



Soft Educational Geometria cercului

Proiect la disciplina Proiectare Software
2022

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. Arhitectura client-server.....	4
4.1 Realizarea conexiunii.....	4
5. Diagrame UML.....	5
5.1 Diagramele Use case Aplicația Client.....	5
5.2 <i>Diagramele Use case Aplicația Server</i>	9
5.3 Diagramele de clasă Aplicația Client.....	9
5.4 Diagramele de clasă Aplicația Server.....	13
6. Database și design pattern-ele alese.....	14
7. Descrierea aplicației.....	15
7. Poze de prezentare.....	15
Anexă diagrame de activități client.....	22
1. Diagrama desenare cerc.....	22
2. Diagrama desenare poligon.....	23
3. Diagrama desenare triunghi.....	24
4. Diagrama schimbare limba.....	24
5. Diagrama salvare cerc.....	25
6. Diagrama încărcare cerc.....	25
7. Diagrama autentificare.....	26
8. Diagrama logare.....	27
9. Diagrama quiz.....	28
10. Diagrama statistics.....	29
Anexă diagrame de activități server.....	30
1. Diagrama autentificare.....	30
2. Diagrama logare.....	31
3. Diagrama cerere pentru intrebare.....	32
4. Diagrama cerere data set-uri grafice.....	32
5. Diagrama salvare quiz.....	33
Anexă diagrame de secvență.....	34
1. Diagrama quiz.....	34
2. Diagrama autentificare.....	35
3. Diagrama logare.....	35
4. Diagrama desenare cerc.....	36
5. Diagrama desenare poligon.....	36
6. Diagrama desenare triunghi.....	37
7. Diagrama salare desen.....	37
8. Diagrama incarcare desen.....	37
9. Diagrama schimbare limba.....	38
10. Diagrama view statistics.....	38

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 aleator dintr-un set de 50 și în funcție de dificultate. Rezolvarea acesteia presupune trecerea prin etapele de analiză, proiectare și implementare a softului respectând arhitectura **client-server** și sablonul arhitectural **MVC** pentru aplicația client. Obiectivul temei este acela de familiarizare cu diferite şablonane arhitecturale, în cazul de față **client-server** și **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 de obținere a sesiunii. Conexiunea dintre client și server va fi intermediată printr-un **proxy** client, folosirea acestui design pattern va ajuta la filtrarea clienților conectați la server. Un alt design pattern folosit este **Command** pentru a descrie comportamentul aplicației client în partea de desenare.

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 și a celor de secvență 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 server.

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 IntelliJi 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/educationalsoftclientserver.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 și a unui utilizator by default în script-urile sql ce vor fi rulate de către Flyway.

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. Arhitectura client-server

Aplicația de față a fost structurată după modelul arhitectural **client-server**. Astfel, vor exista două aplicații, una ce se ocupă de partea de server, iar celalătă reprezintă partea de client. Arhitectura client / server este un model de calcul în care serverul găzduiește, livrează și gestionează majoritatea resurselor și serviciilor care urmează să fie consumate de client. Serverul găzduiește și oferă la cerere servicii de înaltă performanță, consumatoare de calcul. În cazul aplicație de soft educațional, serverul se va ocupa de gestionarea resurselor prezente la nivelul bazei de date: accesul la utilizatorii autenificați în aplicație și acces la quiz-urile și întrebările prezente în baza de date necesare testării utilizatorilor. Tot partea de server se va ocupa de relaizarea calculelor necesare pentru determinarea datelor set-urilor folosite pentru generarea graficelor statistice, calcule de costisitoare.

4.1 Realizarea conexiunii

Modelul de comunicație orientat pe conexiune. Acest model se bazează pe protocolul **TCP**. Într-o aplicație rețea întotdeauna avem două părți: o parte client care inițializează conversația și trimite cereri, și o parte server care primește cererile și răspunde la acestea. Clientul întotdeauna crează un socket pentru a iniția conversația și trebuie să cunoască serverul căruia adresează cererea, iar serverul trebuie să fie pregătit pentru a receptiona aceste cereri. În momentul recepționării mesajului crează un socket pe partea serverului, socket care va facilita deservirea clientului. Atât pe partea de client cât și pe partea de server se utilizează câte un obiect de tip **Socket** pentru

comunicare. Pe partea de server mai trebuie să creăm un obiect de tip **ServerSocket**, care are sarcina primirii conexiunilor și acceptarea acestora. Clientul trebuie să cunoască două lucruri despre server: *numele serverului* (utilizat pentru determinarea adresei IP al serverului) și *numărul portului* la care acesta ascultă cererile clientilor. Același calculator gazda poate oferi mai multe servicii, deci poate gazdui mai multe procese de tip server. [1]

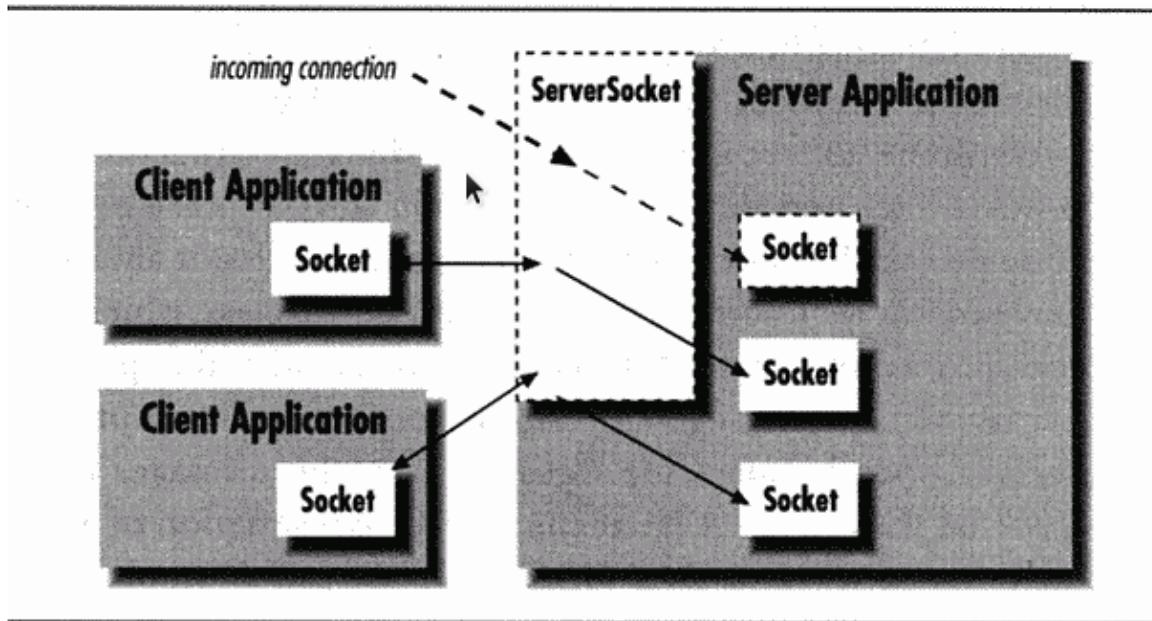


Figure 1: Client-server via Sockets

5. Diagrame UML

5.1 Diagramele Use case Aplicația Client

În figura de mai jos sunt sintetizate grafic posibilele interacțiuni ale utilizatorului cu sistemul creat. Accesul la tool-ul de desenare îl are orice tip de utilizator fără un proces de autentificare și logare anterior, dar pentru partea de quiz este nevoie de logare în aplicație.

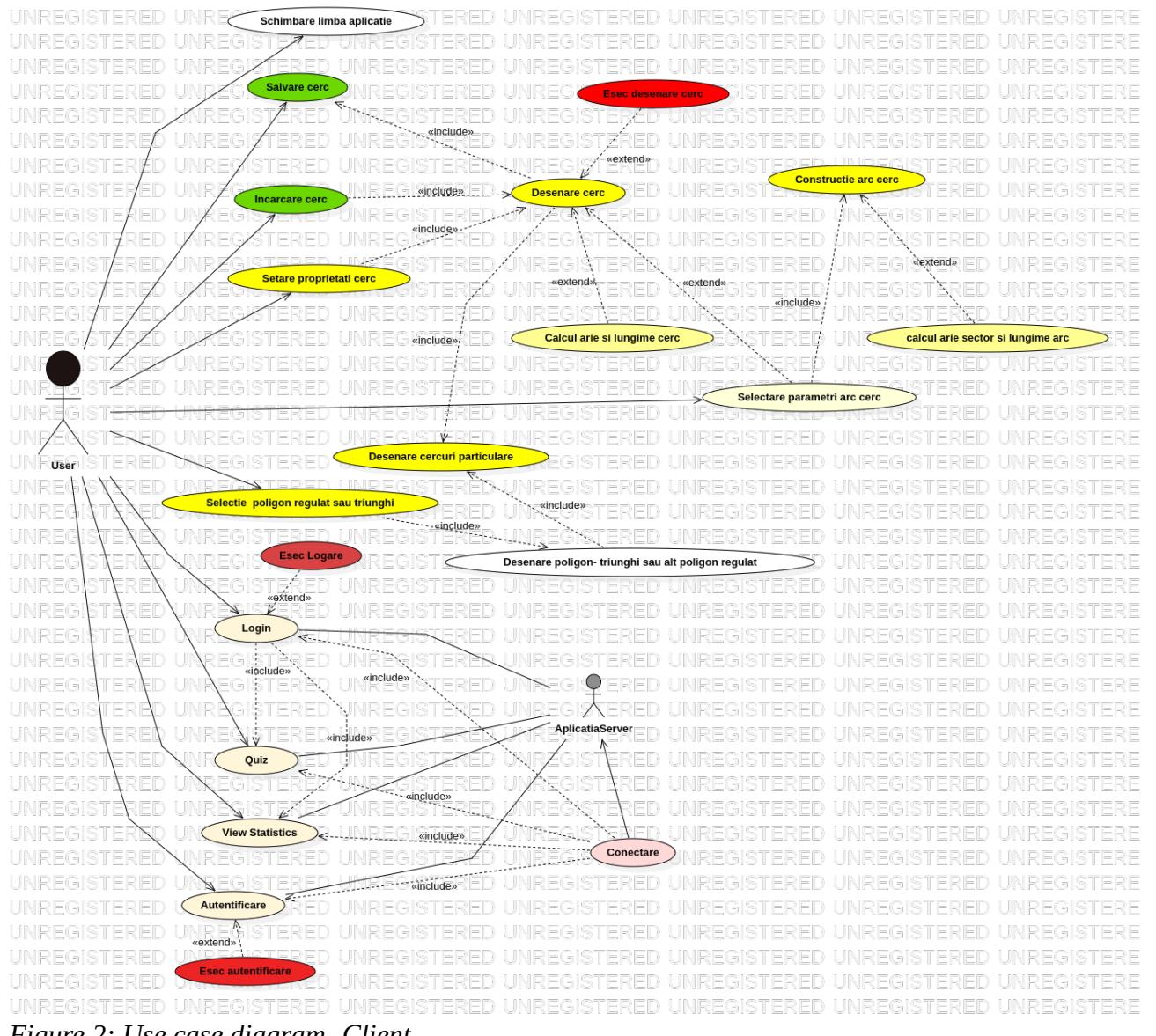


Figure 2: Use case diagram -Client

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 punctata arcul de cerc și se va calcula în timpul desenării Lungimea arcului și Arealui 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
- Aplicația client va trimite requestul de logare spre server
- Serverul va interoga baza de date și va return clientul datele complete ale utilizatorului
- Aplicația client va deschide o fereastră nouă cu mesajul de întâmpinare a quizului

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

- Aplicația client va trimite requestul de autentificare cu informațiile introduse de utilizator serverului
- Serverul va analiza datele introduse și va interoga baza de date cu privire la existența username-ului numai dacă datele au fost validate
- Serverul va crea un nou utilizator pe care îl va salva în baza de date
- Serverul va transmite mesajul de creare aplicației client
- Aplicația client va afișa un mesaj de întampinare și va deschide fereastra aferentă quiz-ului

Eșec:

- Utilizatorul introduce date invalide: username sau/și parolă care nu corespund validatorilor sau există deja un cont cu username-ul introdus
- Afișare mesaj de eroare cu informațiile corespunzătoare
- Utilizatorul poate introduce noi date pentru autentificare(crează 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 afișate pe ecran pe rând
- Aplicația client va cere la fiecare pas server-ului o nouă întrebare, pe care acesta din urmă o va extrage din baza de date
- La întrebarea cu index 10 utilizatorul apasă pe butonul *Finish quiz* și aplicația client va transmite serverului informațiile necesare salvării quizului în baza de date
- Se vor afișa 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**
- Aplicația client va cere data set-urile necesare generării graficelor serverului
- Serverul extrage din baza de date valorile corespunzătoare tipului de analiză la nivel statistic
- Serverul trimite datele obținute și prelucrate clientului
- Se vor afișa 3 grafice: pie chart cu gruparea zuițurilor pe punctaje obținute, bar chart cu punctajele medii ale tuturor utilizatorilor, time chart cu evoluția zuițurilor utilizatorului

Use case: schimbare limbă

Actor principal: orice utilizator

Principalul scenariu de succes:

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

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

5.2 Diagramele Use case Aplicația Server

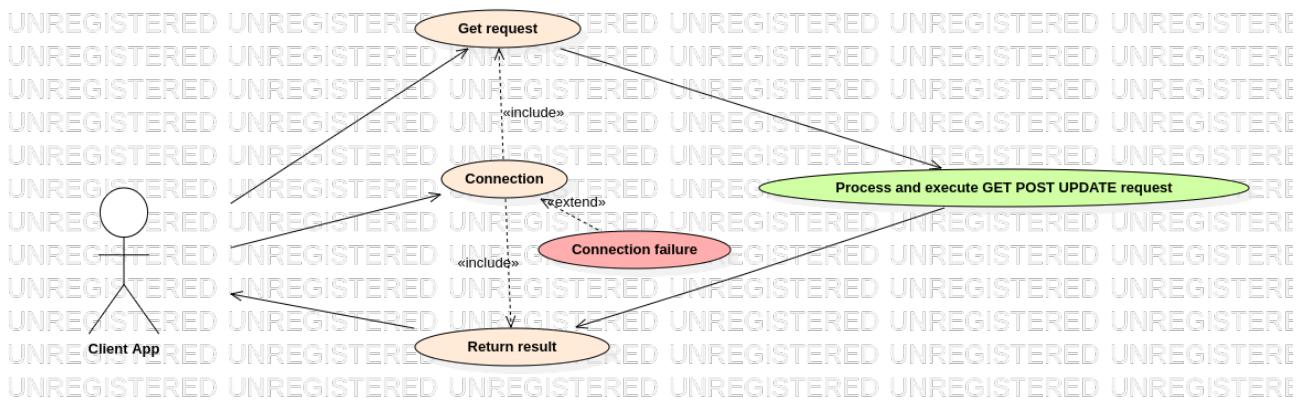


Figure 3: Use Case Diagram Server App

Pentru ca aplicația client să comunice cu aplicația server este nevoie de conexiune ce se realizează pe principiul descris mai sus. La nivelul aplicației de soft educațional, descrisă în lucrarea de față, clientul cere aplicației server date ce se găsec la nivelul bazei de date. Gestionarea și accesul la informațiile cu privire la utilizatorii autentificați, quizurile date de aceștia și întrebările pentru testare se fac de către server. De asemenea, server-ului îi sunt rezervate și calculele statistiche pentru determinarea dată set-urilor graficelor create în partea de client.

5.3 Diagramele de clasă Aplicația Client

Proiectarea aplicației client are în vedere respectarea principiilor SOLID și a designului arhitectural *Model-View-Controller*. Clasele au fost grupate după cum se observă în *figura 5* 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.

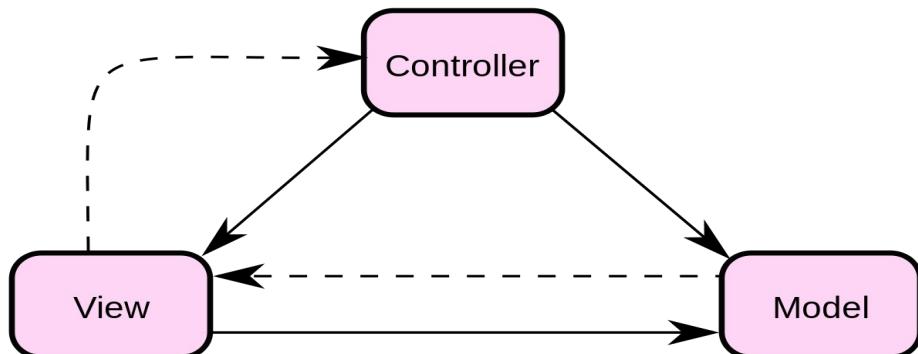


Figure 4: Model-View-Contorller

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 acesteia. 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
- S-a folosit o clasă abstractă **OperationCircleProperties** care va fi moștenită de clase specializezate 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 încarcă 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**

- 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.

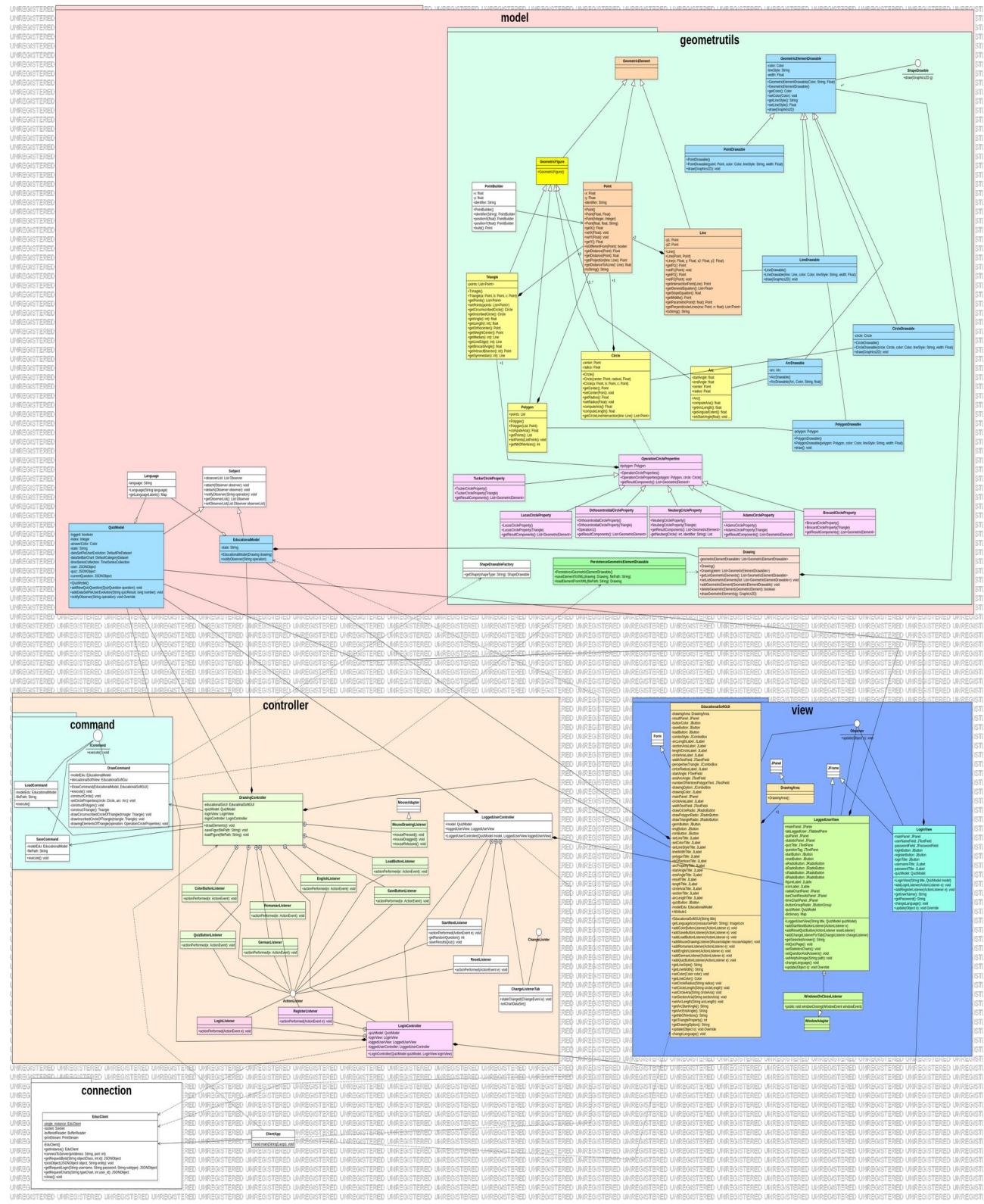


Figure 5: Diagrama de clase -Client

Clasele din **controller**:

- **DrawingController** se ocupă de gestionarea funcționalității de desenare a figurilor geometrice. Acesta este format din inner classes, ce pot fi văzute în *figura 5*, 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. Comportamentul de determinare a desenului ce are să fie afișat pe interfața grafică este dat de clase specialize, numite Command. Clasele **DrawCommand**, **SaveCommand** și **LoadCommand** vor implementa interfața *Icommand*, suprascriind metoda *execute()*;
- **LoginController** această clasă se va ocupa de partea de logare în aplicație prin cererea de la server a existenței contului dat de către utilizator în baza de date. Se creează de către client un obiect de tipul JSON în care se specifică tipul și scopul cererii și câmpurile date de utilizator(username și parolă), și va fi transmis către server. Serverul va verifica în baza de date existența contului și va returna clientului tot un obiect de tip JSON cu informațiile relevante ale contului. În momentul în care se primește acceptul de server, atunci în login controller sa va actualiza starea obiectului **quizModel** cu valorile curente ale userului logat și se va deschide fereastra aferentă quizului.
- **LoggedUserController** se va ocupa de crearea cererilor și prelucrarea răspunsurilor venite de la server cu privire la accesul întrebărilor pentru quiz, dar și pentru generarea statisticilor. În login controller va fi definit automatul pentru generarea unui quiz. De asemnea se vor seta în model și datele statistice necesare generării graficelor venite de la server.

Clasele din **connection**:

- În connection este definită o clasă **EduClient** ce se va ocupa de conexiunea la server. Aceasta este implementată cu designul patternul **singleton** pentru a realiza doar o singură conexiune la server și aceasta să rămână activă pe tot parcursul rulării aplicației client (dacă serverul este activ).
- În cadrul acesteia, sunt definite metode specifice cererilor de date pe care le dorește clientul. O cerere, este sub forma unui obiect JSON și are una din următoarele forme:
 - HEADER REQUEST{
 - "ENTITY": "quiz",
 - "OBJECT": {
 - "date": "2022-05-15T15:16:24.334323846",
 - "score": 60,
 - "user_id": 91
 - },
 - "TYPE": "POST"
 - }
- Cereare de mai sus este de tipul POST, pentru entitatea „QUIZ”. Astfel, în decodificarea request-ului serverul va ști să salveze în baza de date valoarea obiectului.
- Conexiunea la server se va realiza prin socket-uri, prin protocolul **TCP**. În EduClient este definit un câmp Socket ce se va conecta la serverul **127.0.0.2**, ce ascultă la portul **5000**.

5.4 Diagramele de clasă Aplicația Server

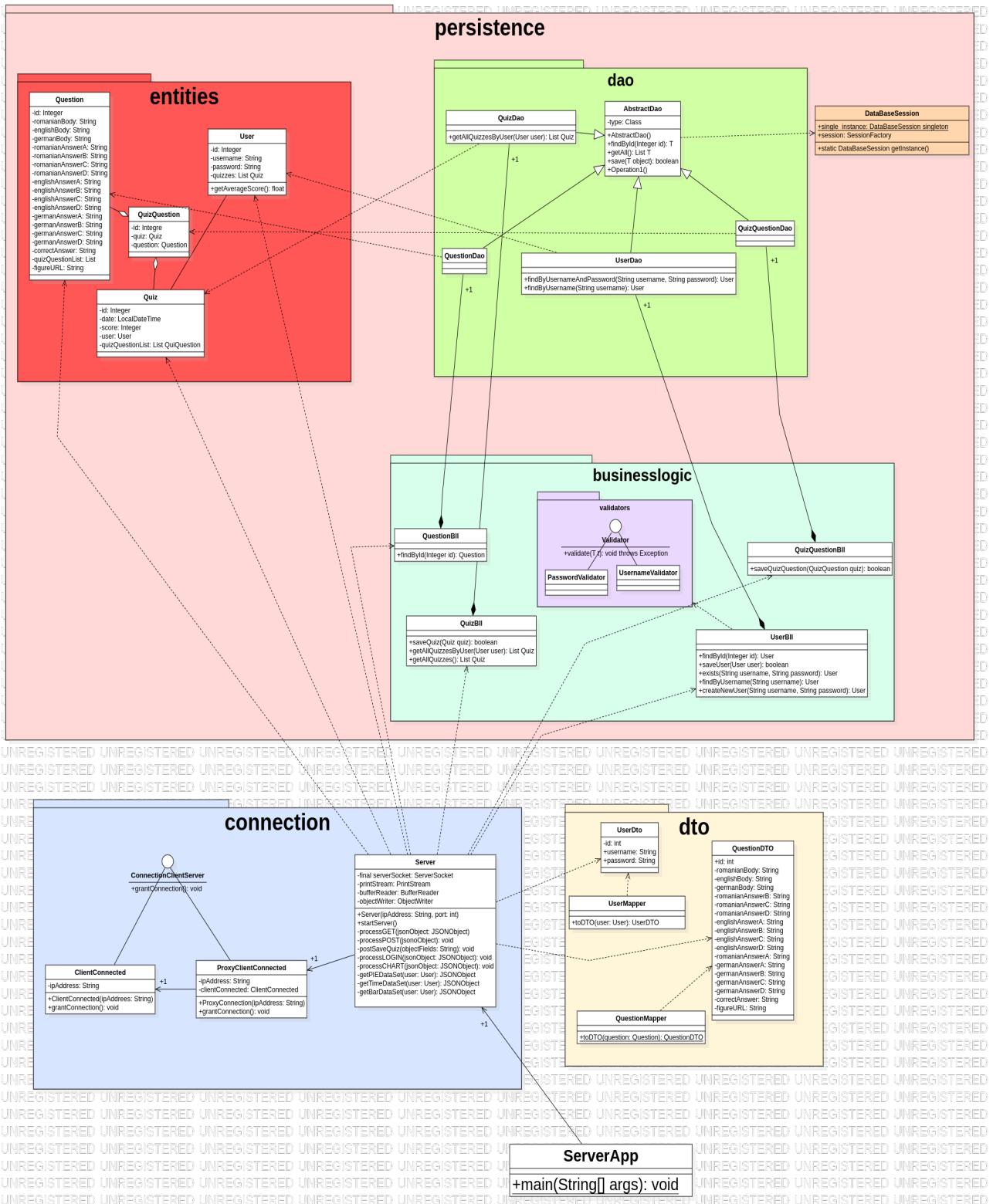


Figure 6: Diagram de clase Server

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 întrebări 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**.

Operațiile ce se realizează la nivelul bazei de date sunt definite în clasele de tip DAO. **AbstractDAO** descrie principalele operații CRUD, comune tuturor entităților. Accesul la aceste operații nu se va face direct, ci prin clasele speciale de **businesslogic** în care sunt definite și validări și constrângeri, înainte de a executa operații la nivelul bazei de date.

În pachetul connection este definită partea de crearea instanței de server (**ServerSocket**) care va fi bound la adresa 127.0.0.2 și ascultă pe portul 5000. În momentul în care se conectează un client la server, acesta va fi filtrat cu un obiect wrapper, definit prin design patternul **proxy**. Grantul conexiunii adevărate va fi dat doar în urma filtrării.

Serverul primește de la client request-uri sub formă de obiect JSON. Acesta decodifică cerearea și va procesa o acțiune specifică acesteia. Va trimite ca răspuns tot un obiect de tipul JSON.

Pentru a serializa valorile unui obiect sub formă de JSON, s-au creat clase DTO pentru entitățile **user** și **question** doar cu câmpurile relevante pentru aplicația client, și mapper-ele corespunzătoare.

6. Database și design pattern-ele alese

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.

Design pattern-uri prezente:

1. **Singleton** -- baza de date și o singură instanță de socket
2. **Proxy** – conexiune client server
3. **Builder** – la nivelul câmpurilor de la clasa Point; folosit și ca adnotarea **Lombok** pentru entități și dto-uri.
4. **Command** – design pattern comportamental folosit operațiile de desenare și persistență desen
5. **Observer** - pentru notificarea interfețelor grafice în momentul schimbării stării unui obiect
6. **Factory methods** – pentru obiectele/shape-urile de desenat (de tip drawable)

7. 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

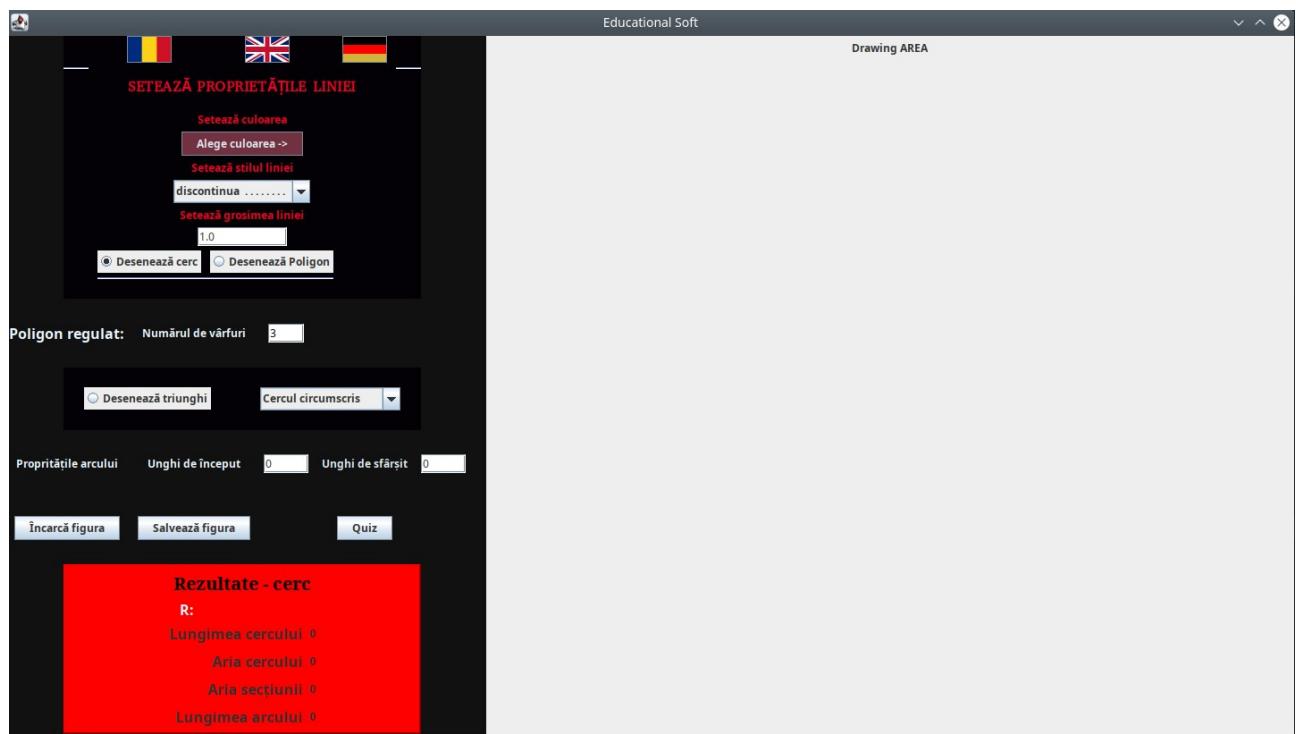


Figura 4 Pornirea Aplicatiei

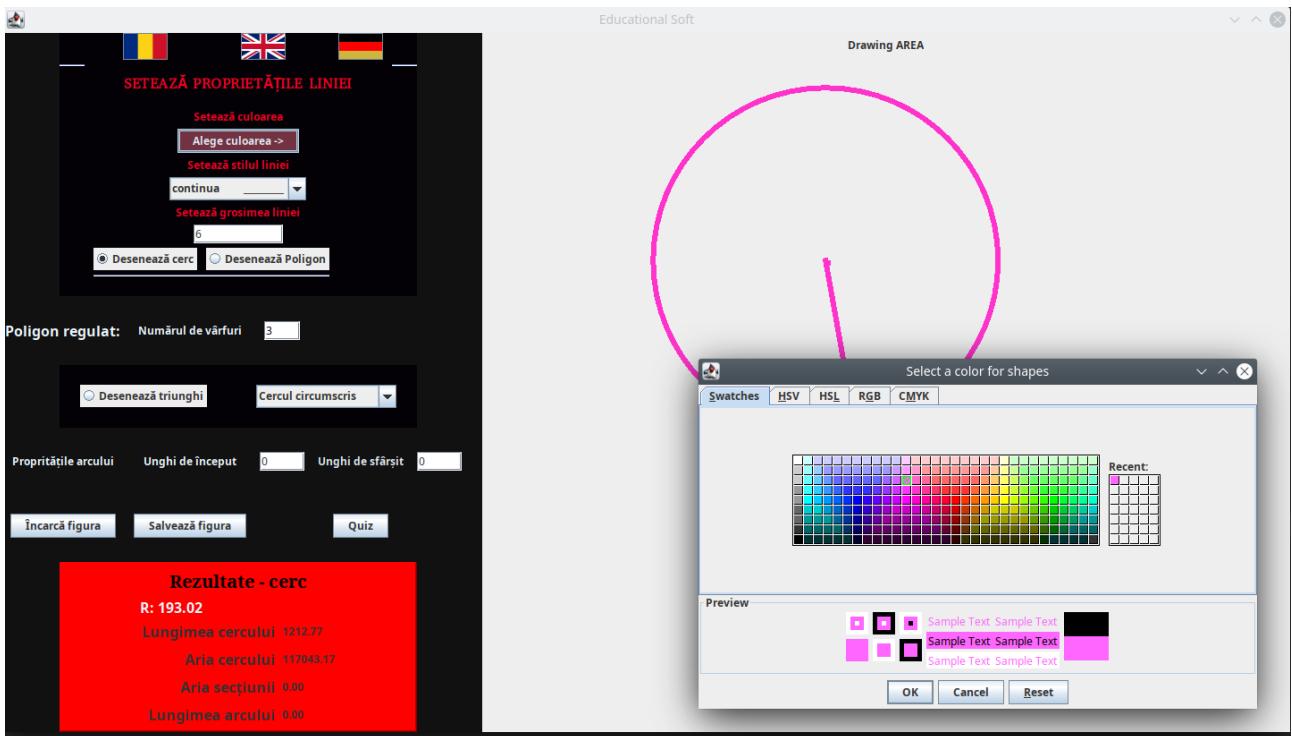


Figura 5 Setare stil desenare

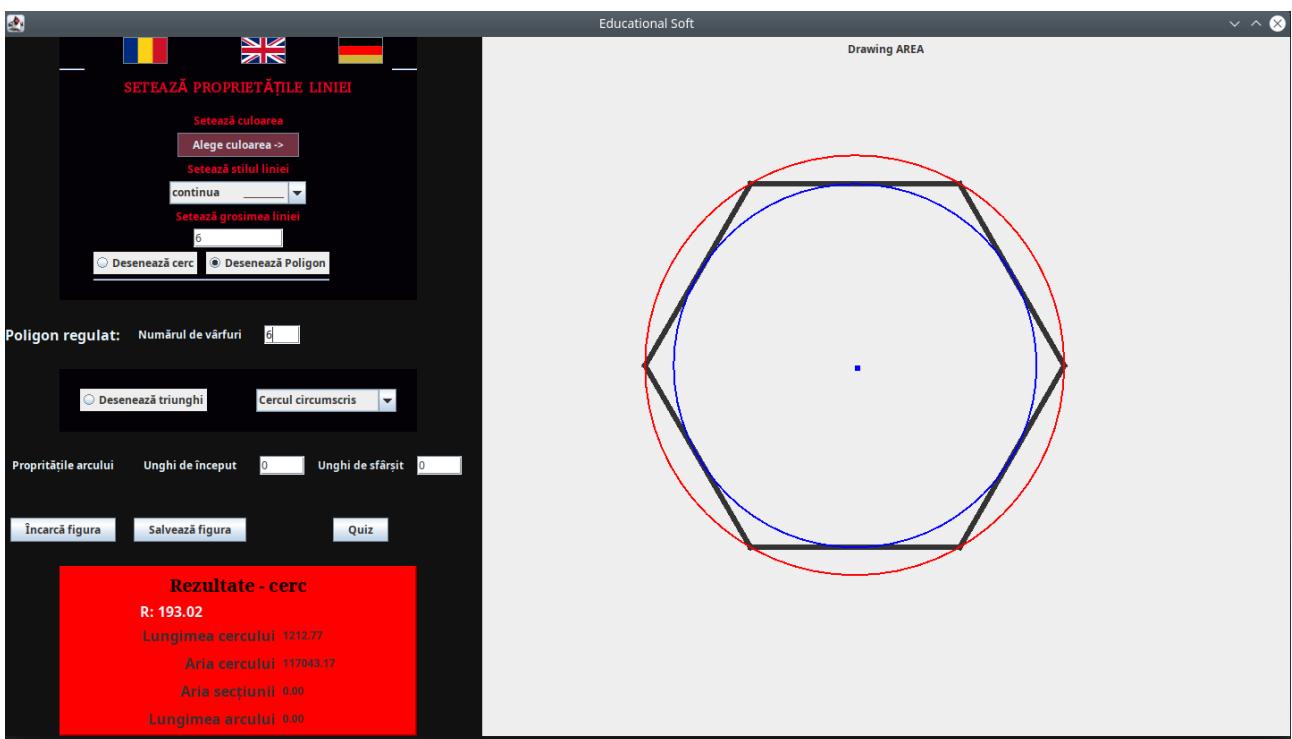


Figura 6 Desenare poligon cu cercul incris si circumscris

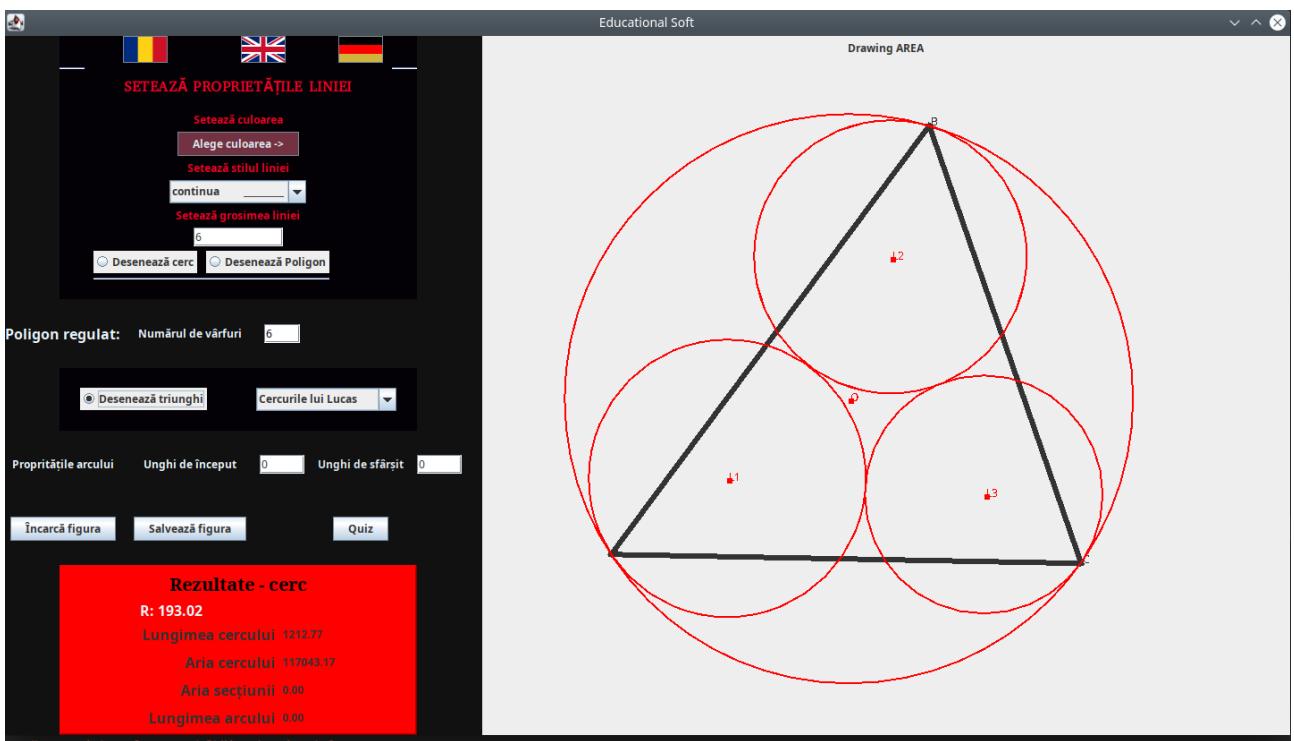


Figura 7 Cercurile lui Lucas

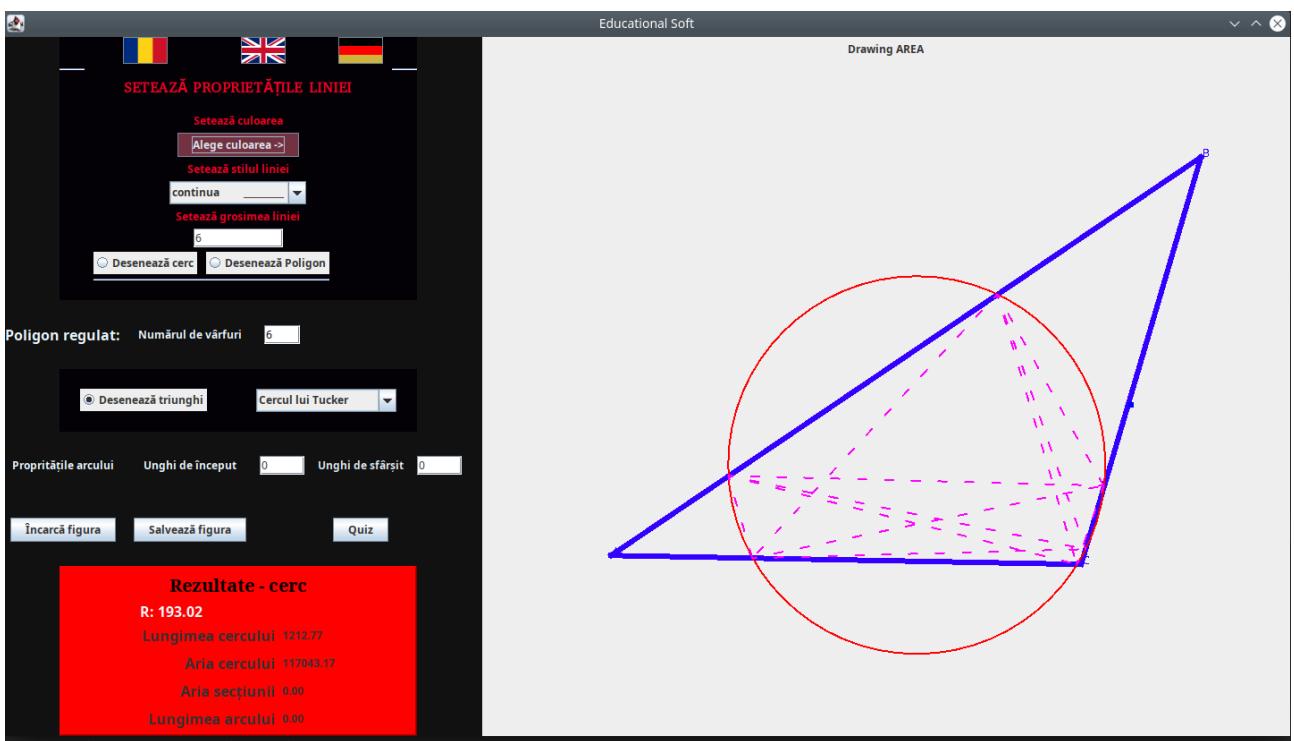


Figura 8 Cercul lui Tucker

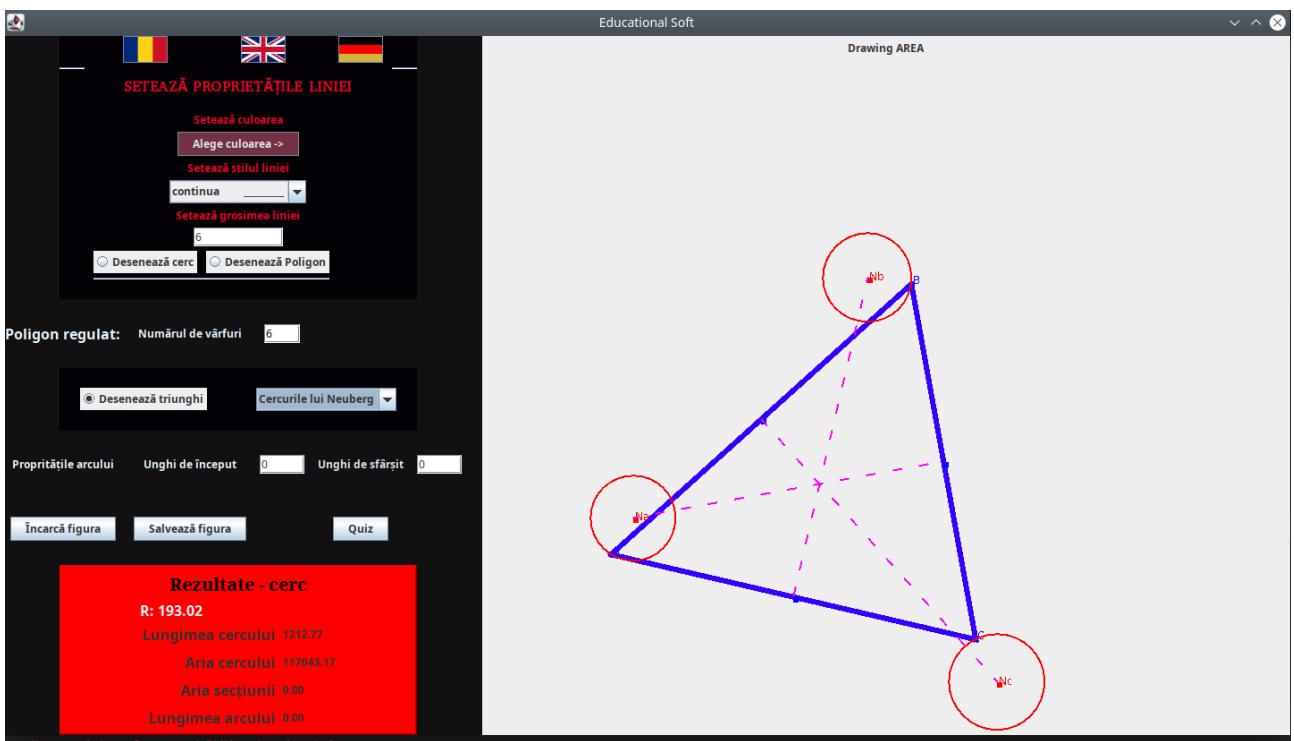


Figura 9 Cercurile lui Neuberg

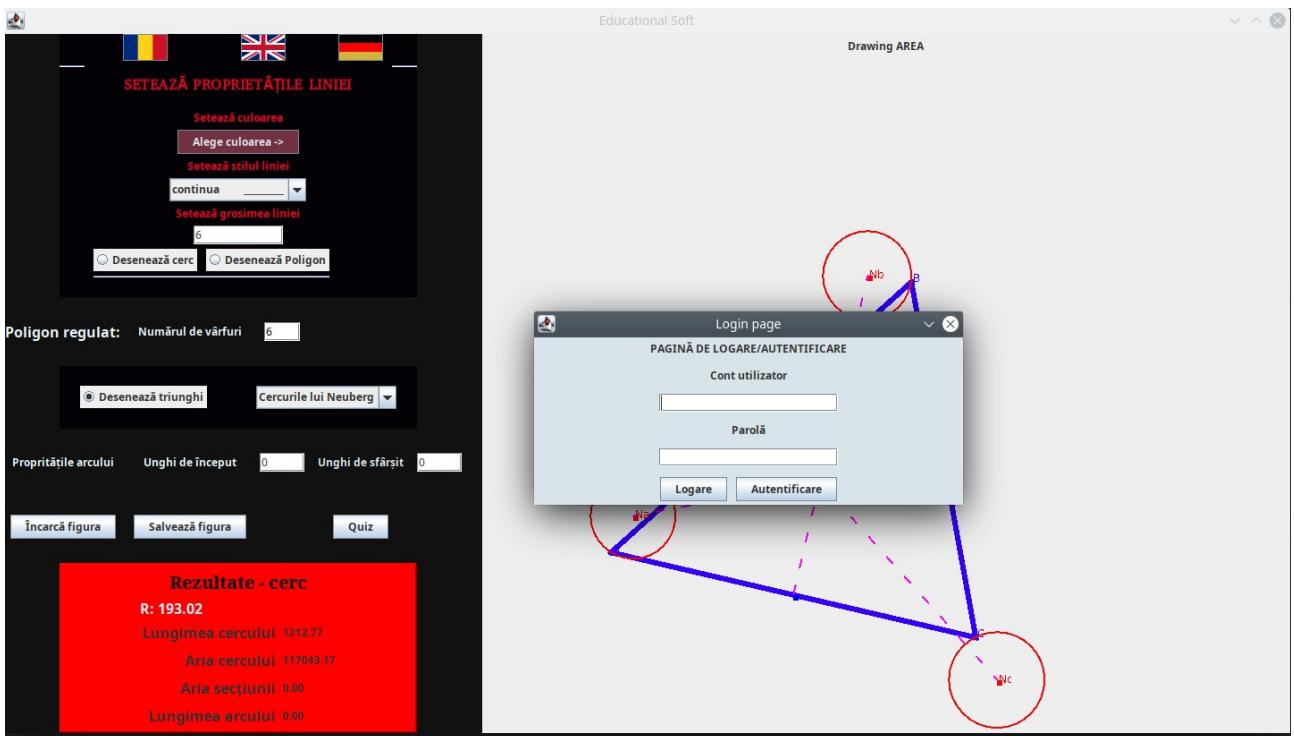


Figura 10 Logare/ autentificare

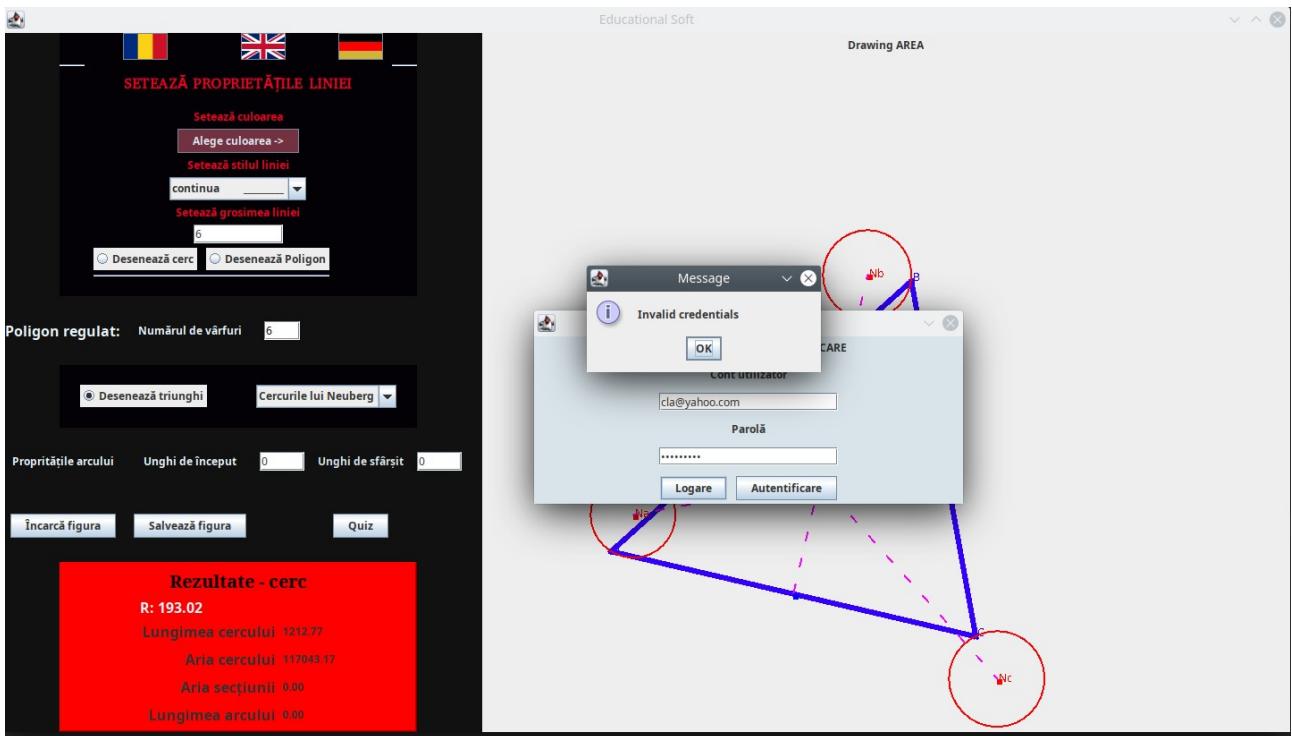


Figura 11 Failed login

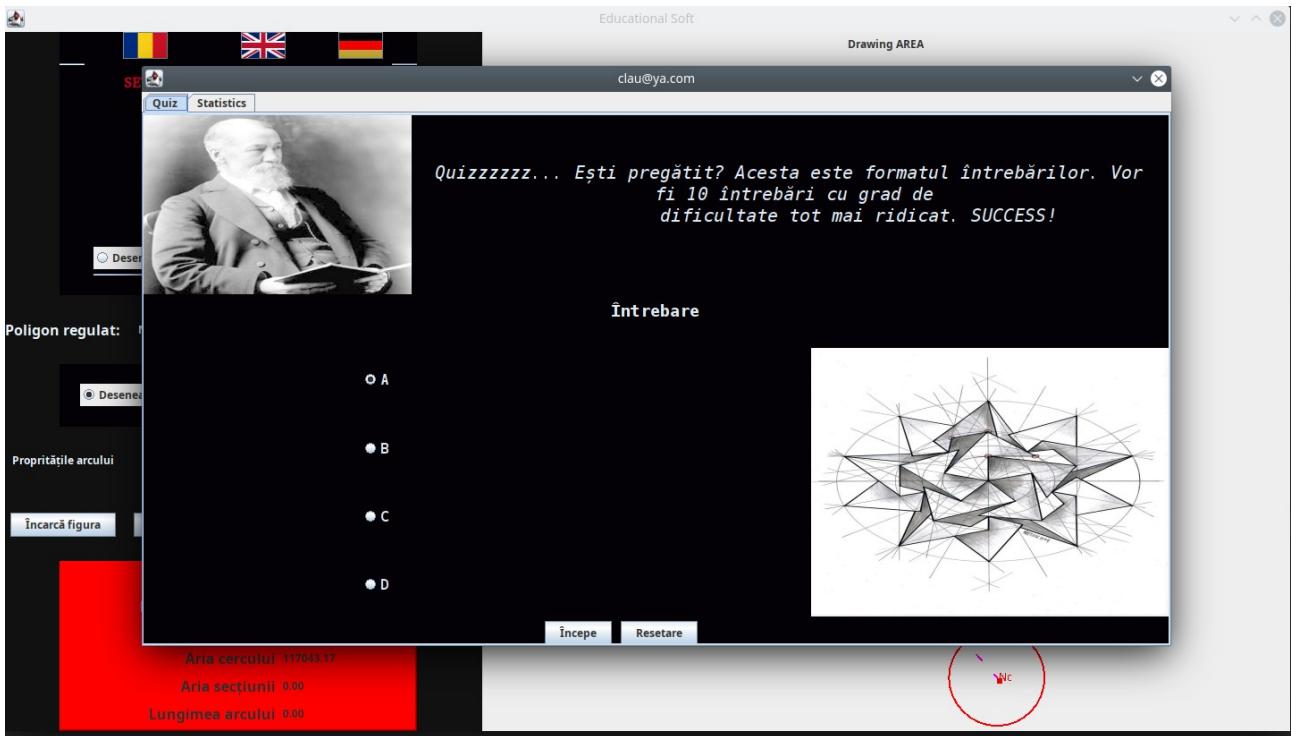


Figura 12 Pagina de start quiz

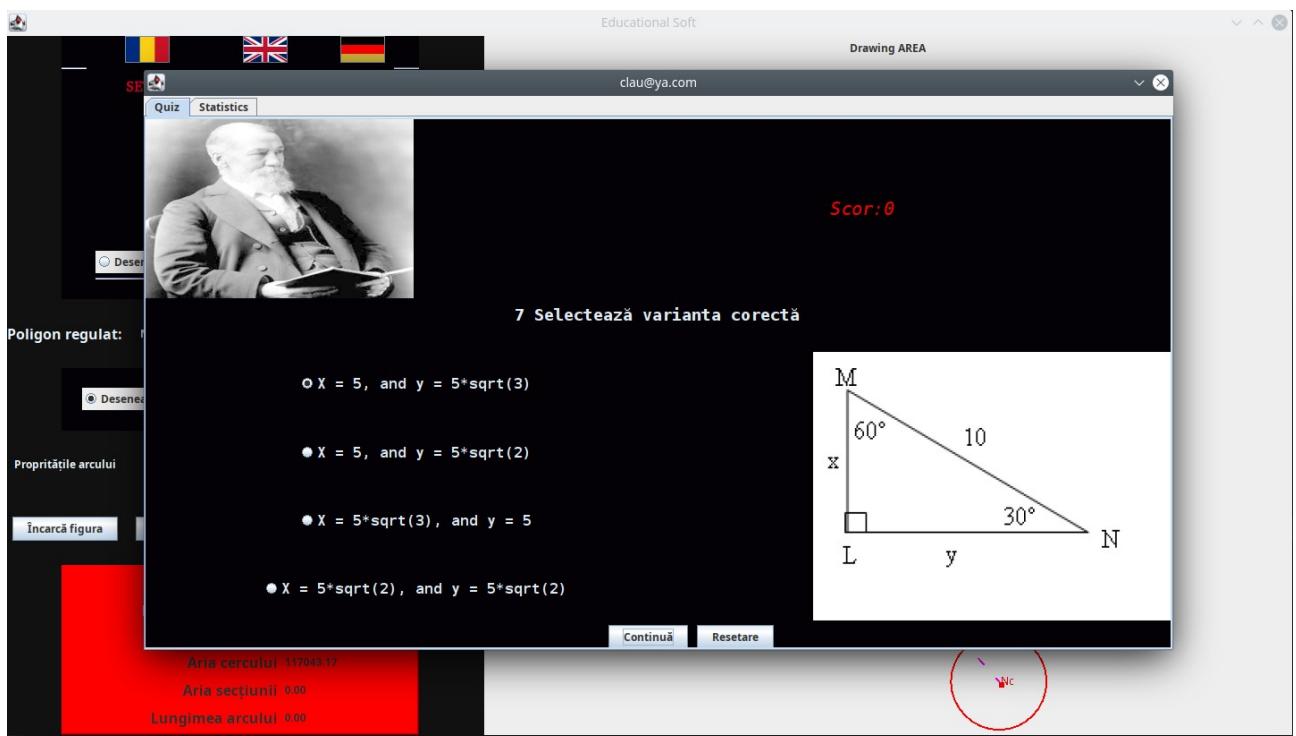


Figura 12 Intrebare quiz cu imagine alaturata

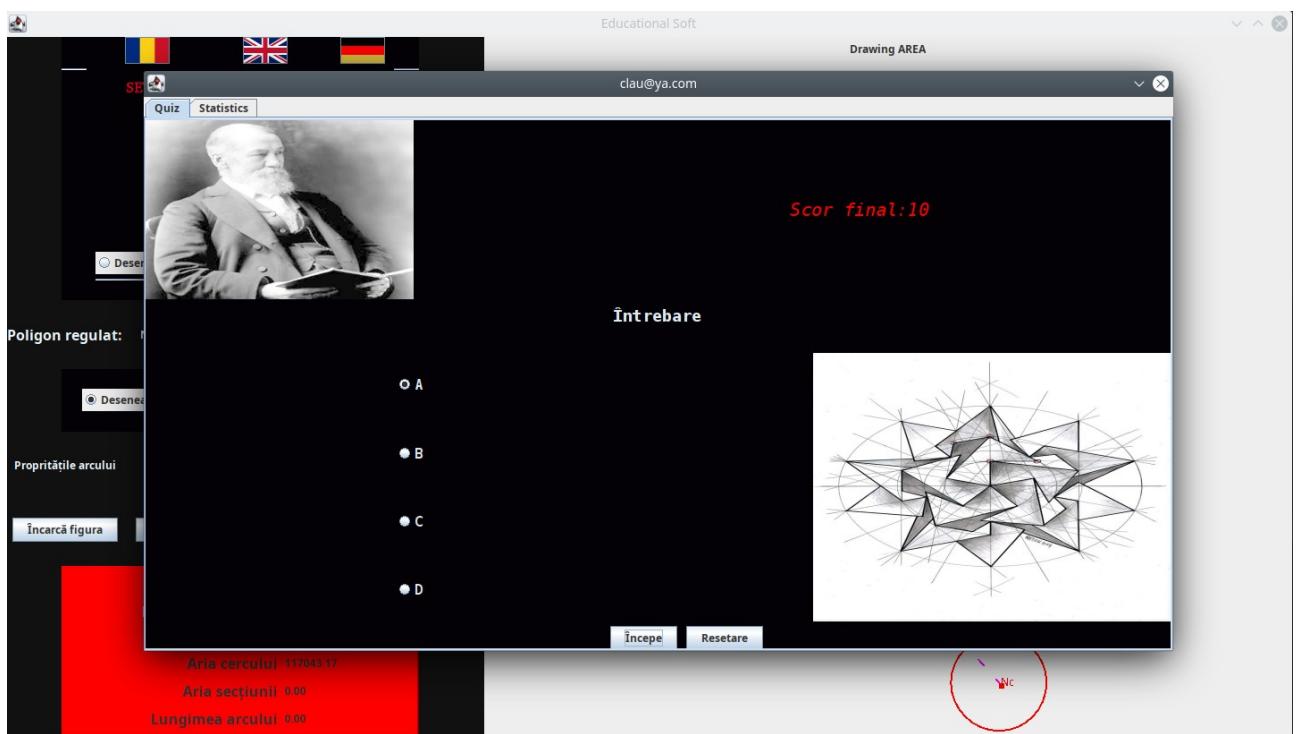


Figura 13 Scor final quiz

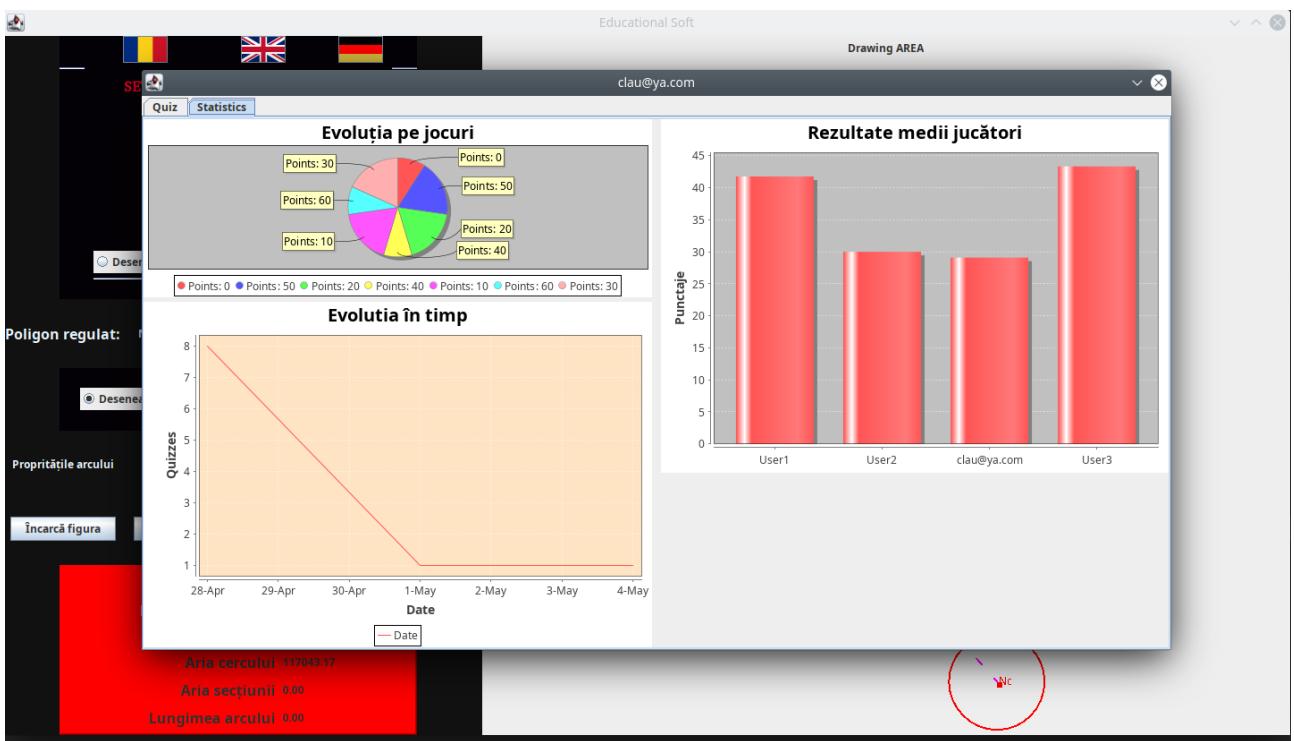
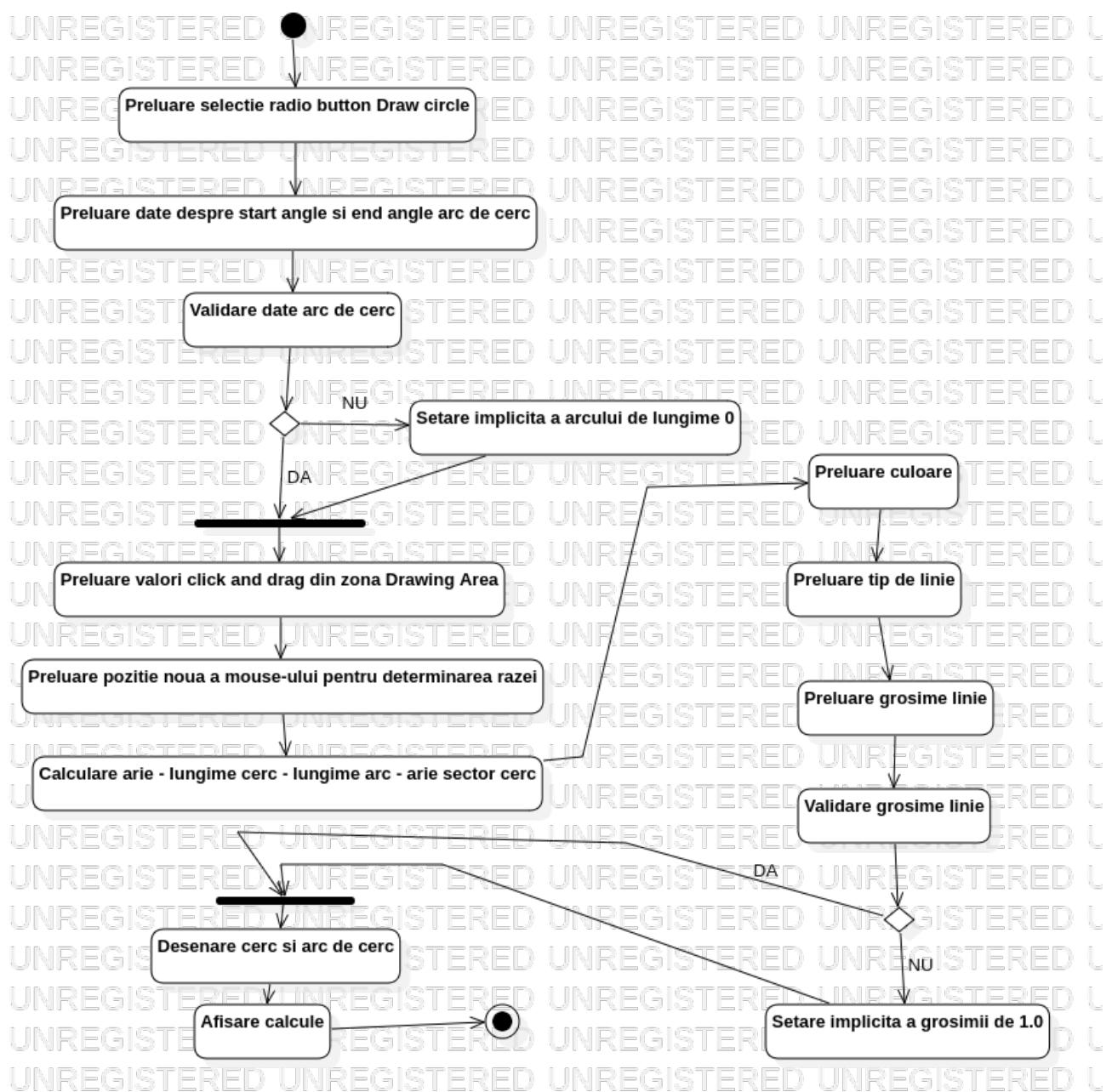


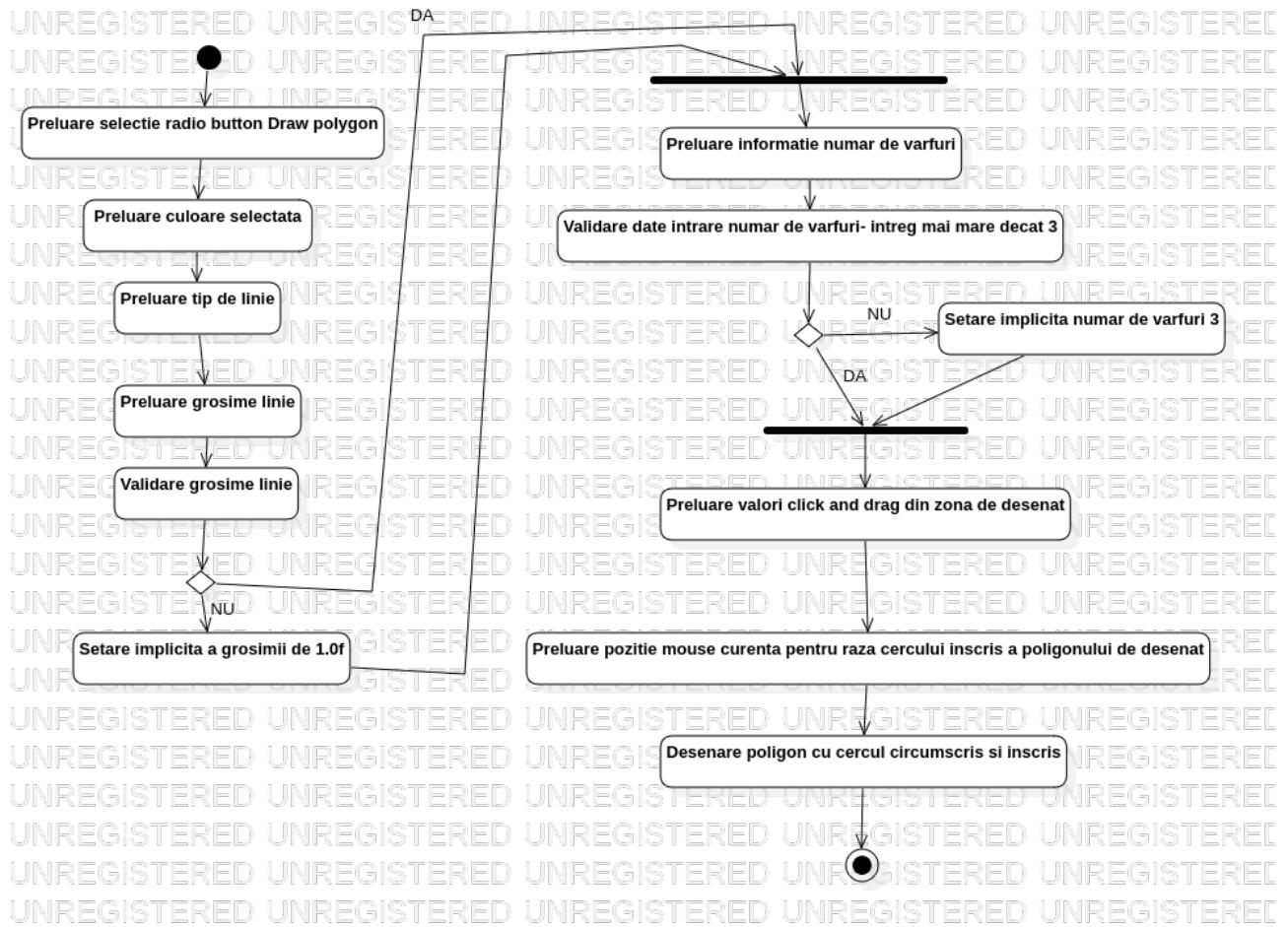
Figura 14 Scor final quiz

Anexă diagrame de activități client

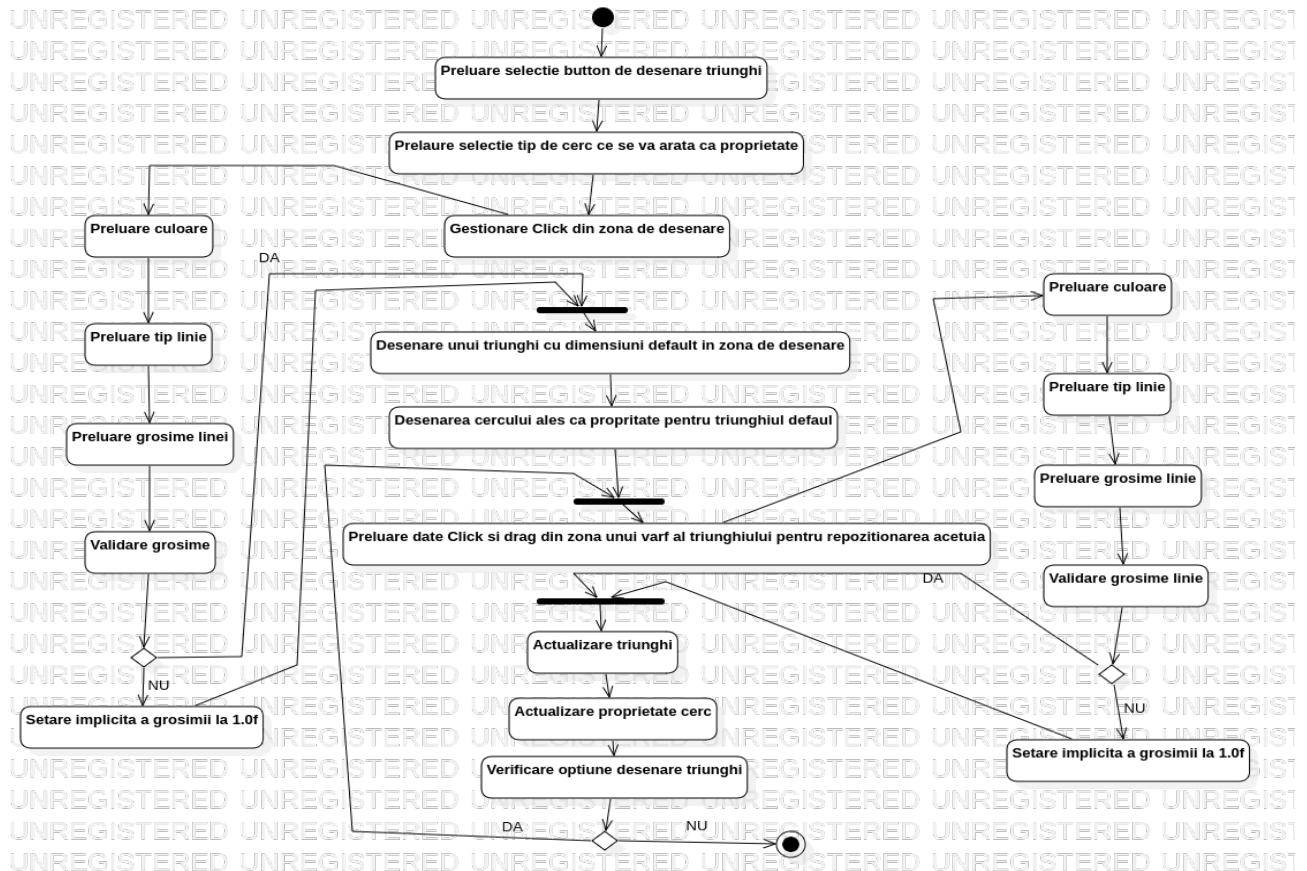
1. Diagrama desenare cerc



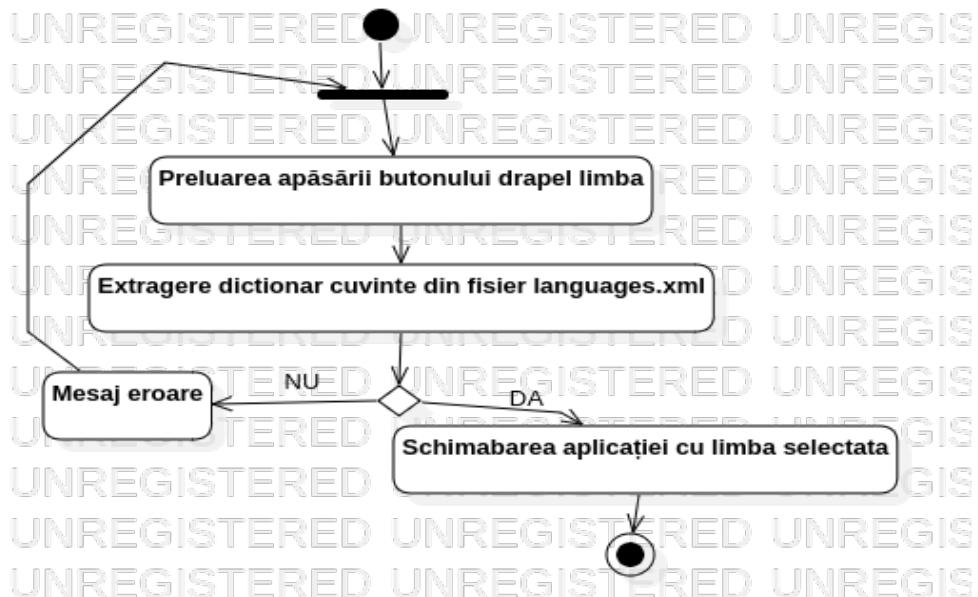
2. Diagrama desenare poligon



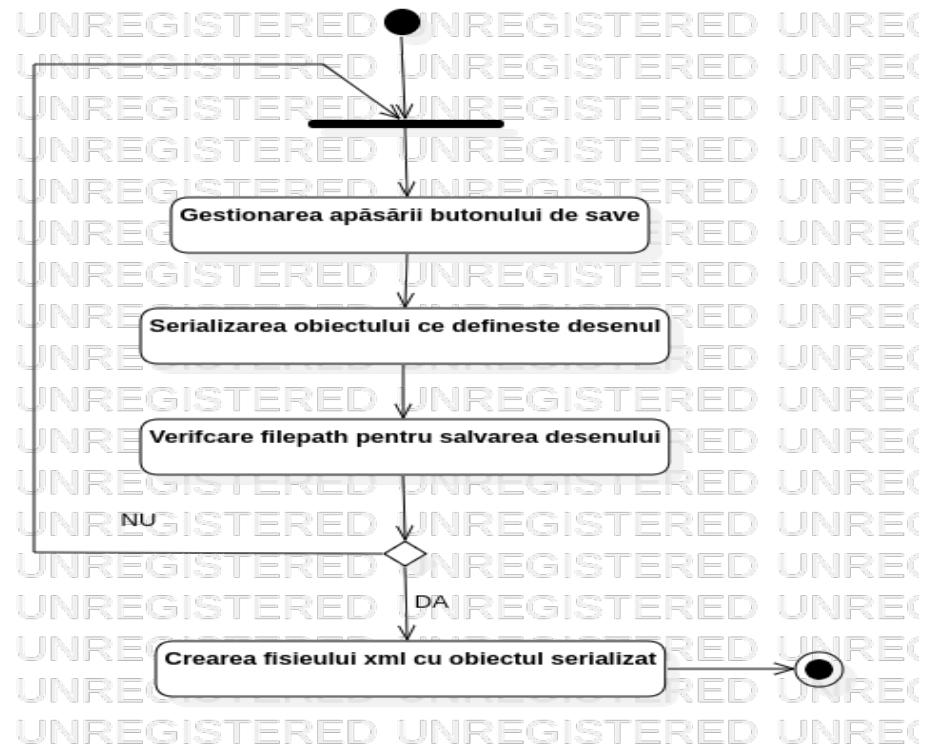
3.Diagrama desenare triunghi



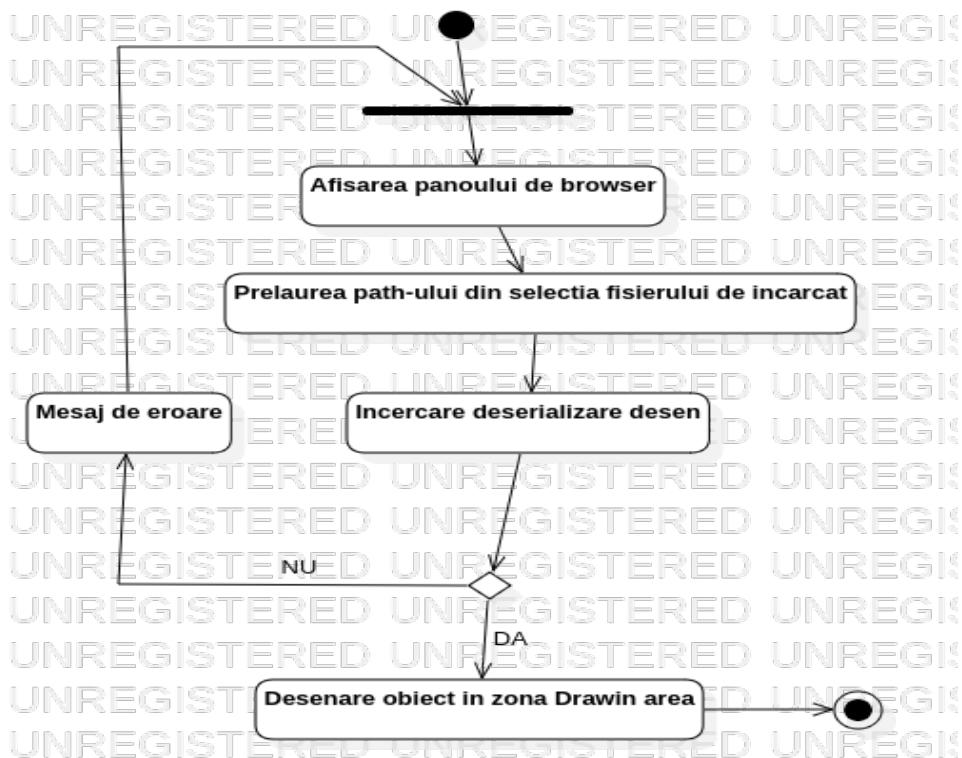
4.Diagrama schimbare limba



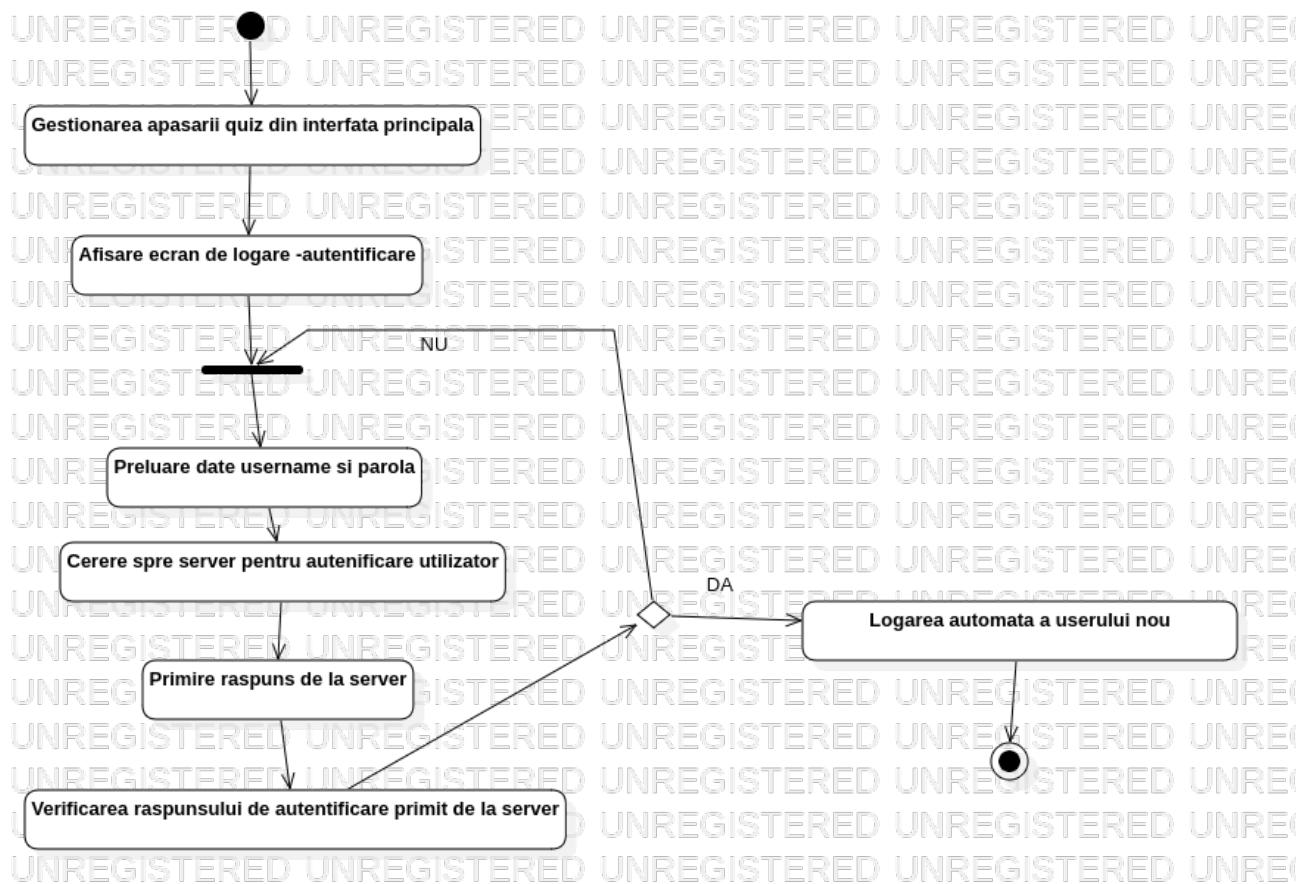
5.Diagrama salvare cerc



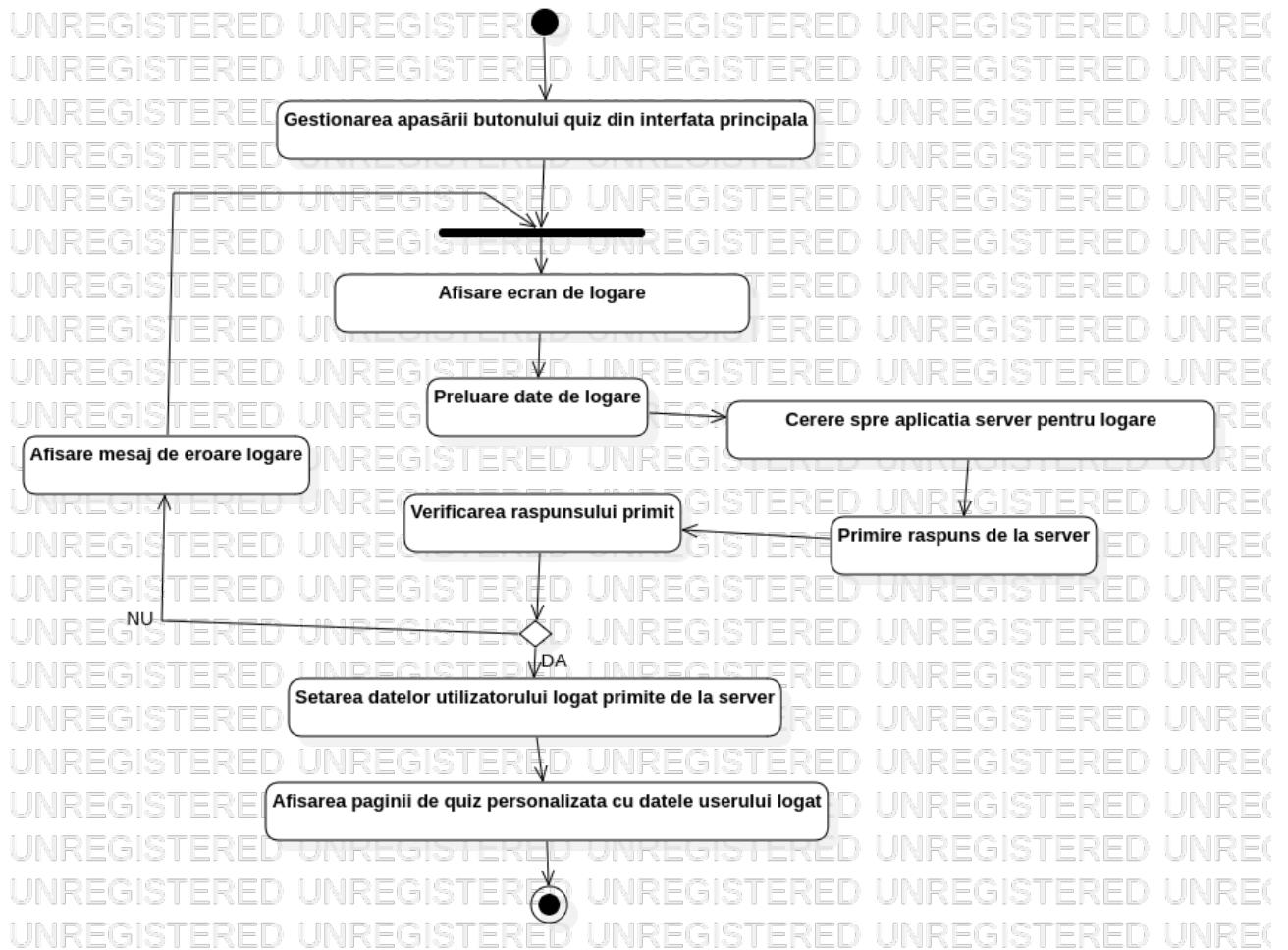
6.Diagrama încărcare cerc



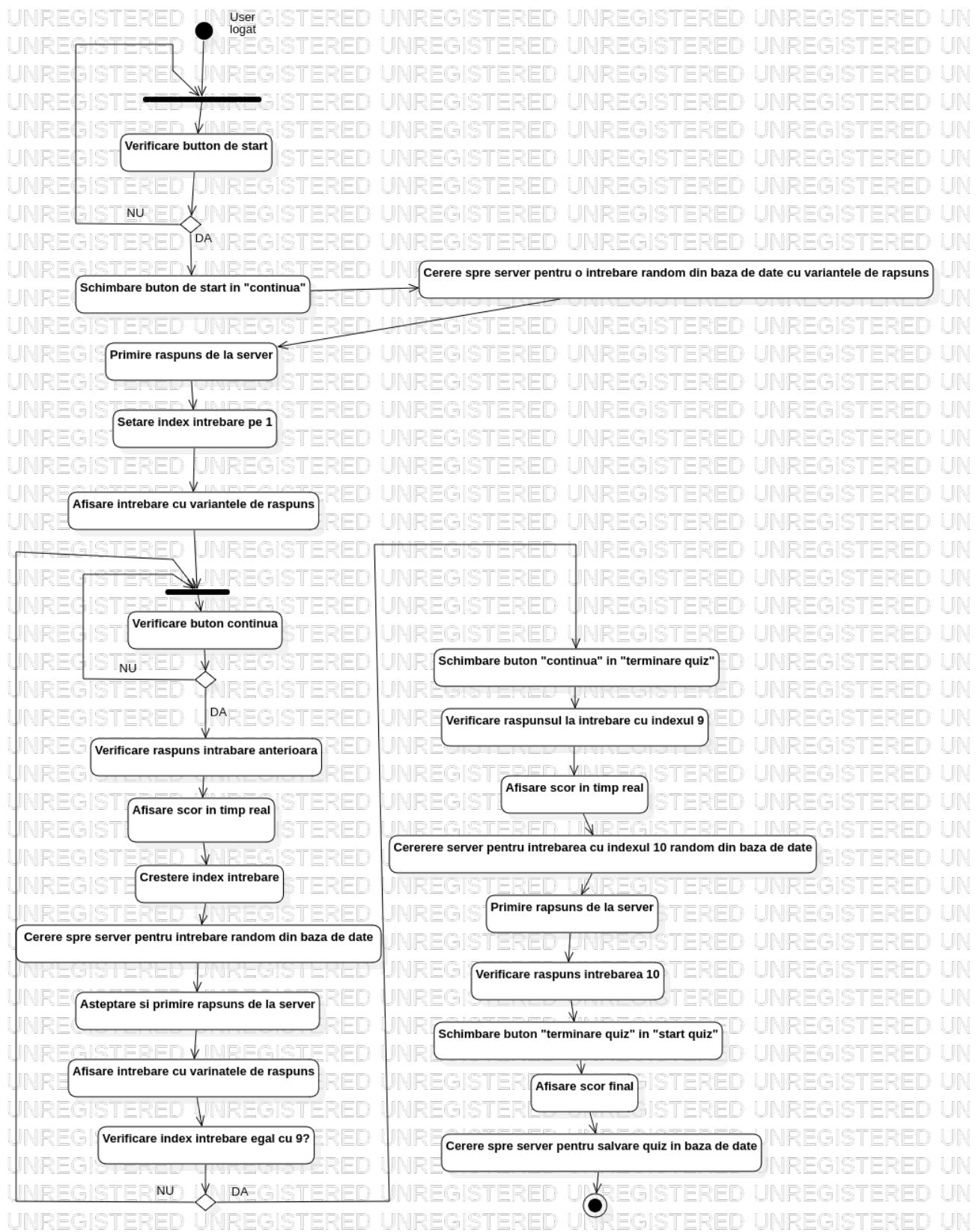
7. Diagrama autentificare



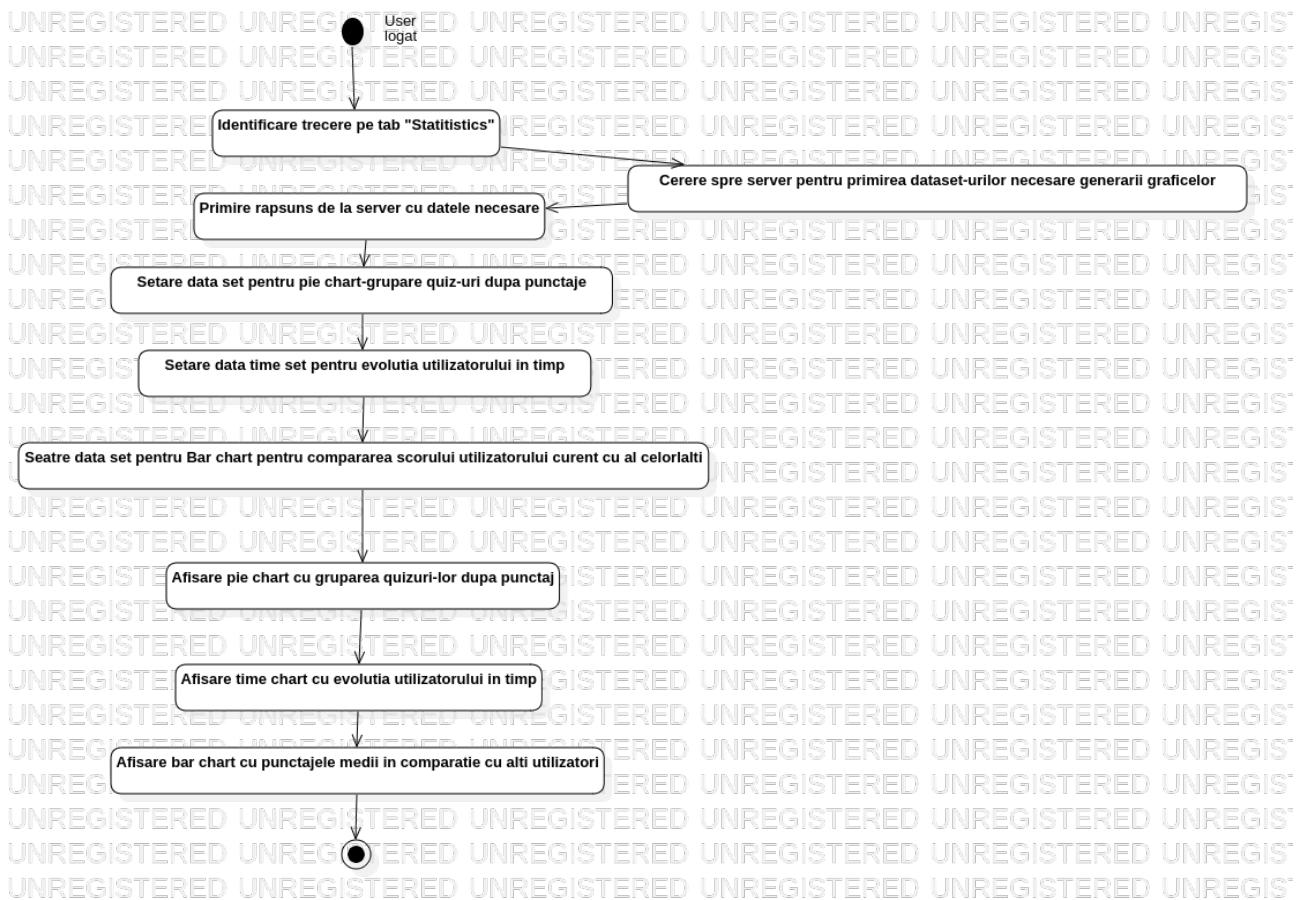
8. Diagrama logare



9. Diagrama quiz

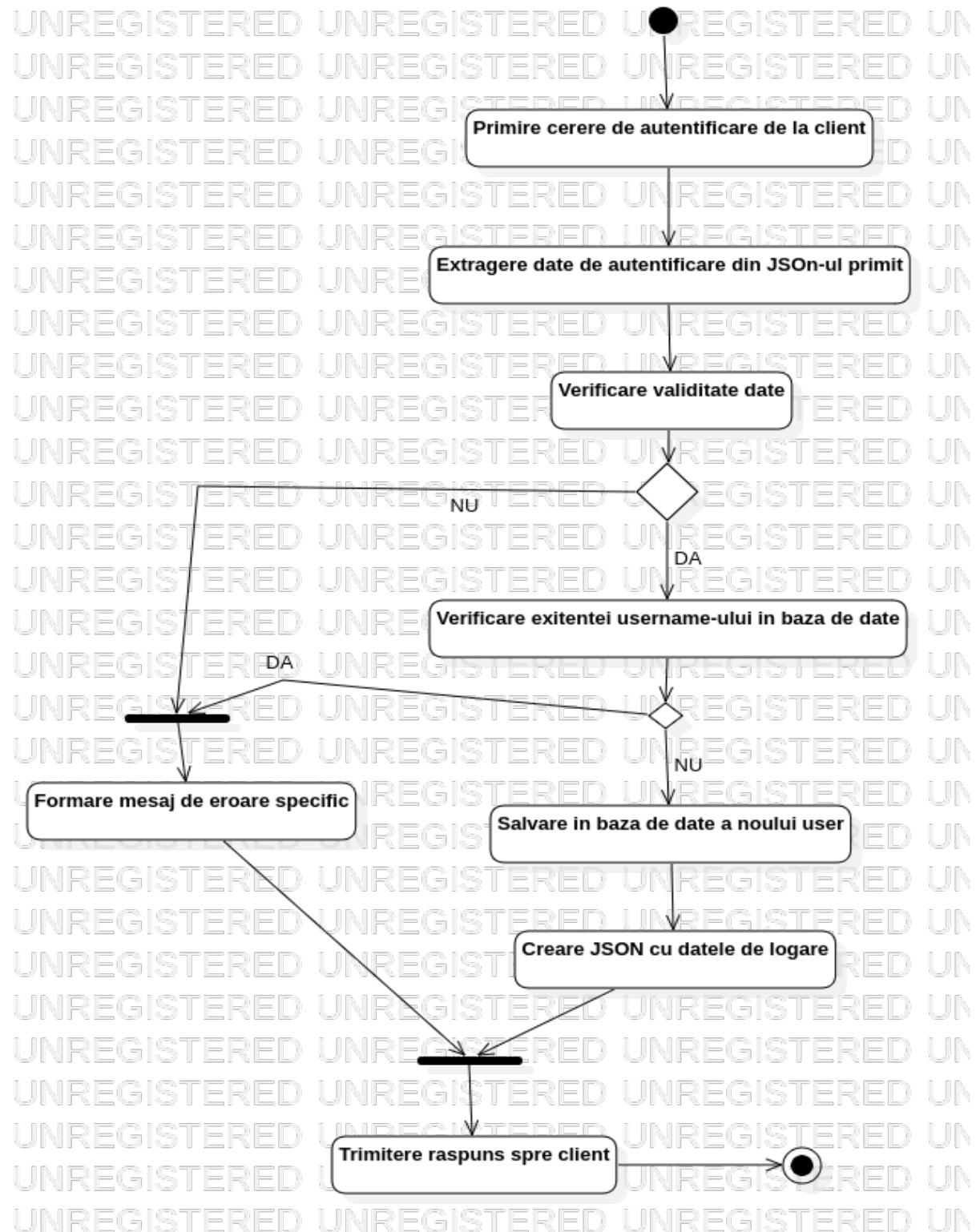


10. Diagrama statistics

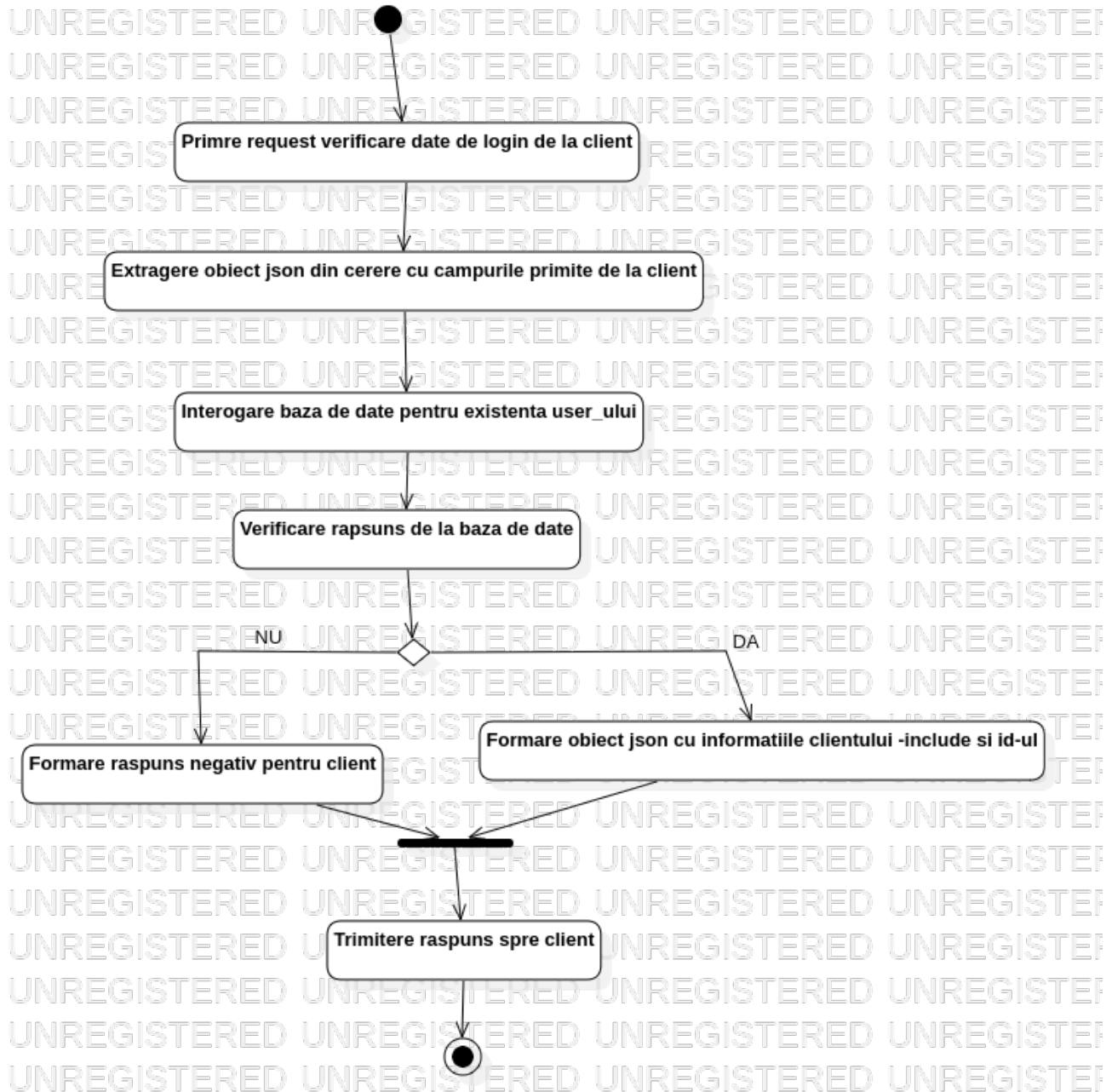


Anexă diagrame de activități server

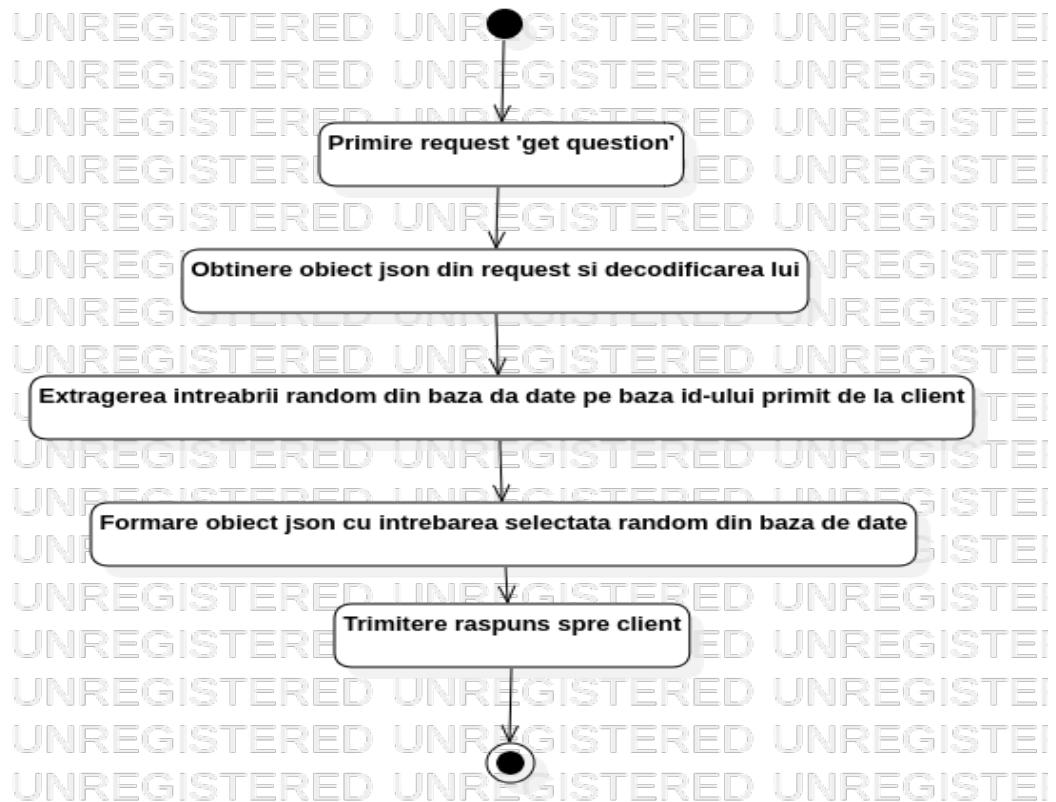
1. Diagrama autentificare



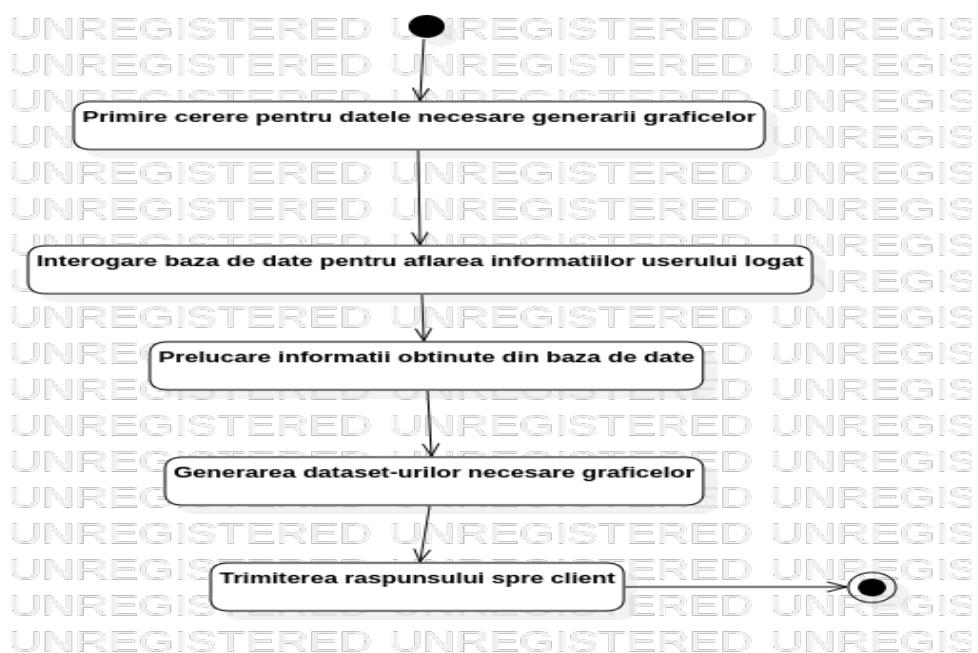
2. Diagrama logare



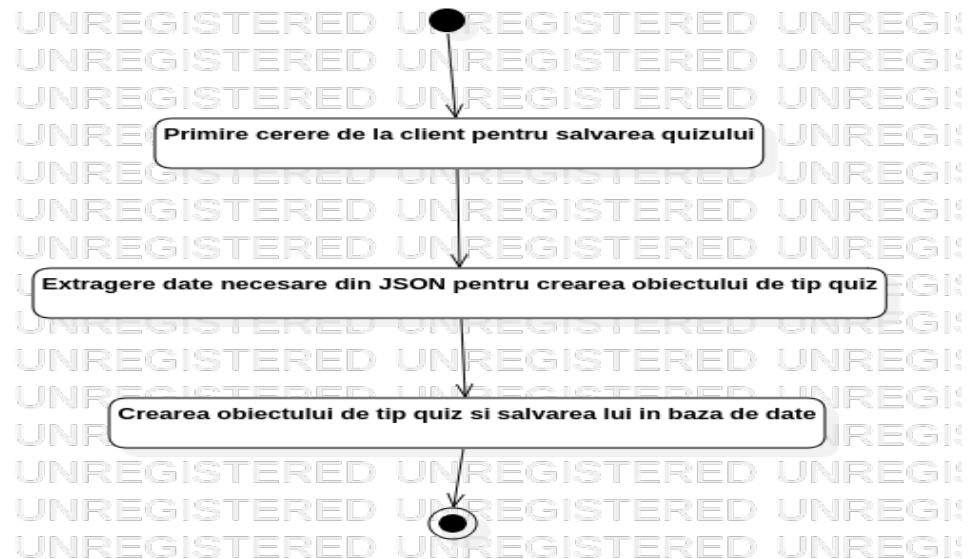
3. Diagrama cerere pentru intrebare



4. Diagrama cerere data set-uri grafice

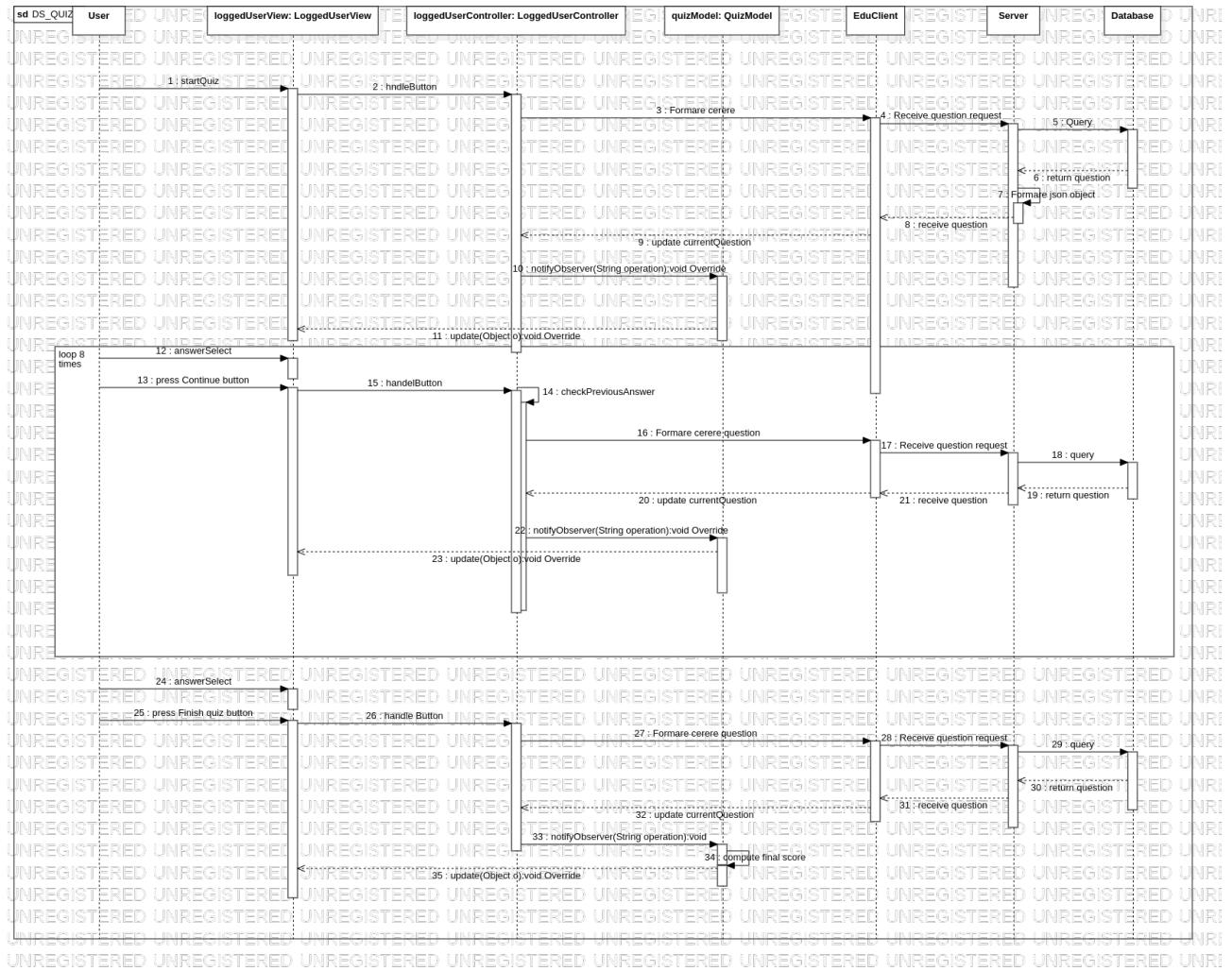


5. Diagrama salvare quiz

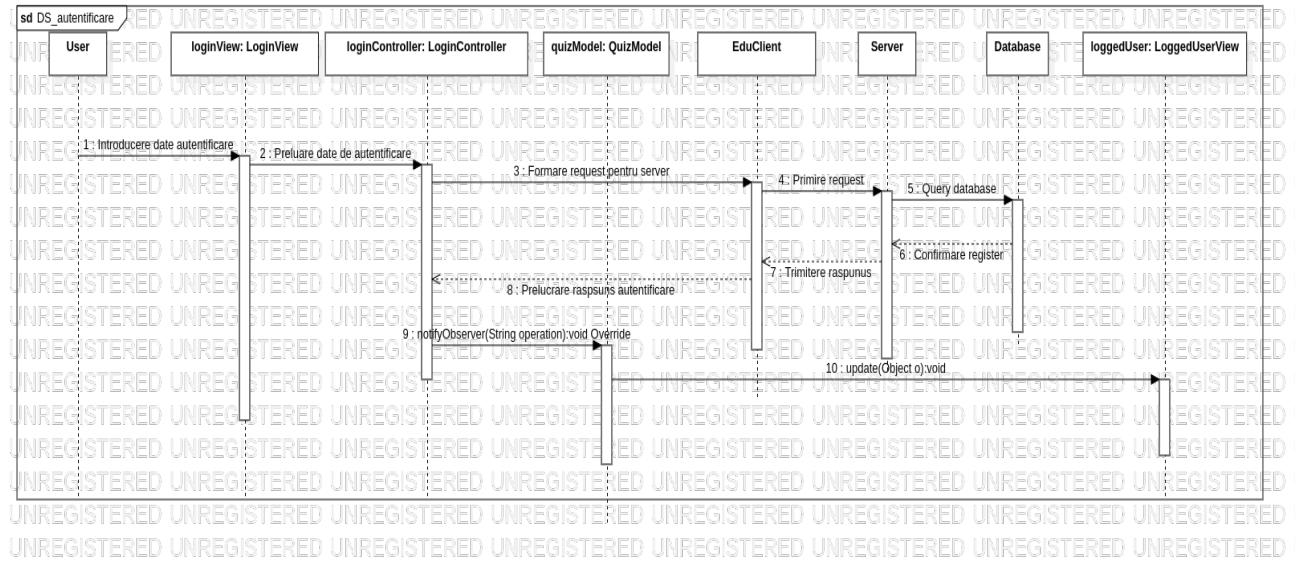


Anexă diagrame de secvență

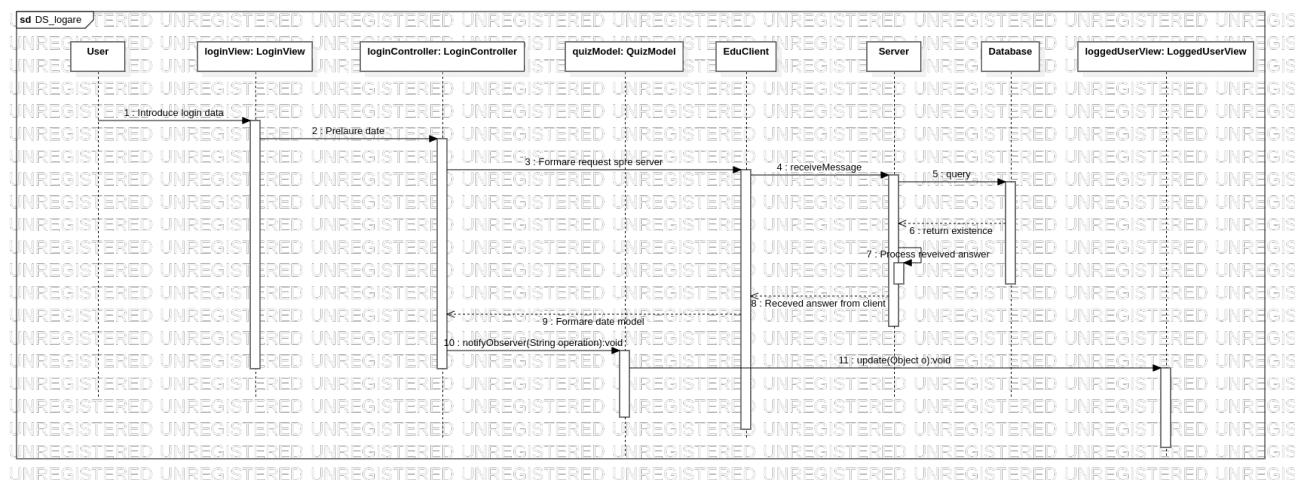
1. Diagrama quiz



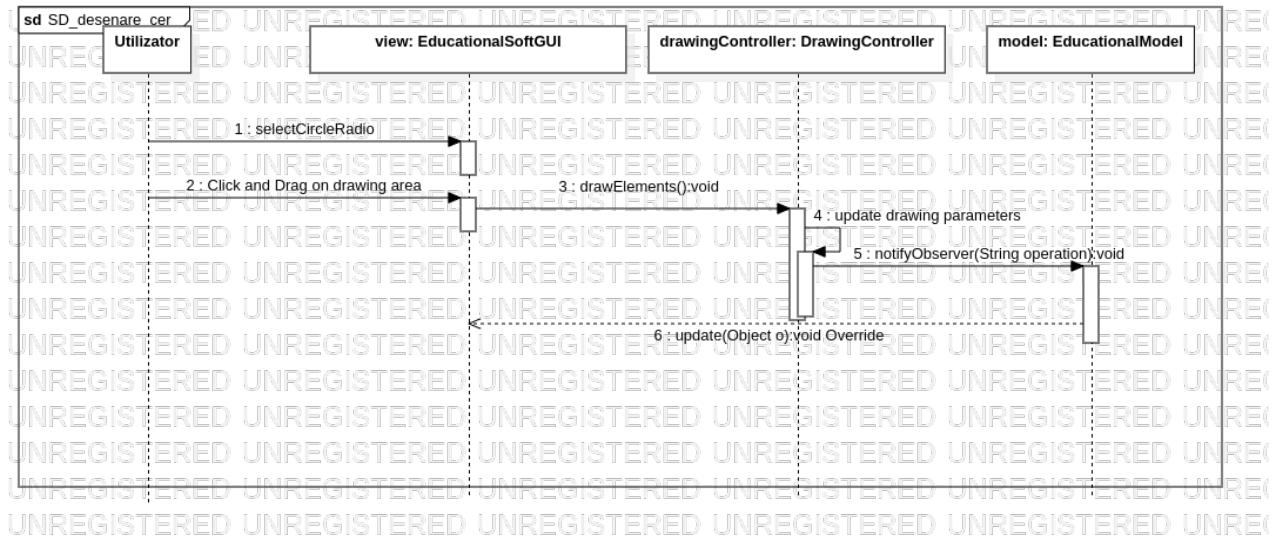
2. Diagrama autentificare



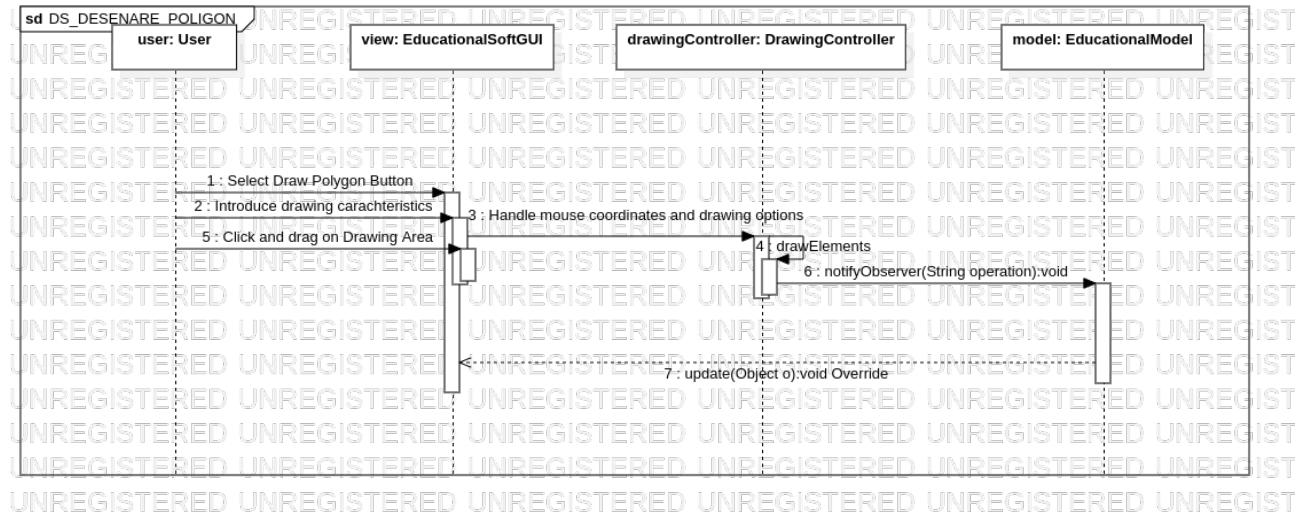
3. Diagrama logare



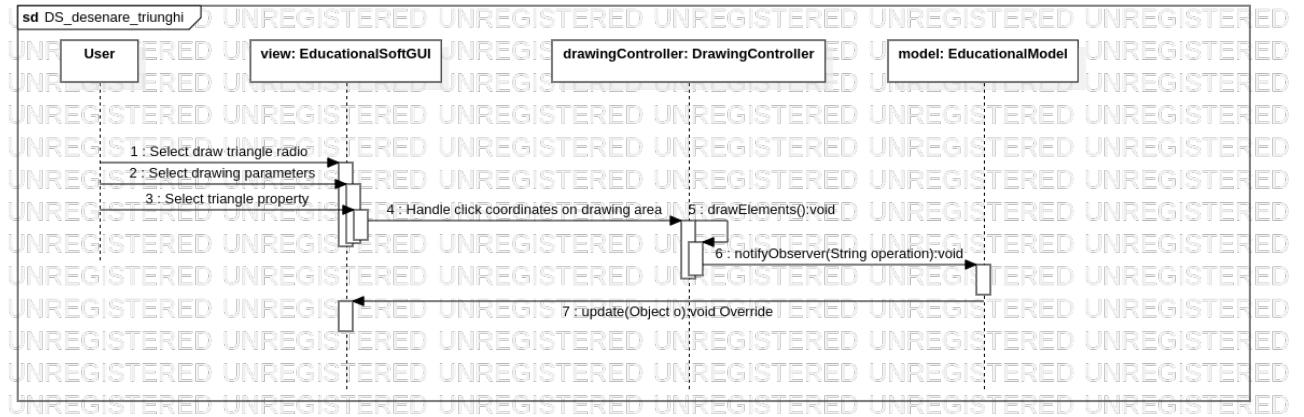
4. Diagrama desenare cerc



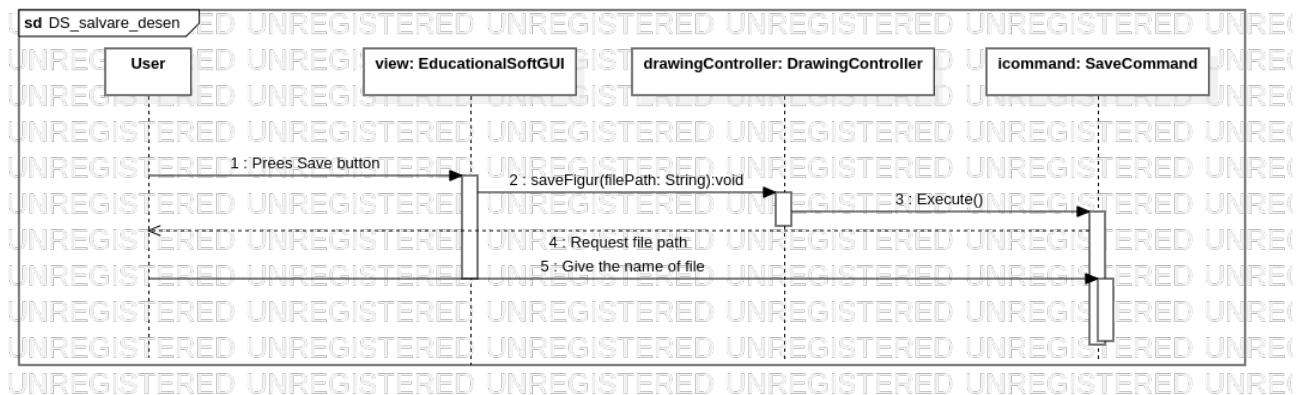
5. Diagrama desenare poligon



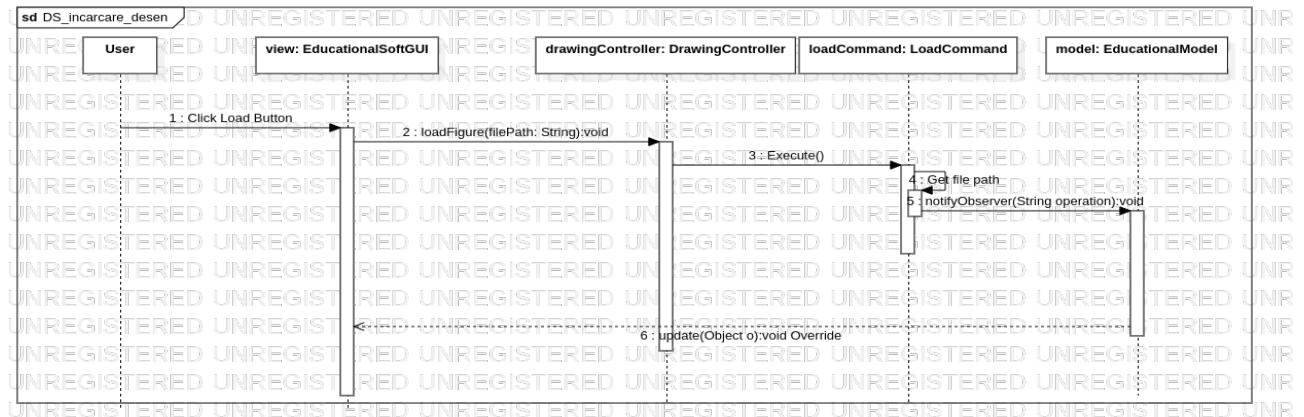
6. Diagrama desenare triunghi



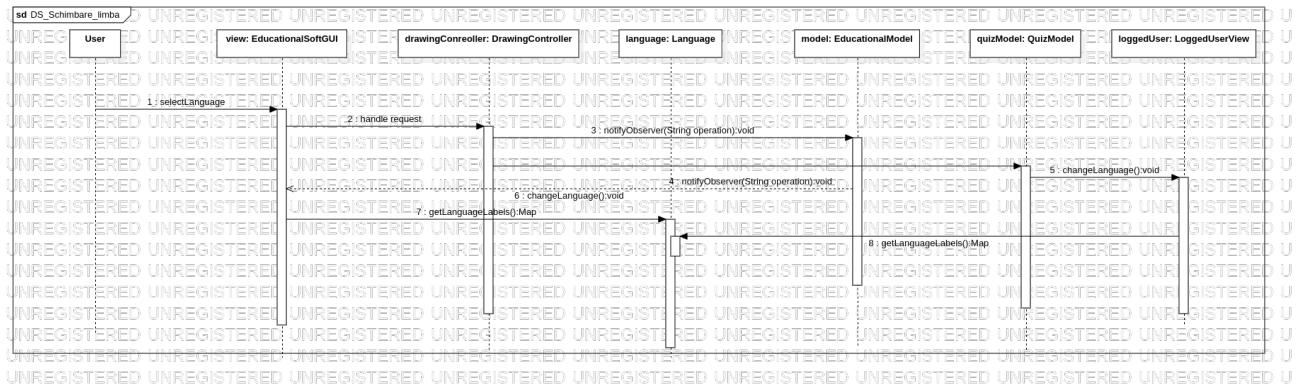
7. Diagrama salare desen



8. Diagrama incarcare desen



9. Diagrama schimbare limba



10. Diagrama view statistics

