

Documentație proiect: Manager de parole

Disciplina: Structuri software pentru aplicații de timp-real

Student: Știole Alexandru-George

An universitar: 2024-2025

Cuprins

1. Descriere generală a proiectului
 - 1.1. Scopul proiectului
 - 1.2. Structura generală a aplicației
 - 1.3. Clasa PasswordDAO
 - 1.4. Clasa UserDAO

2. Detalii de implementare
 - 2.1. Proiectarea bazei de date - Modelul conceptual - Modelul logic - Implementare SQL - Popularea bazei de date
 - 2.2. 2.2 Modulele aplicației - Interfața grafică - Criptarea datelor - Conexiunea cu baza de date
 - 2.3. 2.3 Funcționalități implementate

3. Utilizarea aplicației
 - 3.1. Pagina Login
 - 3.2. Funcția Register
 - 3.3. Popularea bazei de date - users
 - 3.4. Pagina main
 - 3.5. Adăugarea unei parole
 - 3.6. Popularea bazei de date - passwords
 - 3.7. Decriptarea parolelor
 - 3.8. Stergerea parolelor

4. Concluzii și perspective de dezvoltare

1. Descriere generală a proiectului

1.1 Scopul proiectului

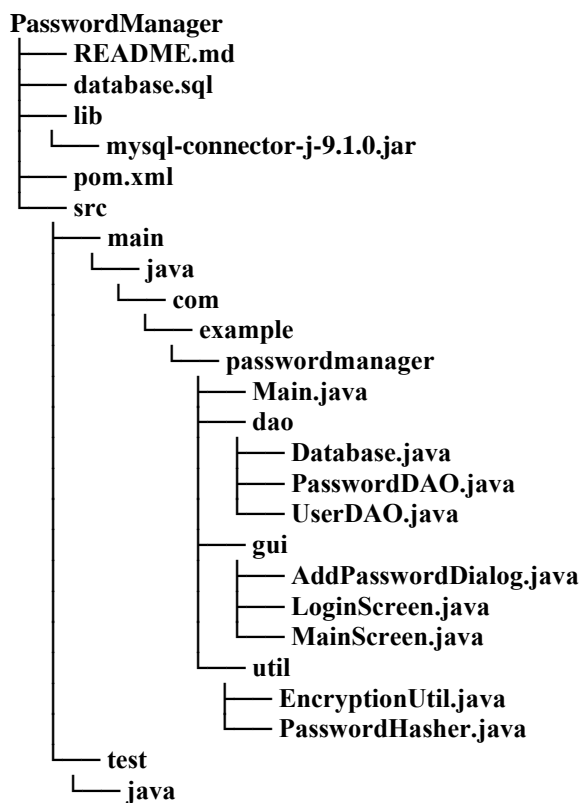
Proiectul are ca scop dezvoltarea unei aplicații pentru gestionarea sigură a parolelor utilizatorilor. Aplicația oferă funcționalități precum:

- Stocarea criptată a parolelor în baza de date.
- Gestionarea conturilor utilizatorilor (autentificare și înregistrare).
- Adăugarea, vizualizarea și ștergerea parolelor asociate unui cont.

1.2 Structura generală a aplicației

Aplicația este compusă din următoarele module principale:

- Interfața grafică (folosind Swing pentru Java).
- Sistemul de autentificare și gestionare a utilizatorilor.
- Un modul pentru criptarea și decriptarea parolelor.
- O bază de date MySQL pentru stocarea datelor utilizatorilor și a parolelor.



Structura directoarelor și fișierelor proiectului.

Descrierea Componentelor

1. **Main.java**

Acesta este punctul de intrare al aplicației. Initializează și lansează ecranul de autentificare.

2. **gui/**

- **LoginScreen.java:** Gestionează autentificarea și înregistrarea utilizatorilor.
- **MainScreen.java:** Oferă o interfață pentru vizualizarea, adăugarea și ștergerea parolelor.
- **AddPasswordDialog.java:** Oferă un dialog pentru adăugarea unei parole noi.

3. **dao/**

- **Database.java:** Gestionează conexiunea la baza de date.
- **UserDAO.java:** Gestionează operațiunile legate de utilizatori (autentificare, înregistrare).
- **PasswordDAO.java:** Gestionează operațiunile legate de parole (adăugare, ștergere, listare).

4. **util/**

- **EncryptionUtil.java:** Gestionează criptarea și decriptarea parolelor.

1.3 Clasa *PasswordDAO*

Clasa **PasswordDAO** este responsabilă pentru interacțiunile cu tabela **passwords** din baza de date. Aceasta implementează funcționalități precum:

Adăugarea unei parole:

```
public boolean addPassword(int userId, String siteName, String username, String encryptedPassword) {  
    String query = "INSERT INTO passwords (user_id, site_name, username, password) VALUES (?, ?, ?, ?)";  
    try (Connection conn = Database.getConnection();  
        PreparedStatement stmt = conn.prepareStatement(query)) {  
        stmt.setInt(1, userId);  
        stmt.setString(2, siteName);  
        stmt.setString(3, username);  
        stmt.setString(4, encryptedPassword);  
        stmt.executeUpdate();  
        return true;  
    }  
}
```

Obținerea parolelor pentru un utilizator:

```
public List<String[]> getPasswords(int userId) {  
    List<String[]> passwords = new ArrayList<>();  
    String query = "SELECT site_name, username, password FROM passwords WHERE user_id = ?";  
}
```

```

try (Connection conn = Database.getConnection();
    PreparedStatement stmt = conn.prepareStatement(query)) {
    stmt.setInt(1, userId);
    ResultSet rs = stmt.executeQuery();
    while (rs.next()) {
        passwords.add(new String[]{
            rs.getString("site_name"),
            rs.getString("username"),
            rs.getString("password")
        });
    }
}

```

Ștergerea unei parole:

```

public boolean deletePassword(int userId, String siteName) {
    String query = "DELETE FROM passwords WHERE user_id = ? AND site_name = ?";
    try (Connection conn = Database.getConnection();
        PreparedStatement stmt = conn.prepareStatement(query)) {
        stmt.setInt(1, userId);
        stmt.setString(2, siteName);
        stmt.executeUpdate();
        return true;
    }
}

```

1.4 Clasa *UserDAO*

Clasa *UserDAO* gestionează interacțiunile cu tabela **users**. Aceasta implementează funcționalități precum:

Înregistrarea unui utilizator nou:

```

public boolean registerUser(String username, String passwordHash) {
    String query = "INSERT INTO users (username, password_hash, encryption_key) VALUES (?, ?, ?)";
    try (Connection conn = Database.getConnection();
        PreparedStatement stmt = conn.prepareStatement(query)) {
        stmt.setString(1, username);
        stmt.setString(2, passwordHash);

        SecretKey key = EncryptionUtil.generateKey();
        String encryptedKey = EncryptionUtil.keyToString(key);
        stmt.setString(3, encryptedKey);
        stmt.executeUpdate();
        return true;
    }
}

```

Autentificarea utilizatorilor:

```
public boolean authenticateUser(String username, String passwordHash) {
    String query = "SELECT id, encryption_key FROM users WHERE username = ? AND password_hash = ?";
    try (Connection conn = Database.getConnection();
        PreparedStatement stmt = conn.prepareStatement(query)) {
        stmt.setString(1, username);
        stmt.setString(2, passwordHash);
        ResultSet rs = stmt.executeQuery();
        return rs.next();
    }
```

Obținerea cheii de criptare a utilizatorului:

```
public SecretKey getUserEncryptionKey(int userId) throws SQLException, Exception {
    String encryptionKey = null;
    String query = "SELECT encryption_key FROM users WHERE id = ?";

    try (Connection conn = Database.getConnection();
        PreparedStatement ps = conn.prepareStatement(query)) {
        ps.setInt(1, userId);
        try (ResultSet rs = ps.executeQuery()) {
            if (rs.next()) {
                encryptionKey = rs.getString("encryption_key");
            }
        }
    }
    return EncryptionUtil.getKeyFromString(encryptionKey);
}
```

2. Detalii de implementare

2.1 Proiectarea bazei de date

Modelul conceptual

Modelul bazei de date include următoarele entități:

1. **Utilizatori:** conțin date despre utilizatorii aplicației.
 - Atribute: ID, nume utilizator, hash parolă, cheie de criptare.
 - Id, username, password_hash, encryption_key
2. **Parole:** conțin datele criptate ale parolelor utilizatorilor.
 - Atribute: ID, ID utilizator, nume site, nume utilizator pentru site, parolă criptată.
 - Id, user_id, site_name, username, password

Modelul logic

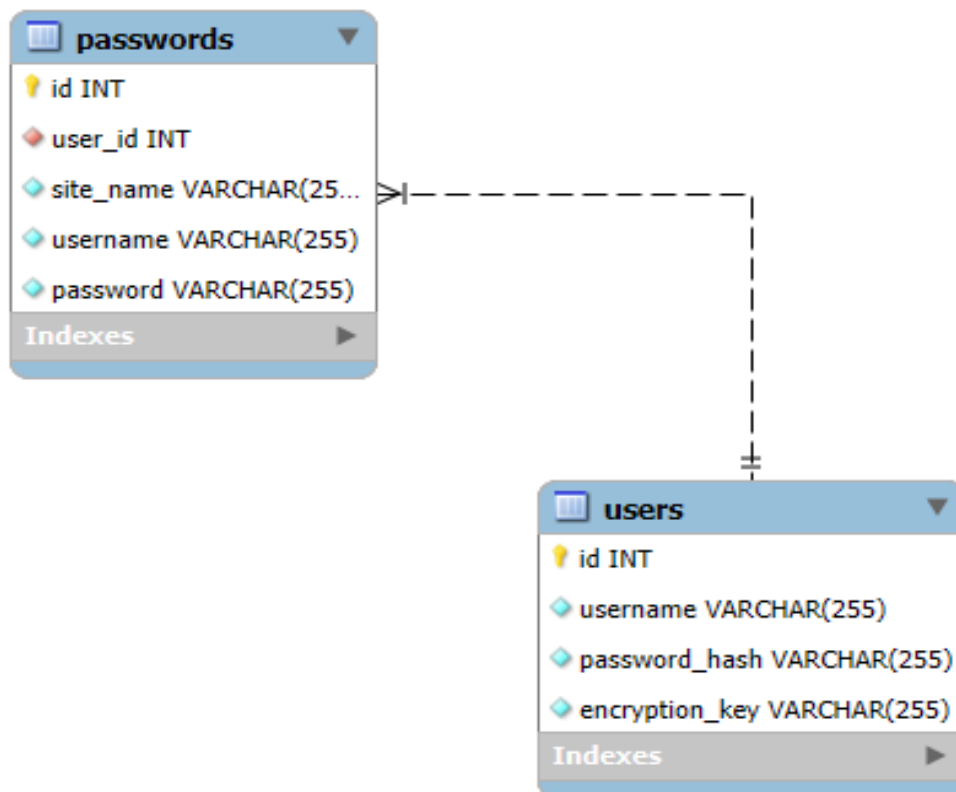
Tabelele principale sunt:

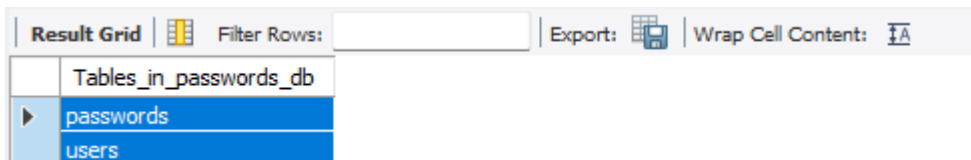
Utilizatori:

```
CREATE TABLE users (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  username VARCHAR(50) NOT NULL,  
  password_hash VARCHAR(256) NOT NULL,  
  encryption_key TEXT NOT NULL  
);
```

Parole:

```
CREATE TABLE passwords (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  user_id INT NOT NULL,  
  site_name VARCHAR(100) NOT NULL,  
  username VARCHAR(100),  
  password TEXT NOT NULL,  
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE  
);
```





Popularea bazei de date

Exemplu de înregistrări inițiale:

```
INSERT INTO users (username, password_hash, encryption_key) VALUES  
( 'test_user', 'hash_exemplu', 'cheie_criptare_exemplu');
```

```
INSERT INTO passwords (user_id, site_name, username, password) VALUES  
(1, 'gmail.com', 'test@gmail.com', 'parola_criptată');
```

2.2 Modulele aplicației

Interfața grafică

Aplicația utilizează Swing pentru a oferi o interfață grafică prietenoasă utilizatorului. Exemple de componente:

- **LoginScreen.java**: Pagina principală de autentificare.
- **MainScreen.java**: Ecranul principal unde utilizatorul își poate gestiona parolele.

Criptarea datelor

Modulul de criptare utilizează algoritmul **AES** pentru a proteja parolele utilizatorilor. Principalele funcții implementate în clasa **EncryptionUtil** sunt:

- **encrypt(String plaintext, SecretKey key)**: CripTEAZĂ textul utilizând cheia AES.
- **decrypt(String encryptedText, SecretKey key)**: DecripTEAZĂ textul criptat.
- **generateKey()**: Generează o cheie AES aleatorie.

Conexiunea cu baza de date

Conexiunea cu baza de date este gestionată de clasa **Database**, care folosește JDBC pentru conectare:

```
public static Connection getConnection() throws SQLException {  
    return DriverManager.getConnection(  
        "jdbc:mysql://localhost:3306/passwords_db", "app_user", "app_password");  
}
```

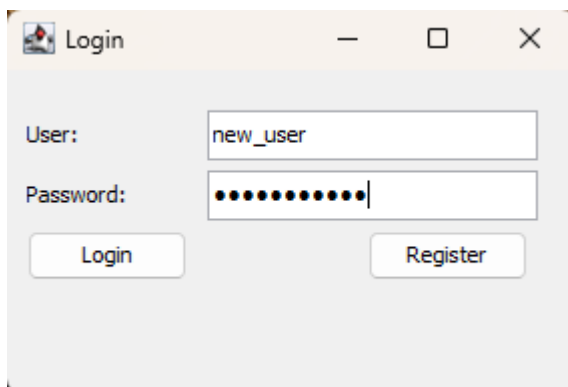

2.3 Funcționalități implementate

Funcționalități principale

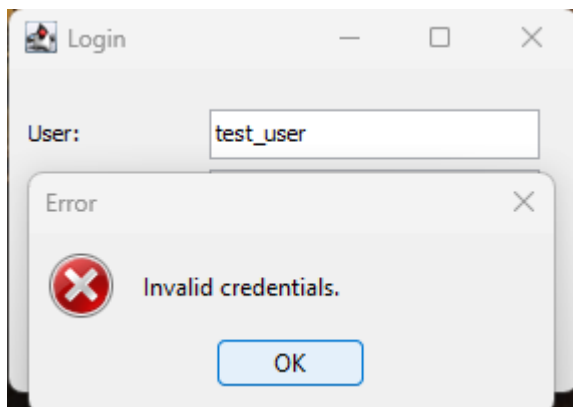
1. **Autentificare și înregistrare utilizatori**
 - Implementate în clasele `LoginScreen` și `UserDAO`.
 - Verificarea datelor de autentificare se realizează pe baza hash-ului parolei.
2. **Gestionarea parolelor**
 - Adăugare: Realizată prin intermediul clasei `AddPasswordDialog`.
 - Vizualizare și ștergere: Implementate în `MainScreen`.
3. **Criptare și decriptare parole**
 - Parolele sunt criptate înainte de a fi stocate în baza de date și decriptate la cerere.

3. Utilizarea aplicației

3.1 Pagina Login

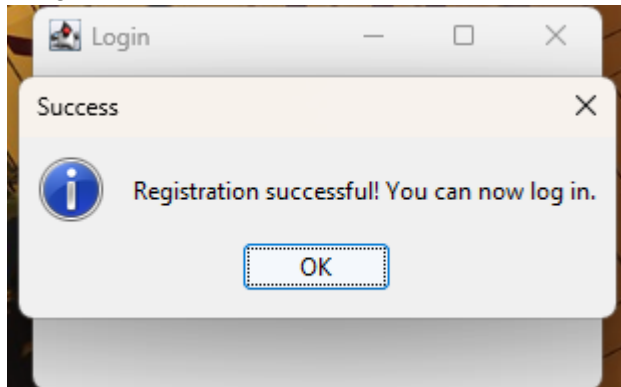


Autentificare esuata dacă datele nu sunt existente în baza de date.

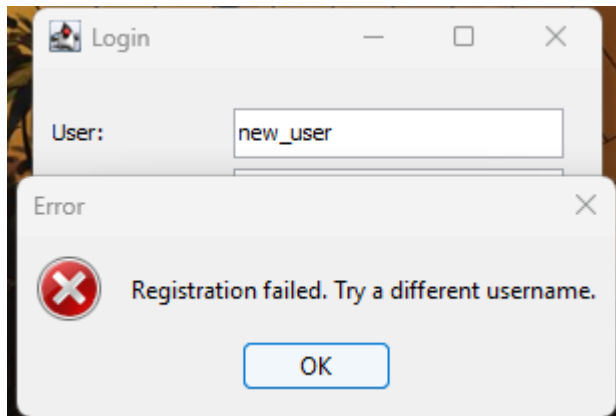


3.2 Functia Register

Înregistrarea unui utilizator nou.

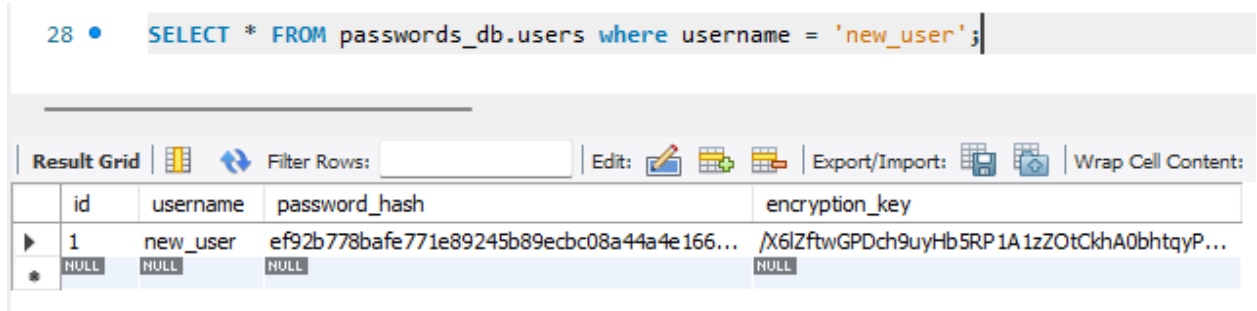


Inregistrare esuata, în cazul în care datele sunt deja existente în baza de date.



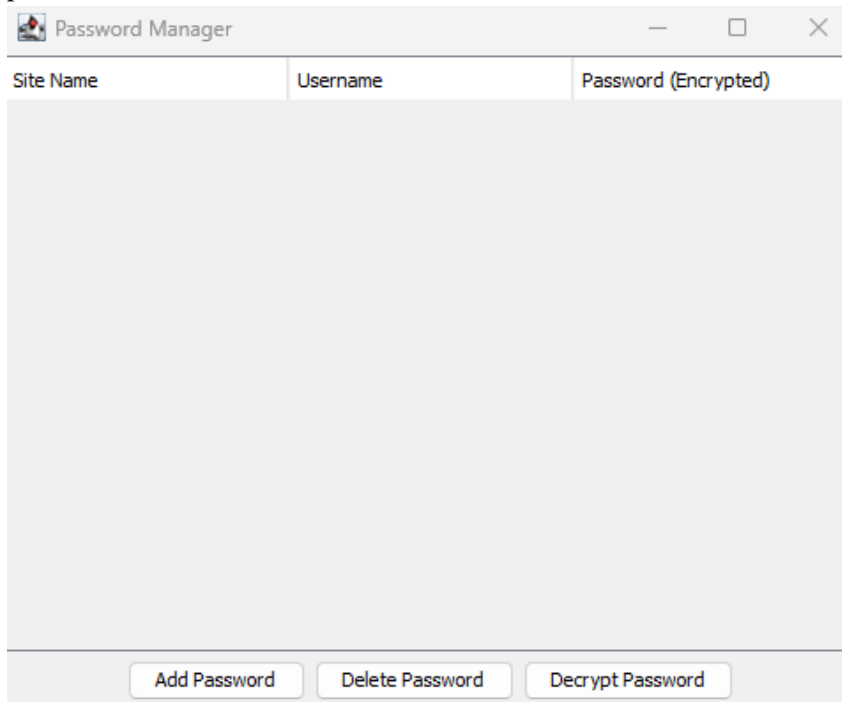
3.3 Popularea bazei de date - users

După înregistrarea cu succes în interfața de “Register”, datele apar în baza de date.

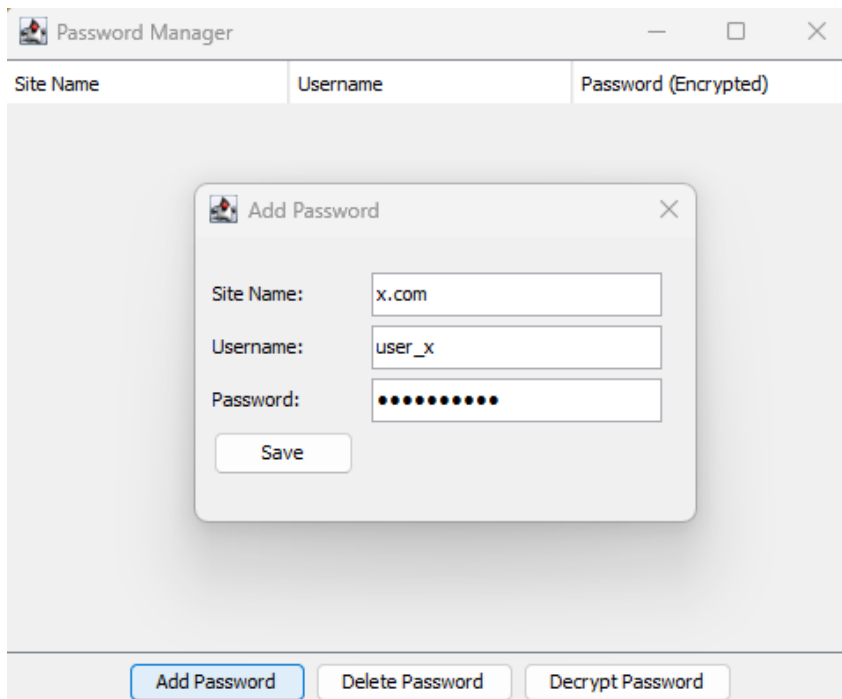


3.3 Pagina Main

În urma procesului de autentificare, utilizatorul are acces la pagina principala de gestionare a parolelor.



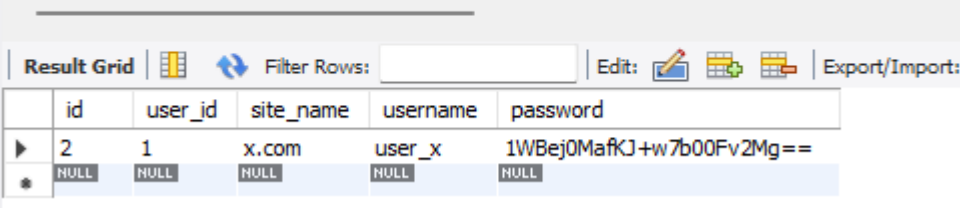
3.4 Adaugarea unei parole



3.5 Popularea bazei de date - passwords

În urma procesului de adaugare a contului și parolei, parola este encryptata cu cheia user-ului din baza de date si este stocată la randul ei în baza de date.

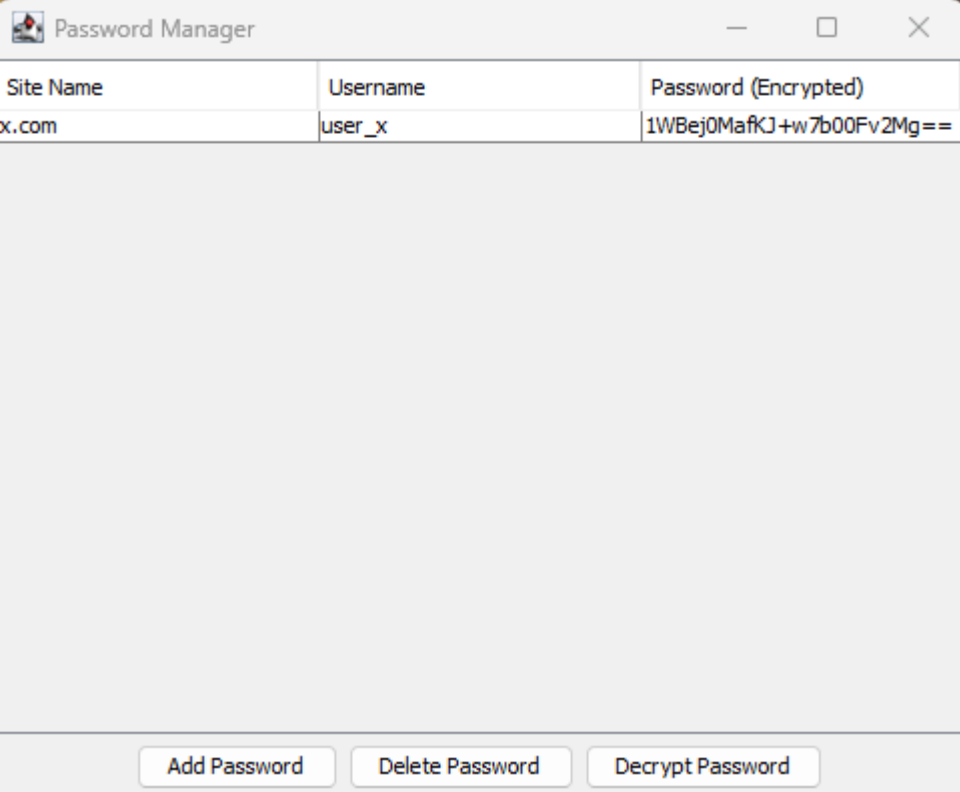
```
28 • SELECT * FROM passwords_db.passwords WHERE user_id = 1;
```



The screenshot shows a database interface with a toolbar containing icons for 'Result Grid', 'Filter Rows', 'Edit', and 'Export/Import'. Below the toolbar is a table with the following data:

	id	user_id	site_name	username	password
▶	2	1	x.com	user_x	1WBej0MafKJ+w7b00Fv2Mg==
*	NULL	NULL	NULL	NULL	NULL

Aceasta apare in interfata in forma ei encryptata.



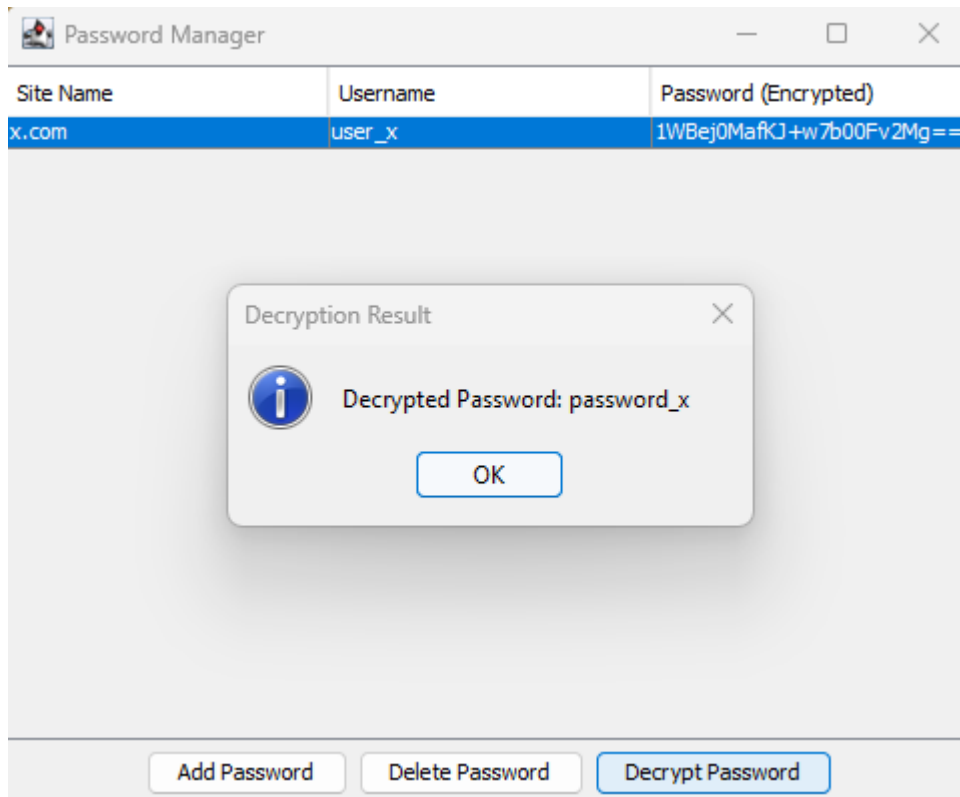
The screenshot shows a window titled 'Password Manager' with a table containing the following data:

Site Name	Username	Password (Encrypted)
x.com	user_x	1WBej0MafKJ+w7b00Fv2Mg==

At the bottom of the window, there are three buttons: 'Add Password', 'Delete Password', and 'Decrypt Password'.

3.6 Decriptarea parolelor

Se realizeaza prin selectarea liniei corespunzătoare contului dorit în interfața cu mouse-ul și apăsarea butonului “**Decrypt Password**”.



3.7 Stergerea parolelor

Se realizeaza prin selectarea liniei corespunzătoare contului dorit în interfața cu mouse-ul și apăsarea butonului “**Delete Password**”.

După procesul de ștergere putem verifica baza de date, observand ca parola nu mai exista.

```
28 • SELECT * FROM passwords_db.passwords WHERE user_id = 1;
```

The screenshot shows a database query result grid. The grid has five columns: id, user_id, site_name, username, and password. The first row is highlighted in blue and contains the values 'id', 'user_id', 'site_name', 'username', and 'password'. Below the table, there is a row with five 'NULL' values. The table is titled 'Result Grid' and has a 'Filter Rows' field. There are also buttons for 'Edit', 'Export/Import', and 'Import/Export'.

id	user_id	site_name	username	password
NULL	NULL	NULL	NULL	NULL

4. Concluzii și perspective de dezvoltare

Aplicația dezvoltată oferă o soluție simplă și eficientă pentru gestionarea parolelor în mod securizat.

În viitor, aplicația ar putea fi extinsă prin:

- Adăugarea suportului pentru autentificare cu doi factori.
- Implementarea unei aplicații mobile.
- Integrarea cu un serviciu de backup securizat pentru stocarea datelor criptate.