Logo, company name

Description automatically generated

Aplicatie pentru studiul geometriei patrulaterului

Studenta: Maracine Stefania-Maria

Grupa: 30233

**Cuprins**:

1. Enuntul problemei
2. Instrumente utilizate
3. Justificarea limbajului de programare ales
4. Descrierea aplicatie
5. Descrierea diagramelor UML
   * + 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;

➢ cercuri 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**

Instrumentele pe care le-am utilizat in crearea aplicatiei sunt:

* IntelliJ Idea ca IDE
* MySQL ca si baza de date

**IntelliJ IDEA**

IntelliJ IDEA este un mediu de dezvoltare integrat (IDE) utilizat în special pentru dezvoltarea de aplicații Java, dar suportă și alte limbaje de programare precum Kotlin, Scala, Groovy, Python și altele. Acest IDE este cunoscut pentru performanța și ușurința de utilizare, având o gamă largă de funcții precum completare automată a codului, refactorizare, testare și analiză statică a codului. IntelliJ IDEA vine cu o mulțime de plugin-uri și suportă dezvoltarea într-o varietate de framework-uri și tehnologii, cum ar fi Spring, JavaFX, Maven, Gradle și multe altele.

**MySQL**

MySQL este un sistem de gestiune a bazelor de date relationale (RDBMS - Relational Database Management System) open-source, care utilizează limbajul SQL (Structured Query Language) pentru a gestiona și manipula datele stocate. Acesta este utilizat frecvent în aplicații web și este compatibil cu majoritatea platformelor și limbajelor de programare. MySQL oferă o gamă largă de funcții, precum gestionarea utilizatorilor și a permisiunilor de acces, criptarea datelor, replicarea datelor pentru a asigura redundanță și disponibilitate și multe altele. Este una dintre cele mai populare baze de date, fiind folosită în mod obișnuit de dezvoltatori și companii pentru a stoca și gestiona datele.

* + - 1. **Justificarea limbajului de programare ales**

Limbajul de programare Java are multe avantaje față de alte limbaje de programare. În primul rând, Java este un limbaj independent de platformă, ceea ce înseamnă că codul Java poate fi rulat pe orice sistem de operare, indiferent de arhitectura acestuia.

Un alt avantaj al limbajului Java este securitatea ridicată. Java utilizează o tehnologie de securitate puternică care oferă o protecție excelentă împotriva atacurilor cibernetice și a vulnerabilităților de securitate. Această tehnologie de securitate face Java un limbaj ideal pentru dezvoltarea de aplicații critice pentru securitate, cum ar fi cele din domeniul bancar sau medical.

Java oferă și o performanță excelentă și o memorie eficientă, datorită tehnologiei de gestionare a memoriei cunoscută sub numele de "garbage collection". Această tehnologie elimină automat obiectele care nu sunt utilizate, ceea ce face ca aplicațiile Java să ruleze mai rapid și mai eficient.

Un alt avantaj al limbajului Java este vasta colecție de librării și framework-uri disponibile, care permit dezvoltatorilor să dezvolte aplicații mai rapid și mai eficient. Cateva tehnologii utilizate in acest proiect sunt Hibernate, testare prin Junit si Framework-ul Java Swing.

Hibernate este o tehnologie open-source pentru mapping-ul obiect-relațional (ORM - Object-Relational Mapping) în Java, care facilitează gestionarea datelor din baza de date printr-un model de obiecte Java. Hibernate oferă o modalitate eficientă și simplă de a mapa obiecte Java pe structurile de date ale bazei de date, eliminând necesitatea de a scrie manual codul SQL pentru interogarea și manipularea datelor.

JUnit este o bibliotecă de teste unitare pentru Java, care facilitează dezvoltarea de teste automate pentru a verifica corectitudinea funcționării codului Java. JUnit oferă o gamă largă de funcții și metode pentru a testa diverse scenarii de utilizare ale aplicațiilor, ajutând la descoperirea rapidă a erorilor și la menținerea calității codului.

Java Swing este o bibliotecă de interfață grafică pentru utilizator (GUI - Graphical User Interface) pentru Java, care permite dezvoltatorilor să creeze aplicații desktop personalizate și estetice. Swing oferă o gamă largă de componente grafice și controale, inclusiv butoane, etichete, meniuri, tabele și multe altele, ceea ce face dezvoltarea de aplicații desktop în Java mai ușoară și mai intuitivă.

* + - 1. **Descrierea aplicatiei**

Arhitectura Model-View-Controller (MVC) este un pattern arhitectural utilizat în dezvoltarea de aplicații software, care împarte responsabilitățile în trei componente distincte: Model, View și Controller. MVC facilitează dezvoltarea de aplicații mai ușor de întreținut, scalabile și testabile. Prin separarea responsabilităților în aceste trei componente, MVC permite separarea logicii aplicației și a datelor de interfața grafică, făcând mai ușoară schimbarea uneia dintre acestea fără a afecta celelalte.

Modelul reprezintă componenta care se ocupă de accesul la date și logica aplicației, fiind responsabilă pentru gestionarea datelor și informațiilor. Aceasta este o componentă ce nu cunoaște nimic despre interfața grafică și care poate fi utilizată de mai multe componente View și Controller. Modelul poate fi reprezentat sub formă de obiecte, structuri de date sau baze de date.

Pachetul **Model** are doua subpachete, **Entities** unde sunt declarate clasele principale, clasele de baza cum ar fii user, punct, segment, patrulater si subpachetul **Persistence** unde avem clasele AbstractPersistence si ConnectionFactory pentru conexiunea cu baza de date si operatiile CRUD pe aceasta si clasa AbstractPersistenceTest unde au fost realizate testele CRUD pentru baza de date.

Pachetul **View** cuprinde clasele care reprezintă componenta care se ocupă de interfața grafică a aplicației și de afișarea datelor, fiind responsabilă pentru prezentarea datelor către utilizator și pentru interacțiunea cu acesta. View-ul nu cunoaște nimic despre logica aplicației și nici despre datele utilizate.

Pachetul **Controller** reprezintă componenta care se ocupă de gestionarea evenimentelor și a interacțiunii utilizatorului, fiind responsabilă pentru gestionarea fluxului de date între Model și View. Controllerul primește date de la utilizator prin intermediul View-ului și actualizează Modelul în consecință. De asemenea, Controllerul poate primi date de la Model și le poate transmite View-ului pentru afișare.

Ca si pattern comportamental am utilizat Observer care notifica automat alte obiecte despre modificările survenite asupra unui obiect. Este util pentru a reduce cuplajul dintre obiecte și pentru a permite unui număr mare de obiecte să urmărească schimbările unui singur obiect, fără să fie nevoie să verifice periodic obiectul respectiv.

In aplicatia mea l-am implementat pentru a schimba limba proiectului din interiorul fiecarei interfete grafice(dupa cum se poate vedea in fiecare figura din proiect) dintr-un combo box in care avem trei variante: engleza, romana si germana.

In continuare o sa descriu fiecare interfata grafica si functionalitatile acesteia.

Prima interfata grafica este interfata de Start, cea din figura 1.1, prima pagina ce porneste la rularea programului si de unde utilizatorii isi pot alege ce doresc sa faca, pot desena un patrulater si afla informatii despre el, se pot loga sau pot solicita un cont.

Fig 1.1

Graphical user interface, application, Word

Description automatically generated

A doua interfata grafica este cea de RequestLogIn, din figura 1.2, unde membrii care nu au cont pot solicita unul. Acestia trebuie sa isi introduca numele, prenumele si o adresa de email care vor fii stocate in baza de date si vizualizate ulterior de un administrator care le va putea crea contul.

Fig 1.2

Graphical user interface, application

Description automatically generated

A treia interfata este cea de LogIn, din figura 1.3, unde utilizatorul trebuie sa isi introduca username-ul si parola pentru a-si accesa contul. In functie de tipul utilizatorului, se va deschide o pagina diferita.

Graphical user interface, application, Word

Description automatically generatedFig 1.3

Pentru un utilizator de tip Elev se va deschide, cea din figura 1.4, o pagina unde acesta va putea da un test de cultura generala din geometria patrulaterelor cu 10 intrebari care sunt alese random dintr-o lista de 50 de intrebare stocate in baza de date, vor putea scrie un raspuns si vor primi un rezultat o nota de la 0 la 10. Butonul Generare intrebare ia intrebarile din baza de date, le da shuffle si ia primele 10, afisandu-le apoi in TextField. Butonul afisare rezultat ia raspunsurile din text field-urile in care a scris elevul, le compara cu cele din baza de date, si afiseaza intr-un alt textfiel rezultatul testului.

Graphical user interface, text, application, email

Description automatically generatedFig 1.4

Pentru un utilizator de tip Administrator se va deschide o pagina, cea din figura 1.5, unde va putea face request prin butonul Show List in baza de date si va putea vizualiza in partea stanga intr-un tabel toate cererile de creare cont, iar in partea dreapta, cu datele furnizate in table va da un username, o parola si un tip si va putea genera contul prin apasarea butonului Create Account care va face un insert de user in baza de date.

Ultima interfata grafica este cea de drawing, din figura 1.6, care poate fii accesata de orice utilizator, nu este necesar un cont. In aceasta interfata este posibila in partea stanga desenarea interactiva a patrulaterelor prin apasarea click-ului in 4 puncte diferite si in dreapta vizualizarea anumitor caracteristici ale patrulaterului precum lungimile laturilor, masurile unghiurilor, daca este concave sau convex etc.

Fig 1.5

Graphical user interface, application

Description automatically generated

Fig 1.6

Graphical user interface

Description automatically generated

* + - 1. **Descrierea diagramelor UML**

A picture containing text, diagram, plan, schematic

Description automatically generatedFig 2

In acest capitol am pus poze cu diagrama UML intreaga(fig 2), dupa care 3 poze din diagrama UML in functie de pachetul din care fac parte clasele(view – fig 2.1; command – fig 2.2; model si persistence – fig 2.3).

Fig 2.1

A picture containing text, font, receipt, number

Description automatically generated

Fig 2.2

A picture containing text, diagram, receipt, line

Description automatically generated

Fig 2.3

A picture containing diagram, text, plan, line

Description automatically generated

A screenshot of a computer program

Description automatically generated with low confidence**Diagrama ERD:**

**Diagrama Use-Case:**

A picture containing drawing, sketch, diagram, text

Description automatically generated

**Diagramele de activitate:**

A picture containing text, number, document, parallel

Description automatically generated