

**Tema 3 -Arrow Puzzle Game**

Model – View – Controller

Problema 35

**Proiectare Sofware**

**-Calculatoare si Tehnologia Informatiei-**

Student: Iamnitchi Bogdan

Grupa: 30235

An: 3

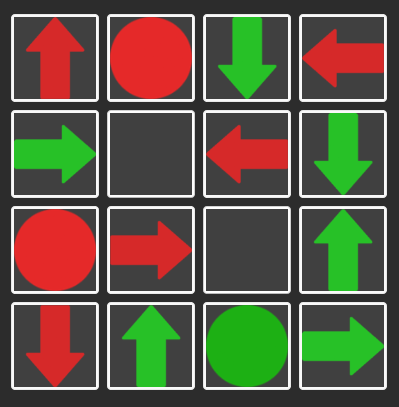
1. **Enuntul problemei**

Dezvoltați o aplicație pentru următorul joc cu 2 jucători: Se consideră săgeți de 2 culori, iar scopul jocului este de a completa zonele din pătrat astfel încât pe aceeași linie, pe aceeași coloană și pe aceeași diagonală să nu se găsească 2 săgeți orientate în aceeași direcție, indiferent de culoarea acestora. Pierde jucătorul care nu mai are nicio posibilitate de a așeza săgeți în zonele libere ale pătratului. Implementarea se va realiza astfel încât un utilizator al aplicației (jocului) să joace cu calculatorul. Se va utiliza o variantă a algoritmului MINIMAX.

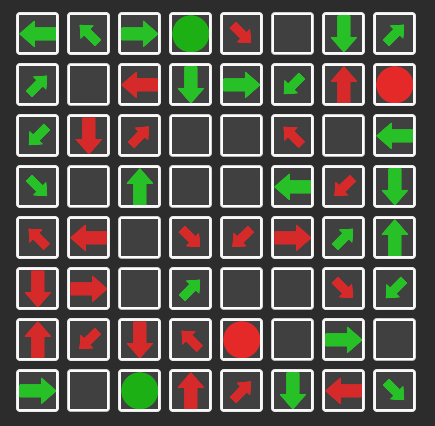
**Jucătorii** vor putea să se autentifice și să-și vizualizeze punctajul.

Utilizatorii de tip **administrator** pot efectua următoarele operații după autentificare:

* Operații CRUD pentru informațiile legate de jucătorii care s-au autentificat;
* Vizualizarea listei tuturor jucătorilor autentificați.

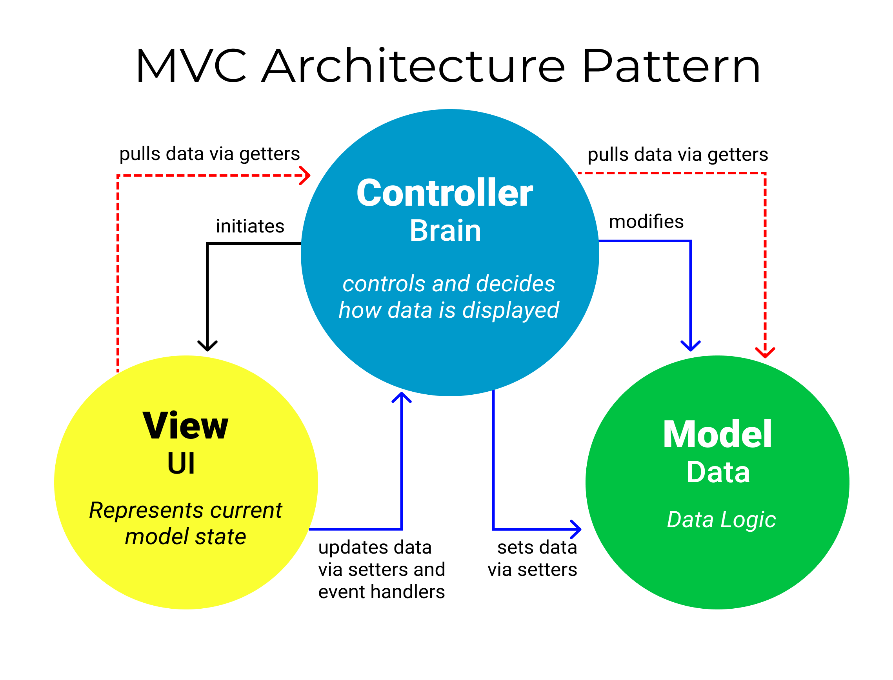
**Nivel 1:**

*Figura 1: Nivel 1*

**Nivel 2:**

*Figura 2: Nivel 2*

**Arhitectutura MVC**

**Model-view-controller (MVC)** (din engleză, aproximativ: model-vizualizare-controlor) este un model arhitectural utilizat în ingineria software. Succesul modelului se datorează izolării logicii de business față de considerentele interfeței cu utilizatorul, rezultând o aplicație unde aspectul vizual sau/și nivelele inferioare ale regulilor de business sunt mai ușor de modificat, fără a afecta alte nivele.

*Figura 3: Arhitectura MVC*

Modelul (Model):

* Reprezintă datele și logica de business a aplicației.
* Poate include accesul la baza de date, manipularea datelor și alte operații de business logic.
* Nu cunoaște nimic despre modul în care datele sunt afișate sau despre interacțiunea utilizatorului.

Vizualizarea (View):

* Este responsabilă pentru afișarea datelor utilizatorului și interacțiunea cu utilizatorul.
* Nu conține logica de business sau acces la date.
* Primește datele de la Model și le afișează într-un mod adecvat pentru utilizator.

Controlerul (Controller):

* Este responsabil pentru gestionarea interacțiunii utilizatorului și actualizarea Modelului sau Vizualizării în consecință.
* Primește intrările de la utilizator (de exemplu, clicuri de buton, introducerea de date) și inițiază modificările corespunzătoare în Model sau View.
* Servește ca intermediar între Model și View, asigurându-se că acestea două nu sunt dependente direct unele de altele.

Arhitectura MVC este utilizată pe scară largă în dezvoltarea aplicațiilor web și desktop, deoarece promovează o separare clară a preocupărilor și un cod bine organizat, ceea ce face ca aplicațiile să fie mai ușor de întreținut și de dezvoltat în echipă.

1. **Instrumente utilizate**

Am dezvoltat aplicația în C# folosind mediul de dezvoltare integrat (IDE) Visual Studio. Acest IDE mi-a oferit un mediu puternic de dezvoltare, incluzând funcții precum completarea automată a codului, depanare și gestionarea proiectelor, facilitând astfel procesul de dezvoltare.

Pentru a scrie codul aplicației, am folosit limbajul de programare C#. Acesta este un limbaj puternic, orientat pe obiecte, dezvoltat de Microsoft, cu o sintaxă curată și ușor de înțeles, ceea ce mi-a permis să dezvolt aplicația cu eficiență și să obțin rezultatele dorite.

Aplicația desktop pe care am dezvoltat-o a fost construită folosind framework-ul .NET. Acest framework a oferit o colecție bogată de biblioteci și un mediu de rulare pentru dezvoltarea și executarea diferitelor tipuri de aplicații, inclusiv aplicații desktop, ceea ce mi-a permis să creez un produs final robust și funcțional.

Pentru a gestiona interfața utilizatorului și interacțiunile cu utilizatorul, am utilizat biblioteci grafice precum Windows Presentation Foundation (WPF) sau Windows Forms. Aceste biblioteci au facilitat crearea unei interfețe grafice atractivă și intuitivă pentru utilizator, contribuind la experiența generală a jocului.

În ceea ce privește sunetul și muzica din joc, am folosit biblioteci precum NAudio sau alte soluții care permit redarea și manipularea sunetului, adăugând astfel o dimensiune suplimentară experienței de joc.

Deși opțional, am integrat și un sistem de bază de date pentru a gestiona salvarea datelor utilizatorului sau a progresului în joc. Am putut folosi baze de date precum SQL Server, SQLite sau alte soluții de stocare a datelor pentru a asigura o experiență personalizată și persistentă pentru utilizatori.

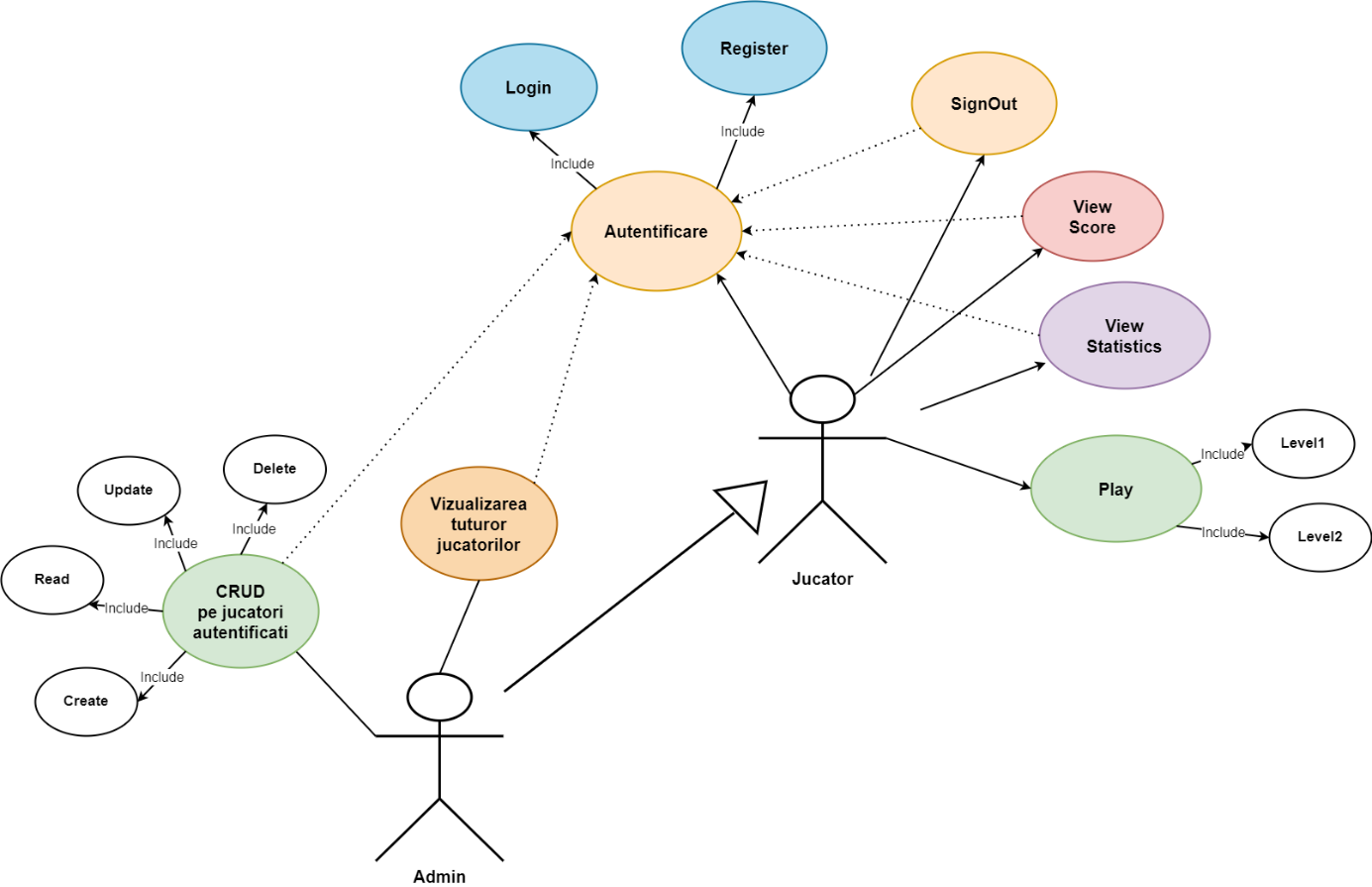
Pentru gestionarea codului sursă al aplicației și pentru a lucra eficient în echipă, am utilizat un sistem de control al versiunilor, cum ar fi Git. Acest lucru mi-a permis să monitorizez și să gestionez modificările în codul sursă al aplicației într-un mod organizat și colaborativ.

Acestea sunt instrumentele principale pe care le-am utilizat în dezvoltarea aplicației mele în C#, contribuind la crearea unui joc captivant și funcțional pentru utilizatori.

1. **Proiectare si implementare**

**Diagrama USE CASE**

Prima etapa in realizarea acestei teme, am creat diagrama de use-case in aplicatia Drawio unde am pus actorii principali precum si cazurile de utilizare cele mai importante pentru a avea o privire de ansamblu asupra a ceea ce urmeaza sa implementez.



*Figura 4: Diagrama UseCase*

Aici se afla Use-Case-ul problemei. Drept actori, avem un administrator si un Jucator, acestia putand face diverse actiuni.

**Autentificare Administrator:**

* Actor Principal: Administratorul
* Descriere: Acest caz de utilizare permite administratorului să se autentifice în sistem pentru a avea acces la funcționalitățile administrative.
* Acțiuni:
* Administratorul introduce numele de utilizator și parola în interfața de autentificare.
* Sistemul verifică credențialele și autentifică administratorul.
* Dacă autentificarea reușește, administratorul are acces la funcționalitățile administrative (CRUD pe jucători etc.).
* Dacă autentificarea eșuează, administratorul primește un mesaj de eroare și nu are acces la funcționalitățile administrative.

**CRUD Player (Create, Read, Update, Delete Player):**

* Actor Principal: Administratorul
* Descriere: Acest caz de utilizare permite administratorului să creeze, să vizualizeze, să actualizeze și să șteargă informații despre jucători în sistem.
* Acțiuni:
  + Administratorul accesează secțiunea de administrare a jucătorilor din interfața de administrare.
  + Administratorul poate crea un nou jucător, specificând detalii precum nume, scor etc.
  + Administratorul poate vizualiza detalii despre un jucător existent, inclusiv scorul său.
  + Administratorul poate actualiza informațiile unui jucător existent, cum ar fi scorul acestuia.
  + Administratorul poate șterge un jucător din sistem.

**Autentificare Jucător:**

* Actor Principal: Jucătorul
* Descriere: Acest caz de utilizare permite jucătorului să se autentifice în sistem pentru a avea acces la funcționalitățile destinate jucătorilor.
* Acțiuni:
  + Jucătorul introduce numele de utilizator și parola în interfața de autentificare.
  + Sistemul verifică credențialele și autentifică jucătorul.
  + Dacă autentificarea reușește, jucătorul are acces la funcționalitățile destinate jucătorilor (vizualizarea scorului, jocul etc.).
  + Dacă autentificarea eșuează, jucătorul primește un mesaj de eroare și nu are acces la funcționalitățile destinate jucătorilor.

**Vizualizare Scor:**

* Actor Principal: Jucătorul
* Descriere: Acest caz de utilizare permite jucătorului să vizualizeze scorul său în sistem.
* Acțiuni:
  + Jucătorul accesează secțiunea destinată vizualizării scorului din interfața de joc.
  + Sistemul afișează scorul jucătorului în interfața de joc.

**Joc:**

* Actor Principal: Jucătorul
* Descriere: Acest caz de utilizare permite jucătorului să joace jocul.
* Acțiuni:
  + Jucătorul accesează secțiunea de joc din interfața de joc.
  + Jucătorul interacționează cu jocul conform mecanicilor și regulilor stabilite.
  + Sistemul actualizează scorul jucătorului în timpul jocului conform performanței sale.

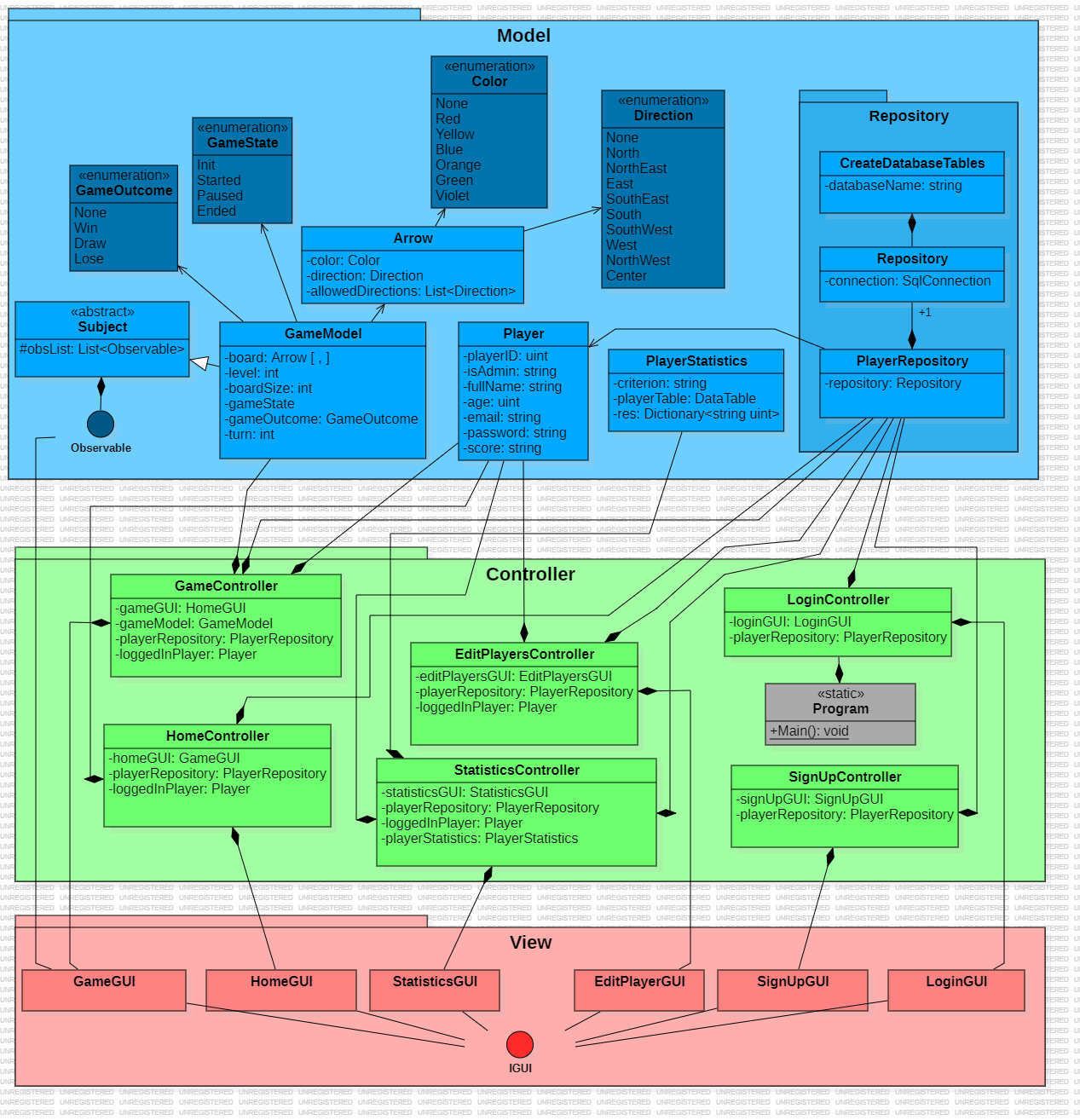
**Vizulaizare Statistici:**

* Actor Principal: Jucătorul
* Descriere: Acest caz de utilizare permite jucătorului să joace jocul.
* Acțiuni: Jucătorul accesează secțiunea de joc din interfața de joc.

**Diagrama de clase**

De asemenea, tot in acest tool, am facut si diagrama de clase a aplicatiei.

Voi prezenta o diagrama de clase in care am pus doar atributele ca sa se poata vedea cu usurina legatura dintre clase si pachete. Acestea au un nume sugestiv: Model, View si Controller. Fiecare pachet are mai multe clase implementate in aplicatie, acestea avand atributele si operatiile necesare functionarii aplicatiei.



*Figura 5: Diagrama de Clase*

Diagrama de clase în cadrul arhitecturii MVC (Model-View-Constroller) este concepută pentru a ilustra modul în care diferitele componente ale modelului, vederii și controller-ul sunt conectate și interacționează între ele. În cadrul MVC, fiecare componentă are roluri și responsabilități distincte.

**Modelul (Model):**

* Clasa Arrow: Această clasă definește caracteristicile săgeților din joc, cum ar fi culoarea și direcția, și gestionarea direcțiilor în care pot fi mutate.
* Clasa GameModel: Responsabilă pentru gestionarea stării generale a jocului, inclusiv a tablei de joc, a nivelului curent, a stării jocului și a rezultatelor.

**Vizualizarea (View):**

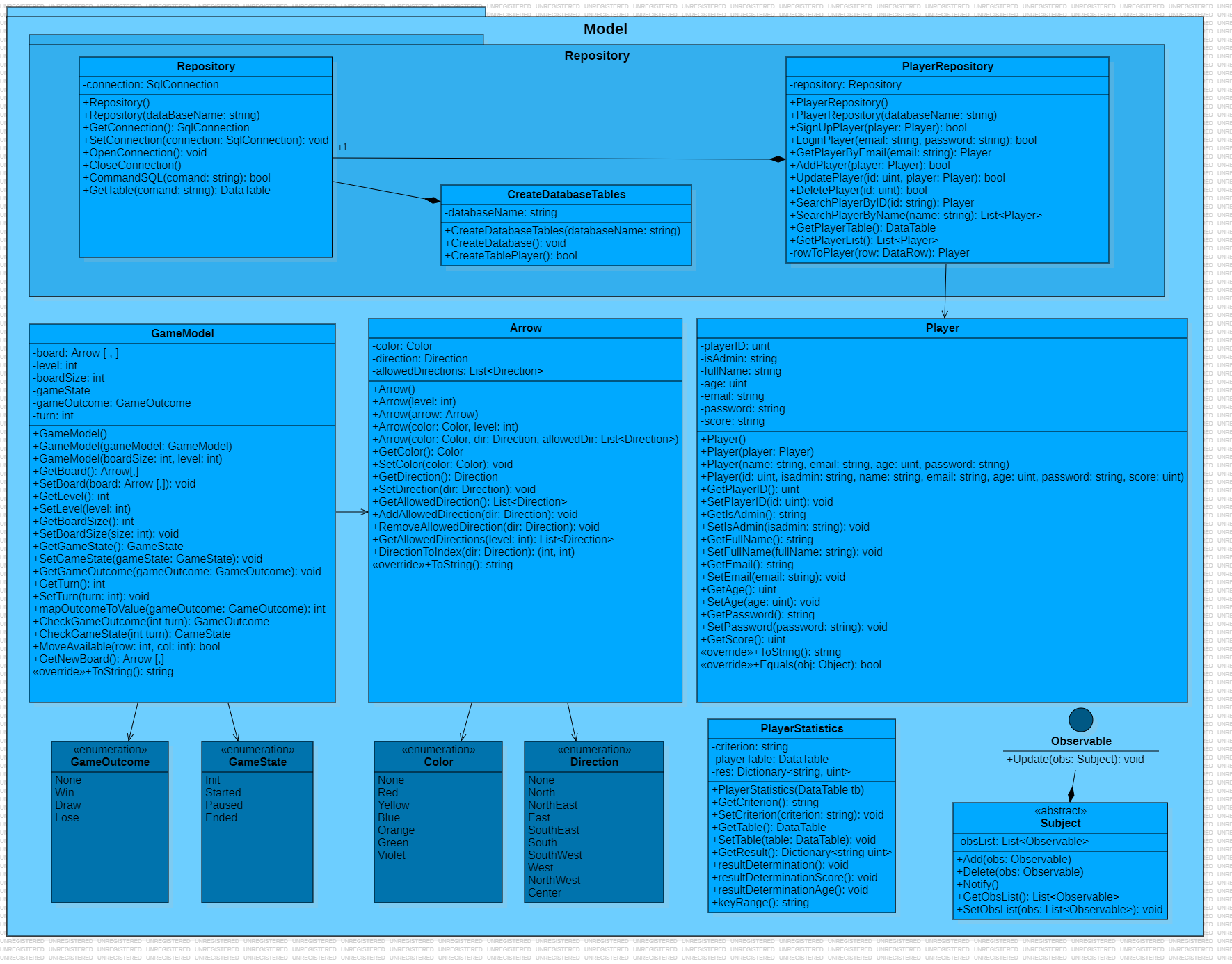
* Controllere GUI:
* SignUpGUI: Interfața grafică pentru înregistrarea unui nou jucător.
* LoginGUI: Interfața grafică pentru conectarea unui jucător existent.
* HomeGUI: Ecranul principal al jocului, unde utilizatorul poate naviga către diferite secțiuni ale aplicației.
* StatisticsGUI: Interfața grafică pentru vizualizarea statisticilor jocului.
* GameGUI: Interfața grafică pentru desfășurarea jocului.
* EditPlayersGUI: Interfața grafică pentru editarea informațiilor despre jucători.

**Controlerul (Controller):**

* HomeController: Se ocupă de coordonarea acțiunilor și interacțiunilor din ecranul principal al jocului.
* LoginController: Responsabil pentru gestionarea autentificării jucătorilor și interacțiunile din interfața de conectare.
* SignUpController: Gestionează evenimentele și interacțiunile din interfața de înregistrare a jucătorilor.
* StatisticsController: Gestionarea afișării și interacțiunilor în cadrul interfeței grafice de statistică.
* GameController: Responsabil pentru gestionarea logicii de joc și a interacțiunilor în cadrul interfeței grafice de joc.
* EditPlayersController: Se ocupă de gestionarea evenimentelor și interacțiunilor din interfața de editare a informațiilor despre jucători.

În cadrul acestei arhitecturi MVC, Modelul (Model) gestionează datele și logica jocului, inclusiv caracteristicile săgeților și starea generală a jocului. Vizualizarea (View) reprezintă interfața utilizatorului, cu diferitele sale ecrane și elemente grafice, în timp ce Controlerul (Controller) acționează ca intermediar între Model și View, coordonând interacțiunile utilizatorului și logica de afișare a datelor. Prin separarea clară a acestor componente și definirea rolurilor lor distincte, se asigură o structură bine organizată a jocului, ușor de dezvoltat, testat și întreținut în timp.

**Pachetul Model:**



*Figura 6: Pachetul Model*

**CreateDatabaseTables:**

* Această clasă este responsabilă pentru crearea tabelelor din baza de date necesare pentru stocarea datelor jucătorilor sau altor informații relevante pentru joc.
* Poate conține metode pentru a executa scripturi SQL pentru crearea tabelelor sau alte operațiuni necesare pentru inițializarea bazei de date.

**PlayerRepository:**

* Această clasă este responsabilă pentru gestionarea datelor jucătorilor din baza de date. Ea oferă operații CRUD (Create, Read, Update, Delete) pentru a interacționa cu datele jucătorilor.
* Metodele acestei clase pot include:
  + addPlayer(): pentru a insera un nou jucător în baza de date.
  + getPlayer(): pentru a obține informații despre un jucător specific din baza de date.
  + updatePlayer(): pentru a actualiza informațiile unui jucător existent în baza de date.
  + deletePlayer(): pentru a șterge un jucător din baza de date.

**Repository:**

* Această clasă este responsabilă pentru gestionarea conexiunii cu baza de date. Ea oferă metode pentru a stabili și a închide conexiunea cu baza de date.
* În plus, poate conține și metode pentru a executa interogări SQL și a gestiona tranzacțiile în cadrul bazei de date.
* Prin intermediul acestei clase, celelalte clase din pachetul repository pot comunica cu baza de date și pot efectua operațiile necesare.
* Prin intermediul acestor clase, pachetul repository din cadrul modelului oferă o abstracție eficientă pentru gestionarea interacțiunii cu baza de date. Acest lucru permite modelului să rămână independent de detaliile specifice ale bazei de date și să ofere o interfață simplificată pentru a efectua operațiile CRUD necesare pentru gestionarea datelor jucătorilor în cadrul jocului tău.

**Clasa Arrow:**

* Această clasă reprezintă o săgeată în jocul cu săgeți. Fiecare săgeată are o culoare și o direcție specifică, precum și o listă de direcții permise pentru deplasare.
* Arrow se ocupă de definirea caracteristicilor individuale ale săgeții, inclusiv culoarea și direcția, și de specificarea direcțiilor în care poate fi mutată în timpul jocului.

**Clasa GameModel:**

* Această clasă gestionează starea și logica generală a jocului. Ea se ocupă de lucruri precum tabla de joc, nivelul curent, starea jocului, rezultatul și al cui este rândul.
* GameModel se ocupă de inițializarea și gestionarea stării generale a jocului, inclusiv gestionarea tablei de joc și a evoluției partidei de-a lungul timpului.

**Clasa Player:**

* Această clasă reprezintă un jucător în cadrul jocului. Ea gestionează informații specifice despre fiecare jucător, precum numele, vârsta, scorul și drepturile de administrator.
* Player se ocupă de definirea și gestionarea datelor individuale ale fiecărui jucător, inclusiv informații personale și progresul în joc.

**Interfața Observable:**

* Această interfață definește metoda UPDATE care pe care fiecare observator trebuie sa o implementeze

**Clasa Subject:**

* Această clasa reprezintă subiectul care este observat de obiectele observatoare.
* Include metode pentru gestionarea listei de observatori (cum ar fi add(Observable observer) și delete(Observable observer)) și pentru notificarea observatorilor (Notify()).

**Clasa PlayerStatistics:**

* Această clasă primește un tabel din baza de date și creează un dicționar cheie-valoare în funcție de un criteriu specificat.
* Poate include o metodă cum ar fi generateStatistics(DatabaseTable table, String criterion) care primește un tabel din baza de date și un criteriu specific și returnează un dicționar ce conține statisticile corespunzătoare.

**Pachetul Controller:**



*Figura 7: Pachetul Controller*

Pachetul controller servește ca nucleu al aplicației, găzduind clasele care mediează interacțiunea între interfața utilizatorului și logica de business a sistemului. Aceste clase preiau cererile și acțiunile utilizatorului din interfețele grafice și le direcționează către modelele și serviciile corespunzătoare pentru a le procesa. Ele facilitează transferul datelor și comunicarea între diferitele componente ale aplicației, contribuind la crearea unei structuri bine organizate și modularizate. Prin intermediul acestui pachet, se asigură separarea clară a responsabilităților și un nivel ridicat de coeziune în cadrul aplicației, ceea ce facilitează dezvoltarea, testarea și întreținerea acesteia.

**LoginController:**

* Se ocupă de gestionarea procesului de autentificare a jucătorilor în aplicație.
* Primește cererile de autentificare direct din interfața de autentificare și validează informațiile introduse de utilizator.
* Interacționează direct cu GUI-ul de autentificare pentru a valida și a procesa cererile.

**SignUpController:**

* Gestionează procesul de înregistrare a noi jucători în aplicație.
* Primește datele introduse de utilizator direct din interfața de înregistrare și validează aceste informații.
* Interacționează direct cu GUI-ul de înregistrare pentru a procesa cererile și a crea un nou jucător în sistem.

**HomeController:**

* Se ocupă de gestionarea interacțiunilor legate de ecranul principal al aplicației.
* Primește comenzi și acțiuni din partea utilizatorului direct din interfața de ecran principal și le procesează.
* Gestionează navigarea între diferitele secțiuni ale aplicației direct din GUI-ul ecranului principal.

**StatisticsController:**

* Gestionarea afișării și interacțiunilor în cadrul interfeței grafice de statistică.
* Primește solicitările utilizatorului pentru afișarea statisticilor direct din interfața grafică și le procesează.
* Interacționează direct cu GUI-ul de statistică pentru a obține și a afișa datele necesare.

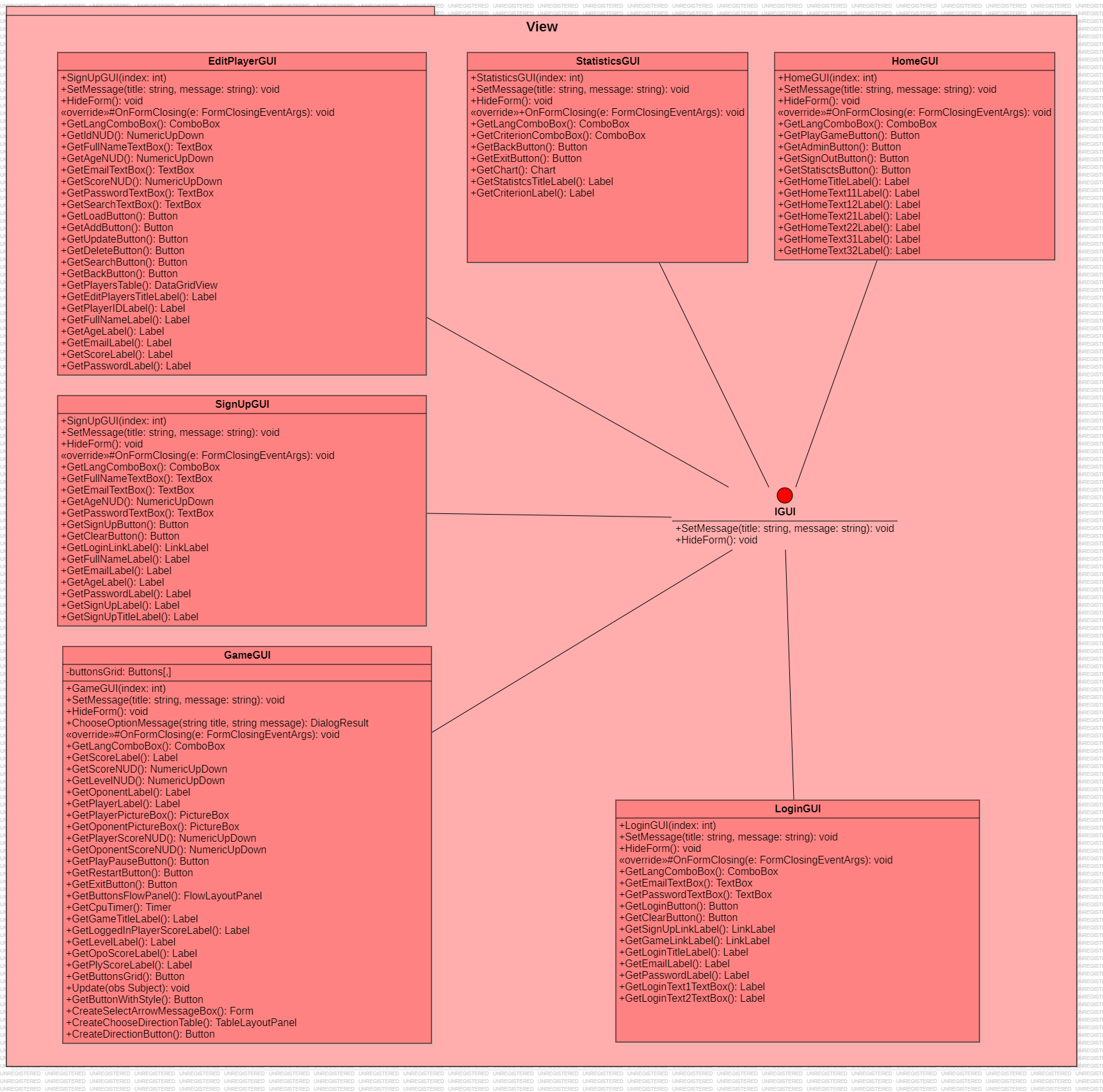
**EditplayersController:**

* Responsabil pentru gestionarea interacțiunilor legate de editarea jucătorilor în joc.
* Primește comenzi de la utilizator direct din interfața de editare a jucătorilor și le procesează.
* Interacționează direct cu GUI-ul de editare a jucătorilor pentru a gestiona adăugarea, editarea și ștergerea jucătorilor din sistem.

**GameController:**

* Se ocupă de gestionarea logicii de joc și a interacțiunilor în cadrul interfeței grafice de joc.
* Primește acțiuni și comenzi de la utilizator direct din interfața de joc și le procesează.
* Interacționează direct cu GUI-ul de joc pentru a actualiza starea jocului și a afișa informațiile relevante.

**Pachetul View:**



*Figura 8: Pachetul View*

**StatisticsGUI:**

* Afișează și gestionează interfața grafică pentru statisticile aplicației.
* Implementează interfața IGUI pentru funcționalitatea interfeței grafice.
* Afisează două grafice distincte:
  + Graficul scorului.
  + Graficul vârstei.
* Oferă o perspectivă vizuală asupra datelor și tendințelor relevante.

**GameGUI:**

* Se ocupă de afișarea și gestionarea interfeței grafice asociate jocului.
* Oferă metode pentru adăugarea și modificarea componentelor GUI specifice jocului, cum ar fi tabla de joc sau elementele pentru afișarea scorului.
* Are gettere pentru componente necesare pentru controller și gettere pentru labeluri, care vor fi traduse în funcție de limba selectată.

**HomeGUI:**

* Prezintă ecranul principal al aplicației și afișează regulile jocului și opțiunile de navigare.
* Oferă metode pentru adăugarea și modificarea componentelor GUI specifice ecranului principal, precum și pentru gestionarea navigării între diferitele ecrane ale aplicației.
* Dispune de gettere pentru componente esențiale pentru controller și pentru labeluri care vor fi traduse în funcție de limba selectată.

**EditPlayersGUI:**

* Se ocupă de afișarea și gestionarea interfeței grafice asociate editării jucătorilor din joc.
* Furnizează metode pentru adăugarea și modificarea componentelor GUI specifice editării jucătorilor, cum ar fi lista jucătorilor sau opțiunile de editare.
* Conține gettere pentru componente necesare pentru controller și pentru labeluri care vor fi traduse în funcție de limba selectată.

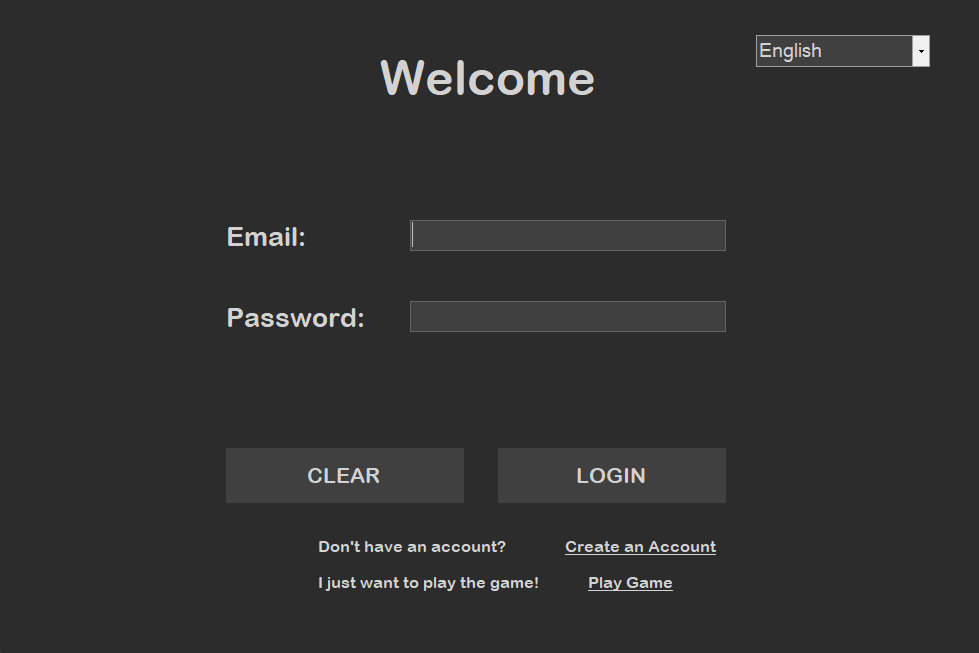
**LoginGUI:**

* Responsabil pentru afișarea și gestionarea interfeței grafice asociate procesului de autentificare a jucătorilor.
* Furnizează metode pentru adăugarea și modificarea componentelor GUI specifice procesului de autentificare, cum ar fi câmpurile pentru introducerea numelui de utilizator și a parolei.
* Include gettere pentru componente esențiale pentru controller și pentru labeluri care vor fi traduse în funcție de limba selectată.

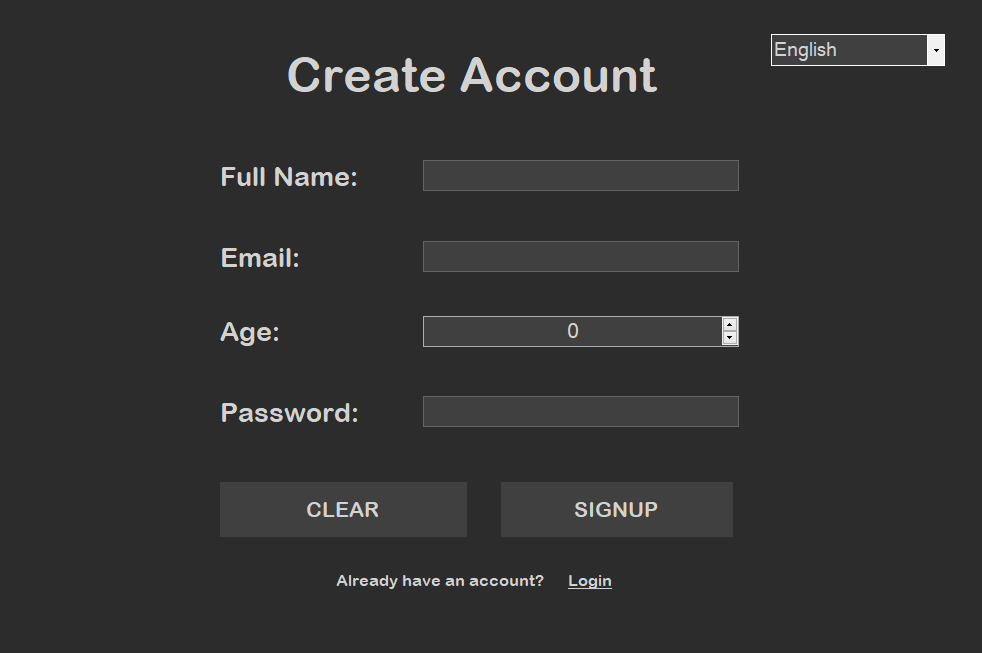
**SignUpGUI:**

* Se ocupă de afișarea și gestionarea interfeței grafice asociate procesului de înregistrare a unui nou jucător în aplicație.
* Furnizează metode pentru adăugarea și modificarea elementelor GUI specifice înregistrării.
* Include gettere pentru componente necesare pentru controller și pentru labeluri care vor fi traduse în funcție de limba selectată.

1. **Descrierea aplicatiei:**

Cand pornim aplicatia clasa program va instantia clasa LoginGUI care ne va intampina cu o fereastra in care sa ne introducem credentialele pentru a ne putea autentifica. Daca nu avem cont putem apasa mai jos pe buttonu de create account care ne va duce pe pagina de SignUp pentru a ne creea un cont. Desigur user-ul poate trece direct la jocul propriu zis fara sa isi faca cont insa scorul acestuia nu va fi retinut astfel. Desigur mai exista si butonul de exit care inchide aplicatia, si buton de clear care sterge campurile completate.

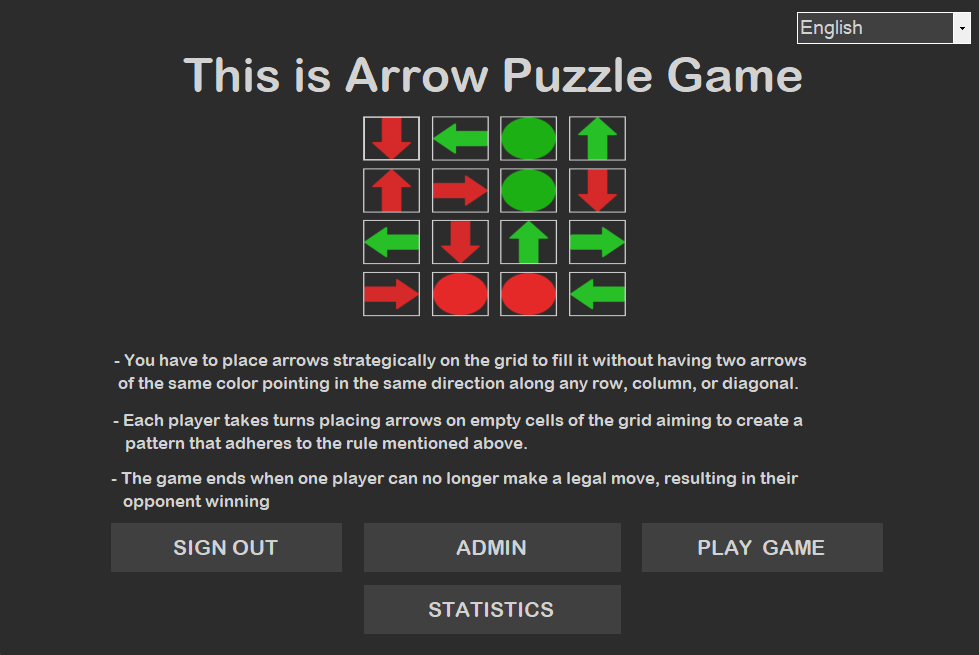
*Figura 9:Pagina de Login*

Daca algem optiuna de create account aceasta ne va duce spre pagina de SignUp in care o sa ni se ceara mai multe detalii care vor putea identifica unic jucatorul, de exemplu emai-ul. Si alte detalii precum varsta numele, scorul. Observam ca avem si optiunea de a naviga inapoi la pagina de Login daca am intrat din greseala.

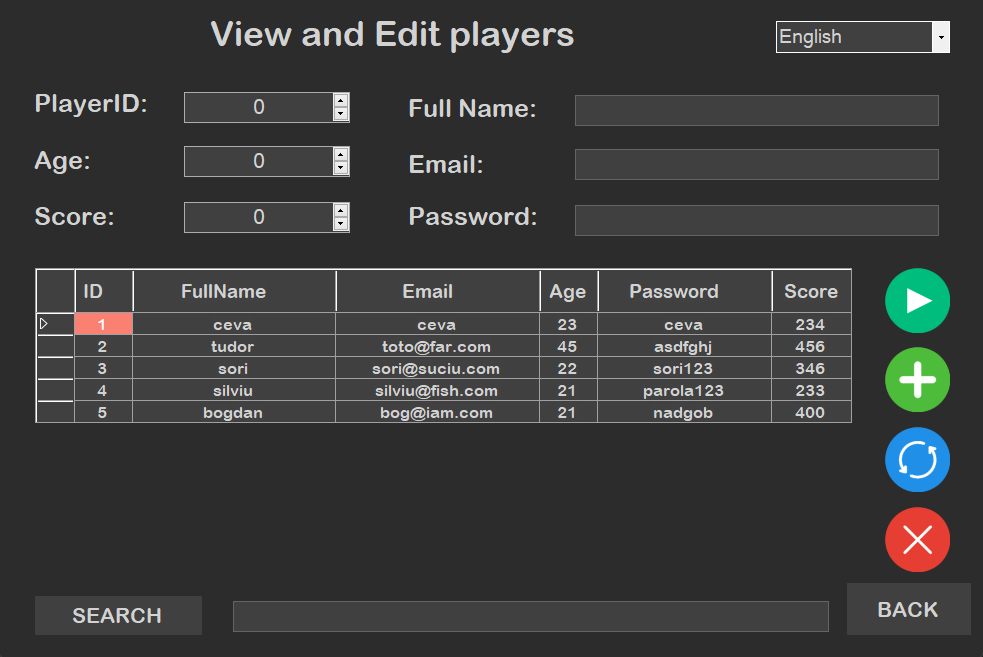
*Figura 10: Pagina de SignUp*

Si aici avem buton de exit care inchide aplicatia Clear pentru a sterge toate campurile completate si buton-ul de signup care ne va inregistra in baza de date iar dupa ne va duce inapoi pe login ca sa ne autentificam cu noile credentiale.

Daca acesti pasi au avut loc cu succes o sa ne apara un MessageBox care ne arata ca ne am

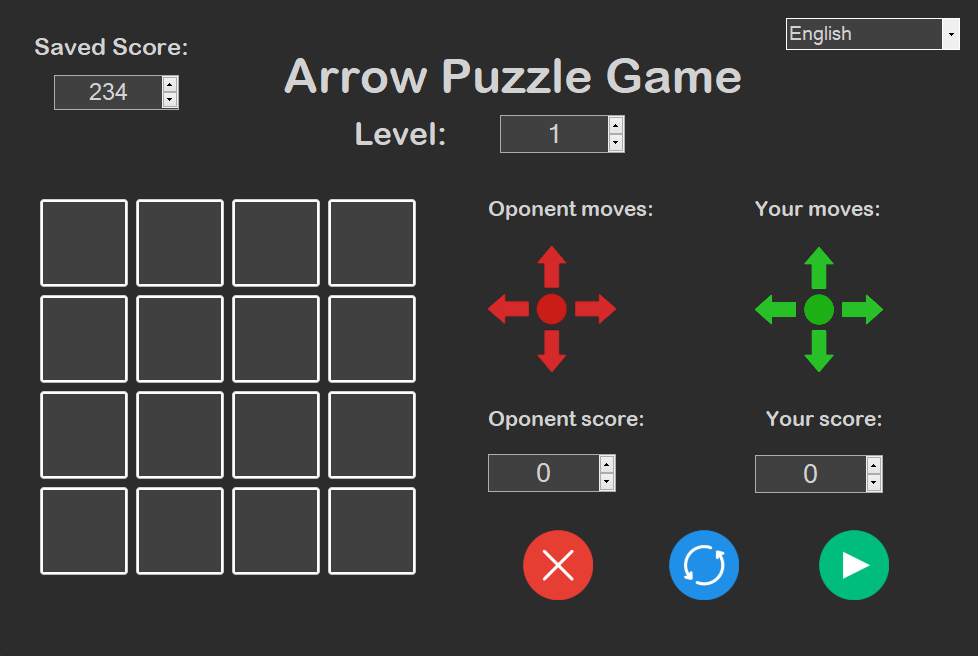
autentificat si totul este in regula si vom ajunge astfel pe pagina de Home unde ne sunt explicate regulile jocului si de asemenea si o ilustratie. Desigur mai avem si aici butonul de exit care va inchide aplicatia daca asta e ceea ce ne dorim.

*Figura 11: Pagina de Home*

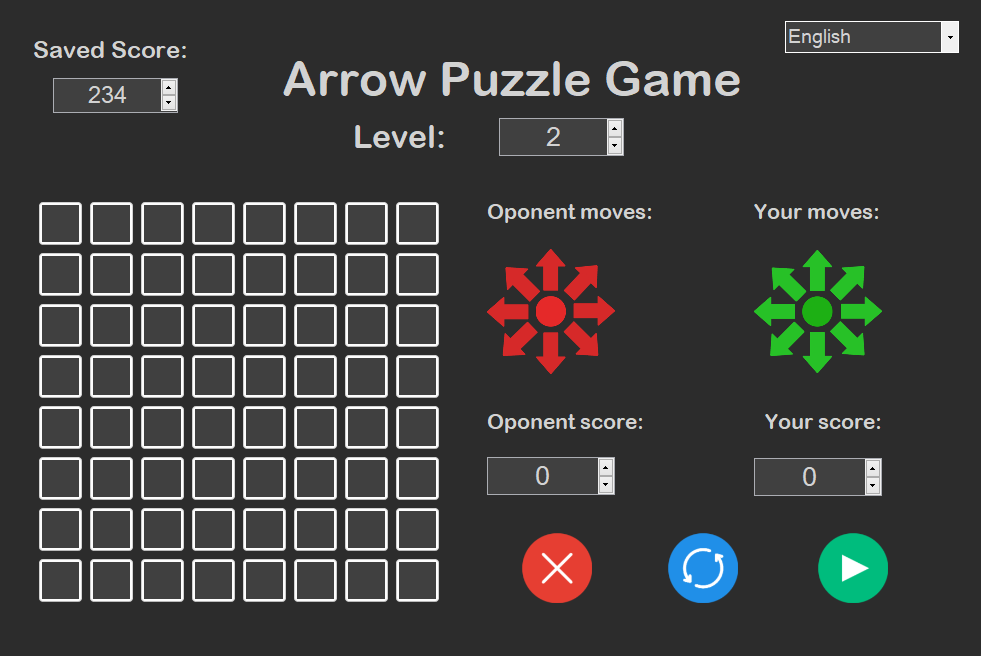
Exista doua tipuri de jucatori, jucatori simpli care isi pot face cont si se pot juca si cei de tip admin care sunt administratori, acestia au un cont special gata creat inainte ca aplicatia sa ruleze. Acestia trebuie sa detina email-ul si parola corespunzatoare si vor fi logati ca si admin. Ce pot dace ei in plus este sa acceseze pagina ADMIN unde vor putea sterge adauga sau actualiza datele tuturor jucatorilor care si au facut cont in aplicatie.

*Figura 12: Pagina EditPlayers*

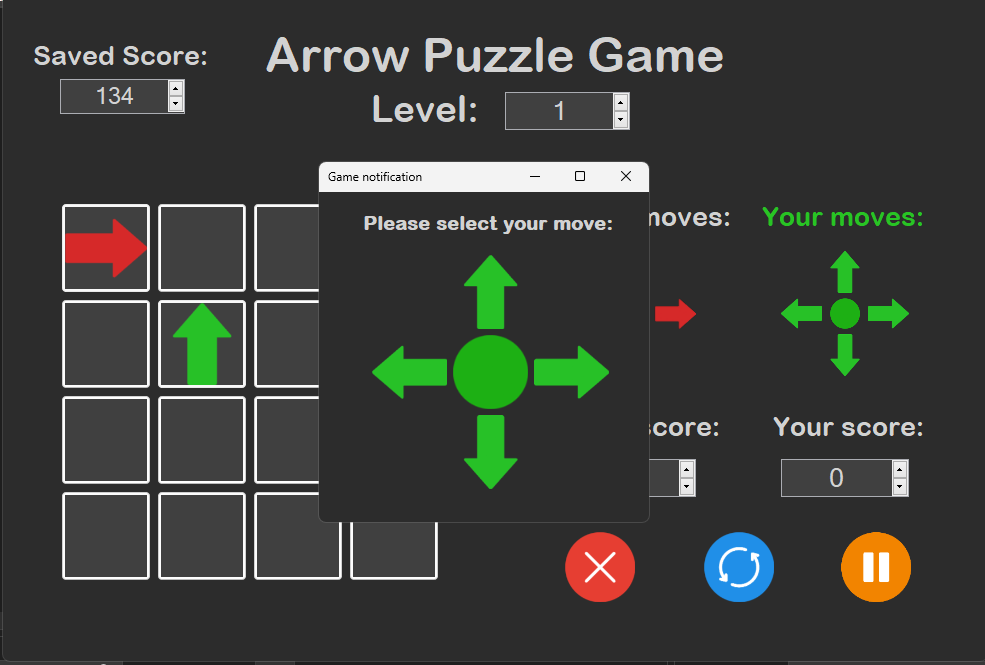
Din aceasta pagina se pot aduga noi jucatori completand campurile de sus si selectand “+” sau a sterge si actualiza jucatori deja existenti acest lucru se face prin selectarea randului corespunzator jucatorului dorit, astfel se vor completa toate campurile de sus si vom putea modifica ce camp dorim. Admin-ul poate desigur si sa caute jucatori dupa numele sau id ul acestora folosind campul Search din josul pagini.

Atat admin-ul cat si jucatorul normal pot acesa pagina de Play Game. Cand aceasta pagina este accesata se creeaza un grid corespunzator nivelului: pentru nivelul 1 este 4x4 iar pt nicelul 2 este 8x8, initial gol. Daca player-ul are cont isi poate vedea scorul salvat in baza de date in stanga sus si scorul din sesiunea curenta in dreapta jos. Jocul se desfasoara impotriva calculatorului si acesta are un scor in sesiunea curenta de joc. La fiecare meci castigat, castigatorul primeste un punct, iar la egalitate primesc ambele parti cate un punct.

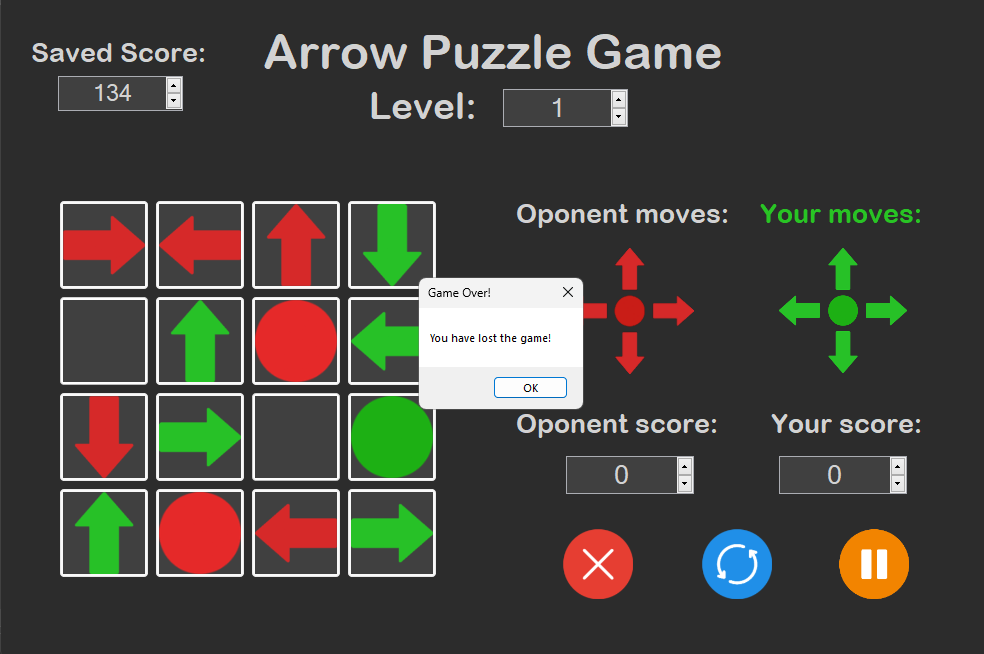
*Figura 13: Pagina Game cu nivel 1*

In partea din dreapta jucatorul poate vedea pt fiecare casuta ce miscari posibile are. Si acestea sunt in functie de nivel: nivelul 1 jucatorul are doar 5 miscari iar in nivelul 2 jucatorul are 9 miscari posibile. Mai gasim in partea de jos butoane precum play care incepe jocul, restart pt a reste sesiunea de joc curenta (se reseteaza si scorul), si parasirea sesiunei curente de joc, unde jucatorul va fi intrebat daca vrea sa salveze nou scor obtinut sau nu?

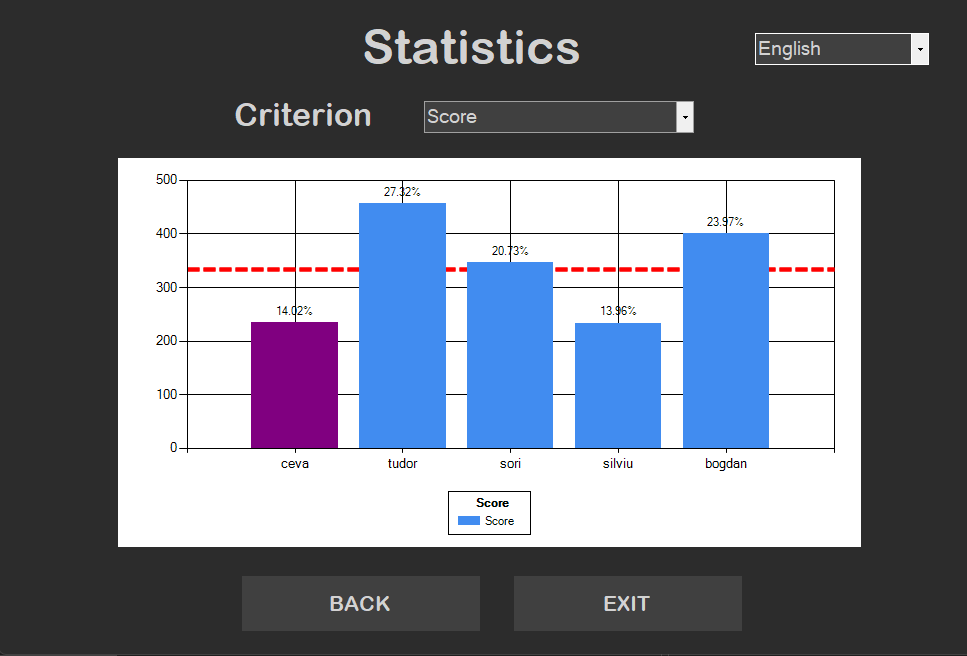
*Figura 14: Pagina Game cu nivel 2*

Dupa ce apasam Play, jocul incepe si se coloreaza cu o culaore specifica a cui e randul sa mute, sau sa faca o miscare. Initial incepe jucatorul iar apoi calculatorul, care implementeaza un algoritm de inteligenta artificiala numit minmax pt a determina cea mai buna miscare de la un moment respectiv, uitandu-se la mai multe configuratii. Odata ce ne vine randu apasam cu mouse-ul unul din pozitile din grid dupa care se va deschide o alta fereastra care ne va zice sa selectam sageata cu directia dorita. Odata o sageata cu o directie pusa aceasta nu se mai poate repeta nici pe aceasi linie nici pe aceasi coloana, iar in nicelul 2 nici pe aceasi diagonala.

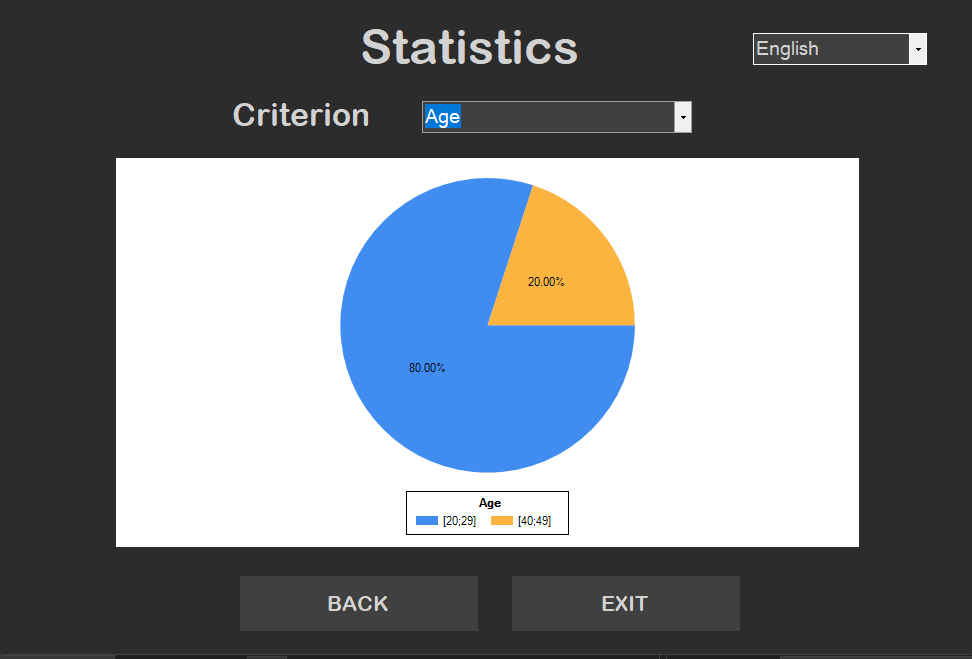
*Figura 15: Pagina Game, exemplu de joc*

Jocul se incheie atunci cand nu mai exista miscari posibile care pot fi puse. Pierde cel care trebuie sa mute insa nu mai are casute in care poate pune sageti. Dupa ce se termina o runda incepe automat alta.

*Figura 16: Pagina Game, exemplu final de joc*

Daca din pagina de Home se slecteaza optiunea de Statistics, atunci user-ul este intampinat cu o noua pagina numita Statistics, in care se pot vizualiza doua tipuri de grafice in functie de un criteriu care poate fi schimbat la orice moment: score si varsta.

*Figura 17: Pagina Statistics, garfic cu criteriul score*



*Figura 18: Pagina Statistics, garfic cu criteriul age*