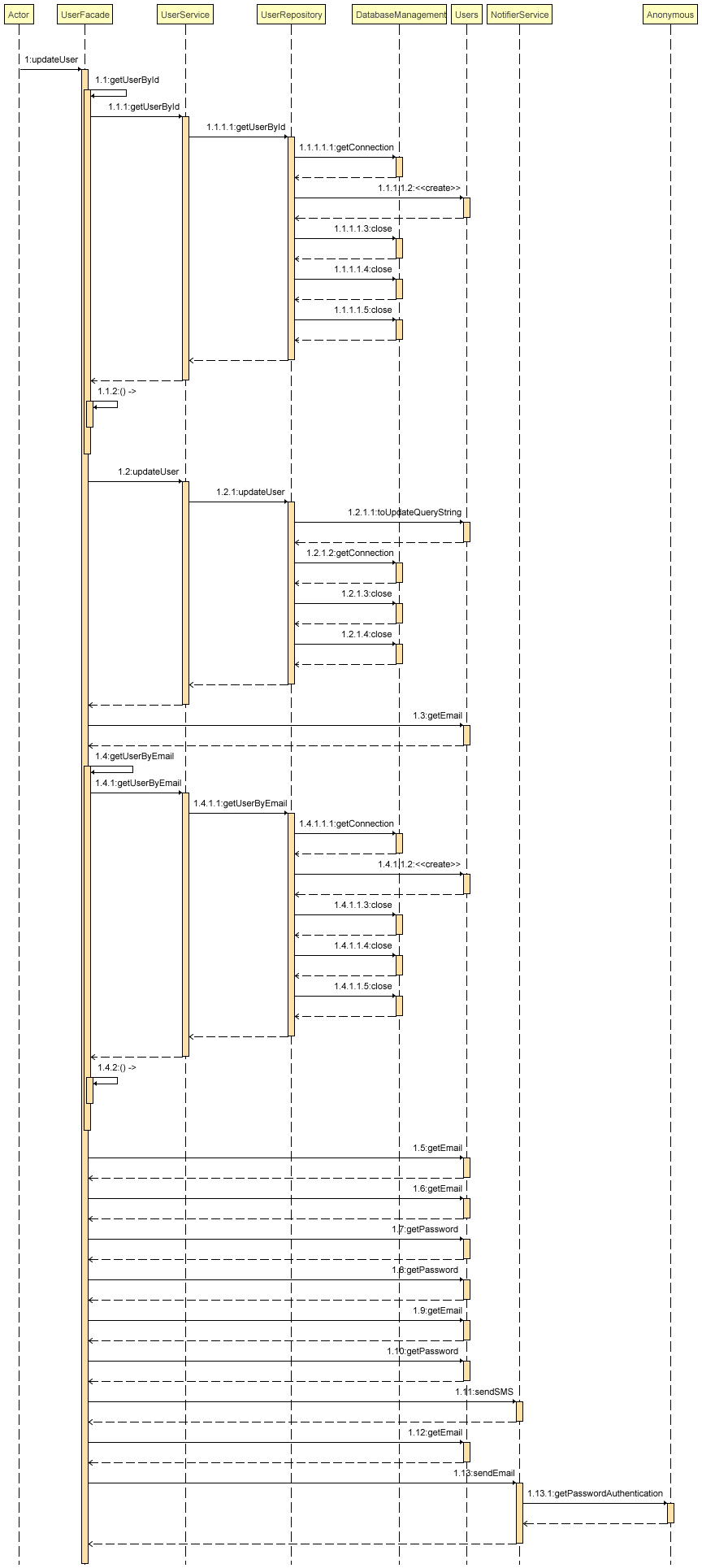


Figure 37 Diagrama de secventa pentru notificarea unui utilizator actualizat

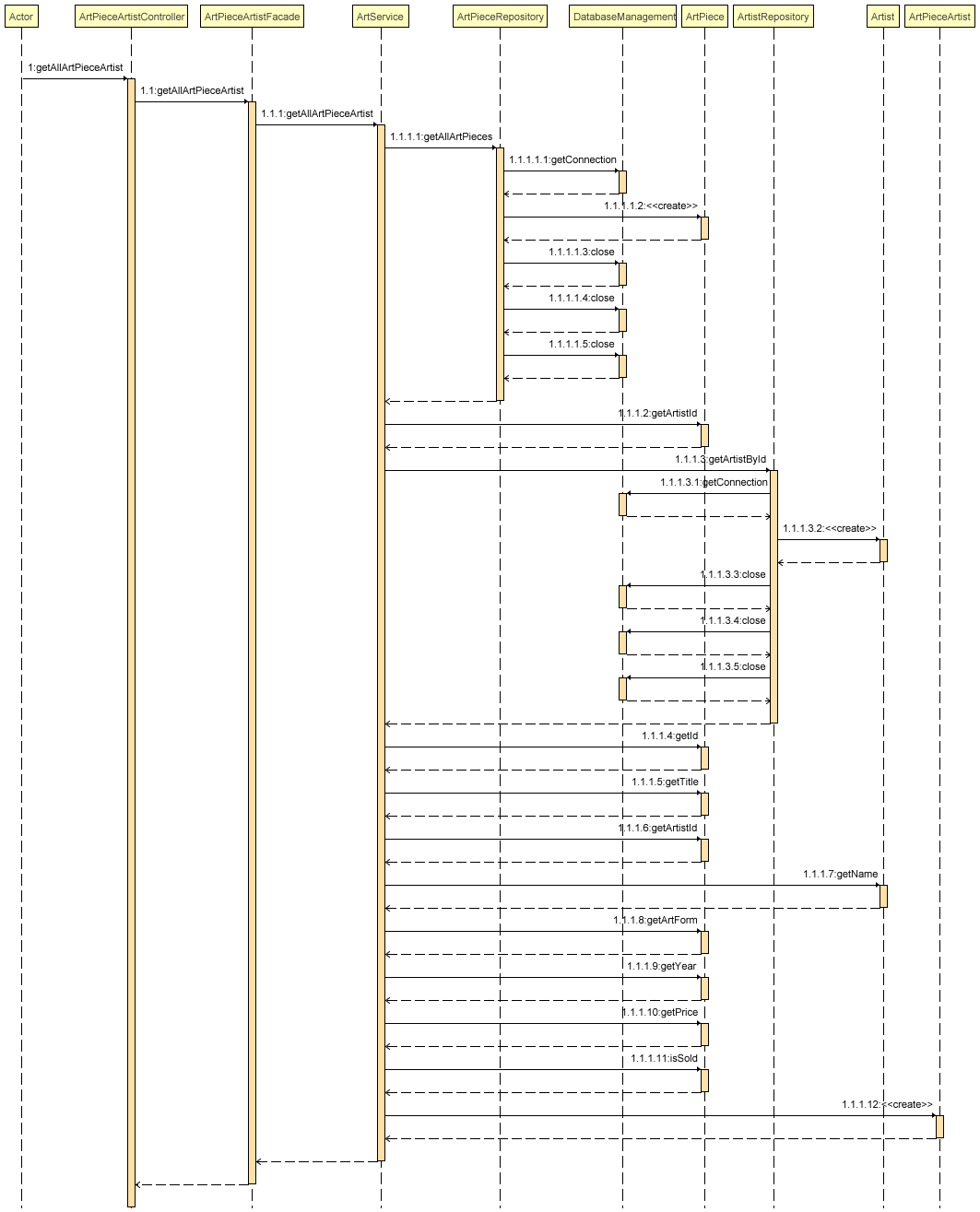


Figure 38 Diagrama de secventa pentru obtinerea listei cu toate operele de arta de catre server

5. Descrierea aplicatiei

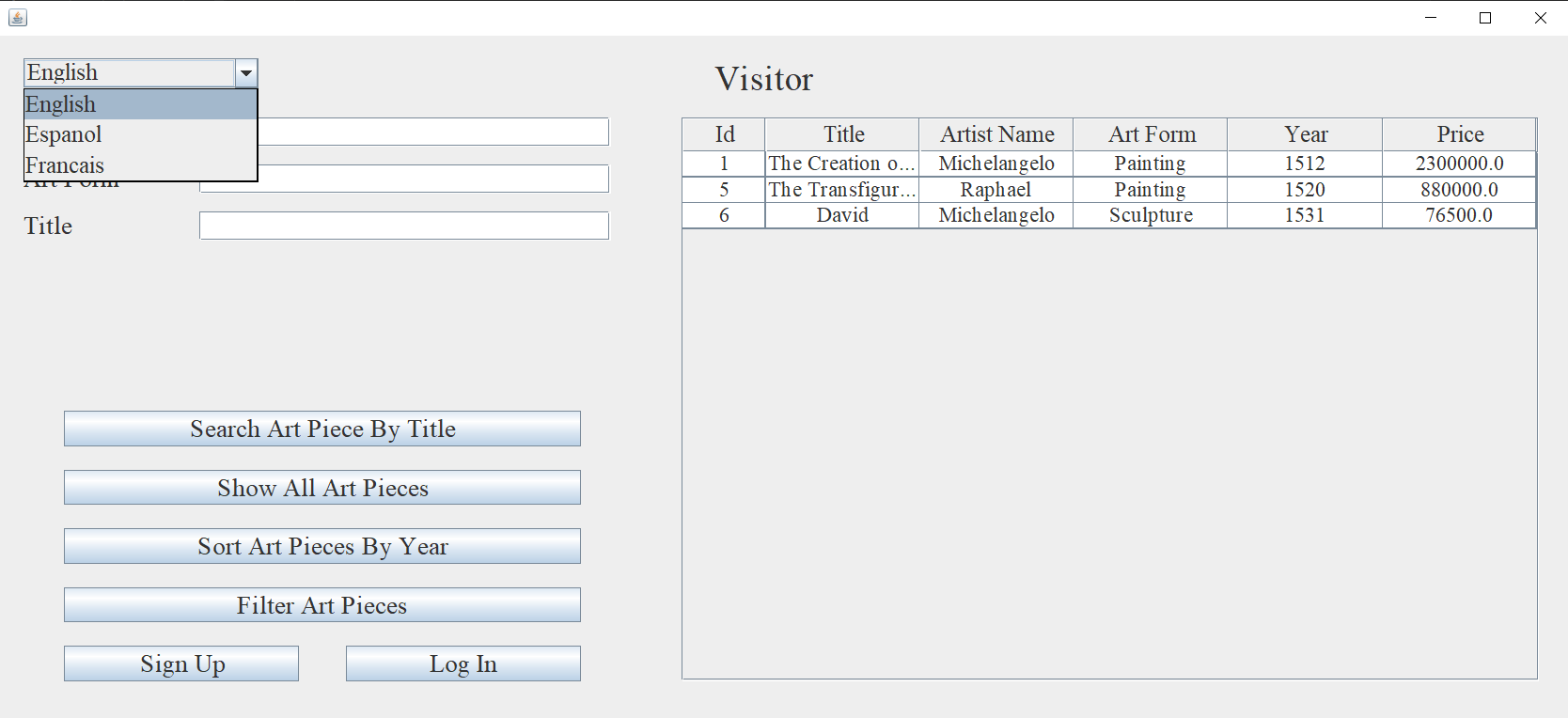
Aplicatia se deschide direct cu pagina utilizatorilor de tip Visitor intrucat acestia nu necesita autentificare.  
In momentul deschiderii paginii este afisat un tabel cu operele de arta alaturi de numele artistului de care au fost create. Prin intermediul a 4 butoane vizitatorul poate vizualiza toate operele de arta, sa le sorteze dupa anul realizarii, sa le filtreze dupa numele artistului si/sau dupa tipul operei de arta si sa caute dupa titlu. Filtrarea cat si cautarea permit potrivirea partiala a parametrilor pentru a facilita cautarea. Tot in aceasta pagina este prezent si un meniu drop-down din care se poate selecta limba in care vor fi traduse toate paginile aplicatiei

Figure 39 Pagina Visitor cu meniul pentru alegerea limbii

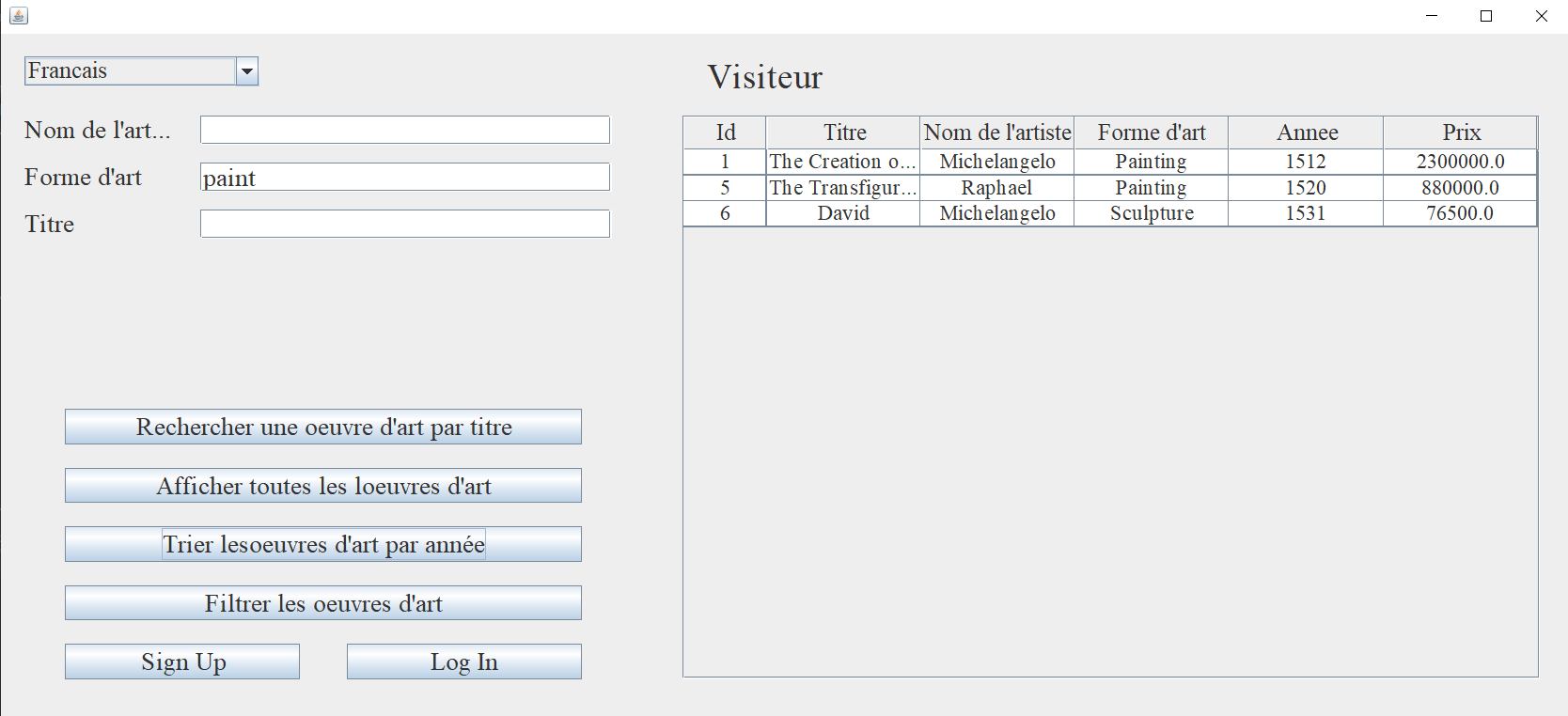


Figure 40 Pagina Visitor tradusa in Franceza cu rezultate filtrate

De asemenea, prezente pe aceasta pagina sunt butoanele de Sign Up si Log In ce permit creearea unui nou cont, respectiv accesul la un cont.

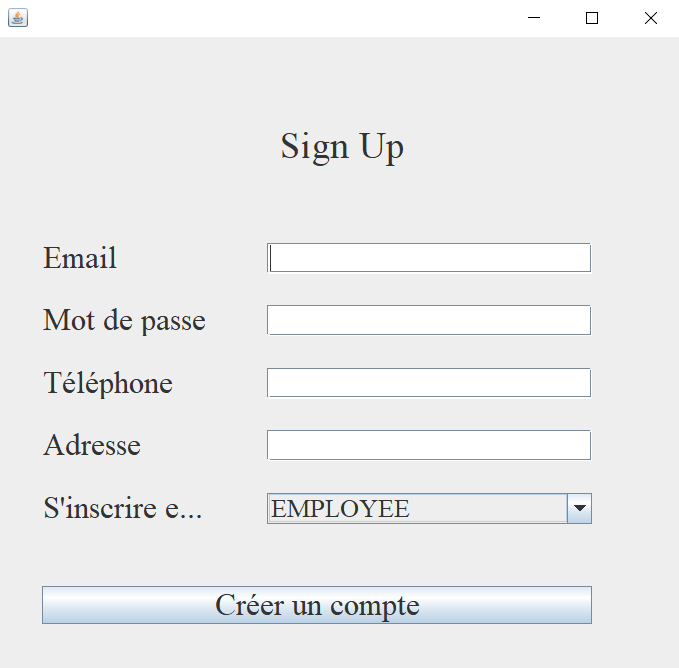


Figure 41 Pagina Sign Up

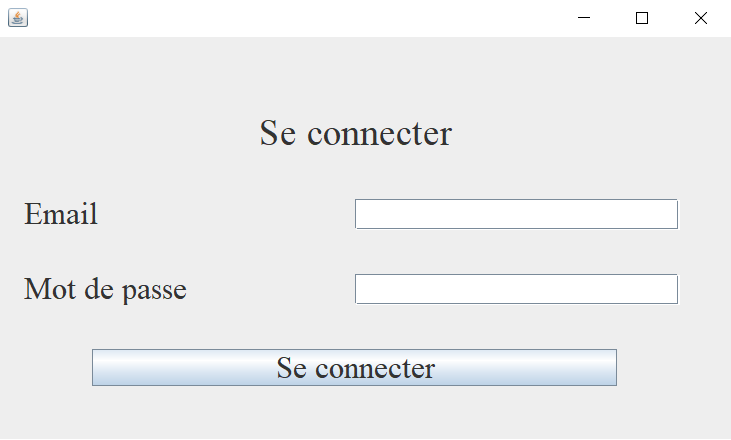


Figure 42 Pagina Log In

Pentru accesul la celelate view-uri utilizatorul trebuie sa faca operatia de Log In, aceasta rezultand in diferite erori in cazul in care email-ul nu a fost inregistrat, parola e gresita sau contul nu a fost enabled de catre un administrator.

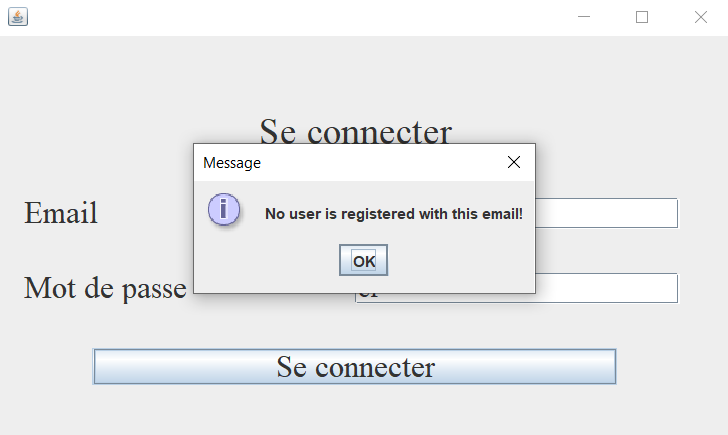


Figure 43 Eroare la Log In

View-ul pentru employee contine tabele pentru Artist si pentru Opere de Arta. Din acest view, employee-ul poate realiza aceleasi operatii ca si vizitatorul, cat si operatii CRUD asupra operelor de arta, si implicit asupra artistilor. De asemenea, din acest view employee-ul poate exporta liste cu operele de arta in diferite formate. Employee-ul este singurul care poate vedea si operele de arta vandute. La relizarea unui click pe unul dintre randurile tabelei, campurile aferente acestuia vor fi completate pentru a facilita editarea.

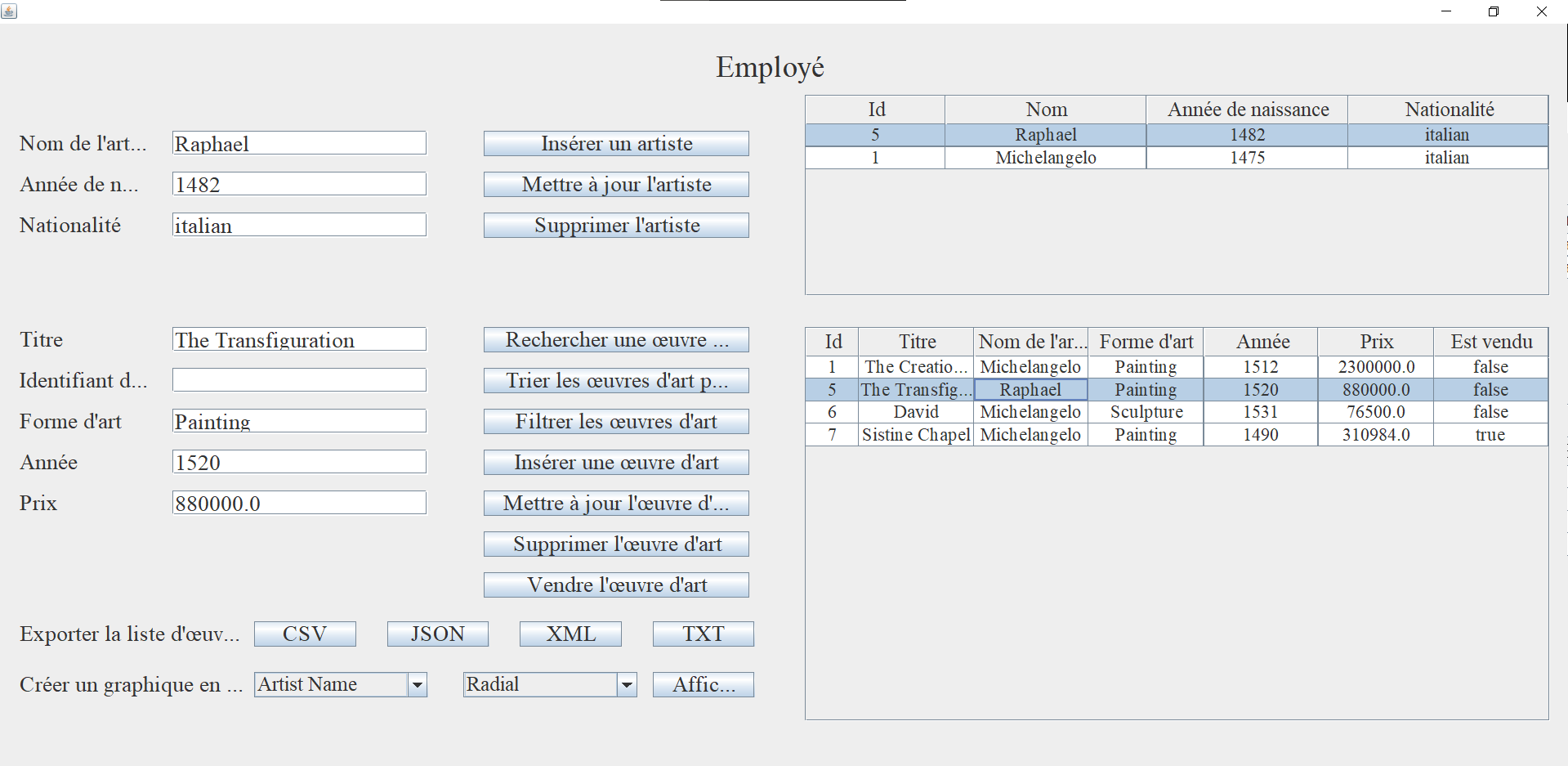


Figure 44 Pagina Employee cu randuri selectate din tabele

De asemenea, din cadrul acestei pagini, cu ajutorul a 2 liste drop-down se pot selecta parametrii ce vor fi utilizati pentru analiza unor statistici legate de operele de arta sub forma unor grafice, cat si structura acestor grafice

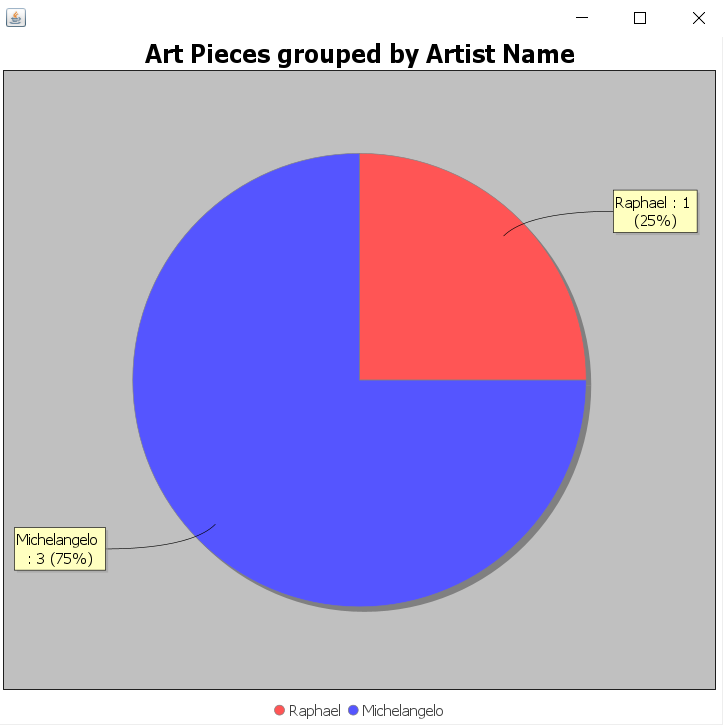


Figure 45 Chart ce prezinta procentajul operelor de arta clasificat dupa numele artistului

View-ul pentru administrator permite aceleasi operatii ca si cele ale vizitatorului, cat si vizualizarea listei utilizatorilor care necesita autentificarea si gestionarea acestora. Intrucat atunci cand se creeaza un cont nou, acesta nu este enabled, administratorul are rolul esential de a modifica aceasta valoare pentru a permite accesul ulterior la cont.

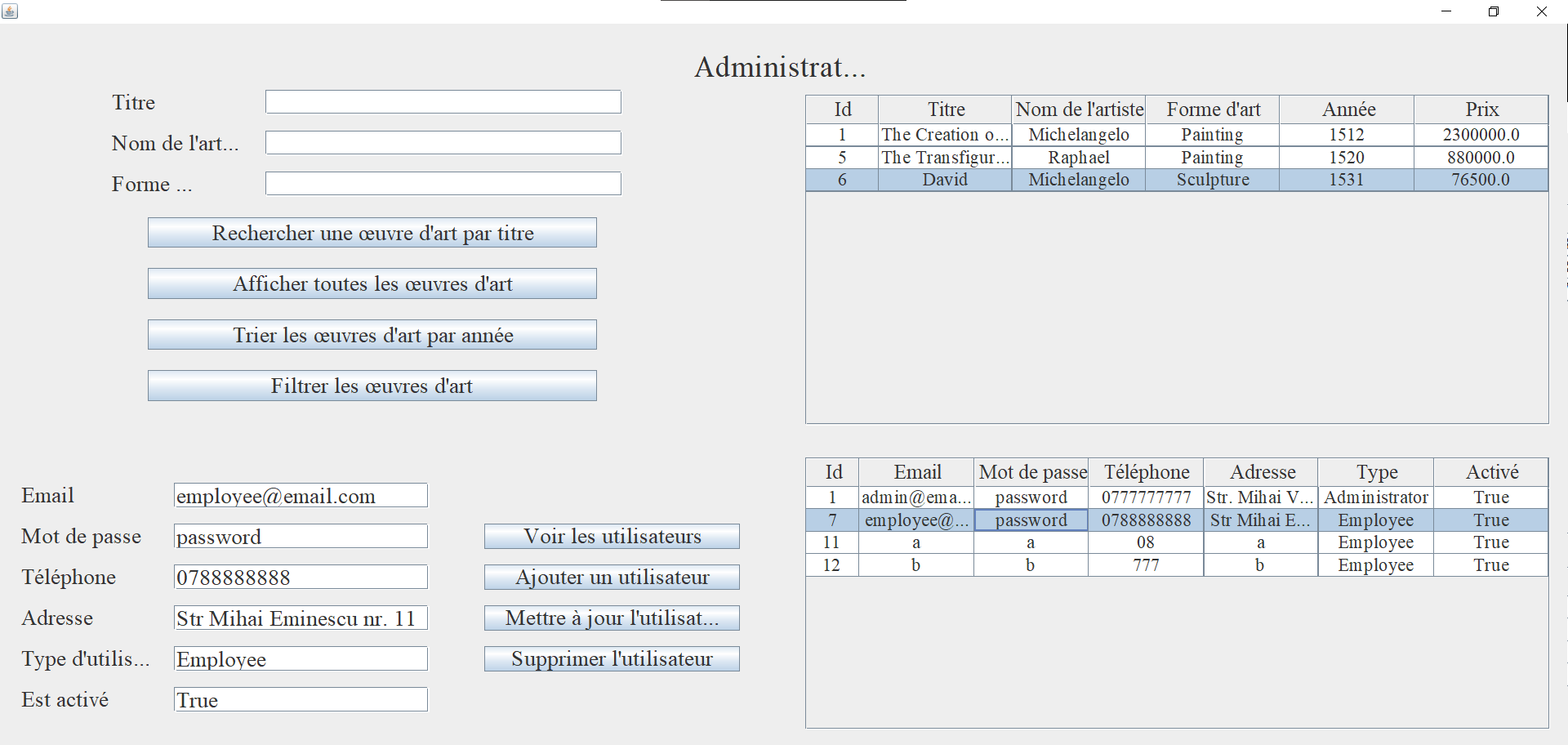


Figure 46 Pagina Administrator

In proiectul meu, am folosit mai multe design patterns pentru a imbunatati structura si modularitatea aplicatiei.

Observer: Am implementat acest pattern pentru a permite o comunicare eficienta intre diferitele componente ale aplicatiei. Observer-ul permite notificarea automata a obiectelor atunci cand se produce o schimbare de stare intr-un subiect, astfel incat sa se mentina sincronizarea si coerenta intre componente, astfel folosindu-l pentru a notifica toate paginile despre modificarea limbii.

Singleton: Am utilizat Singleton pentru a asigura ca exista o singura instanta a clasei ce realizeaza operatiile pe baza de date in intreaga aplicatie. Acest pattern este util atunci cand dorim sa avem acces la o singura resursa globala si sa evitam crearea repetata a instantelor.

Prototype: Am aplicat acest pattern atunci cand am avut nevoie sa cream noi obiecte bazate pe un model existent. Prototype permite clonarea unui obiect pentru a crea rapid si eficient instante noi cu aceleasi caracteristici, evitand astfel procesul costisitor de initializare.

Facade: Am utilizat Facade pentru a oferi o interfata simplificata si unificata catre client, ascunzand complexitatea din spatele unor subsisteme sau componente. Acest pattern simplifica interactiunea client-server, oferind o interfata clara si usor de utilizat pentru client.

Factory Method: Am folosit Factory Method pentru a permite crearea obiectelor intr-un mod flexibil si modular. Acest pattern ofera o metoda de fabrica care abstractizeaza procesul de creare a instantelor si permite extinderea facila a logicii de creare prin intermediul claselor derivate.