

**Tema 1 -Arrow Puzzle Game**

Model – View – Presenter

Problema 35

**Proiectare Sofware**

**-Calculatoare si Tehnologia Informatiei-**

Student: Iamnitchi Bogdan

Grupa: 30235

An: 3

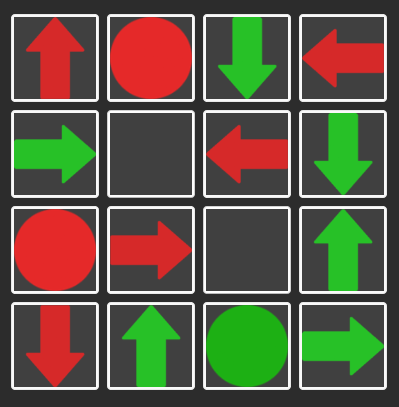
1. **Enuntul problemei**

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.

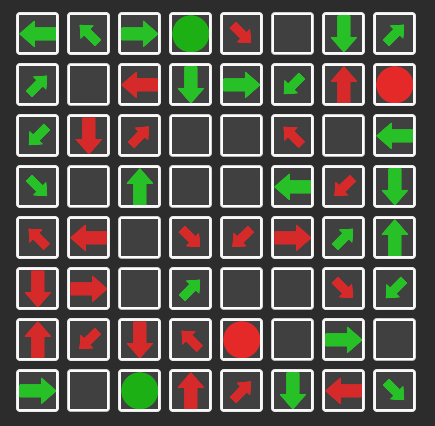
**Jucătorii** vor putea să se autentifice și să-și vizualizeze punctajul.

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.

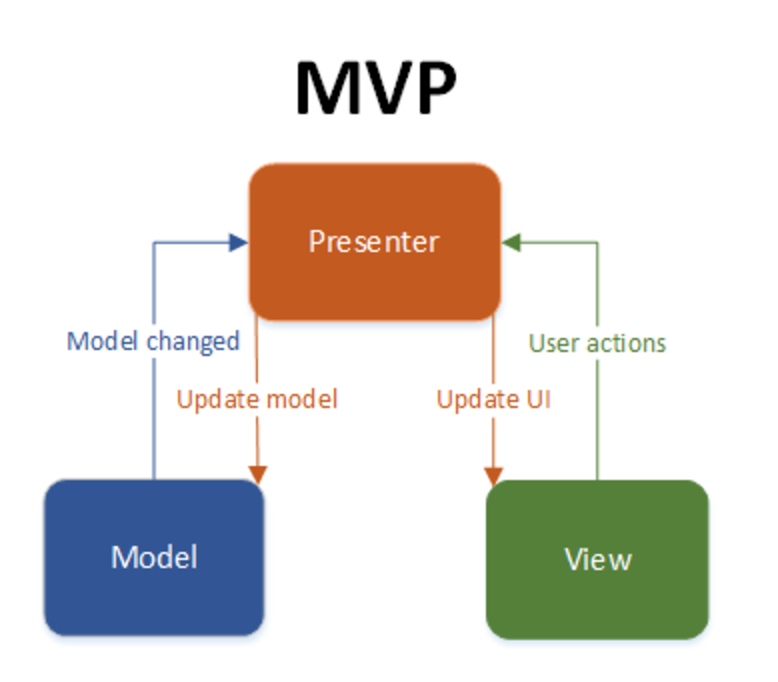
**Nivel 1:**

**Nivel 2:**



**Arhitectutura MVP**

**Model–view–presenter** ( **MVP** ) este o derivație a modelului [arhitectural model](https://en.wikipedia.org/wiki/Architectural_pattern)[–view–controller](https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller) (MVC) și este folosit mai ales pentru construirea de intefețe cu utilizatorul.

În MVP, *prezentatorul* își asumă funcționalitatea „omul de mijloc”. În MVP, toată logica de prezentare este împinsă către prezentator. [[1]](https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93presenter#cite_note-1)

Model (Modelul):

* Modelul reprezintă componenta responsabilă cu gestionarea datelor și a logicii de afaceri a aplicației. Aici sunt incluse operațiile de acces la date, validare și manipulare a acestora. Modelul nu are cunoștință despre View sau Presenter.

View (Vedere):

* Vederea reprezintă componenta responsabilă cu prezentarea informațiilor utilizatorului și interacțiunea cu acesta. Este o reprezentare a datelor din Model și se ocupă de afișarea lor într-o formă comprehensibilă utilizatorului. În MVP, View-ul este pasiv și nu conține logica de afaceri.

Presenter (Prezentator):

* Prezentatorul acționează ca un intermediar între Model și View. Este responsabil pentru actualizarea View-ului cu informațiile din Model și pentru gestionarea interacțiunii utilizatorului. Presenterul conține logica de afaceri și răspunde de acțiunile utilizatorului, transmitând apoi aceste informații către Model pentru procesare.

Prin separarea logică a acestor componente, MVP facilitează testarea unitară și întreținerea codului, deoarece fiecare componentă are rolul său bine definit și nu se amestecă cu responsabilitățile celorlalte. MVP încurajează dezvoltarea modulară și scalabilă a aplicațiilor și poate fi implementat în diferite tehnologii și platforme de dezvoltare.

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.

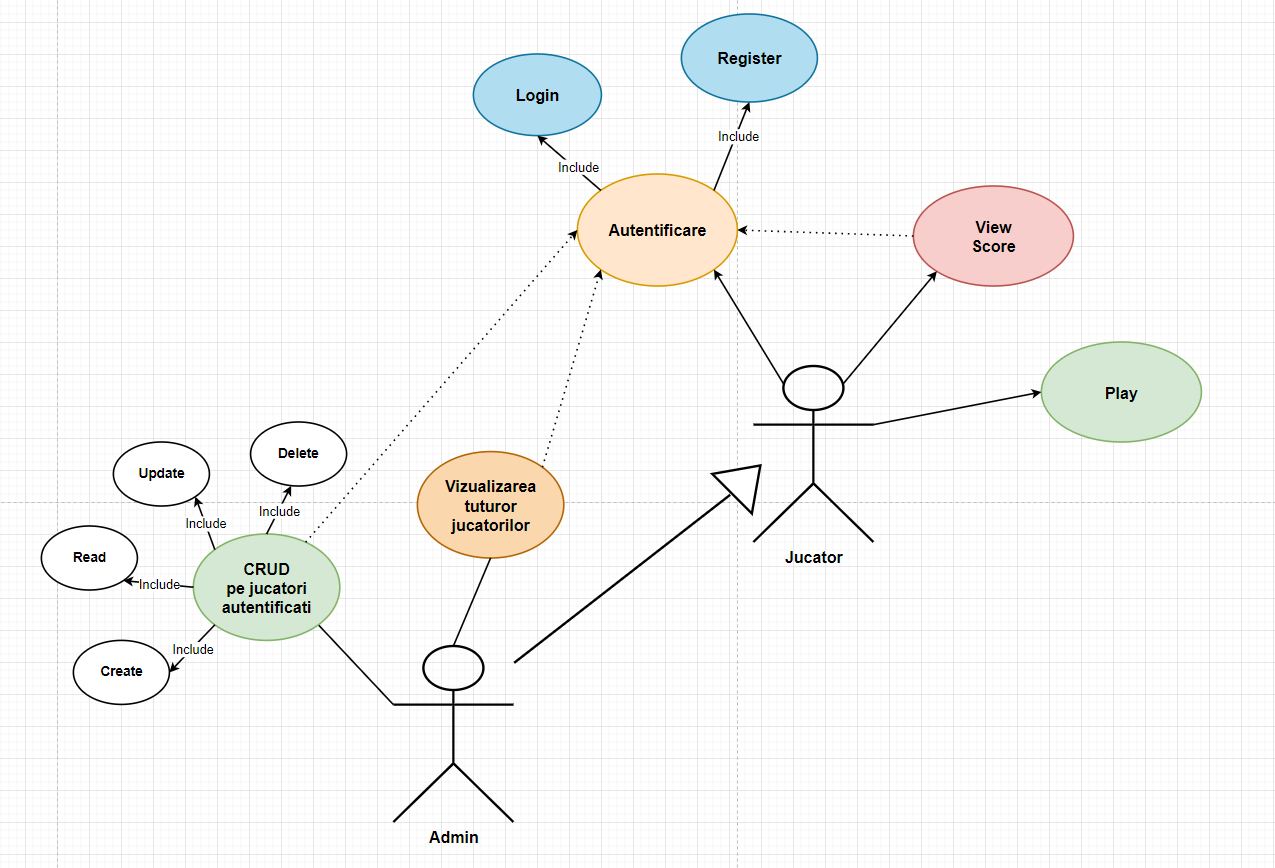
Pentru gestionarea codului sursă al aplicației și pentru a lucra eficient în echipă, am utilizat un sistem de control al versiunilor, cum ar fi Git. Acest lucru mi-a permis să monitorizez și să gestionez modificările în codul sursă al aplicației într-un mod organizat și colaborativ.

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.



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.

**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 Presenter. Fiecare pachet are mai multe clase implementate in aplicatie, acestea avand atributele si operatiile necesare functionarii aplicatiei.

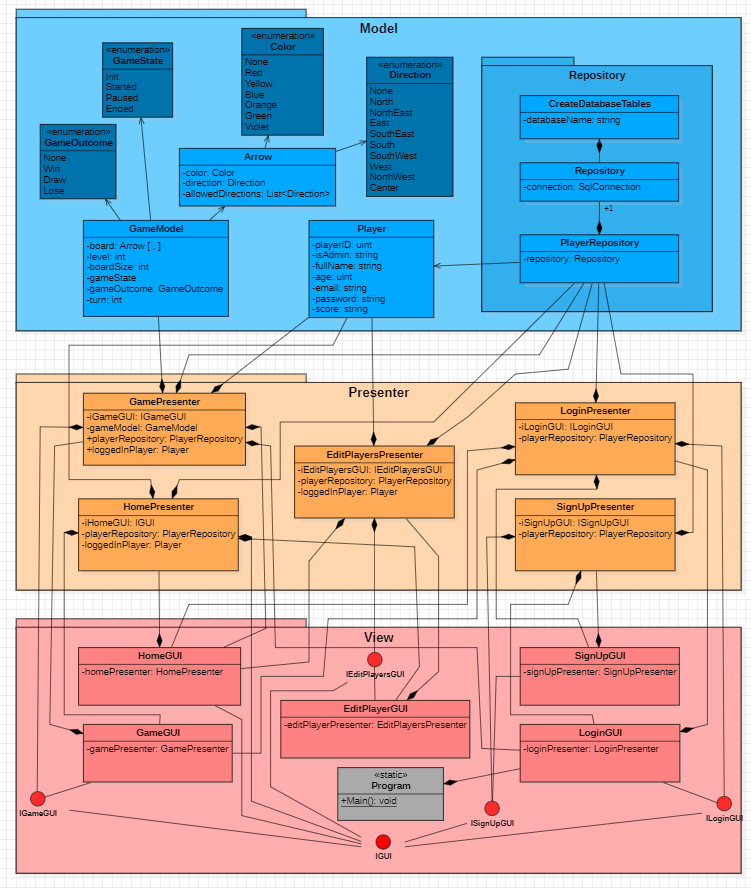


Diagrama de clase în cadrul arhitecturii MVP (Model-View-Presenter) este concepută pentru a ilustra modul în care diferitele componente ale modelului, vederii și presenterului sunt conectate și interacționează între ele. În cadrul MVP, fiecare componentă are roluri și responsabilități distincte. Iată o descriere generală a diagrama de clase în cadrul modelului MVP:

**Model (Modelul):**

* În diagrama de clase, clasa reprezentând Modelul conține logica de afaceri și datele aplicației.
* Aceasta include clase pentru gestionarea datelor jucătorilor, cum ar fi nume, scoruri etc.
* De asemenea, poate include metode pentru a efectua operațiuni CRUD (Create, Read, Update, Delete) pe datele jucătorilor.

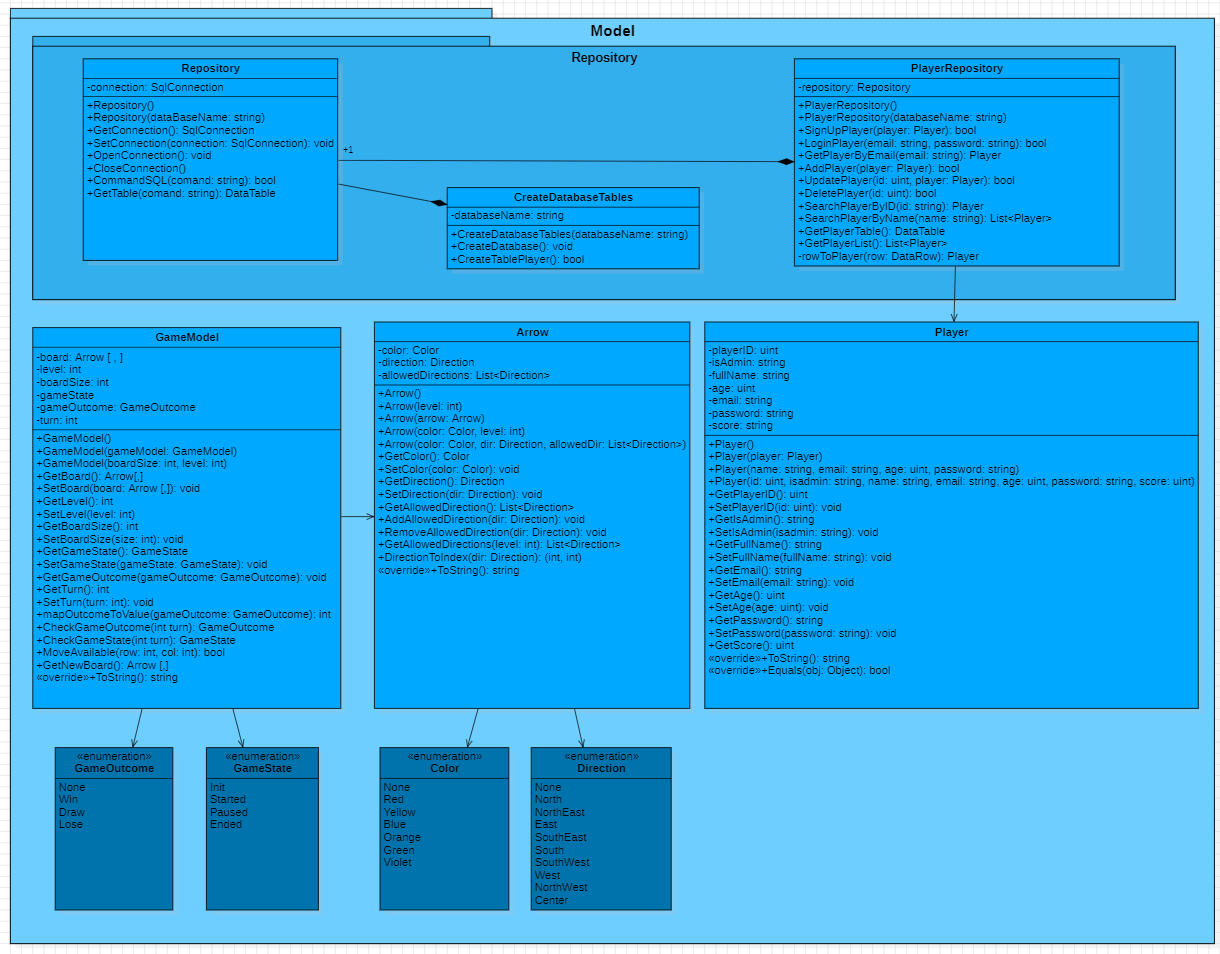
**View (Vedere):**

* Clasa View reprezintă interfața utilizatorului și este responsabilă pentru afișarea informațiilor utilizatorului și interacțiunea cu acesta.
* În diagrama de clase, aceasta poate conține elemente UI, cum ar fi butoane, etichete, casete de text etc.
* View-ul este pasiv și nu conține logica de afaceri. În schimb, trimite acțiunile utilizatorului către Presenter pentru procesare.

**Presenter (Prezentator):**

* Clasa Presenter acționează ca un intermediar între Model și View. Este responsabil pentru gestionarea logică a afacerilor și actualizarea View-ului cu informații din Model.
* În diagrama de clase, Presenterul poate conține referințe către Model și View, permițându-i să interacționeze cu acestea.
* De asemenea, poate conține metode pentru a gestiona acțiunile utilizatorului și a comunica cu Modelul pentru a efectua operațiuni CRUD sau alte operațiuni de procesare a datelor.
* În diagrama de clase MVP, conexiunile între aceste componente arată cum acestea interacționează între ele. De exemplu, View-ul poate să asculte evenimente de la interfața utilizatorului și să le trimită către Presenter pentru procesare. Presenterul, la rândul său, poate să actualizeze View-ul cu informațiile din Model și să primească notificări de la View pentru a răspunde la acțiunile utilizatorului.

**Pachetul Model:**



În cadrul pachetului model al arhitecturii MVP, ai un subpachet numit "repository" care se ocupă de interacțiunea cu baza de date. Iată o descriere mai detaliată a claselor din acest subpachet:

**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.

Ceea ce priveste modelul in sine avem uramtoarele:

**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:**

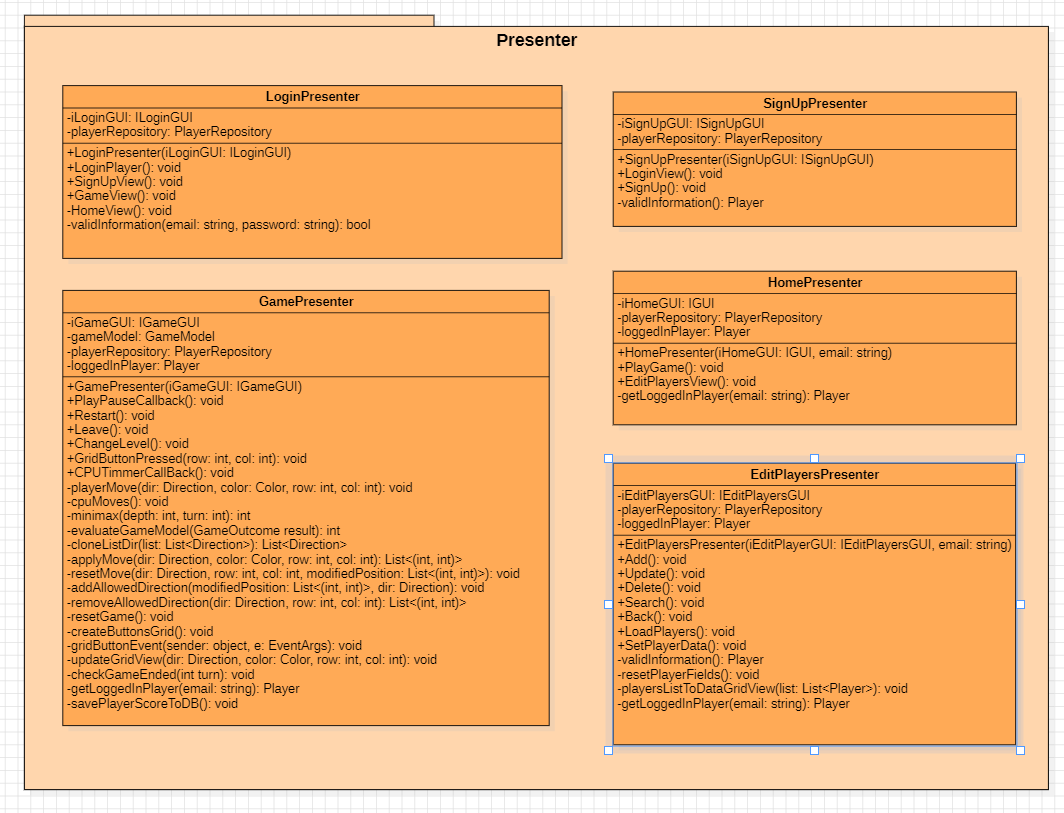
* Această clasă gestionează starea și logica generală a jocului. Ea se ocupă de lucruri precum tabla de joc, nivelul curent, starea jocului, rezultatul și al cui este rândul.
* GameModel se ocupă de inițializarea și gestionarea stării generale a jocului, inclusiv gestionarea tablei de joc și a evoluției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

Am mai introdus si cateva enumeratii in ceea ce priveste **Directia** , **Culoarea**, **Starea Jocului** (initiala, in desfasurare, terminata) si **Rezultatul Joucului** (castig, pierdere, sau egalitate).

**Pachetul Presenter:**



**EditPlayersPresenter:**

* Acest prezentator se ocupă de gestionarea interacțiunilor legate de editarea jucătorilor în joc.
* Atributul IEditPlayersGUI reprezintă interfața pentru interacțiunea cu GUI-ul de editare a jucătorilor.
* Prezentatorul interacționează cu View-ul pentru a permite administratorului să adauge, să editeze sau să șteargă jucători din sistem prin intermediul atributului de tip **PlayerRepository**.
* De asemenea, acesta utilizează atributul loggedinplayer pentru a ști cine este jucătorul autentificat în sistem.

**GamePresenter:**

* Acest prezentator se ocupă de gestionarea logicii jocului în sine.
* Atributul IGameGUI reprezintă interfața pentru interacțiunea cu GUI-ul jocului.
* Prezentatorul gestionează toate aspectele legate de desfășurarea jocului, inclusiv manipularea tablei de joc, actualizarea scorului și determinarea rezultatului jocului.
* De asemenea, utilizează atributul loggedinplayer pentru a ști cine este jucătorul care participă la joc.

**HomePresenter:**

* Acest prezentator se ocupă de gestionarea interacțiunilor legate de ecranul principal al aplicației, unde se prezintă regulile jocului și se oferă opțiuni pentru navigare.
* Atributul IHomeGUI reprezintă interfața pentru interacțiunea cu GUI-ul ecranului principal.
* Prezentatorul permite afișarea regulilor jocului și oferă opțiuni pentru a naviga către ecranele de editare a jucătorilor sau de joc.
* De asemenea, utilizează atributul loggedinplayer pentru a ști cine este jucătorul autentificat în sistem.

**LoginPresenter:**

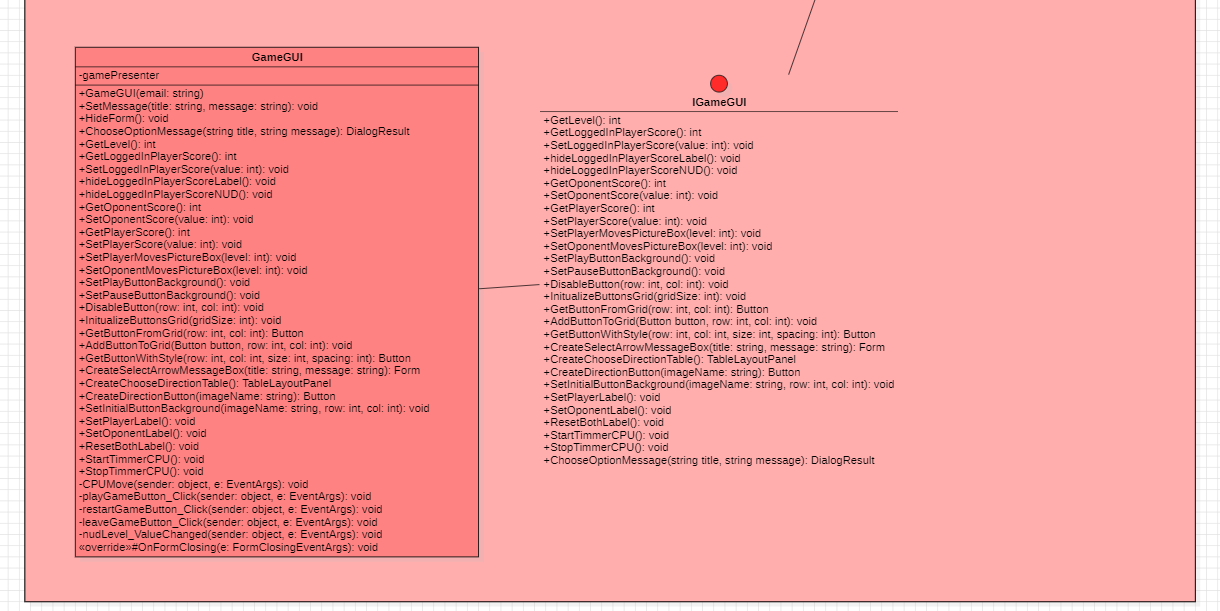
* Acest prezentator se ocupă de gestionarea procesului de autentificare a jucătorilor în aplicație.
* Atributul ILoginGUI reprezintă interfața pentru interacțiunea cu GUI-ul de autentificare.
* Prezentatorul gestionează validarea credențialelor introduse de utilizator și autentificarea acestuia în sistem.
* Poate actualiza loggedinplayer cu informațiile jucătorului autentificat.

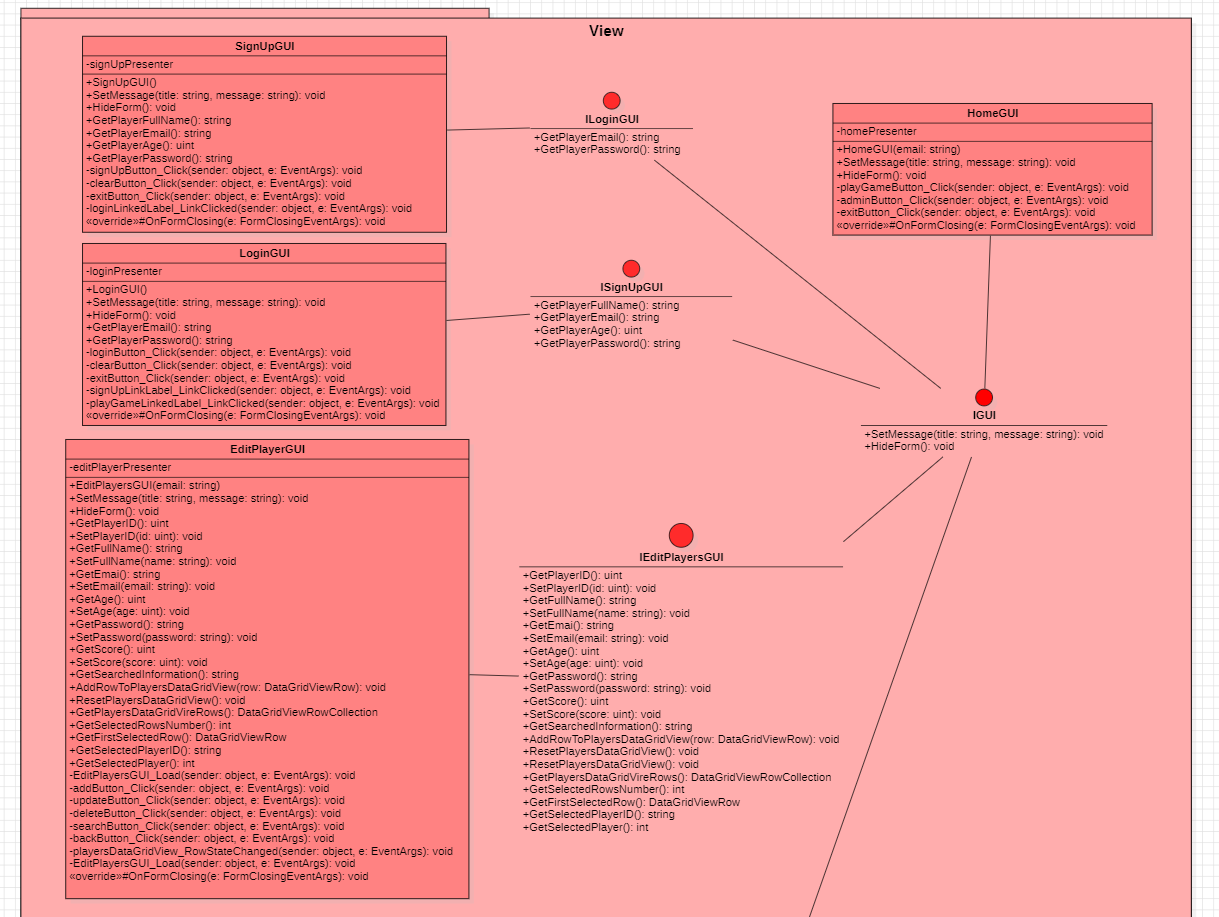
**SignUpPresenter:**

* Acest prezentator se ocupă de gestionarea procesului de înregistrare a noi jucători în aplicație.
* Atributul ISignUpGUI reprezintă interfața pentru interacțiunea cu GUI-ul de înregistrare.
* Prezentatorul gestionează procesul de colectare a informațiilor necesare pentru înregistrare și crearea unui nou jucător în sistem.

Acești prezentatori reprezintă punțile de legătură între logica aplicației și interfețele utilizatorului, gestionând interacțiunile și actualizările între acestea în conformitate cu regulile și funcționalitățile specificate. De asemenea, utilizează atributul loggedinplayer pentru a urmări și gestiona sesiunile jucătorilor care sunt autentificați în sistem.

**Pachetul View:**





**GameGUI:**

* Acest View este responsabil pentru afișarea și gestionarea interfeței grafice asociate jocului.
* Implementează interfața IGameGUI, care definește metodele necesare pentru afișarea și actualizarea elementelor grafice specifice jocului.
* Atributul GamePresenter reprezintă presenterul asociat cu acest View și este folosit pentru a delega logica și interacțiunile din spatele interfeței grafice către acesta.
* GameGUI oferă metode pentru adăugarea și modificarea componentelor GUI specifice jocului, cum ar fi tabla de joc sau elementele pentru afișarea scorului.

**HomeGUI:**

* Acest View prezintă ecranul principal al aplicației și este responsabil pentru afișarea regulilor jocului și a opțiunilor de navigare.
* Implementează interfața IHomeGUI, care definește metodele necesare pentru afișarea și actualizarea elementelor grafice specifice ecranului principal.
* Atributul HomePresenter reprezintă presenterul asociat cu acest View și este folosit pentru a gestiona logica și interacțiunile din spatele interfeței grafice.
* HomeGUI oferă metode pentru adăugarea și modificarea componentelor GUI specifice ecranului principal, precum și pentru gestionarea navigării între diferitele ecrane ale aplicației.

**EditPlayersGUI:**

* Acest View este responsabil pentru afișarea și gestionarea interfeței grafice asociate editării jucătorilor din joc.
* Implementează interfața IEditPlayersGUI, care definește metodele necesare pentru afișarea și actualizarea elementelor grafice specifice editării jucătorilor.
* Atributul EditPlayersPresenter reprezintă presenterul asociat cu acest View și este folosit pentru a gestiona logica și interacțiunile din spatele interfeței grafice.
* EditPlayersGUI oferă metode pentru adăugarea și modificarea componentelor GUI specifice editării jucătorilor, cum ar fi lista jucătorilor sau opțiunile de editare.

**LoginGUI:**

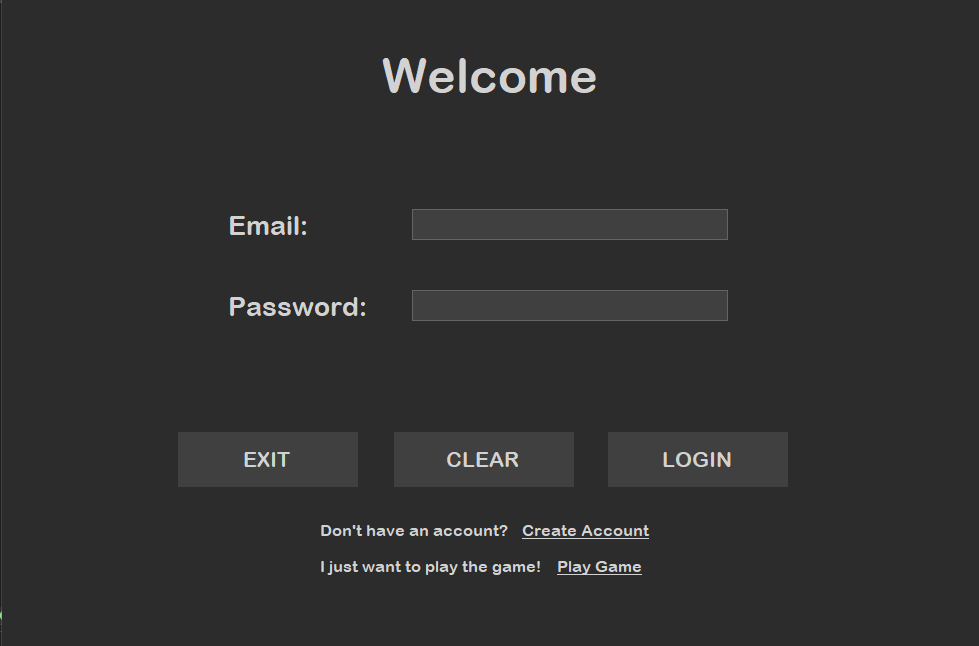
* Acest View este responsabil pentru afișarea și gestionarea interfeței grafice asociate procesului de autentificare a jucătorilor.
* Implementează interfața ILoginGUI, care definește metodele necesare pentru afișarea și actualizarea elementelor grafice specifice autentificării.
* Atributul LoginPresenter reprezintă presenterul asociat cu acest View și este folosit pentru a gestiona logica și interacțiunile din spatele interfeței grafice.
* LoginGUI oferă metode pentru adăugarea și modificarea componentelor GUI specifice procesului de autentificare, cum ar fi câmpurile pentru introducerea numelui de utilizator și a parolei.

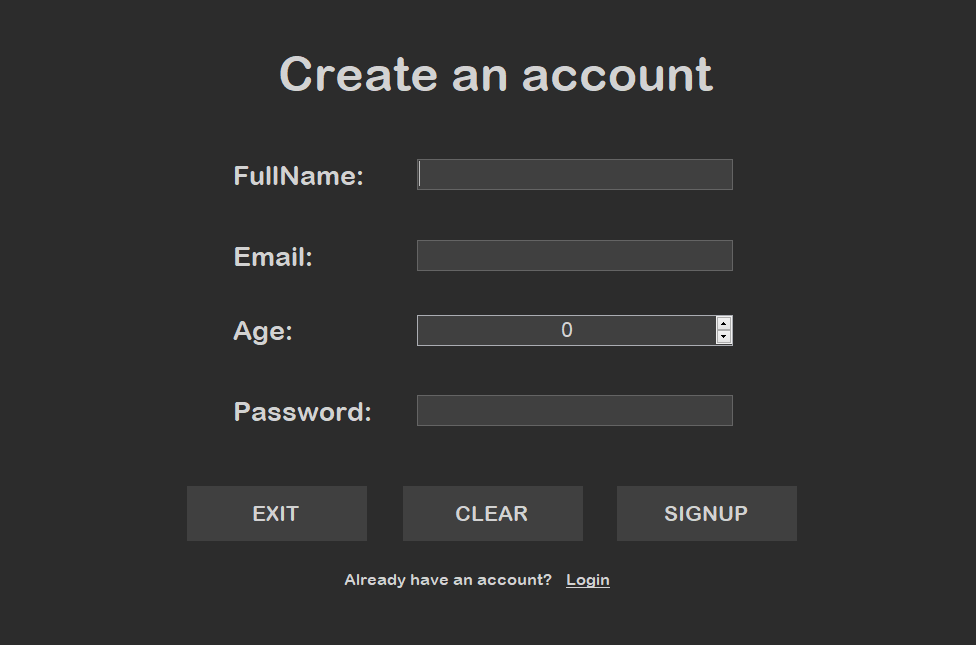
**SignUpGUI:**

* Acest View este responsabil pentru afișarea și gestionarea interfeței grafice asociate procesului de înregistrare a unui nou jucător în aplicație.
* Implementează interfața ISignUpGUI, care definește metodele necesare pentru afișarea și actualizarea elementelor grafice specifice înregistrării.
* Atributul SignUpPresenter reprezintă presenterul asociat cu acest View și este folosit pentru a gestiona logica și interacțiunile din spatele interfeței grafice.

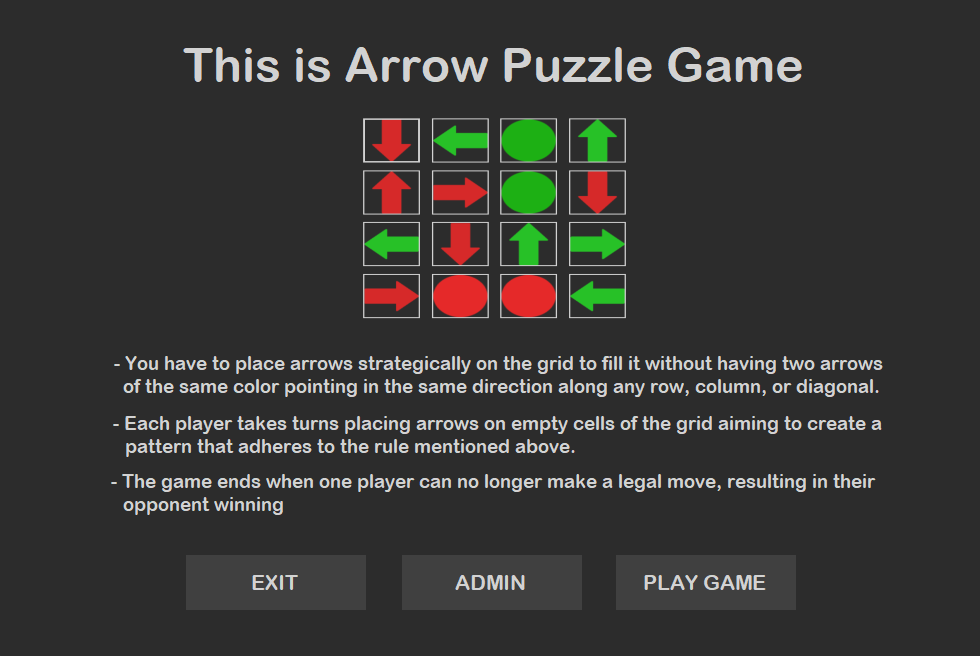
1. **Descrierea aplicatiei:**

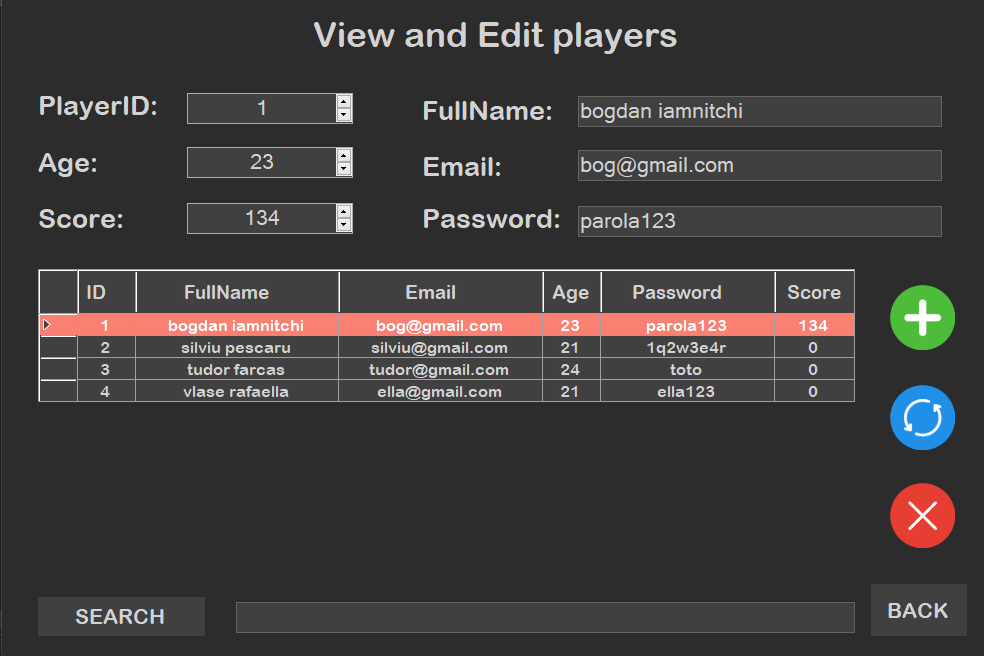
Cand pornim aplicatia clasa program va instantia clasa LoginGUI care ne va intampina cu o fereastra in care sa ne introducem credentialele pentru a ne putea autentifica. Daca nu avem cont putem apasa mai jos pe buttonu de create account care ne va duce pe pagina de SignUp pentru a ne creea un cont. Desigur 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.



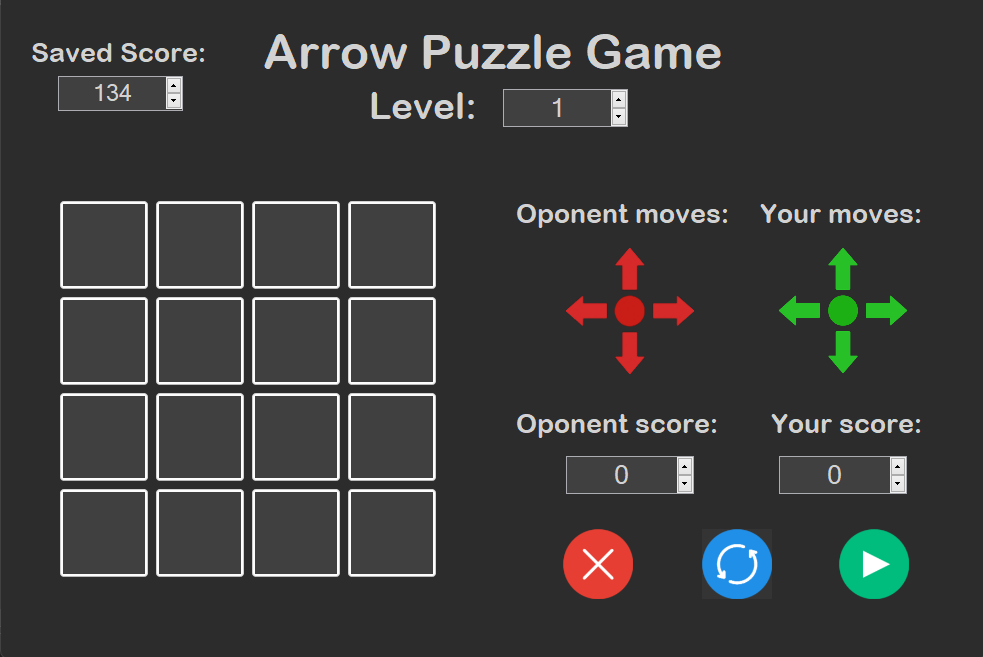
Daca algem optiuna de create account aceasta ne va duce spre pagina de SignUp in care o sa ni se ceara mai multe detalii care vor putea identifica unic jucatorul, de exemplu emai-ul. Si alte detalii precum varsta numele, scorul. Observam ca avem si optiunea de a naviga inapoi la pagina de Login daca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

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.

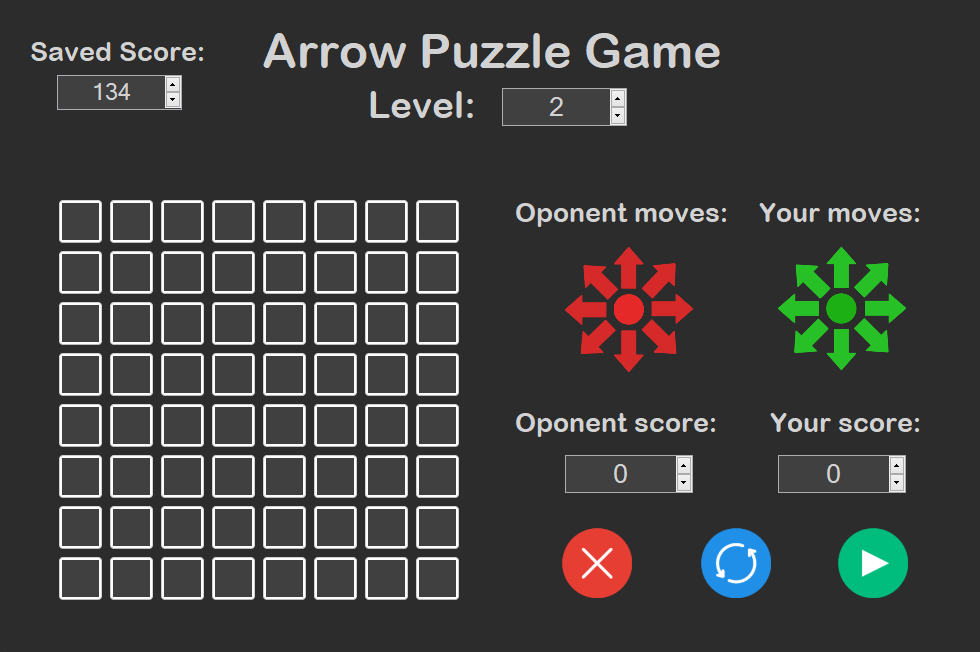
Daca acesti pasi au avut loc cu succes o sa ne apara un MessageBox care ne arata ca ne am autentificat si totul este in regula si vom ajunge astfel pe pagina de Home unde ne sunt explicate regulile jocului si de asemenea si o ilustratie. Desigur mai avem si aici butonul de exit care va inchide aplicatia daca asta e ceea ce ne dorim.

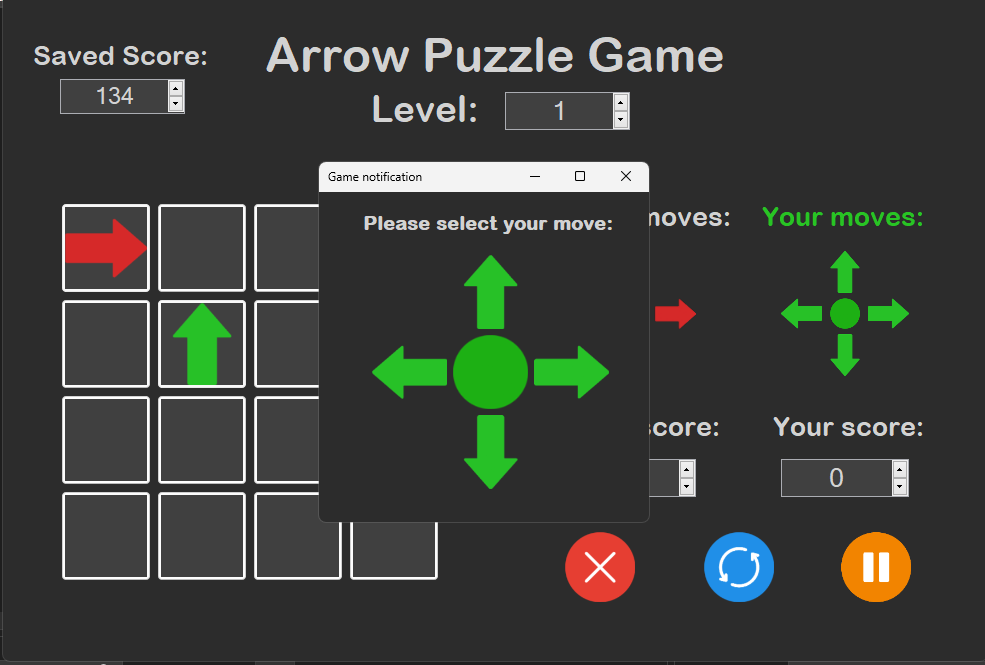
Exista doua tipuri de jucatori, jucatori simpli care isi pot face cont si se pot juca si cei de tip admin care sunt administratori, acestia au un cont special gata creat inainte ca aplicatia sa ruleze. Acestia trebuie sa detina email-ul si parola corespunzatoare si vor fi logati ca si admin. Ce pot dace 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.

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.

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?



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.

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.

