**Wordle Game in WPF**

**Documentatie**

Proiectul reprezintă o implementare a jocului Wordle în WPF, adoptând arhitectura Model-View-ViewModel (MVVM).

**Despre joc**

Proiectul propune o experiență interactivă în cadrul jocului Wordle, unde utilizatorul are misiunea de a ghici un cuvânt în decursul a 6 încercări. La fiecare introducere a unui cuvânt, acesta primește indicii esențiale despre cuvântul secret. Aceste indicii sunt furnizate sub forma culorilor verde, portocaliu și gri, indicând:

Verde: Literele care sunt corect plasate în cuvântul secret.

Portocaliu: Literele care sunt prezente în cuvântul secret, dar nu la poziția corectă.

Gri: Literele care nu fac parte din cuvântul secret.

**Despre implementare**

Primul ecran care apare în aplicație este fereastra de autentificare (login), unde utilizatorii au opțiunea de a se conecta la contul lor sau de a-și crea un cont nou. Acesta servește ca punct de intrare în experiența de joc. Utilizatorul are posibilitatea de a furniza un nume de utilizator și o parolă pentru a accesa sau crea un cont.

A screenshot of a computer

Description automatically generated

A computer screen shot of text

Description automatically generated

Logica pentru fereastra de autentificare (login) este gestionată de către LoginViewModel. Acesta este responsabil pentru interacțiunea cu serviciul de autentificare, care la rândul său se conectează la baza de date Microsoft SQL Server Management Studio. Atunci când un utilizator introduce un nume de utilizator și o parolă, LoginViewModel utilizează serviciul de autentificare pentru a valida aceste informații în baza de date.

După autentificare cu succes, utilizatorul este redirecționat către fereastra de meniu. În cadrul acestei ferestre, utilizatorul are opțiunea de a începe un joc nou sau de a relua ultimul joc salvat. Dacă utilizatorul alege să reia ultimul joc, MenuViewModel interacționează cu serviciul corespunzător pentru a obține progresul ultimului joc salvat.

Atunci când se reluează ultimul joc, este deschisă fereastra de joc cu progresul precompletat, oferind utilizatorului posibilitatea de a continua acolo unde a rămas înainte.

A screenshot of a computer

Description automatically generated

A screen shot of a computer program

Description automatically generated

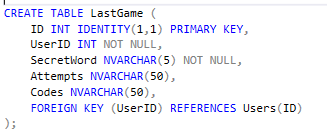
Structura unui joc este după cum urmează:

A screen shot of a computer

Description automatically generated

UserId-ul pentru a identifica jucătorul, SecretWord - cuvântul de ghicit, și Attempts - array-ul de cuvinte încercate, precum și codurile aferente fiecărui cuvânt.

În baza de date arată așa:



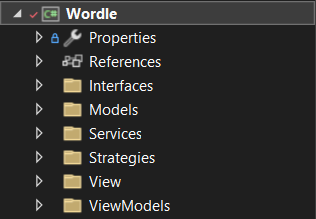
Exemplu pentru desfășurarea unui nou joc:A screenshot of a computer

Description automatically generated

Metoda cea mai importantă este cea de formare a codului unui cuvânt, fiecărui nou cuvânt introdus fiindu-i corespunzător un cod.A screen shot of a computer program

Description automatically generated

Arhitectura proiectului este MVVM, astfel încât separarea dintre logica și UI este asigurată.



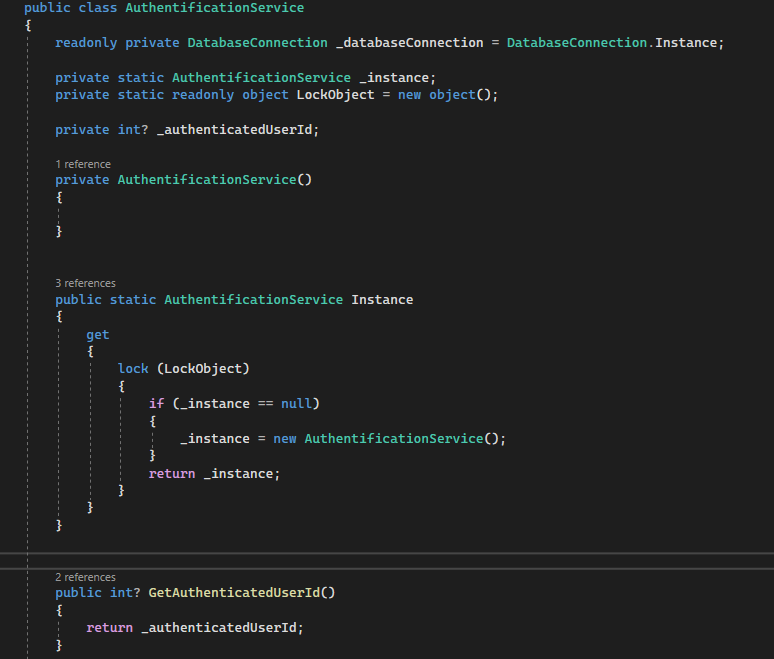
**Design Patterns**

Design pattern-urile utilizate sunt:

**1.Singleton**

Am ales sa folosim Singleton pentru api-ul pentru cuvinte. Autentificare si conexiunea cu baza de date pentru a asigură că există o singură instanță a unei clase în întregul proces.

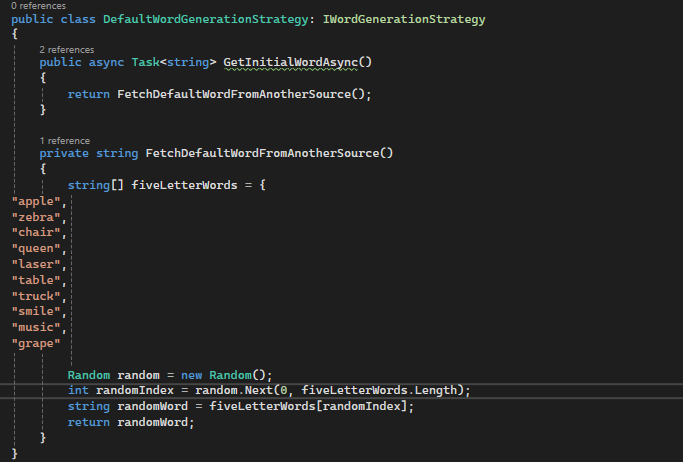
În cazul autentificării, păstrarea stării autentificării într-o instanță singleton a falicitat accesul la informații despre utilizator în întreaga aplicație. Singleton poate centraliza și gestiona aspecte cheie ale aplicației. În cazul conexiunii la baza de date, o instanță singleton poate gestiona eficient deschiderea și închiderea conexiunilor, evitând astfel crearea unor conexiuni ineficiente sau unmanaged. Singleton oferă un mecanism de acces global la instanța sa, ceea ce poate fi util în cazul serviciilor care trebuie să fie accesate din mai multe locuri ale aplicației, precum API-ul pentru cuvinte.

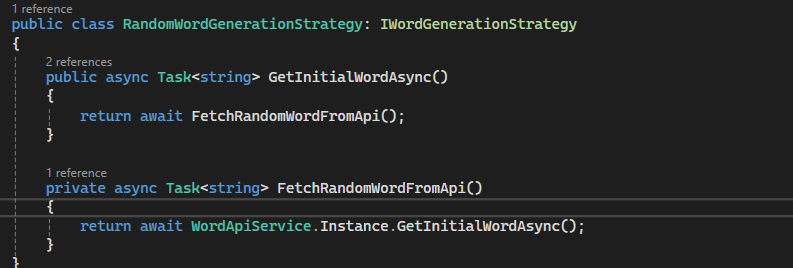


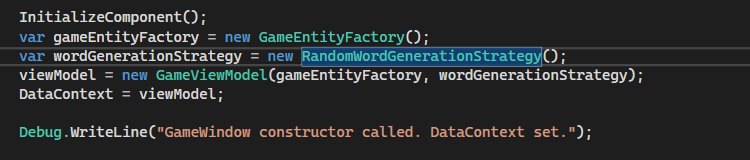
În acest fragment de cod, Singleton este utilizat în cadrul serviciului de autentificare. Se poate observa cinstructorul privat ce permite o singura instanta a aplicatiei, metoda de returnare a instantei. Blocarea (lock) este necesară pentru a preveni crearea simultană a mai multor instanțe Singleton în medii concurente. Informatia de autentificare stocata pentru accesul in toata aplicatia este \_authenticatedUserId.

**Strategy**

Am folosit design pattern-ul Strategy pentru alegerea metodei de obținere a cuvântului de ghicit. Există doua posibili tați prin care putem obține un cuvânt, prin extragerea acestuia dintr-un API sau prin selectarea aleatorie dintr-un șir de cuvinte. Datorită arhitecturii acestui design, putem implementa ulterior alte strategii de alegere a unui cuvânt.

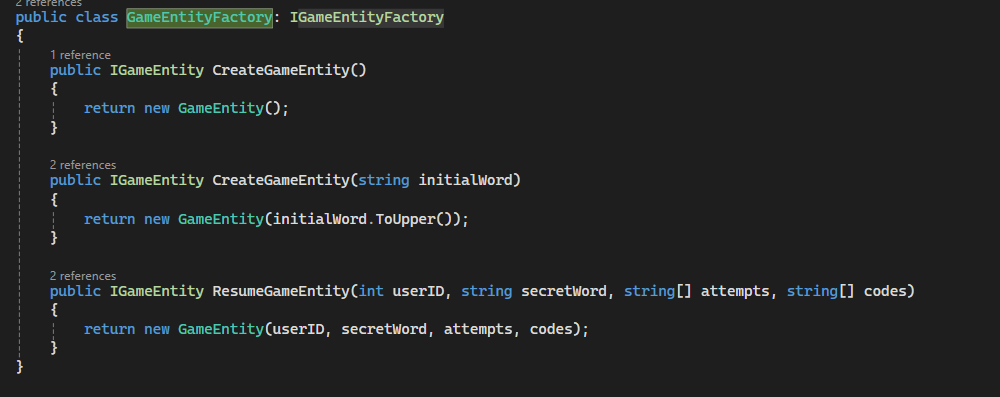




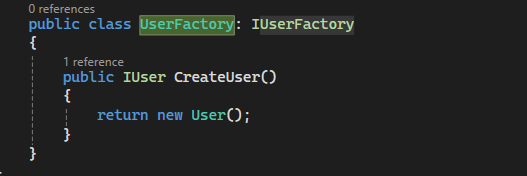


**Factory**

Metoda Factory este folosită în limbajul C# pentru nivelul mare de flexibilitate oferit. Aceste procedeu presupune definirea unor metode care returnează instanțierea unui obiect. Avantajul este că subclasele pot suprascrie această metoda. În proiectul nostru am folosit Factory pentru crearea claselor GameEntityFactory -



și UserFactory -



**Observer**

In GameViewModel INotifyPropertyChanged este implementat pentru a crea un model de date care notifică vizualizarea (interfața de utilizator) despre schimbările în proprietățile sale. Acest lucru se potrivește cu pattern-ul de design Observer, unde un obiect (observabil) menține o listă de obiecte dependente (observatori) care trebuie notificate automat la orice schimbare de stare, astfel încât acestea să se actualizeze.

A computer screen shot of a program code

Description automatically generated

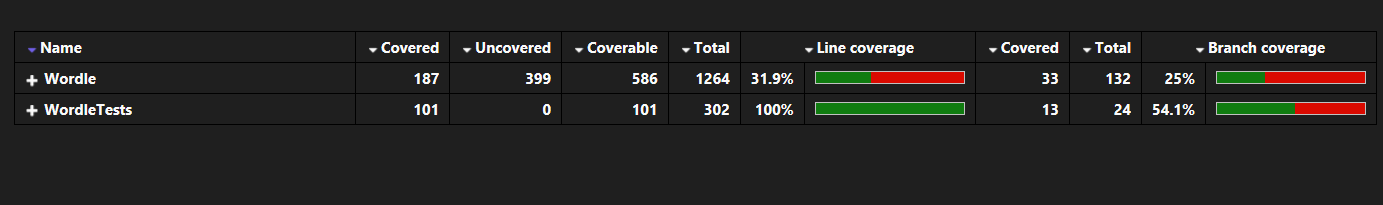
Interfața INotifyPropertyChanged declară evenimentul OnPropertyChanged, care trebuie declanșat atunci când o proprietate a clasei este modificată.

A screenshot of a computer program

Description automatically generated

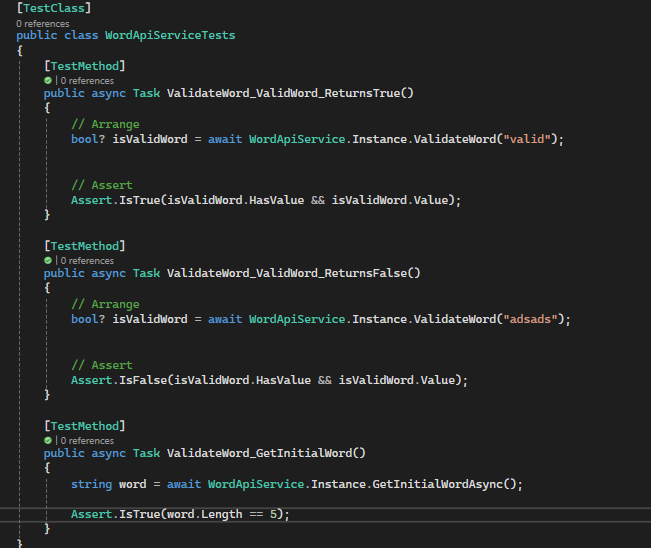
Cand o proprietate se modifica, aceste modificari sunt reflectate in UI.

**UNIT TESTING**



Proiectul conține teste per unitate după cum se poate vedea în poza de mai sus. Deoarece, aplicația este construită în .Net Framework, versiunea 4.8, am folosit framework-ul MSUnit.

Câteva elemente de bază din MSUnit sunt atributele, care definesc și localizează testele, precum [TestClass] și [TestMethod] care adnotează un test dintr-o suită de teste.



**Diagrame**

**1.Use-caseA diagram of a person with text

Description automatically generated**

Diagrama include diverse functionalitati ale proiectului cu care interactioneaza actorul principal, Player-ul.

**2.Class Diagram**

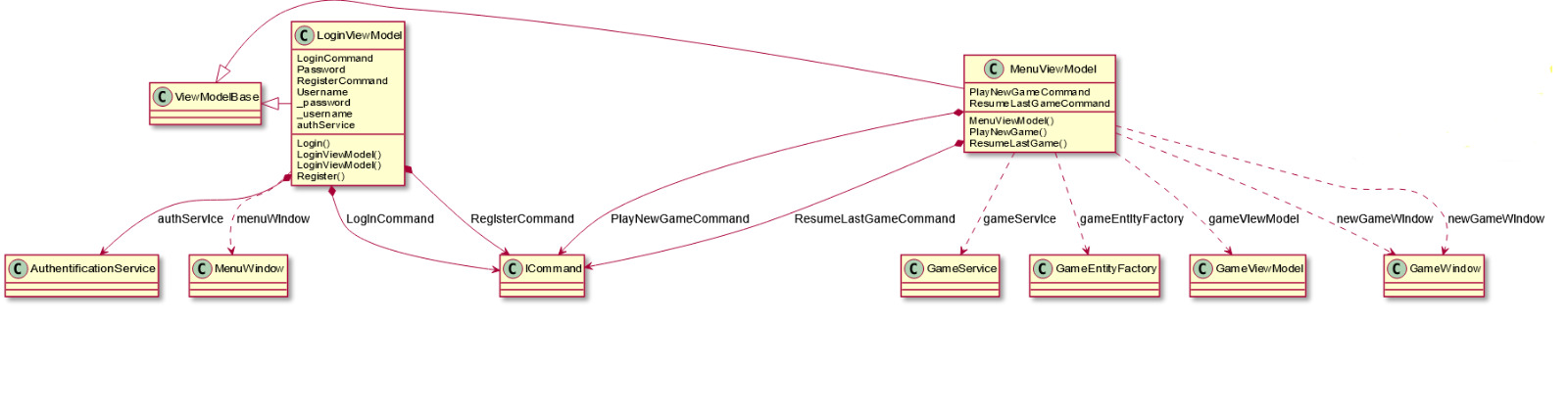
****

Diagrama descrie clasele si metodele principale.

**3.State Chart Diagram**

**A screenshot of a computer screen

Description automatically generated**

**4.Activity Diagram**

**A screenshot of a computer

Description automatically generated**

**5.SequenceDiagram**

**A diagram of a computer program

Description automatically generated**

**6.Collaboration Diagram**

**A diagram of a server

Description automatically generated**

**Membrii proiect**

**Michea Adriana-Georgiana 10LF212**

**Smeu Rares 10LF213**