***DOCUMENTAȚIE – TEMA 3 – MUZEE - MVC***

***Sîrbu Cosmin-Claudiu, grupa 30231***

***1. Introducere:***

Acest proiect, denumit "MuseumApp", reprezintă o aplicație inovatoare concepută pentru a revoluționa experiența vizitatorilor în muzee, aducând în prim-plan tehnologia și accesibilitatea. Într-o lume în care cultura și arta sunt esențiale pentru îmbogățirea spirituală și educațională a societății, MuseumApp servește ca o punte între patrimoniul cultural și publicul modern, facilitând accesul la informații valoroase și îmbunătățind interacțiunea cu operele de artă.

Prin intermediul unei interfețe prietenoase și ușor de utilizat, MuseumApp oferă utilizatorilor posibilitatea de a explora colecții de artă din diverse muzee, oferind detalii fascinante despre operele de artă și artiștii lor. Aplicația încurajează descoperirea și învățarea autonomă, oferind totodată funcționalități interactive care transformă vizita la muzeu într-o experiență personală și profundă.

Dezvoltat cu un puternic accent pe accesibilitate și interactivitate, MuseumApp se adresează atât publicului larg, cât și profesioniștilor din domeniul muzeografic, educatorilor și studenților. Utilizatorii pot căuta opere de artă bazându-se pe criterii diverse, pot să învețe despre istoria și contextul fiecărei piese și pot chiar să participe la tururi virtuale ale expozițiilor.

Pentru muzee, MuseumApp reprezintă un instrument valoros de marketing și educație, permițând instituțiilor să-și extindă audiența și să ofere vizitatorilor o experiență îmbogățită, chiar și dincolo de zidurile fizice ale muzeului. Prin funcționalitățile sale, aplicația deschide noi orizonturi pentru explorarea artelor, făcând cultura mai accesibilă și atrăgătoare pentru publicul din era digitală.

***2. Cerința proiectului:***

Dezvoltati o aplicatie care poate fi utilizata in muzee. Aplicatia va avea 3 tipuri de utilizatori: vizitator al muzeului, angajat al muzeului si administrator.

Utilizatorii de tip vizitator pot efectua urmatoarele operatii fara autentificare:

1. Vizualizarea listei tuturor operelor de arta expuse in muzeu sortate dupa artist.(vizualizarea include si redarea unor imagini cu operele de arta; intre 1 si 3 imagini pentru fiecare opera de arta);

2. Filtrarea listei operelor de arta plastica dupa urmatoarele criterii: artist, tipul operei de arta, etc.

3. Căutarea unei opere de artă după titlu.

Utilizatorii de tip angajat pot efectua urmatoarele operatii dupa autentificare:

1. Toate operatiile permise utilizatorilor de tip vizitator.

2. Operatiile CRUD in ceea ce priveste persistenta operelor de arta expuse.

3. Salvare liste cu operele de artă expuse în mai multe formate: csv, json, xml, doc.

4. Vizualizarea unor statistici legate de operele de arta din muzeu utilizand grafice (structira radiala, structura inelara, de tip coloana, etc.).

Utilizatorii de tip administrator pot efectua urmatoarele operatii dupa autentificare:

1. Toate operatiile permise utilizatorilor de tip vizitator.

2. Operatii CRUD pentru informatiile legate de utilizatorii care necesita autentificare.

3. Vizualizarea listei utilizatorilor care necesita autentificare.

4. Filtrarea listei utilizatorilor după tipul utilizatorului.

Interfata grafica a aplicatiei va fi disponibila in cel putin 3 limbi de circulatie internationala.

***3. Instrumente utilizate:***

IntelliJ IDEA este un mediu de dezvoltare integrat (IDE) robust și sofisticat, creat de JetBrains, destinat dezvoltării software-ului cu o eficiență și productivitate îmbunătățite. Apreciat de programatori pentru interfața sa intuitivă și setul bogat de funcționalități, IntelliJ IDEA sprijină mai multe limbaje de programare, inclusiv Java, Kotlin, Scala și Groovy, fiind ideal pentru dezvoltarea aplicațiilor web, mobile și enterprise.

Una dintre caracteristicile principale ale IntelliJ IDEA este capacitatea sa de analiză profundă a codului, care permite identificarea rapidă a erorilor și sugerează optimizări, îmbunătățind calitatea și performanța codului. Autocompletarea inteligentă, refactoringul puternic și sistemul de gestionare a versiunilor sunt alte funcții care fac din IntelliJ o alegere populară printre dezvoltatorii profesioniști.

Prin plugin-uri și personalizări, utilizatorii pot extinde funcționalitățile IDE-ului pentru a se potrivi nevoilor specifice ale proiectului. Fie că este vorba de dezvoltarea unei aplicații Android sau de construirea unui sistem enterprise complex, IntelliJ IDEA oferă un mediu de lucru coerent și puternic, menit să stimuleze creativitatea și eficiența dezvoltatorilor.

***4. Justificarea limbajului ales:***

Java este un limbaj de programare robust și versatil, ales datorită abilității sale de a răspunde eficient unei game largi de necesități de dezvoltare software. Este apreciat pentru portabilitatea sa, codul Java putând rula nealterat pe diverse platforme, fără a necesita modificări, grație sloganului „Write Once, Run Anywhere”. Această caracteristică îl face ideal pentru dezvoltarea de aplicații cross-platform, de la aplicații desktop și mobile, până la soluții complexe pentru servere.

Java se bucură de o comunitate vastă de dezvoltatori, ceea ce îi conferă un avantaj semnificativ prin bibliotecile extinse și framework-urile disponibile, facilitând astfel dezvoltarea rapidă și eficientă a aplicațiilor. De asemenea, Java este recunoscut pentru gestionarea automată a memoriei prin colectorul de gunoi, eliminând unele dintre cele mai comune probleme și erori asociate cu managementul memoriei.

Siguranța tipurilor, orientarea obiect și scalabilitatea sunt alte aspecte care contribuie la popularitatea Java. Aceste caracteristici permit dezvoltarea de sisteme robuste și sigure, care pot evolua ușor pe măsură ce necesitățile se schimbă. În plus, Java continuă să inoveze și să se adapteze la noile tehnologii, asigurându-și relevanța și eficacitatea într-o lume a tehnologiei în constantă schimbare. Astfel, pasiunea pentru Java nu este doar o preferință personală, ci și o alegere strategică pentru dezvoltarea de soluții software durabile și performante.

5. ***Diagrame UML:***

O imagine care conține text, diagramă, Font, Plan

Descriere generată automat

*Figura 1. Diagrama de clase restransă.*

Aceasta diagramă de clase reprezintă arhitectura restrânsă a unui sistem bazat pe modelul MVC (Model-View-Controller).

Modelul:

- ArtWork: Reprezintă o operă de artă cu atribute precum `id`, `title`, `artist`, `year`, `type`, și `imagePath`.

- MuseumAdmin, MuseumEmployee, MuseumVisitor: Extind clasa `User` și adaugă specificații pentru fiecare tip de utilizator al sistemului muzeal.

- User: Conține informații generale despre utilizatori, cum ar fi `id`, `username`, `password`.

Controlere:

- ArtWorkController: Gestionază logica legată de operele de artă, inclusiv adăugarea, actualizarea, ștergerea și căutarea acestora.

- EmployeeController, AdminController: Gestionază funcționalitățile specifice angajaților și administratorilor, cum ar fi manipularea utilizatorilor și a operelor de artă.

- LogInController, MainScreenController: Gestionază funcționalitățile legate de autentificare și navigare în interfața principală.

Repozitorii:

- ArtWorkRepository, UserRepository: Interacționează cu baza de date pentru a efectua operațiuni CRUD (Create, Read, Update, Delete) pe opere de artă și utilizatori.

- ConnectionBD: Gestionază conexiunile la baza de date.

Vizualizări:

- MainScreen: Ecranul principal al aplicației.

- ArtWorkListView, ViewImages: Vizualizări pentru listarea și vizualizarea imaginilor operelor de artă.

- LogIn, AdminUI, EmployeeUI: Interfețe pentru autentificare și pentru utilizatorii de tip admin și angajat.

Observer Pattern:

- Observer, LanguageObservable: Gestionază schimbările de limbă în aplicație, implementând pattern-ul Observer pentru actualizarea dinamică a interfeței utilizatorului în funcție de limbă.

Diagrama ilustrează relațiile și interacțiunile dintre aceste componente, evidențiind cum este structurată aplicația pentru a separa clar logica de afaceri (Model), logica de prezentare View) și logica de control (Controller). Astfel, asigură o dezvoltare modulară și ușor de întreținut.

O imagine care conține captură de ecran, text, diagramă

Descriere generată automat

*Figura 2. Diagrama Use-Case*

Diagrama cazurilor de utilizare ilustrează interacțiunile utilizatorilor cu sistemul muzeal, evidențiind funcționalitățile disponibile pentru fiecare tip de utilizator (vizitator, angajat și administrator). Diagrama este împărțită în trei secțiuni majore, corespunzând fiecărui tip de utilizator și operațiunilor pe care le pot efectua.

Vizitator

Vizitatorul are acces la următoarele cazuri de utilizare:

- Filtrare Opere de Artă: Vizitatorul poate aplica filtre pentru a căuta opere de artă pe baza unor criterii specifice (titlu, autor, an, tip).

- Căutare Opere de Artă: Vizitatorul poate căuta opere de artă specificând titlul.

- Vizualizare Imagini: Vizitatorul poate vizualiza imagini ale operelor de artă disponibile în muzeu.

Angajat

Angajatul are acces la cazurile de utilizare ale vizitatorului, plus următoarele operațiuni:

- Adăugare Operă: Angajatul poate adăuga o nouă operă de artă în sistem, specificând detalii precum titlu, artist, an, tip și calea imaginii.

- Căutare Operă: Angajatul poate căuta opere de artă în sistem.

- Generare Documente: Angajatul poate genera documente care să includă informații despre operele de artă.

- Ștergere Operă: Angajatul poate șterge opere de artă existente din sistem.

- Actualizare Operă: Angajatul poate actualiza informațiile despre operele de artă existente.

Admin

Administratorul are acces la toate cazurile de utilizare ale vizitatorului și angajatului, plus operațiuni suplimentare legate de gestionarea utilizatorilor:

- Adăugare Utilizator: Administratorul poate adăuga noi utilizatori în sistem.

- Căutare Utilizator: Administratorul poate căuta utilizatori în sistem.

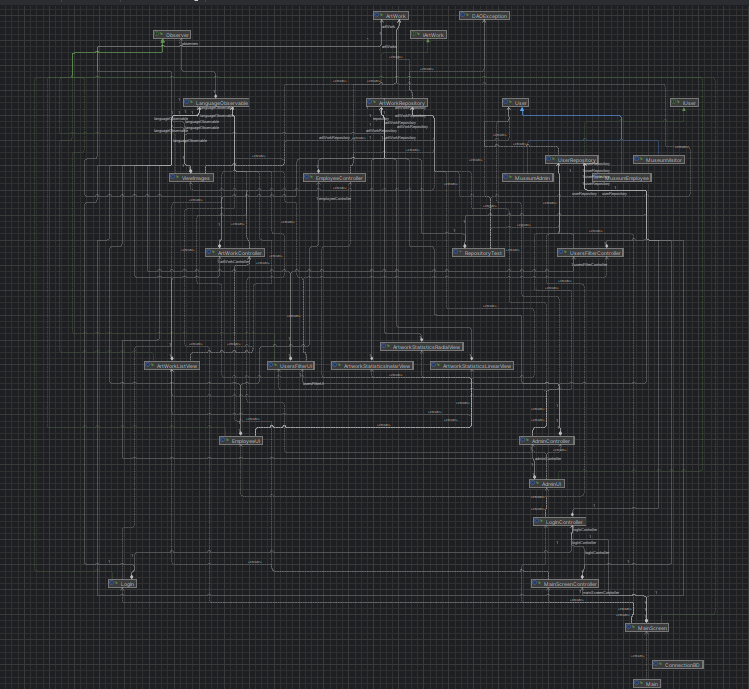
- Filtrare Utilizatori: Administratorul poate aplica filtre pentru a căuta utilizatori pe baza unor criterii specifice.

- Ștergere Utilizator: Administratorul poate șterge utilizatori existenți din sistem.

- Actualizare Utilizator: Administratorul poate actualiza informațiile despre utilizatorii existenți.

- Generare Grafice: Administratorul poate genera grafice pentru a vizualiza statistici legate de activitățile din sistem.

Această diagramă oferă o privire clară asupra funcționalităților oferite de sistem și modul în care acestea sunt distribuite între diferiți utilizatori, asigurând o gestionare eficientă și bine organizată a muzeului.



*Figura 3. Diagrama UML – IntelliJ*

Diagrama UML prezentată reprezintă o arhitectură complexă a unui sistem bazat pe MVC (Model-View-Controller) pentru gestionarea unui muzeu. Aceasta include multiple clase, cum ar fi `ArtWork`, `MuseumAdmin`, `MuseumEmployee`, `User`, și `LogIn`, împreună cu relațiile dintre ele. Controller-urile (`ArtWorkController`, `EmployeeController`) și repository-urile (`ArtWorkRepository`, `UserRepository`) gestionează logica aplicației și interacțiunile cu baza de date. View-urile (`ArtWorkListView`, `LogInView`, `MainScreen`) sunt responsabile pentru interfața grafică, iar modelele (`ArtWork`, `User`) reprezintă entitățile de bază. Diagrama evidențiază interdependențele și colaborările dintre aceste componente pentru a oferi funcționalități complexe utilizatorilor (vizitatori, angajați și administratori).

***6. Descrierea aplicației:***

Structura Aplicatiei:

1. Controller:

- `AdminController`: Gestionează funcționalitățile specifice administratorilor.

- `ArtWorkController`: Gestionează operațiile CRUD și alte funcționalități legate de operele de artă.

- `EmployeeController`: Gestionează funcționalitățile specifice angajaților.

- `LogInController`: Gestionează procesul de autentificare.

- `MainScreenController`: Controlează ecranul principal al aplicației.

- `UsersFilterController`: Gestionează filtrarea utilizatorilor.

2. Languages:

- `LanguageObservable`: Permite actualizarea dinamică a limbii în interfața utilizator.

- `Observer`: Interfață pentru observatori care sunt notificați la schimbările de limbă.

3. Model:

- `ArtWork`: Reprezintă o operă de artă.

- `MuseumAdmin`, `MuseumEmployee`, `MuseumVisitor`: Reprezintă diferite tipuri de utilizatori.

- `User`: Clasă de bază pentru utilizatori.

4. Repo:

- `ArtWorkRepository`: Gestionează interacțiunea cu baza de date pentru operele de artă.

- `ConnectionBD`: Gestionarea conexiunii cu baza de date.

- `DAOException`: Tratarea excepțiilor de tip DAO.

- `IArtWork`: Interfață pentru repository-ul de opere de artă.

- `UserRepository`: Gestionează interacțiunea cu baza de date pentru utilizatori.

5. View:

- `AdminUI`: Interfața utilizator pentru administratori.

-`ArtWorkListView`, `ArtWorkStatisticsInelarView`, `ArtWorkStatisticsLinearView`, `ArtWorkStatisticsRadialView`: Interfețe pentru vizualizarea și gestionarea operelor de artă.

- `EmployeeUI`: Interfața utilizator pentru angajați.

- `LogIn`: Ecranul de autentificare.

- `MainScreen`: Ecranul principal al aplicației.

- `UsersFilterUI`: Interfața pentru filtrarea utilizatorilor.

- `ViewImages`: Vizualizarea imaginilor operelor de artă.

Această structură modulară și funcționalitățile bogate asigură o gestionare eficientă și ușoară a informațiilor legate de operele de artă și utilizatorii muzeului.

În continuare vom lua pe pachete si vom descrie codul:

*Pachetul Model:*

Pachetul Model din proiectul MuseumApp cuprinde entitățile de bază ale aplicației, reprezentând structura de date și logica de afaceri. Clasa `ArtWork` reprezintă o operă de artă din cadrul muzeului, având proprietăți precum titlu, artist, anul creării și tipul operei de artă, fie că este vorba de pictură, sculptură sau altă formă de artă. Această clasă funcționează ca un DTO (Data Transfer Object), facilitând transferul de informații între straturile aplicației și baza de date.

Clasele `MuseumAdmin`, `MuseumEmployee` și `MuseumVisitor` extind clasa `User`, oferind roluri și permisiuni specifice în cadrul aplicației. De exemplu, un `MuseumAdmin` ar putea avea drepturi depline pentru a gestiona toate datele din sistem, în timp ce un `MuseumEmployee` ar putea avea acces limitat, concentrându-se pe operațiunile zilnice ale muzeului. `MuseumVisitor` ar putea reprezenta un utilizator al aplicației care nu necesită credențiale de autentificare, având acces doar la funcționalități de bază, cum ar fi vizualizarea operelor de artă.

Clasa `User` este o clasă abstractă de bază pentru toate tipurile de utilizatori ai aplicației, gestionând informații comune precum numele de utilizator, parola și tipul utilizatorului. Într-un mediu de producție real, ar trebui să implementăm măsuri de securitate robuste, deoarece în acest cod de exemplu, parolele sunt gestionate ca șiruri de caractere în text clar, ceea ce nu este sigur.

În esență, pachetul Model oferă o fundație solidă pentru reprezentarea datelor în cadrul aplicației, asigurând că toate informațiile esențiale sunt încapsulate și pot fi gestionate într-un mod care respectă principiile SOLID de programare orientată pe obiecte. Aceste clase permit o manipulare ușoară a datelor și oferă o punte între interfața utilizatorului, logica de afaceri și persistența datelor.

*Pachetul Controller:*

Pachetul `Controller` din aplicație joacă un rol crucial în arhitectura Model-View-Controller (MVC). Acesta include următoarele clase:

1. AdminController: Gestionează funcționalitățile specifice administratorilor, inclusiv operațiile CRUD pentru utilizatori și vizualizarea listei de utilizatori.

2. ArtWorkController: Responsabil pentru gestionarea operelor de artă, inclusiv adăugarea, actualizarea, ștergerea și căutarea acestora. De asemenea, gestionează vizualizarea statisticilor legate de operele de artă.

3. EmployeeController: Administrează operațiile pe care angajații le pot efectua după autentificare, cum ar fi gestionarea operelor de artă și generarea de rapoarte și grafice.

4. LogInController: Controlează procesul de autentificare pentru utilizatori, verificând credențialele și determinând nivelul de acces al utilizatorilor.

5. MainScreenController: Controlează logica ecranului principal al aplicației, gestionând navigarea și interacțiunile inițiale ale utilizatorilor.

6. UsersFilterController: Se ocupă de filtrarea utilizatorilor în funcție de diferite criterii, cum ar fi tipul utilizatorului, pentru a facilita gestionarea eficientă a acestora.

Aceste controlere facilitează separarea logicii de afaceri de interfața utilizator, asigurând o gestionare modulară și extensibilă a aplicației. Ele mediatizează interacțiunile între model (date) și view (interfața grafică), asigurând actualizări coerente și o experiență utilizator optimizată.

*Pachetul Repo:*

Pachetul Repo este inima funcționalității de backend a aplicației MuseumApp, gestionând persistența datelor și comunicarea cu baza de date. Acest pachet conține clasele și interfețele necesare pentru a interoga și manipula datele referitoare la operele de artă și utilizatorii sistemului.

`ArtWorkRepository` este responsabil pentru operațiunile CRUD (Create, Read, Update, Delete) pentru obiectele ArtWork. Aici sunt definite metode precum `addArtwork`, care inserează o nouă operă de artă în baza de date, `getAllArtworks`, care recuperează toate operele de artă, sau `deleteArtwork`, care șterge o operă de artă specifică. Această clasă folosește `ConnectionBD` pentru a stabili conexiunea cu baza de date și pentru a executa interogările SQL necesare.

Clasa `ConnectionBD` este utilitarul care gestionează conexiunea cu baza de date, asigurându-se că toate tranzacțiile sunt executate într-un mod sigur și eficient. Aceasta poate folosi un pool de conexiuni pentru a optimiza accesul la resursele de rețea și la baza de date.

`DAOException` este o clasă de excepții care este aruncată când apar erori în operațiunile de baza de date, oferind informații detaliate despre problema întâmpinată pentru a facilita depanarea.

Interfețele `IArtWork` și `IUser` definesc contractele pentru operarea cu obiectele de tip ArtWork și User. Acestea sunt esențiale pentru a asigura decuplarea logică și flexibilitatea codului, permițând substituirea implementărilor pentru diferite surse de date sau pentru simularea comportamentului în testele unitare.

`UserRepository` îndeplinește un rol similar cu `ArtWorkRepository`, dar pentru entitățile de tip User. Acesta oferă metode pentru adăugarea, actualizarea, ștergerea și interogarea detaliilor utilizatorilor, precum și pentru autentificarea acestora în sistem.

În concluzie, pachetul Repo este crucial pentru asigurarea interacțiunii eficiente între aplicație și baza de date, encapsulând logica specifică de acces la date, oferind astfel o separare clară între straturile de logică de afaceri și persistență, ceea ce duce la o arhitectură de software solidă și ușor de întreținut.

*Pachetul View:*

Pachetul `View` din aplicație este responsabil pentru interfața grafică cu utilizatorul (GUI). Acesta include diverse componente UI (User Interface) care permit interacțiunea utilizatorilor cu sistemul. Fiecare clasă din acest pachet reprezintă un ecran sau un dialog specific al aplicației, iar fișierele `.form` asociate reprezintă descrieri vizuale pentru design-ul GUI în formatele suportate de IDE-uri precum IntelliJ IDEA.

1. AdminUI: Gestionarea interfeței grafice pentru administrator. Aceasta include componente pentru efectuarea operațiunilor CRUD pe utilizatori și pentru gestionarea altor funcționalități administrative.

2. ArtWorkListView: Vizualizarea listei operelor de artă. Permite utilizatorilor vizitatori și angajați să vizualizeze operele de artă, să le filtreze și să le caute după diverse criterii.

3.ArtworkStatisticsInelarView ArtworkStatisticsLinearView ArtworkStatisticsRadialV: Aceste componente sunt dedicate vizualizării statisticilor operelor de artă în diferite formate grafice. Ele oferă grafice de tip inelar, liniar și radial, permițând utilizatorilor să înțeleagă mai bine datele despre operele de artă expuse.

4. EmployeeUI: Interfața grafică pentru angajați. Permite gestionarea operelor de artă și alte funcționalități specifice angajaților, precum adăugarea, actualizarea și ștergerea operelor de artă, precum și generarea de rapoarte.

5. LogIn: Ecranul de autentificare pentru utilizatori. Permite utilizatorilor să se autentifice în aplicație folosind credențiale.

6. MainScreen: Ecranul principal al aplicației, de unde utilizatorii pot naviga către alte secțiuni ale aplicației, în funcție de rolul lor (vizitator, angajat, administrator).

7. UsersFilterUI: Permite administratorilor să filtreze lista utilizatorilor în funcție de diverse criterii, facilitând gestionarea utilizatorilor care necesită autentificare.

8. ViewImages: Vizualizarea imaginilor operelor de artă. Permite utilizatorilor să vizualizeze imagini asociate operelor de artă și să navigheze între ele.

Aceste clase și form-uri asigură interfața grafică și interacțiunea utilizatorilor cu funcționalitățile aplicației, respectând principiile arhitecturii MVC (Model-View-Controller). Pachetul `View` joacă un rol esențial în asigurarea unei experiențe utilizator fluide și intuitive, oferind acces ușor la funcționalitățile aplicației.

Aplicatia vizualizată arată ca și în figura 4:

O imagine care conține text, captură de ecran, Software multimedia, software

Descriere generată automat

*Figura 4. Rularea aplicației și vizualizarea fiecărei ferestre.*

7. Concluzii:

Acest proiect MVC (Model-View-Controller) pentru muzeu demonstrează o arhitectură bine structurată și modulară, facilitând extinderea și întreținerea codului. Proiectul include trei tipuri de utilizatori: vizitatori, angajați și administratori, fiecare având acces la funcționalități specifice. Vizitatorii pot vizualiza și filtra operele de artă fără autentificare, angajații pot gestiona CRUD operele de artă și genera diverse formate de fișiere, iar administratorii pot gestiona utilizatorii autentificați.

Pachetul `Controller` coordonează logica aplicației, pachetul `Model` definește structura datelor, iar pachetul `View` gestionează interfața grafică și interacțiunea cu utilizatorii. Pachetul `Languages` asigură suportul multilingv, permițând schimbarea limbii interfeței grafice. Diagrama UML și structura proiectului reflectă o clară separare a responsabilităților și o interacțiune eficientă între componente, respectând principiile MVC.

Concluzionând, proiectul oferă o soluție completă și extensibilă pentru gestionarea muzeelor, facilitând atât experiența vizitatorilor, cât și operativitatea angajaților și administratorilor, menținând în același timp o arhitectură clară și bine definită.