UNIVERSITATEA TEHNICĂ „GHEORGHE ASACHI" IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
INGINERIA PROGRAMĂRII

DOCUMENTAȚIE

TITLUL LUCRĂRII: Aplicație de cultură generală Vrei-să-fii-inginer

AUTORI:
Butnaru Silviu,
Paraschiv Florin-Vlăduț,
Iluca Alexandru-Marian,
Buliga Diana-Marinela
GRUPA: 1306B

CUPRINS

Documentul specificațiilor cerințelor (SRS)

1. **Introducere**
2. **Descriere generală**
3. **Cerințe specifice**

Diagrame UML

1. **Diagrama de clase**
2. **Diagrama cazurilor de utilizare**
3. **Diagrama de activitate**

Modul de utilizare al programului

1. **Introducere**
2. **Interfața de start**
3. **Interfața jocului**
4. **Interfața record**
5. **Interfața de final**

Singleton – SQLCommand

Unit testing

Documentul specificațiilor cerințelor (SRS)

1.Introducere

1.1. Scop

Scopul aplicației este de a verifica cunoștințele unui utilizator în domeniul tehnic. Obiectivul este de a acumula noi cunoștințe, dar și de a proba cele deja existente.

1.2. Obiective

Vrei-să-fii-inginer este o aplicație care se derulează pe mai multe arii de cunoștințe din domeniul tehnic. Aceasta furnizează o serie de întrebări cu răspunsuri generate aleatoriu. Astfel, se vor stoca atât utilizatorii cât și scorurile acumulate, în urma folosirii aplicației, într-o bază de date locală asociată.

1.3. Prezentarea generală a sistemului

Pentru a implementa aplicația, s-au folosit Visual Studio proiect de tip Form Application, baze de date Oracle.

2. Descriere generală

2.1. Perspectiva produsului

Vrei-să-fii-inginer este o aplicație independentă, care permite logarea ca utilizator și de a se deplasa prin interfață pentru a răspunde la întrebări, a afla clasamentul sau de a afla mai multe despre funcționalitatea programului. Pentru a se implementa produsul final, s-a folosit limbajul C# și se poate rula pe sistemul de operare Windows, în formatul Windows Form Application. Se poate utiliza de orice persoană doritoare de a-și supune la testare cunoștințele în inginerie.

2.2. Funcțiile produsului

Programul livrează utilizatorilor următoarele funcții:

- Conturi personale: aplicația permite crearea de noi conturi, la finalul jocului, a unui nou utilizator sau deja existent, căruia i se atribuie scorul obținut
- Intrarea pe interfața de chestionar: se permite vizualizarea în format de test, în care apare întrebarea și cu 4 variante de răspuns
- Vizionarea scorului acumulat: se admite vizualizarea scorului la finalul turului de răspuns

2.3. Clase utilizator și caracteristici

Utilizatorii ar trebui să poată răspunde la întrebări într-un anumit interval de timp, scorul salvându-se în baza de date.

Sistemul suportă doar un tip de privilegii de utilizator: user, nefiind implementat un sistem de autentificare. Acesta va avea acces la toate funcționalitățile programului, însă nu va putea introduce modificări în baza de date în ceea ce privește întrebările și răspunsurile.

Procesul de implementare este unul relativ simplu, având în vedere scopul programului. Astfel, nu s-a supraîncărcat procesul, dar s-a ținut cont de obiectivele aferente aplicației.

2.4. Design și constrângeri de implementare

Constrângerile întâmpinate au fost de natură organizațională, de tipul timpului acordat în crearea aplicației, repartizarea task-urilor în echipă, stabilirea temei, cât și design-ul acesteia.

La nivel de implementare, constrângerile au fost date de Visual Studio în privința designului ales. Mai exact, fontul folosit în afișarea textului atât întrebărilor, cât și răspunsurilor a avut nevoie de schimbări, astfel încât acesta să fie același pe orice dispozitiv rulează executabilul pus la dispoziție pentru utilizatori. Designul ales este unul futuristic, care nu irită la nivel vizual userul.

3. Cerințe specifice interfață

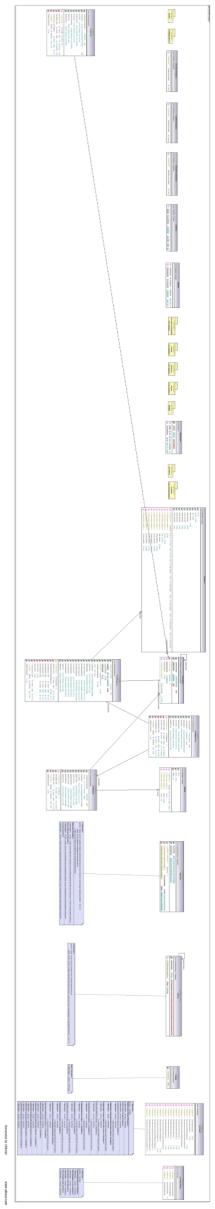
3.1. Interfața Grafică cu Utilizatorul

La prima deschidere, utilizatorul ar trebui să vadă interfața de start, în cadrul în care poate alege dacă să răspundă la chestionar, punctajele acumulate sau să iasă din interfață. Astfel, dacă alege să vadă punctajele acumulate, se va deschide o nouă pagină cu clasamentula actual. Dacă utilizatorul alege play, atunci va fi redirecționat la o nouă pagină în care va apărea aleatoriu câte o întrebare și variantele ei de răspuns.

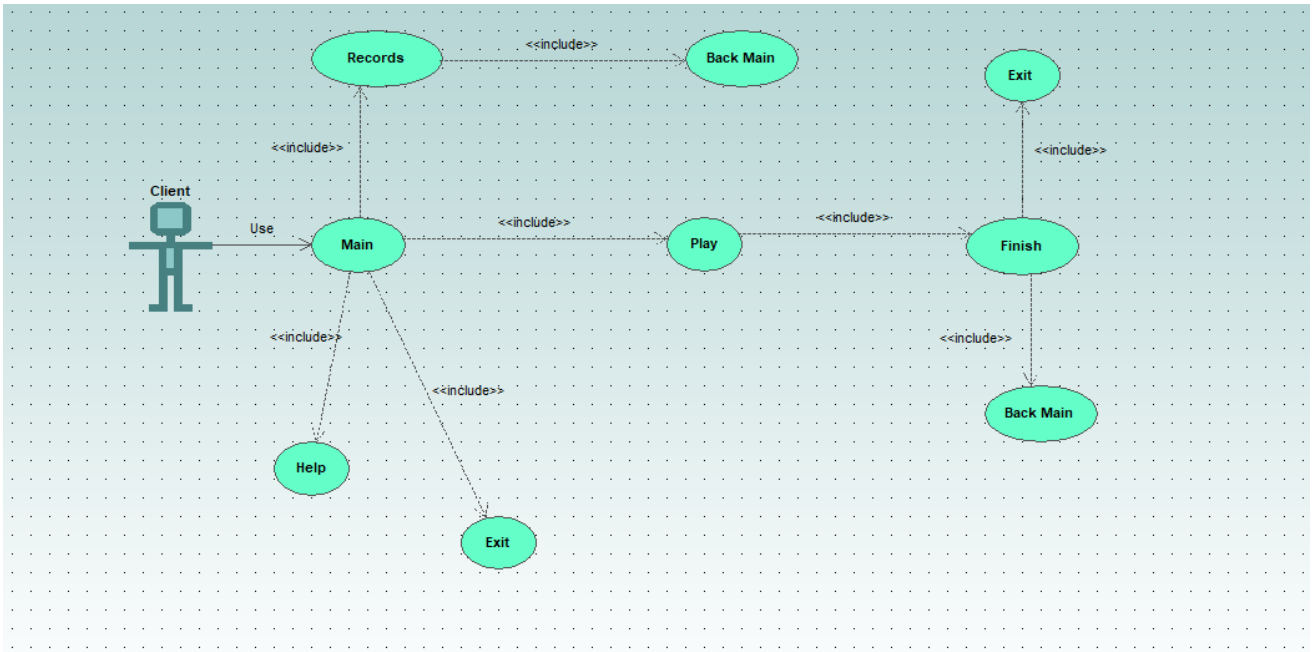
În cadrul interfeței de chestionar, utilizarea acesteia este destul de intuitivă, astfel că odată ales răspunsul în timpul dedicat întrebării, va trece automat la următoarea întrebare, fiind asemenea chestionarelor deja implementate pe piață.

Diagrame UML

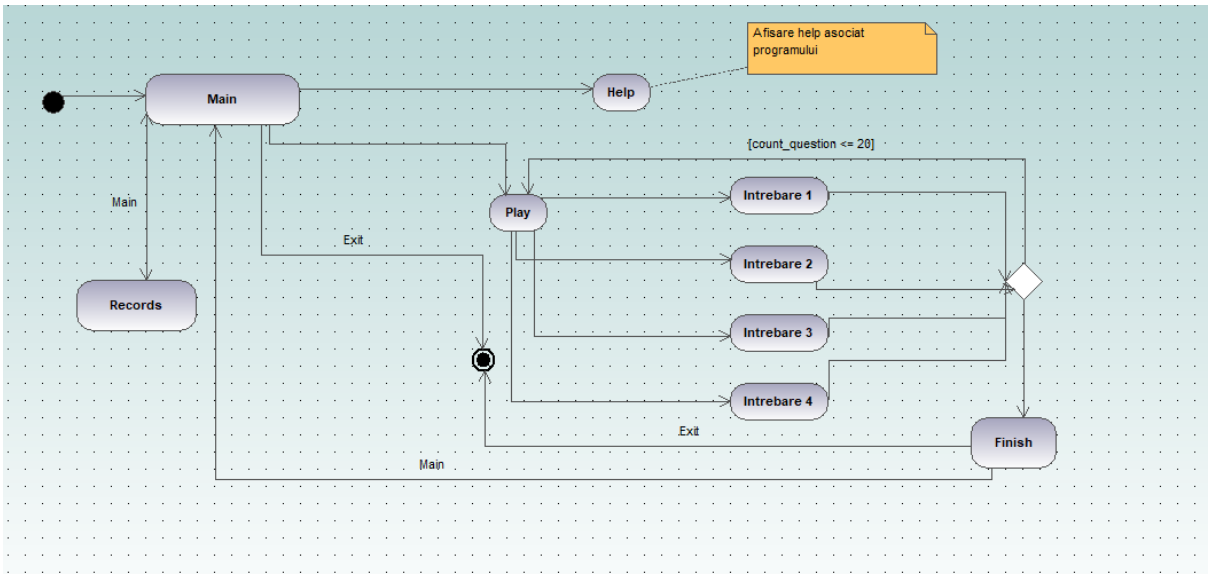
1. Diagrama de clase



2. Diagrama cazurilor de utilizare



3. Diagrama de activitate



Modul de utilizare al programului

1. **Introducere**
   Vrei-să-fii-inginer este o aplicație de tip quiz, care salvează într-o bază de date utilizatori, alături de scoruil acestora, și, tot din baza de date, afișează întrebările cu răspunsurile aferente.
   Facilitatea aplicației oferă oportunitatea de utilizare a tuturor doritorilor.

2. **Interfața de start**
   Oferă utilizatorului oportunitatea de a rula chestionarul, de a vedea clasamentul actual, dar și de a afla mai multe informații privind funcționalitatea și autorii programului implementat.

3. Interfața jocului

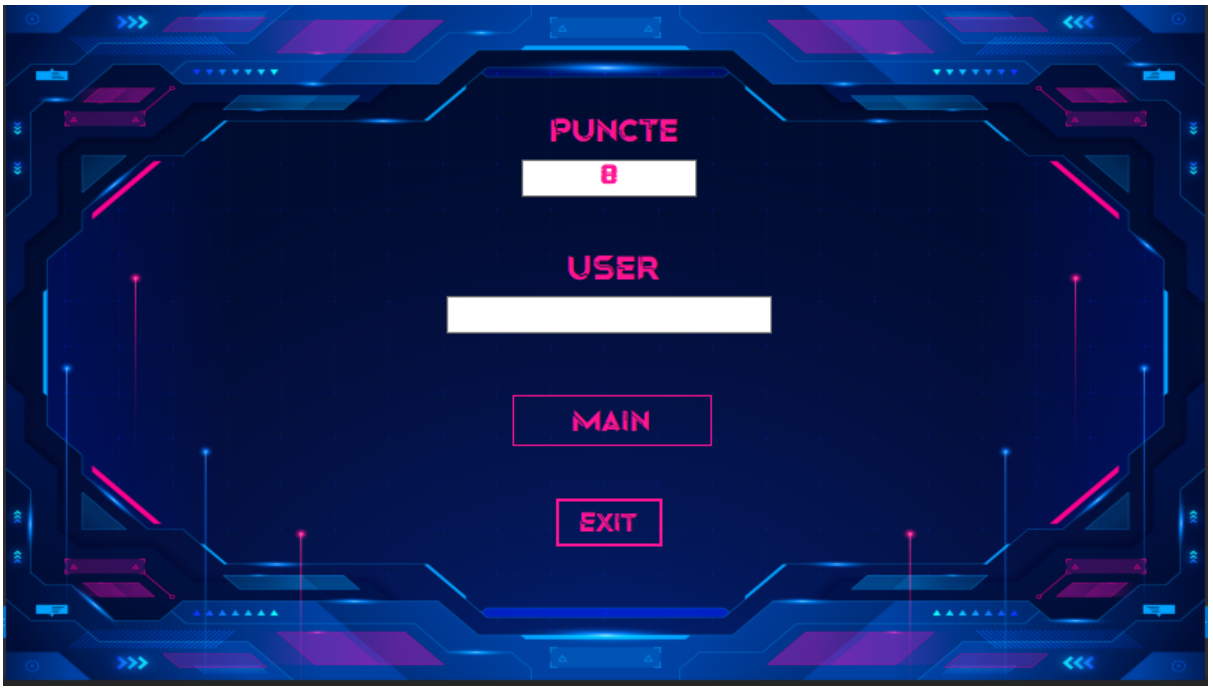
Pune la dispoziție un layout, atât cu întrebarea curentă, cât și cu răspunsurile aferente. Pentru a reuși să se încadreze în timp, s-a implementat un timer de 30 de secunde pentru fiecare întrebare, dar și numărul întrebării aferente.

4. Interfața record

În această pagină, se pot observa punctajele cele mai mari, rezultate în urma rulării și răspunderii la întrebările propuse.

5. Interfața de final

Această pagină apare în urma încheierii chestionarului. Aici se poate observa punctajul acumulat. Tot aici se poate înregistra utilizatorul, având la dispoziție oportunitatea de a întoarce și a vedea locul în clasament sau reîncepere a jocului sau de a părăsi programul.



Singleton – SQLCommand

```
namespace DataBase.ConnectionToOracleDB

{

        public class Connection

        {

                #region Fields

                private static OracleConnection _connectionString = null;

                private static Connection _singleInstance = null;

                #endregion
```

```
#region Private Constructor

/// <summary>

/// Initialize connection string for connection

/// </summary>

private Connection()

{

        _connectionString = new OracleConnection("Data
Source=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=152
1)))(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=orcl)));User
Id=C##ip_admin;Password=proiect;");

}

#endregion


#region Public Static Method

/// <summary>

/// Open the connection to the Oracle DB with connection string

/// Singleton Design Pattern used

/// </summary>

/// <returns> An singleton object used for create, close and get DB </returns>

public static Connection createConnection()

{

try

{

    if (_singleInstance == null)

    {

        _singleInstance = new Connection();

         _connectionString.Open();

        Console.WriteLine("Connection to oracle db succesfully");

    }
```

```
                return _singleInstance;
            }
            catch
            {
                throw new ConnectionException("PROBLEM WITH CONNECTION WITH
ORACLE DATABASE");
            }
            }
            #endregion

            #region Public Method
            /// <summary>
            /// Close connection with Oracle db
            /// </summary>
            public void closeConnection()
            {
            try
            {
            if (_singleInstance != null)
            {
                _connectionString.Close();
                Console.WriteLine("Closed connection to oracle db succesfully");
                _singleInstance = null;
            }
            }
            catch
            {
                 throw new ConnectionException("PROBLEM WITH CLOSING ORACLE
DATABASE");
```

```
        }
        }
        #endregion

        #region Getter
        /// <summary>
        /// Return connection string for Oracle db
        /// </summary>
        public OracleConnection ConnectionString
        {
        get
        {
                return _connectionString;
        }
        }
        #endregion
        }
        }
```

Unit testing

```
[TestClass]
        public class UnitTestDB
        {
        [TestMethod]
        public void TestConnectionOpen()
        {
        Connection connectionToOracleDB;
```

```
try
{
connectionToOracleDB = Connection.createConnection();
}
catch (ConnectionException exception)
{
Assert.Fail("Expected no exception, but got: " + exception.Message);
}
}

[TestMethod]
public void TestConnectionClose()
{
Connection connectionToOracleDB = null;

try
{
connectionToOracleDB = Connection.createConnection();
}
catch (ConnectionException exception)
{
Assert.Fail("Expected no exception, but got: " + exception.Message);
}

try
{
 connectionToOracleDB.closeConnection();
```

```
connectionToOracleDB = null;
}
catch (ConnectionException exception)
{
Assert.Fail("Expected no exception, but got: " + exception.Message);
}
}
}
```