A black background with red lines

Description automatically generated

**FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE**

**SPECIALIZAREA CALCULATOARE**

**PROIECT**

**JOC LABIRINT IEPURAȘ**

Disciplina : Proiectare software

Student : Pojar Andrei-Gabriel

An : III

Grupa : 30233

Profesor coordonator : Anca Iordan

**CUPRINS**

1. Introducere

* Enunțul problemei
* Motivație

1. Instrumente utilizate
2. Justificarea limbajului de programare ales
3. Descrierea diagramelor UML
4. Descrierea aplicației
5. Concluzii
6. **Introducere**

* **Enunțul problemei**

Obiectiv:

Obiectivul acestui proiect este familiarizarea cu șablonul architectural Client/Server, cu șabloanele arhitecturale orientate pe servicii (SOA) și cu șabloanele de proiectare.

Pentru persistența informației se va utiliza o bază de date relațională (SQL Server, MySQL, etc.).

Cerințe: Transformați aplicația implementată la tema 3 într-o aplicație client/server astfel încât să utilizați minim 5 șabloane de proiectare (minim un șablon de proiectare creațional, un șablon de proiectare comportamental și un șablon de proiectare structural) și o arhitectură orientată pe servicii SOA pentru comunicare între aplicația server și aplicația client.

❖ În faza de analiză se va realiza diagrama cazurilor de utilizare și diagramele de activități pentru fiecare caz de utilizare (Observație: numărul diagramelor de activități trebuie să fie egal cu numărul de cazuri de utilizare din diagrama cazurilor de utilizare).

❖ În faza de proiectare se vor realiza:

➢ 2 diagrame de clase corespunzătoare aplicației soft server și aplicației soft client respectând

principiile DDD și folosind o arhitectură orientată pe servicii (SOA) și minim 5 șabloane de

proiectare;

➢ diagrama entitate-relație corespunzătoare bazei de date;

➢ diagrame de secvență corespunzătoare tuturor cazurilor de utilizare (Observație: numărul

diagramelor de secvență trebuie să fie cel puțin egal cu numărul de cazuri de utilizare din diagrama

cazurilor de utilizare).

❖ În faza de implementare se va scrie cod pentru îndeplinirea tuturor funcționalităților precizate de diagrama cazurilor de utilizare utilizând:

➢ proiectarea dată de diagramele de clase și diagramele de secvență;

➢ unul dintre următoarele limbaje de programare: C#, C++, Java, Python.

❖ Finalizarea temei va consta în predarea unui director ce va cuprinde:

➢ Un fișier cu diagramele UML realizate;

➢ Baza de date;

➢ Aplicația soft;

➢ Documentația (minim 20 pagini) - un fișier care cuprinde:

▪ numele studentului, grupa;

▪ enunțul problemei;

▪ instrumente utilizate;

▪ justificarea limbajului de programare ales;

▪ descrierea diagramelor UML (inclusiv figuri cu diagramele UML realizate);

▪ descrierea aplicației (inclusiv figuri reprezentând interfețele grafice ale aplicației client).

Dezvoltați o aplicație client/server pentru următorul joc: se un labirint de dimensiune nxn căsuţe

în care pot să existe capcane pentru animale sălbatice şi o “comoară” de morcovi. Într-un colţ al

labirintului se află un iepuraș flămând. Ajutaţi-l pe iepuraș să ajungă la “comoara” de morcovi astfel încât să nu fie prins într-una din capcane, ştiind că el se poate deplasa prin sărituri pe diagonală în căsuțele care au un vârf comun cu căsuța curentă. Jocul va fi dezvoltat pe mai multe niveluri (n=4,5,6,7,…). După ce un jucător a obținut soluția, se va afișa numărul de deplasări utilizat de către acesta în soluția furnizată și i se va comunica dacă a obținut soluția optimă. Dacă nu a obținut soluția optimă, aceasta va fi afișată pas cu pas. Soluția optimă se va determina utilizând un algoritm euristic (de exemplu A\*). Jucătorii vor putea să-și creze cont, să se autentifice și să-și vizualizeze punctajul. Jucătorii vor avea posibilitatea, de asemenea, să vizualizeze un grafic cu puntajul propriu raportat la punctajelor celorlați jucători (cum este punctajul său față de punctajul minim, punctajul maxim și punctajul mediu pentru nivelul respectiv).

Utilizatorii de tip administrator pot efectua următoarele operații după autentificare:

❖ Operații CRUD pentru utilizatori;

❖ Vizualizarea listei tuturor jucătorilor care și-au creat cont;

❖ Salvare liste cu informații despre utilizatori în mai multe formate: csv, json, xml, doc;

❖ Vizualizarea unor statistici referitoare la punctajele jucătorilor pe nivele de dificultate utilizând grafice (structură radială, structură inelară, etc.);

❖ Notificarea fiecărui utilizator care necesită autentificare prin cel puțin 2 variante (email, SMS,

WhatsApp, Skype, etc.) la orice modificare a informațiilor de autentificare aferente acelui utilizator. Interfața grafică a aplicației client va fi disponibilă în cel puțin 3 limbi de circulație internațională.

* **Motivație**

Aplicația dezvoltată aduce beneficii multiple prin furnizarea unei experiențe distractive și educative, oferind utilizatorilor un joc de labirint cu mai multe nivele de dificultate, îmbunătățind abilitățile cognitive și promovând gândirea strategică și rezolvarea de probleme. Integrarea unui sistem de autentificare și punctaj stimulează competiția și motivarea utilizatorilor, în timp ce funcționalitățile pentru administratori facilitează gestionarea eficientă a utilizatorilor și a datelor asociate. De asemenea aplicația este disponibilă în 4 limbi de circulație internațională.

1. **Instrumente utilizate**

Instrumentele utilizate pentru proiectarea, implementarea si testarea acestei aplicatii software, sunt:

* IntelliJ IDEA este utilizat ca mediu principal de dezvoltare pentru codul sursă al aplicației. Dezvoltat de compania JetBrains, care oferă suport pentru dezvoltarea de aplicații software în mai multe limbaje de programare, inclusiv Java, Kotlin, Scala, Groovy, JavaScript, TypeScript, HTML, CSS și altele. IntelliJ IDEA vine cu o serie de funcționalități avansate, cum ar fi completarea automată a codului, refactorizarea inteligentă, depanarea integrată, controlul versiunilor și multe altele, care facilitează procesul de dezvoltare și îmbunătățesc productivitatea dezvoltatorilor.
* MySQL utilizat ca sistem de management al bazelor de date (SGBD). MySQL oferă suport pentru Structured Query Language (SQL), facilitând interacțiunea cu baza de date prin intermediul unui set standardizat de comenzi și operații. Am mai folosit MySQL pentru crearea tabelelor, operatii CRUD pe datele tabelelor, tabele relationale.
* StarUML care permite dezvoltatorilor să creeze diagrame UML într-un mod vizual și intuitiv. Am folosit StarUML pentru modelarea si reprezentarea diagramei cazurilor de utilizare, diagramei de clase respectiv a diagramei entity-relationship corespunzatoare bazei de date.

1. **Justificarea limbajului de programare ales**

Am ales ca limbaj de programare Java pentru implementarea acestui joc din urmatoarele motive:

* Fiabilitate și performanță: Java este un limbaj robust și fiabil, ceea ce îl face potrivit pentru dezvoltarea de aplicații critice, cum ar fi jocuri și aplicații interactive. De asemenea, Java beneficiază de un sistem de gestionare a memoriei eficient și de tehnici de optimizare care îmbunătățesc performanța aplicațiilor.
* Portabilitate: Java este cunoscut pentru portabilitatea sa, ceea ce înseamnă că aplicațiile dezvoltate în Java pot fi rulate pe diverse platforme și sisteme de operare fără modificări semnificative.
* Versatilitate: Java este un limbaj versatil, potrivit pentru o varietate largă de aplicații, inclusiv dezvoltarea de jocuri, aplicații desktop, aplicații web și multe altele.
* Siguranța: Java este un limbaj securizat și robust, cu un sistem de securitate puternic integrat. Acest lucru este crucial pentru aplicațiile care gestionează date sensibile sau care rulează în medii critice unde fiabilitatea este esențială.

Pentru partea de interfață grafică a acestui joc am ales Java Swing din următoarele motive:

* Flexibilitate și personalizare: Java Swing oferă o gamă largă de componente grafice și un set flexibil de funcționalități de personalizare. Aceasta vă permite să creați interfețe utilizator complexe și atractive, adaptate nevoilor și cerințelor specifice ale aplicației dvs.
* Compatibilitate cu Java: Fiind parte a platformei Java standard, Java Swing se integrează perfect cu alte tehnologii Java, cum ar fi JDBC pentru accesul la baza de date MySQL și altele. Acest lucru facilitează dezvoltarea și integrarea întregii aplicații.
* Ecosistem larg de biblioteci și framework-uri: Datorită popularității și istoricului său lung, Java Swing are un ecosistem bogat de biblioteci și framework-uri care îmbunătățesc funcționalitățile sale și permit dezvoltatorilor să extindă și să îmbunătățească aplicațiile lor cu ușurință.

1. **Descrierea diagramelor UML**

* **Diagrama de use case**

O diagramă de cazuri de utilizare (Use Case Diagram) este o diagramă în UML (Unified Modeling Language) care descrie interacțiunile între actori (utilizatori externi sau sisteme) și un sistem software în procesul de modelare a cerințelor.

A diagram of a diagram

Description automatically generated

**Figura 1: Diagrama de use case**

* **Diagrama de clase**

Diagrama de clase este un tip de diagramă în cadrul limbajului de modelare UML (Unified Modeling Language) care prezintă structura statică a unui sistem software. Această diagramă descrie clasele din sistem, atributele și metodele acestora, precum și relațiile între clase.

**Diagrama de clase corespunzătoare aplicației soft SERVER:**

A screenshot of a computer

Description automatically generated

**Figura 2: Diagrama de clase pentru Server**

**Diagrama de clase corespunzătoare aplicației soft CLIENT:**

A close-up of a computer screen

Description automatically generated

**Figura 3: Diagrama de clase pentru Client**

* **Diagrama entitate-relație corespunzătoare bazei de date**

Diagrama de entitate-relație (ER) este un instrument de modelare a datelor care este utilizat pentru a reprezenta structura datelor într-o bază de date. Această diagramă se concentrează pe entități (obiecte sau lucruri distincte despre care se dorește să se păstreze informații) și relațiile dintre aceste entități.

A screenshot of a computer

Description automatically generated

**Figura 4: Diagrama entitate-relație corespunzătoare bazei de date**

* **Diagramele de activități**

Pentru fiecare use case din diagrama de use case am realizat o diagramă de activitate(adică numărul diagramelor de activități este egal cu numărul de cazuri

de utilizare din diagrama cazurilor de utilizare).

O să dau câteva exemple de diagrame de activitate pentru anumite use case-uri:

A diagram of a company

Description automatically generated

**Figura 5: Diagrama de activitate pentru use case-ul jucătorului de a vizualiza soluția optimă dacă aceasta nu a fost atinsă**

A diagram of a flowchart

Description automatically generated

**Figura 5.1: Diagrama de activitate pentru use case-ul adminului de a realiza operații CRUD**

A diagram of a diagram

Description automatically generated with medium confidence

**Figura 5.2: Diagrama de activitate pentru use case-ul adminului de a putea salva liste sub format csv,json,xml și doc**

* **Diagramele de secvență**

Am realizat diagrame de secvență corespunzătoare tuturor cazurilor de utilizare (numărul diagramelor de secvență trebuie să fie cel puțin egal cu numărul de cazuri de utilizare din diagrama cazurilor de utilizare).O să dau câteva exemple de diagrame de secvență pentru anumite use case-uri:

A diagram of a diagram

Description automatically generated

**Figura 6: Diagrama de secvență pentru use case-ul jucătorului de a se autentifica**

A diagram of a game

Description automatically generated

**Figura 6.1 : Diagrama de secvență pentru use case-ul jucătorului de a vizualiza numărul de mutări**

A screenshot of a computer

Description automatically generated

**Figura 6.2 : Diagrama de secvență pentru use case-ul jucătorului de a vizualiza soluția optimă dacă aceasta nu a fost atinsă**

1. **Descrierea aplicației**

Această aplicație reprezintă un joc de labirint în care jucătorii trebuie să ajute un iepuraș flămând să găsească "comoara" de morcovi fără a fi prins în capcanele pentru animale sălbatice. Jocul este disponibil în patru niveluri de dificultate și 4 limbi de circulație internațională, iar după ce un jucător completează un nivel, se va afișa numărul de deplasări efectuate și dacă jucătorul a obținut soluția optimă. Dacă aceasta a fost atinsă se va afișa un mesaj, în caz contrar se va afișa soluția optima pas cu pas după încheierea jocului. Jucătorul își poate crea cont și se poate autentifica pentru a-și vizualiza scorul și să vizualizeze un grafic cu punctajul propriu raportat la punctajelor celorlați jucători, iar dacă rolul utilizatorului este de administrator va putea gestiona făcând operații CRUD pe jucătorii autentificați, poate să îi vizualizeze pe toți jucătorii autentificați, poate să vizioneze o statistică sub forma de pie chart cu scorurile tuturor jucătorilor autentificați și va mai putea să salveze liste cu informații despre utilizatori în mai multe formate: csv, json, xml, doc.

* **Partea de joc:**

Interfața Utilizator (UI): Interfața va afișa labirintul, poziția iepurașului și a capcanelor, precum și "comoara" de morcovi. Va permite utilizatorului să selecteze nivelul de dificultate și limba de circulație internaționala în care vrea să navigheze în aplicație și de asemenea să interacționeze cu labirintul pentru a-și ghida iepurașul către "comoară".

Logica Jocului: Aceasta va gestiona generarea labirintului, plasarea capcanelor și a "comoarei" de morcovi și validarea mișcărilor iepurașului conform regulilor jocului.

Gestionarea Scorului: Se va ține evidența numărului de deplasări făcute de utilizator pentru a ajunge la "comoara". Scorul poate fi comparat cu soluția optimă pentru a determina eficiența mișcărilor utilizatorului. Dacă jucătorul a făcut mai multe deplasări decât numărul minim (care reprezintă defapt soluția optimă) atunci lui îi va afișat pe ecran pas cu pas soluția optimă.

Operații jucător autentificat: Un jucător autentificat poate să se joace și să vizualizeze numărul de mutări, să își aleagă nivelul jocului și să vizualizeze un grafic cu punctajul propriu împreuna cu punctajele celorlalți jucători pentru a-și putea compara performanța cu celelalte punctaje obținute de ceilalți jucători.

* **Partea de administrare:**

Autentificare și Autorizare: Se va implementa un sistem de autentificare pentru administratori, care le va permite să acceseze funcționalitățile de administrare.

Gestionarea Jucătorilor: Adăugarea, modificarea și ștergerea jucătorilor din sistem și

vizualizarea listei de jucători autentificați și a informațiilor asociate lor. De asemenea vor putea să saleveze liste cu informațiile despre fiecare jucător în mai multe formate și sa vizualizeze un grafic de tip pie chart cu toți jucătorii autentificați împreună cu scorurile lor

* **Autentificare/înregistrare ca utilizator nou:** În cazul în care vine un nou utilizator el se poate înregistra după care să se logheze și să se bucure de aplicație.

**Arhitectura Client-Server**

Aplicația a fost structurată dupa modelul șablonului arhitectural Client-Server. Vor exista două aplicații, una pentru partea de Server și una pentru partea de Client.

Arhitectura client-server este un model de rețea în care mai multe computere (clienți) solicită și primesc servicii de la un alt computer (server). Acest model este folosit pe scară largă datorită capacității sale de a distribui eficient resursele și de a oferi scalabilitate și flexibilitate.

Elemente principale ale arhitecturii client-server:

Clientul:

* este un dispozitiv sau o aplicație care inițiază cereri către server.
* clientul poate fi un browser web, o aplicație mobilă, un program desktop, etc.
* funcționează la nivelul utilizatorului final, oferind interfața prin care utilizatorul interacționează cu sistemul.

Serverul:

* este un computer sau un program care furnizează servicii clienților.
* serverul procesează cererile primite de la clienți, execută acțiunile necesare și trimite răspunsurile înapoi.
* poate gestiona diverse tipuri de servicii, cum ar fi stocarea de date, autentificarea utilizatorilor, rularea aplicațiilor, etc.

În aplicația noastră, serverul se va ocupa de gestionarea bazelor de date și a operațiilor realizate cu ajutorul bazei de date, în timp ce clientul se va ocupa de interacțiunea cu serverul pentru gestionarea utilizatorilor și a logici jocului și afișarea interfeței grafice pentru ca utilizatorul să poată interacționa cu ea.

Funcționalitatea Client-Server:

Serverul așteaptă conexiuni de la clienți pe portul 8080. Când un client se conectează, serverul acceptă conexiunea și creează un nou UserCommandHandler pentru a gestiona comenzile clientului. Clientul interacționează cu utilizatorul prin interfața grafică și trimite comenzi către server prin intermediul metodelor din UserServer. Serverul primește comenzile, le procesează utilizând UserCommandHandler și interacționează cu baza de date dacă este necesar. Serverul poate returna răspunsuri la client prin intermediul ObjectOutputStream. Comunicarea între client și server este gestionată prin trimiterea și primirea de obiecte serializate. De asemenea atât pe partea de server cât și pe cea de client se utilizează câte un obiect de tip Socket pentru comunicarea între ele.

**Arhitectura Client-Server:**

**Partea de Server:**

* **Pachetul Handler**
* Clasa UserCommandHandler realizează comunicarea între client și baza de date pentru operațiile legate de utilizatori, asigurând o interacțiune corespunzătoare între cele două componente ale aplicației.
* **Pachetul Model**
* Clasa User reprezintă un utilizator al aplicației.Are câmpuri pentru nume de utilizator, email,parolă, scor și rol.
* Clasa Usertype este o enumerare ce definește rolurile posibile ale utilizatorilor: ADMIN sau PLAYER.

**Subpachetul Repo:**

* Clasa DatabaseConnection este responsabilă pentru gestionarea conexiunii cu baza de date MySQL.
* Clasa UserRepo se ocupă de manipularea datelor utilizatorilor în ceea ce privește baza de date. Conține metode pentru crearea, actualizarea, ștergerea, vizualizarea(operații CRUD) utilizatorilor autentificați și metode pentru găsire a utilizatorului după nume și de actualizare a scorului în baza de date. Folosește obiecte de tipul User pentru a reprezenta utilizatorii și enumerarea UserType pentru a defini rolurile utilizatorilor (admin sau jucător).
* Clasa ServerApp reprezintă punctul de intrare al serverului pentru aplicația client-server. Funcția principală main inițializează serverul, așteaptă conexiuni de la clienți și gestionează comenzile primite de la aceștia.

**Partea de Client:**

* **Pachetul Controller**
* Clasa CRUDOperationsController gestionează operațiunile CRUD (Create, Read, Update, Delete) pentru entitatea User într-o aplicație cu interfață grafică. Folosește un obiect UserServer pentru a realiza operațiunile pe utilizatori și un CRUDPageView pentru a afișa și manipula datele utilizatorilor. Metodele clasei permit inserarea, actualizarea și ștergerea utilizatorilor, precum și exportul datelor acestora în formate CSV, JSON, XML și text. De asemenea, clasa poate genera un grafic de tip plăcintă pentru a vizualiza statisticile scorurilor utilizatorilor și permite schimbarea limbii interfeței.
* Clasa GameController gestionează logica jocului labirintului. De asemenea verifică mișcările jucătorului în funcție de evenimentele tastaturii și interacțiunile cu labirintul.
* Clasa HomeController servește ca un controller principal pentru aplicație. Ea gestionează navigarea între diferitele funcționalități ale aplicației, cum ar fi jocul, operațiile CRUD și afișarea graficului de scoruri. Metoda setupEventHandlers setează ascultătorii pentru diferitele evenimente din interfața de utilizator. Metodele playRabbitMazeGame și showGraphScore sunt responsabile pentru pornirea jocului și afișarea graficului de scoruri a tuturor jucătorilor și comparația între acestea. Metoda loginSuccessful este apelată după autentificarea cu succes a unui utilizator și inițializează interfața de utilizator. De asemenea are și aceasta clasă o metodă de changeLanguage care se ocupă de schimbarea limbii interfeței în funcție de ce limbă a fost aleasă de utilizator în momentul când s-a autentificat.
* Clasa LoginController gestionează logica pentru autentificarea utilizatorilor în aplicație. Ea interacționează cu interfața de utilizator pentru a obține numele de utilizator și parola introduse de utilizator și le verifică în baza de date. În funcție de rezultatul verificării, utilizatorul este autentificat sau i se indică că datele introduse sunt incorecte. Metodele createLoginBtnActionListener, createRegisterBtnActionListener și createLangChoiceActionListener sunt responsabile pentru gestionarea evenimentelor declanșate de apăsarea butoanelor din interfața de autentificare. Metoda changeLanguage schimbă limba interfeței de autentificare în funcție de preferințele utilizatorului.
* Clasa RegisterController gestionează logica pentru înregistrarea utilizatorilor în aplicație. Ea interacționează cu interfața de utilizator pentru a obține datele de înregistrare introduse de utilizator și le validează înainte de a crea un nou cont în baza de date.
* **Pachetul Model**
* Clasa MazeGame definește logica jocului de labirint cu iepurași. Inițializează labirintul, poziția iepurașului și poziția "comoarei" de morcovi. Are metode pentru a verifica victoria și capcanele, precum și pentru a valida mișcările iepurașului.
* Clasa MazeGrid este responsabilă pentru gestionarea gridului de joc. Verifică câștigul, capcanele și pozițiile valide în grid. Are și o metodă pentru a începe un joc nou la un anumit nivel și pentru a determina cel mai scurt drum folosind un algoritm de căutare euristică.
* Clasa User reprezintă un utilizator al aplicației.Are câmpuri pentru nume de utilizator, email,parolă, scor și rol.
* Clasa Usertype este o enumerare ce definește rolurile posibile ale utilizatorilor: ADMIN sau PLAYER.
* **Pachetul Server**
* Clasa ServerConnection este o clasă care gestionează conexiunea cu serverul. Ea se ocupă de inițializarea și închiderea conexiunii, precum și de trimiterea și primirea de obiecte către și de la server. Clasa utilizează un singur socket pentru comunicație, creat la inițializarea obiectului.
* Clasa ServerProxy este o clasă intermediară care facilitează comunicarea între alte clase din aplicație și serverul de pe rețea. Ea utilizează o instanță a clasei ServerConnection pentru a trimite și primi obiecte către și de la server.
* Clasa UserServer gestionează operațiile legate de utilizatori în cadrul serverului, comunicând cu acesta prin intermediul clasei ServerProxy. Aceasta facilitează interacțiunea cu baza de date a utilizatorilor și manipularea datelor utilizatorilor.
* **Pachetul View**
* Clasa CRUDPageView este o interfață de utilizator pentru gestionarea operațiilor CRUD (Create, Read, Update, Delete) asociate utilizatorilor. Aceasta include butoane pentru reîmprospătarea, inserarea, ștergerea și actualizarea utilizatorilor dintr-o bază de date. De asemenea, oferă opțiuni pentru afișarea statisticilor de scoruri ale utilizatorilor și descărcarea datelor sub diferite formate.
* Clasa GamePageView reprezintă partea de vizualizare a paginii jocului labirintului. Definește interfața grafică pentru jocul labirintului cu iepuraș și afișează labirintul, iepurașul și elementele labirintului (capcane și morcovi). De asemenea permite afișarea mesajelor de câștig, pierdere și de urmarea afișării a soluției optime.
* Clasa HomePageView reprezintă partea de vizualizare a paginii principale a aplicației. Interacționează cu un obiect de tip HomeController și implementează o interfață grafică pentru navigarea și interacțiunea utilizatorului cu diverse funcționalități ale aplicației. De asemenea, getterele și setterele facilitează comunicarea între partea de vizualizare și partea de control a aplicației în cadrul arhitecturii MVC, permițând accesul controlat la date și acțiuni. Controller-urile pot utiliza gettere pentru a obține datele introduse de utilizator din interfața grafică și settere pentru a actualiza modelul cu datele introduse de utilizator sau cu datele obținute din alte surse.
* Clasa LoginPageView este responsabilă pentru reprezentarea interfeței de utilizator asociate paginilor de login. Clasa LoginPageView conține: elementele vizuale pentru pagina de login, cum ar fi câmpurile de introducere a numelui de utilizator și a parolei, butoanele de login și înregistrare etc, metode pentru a obține textul introdus în câmpurile de utilizator și parolă, metode pentru a seta textul butoanelor și etichetelor.
* Clasa RegisterPageView este responsabilă pentru reprezentarea interfeței de utilizator asociate paginii de înregistrare. Clasa RegisterPageView conține: elementele vizuale pentru pagina de înregistrare, cum ar fi câmpurile de introducere a emailului, parolei și numelui de utilizator, butonul de înregistrare etc. ,metode pentru a obține textul introdus în câmpurile de email, parolă și nume, metode pentru a seta textul butonului și etichetelor.

**Design pattern-uri folosite:**

* Singleton - utilizat pentru obținerea unei singure instanțe de conexiune la baza de date
* Proxy - acționează ca un proxy pentru ServerConnection, abstractizând complexitatea conexiunii și comunicării cu serverul.
* MVC(Model-View-Controller) - Separă aplicația în trei componente distincte pentru gestionarea logicii de joc, prezentarea datelor și interacțiunea cu utilizatorul. MVC practic este o arhitectură.

**Câteva poze din aplicație:**

* Pagina de login

A screenshot of a computer

Description automatically generated

**Figura 7: Pagina de login**

* Pagina de Register în care orice utilizator care vine își poate face cont

A screenshot of a computer

Description automatically generated

**Figura 7.1: Pagina de Register(în limba franceză)**

* Pagina de Home pentru ADMIN

A screenshot of a computer

Description automatically generated

**Figura 7.1: Pagina de Home pentru utilizatorul de tip ADMIN( în limba italiană)**

* Pagina de Home pentru Player

A screenshot of a computer

Description automatically generated

**Figura 7.2: Pagina de Home pentru utilizatorul de tip Player( în limba română)**

* Pagina de Rabbit Maze Game

A rabbit and carrots and carrots

Description automatically generated

**Figura 7.3: Pagina de Rabbit Maze Game**

* Pagina de Rabbit Maze Game și momentul când ai câștigat dar nu ai obținut soluția optimă => soluția optimă va fi afișată pas cu pas pe ecran

A screenshot of a check mark and check boxes

Description automatically generated

**Figura 7.3: Pagina de Rabbit Maze Game + win dar fără soluție optimă**

* Pagina de vizualizare a graficului unui jucător cu puntajul propriu raportat la punctajelor celorlați jucători și comparația cu acestea

A screenshot of a graph

Description automatically generated

**Figura 7.4: Pagina de vizualizare a punctajului propriu și a punctajelor celorlalți jucători**

* Pagina dedicată pentru Admin în care acesta poate realiza operații CRUD, să salveze liste cu utilizatori în mai multe formate și să vizualizeze un pie chart cu punctajele tuturor jucătorilor

A screenshot of a computer

Description automatically generated

**Figura 7.5: Pagina dedicată utilizatorului de tip ADMIN**

* Pagina pentru a vizualiza punctajele tuturor jucătorilor sub forma de pie chart

A screenshot of a pie chart

Description automatically generated

**Figura 7.6: Pie chart pentru ADMIN cu punctajele tuturor jucătorilor**

* Listele cu informații despre utilizatori pe care utilizatorul de tip ADMIN o poate descărca și vizualiza apoi

A number of days and months

Description automatically generated

**Figura 7.7: Listele cu informații sub formatele csv, doc, json și xml**

1. **Concluzii**

În concluzie, acest proiect reprezintă o oportunitate excelentă de a se familiariza cu șablonul architectural Client/Server, cu arhitectura orientată pe servicii (SOA) și cu diverse șabloane de proiectare. Prin adoptarea arhitecturii Client/Server, am separat clar funcționalitățile între aplicația client și server. Aplicația client se ocupă de interfața grafică, de interacțiunea cu utilizatorii și de prezentarea datelor, în timp ce serverul gestionează logica afacerii, procesarea datelor și interacțiunea cu baza de date relațională. Această separare permite actualizarea și extinderea independentă a fiecărei componente, ceea ce crește flexibilitatea și ușurința în mentenanță a sistemului.

Prin utilizarea șabloanelor de proiectare și adoptarea arhitecturilor Client/Server și SOA, am reușit să îndeplinim cerințele specificate și să dezvoltăm un joc de labirint complex, accesibil și util atât pentru utilizatorii obișnuiți, cât și pentru administratori.