**STUDIUL PATRULATERELOR**

**Proiect realizat de: Salajan Madalina Maria**

**Grupa: 30235**

1. **Enuntul problemei**

Dezvoltați o aplicație care poate fi utilizată ca soft educațional pentru studiul geometriei patrulaterului. Aplicația va avea 2 tipuri de utilizatori: elev și administrator.

Utilizatorii de tip elev pot efectua următoarele operații fără autentificare:

❖desenarea interactivă a patrulaterelor prin înlocuirea creionului și a riglei cu mouse-ul și alegerea stilului de desenare (inclusiv culoarea);

❖verificarea și afișarea unor caracteristici: patrulater convex/concav, patrulater înscriptibil, patrulater circumscriptibil;

❖calcularea și afișarea unor proprietăți: lungimile laturilor, măsurile unghiurilor, perimetrul, aria, raza cercului înscris (dacă patrulaterul este circumscriptibil), raza cercului circumscris (dacă patrulaterul este inscriptibil);

❖vizualizarea unor elemente specifice unui patrulater:

➢ceruri speciale: cercul circumscris, cercul înscris ;

❖solicitarea unui cont pentru testarea cunoștințelor.

Utilizatorii de tip elev pot efectua următoarele operații după autentificare:

❖verificarea cunoștințelor prin efectuarea unui test de 10 întrebări (alese aleator dintr-un set de 50 de întrebări)și vizualizarea punctajului obținut după finalizarea testului;

Utilizatorii de tip administratorpot efectua următoarele operații după autentificare:

❖Operații CRUD pentru informațiile legate de utilizatorii aplicației care necesită autentificare;

❖Vizualizarea listei tuturor utilizatorilor care necesită autentificare.

1. **Instrumente utilizate**

Urmatoarele instrumente au fost utilizate:

* Java: limbajul de programare principal folosit în clasa prezentată
* JUnit: un cadru de testare unitară pentru Java, utilizat în clasa UserTest
* TestNG: un alt cadru de testare unitară pentru Java, utilizat în clasa ConnectionFactoryTest
* JDBC (Java Database Connectivity): un API Java pentru accesul la baze de date relaționale, utilizat în clasa ConnectionFactory și în clasele DAO (Data Access Object) pentru a efectua operațiuni CRUD pe baza de date
* MySQL: un sistem de gestionare a bazelor de date relaționale, utilizat ca baza de date pentru aceste exemple.

1. **Justificarea limbajului de programare ales**

Printre motivele pentru care am ales sa realizez acest proiect in Java se numara urmatoarele:

* Java este un limbaj de programare popular și larg utilizat în industrie, astfel încât o astfel de aplicație ar putea fi ușor integrată în proiecte mai mari sau utilizată în contexte profesionale.
* Java oferă o serie de funcționalități și librării care au fost utile în realizarea proiectulului meu, cum ar fi MouseListener-ul folosit pentru desenarea formelor geometrice și biblioteci pentru efectuarea de calcule matematice complexe.
* Studiul patrulaterelor este un subiect interesant și util în matematică și poate fi util pentru elevi, studenți și profesioniști care lucrează în domenii care implică geometrie sau fizică, iar realizarea acetui proiect in Java, care este un limbaj cunoscut de o foarte mare parte din programatori, ajuta la cresterea gradului de intelegere a proiectului si a modului de implementare.
* Proiectul realizat in Java nu a fost o noutate deoarece in timpul anilor universitari precedenti s-a pus accentul pe studierea acestui limbaj, lucru ce a usurat munca depusa.
* Tool-ul Java Swing este usor de folosit si ofera multa flexibilitate in realizarea GUI-ului.

În final, alegerea de a realiza un proiect în Java care să se ocupe cu studiul patrulaterelor a depins de interesele și obiectivele mele personale. Dacă ești interesat de matematică și programare și dorești să îți îmbunătățești abilitățile în aceste domenii, acest proiect ar putea fi o alegere excelentă pentru tine.

1. **Descrierea diagramei UML**

Diagrama UML contine in plus, pe langa clase, si relatiile dintre acestea. Clasele UserDAO, TestDAO, IntrebareDAO extind clasa AbstractDAO.

Fiecare clasa din pachetul View extinde interfata corespunzatoare si fiecare tip de presenter din pachetul Presenter este atribuit unui View din pachetul View pentru a compune arhitectura MVP.

Clasele Patrulater, Cerc mostenesc clasa FiguraGeometrica. Clasele Punct, Dreapta, FiguraGeometrica extind clasa ElementGeometric. Clasa Desen contine o lista de ElementGeometric.

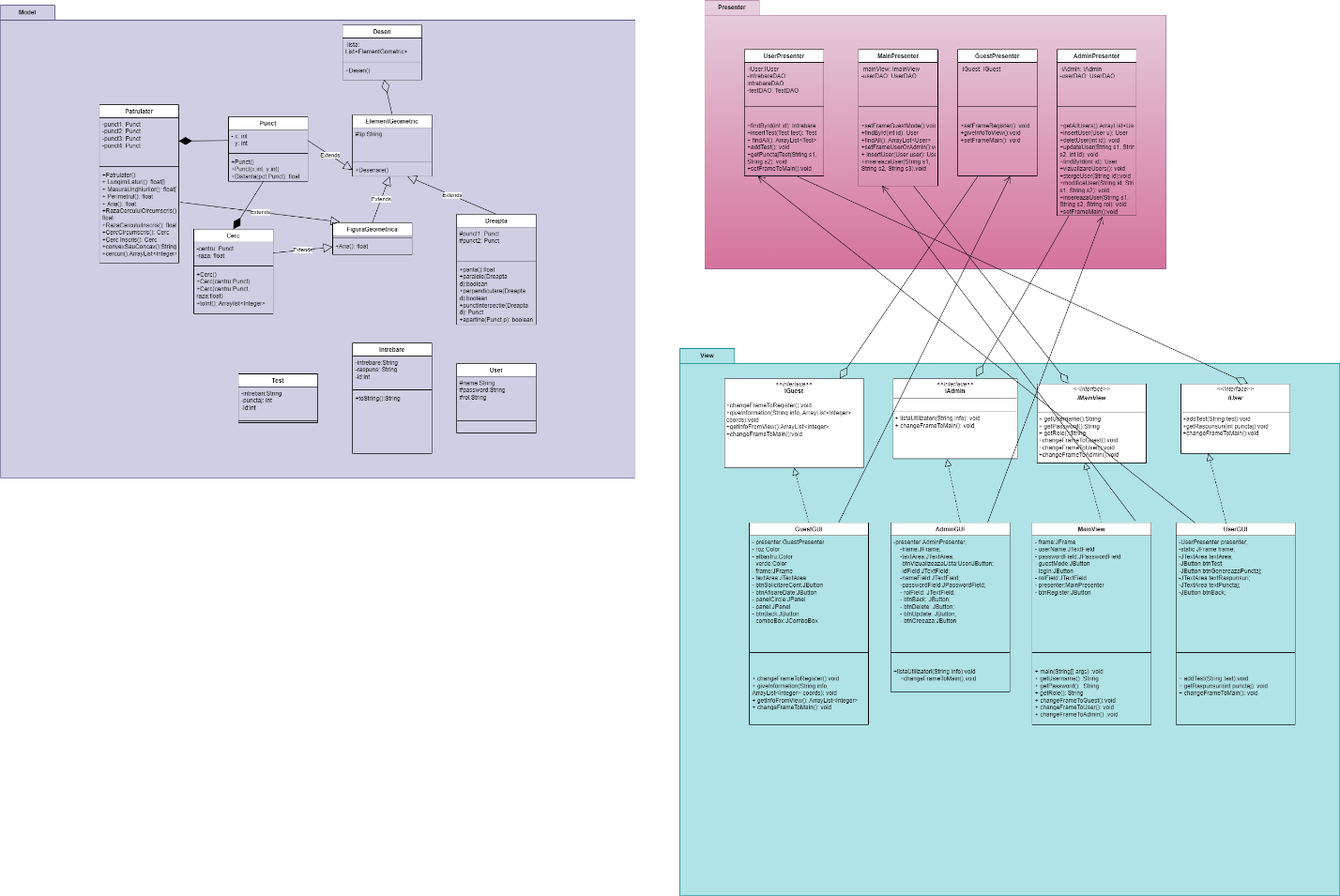
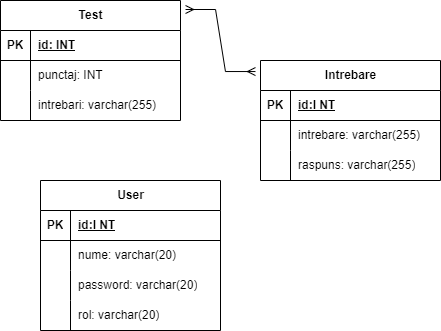
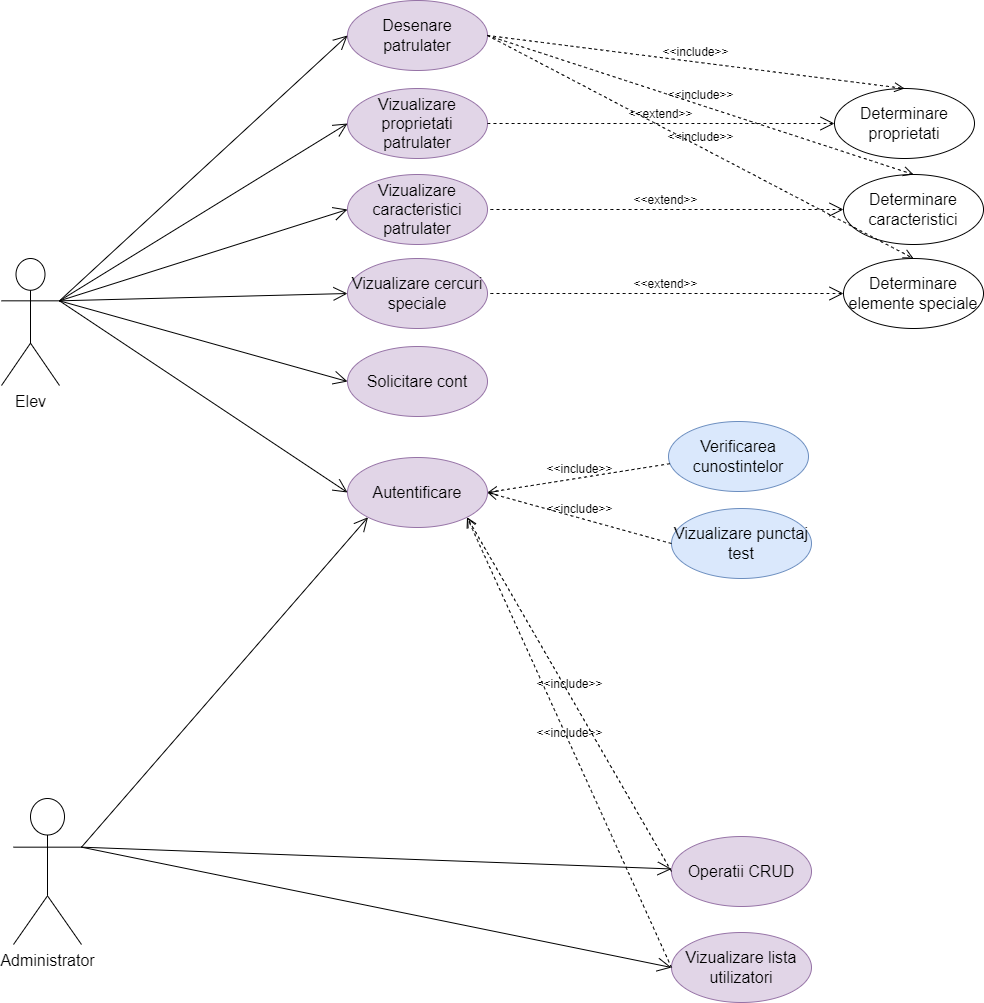


Diagrama entitate-relatie contine 3 tabele: Tabela Test poate avea mai multe intrebari si o intrebare poate aparea in mai mulet teste.



Am adaugat si diagrama cazurilor de utilizare pentru aplicatia mea.



1. **Descrierea aplicatiei**

Aplicatia este alcatuita din 3 pachete si are o arhitectura de tip MVP.

**MVP (Model-View-Presenter)** este o arhitectură software utilizată în dezvoltarea de aplicații cu interfață grafică (GUI). Arhitectura MVP separă aplicația în trei componente principale: Model, View și Presenter.

* Model: reprezintă datele și logica de business a aplicației. În general, modelul este format dintr-o serie de clase care gestionează datele și oferă metode pentru a le accesa și manipula.
* View: este componenta vizuală a aplicației și este responsabilă pentru prezentarea datelor utilizatorului. View-ul poate fi format dintr-o interfață grafică (GUI), compusă din elemente precum butoane, meniuri, liste, etc.
* Presenter: este responsabil pentru interacțiunea dintre Model și View. Presenter-ul primește datele din Model, le procesează și le transmite View-ului pentru a fi afișate. De asemenea, Presenter-ul primește evenimente de la View și le transmite către Model pentru a fi prelucrate.

Arhitectura MVP se concentrează pe separarea clară a responsabilităților între cele trei componente, ceea ce face aplicația mai modulară și mai ușor de întreținut. MVP este o arhitectură foarte populară pentru dezvoltarea de aplicații cu interfață grafică (GUI), cum ar fi aplicații mobile sau desktop.

**Baze de date relationale**

O bază de date relațională este un tip de bază de date care stochează și oferă acces la punctele de date care sunt legate între ele. [Bazele de date](https://www.oracle.com/ro/database/what-is-database/) relaționale se bazează pe modelul relațional, o modalitate intuitivă și simplă de reprezentare a datelor în tabele. Într-o bază de date relațională, fiecare rând din tabel este o înregistrare cu un cod unic numit cheie. Coloanele din tabel au atribute ale datelor și fiecare înregistrare are, de obicei, o valoare pentru fiecare atribut, facilitând stabilirea relațiilor dintre punctele de date.

**Reflection**

Reflection este o caracteristică a limbajului de programare Java. Permite unui program Java care se execută să examineze sau să „introspecteze” asupra lui însuși și să manipuleze proprietățile interne ale programului. De exemplu, este posibil ca o clasă Java să obțină numele tuturor membrilor săi și să le afișeze.

Capacitatea de a examina și de a manipula o clasă Java din interior poate să nu spuna prea mult, dar în alte limbaje de programare această caracteristică pur și simplu nu există. De exemplu, nu există nicio modalitate într-un program Pascal, C sau C++ de a obține informații despre funcțiile definite în acel program.

Pachetul **Model** contine 15 clase.

Pachetul **DAO** inclus in pachetul Model contine clasa **AbstractDAO**. Această clasă reprezintă un schelet general pentru orice clasă de acces la date, care are metode generice pentru a efectua operații CRUD (create, read, update, delete) într-o bază de date. Clasa include metode pentru a obține toate înregistrările unei tabele, a găsi o înregistrare după id-ul său, a actualiza o înregistrare și a insera o nouă înregistrare. Aceasta utilizează o conexiune la baza de date, specificată prin clasa ConnectionFactory. Clasa utilizează reflexia pentru a crea obiecte, ceea ce permite ca clasele DAO derivate să poată manipula diferite tipuri de obiecte. Clasele IntrebareDAO, TestDAO si UserDAO extind acesta clasa.

Clasa **ConnectionFactory** face parte dintr-un pachet numit DatabaseConnection care se ocupă de conectarea la o bază de date MySQL folosind JDBC. Aceasta conține metode pentru a obține o conexiune la baza de date, a închide conexiunea, declarațiile și obiectele ResultSet. Metoda createConnection() se ocupă de crearea conexiunii la baza de date și este apelată de metoda getConnection() pentru a obține o conexiune. Aceasta este o clasă utilitară utilă pentru orice aplicație care utilizează o bază de date și ajută la menținerea conexiunii cu baza de date într-un mod eficient și organizat.

Clasele Punct, Dreapta, Patrulater, Cerc, Element Geometric si FiguraGeometrica sunt folosite pentru a modela functionalitatile acestui proiect, mai exact date legate de patrulatere. Pentru a realiza toate calculele, miezul proiectului este in clasa Patrulater.

Clasa **Patrulater** este o subclasa a clasei FiguraGeometrica si este folosita pentru a crea obiecte care reprezinta patrulatere, prin intermediul punctelor corespunzatoare. Aceasta clasa contine un constructor fara parametri si un constructor cu patru parametri, care primesc patru puncte pentru a forma un patrulater. De asemenea, contine un constructor care primeste un ArrayList de coordonate (x, y) si creeaza patrulaterul corespunzator. Aceasta clasa contine metode care calculeaza lungimile laturilor, masurile unghiurilor, perimetrul, aria, raza cercului circumscris si raza cercului inscris al patrulaterului.

Clasa **Test** contine un id unic, punctajul aferent unui test, dar si id-urile intrebarilor continute de test, astfel se face legatura cu clasa **Intrebare** care contine un id unic, intrebarea propriu-zisa, precum si raspunsul.

Clasa **User** este o clasă de modelare a unui utilizator și are rolul de a stoca informațiile referitoare la acesta, cum ar fi numele, id-ul, parola și rolul (ELEV sau ADMIN). De asemenea, clasa include și metode pentru accesarea și modificarea acestor informații. Această clasă poate fi folosită în cadrul unei aplicații care necesită autentificarea utilizatorilor.

In pachetul **View** regasim 4 clase si 4 interfete pentru a respecta principiile arhitecturii MVP. Fiecare clasa implementeaza o interfata din cele prezente. Fiecare clasa din acest pachet mosteneste JFrame pentru a realiza GUI si fiecare clasa contine un presenter. Printre atributele claselor se numara butoanele, textArea-urile, field-urile, comboBox-urile, panel-uri si elementul care le cuprinde pe toate, frame-ul. Butoanele au adaugate MouseListener-uri care, prin presenter, trimit date care vor fi procesate in presenter, iar informatiile procesate vor fi ulterior trimise inapoi catre View prin interfetele implementate.

In pachetul **Presenter** regasim 4 clase, adica 4 presenteri, fiecare gestioneaza actiunile utilizatilor din View, astfel este facuta legatura cu Modelul.

Clasa **AdminPresenter** este o clasă Java care face legătura între clasa de interfață grafică a utilizatorului IAdmin și clasa de acces la baza de date UserDAO, care interacționează cu baza de date pentru a realiza operații CRUD asupra obiectelor User. Această clasă implementează metode pentru a obține toți utilizatorii din baza de date, pentru a insera, actualiza și șterge un utilizator în baza de date, precum și pentru a vizualiza utilizatorii și a efectua operațiuni de modificare și inserare a acestora în interfața grafică a utilizatorului. De asemenea, AdminPresenter are metode pentru a schimba fereastra de interfață grafică de la ecranul de administrare la ecranul principal.

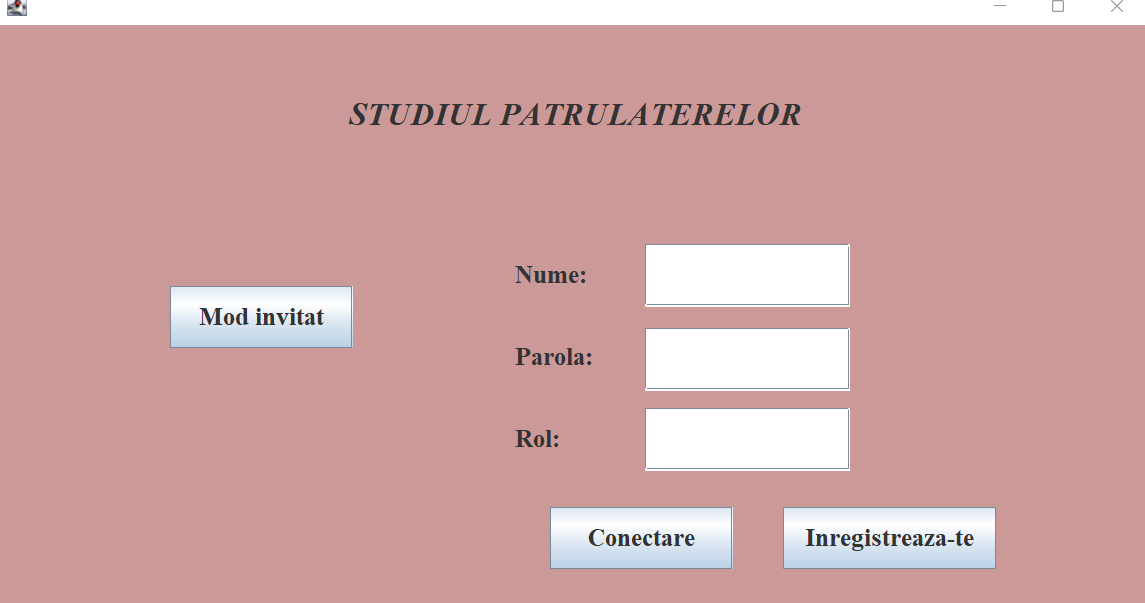
Am implementat o serie de teste pentru baza de date, dar si pentru operatiile relizate pe aceasta.

Clasa **ConnectionFactoryTest** este o clasă de testare JUnit care testează metodele din clasa ConnectionFactory, care este responsabilă de crearea și închiderea conexiunilor la baza de date. Clasa de test conține trei metode de testare, care verifică dacă conexiunile, declarațiile și rezultatele sunt închise corect și dacă nu apar excepții în timpul închiderii acestora. Clasa de testare folosește aserțiuni JUnit pentru a verifica dacă valorile sunt nule și dacă excepțiile sunt aruncate atunci când se așteaptă.

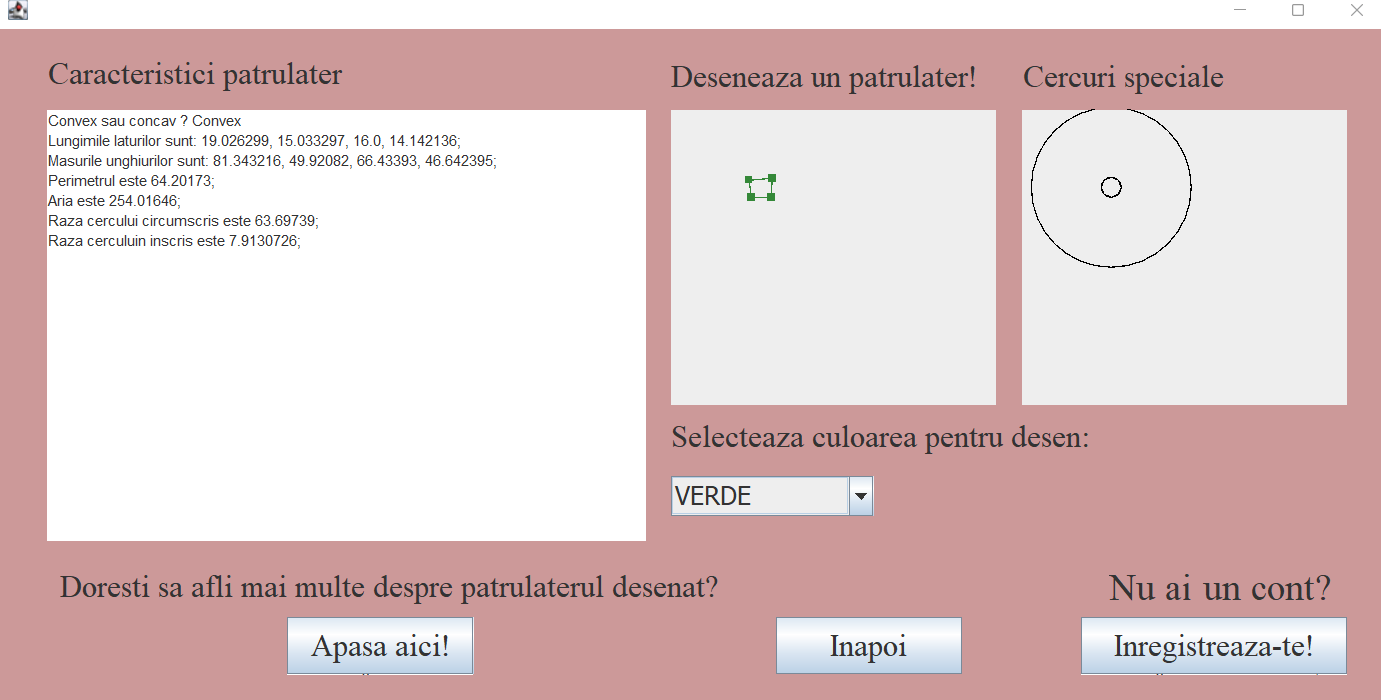
Clasa **UserTest** conține o suită de teste unitare pentru clasa User și pentru operațiile CRUD (create, read, update, delete) din clasa UserDAO. Urmatoarele teste au fost implementate:

* În metoda setUp() se instantiază un obiect UserDAO pentru a fi folosit în toate testele.
* Metoda testInsert() testează operația de inserare a unui nou utilizator în baza de date. Se creează un obiect User, se inserează în baza de date, se extrage din nou obiectul din baza de date și se compară cu obiectul original. Se verifică dacă datele inserate corespund cu cele așteptate.
* Metoda testFindByField() testează operația de găsire a unui utilizator din baza de date în funcție de id. Se extrage un obiect User din baza de date folosind metoda findById(), se compară câmpurile obiectului cu valorile așteptate.
* Metoda testUpdate() testează operația de actualizare a unui utilizator din baza de date. Se extrage un obiect User din baza de date, se actualizează unul dintre câmpurile acestuia, se salvează schimbările în baza de date, se extrage din nou obiectul și se compară dacă valoarea schimbată a fost actualizată.
* Metoda testDelete() testează operația de ștergere a unui utilizator din baza de date. Se extrage un obiect User din baza de date, se șterge acesta din baza de date, se încerce extragerea din nou a obiectului și se așteapt

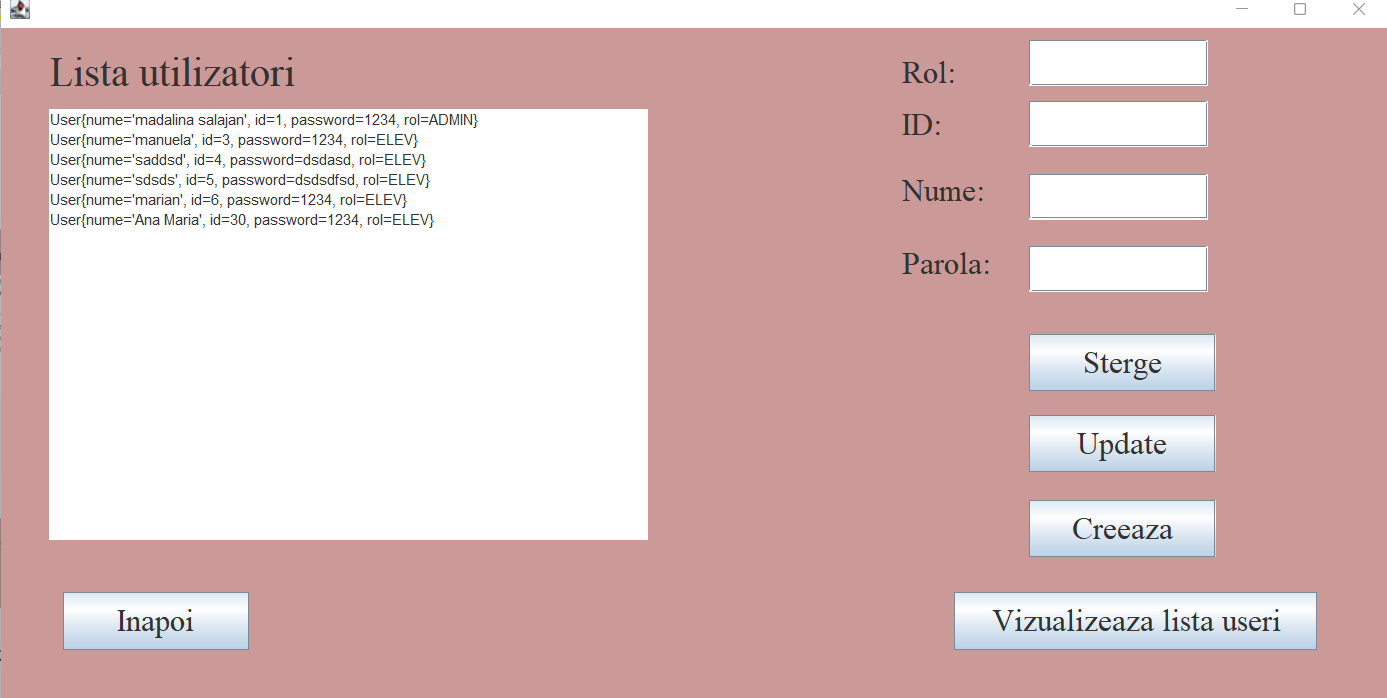
Aceasta este interfata grafica la pornirea aplicatiei. Poti alege sa te conectezi, sa iti creezi un cont sau sa mergi pe modul de invitat.



Daca este apasat butonul de mod invitat, urmatoarea interfata grafica se deschide. Poti selecta culoarea cu care sa desenezi, sa desenezi si sa afli mai multe despre patrulaterul tau. Poti solicita un cont sau sa mergi inapoi in meniul principal.



Daca din meniul principal te conectezi ca un admin, urmatoarea interfata grafica este deschisa. Ca si administrator poti alege sa vizualizezi lista de utilizatori, sa inserezi, sa stergi si sa dai update la utilizatorii ce detin un cont.



Daca din meniul principal iti creezi un cont si te conectezi, urmatoarea interfata grafica se va deschide. Poti alege sa dai un test si sa iti vizualizezi punctajul.

