UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

Tema1 - Problema26

Proiectare software

Autor : CIOBAN FABIAN-REMUS
Grupa: 30233

Facultatea de Automatica
si Calculatoare

Cuprins

1 Cerință

Problema 26 : Dezvoltați o aplicație care poate fi utilizată ca soft educațional pentru studiul cercului. Aplicația va avea 2 tipuri de utilizatori: elev și administrator. Utilizatorii de tip elev pot efectua următoarele operații fără autentificare:

- desenarea interactivă a cercurilor prin înlocuirea creionului și a riglei cu mouse-ul și alegerea stilului de desenare (culoare, stil linie, grosime linie);

- calcularea și afișarea unor proprietăți: aria unui cerc, lungimea unui cerc, aria unui sector de cerc;

- vizualizarea unor cercuri particulare: o cercul circumscris unui poligon (dacă poligonul este inscriptibil), o cercul înscris (dacă poligonul este circumscriptibil).

- 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 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.

2 Technologies

JAVA SWING - Java Swing reprezintă un kit de instrumente destinat dezvoltării interfețelor grafice (GUI - Graphical User Interface) în limbajul de programare Java, oferind posibilitatea creatorilor de aplicații desktop să construiască interfețe grafice complexe și interactive. Această bibliotecă include diverse componente precum butoane, câmpuri de text, etichete, tabele și meniuri, toate configurabile și aranjabile pentru a oferi o experiență utilizatorului plăcută și intuitivă. Java Swing este parte integrantă a Java Foundation Classes (JFC) și furnizează o interfață de programare platformă-independentă, putând fi utilizată pe multiple sisteme de operare.

GRAPHICS2D - Graphics2D este un instrument sofisticat și versatil în lumea graficii 2D, încorporat în API-ul Java 2D. Acesta oferă dezvoltatorilor posibilitatea de a crea și manipula o varietate de elemente grafice, cum ar fi forme, linii, texte și imagini într-un mediu Java. Cu o gamă vastă de funcționalități, inclusiv antialiasing, transformări, compozitare și redare text, Graphics2D permite dezvoltatorilor să producă grafică și animații de înaltă calitate.Construit pe baza clasei de bază Graphics din Java, care furnizează funcționalități de desenare 2D fundamentale, Graphics2D extinde aceste capabilități prin intermediul unui API mai avansat și flexibil. Acesta oferă metode suplimentare pentru crearea de forme complexe, gestionarea transparenței și compozitării, precum și aplicarea transformărilor afine.Datorită acestor caracteristici, Graphics2D poate fi utilizat pentru dezvoltarea unei game variate de aplicații grafice, incluzând jocuri, simulări științifice, vizualizare de date și interfețe de utilizator. Este un instrument esențial pentru cei care doresc să creeze experiențe vizuale captivante și interactivitate în aplicațiile lor Java.

POSTGRESQL - PostgreSQL reprezintă un sistem robust și open-source de gestionare a bazelor de date relaționale (RDBMS), conceput pentru a gestiona volumul mare de date și inte-

rogări complexe. Cu o serie de caracteristici extinse, precum suportul pentru Structured Query Language (SQL), conformitatea cu ACID (Atomicitate, Consistență, Izolare, Durabilitate) și gestionarea concurenței, PostgreSQL este utilizat într-o varietate de aplicații, inclusiv cele web, de analiză de business, de stocare a datelor și de date geospațiale. Remarcabil pentru stabilitatea, fiabilitatea și capacitatea sa de scalabilitate, PostgreSQL este alegerea preferată în mediile de afaceri, atât în întreprinderi mici, cât și mari.

3 Use case

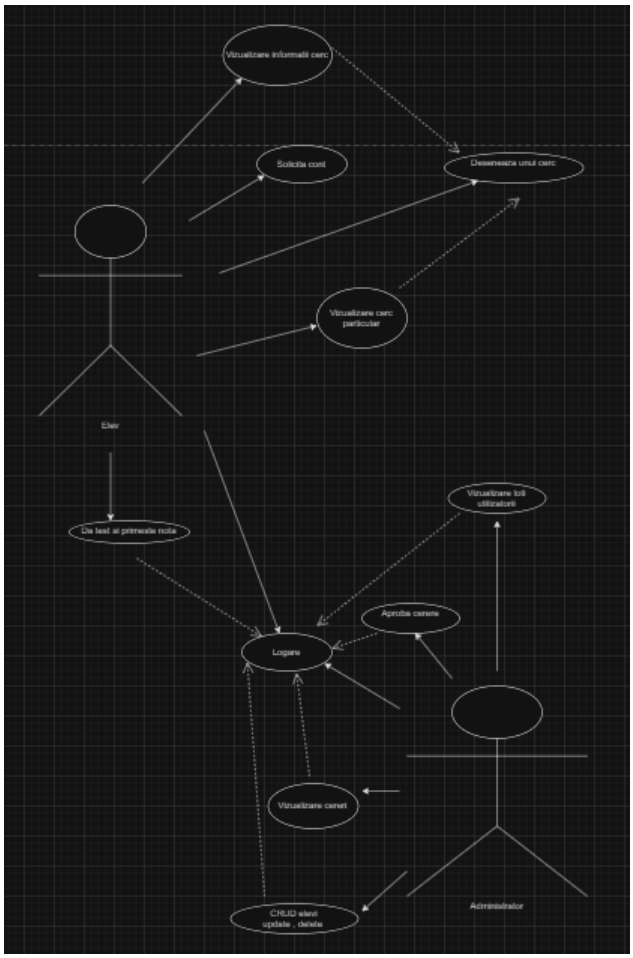

Figura 1:

4 Descriere diagrame

4.1 Model - Geometrie

Clasa ObiectGeometric este o interfață fără nicio definiție de metodă sau câmp. Ea funcționează ca o interfață comună pentru toate obiectele geometrice din acest cod sursă, astfel încât orice clasă care implementează ObiectGeometric este considerată un obiect geometric.

Clasa Cerc reprezintă un cerc în spațiul bidimensional. Ea implementează interfața ObiectGeometric și conține două câmpuri: originea și raza. Metoda computeArea() calculează aria

cercului folosind raza și o returnează, în timp ce metoda computePerimeter() calculează perimetrul cercului și îl returnează.

Clasa Linie reprezintă un segment de dreaptă în spațiul bidimensional. Aceasta implementează interfața ObiectGeometric și are două câmpuri: start și end, ambele fiind instanțe ale clasei Punct. Metoda computeSlope() calculează panta segmentului de linie, în timp ce metoda computeLength() calculează lungimea segmentului de linie. De asemenea, metoda computeAngle() primește un alt obiect Line ca argument și calculează unghiul dintre cele două linii. Metoda computeMiddlePoint() calculează punctul median al segmentului de linie. În cele din urmă, metoda toString() returnează o reprezentare sub formă de șir a segmentului de linie în formatul "startPoint - endPoint".

Clasa Punct reprezintă un punct în spațiul bidimensional. Ea implementează interfața ObiectGeometric și conține două câmpuri: x și y. Metoda computeDistance() primește un alt obiect Punct ca argument și calculează distanța între cele două puncte. Metoda toString() returnează o reprezentare sub formă de șir a punctului în formatul "(x, y)".

Clasa Poligon reprezintă un poligon în spațiul bidimensional. Aceasta implementează interfața ObiectGeometric și are un singur câmp: vertices, care este un ArrayList de obiecte Point. Metoda isInscribed() returnează un boolean care indică dacă poligonul este inscris; adică, este adevărat dacă poligonul are exact trei vârfuri. În caz contrar, calculează lungimea și unghiul fiecărui segment de linie din poligon și returnează false dacă două segmente de linie au lungimi sau unghiuri diferite. Metoda isCircumscribed() returnează un boolean care indică dacă poligonul este circumscris; adică, este adevărat dacă toate vârfurile poligonului se află pe un cerc. În cele din urmă, metoda getCircumcenter() calculează centrul cercului circumscris poligonului găsind punctul de intersecție al bisectoarelor perpendiculare ale oricăror două laturi ale poligonului.

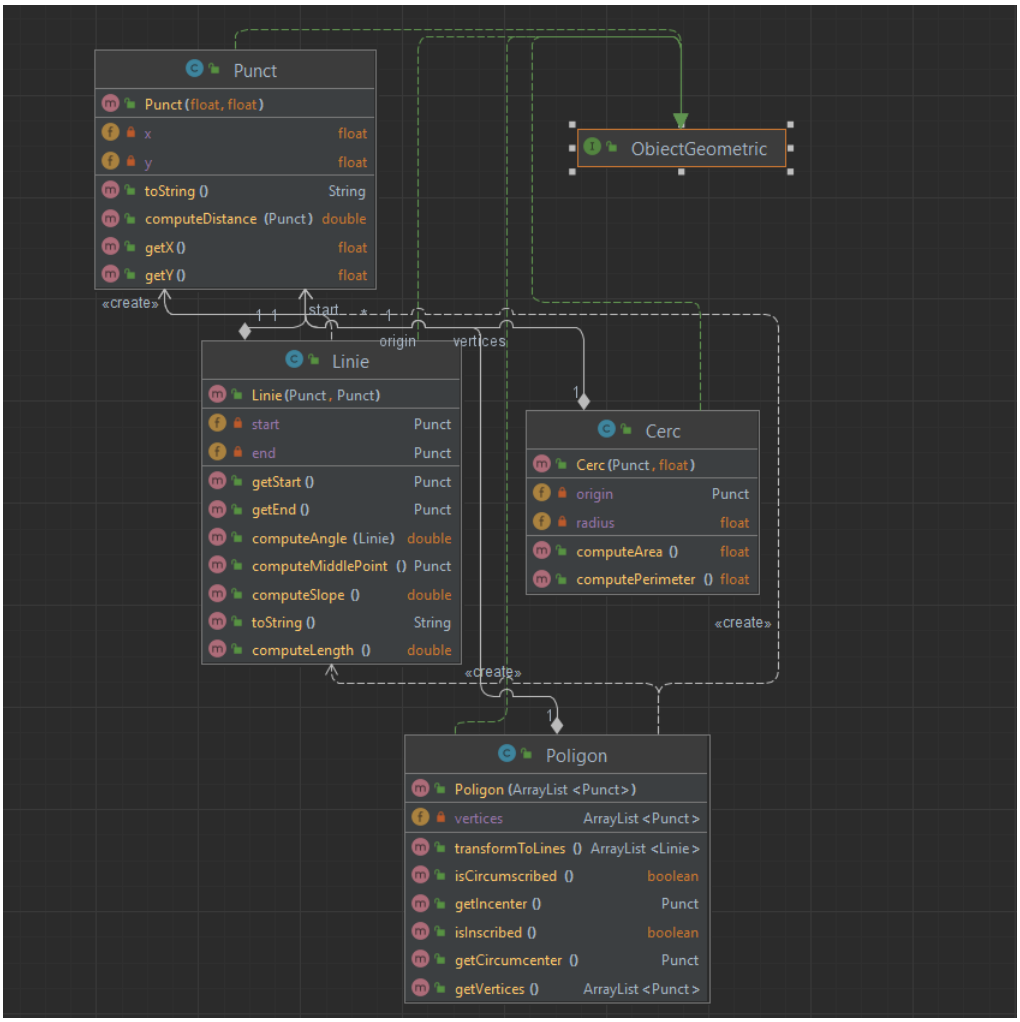

Figura 2:

4.2 Model - User

Clasa Elev reprezintă un utilizator de tip elev în cadrul sistemului. Aceasta conține cinci variabile de instanță private: nume, prenume, poreclă, parolă și statusCont. Primele patru variabile păstrează informațiile personale ale elevului, în timp ce variabila statusCont indică starea contului elevului, fiind o instanță a enumerației StatusCont. Clasa Elev dispune de două constructori: unul care primește cinci parametri pentru a inițializa toate variabilele de instanță și un altul care primește patru parametri și setează implicit statusul contului la REQUESTED. De asemenea, clasa oferă metode getter și setter pentru variabila nume. Enumerația StatusCont este utilizată pentru a specifica stările posibile ale contului unui utilizator. În acest context, stările posibile sunt REQUESTED, APPROVED și ADMIN. Starea REQUESTED este implicită atunci când este creat un nou obiect Elev și indică faptul că utilizatorul a solicitat să se alăture platformei, dar contul său nu a fost încă aprobat. Starea APPROVED semnifică faptul că contul utilizatorului a fost aprobat și acum poate să se autentifice și să utilizeze platforma. Starea ADMIN este atribuită contului unui administrator care deține privilegii speciale pe platformă.

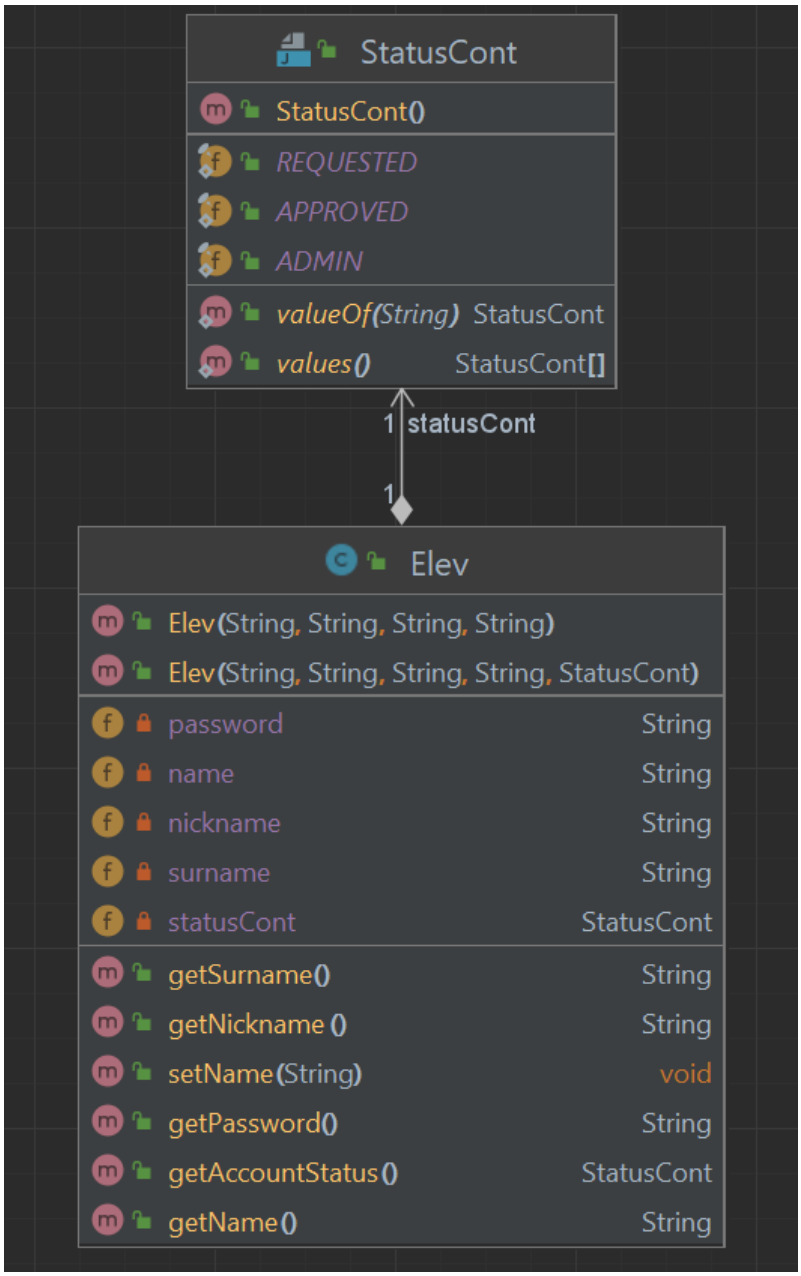

Figura 3:

4.3 Model - Test

Enumerația Dificultate este o enumerare care reprezintă nivelul de dificultate al unei întrebări la un test. Are trei valori posibile: UȘOR, MEDIU și GREU, fiecare cu o valoare întreagă asociată care reprezintă nivelul de dificultate. Clasa Intrebare este o clasă care reprezintă o întrebare la un test. Are patru variabile de instanță: un șir de caractere care reprezintă textul întrebării, o dificultate care reprezintă nivelul de dificultate al întrebării, un ArrayList de șiruri de caractere care reprezintă opțiunile de răspuns și un șir de caractere care reprezintă numele fișierului unei imagini opționale asociate întrebării. Are un constructor care primește aceste patru parametri și le setează ca variabile de instanță. De asemenea, are metode pentru a obține fiecare variabilă de instanță, precum și o metodă pentru a schimba aleator ordinea opțiunilor de răspuns și pentru a returna indexul răspunsului corect. Clasa Test este o clasă care reprezintă un test de quiz. Are

trei variabile de instanță: un ArrayList de întrebări reprezentând întrebările testului, un număr întreg reprezentând numărul maxim de puncte posibile și un ArrayList de numere întregi reprezentând indicii răspunsului corect pentru fiecare întrebare. Are un constructor care generează un set aleatoriu de 10 întrebări dintr-o bază de date de întrebări, calculează numărul maxim de puncte posibile pe baza nivelului de dificultate al fiecărei întrebări și amestecă opțiunile de răspuns pentru fiecare întrebare. De asemenea, are metode pentru a obține fiecare variabilă de instanță. Clasa Punctaj este o clasă care reprezintă o intrare într-o tabelă de rezultate a testului de quiz. Are cinci variabile de instanță: un număr întreg care reprezintă indexul intrării, două șiruri de caractere care reprezintă numele și prenumele persoanei care a susținut testul, un număr întreg care reprezintă numărul de puncte obținute la test și un Timestamp care reprezintă momentul când a fost susținut testul. Are un constructor care primește acești cinci parametri și îi setează ca variabile de instanță. De asemenea, are metode pentru a obține fiecare variabilă de instanță.

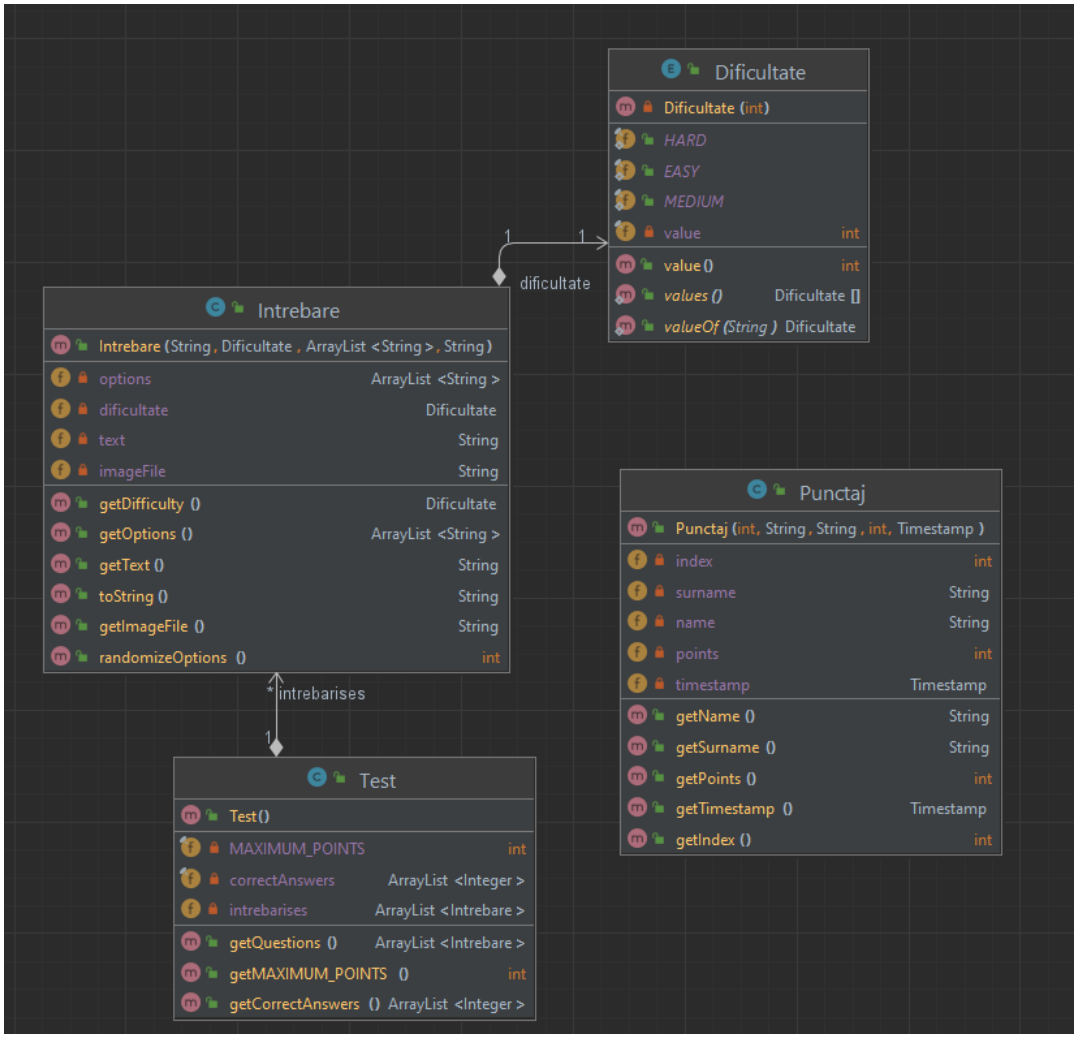

Figura 4:

4.4 Diagrama UML

4.4.1 Diagrama de clase

Figura 5:

5 Detalii implementare

5.1 Resurse

Desenare extinde clasa Canvas din pachetul AWT al limbajului Java. Reprezintă o pânză pentru desen și permite utilizatorului să deseneze puncte și cercuri. Clasa are o instanță DesenareP, care este folosită pentru a interacționa cu stratul de prezentare al programului. Clasa

Desenare gestionează, de asemenea, interacțiunile utilizatorului, cum ar fi clicurile mouse-ului și apăsările tastei, pentru a crea și desena puncte și cercuri pe pânză. Clasa are mai multe câmpuri, cum ar fi punct, cerc, containsPolygon, containsCircle, x, y, startX, startY, endX și endY, care sunt folosite pentru a ține evidența stării pânzei și a formelor desenate. Câmpurile color și stroke reprezintă culoarea și grosimea curentă folosite pentru desenare. Clasa are mai multe metode publice, cum ar fi displayError, setContainsPolygon, setContainsCircle, repaint, getColor, setColor, getStroke și setStroke, care permit utilizatorului să interacționeze cu pânza și să îi modifice proprietățile. Metoda getDrawingCanvasPresenter returnează instanța DesenareP asociată cu pânza.

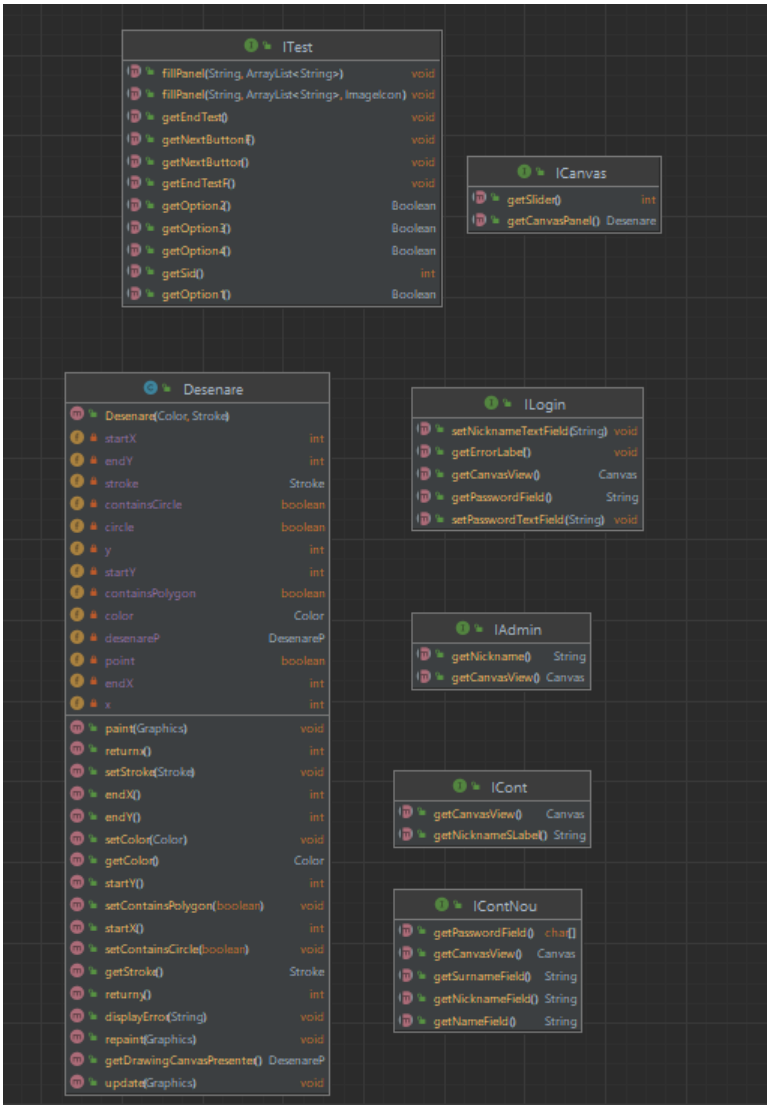

Figura 6:

5.2 Baza de date

Clasa numită creareBD este situată în pachetul 'conectBDpersis'. Ea are mai multe metode pentru crearea și completarea tabelelor într-o bază de date folosind interogări SQL, precum și verificarea existenței tabelelor.

Clasa conține patru metode publice: "createTableStudent()", "fillTableQuestions(String filename)", "createTestsTable()" și "insertAdmin()".

Metoda "createTableStudent()" creează o tabelă numită "STUDENTS" dacă nu există deja. Tabela are coloane pentru numele studentului, prenumele, porecla, parola și starea contului. Ea returnează o valoare booleană indicând dacă tabela a fost creată cu succes și un utilizator administrator a fost inserat.

Metoda "fillTableQuestions(String filename)" citește datele întrebării dintr-un fișier și populează o tabelă numită "QUESTIONS". Tabela are coloane pentru întrebare, dificultatea ei și patru posibile răspunsuri, precum și un fișier de imagine opțional. Ea returnează o valoare booleană indicând dacă tabela a fost creată cu succes și populată cu date.

Metoda "createTestsTable()" creează o tabelă numită "TESTS" dacă nu există deja. Tabela are coloane pentru ID-ul studentului, scorul testului și timpul la care a fost luat testul. Ea returnează o valoare booleană indicând dacă tabela a fost creată cu succes.

Metoda "insertAdmin()" inserează un utilizator administrator implicit în tabela "STUDENTS", cu un nume de utilizator, parolă și starea contului "ADMIN". Ea returnează o valoare booleană indicând dacă inserția a fost reușită.

Clasa folosește o altă clasă numită Persistence pentru gestionarea conexiunii la baza de date și a interogărilor SQL. De asemenea, utilizează o clasă numită Intrebari pentru inserarea datelor întrebării în tabela "QUESTIONS".

Clasa enum numită conectBD conține detaliile necesare pentru stabilirea unei conexiuni la baza de date. Ea are patru constante numite DRIVER, URL, USER și PASSWORD, fiecare reprezentând numele clasei driverului, URL-ul bazei de date, numele de utilizator și parola necesară pentru accesarea bazei de date, respectiv. Constantele sunt atribuite valorile lor respective în constructorii lor, care iau un singur parametru String reprezentând valoarea care trebuie atribuită. Clasa are, de asemenea, o metodă getValue() care returnează valoarea constantei. Această clasă este utilizată în mod obișnuit împreună cu un driver JDBC pentru a stabili o conexiune la o bază de date PostgreSQL.

Clasa Persistence este folosită pentru a se conecta la o bază de date PostgreSQL și a executa interogări SQL. Clasa are un constructor care încearcă să încarce driverul JDBC PostgreSQL folosind valoarea constantei DRIVER definită în enumul conectBD.

Clasa are o metodă numită "connect()" care creează o conexiune la baza de date PostgreSQL folosind valorile constantelor URL, USER și PASSWORD definite în enumul DBConnectionInfo.

Clasa are o metodă numită "close()" care închide conexiunea la baza de date PostgreSQL.

Clasa are o metodă numită "executeQuery(String query)" care primește o interogare SQL ca argument, se conectează la baza de date folosind metoda "connect()", execută interogarea folosind un obiect Statement și returnează o valoare booleană indicând dacă interogarea a fost sau nu a fost reușită.

Clasa are o metodă numită "getDataTable(String query)" care primește o interogare SQL ca argument, se conectează la baza de date folosind metoda "connect()", execută interogarea folosind un obiect Statement și returnează un obiect ResultSet care conține rezultatul interogării.

Atât metodele "executeQuery()" cât și "getDataTable()" utilizează metodele "connect()" și "close()" pentru a se asigura că conexiunea la baza de date este corect stabilită și închisă după fiecare interogare.

În ansamblu, acest cod oferă o modalitate simplă și reutilizabilă de a se conecta la o bază de date PostgreSQL și de a executa interogări SQL.

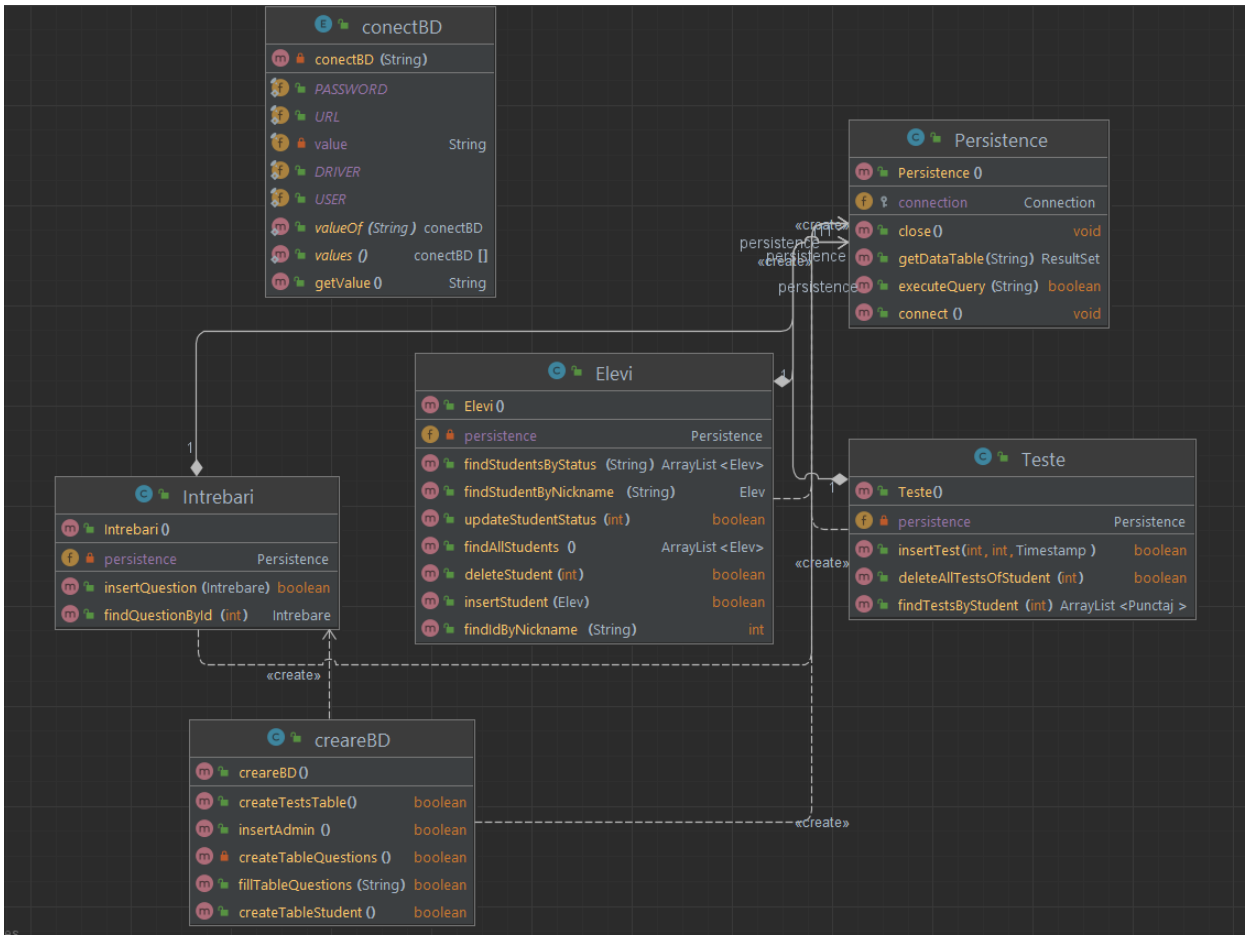

Figura 7:

Diagrama relatie-entitate

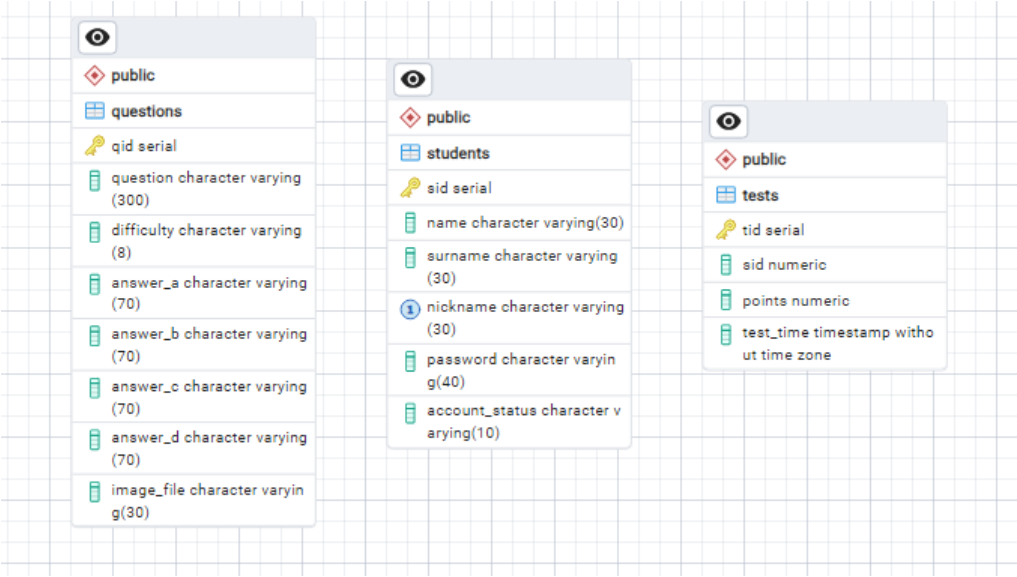

Figura 8:

5.3 Presenter

Clasa ContP face parte din stratul de prezentare și servește ca intermediar între Cont și stratul de acces la date. Clasa are o referință către o instanță a Cont și furnizează metode pentru a recupera teste de elevi, a lua un test și a reveni la ecranul anterior. Metoda retrieveStudentTests recuperează testele elevului curent și le returnează ca un obiect DefaultTableModel, care poate fi folosit pentru a popula un JTable. Metoda folosește o instanță de StudentPersistence pentru a găsi ID-ul elevului pe baza pseudonimului său, și o instanță de TestPersistence pentru a recupera lista de obiecte TestTableEntry asociate cu acel ID de elev. Metoda creează apoi un nou obiect DefaultTableModel și adaugă coloane pentru index, nume, prenume, puncte și timestamp. Apoi, parcurge obiectele TestTableEntry și adaugă un rând nou în DefaultTableModel pentru fiecare intrare, folosind metoda addRow.

Metoda takeTest este apelată când utilizatorul apasă butonul "Dati testul" în Cont. Metoda folosește o instanță de StudentPersistence pentru a găsi ID-ul elevului pe baza pseudonimului său, creează un nou obiect Test cu acel ID ca parametru și setează vizibilitatea Account curent la false. Test este o fereastră nouă unde utilizatorul poate lua un test.

Metoda goBack este apelată când utilizatorul apasă butonul "Înapoi" în Account. Metoda setează vizibilitatea Canvas (ecranul anterior) la true și elimină Account curent.

Codul furnizat arată implementarea clasei AdminP, care servește ca intermediar între Admin și depozitele ElevP și TestP. Clasa furnizează metode pentru aprobarea, ștergerea și vizualizarea tuturor elevilor și a cererilor lor. Metodele folosesc clasa DefaultTableModel pentru a prezenta datele într-un format tabular și afișează mesaje relevante folosind JOptionPane. În plus, clasa furnizează o metodă back() pentru a reveni la vizualizarea anterioară. În ansamblu, clasa este proiectată pentru a facilita gestionarea conturilor elevilor de către administrator.

GeometrieP este responsabil pentru manipularea interacțiunilor utilizatorului cu Canvas, care este un element de interfață grafică care afișează forme geometrice desenate pe un panou.

Clasa GeometrieP conține mai multe metode pentru modificarea aspectului formelor de pe panou, cum ar fi schimbarea culorii lor, lățimea conturului și modelul de contur. De asemenea, furnizează metode pentru afișarea informațiilor despre un cerc desenat pe panou și pentru desenarea unui cerc circumscris sau inscris într-un poligon. În ansamblu, această clasă acționează ca intermediar între interfața utilizatorului și modelul de date subiacent, permițând utilizatorului să interacționeze cu formele de pe panou într-un mod semnificativ.

Prezentatorul CanvasUserP este o clasă prezentator responsabilă de gestionarea interacțiunilor utilizatorului legate de conectare și crearea unui cont nou. Are acces la un obiect Canvas, pe care îl folosește pentru a ascunde vizualizarea curentă și a afișa vizualizarea corespunzătoare pentru conectare sau cont nou. Metoda openLogin() este apelată când utilizatorul dă clic pe butonul "Login" în vizualizarea canvasului. Ascunde vizualizarea canvasului și creează un nou obiect Login, trecând obiectul Canvas ca parametru. Apoi se afișează Login, permițând utilizatorului să-și introducă datele de conectare.

Similar, metoda openNewAccount() este apelată când utilizatorul dă clic pe butonul "Solicita cont" în vizualizarea canvasului. Ascunde vizualizarea canvasului și creează un nou obiect ContNou, trecând obiectul Canvas ca parametru. Apoi se afișează ContNou, permițând utilizatorului să creeze un cont nou.

Clasa conține metode pentru adăugarea acestor forme pe canvas, crearea unui poligon folosind un set de puncte, curățarea canvasului și recuperarea obiectului cercului sau poligonului

aflat în prezent pe canvas. Clasa gestionează, de asemenea, randarea grafică obținând obiectul grafic din canvas și folosindu-l pentru a desena formele.

Codul urmează modelul Model-View-Presenter (MVP), unde prezentatorul acționează ca un intermediar între model (forme geometrice) și vizualizare (canvas de desenare). Prezentatorul primește intrări de la utilizator din vizualizare, manipulează obiectele modelului în consecință și actualizează vizualizarea cu datele noi. Această separare a preocupărilor face ca codul să fie mai modular și mai ușor de întreținut.

Clasa LoginP gestionează procesul de conectare a utilizatorului. Primește intrări din Login și interacționează cu Elevi pentru a prelua un obiect Student din baza de date. Dacă Elevul este găsit, verifică StatusCont elevului. Dacă StatusCont este ADMIN, creează un nou Admin și ascunde Login. Dacă StatusCont este APPROVED, creează un nou Account cu informațiile studentului și ascunde Login. Dacă elevul nu este găsit în baza de date, afișează un mesaj de eroare pe Login. Metoda reset() este apelată pentru a șterge câmpurile text din Login.

LoginP este responsabil pentru controlul fluxului procesului de conectare și gestionarea oricăror erori care pot apărea. Interacționează cu Elevi pentru a prelua date din baza de date și cu Login pentru a actualiza interfața utilizatorului. LoginP se asigură că vizualizarea corespunzătoare este afișată în funcție de StatusCont elevului. Metoda reset() este folosită pentru a asigura că Login este curățat după ce un utilizator se conectează sau întâmpină o eroare. LoginP este un component important al procesului de conectare și ajută la asigurarea unei experiențe de utilizator fără probleme și fără erori.

Clasa ContNouP gestionează acțiunile legate de crearea unui cont nou în sistem. Interacționează cu clasa ContNou pentru a prelua intrările utilizatorului și pentru a afișa informații utilizatorului.

Clasa conține un constructor care primește o instanță a clasei ContNou și o setează pe un câmp privat. În plus, setează vizibilitatea vizualizării canvasului la fals. Metoda request() gestionează crearea unui cont nou. Preia intrările utilizatorului din vizualizare și le validează. Dacă porecla introdusă de utilizator nu este deja în uz, metoda creează un obiect Student nou și îl inserează în baza de date folosind clasa StudentPersistence. Dacă inserția reușește, metoda afișează un mesaj utilizatorului informând că cererea a fost trimisă și afișează vizualizarea canvasului. În caz contrar, afișează un mesaj de eroare. Metoda reset() curăță câmpurile de intrare din vizualizare.

În ansamblu, clasa ContNouP oferă o modalitate pentru utilizatori de a crea conturi noi în sistem și gestionează comunicarea între vizualizare și stratul de persistență.

Clasa TestP este responsabilă pentru controlul comportamentului Test, care este interfața grafică în care utilizatorul poate lua un test. TestP gestionează crearea unui obiect Test nou, care conține o listă de obiecte Question și răspunsurile corecte corespunzătoare. TestP ține, de asemenea, evidența progresului utilizatorului prin test, calculează scorul lor și salvează scorul în baza de date când testul este completat.

Metoda start() inițializează obiectul Test și setează indexul și punctele la zero. Metoda next() este responsabilă pentru trecerea la întrebarea următoare în test. Verifică mai întâi răspunsul utilizatorului pentru întrebarea curentă și își actualizează scorul în consecință. Apoi afișează întrebarea următoare în Test, completând textul întrebării, opțiunile și orice imagine asociată.

Metoda end() este apelată când utilizatorul a completat testul și calculează scorul final și-l salvează în baza de date.

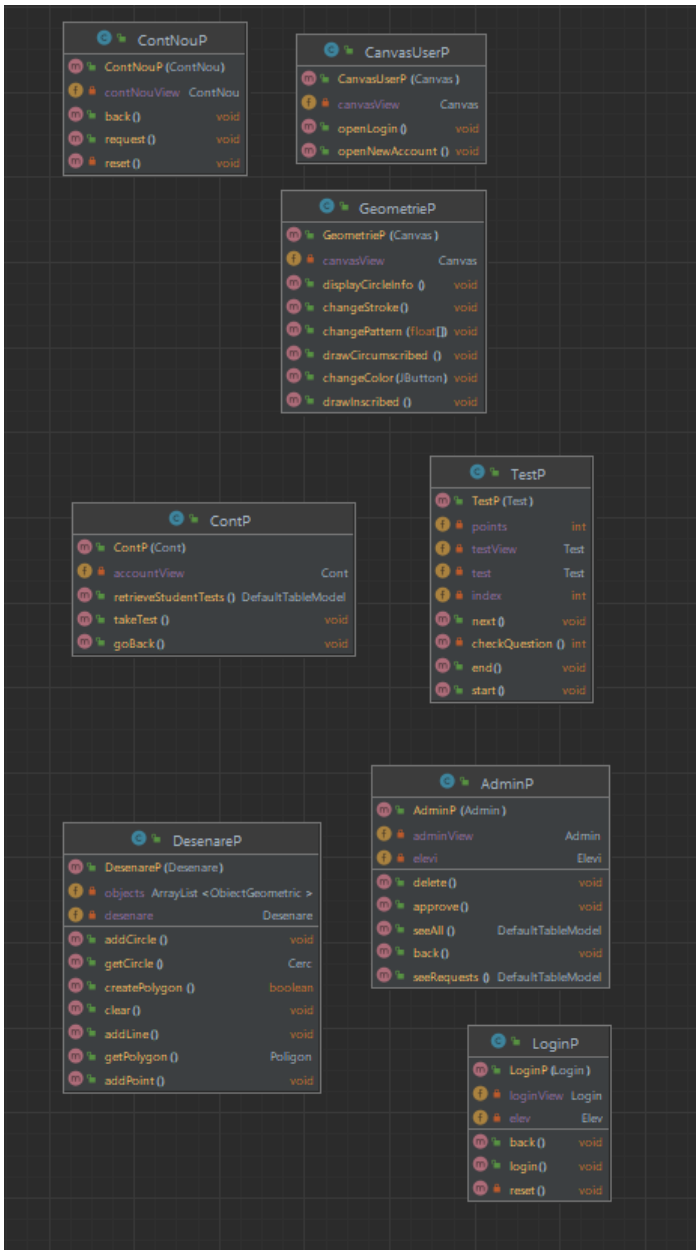
ContNouP
ContNouP (ContNou)
contNouView ContNou
back () void
request () void
reset () void
CanvasUserP
CanvasUserP (Canvas)
canvasView Canvas
openLogin () void
openNewAccount () void
GeometrieP
GeometrieP (Canvas)
canvasView Canvas
displayCircleInfo () void
changeStroke () void
changePattern (float[]) void
drawCircumscribed () void
changeColor (JButton) void
drawInscribed () void
TestP
TestP (Test)
points int
testView Test
test Test
index int
next () void
checkQuestion () int
end () void
start () void
ContP
ContP (Cont)
accountView Cont
retrieveStudentTests () DefaultTableModel
takeTest () void
goBack () void
AdminP
AdminP (Admin)
adminView Admin
elevi Elevi
delete () void
approve () void
seeAll () DefaultTableModel
back () void
seeRequests () DefaultTableModel
DesenareP
DesenareP (Desenare)
objects ArrayList <ObiectGeometric >
desenare Desenare
addCircle () void
getCircle () Cerc
createPolygon () boolean
clear () void
addLine () void
getPolygon () Poligon
addPoint () void
LoginP
LoginP (Login)
loginView Login
elev Elev
back () void
login () void
reset () void

Figura 9: