**STUDIUL PATRULATERELOR**

**MVC**

**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:

➢ puncte importante într-un patrulater: punctul lui Newton al unui patrulater circumscriptibil, punctul lui Miquel al unui patrulater convex, punctul lui Mathot al unui patrulater inscriptibil;

➢ linii importante într-un patrulater convex: diagonale, bimediane, bisectoare, dreapta lui Newton corespunzătoare unui patrulater circumscriptibil, dreapta lui Gauss corespunzătoare unui patrulater complet, dreapta lui Aubert corespunzătoare unui patrulater complet;

➢ ceruri speciale: cercul circumscris, cercul înscris;

❖ salvarea / încărcare unui patrulater într-un/dintr-un fișier xml;

❖ 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;

❖ vizualizarea unei statistici după punctaj utilizând grafice (structură radială, structură inelară, etc.).

Utilizatorii de tip administrator pot efectua următoarele operații după autentificare:

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

❖ Vizualizarea listei tuturor utilizatorilor care necesită autentificare.

Interfața grafică a aplicației va fi disponibilă în cel puțin 3 limbi de circulație internațională.

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.
* Biblioteca XStream este o bibliotecă Java open-source, care oferă suport pentru serializarea și deserializarea obiectelor Java în format XML. XStream poate fi utilizată pentru a converti obiecte Java în șiruri XML și viceversa.
* Lombok este o bibliotecă Java open-source care are ca scop reducerea cantității de cod boilerplate (cod redundant, care se repetă frecvent) din proiectele Java, prin generarea automată de cod getter, setter, constructori și alte metode comune. Această bibliotecă oferă o serie de anotări speciale care pot fi utilizate pentru a genera automat codul necesar. De exemplu, anotarea @Getter poate fi aplicată la nivel de clasă sau de câmpuri pentru a genera automat metodele getter pentru acestea. Anotarea @Setter poate fi folosită pentru a genera metodele setter. În plus, Lombok oferă și alte anotări utile, precum @AllArgsConstructor pentru generarea unui constructor cu toate câmpurile, @NoArgsConstructor pentru generarea unui constructor fără argumente, @ToString pentru generarea automată a metodei toString() și multe altele. Utilizarea bibliotecii Lombok poate reduce semnificativ cantitatea de cod necesar pentru a crea modelele de date și alte clase, astfel încât programatorii pot să se concentreze mai mult pe logica de afaceri decât pe codul boilerplate. Lombok este ușor de instalat și utilizat și este compatibil cu cele mai populare IDE-uri Java, inclusiv Eclipse, IntelliJ IDEA și NetBeans.

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 contine un atribut din pachetul ViewModel si fiecare clasa din View contine niste filed-uri care prin Bind au fost legate cu alte field-uri din clasele din ViewModel. Acest lucru a fost necesar pentru a se respecta arhitectura MVVM.

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

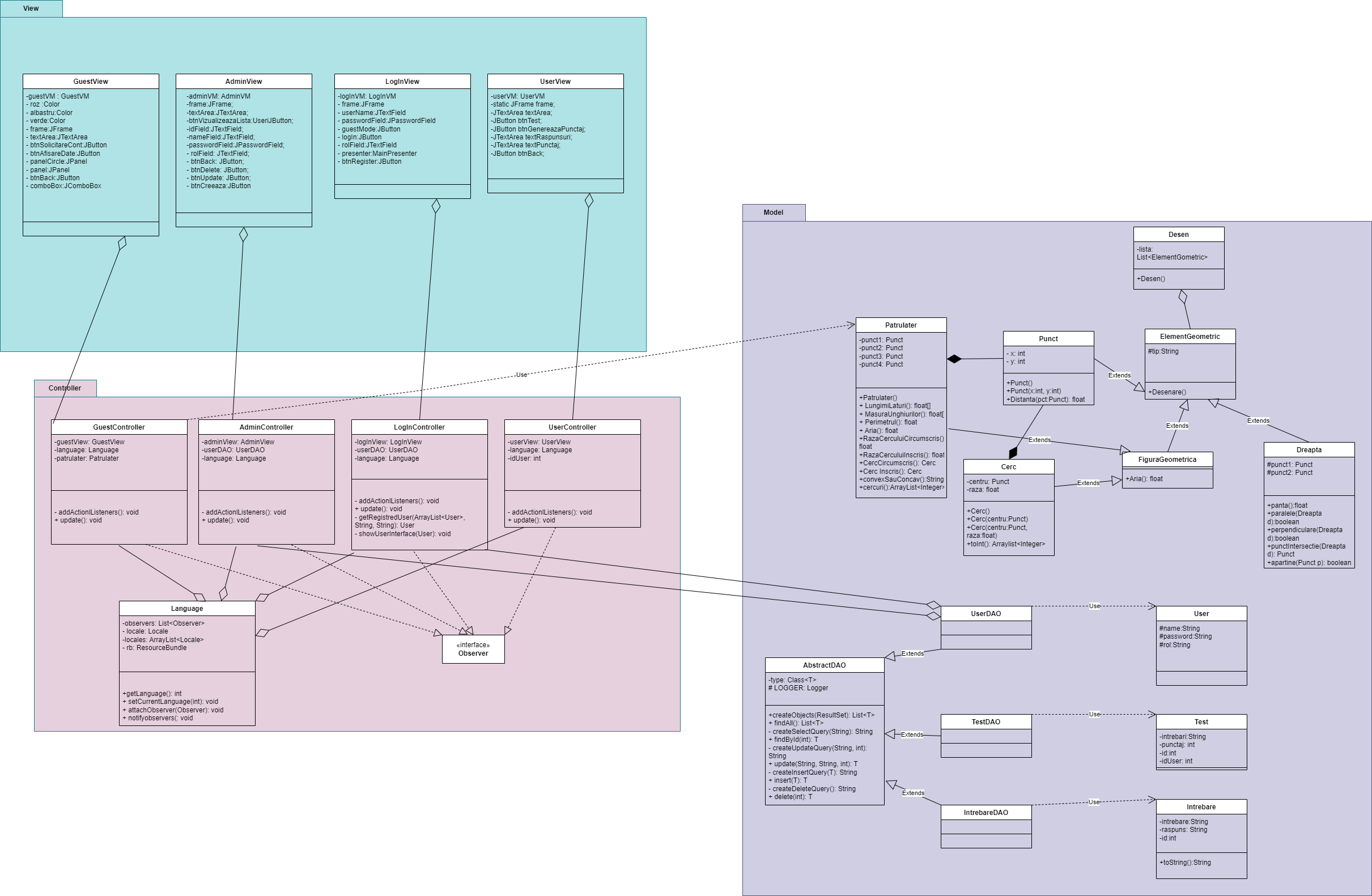
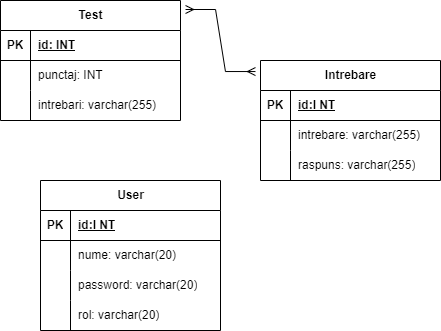
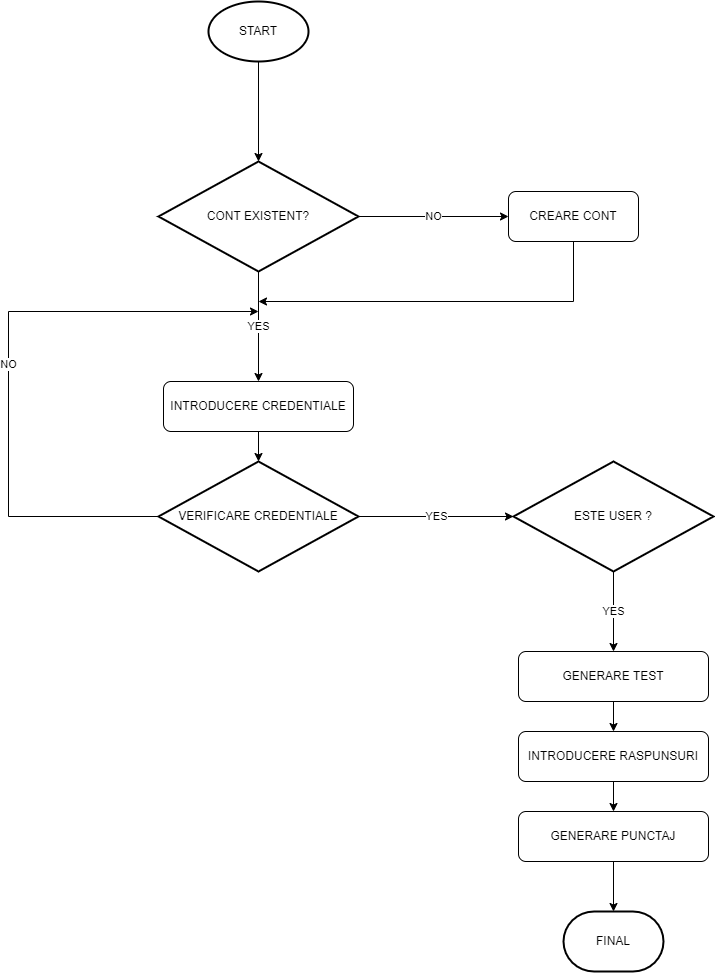


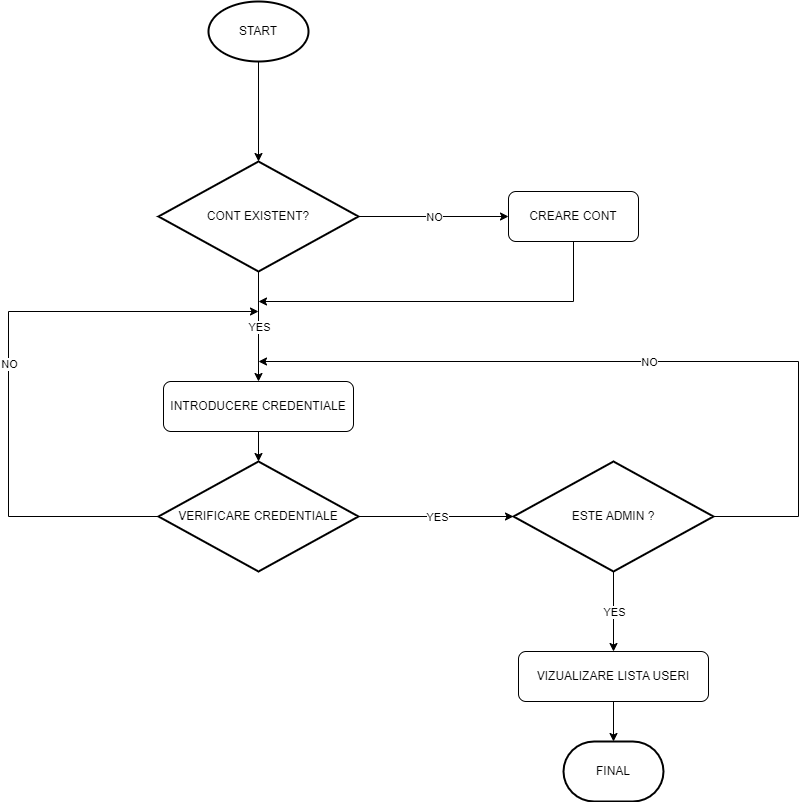
Diagrama entitate-relatie contine 3 tabele: Tabela Test poate avea mai multe intrebari si o intrebare poate aparea in mai mulet teste.



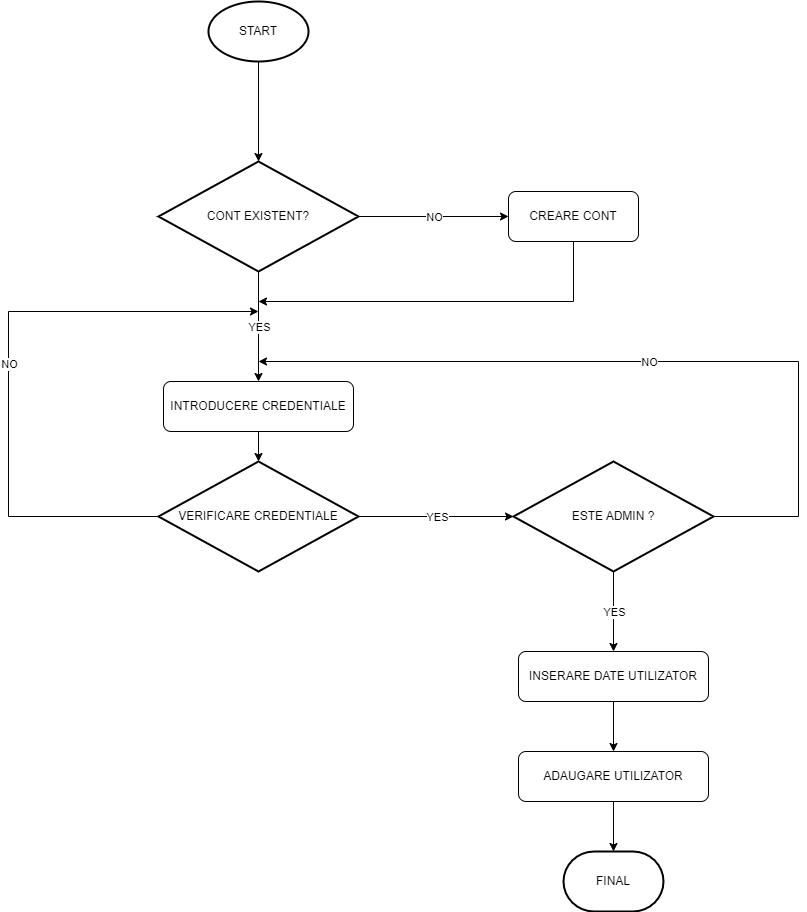
Mai jos am adaugat diagrama de activitate pentru un user care da un test si genereaza punctajul pentru testul respectiv.



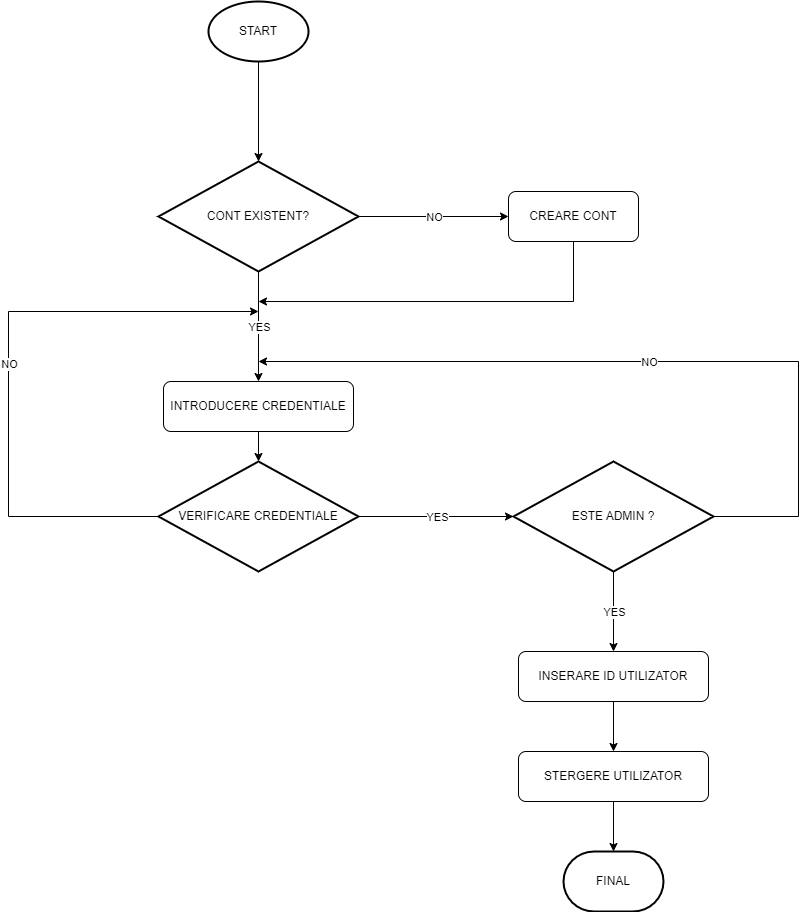
Mai jos am adaugat diagrama de activitate pentru un administartor care se conecteaza si care vizualizeaza lista cu toti userii.



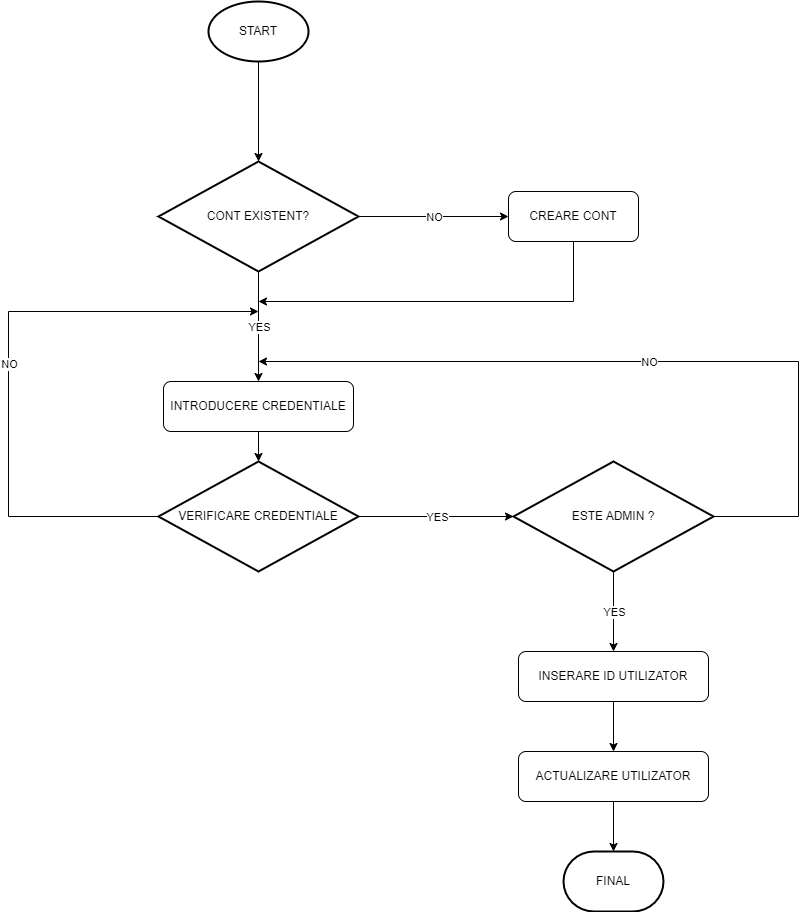
Mai jos am adaugat diagrama de activitate pentru un administartor care se conecteaza si care adauga un utilizator in baza de date.



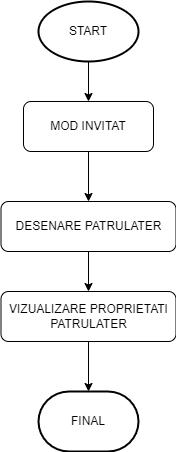
Mai jos am adaugat diagrama de activitate pentru un administartor care se conecteaza si care sterge un utilizator din baza de date.



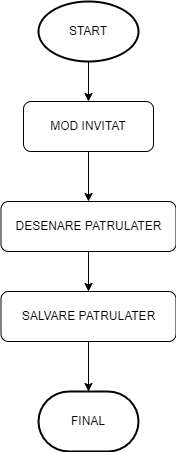
Mai jos am adaugat diagrama de activitate pentru un administartor care se conecteaza si care actualizeaza un utilizator din baza de date.



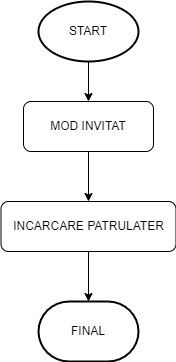
Mai jos am adaugat diagrama de activitate pentru un guest care deseneaza un patrulater si ii vizualizeaza proprietatile.



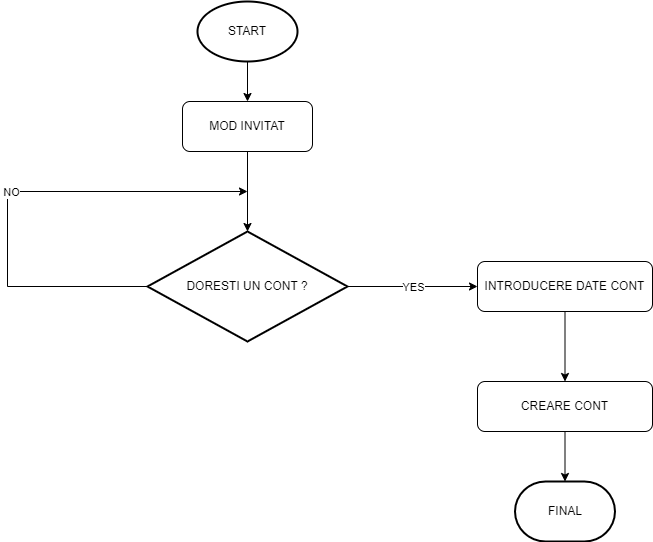
Mai jos am adaugat diagrama de activitate pentru un guest care deseneaza un patrulater si il salveaza sub forma unui fisier .xml.



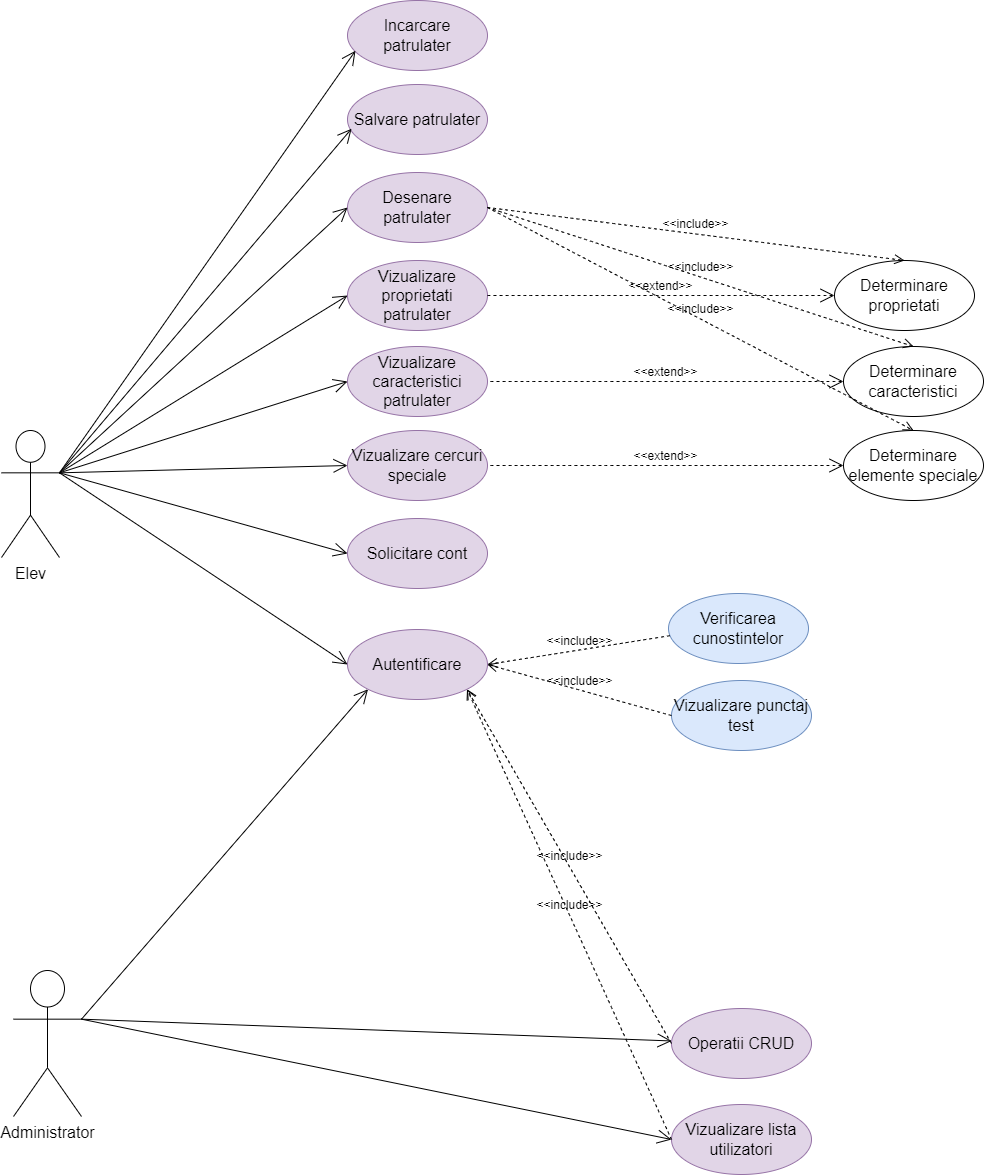
Mai jos am adaugat diagrama de activitate pentru un guest care incarca un patrulater dintr-un fisier .xml.



Mai jos am adaugat diagrama de activitate pentru un guest care doreste sa isi creeze un cont.



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 MVC.

**MVC (Model-View-Controller)** este o arhitectură de software folosită în dezvoltarea aplicațiilor web și desktop care împarte aplicația în trei componente principale: Model, View și Controller.

**Model**: Modelul reprezintă datele și logica de afaceri ale aplicației. Aceasta poate fi formată din obiecte, structuri de date, conexiuni la baze de date și orice alte surse de date utilizate în aplicație. Modelul nu cunoaște nimic despre celelalte două componente și nu are nicio legătură cu aspectul sau comportamentul aplicației.

**View**: Vizualizarea reprezintă partea vizuală a aplicației, adică ceea ce utilizatorul vede și interacționează cu ea. Aceasta poate fi compusă din interfețe grafice, pagini web sau orice altă formă de prezentare a datelor și informațiilor utilizatorului. Vizualizarea nu conține nicio logică de afaceri sau date, ci doar afișează datele furnizate de model.

**Controllerul**: Controlerul este componenta care conectează Modelul și Vizualizarea și gestionează interacțiunea utilizatorului cu aplicația. Acesta primește solicitări de la utilizator și își actualizează modelul și apoi actualizează vizualizarea cu datele actualizate. Controlerul conține și logica de afaceri și fluxul de control al aplicației.

Prin separarea logică a acestor componente, arhitectura MVC asigură o structură clară și organizată a aplicației. Separarea responsabilităților permite dezvoltatorilor să lucreze independent în fiecare componentă și să schimbe una fără a afecta celelalte, ceea ce face întreținerea și dezvoltarea aplicației mai ușoară și mai eficientă.

**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.

**Conversia intr-un fisier XML**

Unul dintre avantajele bibliotecii XStream este că oferă o soluție simplă și rapidă pentru serializarea obiectelor Java în fișiere XML, fără a fi nevoie să implementați propriul cod serializare și deserializare. În plus, XStream oferă o serie de funcții personalizabile, care vă permit să controlați modul în care obiectele sunt serializate și deserializate.

XStream folosește adnotări speciale (cum ar fi @XStreamAlias și @XStreamImplicit) pentru a configura modul în care obiectele Java sunt convertite în fișiere XML. Aceste adnotări oferă o mare flexibilitate și control asupra procesului de serializare și deserializare.

În general, XStream este o bibliotecă puternică și ușor de utilizat pentru serializarea și deserializarea obiectelor Java în fișiere XML. Datorită flexibilității și performanței sale, XStream este utilizată frecvent în cadrul proiectelor Java pentru a gestiona datele în format XML.

**Bundle/Swing Internationalization**

Aceasta(i18n) este o tehnică folosită în dezvoltarea de software pentru a face aplicațiile mai accesibile și mai ușor de utilizat pentru persoanele care vorbesc diferite limbi sau provin din culturi diferite. Implica procesul de adaptare a interfeței de utilizator și a conținutului unei aplicații la diferite locații și limbi, astfel încât aplicația să poată fi utilizată de oameni din întreaga lume. În Java, clasa ResourceBundle oferă o modalitate convenabilă de a implementa i18n. Un ResourceBundle este o modalitate simplă de a stoca șiruri de caractere și alte obiecte care sunt traduse în diferite limbi. Un pachet de resurse conține un set de perechi cheie-valoare în care cheile sunt identificatorii pentru șiruri, iar valorile sunt șirurile localizate.

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. Fiecare clasa din acest pachet mosteneste JFrame pentru a realiza GUI. Printre atributele claselor se numara butoanele, textArea-urile, field-urile, comboBox-urile, panel-uri si elementul care le cuprinde pe toate, frame-ul.

In pachetul **Controller** avem 5 clase si o interfata. Exista cate un controller pentru fiecare tip de utilizator. Interfata Observer este implementata de controllere si cu metoda update modificam limba aplicatiei. Clasa **Language** foloseste un obiect de tipul ResourceBundle pentru a prelua din fisierele .properties cuvintele in cele 3 limbi de circulatie internationala si notifica observerii de modificarea limbii. Clasa **AdminController** contine metoda addActionListeners() care adauga listeneri pentru fiecare buton din interfata grafica si pentru comboBox-ul cu care se selecteaza limba. Cu metoda update() se modifica limba aplicatiei.

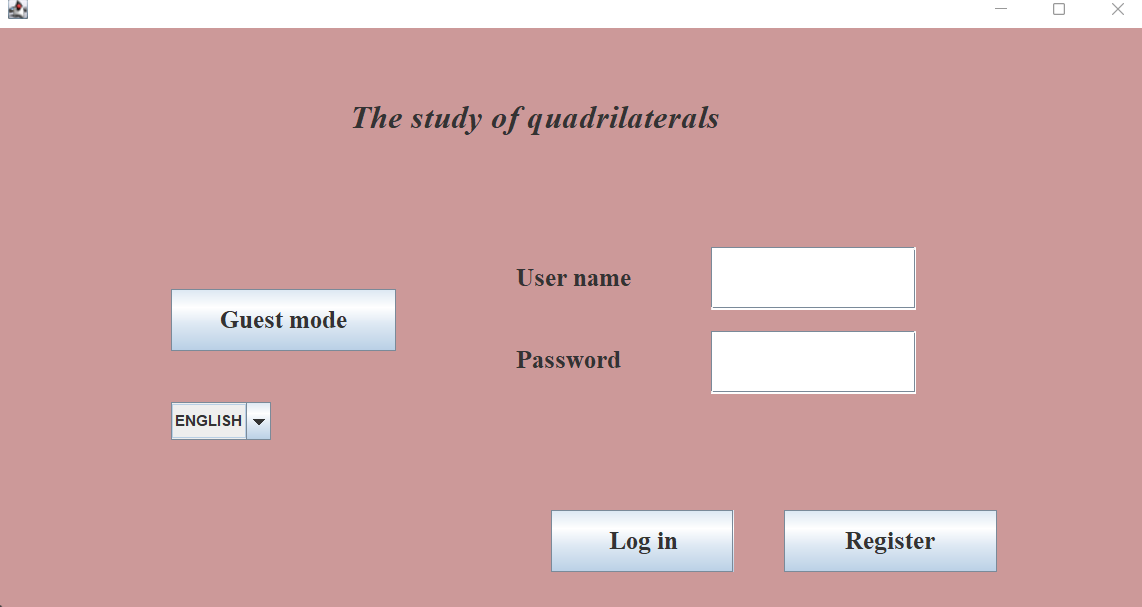
Am implementat o serie de teste pentru baza de date, dar si pentru operatiile relizate pe aceasta.

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.

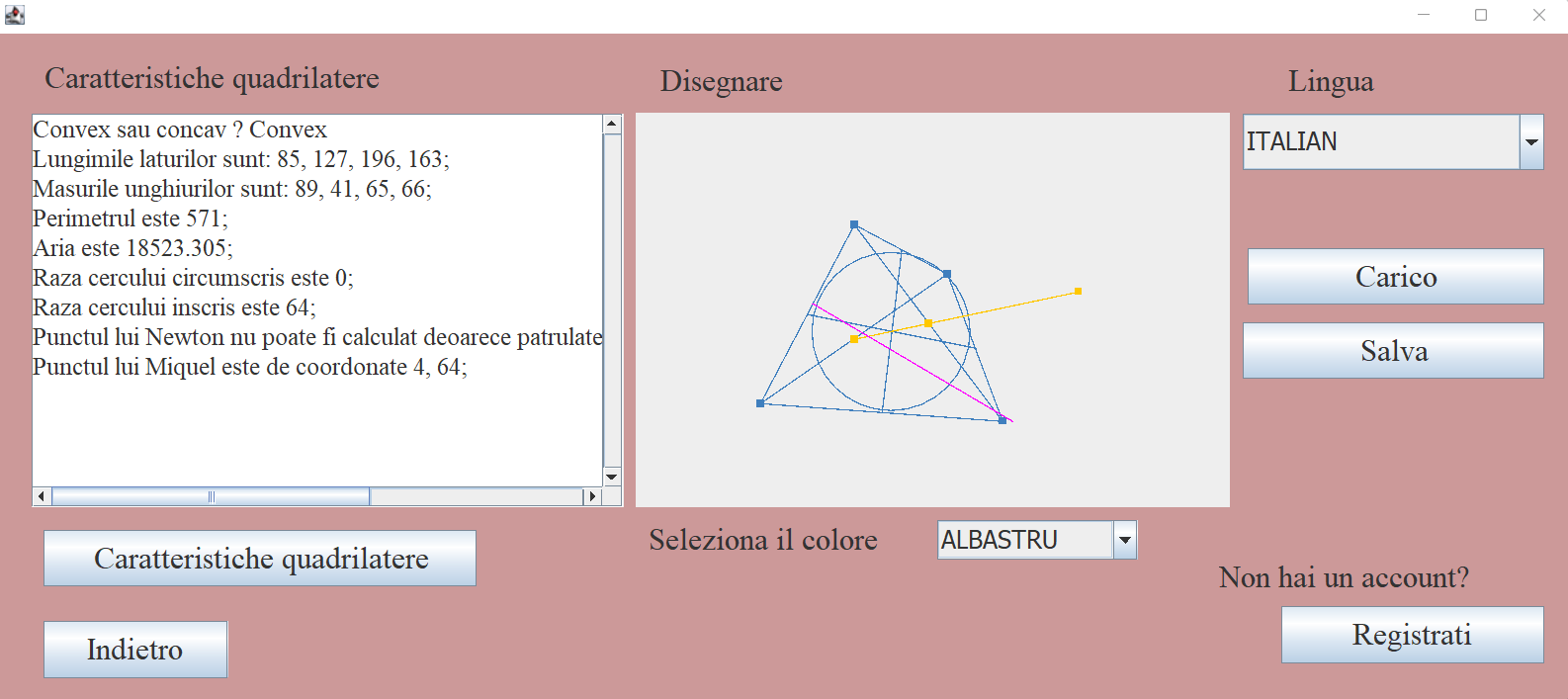
Clasa **UserTest** conține o suită de teste unitare pentru clasa User și pentru operațiile CRUD (create, read, update, delete) din clasa UserDAO. Urmatoarele teste au fost implementate:

* 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

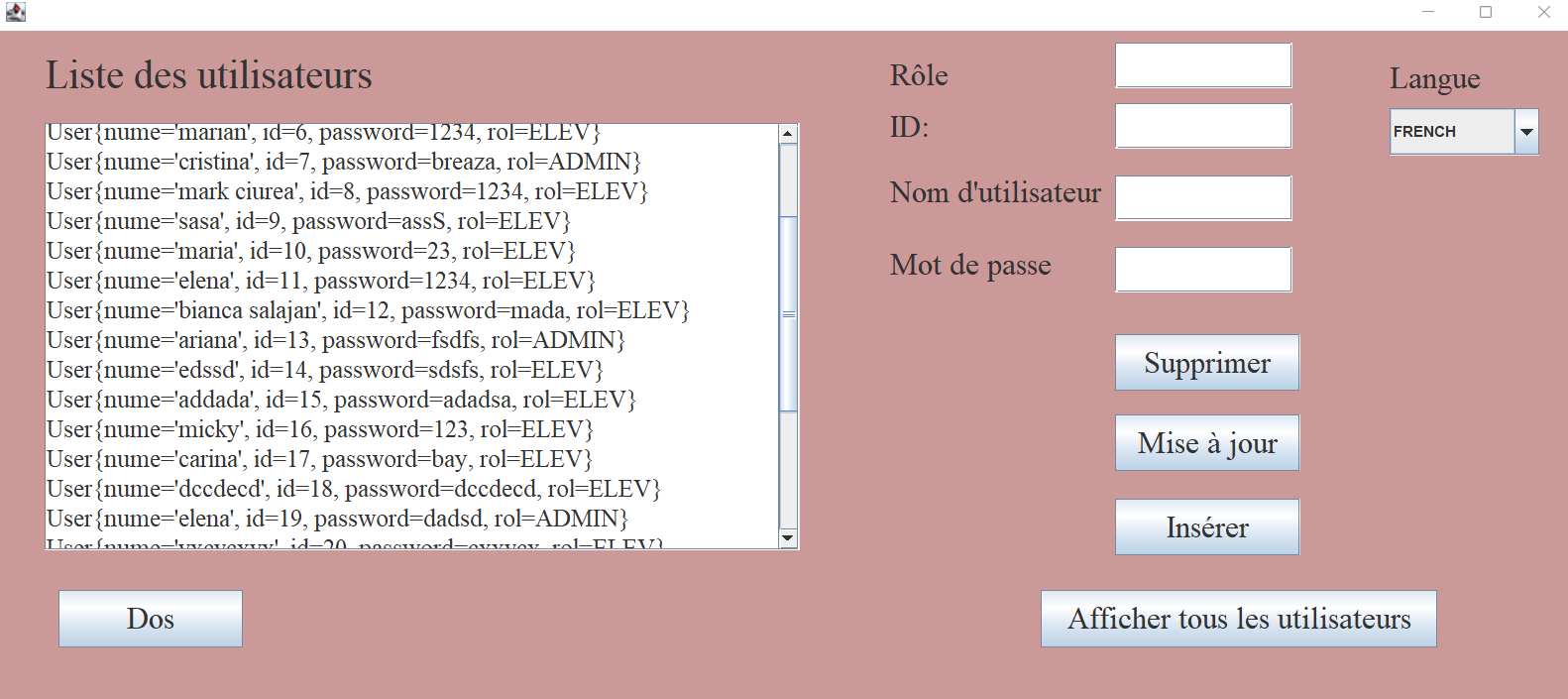
In imaginea de mai jos este interfata grafica la pornirea aplicatiei. Poti alege sa te conectezi, sa iti creezi un cont sau sa mergi pe modul de invitat.



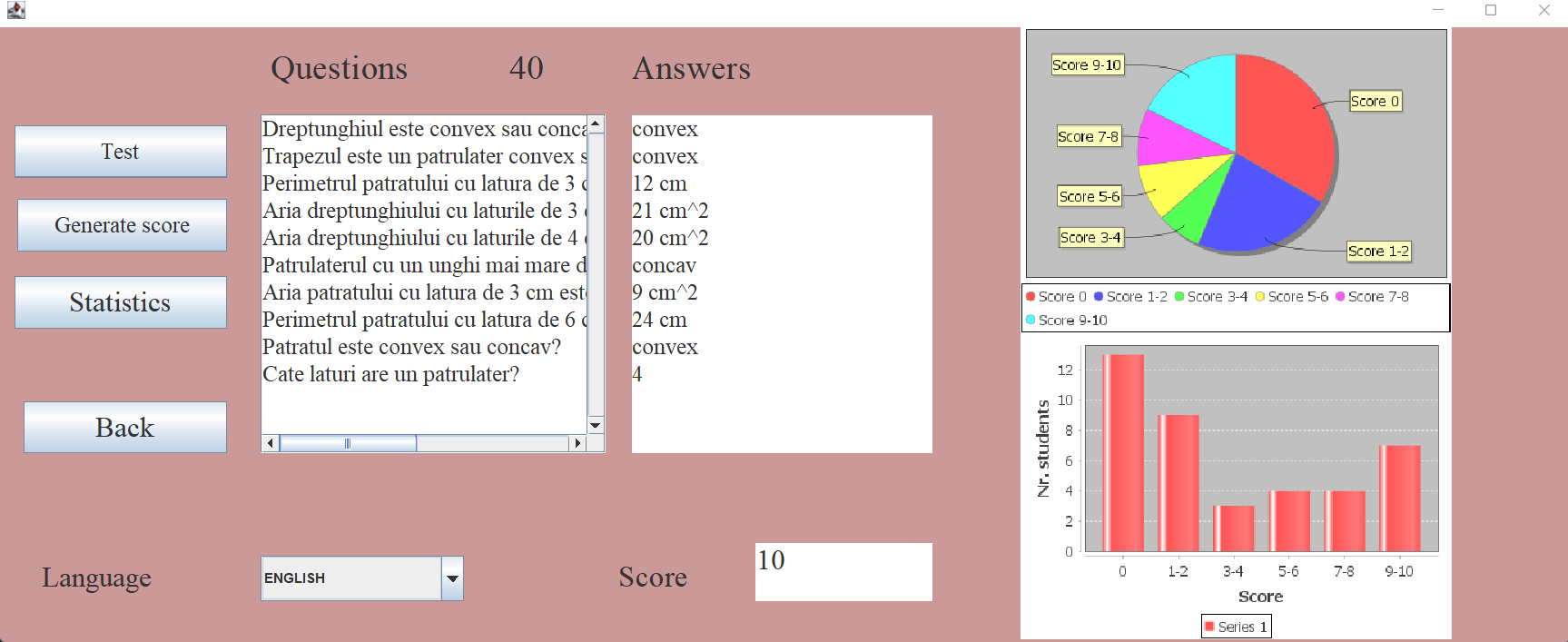
Daca este apasat butonul de mod invitat, urmatoarea interfata grafica se deschide (vezi imaginea de mai jos). Poti selecta culoarea cu care sa desenezi, sa desenezi si sa afli mai multe despre patrulaterul tau. Poti solicita un cont sau sa mergi inapoi in meniul principal. Utilizatorul poate sa salveze intr-un fisier .xml patrulaterul desenat si sa il incarce ulterior.



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.



1. **Bibliografie**
2. <https://www.geeksforgeeks.org/introduction-to-model-view-view-model-mvvm/>
3. <https://docs.oracle.com/javase/tutorial/jndi/ops/bind.html>
4. <https://www.geogebra.org/m/BypKNxsW>
5. <https://www.geogebra.org/m/JzqzHMf5>
6. <https://www.geogebra.org/m/CCmWYJHg>
7. Microsoft Teams/Proiectare Software/Laborator si Proiect/Lab 5