**TEMA 1**

**PROIECTARE SOFTWARE**

**APLICAȚIE DE GESTIUNE A EXPONATELOR DINTR-UN MUZEU FOLOSIND ARHITECTURA MVP**

Proiect realizat de Șerban Sebastian – Mihai

Grupa 30233

2023

Cuprins

**Table of Contents**

[1. Enunțul problemei 3](#__RefHeading___Toc198_2089743297)

[2. Instrumentele utilizate 4](#__RefHeading___Toc200_2089743297)

[3. Justficarea limbajului de programare ales 4](#__RefHeading___Toc202_2089743297)

[4. Proiectarea aplicației 5](#__RefHeading___Toc204_2089743297)

[5. Ghid de utilizare a aplicației 7](#__RefHeading___Toc206_2089743297)

# Enunțul problemei

Dezvoltați o aplicație ce poate fi utilizată în muzee. Aplicația va avea 3 tipuri de utilizatori: vizitator al muzeului, angajat al muzeului și administrator.

Utilizatorii de tip vizitator pot efectua următoarele operații fără autentificare:

* Vizualizarea listei tuturor operelor de artă expuse în muzeu sortate după artist;
* Filtrarea listei operelor de artă plastică după următoarele criterii: artist, tipul operei de artă, etc.

Utilizatorii de tip angajat pot efectua următoarele operații după autentificare:

* Toate operațiile permise utilizatorilor de tip vizitator;
* Operații CRUD în ceea ce privește persistența operelor de artă expuse.

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;
* Vizualizarea listei utilizatorilor care necesită autentificare.

# Instrumentele utilizate

Instrumentele utilizate pentru realizarea acestei aplicații sunt:

* IDE-ul Intellij:
* draw.io pentru diagrama de clase, entitate relație și use case;
* MySQL Workbench pentru verificarea integrității datelor introduse în baza de date:
* Materialele de curs de la disciplina Proiectare Software.

# Justficarea limbajului de programare ales

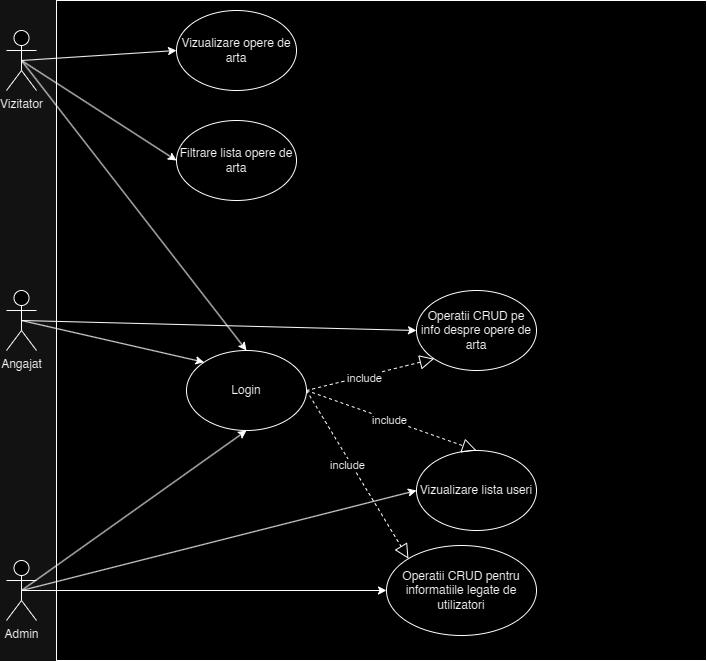
Pentru a realiza această aplicație voi utiliza limbajul de programare Java deoarece consider că este un limbaj de programare ce reușește să satisfacă cel mai bine necesitățile unei aplicații de acest tip, Java fiind un limbaj de programare ce înglobează în multitudinea de biblioteci unelte ce pot fi utilizate pentru a construi aplicații ce se bazează pe principiul Programării Orientate pe Obiecte.

De asemenea, utilizez Java și pentru a utiliza diferite dependițe ce aparțin framework-ului Spring, fără a creea o aplicație ce se bazează pe acest framework. Dependințele pe care le vom folosi nevor ușura munca atât pentru debuging și testare, cât și pentru conexiune la baza de date și interogări pe tabelele bazei de date pe care o vom folosi.

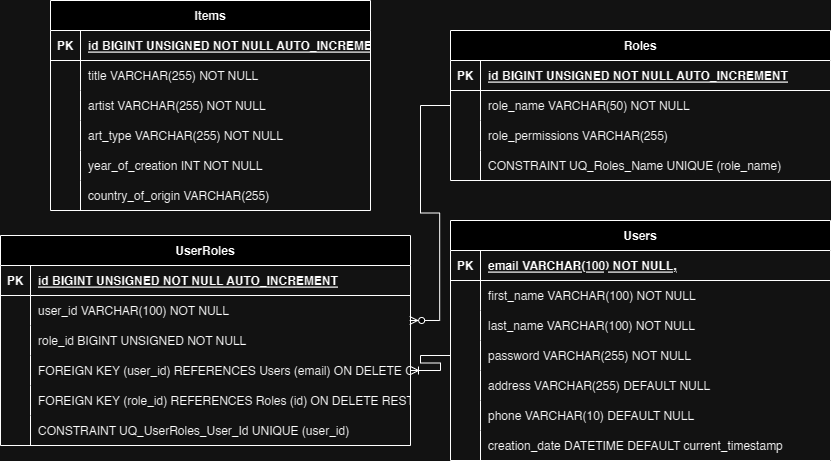
# Proiectarea aplicației

În faza de proiectare a aplicației am conceput 3 diagrame diferite care m-au ajutat să dezvolt aplicația finală, oferind practic un șablon pe care să îl urmez în implementare.

Prima diagramă pe care am realizat-o a fost diagrama use case care să definească totalitatea actorilor și a acțiunilor pe care aceștia le pot efectua.



După cum puteți vedea în diagrama de mai sus, am identificat 3 actori, anume vizitatorul, angajatul și administratorul ce pot efectua diferite acțiuni precum vizualizarea operelor de artă, modificarea acestora, vizualizarea utilizatorilor și modificarea acestora, în funcție de rolul pe care aceștia îl au.

În urma acestei diagrame de cazuri de utilizare, am început modelarea unei baze de date ce ar putea satisface necesitățile aplicației software pe care dorim să o implementăm și am ajuns la următoarea diagramă entitate relație:

După cum putem vedea, baza de date va conține 4 tabele.

Tabela Users reține toți utilizatorii înregistrați ce se pot autentifica în cadrul aplicației.

Tabela Roles reține toate rolurile pe care actanții le pot avea, mai puțin cel de simplu vizitator, deoarece rolul de simplu vizitator nu necesită permisiuni speciale pe întreaga durată a utilizării aplicației.

Tabela UserRoles sparge relația many to many ce se regăsea între tabela Users și tabela Roles, deoarece un utilizator poate fi atât angajat cât și administrator, așa că e mai bine să transformăm relația de many to many în două relații de one to many.

Tabela Items va stoca totalitatea exponatelor ce vor putea fi urmărite de către vizitatori și editate de către angajați sau administrator.

Acum că ambele diagrame au fost realizate, am trecut la proiectarea celei de-a treia diagrame, anume diagrama de clase care să faciliteze implementarea aplicației. Diagrama de clase este prea vastă pentru a putea fi curpinsă în acest document, însă voi explica fiecare pachet din cadrul diagramei și necesitatea lui în cadrul construcției.

Aplicația conține 3 mari pachete corespunzătoare numelui arhitecturii pe care aplicația o utilizează, anume Model, View și Presenter.

Pachetul Model conține toate clasele și interfețele necesare pentru a comunica cu baza de date și a transmite date între utilizator și back end.

**Pachetul Model** conține la rândul său mai multe subpachete ce sunt menite să atribuie o oarecare ordine claselor pe care le găzduiesc.

* Subpachetul Connection este menit să găzduiască toate clasele ce se ocupă cu crearea de conexiuni directe cu baza de date pentru a executa diferite operații CRUD pe tabelele bazei de date, dar de asemenea conțin și request-uri sql menite să creeze baza de date în cazul în care aceasta nu există.
* Subpachetul Data Class conține toate clasele care conțin exact aceleași câmpuri ca și tabelele bazei de date ce au fost descrise în diagrama entitate relație.
* Subpachetul DTO conține toate Data Transfer Object-urile, clase menite să prelucreze informația venită direct de la baza de date într-un mod în care să protejeze și confidențialitatea datelor și să nu transmită mai multe date decât este necesar pentru front-end.
* Subpachetul Query conține clase ce au drept scop înmagazinarea constantelor de tip șir de caractere ce definesc interogările pe care dorim să le facem în limbajul SQL.
* Subpachetul Repository creează legătura între conexiunea la baza de date și clasele din subpachetul Data Class, folosind interogările definite în subpachetul Query pentru a obține informațiile necesare.
* Subpachetul RowMapper este menit să descrie modul în care dorim să mapăm informația venită în urma request-urilor SQL în cadrul obiectelor aplicației.
* Subpachetul Service conține totalitatea request-urilor pe care aplicația le poate face pentru a obține date și le transformă în obiecte de tip DTO pentru a asigura siguranța datelor din cadrul bazei de date.

**Pachetul Presenter** conține totalitatea claselor ce se ocupă cu legătura dintre back-end și front-end.

* Subpachetul ActionListeners conține totalitatea acțiunilor pe care butoanele din cadrul aplicației trebuie să le efectueze și apelurile către baza de date aferente acțiunilor.
* Subpachetul Navigator conține o interfață și o implementarea a acesteia ce este menită să gestioneze navigarea între frame-urile aplicației.

**Pachetul View** conține totalitatea claselor ce descriu interfața grafică a aplicației.

# Ghid de utilizare a aplicației

Atunci când aplicația este lansată în execuție suntem întâmpinați cu o fereastră menită unui vizitator oarecare, ce nu necesită logare cu un cont.

Din acest ecran putem realiza următoarele acțiuni, să ne logăm cu un cont de angajat sau de administrator sau să sortăm exponatele în funcție de diferite caracteristici. A screenshot of a computer

Description automatically generated

A screenshot of a computer login form

Description automatically generatedDacă alegem să apăsăm butonul de login vom fi întâmpinați de un ecran ce ne va cere să introducem credențialele necesare pentru autentificare.

Trebuie să introducem un e-mail valid și parola corectă pentru a putea trece la pasul următor, sau să revenim la ecranul inițial, menit unui simplu vizitator.

Dacă ne vom autentifica cu un cont de angajat atunci vom fi întâmpinați de o interfață asemănătoare cu cea a vizitatorului, singura diferență fiind că putem edita obiectele, le putem șterge sau chiar putem adăuga obiecte noi în colecția muzeului. A screenshot of a computer

Description automatically generated

Dacă alegem să edită sau să adăugăm obiecte vom fi redirecționați spre pagina de editare, respectiv adăugare a unui obiect.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedDacă alegem să ștergem un obiect vom fi întâmpinați de un chenar de dialog ce ne solicită să reconfirmăm ștergerea obiectului.

Dacă ne autentificăm cu un cont de administrator, vom fi întâmpinați de următoarea interfață care ne permite să alegem între a edita obiecte și a edita utilizatori.

A screenshot of a computer

Description automatically generated

Dacă alegem să edităm obiecte practic vom intra în modul angajat.

Dacă alegem să edităm utilizatori vom fi întâmpinați de următorul ecran.

A screenshot of a computer

Description automatically generated

După cum putem observa, se păstrează aceeași logică de funcționare ca la obiecte, doar că edităm utilizatori.

Pentru a opri execuția aplicației pur și simplu apăsăm butonul de închidere al acesteia.