

# **Documentul de Proiectare a Solutiei Aplicatiei Software (Software Design Document)**

**Version 1.0  
30 mai , 2023**

## **Password Management Tool**

**Coordonator,  
Prof. Tiberius Dumitriu**

**Grupa 1306 B**

**Studenti,  
Budacă M. Marius-Andrei  
Budu V. Daniel  
Paşa P.C. Anamaria-Larisa  
Pintilie V. Răzvan-Nicolae**

**Facultatea de Automatică şi Calculatoare, Universitatea  
“Gheorghe Asachi” , IASI**

# **Cuprins**

## **1. Scopul documentului**

## **2. Continutul documentului**

## **3. Modelul datelor**

- 3.1 Structuri de date globale**
- 3.2 Structuri de date de legatura**
- 3.3 Structuri de date temporare**
- 3.4 Formatul fisierelor utilizate**
- 3.5 Descrierea bazei de date**
  - 3.5.1 Diagrama schemei bazei de date**
  - 3.5.2 Descrierea tabelor**

## **4. Model arhitectural si modelul componentelor**

- 4.1 Arhitectura sistemului**
  - 4.1.1 Sabloane arhitecturale folosite**
  - 4.1.2 Diagrama de arhitectura**
- 4.2 Descrierea componentelor**
- 4.3 Restrictiile de implementare**
- 4.4 Interacțiunea dintre componente**

## **5. Modelul interfeței cu utilizatorul**

- 5.1 Succesiunea interfețelor**
- 5.2 Ferestrele aplicației**
  - 5.2.1 Fereastră Log In**
  - 5.2.2 Fereastră Sign Up**
  - 5.2.3 Fereastră Home**
  - 5.2.4 Fereastră Adauga credentiale**
  - 5.2.5 Fereastră Vizualizare date**
  - 5.2.6 Fereastră Setari**

## **6. Elemente de testare**

- 6.1 Componente critice**
- 6.2 Alternative**
- 6.3 Rezultatele testelor**

## **7. Anexa**

## 1. Scopul documentului

Acest document are scopul de a furniza o descriere precisă a soluției proiectate pentru aplicația software Password Management Tool . El îndeplinește rolul de a oferi un ghid integral pentru echipa de dezvoltare a proiectului, asistându-i în construirea soluției.

## 2. Conținutul documentului

Documentul este format din patru secțiuni esențiale:

Modelul datelor prezintă principalele structuri de date folosite, precum și schema bazei de date

Modelul arhitectural și modelul componentelor prezintă șabloanele arhitecturale folosite, arhitectura sistemului și descrie componentele arhitecturii

Modelul interfeței cu utilizatorul – prezintă interfața cu utilizatorul și succesiunea ferestrelor acesteia

Elemente de testare prezintă componentele critice și alternative de proiectare a acestora.

## 3. Modelul datelor

### 3.1. Structuri de date globale

Aplicația are o instanță globală a clasei Controller prezentă în modulul cu același nume care reprezintă legătura între procesarea datelor în clasele din modulul BazaDeDate (operații CRUD), procesarea filtrelor în modulul GenerareParola (folosind șablonul Builder) și inputurile venite din Interfața.

De asemenea, în modulul destinat Interfeței, există mai multe User Control care oferă posibilitatea navigării prin mai multe pagini. Toate acestea au drept Form părinte Form1, clasa parțială care este implementată folosind șablonul Singleton.

### 3.2. Structuri de date de legătură

Aplicația se conectează la o bază de date SQLite, unde asigură existența a două tabele: cea a utilizatorilor și cea a credențialelor stocate. Operațiile pe baza acestor tabele sunt cele de tipul CRUD. Toate acestea sunt implementate într-un modul separat. Modulul de GenerareParola asigură crearea unei parole complexe, creată conform filtrelor dorite de utilizator. Modulul GUI al aplicației comunică cu celelalte module de prelucrare date printr-un modul-legătură numit Controller. Acesta are grijă ca argumentele ce contin cheile primare, de pilda, din baza de date (utilizator, filtre dorite, credențiale noi) să fie populate conform inputurilor provenite din interfața. Astfel, se asigură o organizare eficientă, curată a codului, implementându-se principiul Model-View-Controller.

### 3.3 Structuri de date temporare

Structurile de date temporare nu sunt prezente în această aplicație, întrucât modulele existente asigură orice prelucrare necesară. Astfel, consumul de resurse, prin lipsa structurilor de date temporare, este semnificativ mai redus.

### 3.4 Formatul fișierelor utilizate

Fișierele utilizate în această aplicație sunt clasice: există fișiere sursă cu programul scris în C#, fișiere de tipul Dynamic Link Library care sunt adăugate drept

referinte catre celelalte module din aplicatie, fisiere pentru baza de date. Toate acestea asigura legatura intre module si functionarea corecta a aplicatiei.

### 3.5 Descrierea bazei de date

#### 3.5.1 Diagrama schemei bazei de date

Modelul bazei de date este format din următoarele tabele inter-relaționate ca în figura de mai jos:

Users		Passwords	
User	INTEGER	User	VARCHAR
Token	VARCHAR	Domain_user	VARCHAR
Salt	VARCHAR	Email	VARCHAR
		Domain	VARCHAR
		Password	VARCHAR

#### 3.5.2 Descrierea tabelelor

Schema bazei de date cuprinde următoarele tabele:

- **Users**- contine informatii despre contul utilizatorului si legatura catre urmatoarea tabela de parole : ID, user, PIN-ul PC-ului si salt catre tabela Passwords
- **Passwords** - contine informatii despre credentialele userului, site-ul la care se conecteaza cu acestea .

## 4. Model arhitectural si modelul componentelor

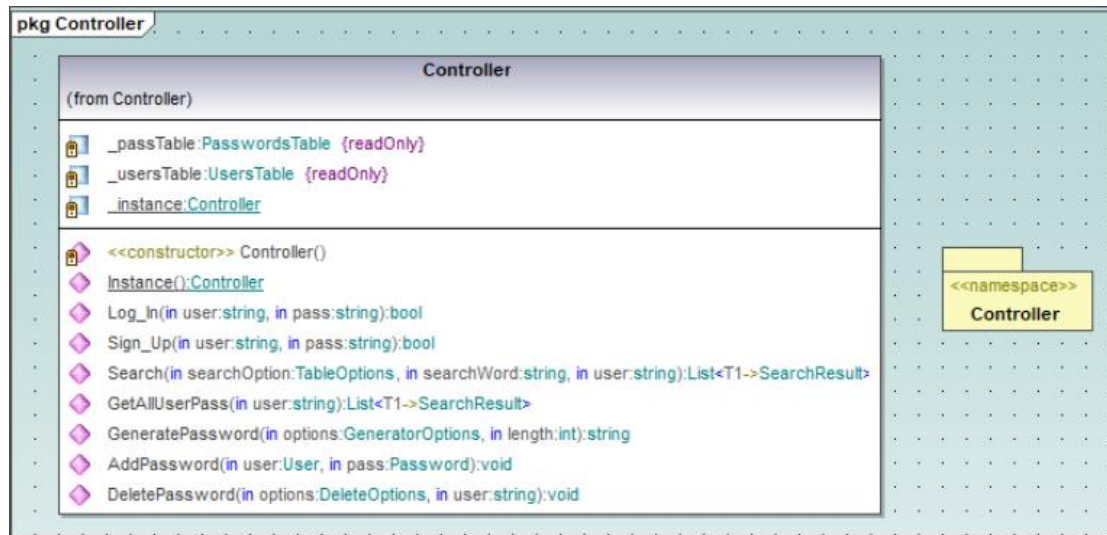
### 4.1 Arhitectura sistemului

#### 4.1.1 Sabloane arhitecturale folosite

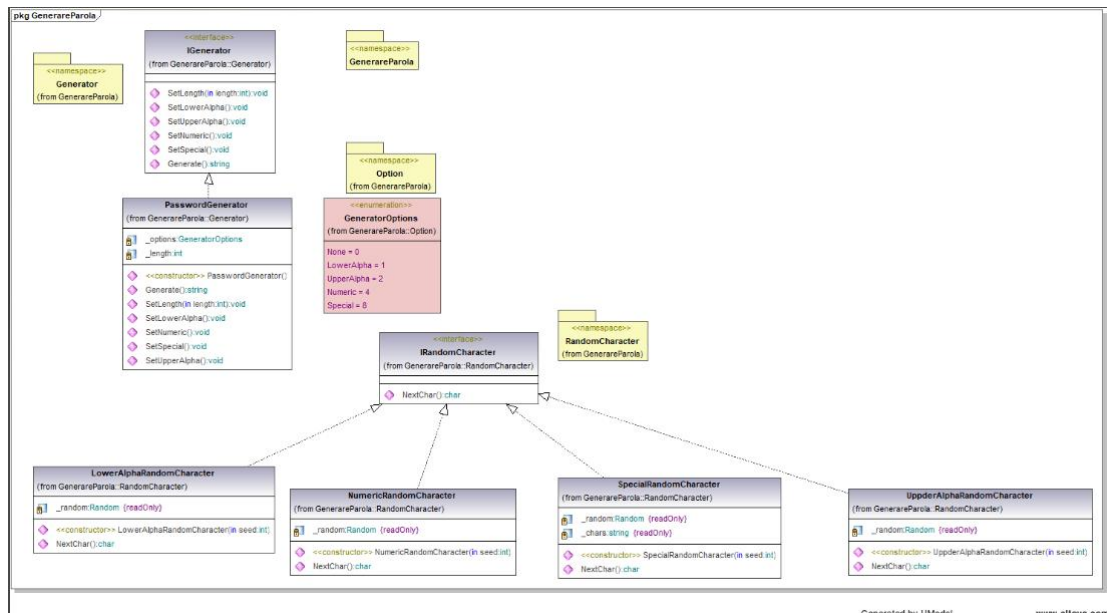
Această aplicație este proiectată utilizând modelul arhitectural Model-View-Controller (MVC). Această arhitectură are scopul de a separa componentele aplicației în mod clar și ușor de gestionat. Modelul se ocupă de stocarea datelor, logica de afaceri și validarea informațiilor. View-ul se concentrează pe prezentarea datelor utilizatorului și pe colectarea și transmiterea informațiilor către Controller. Controller-ul are rolul de a procesa datele, de a interacționa cu modelul și cu view-ul, precum și de a gestiona fluxul de date în cadrul aplicației.

În implementarea aplicației, s-au utilizat următoarele șabloane de proiectare:

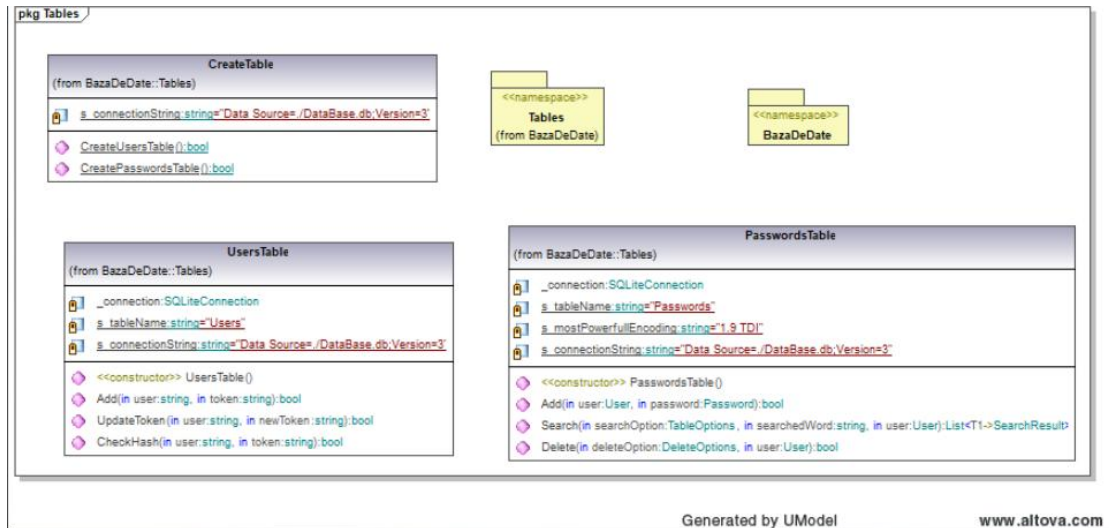
- **Sablonul Singleton** este un model de proiectare care permite crearea unei singure instanțe a unei clase și furnizează un punct global de acces la această instanță . Acest sablon arhitectural este utilizat pentru a garanta ca exista un singur Controller in aplicatie, instanta care doar ea poate gestiona si realiza legatura View - Model. In acest mod se evita eventualele conflicte legate de modul/locul si momentul prelucrării datelor.



- **Sablonul Builder** facilitează construirea obiectelor complexe pas cu pas, separând procesul de construcție de reprezentarea finală a acestora. Acesta oferă flexibilitate și extensibilitate în crearea obiectelor, permițând adăugarea și modificarea pașilor de construcție fără a afecta codul existent. Acest sablon arhitectural este utilizat în crearea unei parole automate noi. Existând posibilitatea de a alege modul în care se creează parola (lungime, caracterele continuate), crearea acesteia devine un proces complex și dificil care poate fi considerabil simplificat prin împartirea acestui proces de creare în mai mulți pași: fiecare tip de caracter solicitat a fi prezent în parola pus în clasa separată etc.

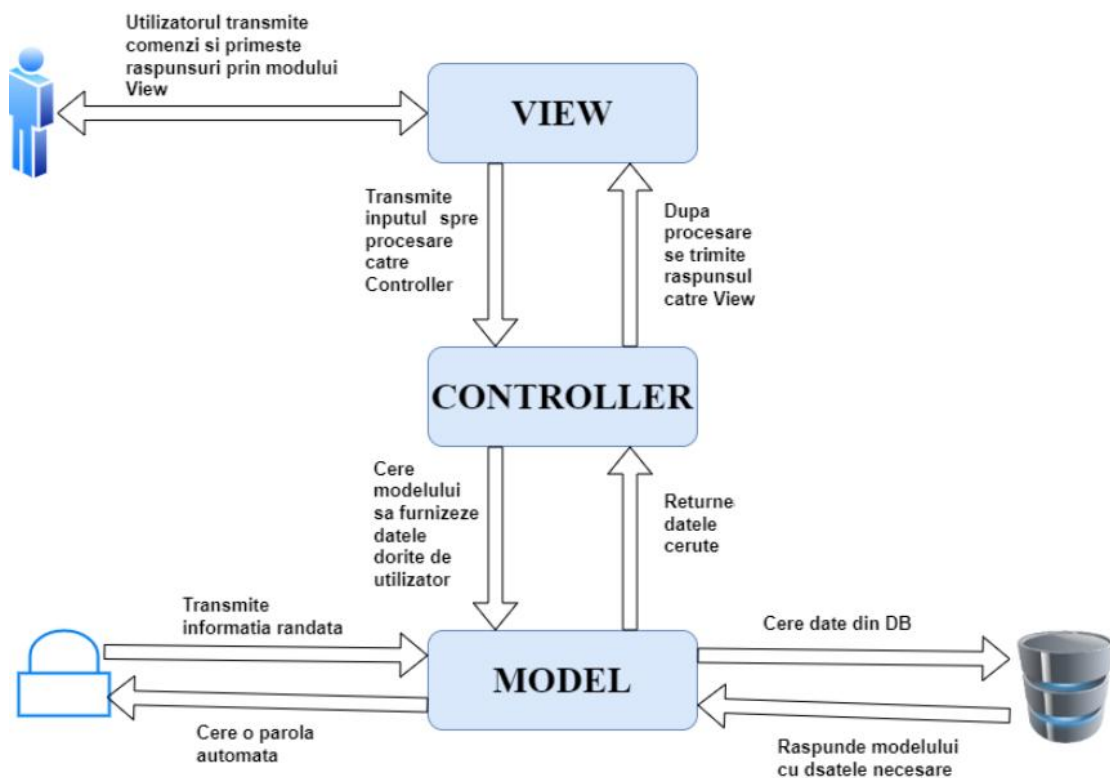


- **Sablonul Repository** separă logica de acces la date și persistența acestora de logica de afaceri a unei aplicații. Prin intermediul său, putem defini operațiuni standardizate pentru manipularea datelor, ascunzând detalii specifice ale surselor de date. Acest sablon arhitectural este utilizat în modulul BazeDeDate.



#### 4.1.2 Diagrama de arhitectura

Diagrama de arhitectură de mai jos descrie componentele arhitecturii aplicației și relațiile de interacțiune dintre acestea.



#### 4.2 Descrierea componentelor

Aplicația este alcătuită dintr-o serie de module interconectate, care colaborează pentru a implementa arhitectura MVC și pentru a asigura o separare distinctă între logica de afaceri, interfața utilizatorului și manipularea datelor.

- **Modulul GUI (Graphical User Interface)**

Modulul GUI (Graphical User Interface) este o componentă esențială a arhitecturii MVC, asumându-și rolul de View. Acest modul se ocupă de aspectul vizual și de interfața cu care utilizatorul interacționează. Prin intermediul său, informațiile sunt prezentate în mod atrăgător utilizatorului, iar acțiunile acestuia sunt recepționate prin elemente interactive, precum butoane, câmpuri de text și altele.

asemenea. Cu toate acestea, modulul GUI nu se implică în logica de afaceri, concentrându-se exclusiv pe aspectul și interacțiunea vizuală cu utilizatorul.

- **Modulul de Logică**

Modulul de Logică reprezintă componenta centrală a arhitecturii MVC, având rolul de Controller. Acesta gestionează logica de afaceri a aplicației, procesează datele și implementează regulile specifice. Interacțiunea cu modulul GUI și modulul de BazeDeDate este realizată de modulul de Logică pentru a efectua operațiile necesare. În aplicația de gestionare a credentialelor pentru diferite site-uri, acest modul include logica pentru alegerea operațiilor CRUD necesare și determinate de utilizator prin inputuri .

- **Modulul BazeDeDate**

Modulul de Baze de date reprezintă componenta Model din arhitectura MVC în cadrul aplicației de gestionare a parolelor. Acesta se ocupă de stocarea și accesul la datele referitoare la parole. Funcționalitatea modulului de Baze de date include crearea, actualizarea și interogarea bazei de date utilizate în aplicație. Prin intermediul claselor și metodelor sale, acest modul facilitează manipularea datelor și comunicarea cu sistemul de gestiune a bazei de date. În aplicația Password Management Tool, acest modul este responsabil de realizarea conectării la baza de date, crearea tabelurilor necesare în funcție de existența/inexistența lor inițială, operațiile CRUD și de rulare a procedurilor stocate.

- **Modulul GenerareParola**

Modulul de GenerareParola reprezintă un element al componentei Model din arhitectura MVC în care se tratează nevoia de generare automată a unei parole conforme cu filtrele aplicate de utilizator. Asemenea modulului BazeDeDate, GenerareParola se ocupă, deci, cu prelucrarea datelor .

### **4.3 Restricțiile de implementare**

- Va fi dezvoltat un modul de Baze de date care să faciliteze comunicarea cu baza de date SQLite utilizând DB Browser SQLite. Acest modul va include funcționalități pentru crearea și administrarea bazei de date, precum și pentru stocarea și preluarea datelor cu caracter personal ale utilizatorului.

- Modulele GUI, BazeDeDate, Controller și GenerareParola vor fi implementate în limbajul de programare C#.

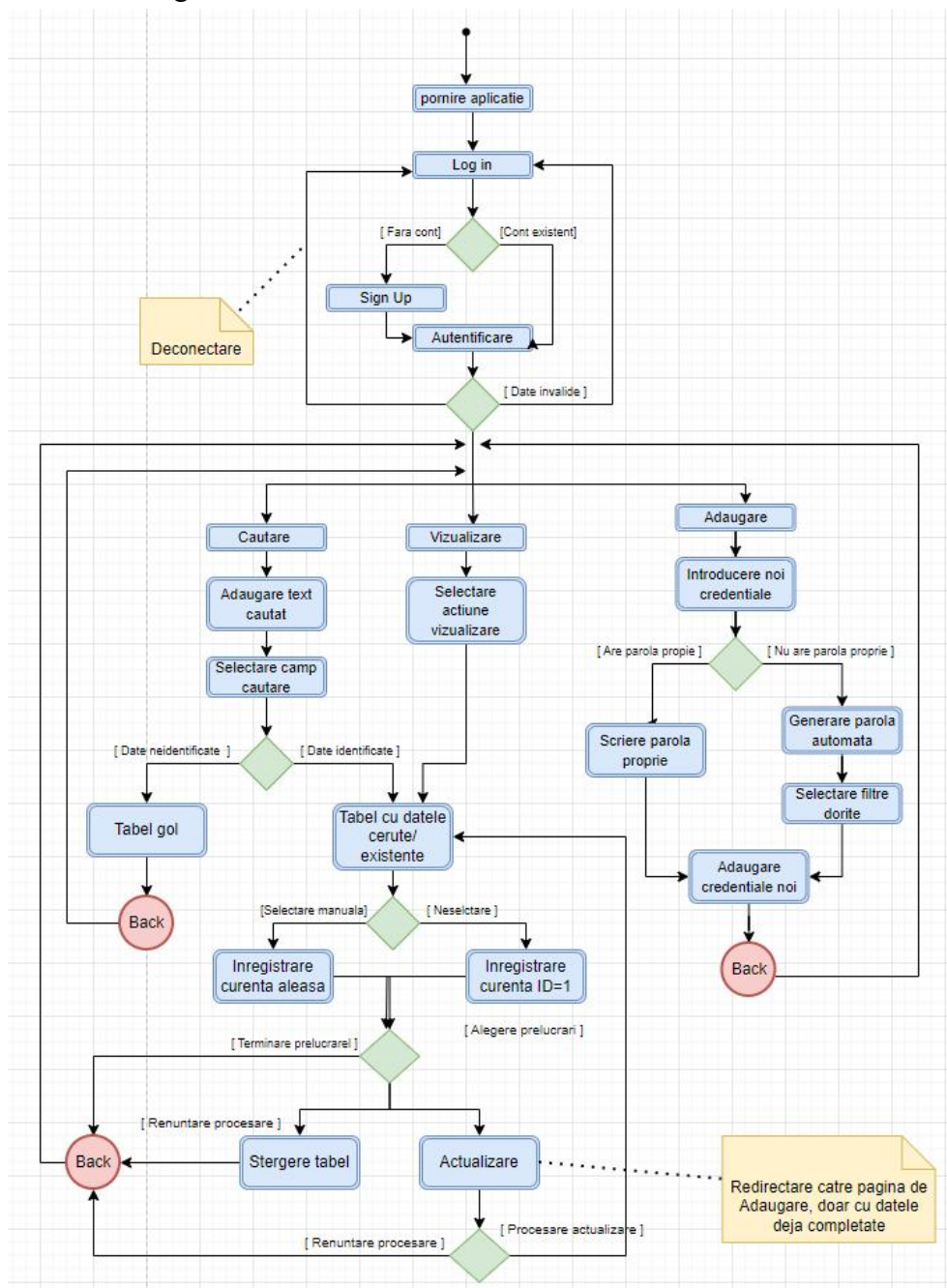
- Modulele vor fi dezvoltate ca proiecte de tip class library, generând DLL-uri separate.

- Se vor respecta principiile și standardele de programare specifice limbajului C# în implementarea modulelor : denumirea corectă a variabilelor ( Camel Case ), adăugarea antetului în fiecare fișier sursă, adăugarea comentariilor pentru fiecare funcție etc

- Se vor utiliza bibliotecile și framework-urile disponibile în C# pentru a obține funcționalitățile dorite în fiecare modul.

## 4.4 Interacțiunea dintre componente

Interacțiunea componentelor aplicației se realizează modularizat, oferind posibilitatea de extindere fără a modifica partea de cod existentă. Responsabilitățile sunt total separate, Controllerul fiind cel care dirijează mersul aplicației. Existența sabloanelor de proiectare eficientizează codul, rezultând un program organizat, curat, fiecare clasă fiind responsabilă doar de un singur aspect. DLL-urile oferă posibilitatea de a împărți codul în funcție de responsabilități și de a oferi modulelor opțiunea de a adăuga doar referințele de care au nevoie.



În continuare, pe baza diagramei afișate mai sus, voi prezenta fluxul natural de navigare prin aplicație. În calitate de utilizator, acesta va avea acces doar la acest flow, de el depinzând buna și ușoară interacțiune client - produs. Pentru început, atunci când utilizatorul porneste aplicația, se va lansa fereastra de Log In a aplicației. Acesta se poate conecta foarte ușor prin nume de PIN-ul PC-ului sau poate alege să

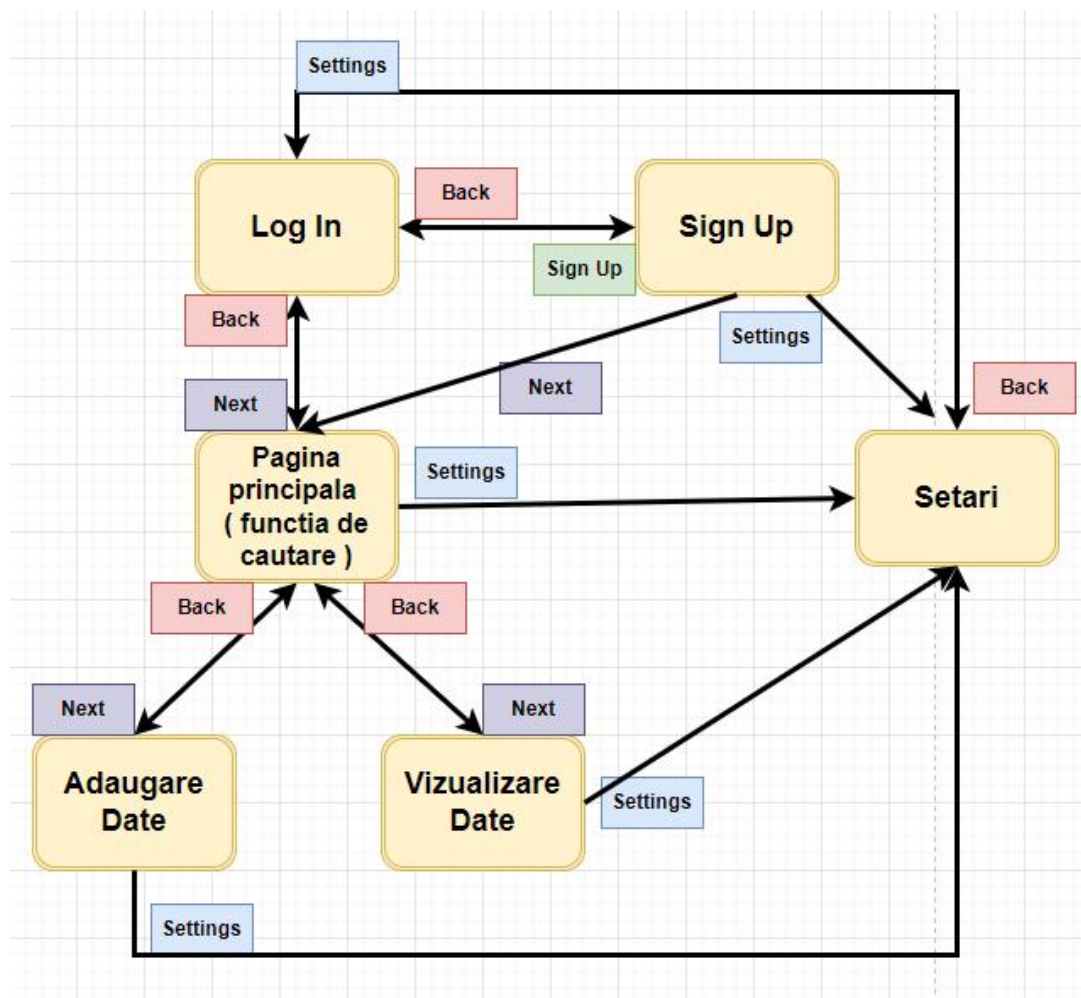


mergea pe pagina de Sign Up, in cazul in care nu dispune de cont. In ambele cazuri, redirectarea se face catre pagina principala numita pagina Home. Aici utilizatorul are acces facil ca cautarea rapida de informatii sau alte doua optiuni , adaugare credentiale noi si vizualizare date. Ambele butoane transfera utilizatorul intr-o pagina noua, acolo unde poate executa operatiile dorite. Pentru vizualizarea datelor selectate/tuturor datelor, exista o singura pagina unde informatiile sunt stocate intr-un tabel. Tot in aceasta pagina de vizualizare, utilizatorul are optiunea de a sterge o inregistrare sau de a actualiza una. In final, in urma aflarii/adaugarii de informatii, fie este redirectat iarasi catre pagina de Home, fie se poate intoarce singur cu ajutorul butonului de Back.

## 5. Modelul interfeței cu utilizatorul

### 5.1 Succesiunea interfețelor

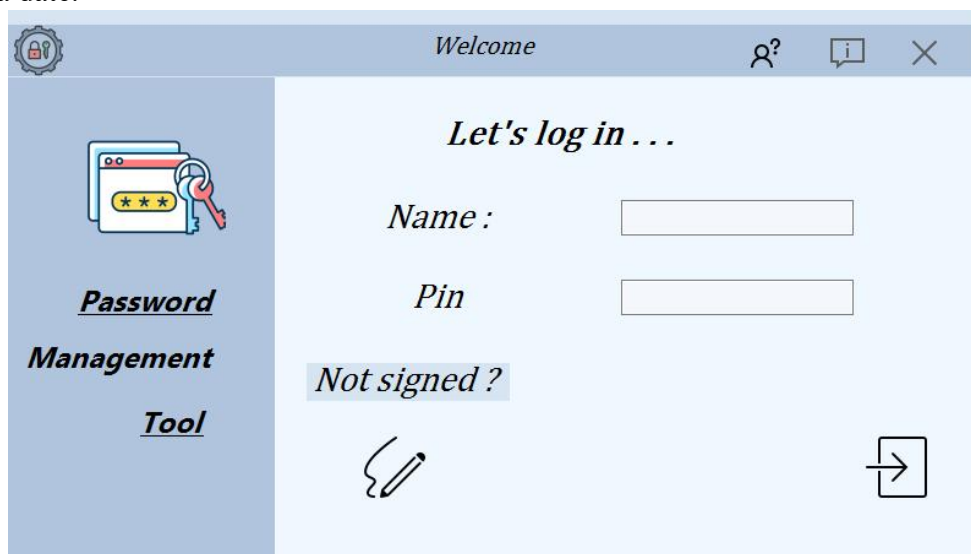
În cadrul modulului GUI, ferestrele aplicației sunt afișate respectând un flux stabilit în conformitate cu standardele academice. Acest flux este descris grafic în figura de mai jos:



## 5.2 Ferestrele aplicației

### 5.2.1 Fereastra Log In

Prima fereastra a aplicației obliga utilizatorul să se autentifice pentru a avea acces la date.



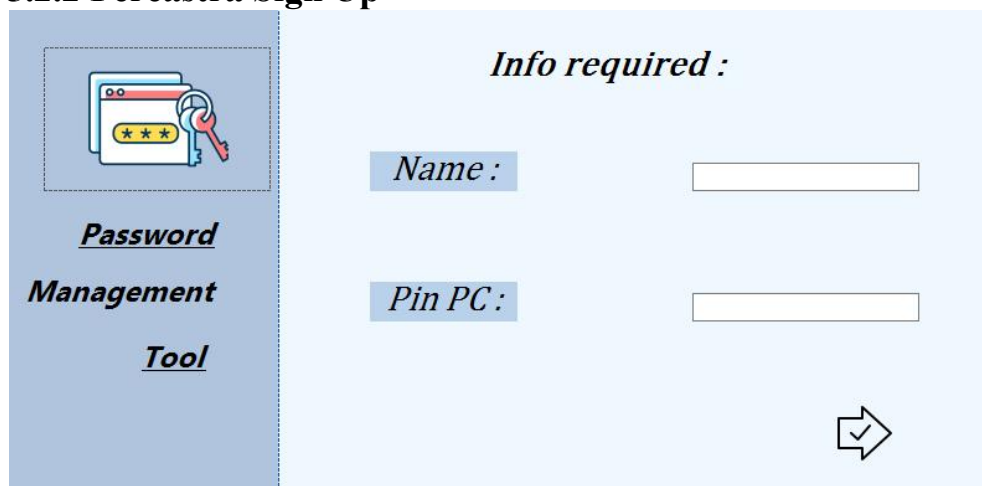
Această fereastră dispune de un meniu în partea de sus unde sunt disponibile următoarele controale:

- **Buton EXIT**
- **Buton INFO** - utilizatorul poate afla detalii despre echipa care a implementat acest tool.
- **Butonul HELP** - unde accesează un document folositor pentru înțelegerea modului în care funcționează aplicația
- **Butonul SETARI** - unde accesează diferite setări ale aplicației: schimbarea temei și altele

În partea centrală, se identifică formularul de autentificare care conține:

- **TextBox-uri** pentru introducerea datelor
- **Buton LOG IN** - redirecționează utilizatorul (în cazul datelor corecte) către pagina HOME
- **Buton SIGN UP** - redirecționează utilizatorul către pagina de creare cont

### 5.2.2 Fereastra Sign Up

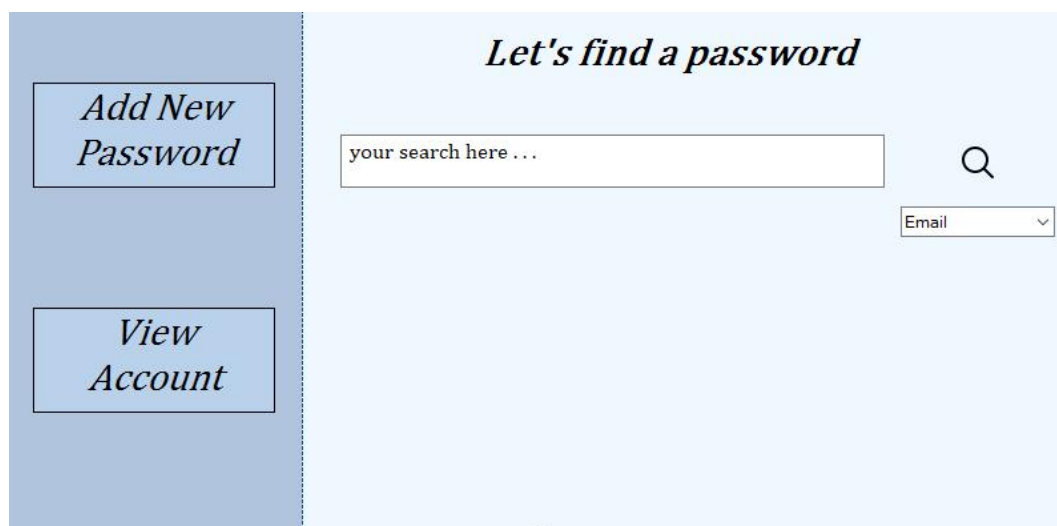


Aceasta fereastră dispune de un meniu in partea de sus unde sunt disponibile urmatoarele controale ( prezentate in partea de Log In)

In partea centrala , se identifica formularul de autentificare care contine:

- **TextBox**-uri pentru introducerea datelor
- **Buton SIGN UP** - redirecteaza utilizatorul (in cazul datelor corecte ) catre pagina HOME

### 5.2.3 Fereastră Home



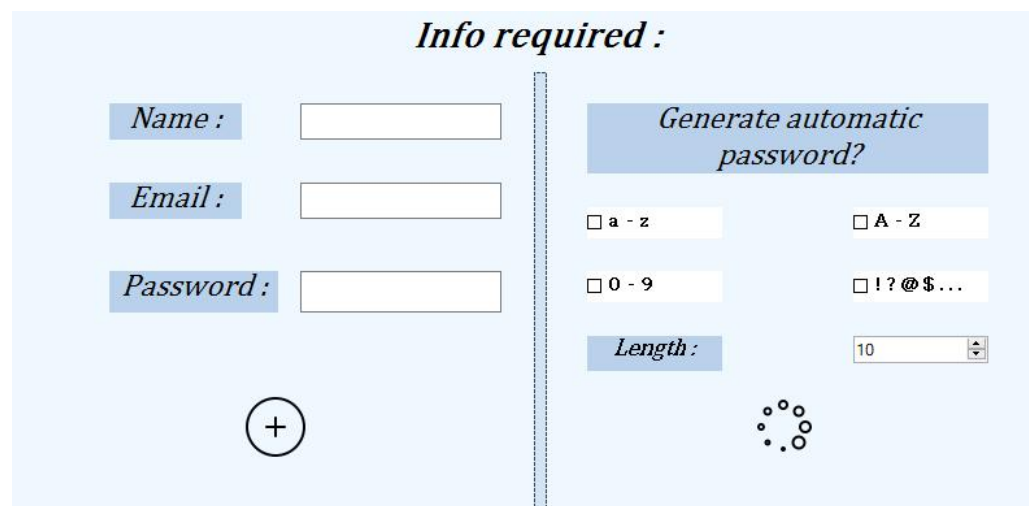
The screenshot shows a web interface for a password management system. On the left, there is a vertical sidebar with two buttons: "Add New Password" and "View Account". The main area has a heading "Let's find a password" and a search bar with the placeholder text "your search here ...". To the right of the search bar is a magnifying glass icon and a dropdown menu labeled "Email".

Aceasta fereastră dispune de un meniu in partea de sus unde sunt disponibile urmatoarele controale ( prezentate in partea de Log In)

In partea centrala , se identifica formularul de autentificare care contine:

- **TextBox** pentru introducerea elementului spre a fi gasit
- **Drop Down List** - folosita pentru a specifica tipul de element cautat
- **Buton SEARCH** - redirecteaza utilizatorul catre pagina de SEARCH unde poate vizualiza, intr-un tabel , datele identificate
- **Buton ADD NEW PASSWORD** - redirecteaza utilizatorul intr-o alta fereastră unde poate adauga credentialele dorite
- **Buton VIEW ACCOUNT** - redirecteaza utilizatorul intr-o pagina pentru a vizualiza usor toate informatiile pe care el le-a introdus

### 5.2.4 Fereastră Adauga credential



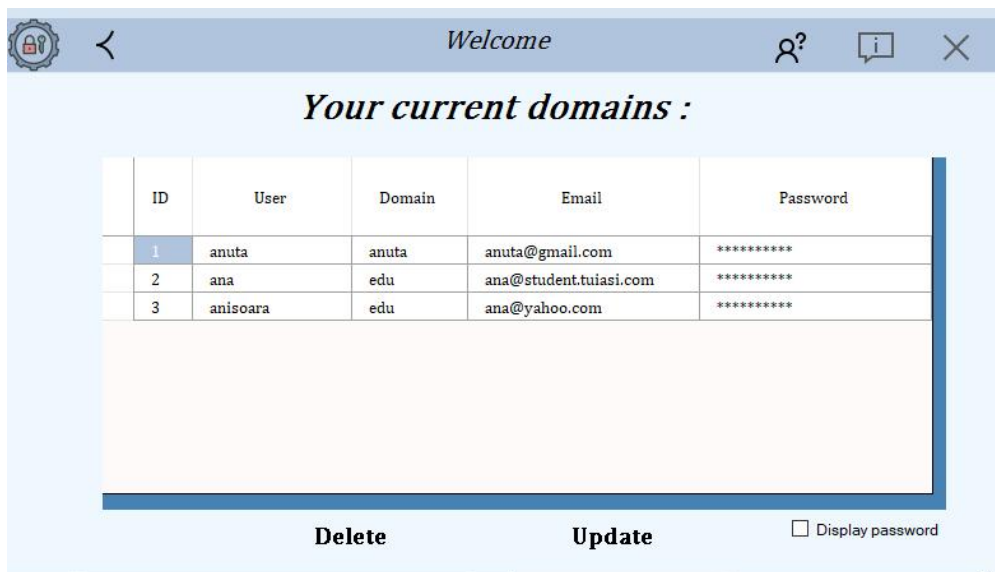
The screenshot shows a form titled "Info required :". It has three input fields on the left: "Name :", "Email :", and "Password :". To the right of these fields is a section titled "Generate automatic password?" which contains four checkboxes: "a - z", "A - Z", "0 - 9", and "! ? @ \$ ...". Below these checkboxes is a "Length :" label and a dropdown menu showing the value "10". At the bottom left of the form is a circular button with a plus sign (+), and at the bottom right is a circular button with a refresh icon.

Aceasta fereastră dispune de un meniu in partea de sus unde sunt disponibile urmatoarele controale ( prezentate in partea de Log In)

In partea centrala , se identifica formularul de autentificare care contine:

- **TextBox-uri** pentru adaugarea datelor noi (nume, email, parola)
- **CheckBox-uri** pentru selectarea filtrelor dorite pentru parola automata
- **Drop Down List** pentru selectarea lungimii parolei
- **Buton GENERATE PASSWORD** - se transmite Controllerului care apeleaza clasa GenerareParola si se creeaza noua parola dorita
- **Button ADD** - adauga in baza de date noua intrare si redirecteaza utilizatorul catre HOME

### 5.2.6 Ferestra Vizualizare date



ID	User	Domain	Email	Password
1	anuta	anuta	anuta@gmail.com	*****
2	ana	edu	ana@student.tuiasi.com	*****
3	anisoara	edu	ana@yahoo.com	*****

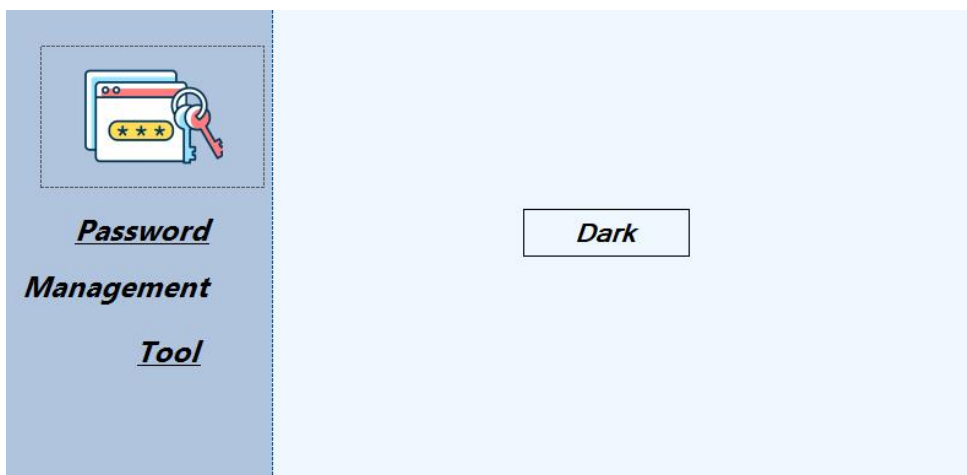
Delete Update ☐ Display password

Aceasta fereastră dispune de un meniu in partea de sus unde sunt disponibile urmatoarele controale ( prezentate in partea de Log In)

In partea centrala , se identifica formularul de autentificare care contine:

- **Data Grid View** - tabelul unde utilizatorul are listate toate datele cerute
- **Buton DELETE** - ofera posibilitatea utilizatorului sastearga o inregistrare selectata
- **Buton UPDATE** - redirecteaza utilizatorul catre pagina de ADAUGARE unde campurile sunt deja completate cu inregistrarea selectata spre a fi actualizata

### 5.2.7 Ferestra Setari



**Password Management Tool**

Dark

Aceasta fereastra dispune de un meniu in partea de sus unde sunt disponibile urmatoarele controale ( prezentate in partea de Log In)

In partea centrala , se identifica formularul de autentificare care contine:

- **Button DARK** - care schimba tema aplicatiei

## 6. Elemente de testare

### 6.1 Componente critice

Componentele critice ale Password Management Tool ar putea fi:

- Modul de interactiune cu baza de date. Mai exact, felul in care toate operatiile CRUD sunt executate si eficienta cu care se randeaza/stocheaza/actualizeaza datele

### 6.2 Alternative

Alternativele pentru a optimiza aceasta aplicatie ar putea fi :

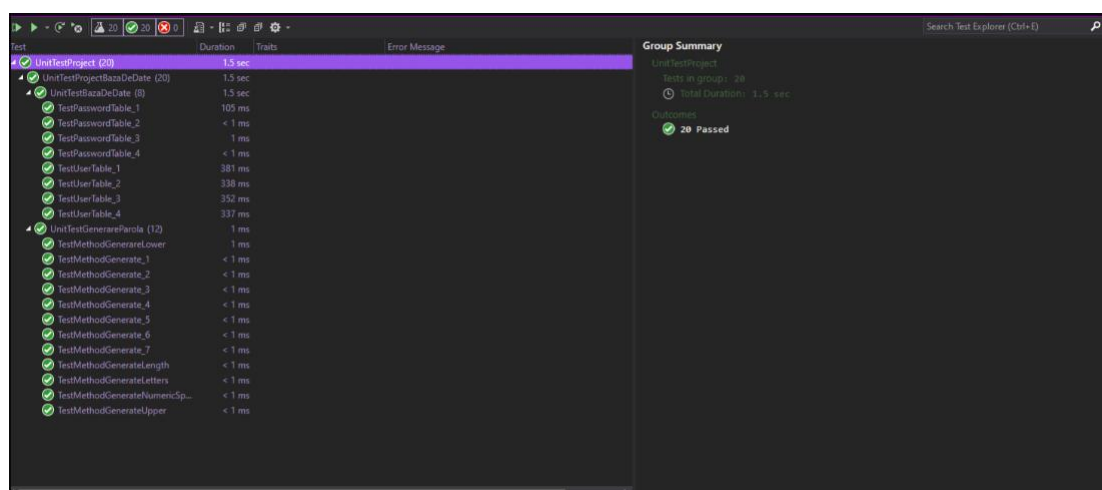
- Implementarea unui modul care simuleaza Cache-ul unei aplicatii. In loc de a cauta mereu in baza de date , se poate apela la o astfel de optiune eficienta cand sunt foarte multe date .

### 6.3 Rezultatele testelor

Aplicatia Password Management Tool dispune de un modul ce are drept scop testarea aplicatiei si functiilor din toate modulele. De pilda:

```
[TestMethod]
0 references
public void TestPasswordTable_1()
{
    PasswordsTable ps = new PasswordsTable();
    Assert.AreEqual(true, ps.Add(new BazaDeDate.DataClasses.User("om"), new BazaDeDate.DataClasses.Password("email", "domain", "pass", "domainUser")));
    Assert.AreEqual(true, ps.Delete(new BazaDeDate.Option.DeleteOptions("om", "email", "domain", "pass"), new BazaDeDate.DataClasses.User("om")));
}
```

In continuare, se va prezenta o imagine cu rezultatele testelor efectuate asupra versiunii finale ale aplicatiei.



## 7. Anexa

În cadrul anexei curente se vor menționa părțile semnificative ale codului aplicației, acele segmente care produc efecte substanțiale în cadrul proiectului.

Un exemplu de cod este cel din modulul `GenerareParola`, mai exact funcția care gestionează toate opțiunile alese de utilizator și randează o parolă complexă:

```
public string Generate()
{
    var seed = (int)(DateTime.Now.Ticks % int.MaxValue);
    var characterSet = new List<IRandomCharacter>();
    var generatedPass = new char[_length];
    var random = new Random(seed);

    if (_options == GeneratorOptions.None)
        _options = GeneratorOptions.LowerAlpha;

    if ((_options & GeneratorOptions.LowerAlpha) != 0)
    {
        characterSet.Add(new LowerAlphaRandomCharacter(seed));
    }

    if ((_options & GeneratorOptions.UpperAlpha) != 0)
    {
        characterSet.Add(new UppderAlphaRandomCharacter(seed));
    }

    if ((_options & GeneratorOptions.Numeric) != 0)
    {
        characterSet.Add(new NumericRandomCharacter(seed));
    }

    if ((_options & GeneratorOptions.Special) != 0)
    {
        characterSet.Add(new SpecialRandomCharacter(seed));
    }

    var setLength = characterSet.Count;

    for (int i = 0; i < generatedPass.Length; i++)
    {
        generatedPass[i] = characterSet[random.Next(setLength)].NextChar();
    }

    // Generate a character from all character generators to ensure
    // that the password will
    // contain a character from all character sets
    int startPos = 0;
    for (int i = 0; i < setLength; i++)
    {
        var posToChange = random.Next(startPos, _length - setLength +
i + 1);
        generatedPass[posToChange] = characterSet[i].NextChar();
        startPos = posToChange + 1;
    }

    return new string(generatedPass);
}
```

Un alt exemplu semnificativ este clas Controller, acolo unde este implementat sablonul Singleton. Controllerul este cel care gestioneaza toate actiunile si legaturile dintre Model si View.

```
/// <summary>
/// Class for controller
/// </summary>
public class Controller
{
    private readonly PasswordsTable _passTable;
    private readonly UsersTable _usersTable;

    private static Controller _instance;

    /// <summary>
    /// Constructor Controller class
    /// </summary>
    private Controller()
    {
        _passTable = new PasswordsTable();
        _usersTable = new UsersTable();

        CreateTable.CreateUsersTable();
        CreateTable.CreatePasswordsTable();
    }

    /// <summary>
    /// Method for return instance of single object(singleton)
    /// </summary>
    /// <returns>New controller object</returns>
    public static Controller Instance()
    {
        if (_instance == null)
            _instance = new Controller();
        return _instance;
    }

    .....
}
```