



Soft Educational Geometria cercului

Proiect la disciplina Proiectare Software
2022
- Tema 3 -

Student:
Bîrluțiu Claudiu-Andrei
Calculatoare și tehnologia informației
Grupa 30236
An universitar 2021/2022

Cuprins

1. Enunțul problemei.....	3
2. Instrumente utilizate.....	3
3. Limbajul de programare.....	4
4. Diagrame UML.....	4
5. Diagrame UML.....	4
5.1 Diagramele Use case.....	4
5.2 Diagramele <i>de clasă</i>	7
7. Database și singleton.....	11
6. Descrierea aplicației.....	11
7. Poze de prezentare.....	11
Anexă diagrame de activități.....	21
1. Diagrama desenare cerc.....	21
2. Diagrama desenare poligon.....	23
3. Diagrama desenare triunghi.....	23
4. Diagrama schimbare limba.....	24
5. Diagrama salvare cerc.....	24
6. Diagrama încărcare cerc.....	25
7. Diagrama autentificare.....	25
8. Diagrama logare.....	26
9. Diagrama quiz.....	27
10. Diagrama statistics.....	28

1. Enunțul problemei

Problema se referă la dezvoltarea unei aplicații desktop ce poate fi utilizată ca soft educațional pentru studiul cercului, precum și verificarea unor cunoștințe de matematică printr-un quiz de 10 întrebări. Rezolvarea acesteia presupune trecerea prin etapele de analiză, proiectare și implementare a softului respectând şablonul arhitectural **Model-View-Controller**. Obiectivul temei este acela de familiarizare cu diferite şabloane arhitecturale, în cazul de față **MVC**, și cu alte tehnici de programare prin care se vor respecta principiile SOLID. De asemenea, se va folosi şablonul comportamental **Observer** pentru a gestiona flow-ul aplicației, dar și alte design pattern-uri cum ar fi **Builder**, pentru crearea de instanțe de obiecte din clasele de model pentru a construi logica de funcționare a aplicației. De asemenea, pentru conexiunea la baza de date se va folosi design patternul **Singleton**, pentru crearea unei singure instanțe.

Aplicația va oferi utilizatorului posibilitatea interactivă de vizualizare a cercurilor importante ale triunghiurilor și analiza principalelor caracteristici ale cercului cum ar fi raza și influența acesteia asupra ariei, arcul de cerc și aria sectorului de cerc determinat de acesta din urmă. Se vor putea observa cercurile circumscris și înscris ale poligoanelor regulate, iar în cadrul triunghiurilor oarecare se vor putea vizualiza următoarele cercuri specifice: cercul lui Tucker, cercurile lui Lucas, cercul ortocentroidal, cercurile lui Neuberg, cercul lui Adams, cercul lui Brocard. Aceste funcționalități descrise mai sus sunt disponibile tuturor utilizatorilor fără a se loga. O funcționalitate care necesită logare este evaluarea userilor prin quiz-uri formate din 10 întrebări alese aleator din 50 de întrebări disponibile. Userii logați pot verifica și statistica quizurilor date.

2. Instrumente utilizate

De-a lungul dezvoltării aplicației s-au folosit mai multe instrumente software ce vin în ajutorul dezvoltatorului, creând un mediu propice în care acesta poate să își analizeze, proiecteze și structureze ideile și organiza munca.

În faza de proiectare a aplicației s-a folosit instrumentul **StarUML**, disponibil la adresa <https://staruml.io/>. Aceasta oferă posibilitatea proiectării aplicației prin sintetizarea cazurilor de utilizare prin diagrame *Use Case*, a *diagramelor de activități*, dar și prin definirea arhitecturii acesteia cu ajutorul unor *Diagrame UML de clase sau pachete* și crearea de diagrame ERD în care se observă relaționarea entităților aplicației.

Pentru faza de implementare s-a folosit mediul de dezvoltare *IntelliJ IDEEA 2022.1 EAP Ultimate Edition (JetBrains)* care oferă o serie de funcționalități cum ar fi versionarea aplicației cu GIT, formatarea codului și de asemenea un sistem de sugestii și autofocus de acuratețe ridicată. Proiectul creat pentru aplicație în IntelliJ este de tip Maven, având dependințele definite în fișierul *pom.xml*.

Pentru versionarea aplicației la nivel local s-a folosit GIT, iar repository-ul remote a fost creat pe Gitlab și este disponibil la: https://gitlab.com/birlutiuciaudiu/educationalsoft_mvc.git

De asemenea, pentru crearea bazei de date s-a folosit **postgresSql** cu environment-ul **Pgadmin4**. Pentru migrarea bazei de date s-a folosit **Flyway**. A fost creat un script de introducere a 50 de întrebări în baza de date.

3. Limbajul de programare

Pentru dezvoltarea aplicației s-a optat pentru limbajul de programare JAVA (17). Cu acest limbaj se poate lucra foarte eficient cu conceptul de obiect. Pentru dezvoltatorul aplicației a reprezentat o nouă posibilitate de familiarizare cu acest limbaj și principiile SOLID și de a exersa tehniciile de clean coding. Un alt motiv pentru care s-a ales acest limbaj este faptul că împrumută o mare parte din sintaxa C și C++, dar are un model al obiectelor mai simplu.

4. Diagrame UML

4.1 Diagramele Use case

În figura de mai jos sunt sintetizate grafic posibilele interacțiuni ale utilizatorului cu sistemul creat. Există un singur actor, orice utilizator care nu are nevoie de un cont de logare pentru accesul la funcționalitățile aplicației.

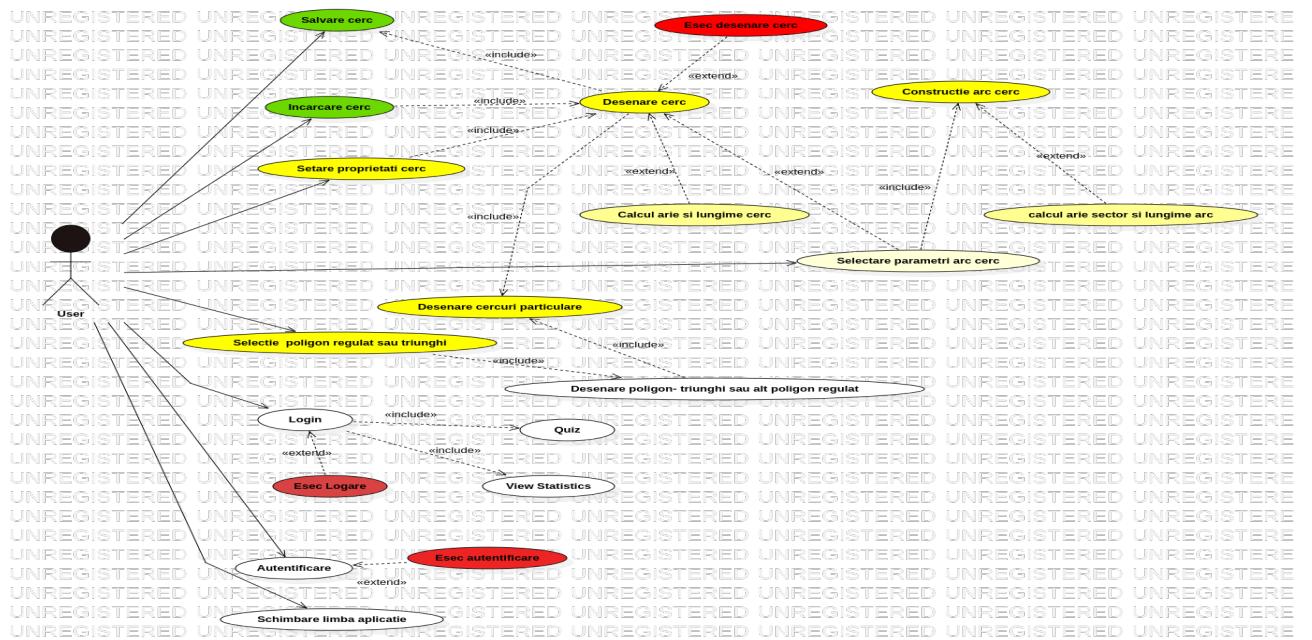


Figure 1 Use case diagram

Cazurile de utilizare ale aplicației:

Use case: desenare cerc

Actor principal: orice utilizator

Principalul scenariu de succes:

- Utilizatorul selectează radio buttonul „Draw circle”
- Utilizatorul selectează opțional tipul de linie cu care se va desena cercul, culoarea și grosimea
- Utilizatorul va modifica dimensiunea cercului prin dragg cu mouse-ul în fereastra albă
- În timpul desenării cercului se vor calcula elementele principalele ale acestuia și vor fi afișate în fereastra de results

Use case: desenare arc cerc

Actor principal: orice utilizator

Principalul scenariu de succes:

- Utilizatorul selectează radio buttonul „Draw circle”
- Utilizatorul va introduce valori pentru StartAngle și EndAngle pentru definirea unghiului arcului de cerc
- Pe cerc să va desena cu linie punctată arcul de cerc și se va calcula în timpul desenării Lungimea arcului și Arei sectorului de cerc

Use case: desenare poligon regulat cu cercul circumscris și inscris

Actor principal: orice utilizator

Principalul scenariu de succes:

- Utilizatorul selectează radio buttonul „Draw polygon”
- Utilizatorul selectează opțional tipul de linie cu care se va desena cercul, culoarea și grosimea
- Utilizatorul va seta numărul de vârfuri ale poligonului. Dacă va introduce o valoare invalidă implicit aplicația va trata cazul și se va desena un triunghi echilateral
- În timpul desenării poligonului regulat se vor desena și cercurile inscrise și circumschrele ale poligonului.

Use case: desenare/urmărire cercuri particulare ale triunghiului

Actor principal: orice utilizator

Principalul scenariu de succes:

- Utilizatorul selectează radio buttonul „Draw triangle”
- Utilizatorul selectează una din opțiunile: Inscribed, Circumscribed, Tucker, Lucas, Orthocentroidal, Neuberg, Adams sau Brocard
- În timpul desenării triunghiului se vor desena și cercurile particulare selectate și alte elemente geometrice utilizate pentru determinarea lor

Use case: logare

Actor principal: orice utilizator

Principalul scenariu de succes:

- Utilizatorul apasă pe butonul „Quiz” și se deschide o fereastră de logare
- Utilizatorul introduce datele de logare: username și parolă
- Utilizatorul apasă pe butonul de Login și apare o nouă pagină cu informații despre quiz

Eșec:

- Utilizatorul introduce date invalide: username sau/și parolă invalide (inexistență cont)
- Afisare mesaj de eroare cu informațiile corespunzătoare
- Utilizatorul poate introduce noi date sau se poate autentifica(crea un nou cont)

Use case: autentificare

Actor principal: orice utilizator

Principalul scenariu de succes:

- Utilizatorul apasă pe butonul „Quiz” și se deschide o fereastră de logare
- Utilizatorul introduce datele de autentificare: username și parolă
- Utilizatorul apasă pe butonul de *Autentificate*
- Afisarea unui mesaj de bun venit
- Logare automată pe pagina de quiz

Eșec:

- Utilizatorul introduce date invalide: username sau/și parolă care nu corespund validatorilor sau există deja un cont cu username-ul introdus
- Afisare mesaj de eroare cu informațiile corespunzătoare
- Utilizatorul poate introduce noi date pentru autentificare(crea un nou cont)

Use case: verificare cunoștiinte(Quiz)

Actor principal: utilizator logat în aplicație

Principalul scenariu de succes:

- Utilizatorul apasă pe butonul de Start quiz
- Un set de 10 întrebări vor fi afisate pe ecran pe rând
- La întrebarea cu index 10 utilizatorul apasă pe butonul *Finish quiz* și se va salva quizul în baza de date
- Se vor afisa rezultatele intermediare și cel final

Eșec:

- Utilizatorul apasă pe butonul de reset și se resetează quizul
- Utilizatoruliese din aplicație, iar quizul în starea curentă se pierde

Use case: vizualizare statistici

Actor principal: utilizator logat în aplicație

Principalul scenariu de succes:

- Utilizatorul apasă pe tabul **Statistics**
- Se vor afisa 3 grafice: pie chart cu gruparea zuiurilor pe punctaje obtinute, bar chart cu punctajele medii ale tuturor utilizatorilor, time chart cu evoluția quizurilor utilizatorului

Use case: schimbare limbă

Actor principal: orice utilizator

Principalul scenariu de succes:

- Utilizatorul apasă pe unul dintre drapele corespunzătoare limbii dorite
- Toată aplicația va fi tradusă în limba selectată

La finalul lucrării se vor regăsi anexele cu diagramele de activități.

4.2 Diagramele de clasă

Proiectarea aplicației are în vedere respectarea principiilor SOLID și a designului arhitectural *Model-View-Controller*. Clasele au fost grupate după cum se observă în *figura 3*, în pachetele corespunzătoare modelului arhitectural. Folosirea acestei arhitecturi duce la izolarea părții logice de interfață proiectului, rezultând în aplicații extrem de ușor de modificat. Ca structură, şablonul arhitectural MVC este alcătuit din trei părți esențiale ce pot fi deduse și din denumirea design pattern-ului: Model, View și Controller:

- **Model** - definește domeniul aplicației, gestionează comportamentul și datele domeniului aplicației, răspunde la cererile referitoare la informații despre starea sa
- **View** - definește interfața utilizator. Gestionează comportamentul și datele domeniului aplicației, răspunde la cererile referitoare la informații despre starea sa.
- **Controller** -definește modul în care interacționează celelalte 2. Acceptă input de la utilizator și instruiește modelul să realizeze acțiuni în funcție de acest input.

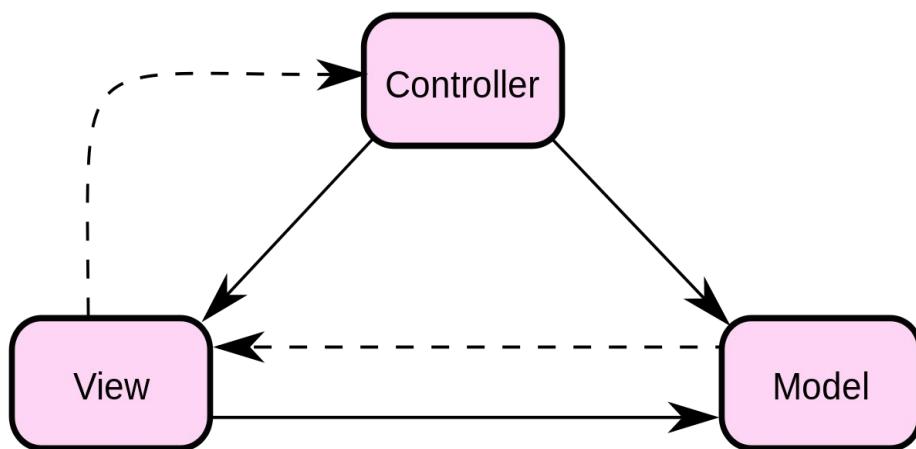


Figure 2 Model-View-Controller

Clasele din **geometryutils**

- Clasa abstractă **GeometricElement** definește conceptul abstract de element geometric. Aceasta va fi extinsă de clasele **Point** și **Line** (componente ce nu au arie) și **GeometricFigure** (o clasă la rândul ei abstractă ce definește figurile geometrice cu arie definite prin clasele **Triangle**, **Polygon**, **Circle**, **Arc**).
- Pentru fiecare element geometric există o clasă specifică pentru desenarea grafică a acestuia având anumite proprietăți precum culoare, grosimea liniei sau tipul acestaia. Există astfel o clasă **GeometricElementDrawable**.
- Pentru manipularea elementelor de desenat s-a definit o clasă **Drawing** ce va conține o lista de elemente de desenat pentru o schemă completă

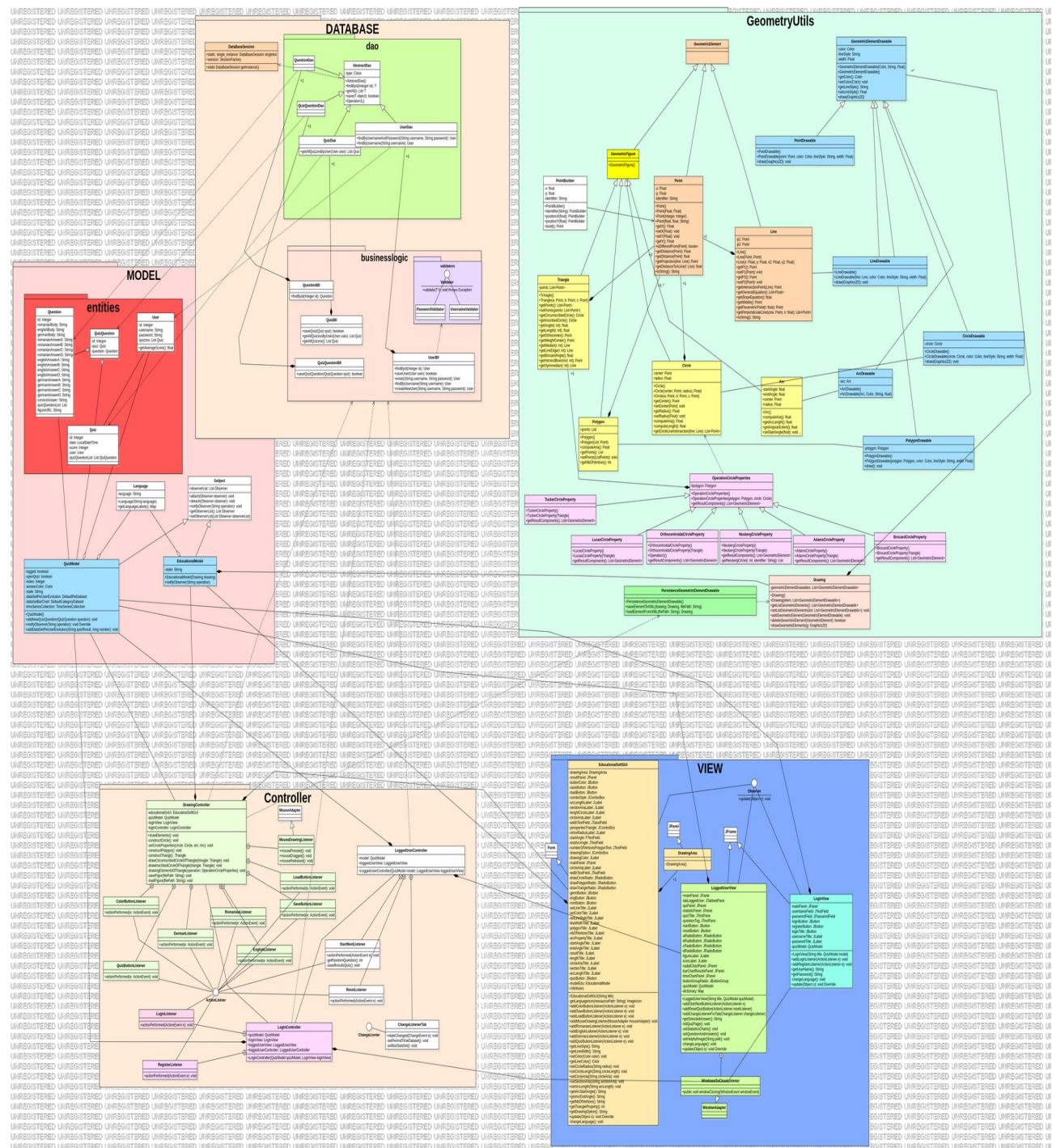


Figura 3 Diagrama de clase

- S-a folosit o clasă abstractă *OperationCircleproperties* care va fi moștenită de clase specialize pentru calcul unor proprietăți a triunghiului și a tipurilor de cerc particulare menționate mai sus
- Pentru persistența datelor din model s-a folosit clasa *PersistenceGeometricDrawable* ce va salva și încărca datele într-un fișier .xml
- De asemenea pentru clasa *Point* s-a creat clasa **PointBuilder** pentru a putea crea obiecte de tip Point mai ușor, în cazul în care se dorește sau nu ca punctul creat să i se atașeze o valoare câmpului *identifier*

Clasele din model

- În subsistemul **entities** sunt incluse clasele de model ce reprezintă entități ale bazei de date. Cu aceste clase se vor instanția obiecte ale căror atribute vor fi salvate în coloanele corespunzătoare ale tabelelor din baza de date. Clasa **Question** descrie modelul de întrebare și are ca atribute textul întrebării cu variantele de răspuns în cele trei limbi de circulație internațională. Clasa **User** cuprinde informațiile relevante de autentificare ale unui utilizator în aplicație. Deoarece pentru crearea sesiunii de conexiune la baza de date s-a folosit **Hibernate**, în user se vor defini și relațiile cu celelalte clase entitate, cum ar fi relația one-to-many între User și Quiz. S-a mai definit o clasă *QuizQuestion* care rezolvă problema relației Many-to-Many dintre entitățile Quiz și Question.
- Pentru interfața de desenare a figurilor geometrice s-a creat un model **EducationalModel** care are ca field principal un obiect de tipul **Drawing** ce reprezintă obiectul actual de desenat. Pentru ca să se notifice interfața grafică atașată acestui model (*EducationalSoftGUI*) acest model extinde clasa **Subject**, pe cînd view-ul *EducationalSoftGUI* implementează interfața **Observer** cu suprascrierea metodei *update*. Clasa abstractă **Subject** cu interfața **Observer** descriu de fapt design patternul **Observer**
- **QuizModel** este o clasa model ce este atașată interfeței grafice **LoggedUserView** și **LoginView**. La fel și acestea respectă design patternul **Observer**. Acest obiect va ține evidența userului logat în aplicație și va avea ca field și un obiect de tip Quiz ce se va actualiza în momentul în care utilizatorul dă testul de verificare.
- Clasa **Language** conține o metodă ce va citi un fișier XML și va returna un dicționar al cuvintelor folosite în aplicație pentru o limbă specifică: română, engleză și germană.

Clasele din view:

- *EducationalSoftGui* este clasa ce descrie interfața cu utilizatorul pentru desenarea figurilor geometrice. Această opțiune este disponibilă pentru toți utilizatorii.
- **LoginView** descrie view-ul de login și autentificare și va fi vizibil în momentul în care un utilizator apasă butonul de Quiz din interfața *EducationalSoftGUI*.
- **LoggedUserView** este clasa ce implementează logica pentru GUI-ul specific testului de verificare a cunoștiințelor și pentru vizualizarea graficelor de statistică.
- În momentul în care apar modificări la nivelul modelului atașat interfețelor descrise mai sus, se va executa metoda **update** și se vor face adaptările necesare ale componentelor interfeței în concordanță cu starea modelului.

Clasele din Controller:

- **DrawingController** se ocupă de gestionarea funcționalității de desenare a figurilor geometrice. Aceasta este format din inner classes, ce pot fi văzute în figura 3, ce implementează logica de acțiune a butoanelor din interfața grafică atașată. De asemenea, acest controller se ocupă și de evenimentul de schimbare a limbii aplicației.
- **LoginController** această clasă se va folosi de businesslogic pentru interogarea bazei de date și obținerea validărilor de logare sau autentificare pentru datele introduse în interfața grafică

- **LoggedUserController** va implemta controlul necesar pentru generarea quizului, dar și pentru generarea statisticilor. Acesta se va folosi la rândul lui de clasele din *businesslogic* pentru a extrage random întrebări pentru quiz sau date despre quizurile date.

Despre pachetul database se va discuta în capitolul următor.

5. Database și design patternul ales (Singleton)

Ca bază de date s-a folosit **postgreSQL**, cu interfața desktop **PgAdmin4**. Pentru conexiunea la baza de date s-a folosit un driver PostgreSQL inclus ca dependință în *pom.xml*. S-a folosit tehnologia specifică dependinței **Hibernate** pentru a face corespondența claselor din subpachetul **entities** cu tabelele din baza de date. De asemenea, crearea tabelelor în baza de date s-a realizat în mod automat de către hibernate, iar folosind tool-ul de migrare a bazei de date **Flyway** s-au definit 2 scripturi SQL de inserare a unui user default și pentru inserarea celor 50 de întrebări din care se vor genera quizurile. Pentru configurarea și conexiunea la baza de date s-a folosit design patternul creațional **Singleton** pentru a crea o singură instanță a conexiunii pentru toată aplicația.

6. Descrierea aplicației

Aplicația prezintă un mediu interactiv pentru exemplificarea proprietăților cercului și a triunghiului cu cercurile sale particulare. Aceasta este ușor de folosit, iar procesul de desenare se face cu ajutorul mouse-ului prin dragg.

Softul poate fi folosit de orice utilizator și vine în completarea vizuală a teoriei cercurilor. De asemenea utilizatorii se pot autentifica și pe urmă loga în aplicația de quiz și vor putea să își urmărească evoluția prin testări repetitive. Quizul va fi format din 10 întrebări de geometrie și gradul de dificultate a întrebărilor va crește.

Din punctul de vedere a implementării s-a încercat pe cât posibil clean code-ul și definirea cât mai eficientă a operațiilor necesare determinării particularităților cercurilor descrise mai sus. S-au translatat formule și metode matematice (de exemplu ecuația cercului definit de 3 puncte din plan) în cod. Pentru desenare s-a folosit Graphics2D.

7. Poze de prezentare

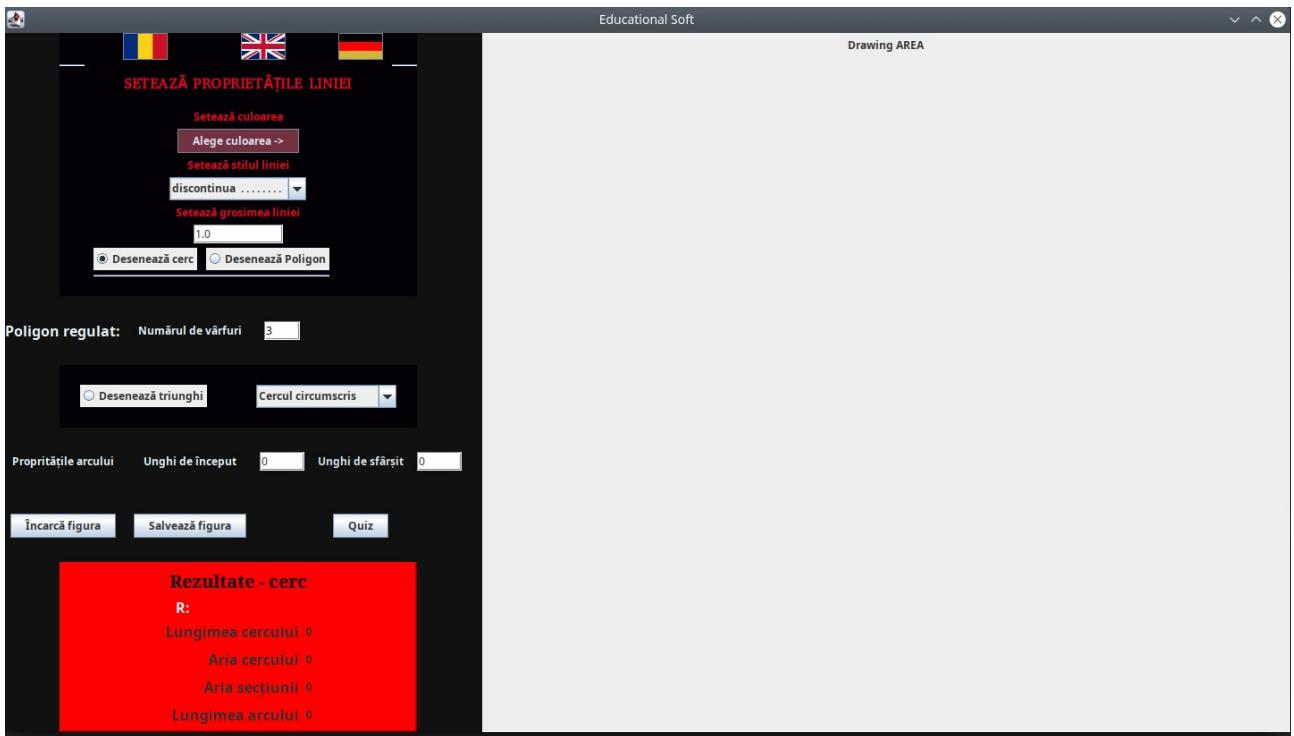


Figura 4 Pornirea Aplicatiei

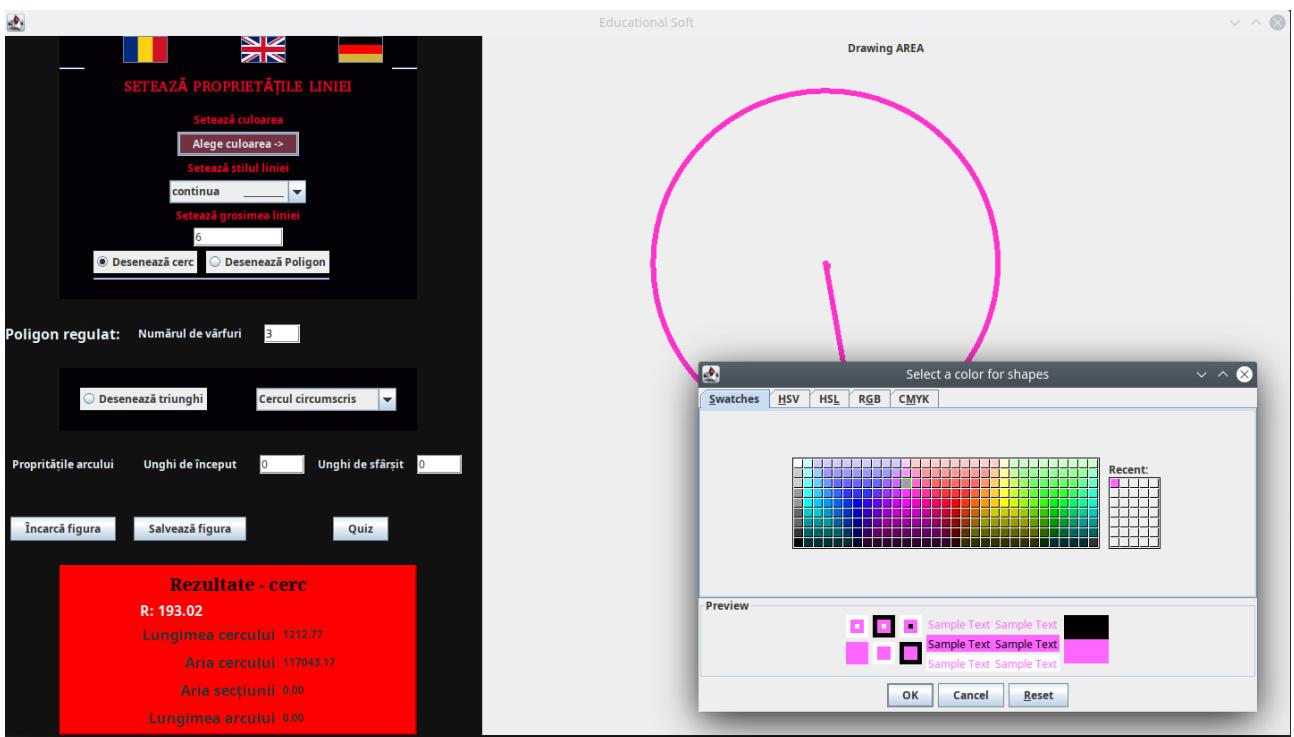


Figura 5 Setare stil desenare

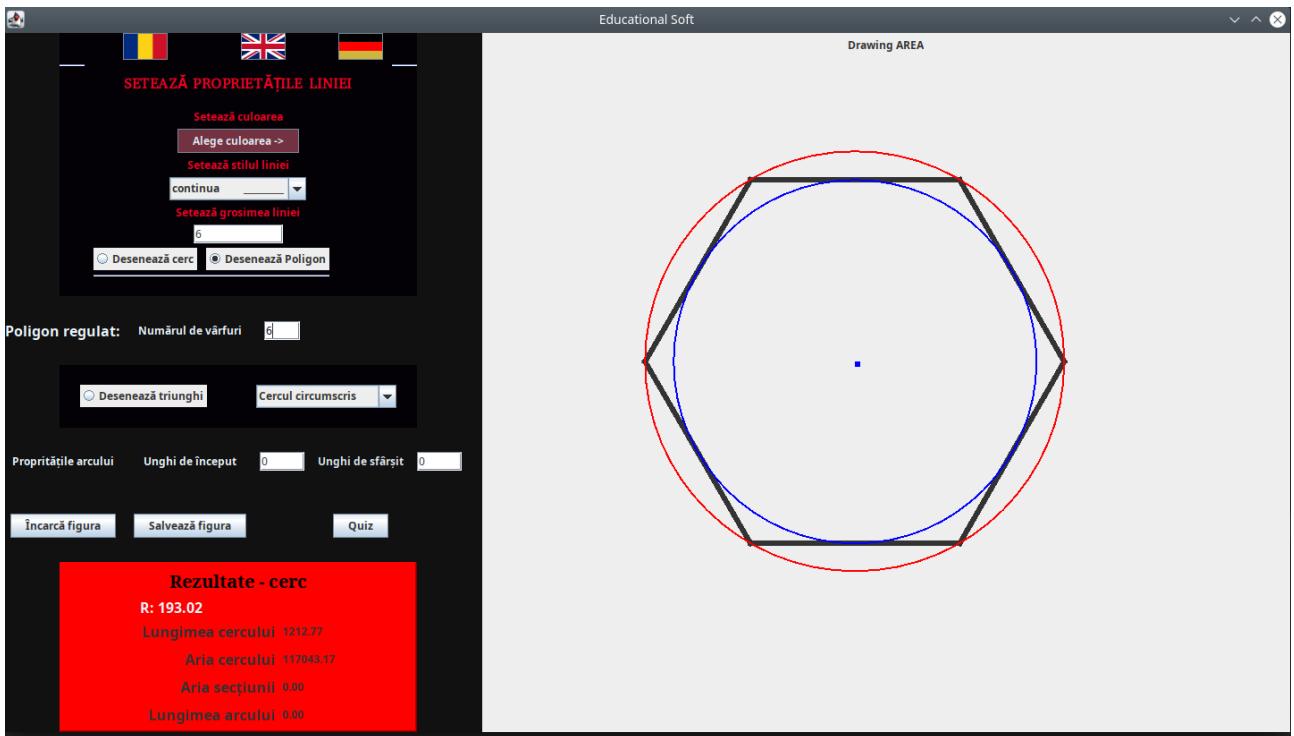


Figura 6 Desenare poligon cu cercul incris si circumscris

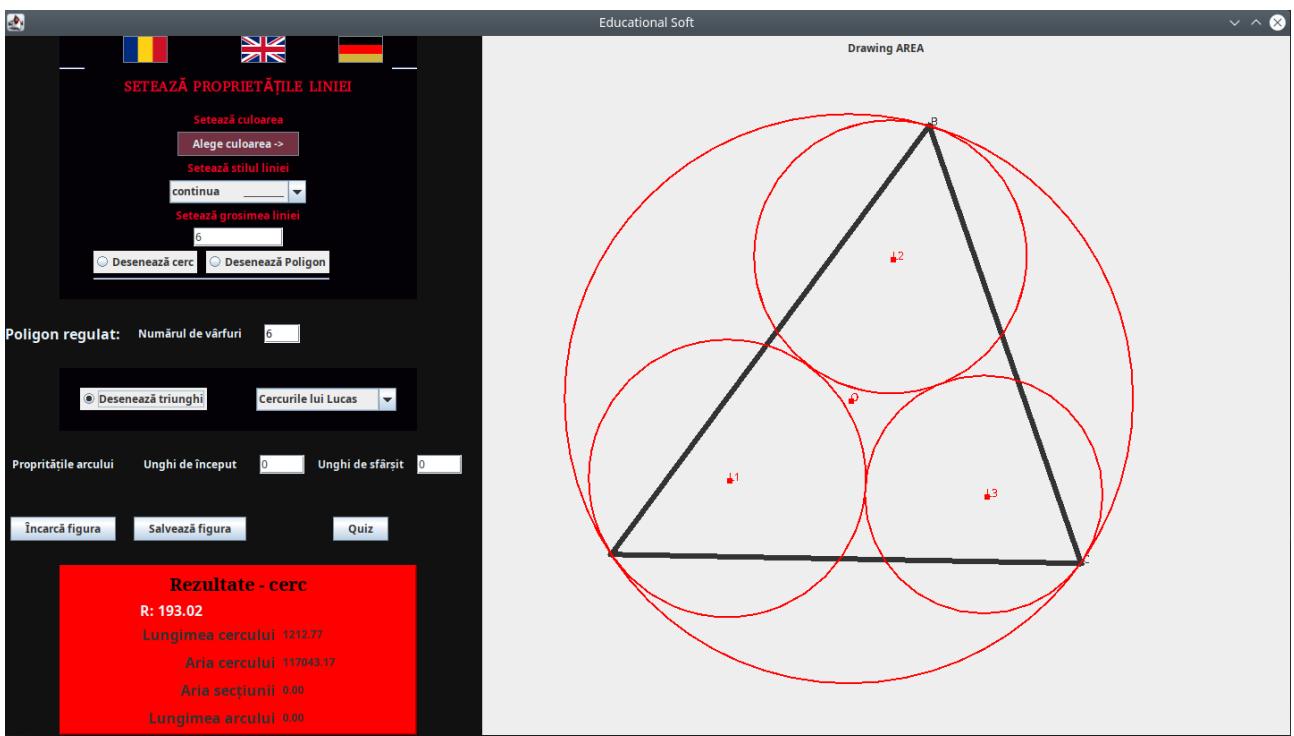


Figura 7 Cercurile lui Lucas

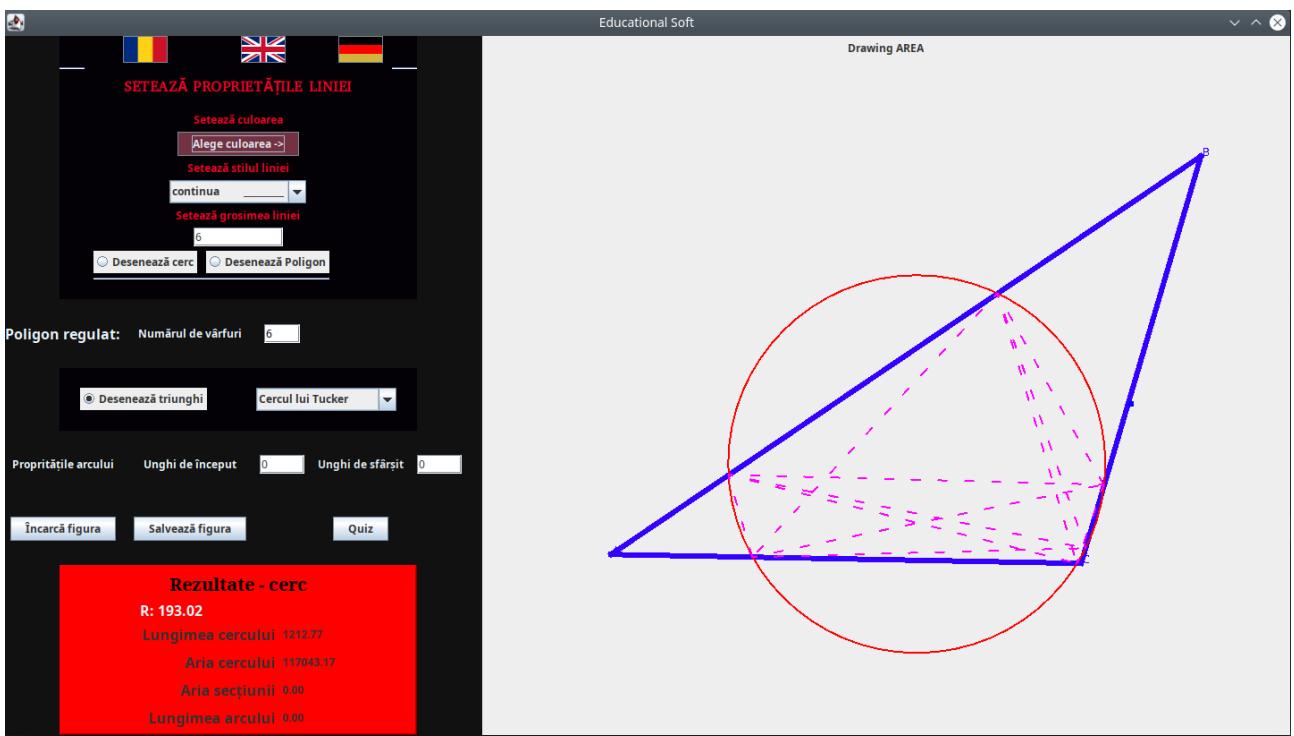


Figura 8 Cercul lui Tucker

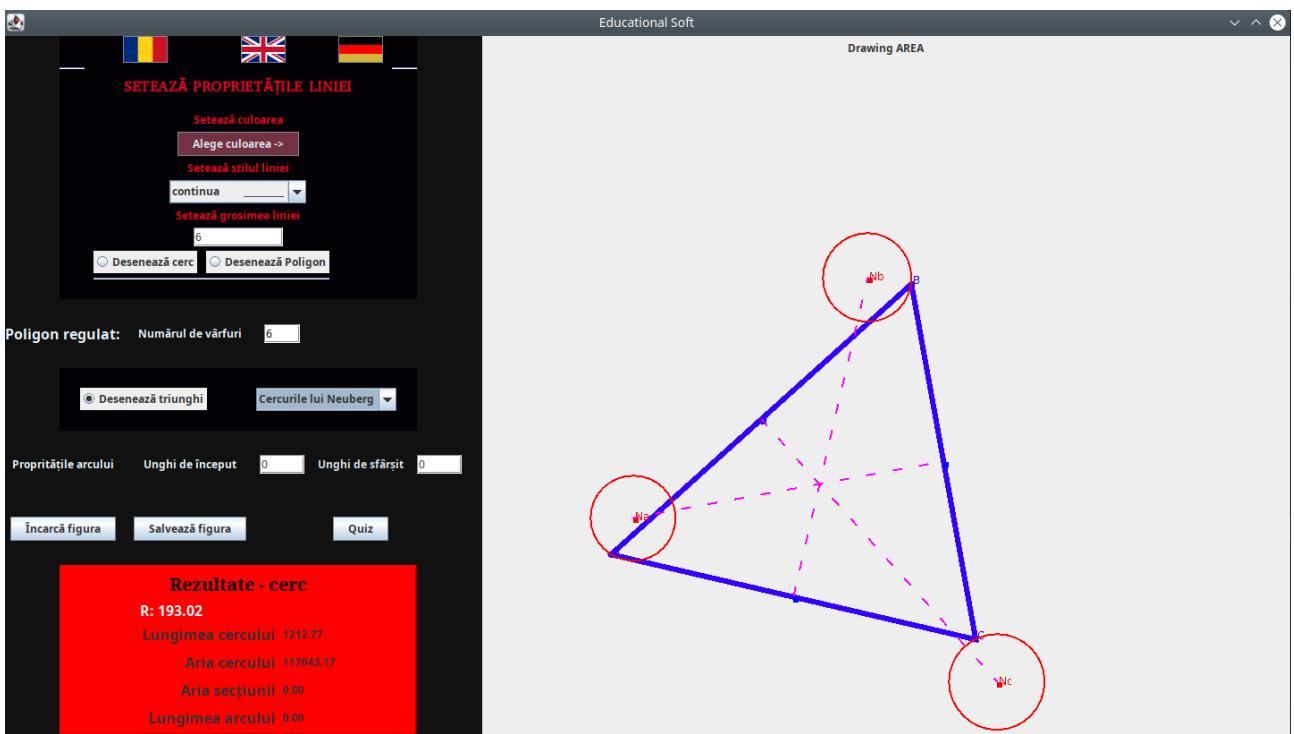


Figura 9 Cercurile lui Neuberg

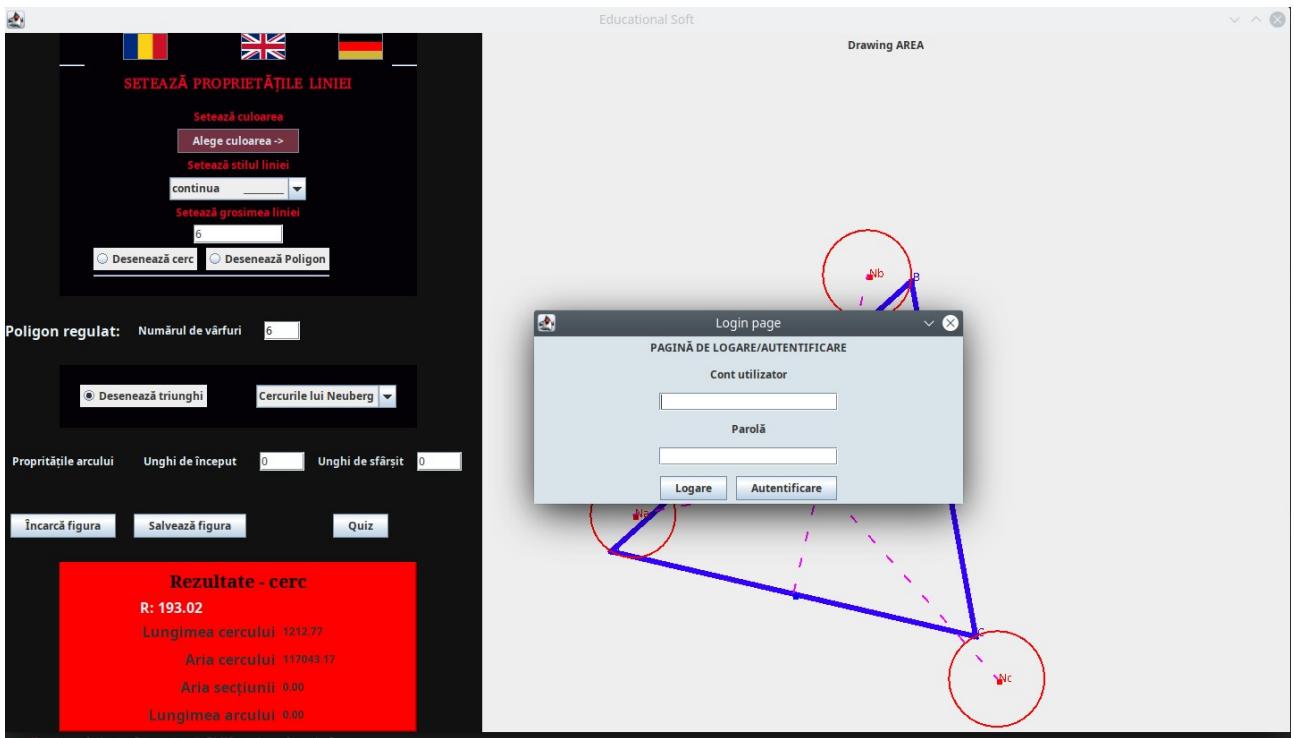


Figura 10 Logare/ autentificare

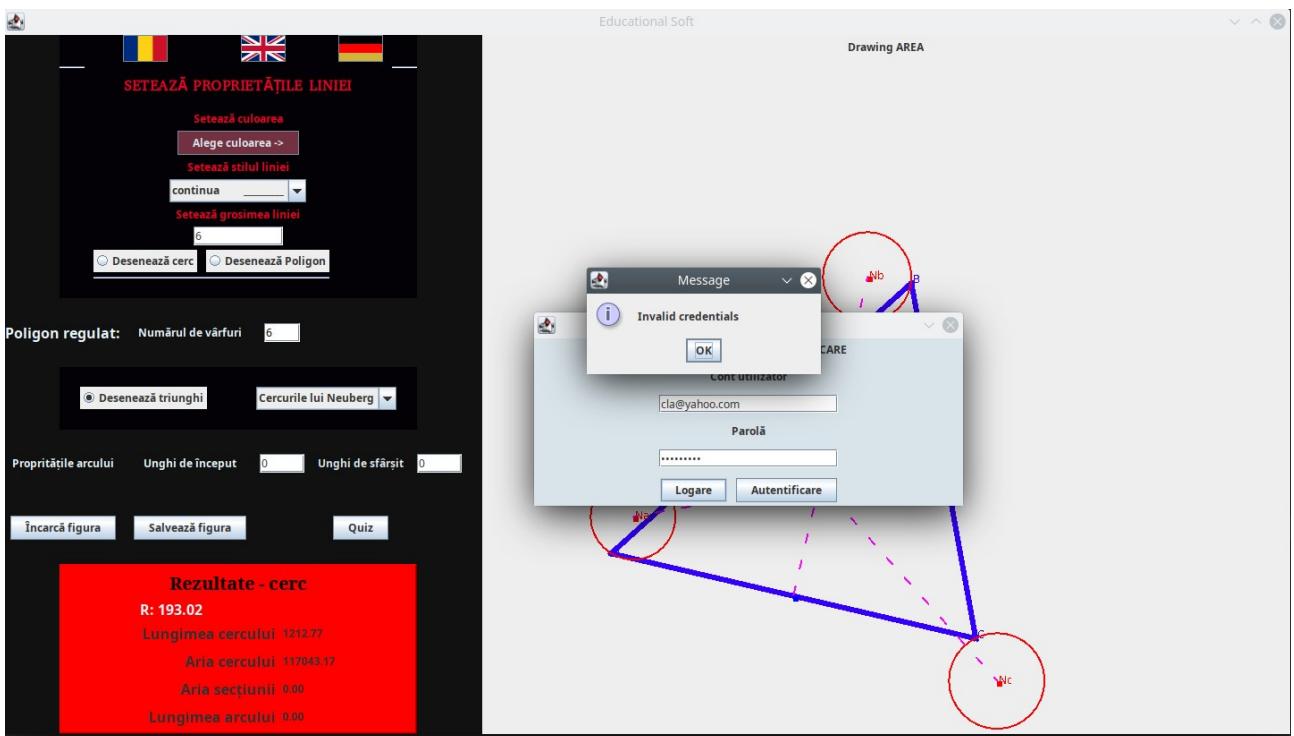


Figura 11 Failed login

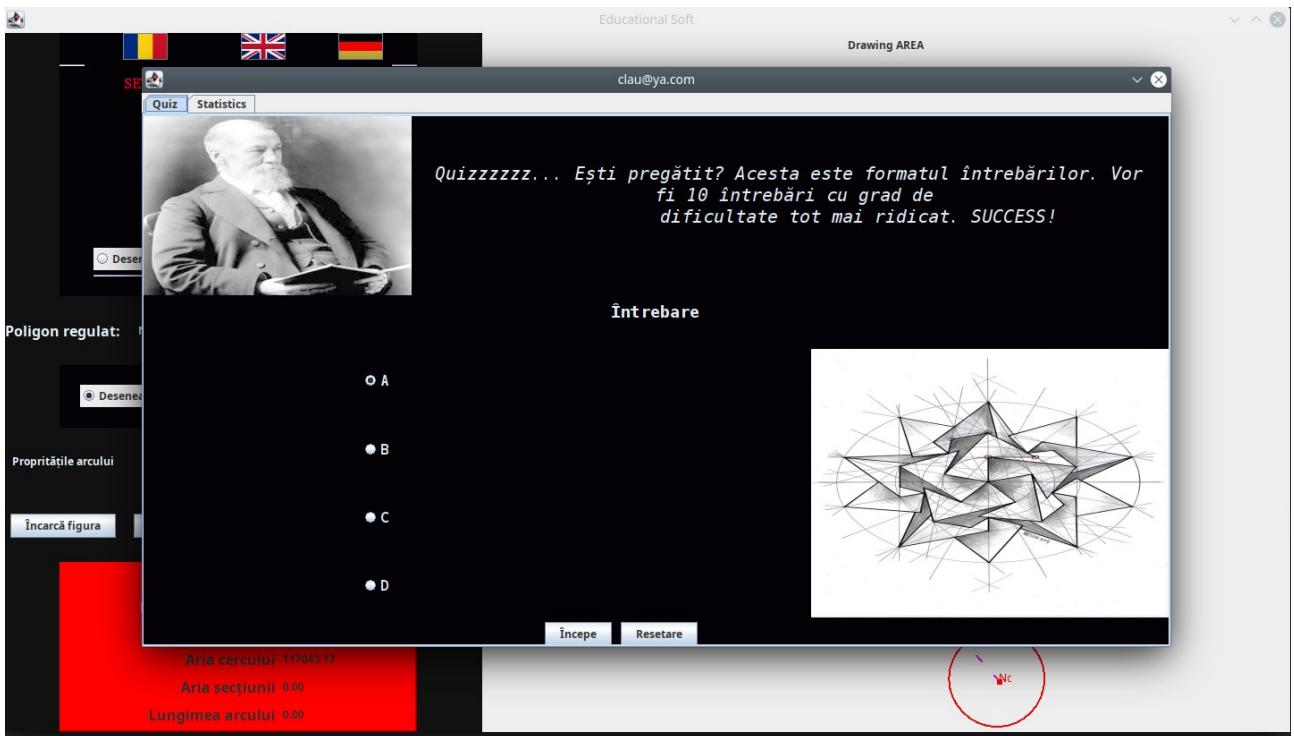


Figura 12 Pagina de start quiz

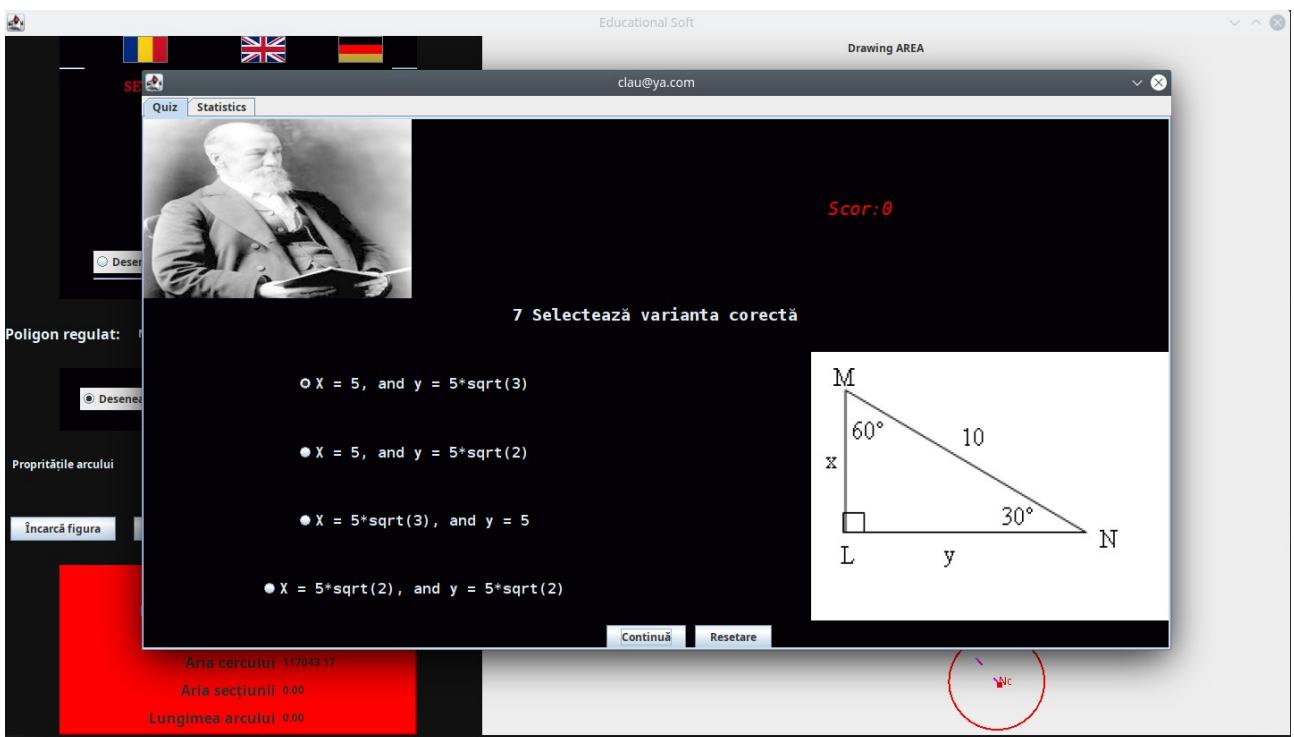


Figura 12 Intrebare quiz cu imagine alaturata

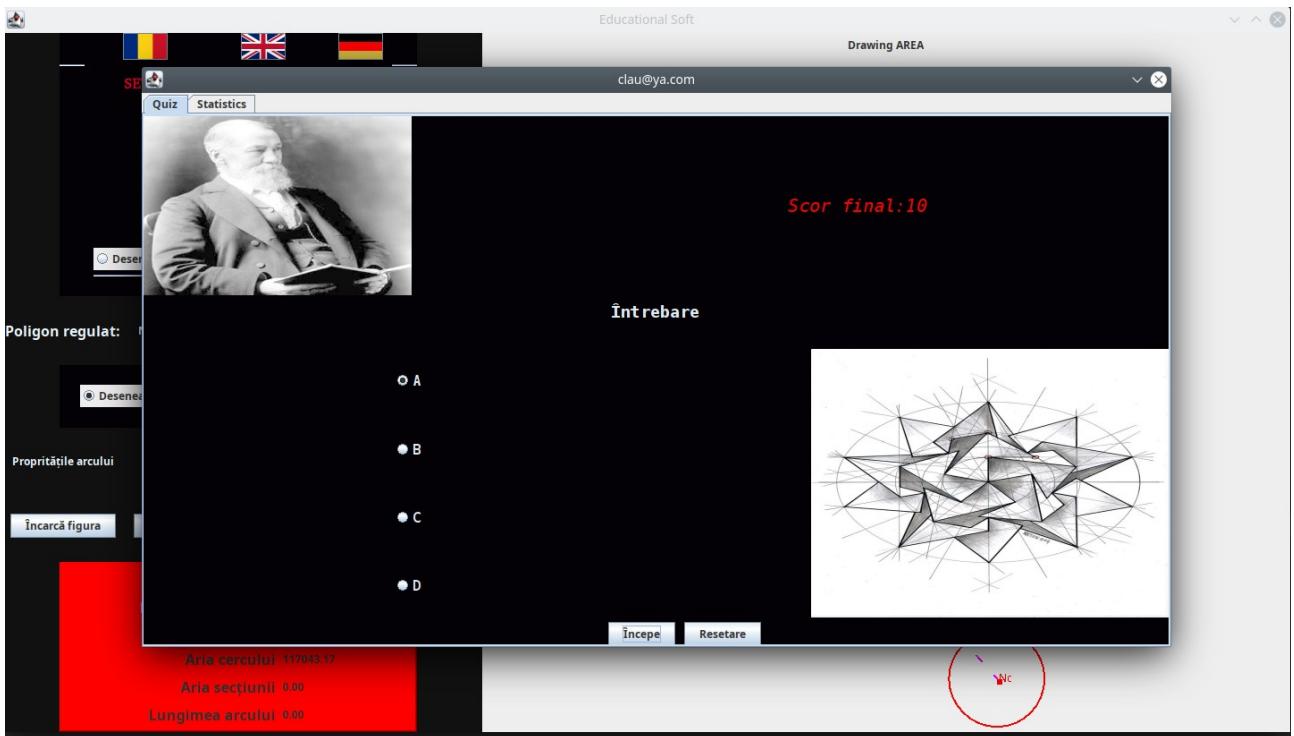


Figura 13 Scor final quiz

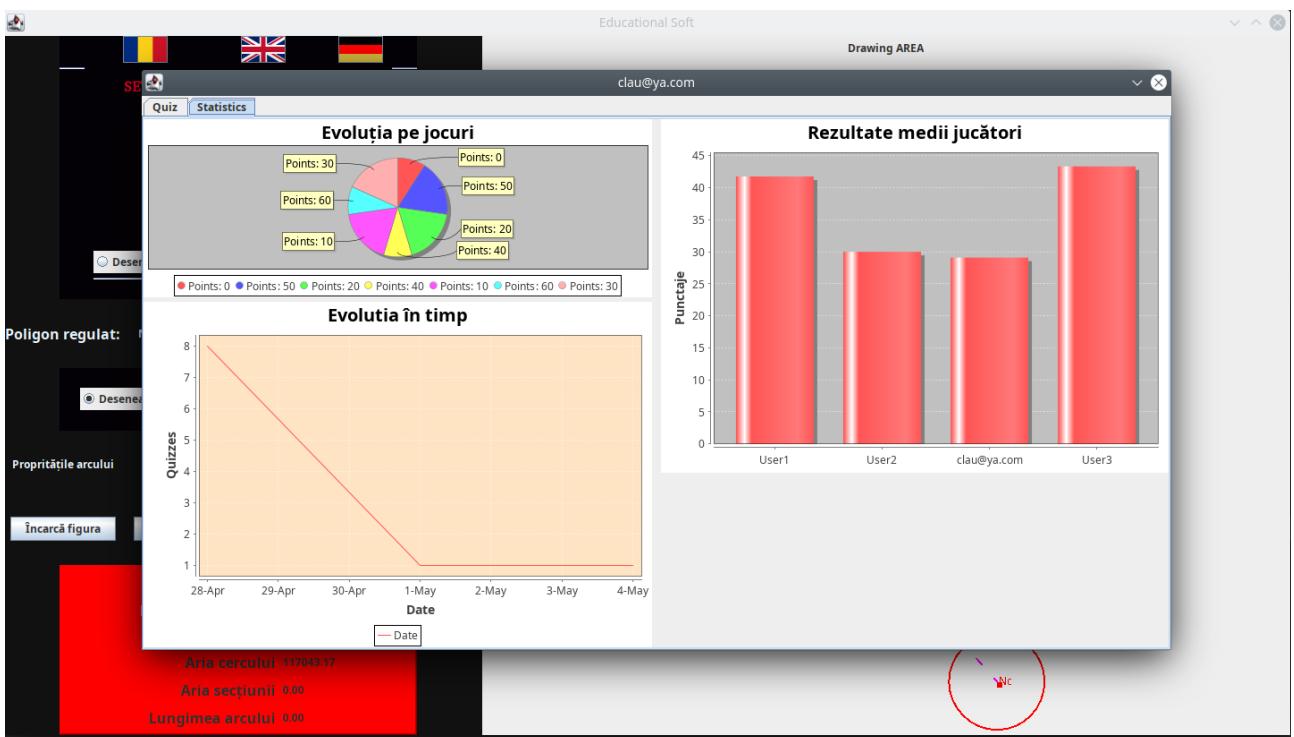
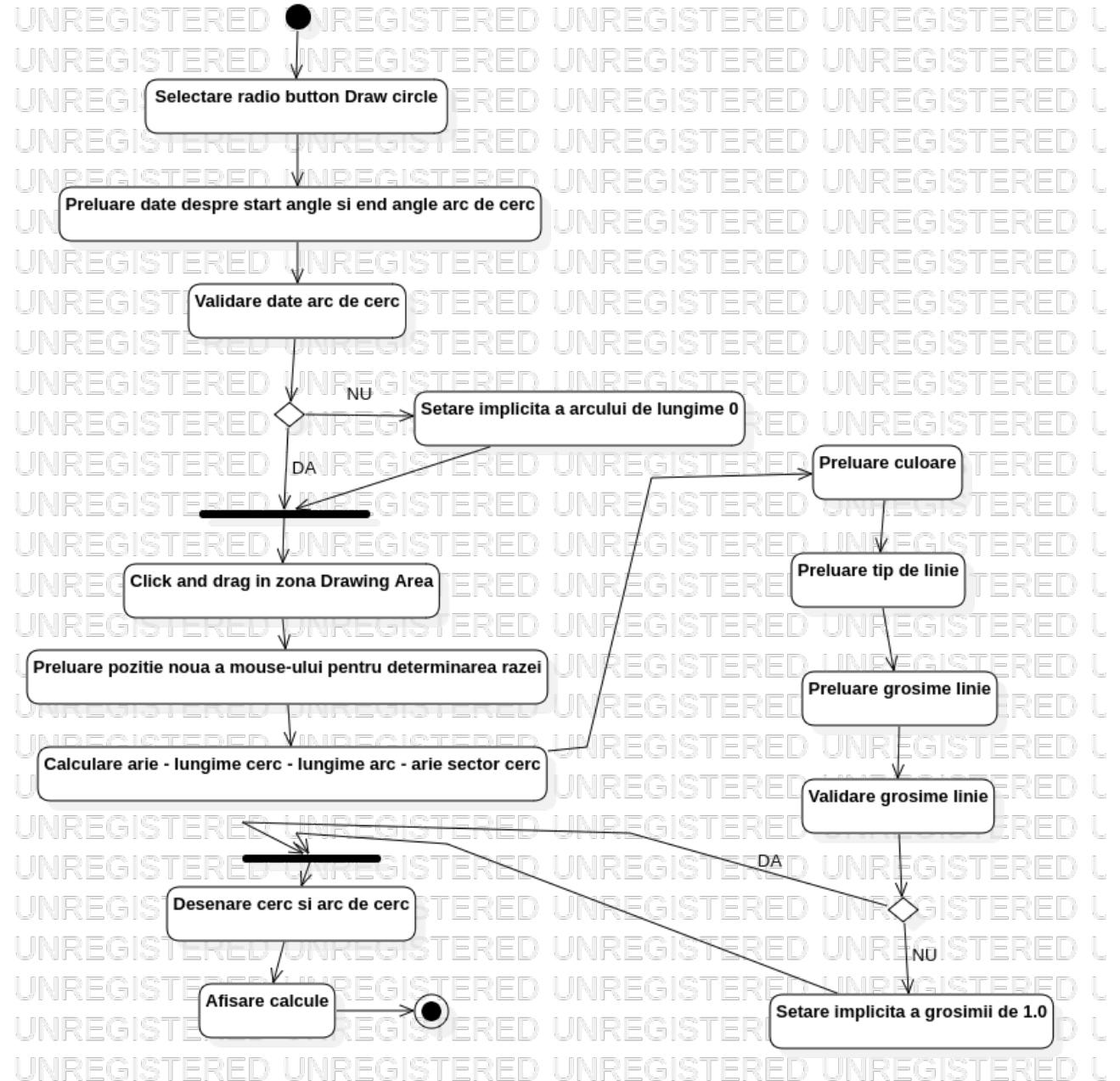


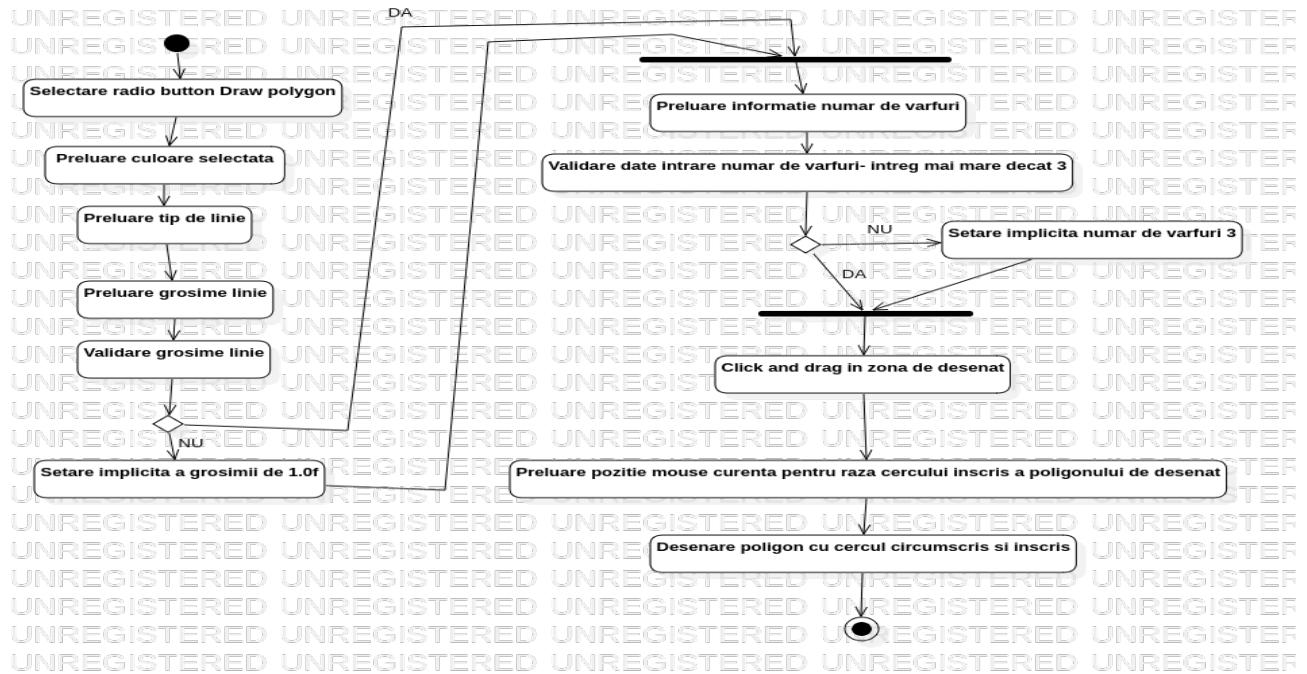
Figura 14 Scor final quiz

Anexă diagrame de activități

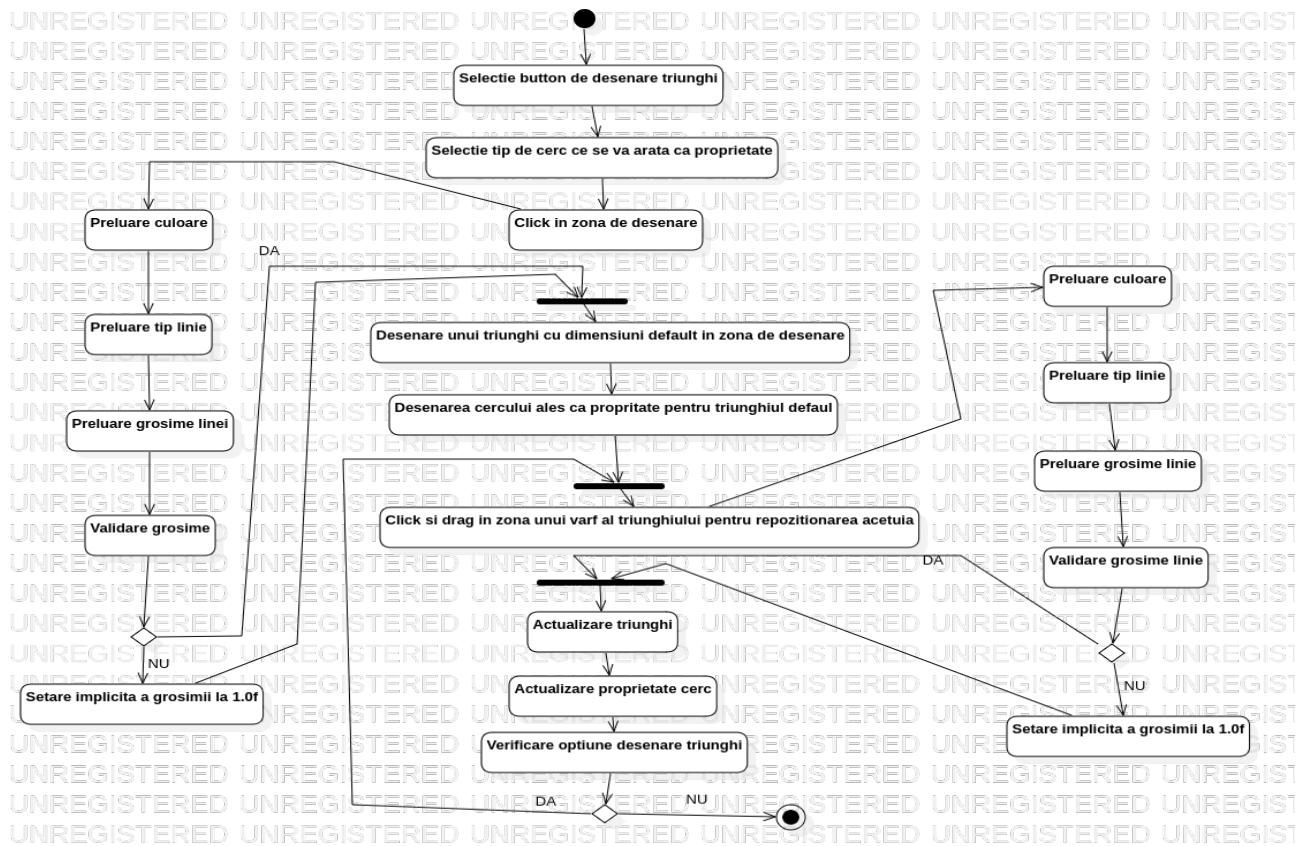
1. Diagrama desenare cerc



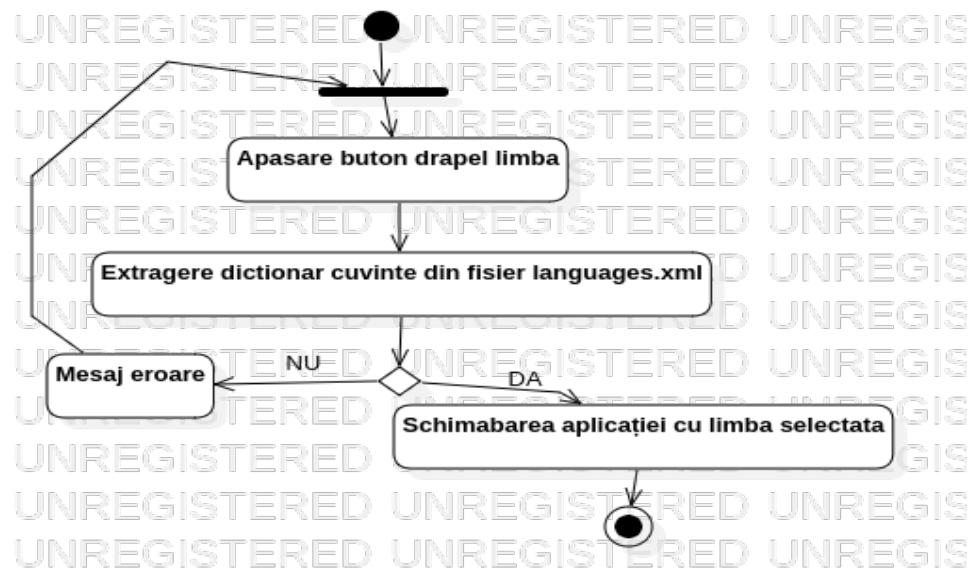
2. Diagrama desenare poligon



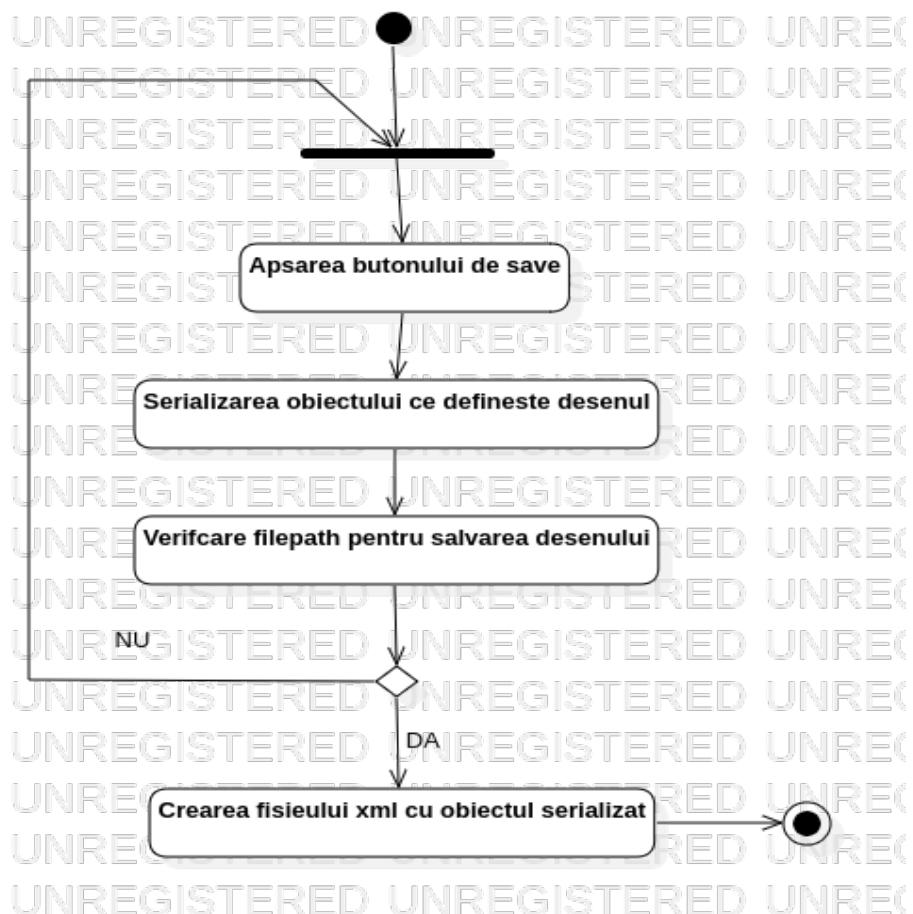
3. Diagrama desenare triunghi



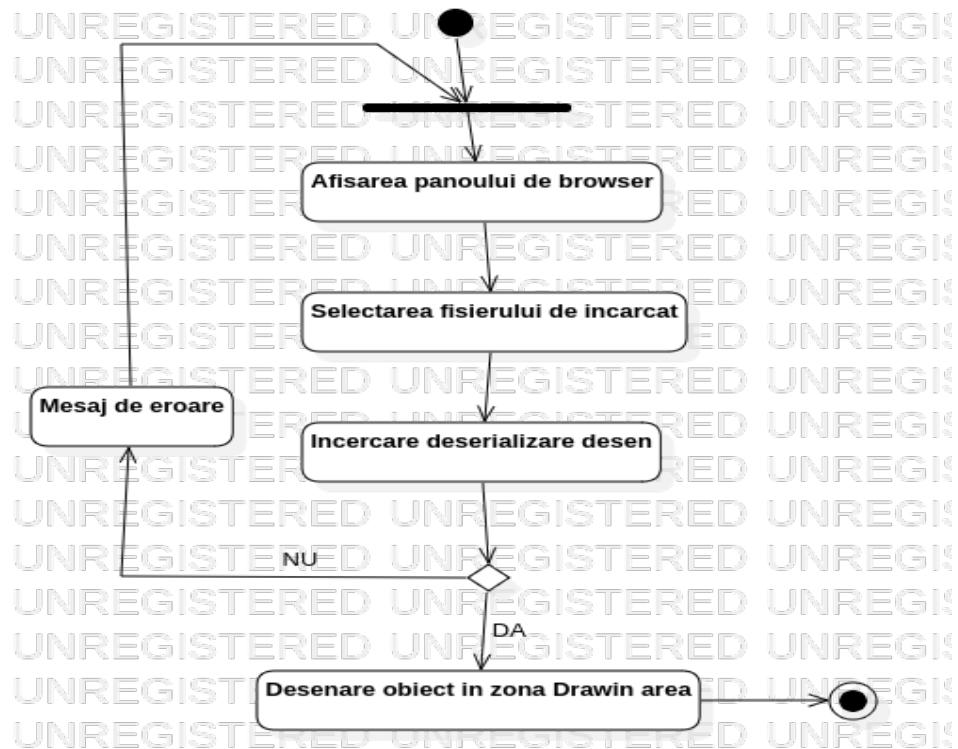
4.Diagrama schimbare limba



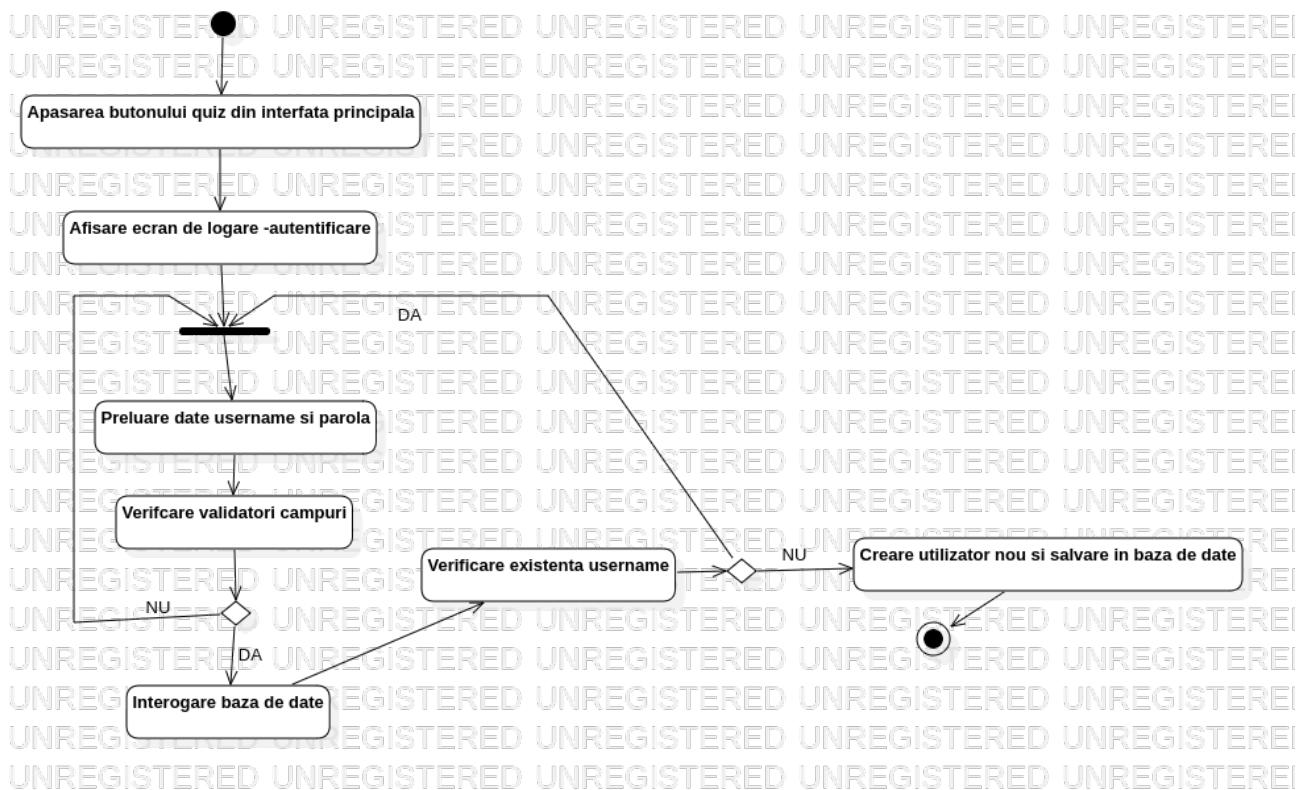
5.Diagrama salvare cerc



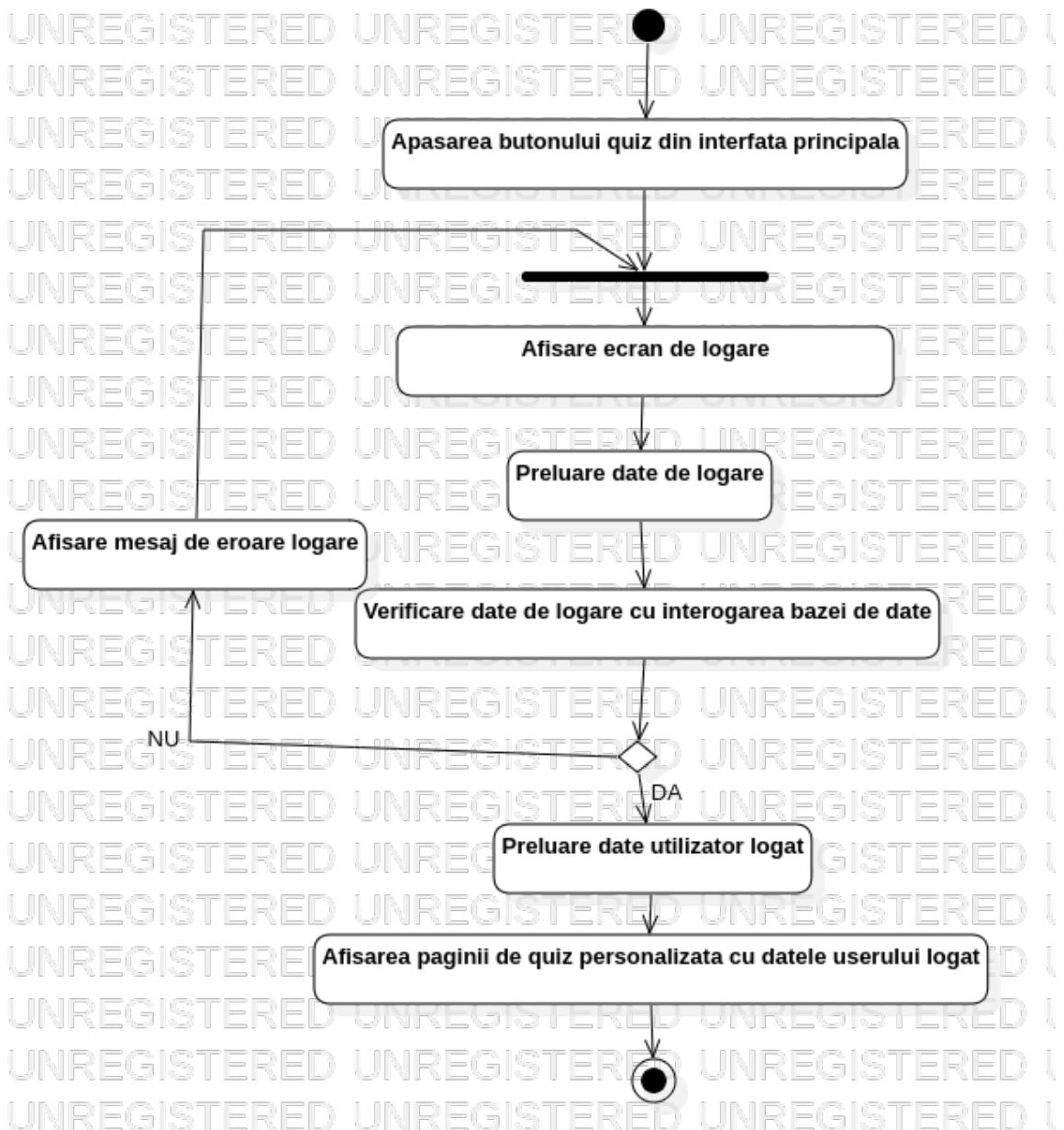
6. Diagrama încărcare cerc



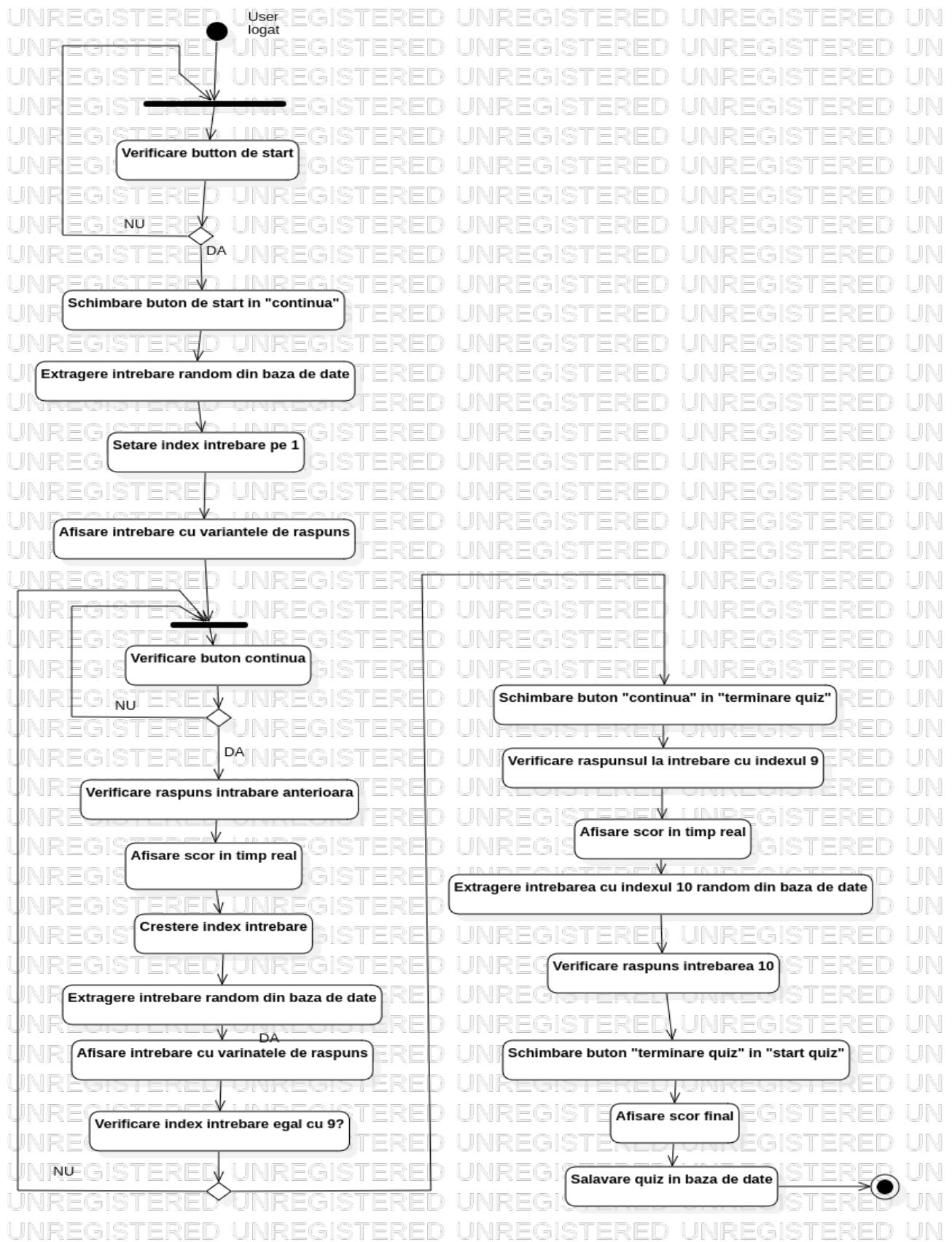
7. Diagrama autentificare



8. Diagrama logare



9. Diagrama quiz



10. Diagrama statistics

