

UNIVESITATEA TEHNICĂ ,,GHEORGE ASACHI” IAȘI

INGINERIA PROGRAMĂRII

DOCUMENTAȚIE

TITLUL LUCRĂRII: Aplicație de închiriere motociclete - Rent-a-Moto

AUTORI:

George-Laurențiu APETRIA

Cristian-Stelian GRECU

Raul-Emilian MIRON

Simona BACIU

GRUPELE:1305A,1308B

IAȘI 2021

Contents

[Documentul specificațiilor cerințelor (SRS) 3](#_Toc72786367)

[1. Introducere 3](#_Toc72786368)

[1.1. Scop 3](#_Toc72786369)

[1.2. Obiective 3](#_Toc72786370)

[1.3. Overview 3](#_Toc72786371)

[2. Descriere Generală 3](#_Toc72786372)

[1.4. Perspectiva Produsului 3](#_Toc72786373)

[1.5. Funcțiile Produsului 3](#_Toc72786374)

[1.6. Clase Utilizator și Caracteristici 4](#_Toc72786375)

[1.7. Design și constrângeri de implementare 4](#_Toc72786376)

[3. Cerințe Specifice 4](#_Toc72786377)

[1.8. Cerințe Interfață Exterioară 4](#_Toc72786378)

[1.9. Cerințe Funcționale 4](#_Toc72786379)

[1.10. Cerințe de Performanță 6](#_Toc72786380)

[1.11. Constrângeri de Design 6](#_Toc72786381)

[1.12. Atribute 7](#_Toc72786382)

[1.13. Design pattern-uri folosite 7](#_Toc72786383)

[Diagrame UML 8](#_Toc72786384)

[**1.** Diagrama de clase 8](#_Toc72786385)

[**2.** Diagrama de activitate 9](#_Toc72786386)

[Modul de utilizare al programului 11](#_Toc72786387)

[1.Introducere 11](#_Toc72786388)

[2.Sistemul de login 11](#_Toc72786389)

[3. Închirierea unei motociclete 12](#_Toc72786390)

[4. Utilizatorii 15](#_Toc72786391)

[5. Dashboard 16](#_Toc72786392)

[Anexă 17](#_Toc72786393)

[6. Singleton – SQLCommand 17](#_Toc72786394)

[7. Proxy – Login 19](#_Toc72786395)

[8. Unit Testing 20](#_Toc72786396)

[9. Delete a rent 20](#_Toc72786397)

# Documentul specificațiilor cerințelor (SRS)

## Introducere

### Scop

Scopul documentației realizate este de a furniza o descriere detaliată a cerințelor proiectului Rent-a-Moto. Aceasta va oferi informații esențiale în legătură cu scopul și caracteristicile, interfața și funcționalitățile interne ale aplicației.

### Obiective

Rent-a-Moto este o aplicație utilizată în scopul închirierii motocicletelor, având o bază de date locală asociată. Aceasta furnizează o serie de funcționalități: adăugarea, ștergerea și modificarea motocicletelor, clienților și utilizatorilor, precum și păstrarea istoricului închirierilor. După popularea tabelelor cu datele necesare, poate începe procesul de închiriere în care se vor putea selecta diferite opțiuni legate de motocicleta care urmează să fie închiriată, clientul responsabil și perioada de închiriere.

Interfața aplicației are un aspect prietenos în raport cu utilizitatorii și este ușor de folosit.

### Overview

Următorul capitol, ,,Descrierea Generală” a acestui document, oferă o perspectivă în legătură cu funcționalitatea aplicației. Acesta descrie caracteristicile așteptate și principalele constrângeri. Cel de-al treilea capitol, ,,Cerințele Specifice” realizează o descriere tehnică a detaliilor de funcționare ale aplicației și prezintă interfața grafică.

## Descriere Generală

### Perspectiva Produsului

Rent-a-Moto este o aplicație de sine stătătoare care permite ținerea unei evidențe asupra disponibilității de închiriere a unor motociclete. Aceasta a fost scrisă în limbajul C# și este suportată de sistemul de operare Windows. Programul poate fi utilizat de către orice firmă care deține o serie de motociclete destinate închirierii pentru a avea o evidență în timp real a motocicletelor care sunt disponibile și a celor care sunt blocate la momentul respectiv.

Aplicația a fost implementată în formatul Windows Forms Application și utilizează un singur fir de execuție.

### Funcțiile Produsului

Aplicația Rent-a-Moto furnizează următoarele funcții pentru utilizatori:

* + 1. Adăugarea, ștergerea și editarea de motociclete în/din baza de date.
    2. Adăugarea, ștergerea și editarea de utilizatori în/din baza de date.
    3. Adăugarea, ștergerea și editarea de custom-eri în/din baza de date.
    4. Adăugarea și ștergerea unei noi închirieri în aplicație pe o anumită perioadă de timp.
    5. Posibilitatea de a verifica în timp real lotul de motociclete disponibile pentru închiriere, precum și numărul de utilizatori ai aplicației și numărul clienților.

### Clase Utilizator și Caracteristici

Rent-a-Moto a fost concepută astfel încât să fie ușor de utilizat, în special pentru persoane care nu au mai folosit înainte aplicații de manageriere a închirierilor în timp real. Mai mult decât atât, aceasta furnizează funcționlitățile unei aplicații obișnuite utilizate deja pe piață, însă mai pot fi adăugate și alte opțiuni particulare, cum ar fi de exemplu un istoric al închirierilor deja efectuate. Dat fiind caracterul didactic al proiectului și intenția de a nu supraîncărca întregul proces de implementare, am decis ca aplicația să cuprindă doar funcționalitățile anterior prezentate, însă să rămână deschisă integrării unor module noi.

### Design și constrângeri de implementare

Cea mai mare constrângere la nivelul implementării este fără doar și poate timpul necesar dezvoltării la care se adaugă și constrângerile cauzate de mediul de dezvoltare. Implementarea întregului proiect (alegerea temei, stabilirea design-ului, dezvoltarea codului, precum și documentarea acestuia) a reprezentat o adevărată provocare în sensul utilizării tehnologiilor actuale, însă parcurgerea pas cu pas a fiecărei etape a reușit să genereze experiența necesară și energia de a finaliza aplicația.

## Cerințe Specifice

### Cerințe Interfață Exterioară

#### Interfața Utilizator

Secțiunea curentă prezintă funcționalitățile și constrângerile interfeței grafice cu utilizatorul (GUI).

* Inițial, la startarea aplicației apare meniul de logare în care utilizatorul trebuie să introducă în TextBox-urile corespunzătoare credențialele.
* După introducerea corectă a datelor necesare pentru logare apar o serie de meniuri, unul dintre ele fiind pentru adăugarea user-ilor, iar unul asemănător este pentru adăugarea customer-ilor.
* Un alt meniu este destinat adăugării unor noi motociclete destinate închirierii.
* Meniul de închiriere propriu-zisă permite introducerea unor date de identificare pentru user, customer, motocicletă și intervalul de timp pentru care motocicleta va fi blocată.

### Cerințe Funcționale

Această secțiune exemplifică funcționalitățile aplicației împreună cu informații despre fiecare caracteristică ce poate fi observată de utilizator.

#### Interfața de login

Utilizatorul își introduce credențialele în două textbox-uri destinate pentru un nume de utilizator și parolă. Există posibilitatea de a apăsa pe butonul CLEAR pentru a elimina conținutul din textbox-uri, iar la apăsarea butonului de login se verifică în baza de date locală existența acestor credențiale prin intermediul unui DLL ce implementează un Proxy. În caz de succes, utilizatorul are acces la meniul principal. În caz contrar, apare un mesaj de eroare ce atenționează asupra datelor eronate introduse.

* + 1. Pagina principală

Pagina principală este constituită din butonul de LOGOUT, butonul de HELP pentru afișarea help-ului specific și câte un buton pentru accesarea informațiilor referitoare la fiecare dintre cele cinci secțiuni: motocicletele, clienții, utilizatorii, închirierile și dashboard-ul. De asemenea, fiecare dintre paginile care pot fi accesate din meniul principal, conțin butoanele de EXIT și BACK. Fiecare dintre acestea, exceptând pagina de dashboard, conține un tabel care se actualizează în timp real conform conținutului tabelelor din baza de date locală. La selectarea unei înregistrări din tabela din interfața grafică se autocompletează anumite câmpuri din formular pentru o mai bună vizualizare.

* + 1. Pagina cu motociclete

Conținutul paginii destinate motocicletelor este format dintr-un formular ce conține următoarele câmpuri: numărul de registru, brand-ul, modeul, prețul și disponibilitatea. Utilizatorul are posibilitatea de a adăuga sau șterge o motocicletă din baza de date sau de a edita informațiile referitoare la aceasta. În caz de câmpuri nule, se aruncă o excepție.

* + 1. Pagina de clienți

Conținutul paginii destinate clienților este format dintr-un formular ce conține următoarele câmpuri: id-ul unic, numele, adresa și telefonul. Clientul are posibilitatea de a adăuga sau șterge un client din baza de date sau de a edita informațiile referitoare la acesta. În caz de câmpuri nule, se aruncă o excepție.

* + 1. Pagina de utilizatori

Conținutul paginii destinate utilizatorilor este format dintr-un formular ce conține următoarele câmpuri: id-ul unic, numele de user și parola. Utilizatorul are posibilitatea de a adăuga sau șterge un user din baza de date sau de a edita informațiile referitoare la acesta. În caz de câmpuri nule, se aruncă o excepție.

* + 1. Pagina de închirieri

Formularul asociat are un id unic, un câmp de selectat din motocicletele disponibile (cele care nu au fost încă închiriate), un câmp cu id-urile clienților, un câmp care se autocompletează cu numele clientului corespunzător id-ului selectat și două câmpuri pentru data de închiriere, respectiv returnare. La introducerea unei noi închirieri, câmpul ,,disponibilitate” al motocicletei selectate se schimbă astefel încât se evită situația ca o motocicetă să fie inchiriată de două ori.

* + 1. Pagina de dashboard

Această pagină actualizează în timp real numărul de motociclete, de clienți și de utilizatori, având doar scop informativ.

* + 1. Pagina de Help

La apăsarea butonului de HELP din meniul principal, se deschide help-ul aferent programului.

### Cerințe de Performanță

Această secțiune acoperă cerințele ce se referă la performanța aplicației ca răspuns la interacțiunea cu utilizatorul.

* + 1. Utilizarea unui meniu

Interfața trebuie să conțină un meniu care facilitează accesul la celelalte secțiuni.

* + 1. Vizualizarea tabelelor în interfața grafică

Fiecare pagină conține un tabel ce se actualizează după efectuarea operațiilor specifice.

* + 1. Utilizarea butoanelor pentru a se efectua o operație specifică

Fiecare pagină reunește un set de butoane intuitive care realizează operații folosind șablonul de proiectare Singleton pentru o performață crescută și un spațiu de memorie mai redus.

* + 1. Autocompletarea câmpurilor

Aplicația este prevăzută cu un sistem de autocompletare pentru a scurta timpul de efectuare a unei operații în baza de date locală.

* + 1. Interacțiunea între componentele sistemului

Conexiunea dintre componentele sistemului este constituită din accesul la baza de date locală asociată și tabele destinate afișării în timp real.

### Constrângeri de Design

Secțiunea prezentă expune limitările impuse de către aplicație, din punct de vedere al caracteristicilor hardware și software ale mediului de lucru.

1. Utilizarea spațiului pe disc

Totalitatea modulelor aplicației nu va depăși capacitatea de stocare de 10 MB pe disc. Acest lucru este un mare avantaj, făcând posibilă execuția aplicației de pe orice dispozitiv la momentul curent.

1. Utilizarea memorie

Necesarul de memorie RAM al aplicației este raportat la un maxim de 10 MB. În mod normal, aplicația nu va depăși limita de 8 MB RAM.

### Atribute

Secțiunea curentă prezintă atributele aplicației, metricele și cerințele acestora.

1. Fiabilitatea sistemului

Fiabilitatea face referire la capacitatea sistemului de a răspunde corect cerințelor în baza datelor de intrare furnizate de către utilizator. Aplicația generează situația disponibilității lotului de închiriere în timp real, Datele sunt stocate corect, iar persistența este, de asemenea realizată în 100% din cazuri.

1. Mentenanță

Aplicația trebuie să fie ușor de întreținut, iar în viitor extinderea acesteia trebuie să fie facilă. De asemenea, testarea se va putea face într-o manieră simplă și noi funcționalități vor putea fi adăugate datorită carcterului deschis al programului pentru noi opțiuni.

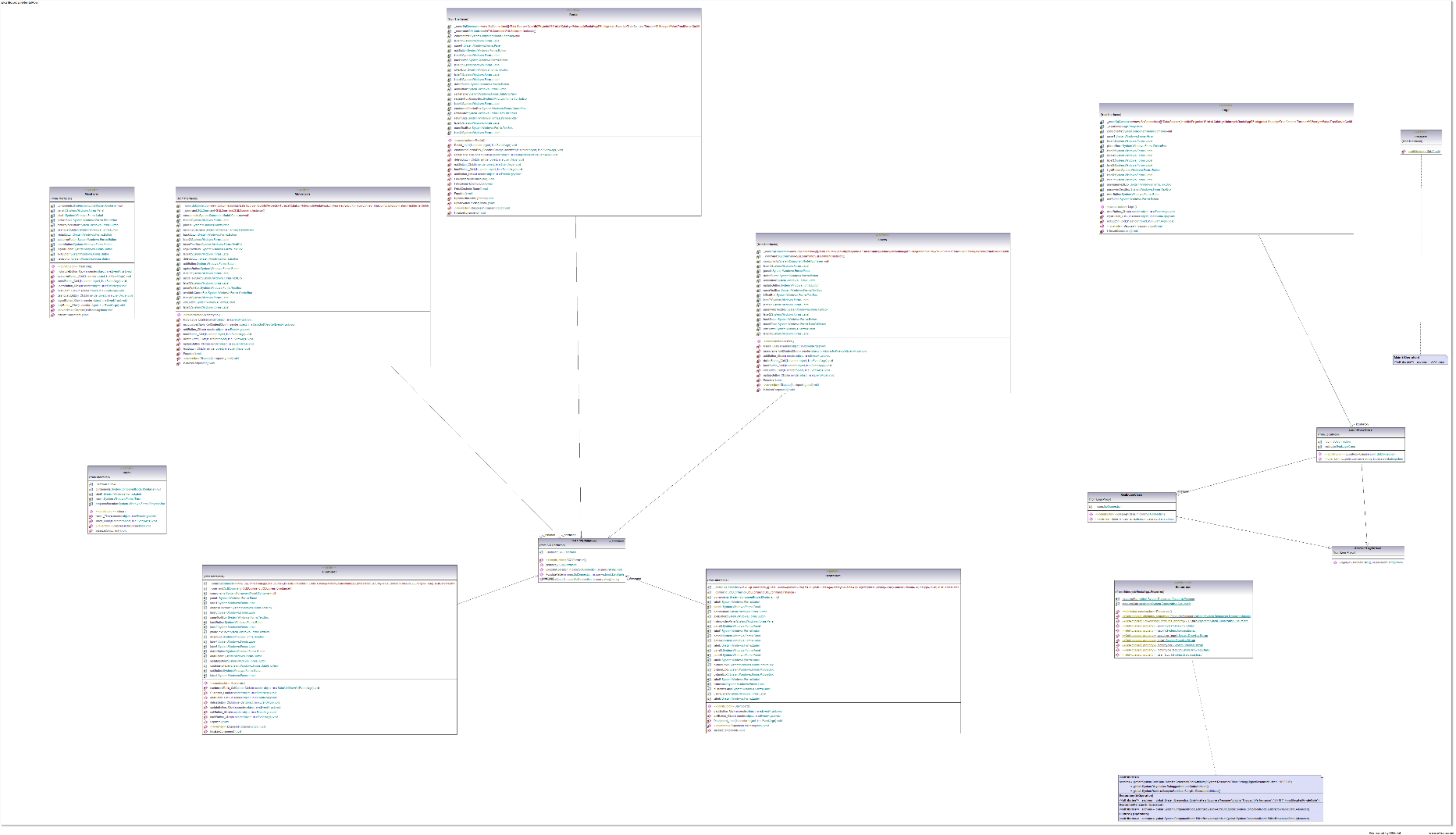
### Design pattern-uri folosite

Programul utilizează două design pattern-uri după cum urmează:

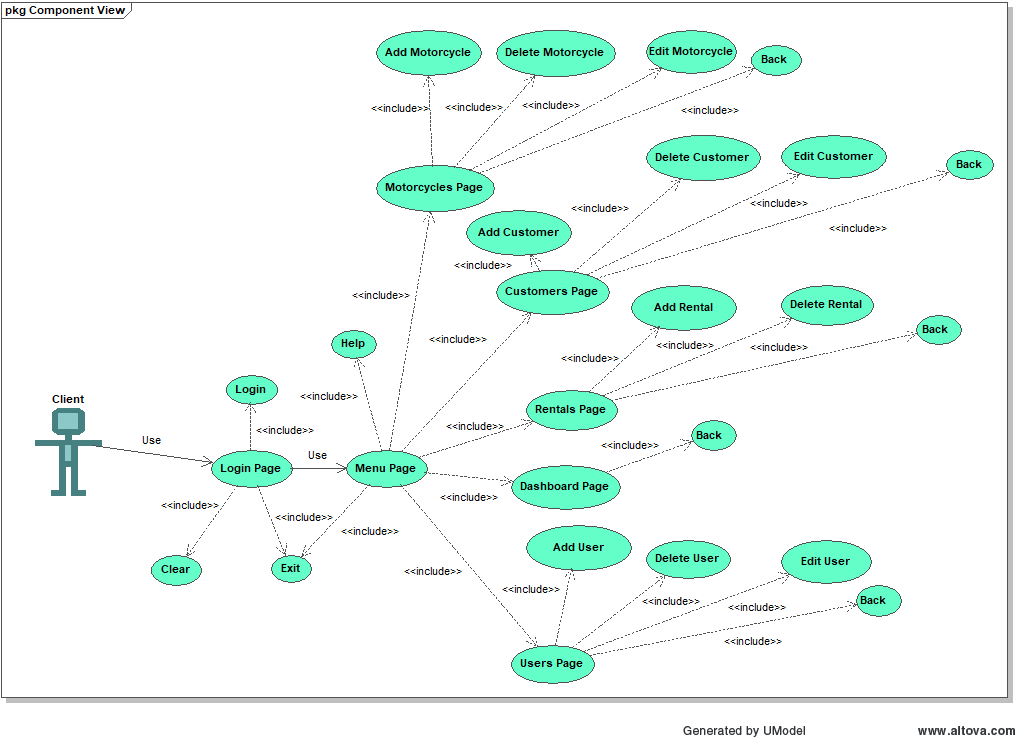
1. Singleton – este folosit pentru a se instanția în fiecare WindowsForm o singură entitate a unei comenzi SQL. Practic, comanda este prevăzută cu metode care previn apariția codului repetitiv.
2. Proxy – este folosit pentru sistemul de login pentru a se oferi un înlocuitor pentru clasa reală ce implementează verificarea existenței numei de utilizator și a parolei introduse în intefața grafică.

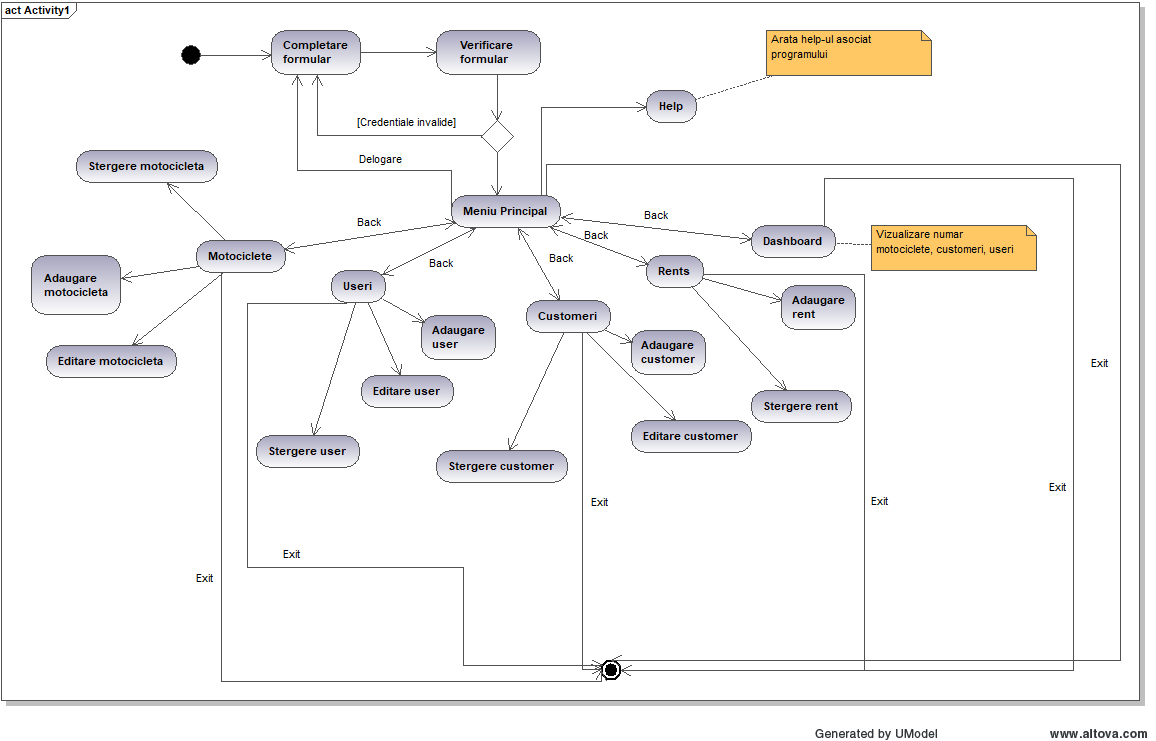
# Diagrame UML

1. Diagrama de clase



## Diagrama de activitate



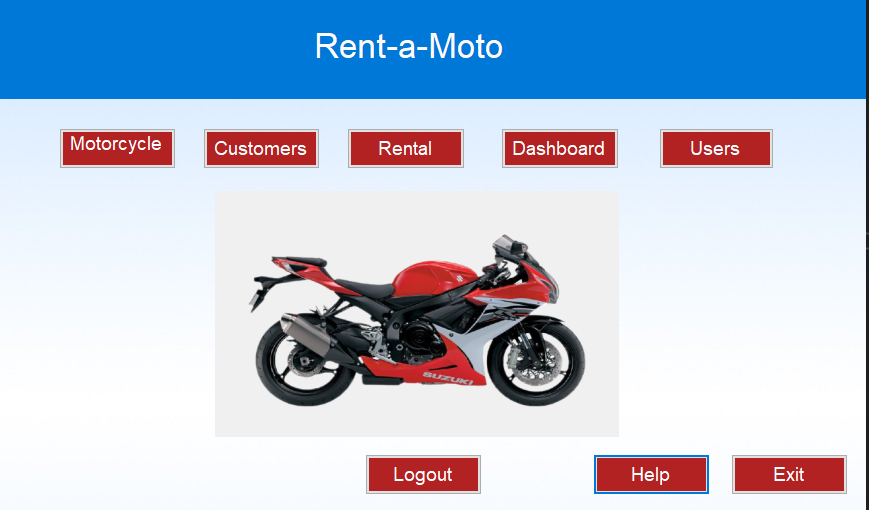


# Modul de utilizare al programului

## 1.Introducere

**Rent-a-Moto** este o aplicație de închiriat motociclete ce are atașată o bază de date pentru memorarea utilizatorilor, motocicletelor, clienților și a istoricului de închirieri.

Este o aplicație facilă oricărui utilizator cu o interfață prietenoasă și care nu prezintă dificultăți de navigare**.**

****

## 2.Sistemul de login

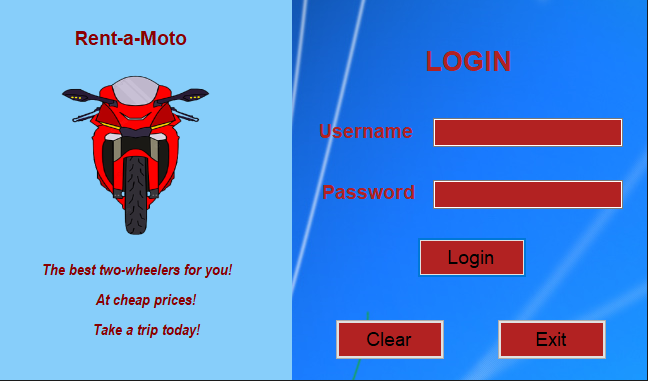
**Sistemul de login** este constituit dintr-o interfață în care utilizatorul introduce username-ul și parola pentru a se loga.

Opțiunile aferente sunt:

1. **Login** - Logare cu datele introduse.

2. **Clear** - Ștergere date introduse.

3. **Exit** - Se iese din aplicație.



## Închirierea unei motociclete

După completarea formularului aferent înregistrării unei motociclete, opțiunile sunt:

1. **Add** - Adaugă o nouă motocicletă.

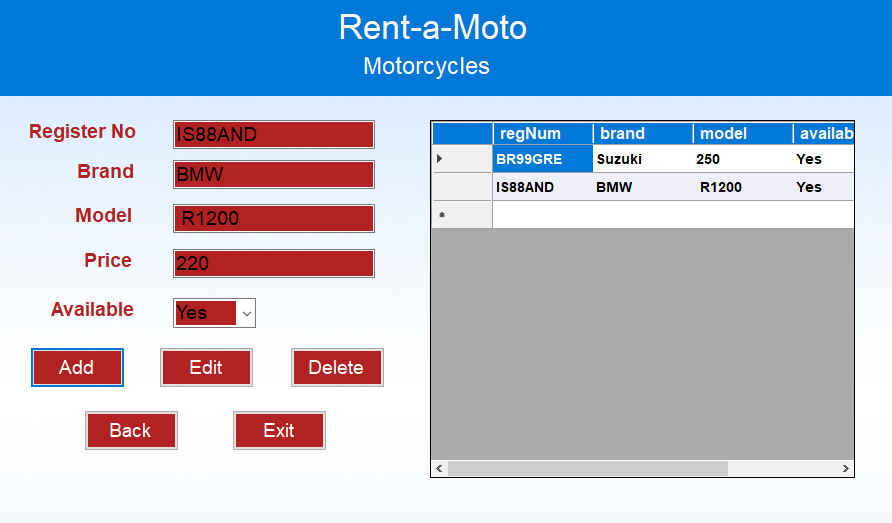
2. **Edit** - Editează informațiile despre o motocicletă.

3. **Delete** - Șterge o motocicletă din baza de date.

4. **Back** - Întoarcere la meniul principal.

5. **Exit** - Se iese din aplicatie.

La selectarea unei motociclete din tabel, se autocompletează formularul cu informații pentru a facilita editarea.



După completarea formularului aferent înregistrării unui client, optiunile sunt:

1. **Add** - Adaugă un nou client.

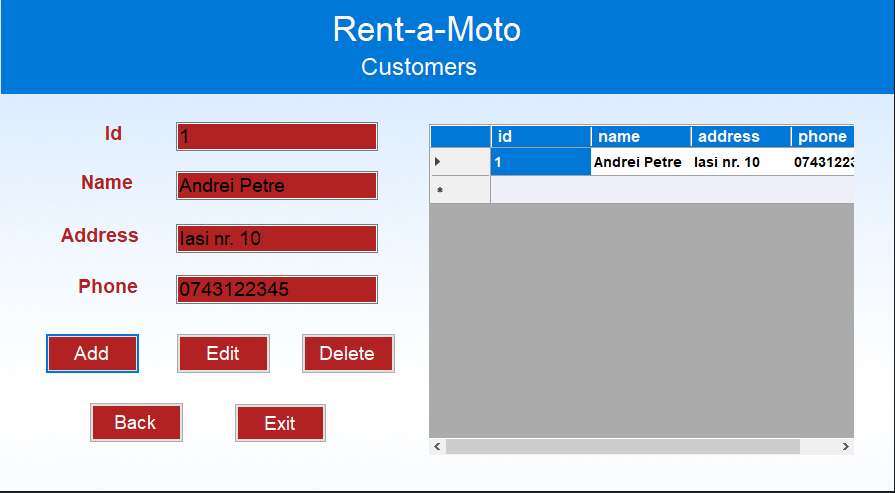
2. **Edit** - Editează informațiile despre un client.

3. **Delete** - Șterge un client din baza de date.

4. **Back** - Întoarcere la meniul principal.

5. **Exit** - Se iese din aplicație.

La selectarea unui client din tabel, se autocompletează formularul cu informații pentru a facilita editarea.



După completarea formularului aferent înregistrării unei închirieri, opțiunile sunt:

1. **Add** - Adaugă o nouă închiriere.

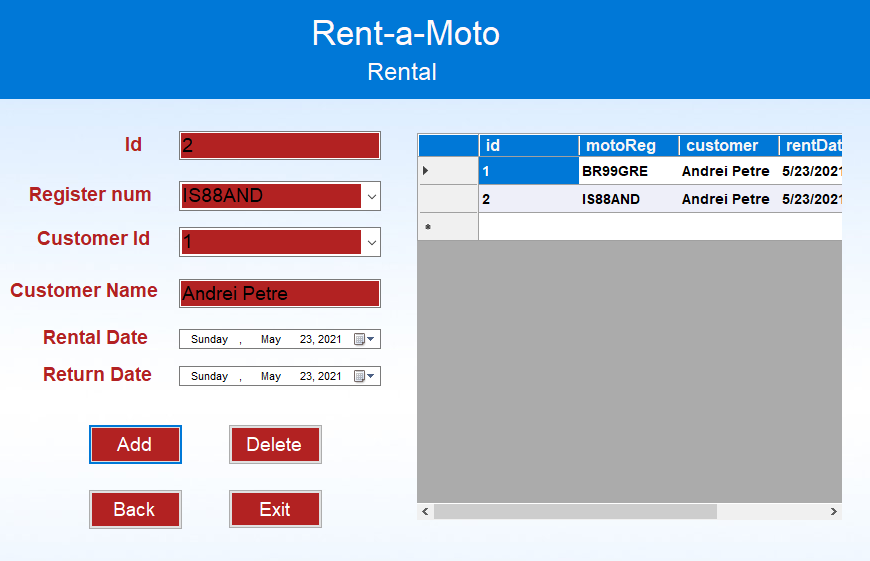
2. **Delete** - Șterge o închiriere din baza de date.

3. **Back** - Întoarcere la meniul principal.

4. **Exit** - Se iese din aplicație.

La selectarea unei închirieri din tabel, se autocompletează formularul cu informații pentru a facilita editarea.

La introducerea unei noi închirieri, se actualizează statusul motocicletei (disponibilă ---> indisponibilă) împreună cu combobox-urile aferente.



## Utilizatorii

După completarea formularului aferent înregistrarii unui utilizator, opțiunile sunt:

1. **Add** - Adaugă un nou utilizator.

2. **Edit** - Editează informațiile despre un utilizator.

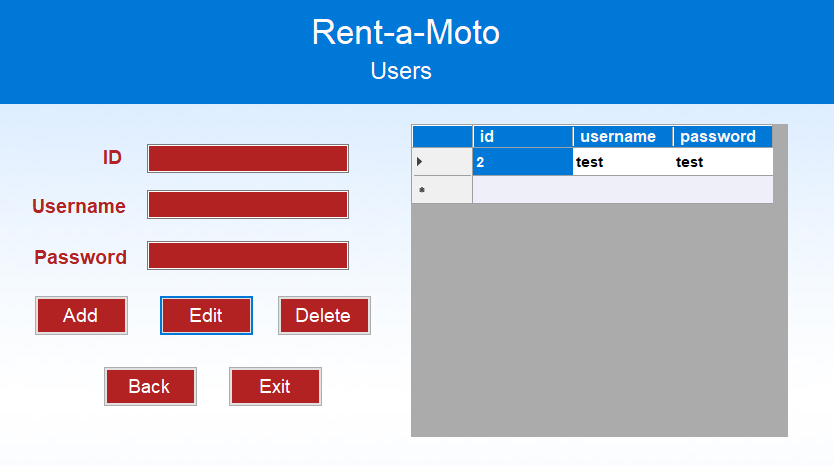
3. **Delete** - Șterge un utilizator din baza de date.

4. **Back** - Întoarcere la meniul principal.

5. **Exit** - Se iese din aplicație.

La selectarea unui utilizator din tabel, se autocompletează formularul cu informații pentru a facilita editarea.

În mod ideal, administratorul va avea acces la această interfață.



## Dashboard

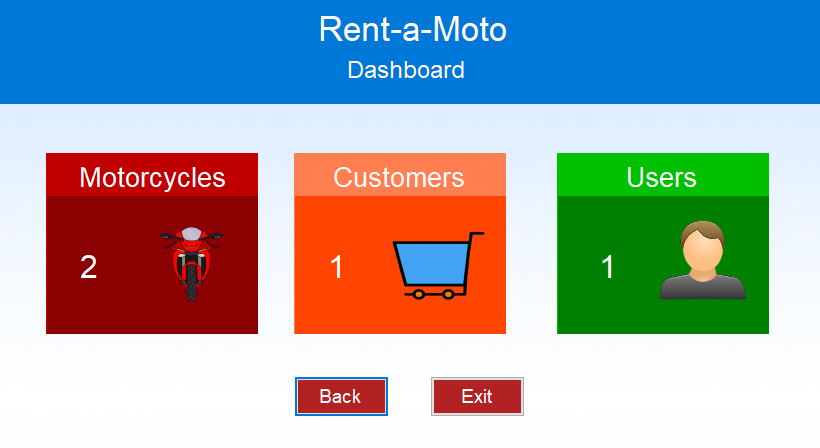
Aplicația folosește și un dashboard în care sunt afișate: numărul de motociclete, numărul de clienți și numărul de utilizatori.

Această interfață are un rol preponderent de informare și mai puțin de interactivitate.

Opțiunile disponibile sunt:

1. **Back** - Întoarcere la meniul principal.

2. **Exit** - Se iese din aplicație.



# Anexă

## Singleton – SQLCommand

namespace SQLCommand

{

/// <summary>

/// SQLCommand Singleton class.

/// </summary>

public class SQLCommand

{

#region Fields

/// <summary>

/// SQLCommand instance for Singleton.

/// </summary>

private static SQLCommand \_instance;

#endregion

#region Constructors

/// <summary>

/// Empty constructor for SQLCommand.

/// </summary>

private SQLCommand()

{

}

#endregion

#region Methods

/// <summary>

/// Lazy initialization of Singleton instance.

/// </summary>

/// <returns>Instance of class.</returns>

public static SQLCommand Instance()

{

if (\_instance == null)

\_instance = new SQLCommand();

return \_instance;

}

/// <summary>

/// Execute simple database operation:add,delete,update.

/// </summary>

/// <param name="conn">Conection to database.</param>

/// <param name="query">Specific operation query.</param>

public void ExecuteOperation(SqlConnection conn, string query)

{

conn.Open();

SqlCommand cmd = new SqlCommand(query, conn);

cmd.ExecuteNonQuery();

conn.Close();

}

/// <summary>

/// Retrieve data from database to be displayed in UI table.

/// </summary>

/// <param name="conn">Conection to database.</param>

/// <param name="query">Specific operation query.</param>

/// <returns>Dataset to be displayed in UI table.</returns>

public DataTable PopulateTable(SqlConnection conn,string query)

{

conn.Open();

SqlDataAdapter adapter = new SqlDataAdapter(query, conn);

SqlCommandBuilder builder = new SqlCommandBuilder(adapter);

var dataset = new DataSet();

adapter.Fill(dataset);

conn.Close();

return dataset.Tables[0];

}

/// <summary>

/// Count the rows in a table from database to display in Dashboard Form.

/// </summary>

/// <param name="conn">Conection to database.</param>

/// <param name="query">Specific operation query.</param>

/// <returns>Count of rows in a specific table.</returns>

public string RetrieveCount(SqlConnection conn,string query)

{

SqlDataAdapter adapter = new SqlDataAdapter(query, conn);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

return dataTable.Rows[0][0].ToString();

}

#endregion

}

}

## Proxy – Login

namespace LoginProxy

{

/// <summary>

/// Real class for Login.

/// </summary>

public class RealLoginClass : AbstractLoginClass

{

#region Fields

/// <summary>

/// Connection to database.

/// </summary>

private SqlConnection \_conn;

#endregion

#region Constructors

/// <summary>

/// Constructor of RealLoginClass.

/// </summary>

/// <param name="conn"></param>

public RealLoginClass(SqlConnection conn)

{

\_conn = conn;

}

#endregion

#region Methods

/// <summary>

/// Verify in database if the pair (username,password) exists.

/// </summary>

/// <param name="username">Username entered in UI.</param>

/// <param name="password">Password entered in UI.</param>

/// <returns>Bool variable which represents if login is successful or not.</returns>

public override bool Login(string username,string password)

{

string query = "select count(\*) from [User] where username='" + username + "' and password='" + password + "';";

\_conn.Open();

SqlDataAdapter adapter = new SqlDataAdapter(query, \_conn);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

\_conn.Close();

if (dataTable.Rows[0][0].ToString() != "0")

{

return true;

}

else

{

return false;

}

}

#endregion

}

}

## Unit Testing

public void TestLogin\_1()

{

LoginProxyClass proxy = new LoginProxyClass(\_conn);

Assert.AreEqual(false, proxy.Login("", ""));

}

[TestMethod]

public void TestLogin\_2()

{

LoginProxyClass proxy = new LoginProxyClass(\_conn);

Assert.AreEqual(true, proxy.Login("admin", "admin"));

}

[TestMethod]

public void TestLogin\_3()

{

LoginProxyClass proxy = new LoginProxyClass(\_conn);

Assert.AreEqual(false, proxy.Login("admin", ""));

}

## Delete a rent

private void deleteButton\_Click(object sender, EventArgs e)

{

if (idTextBox.Text == "")

{

MessageBox.Show("Missing information!", "Error");

}

else

{

try

{

string query = "delete from [Rental] where id=" + idTextBox.Text + ";";

\_command.ExecuteOperation(\_conn, query);

MessageBox.Show("Rental deleted!", "Notification");

Populate();

UpdateAfterDeletingRental();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}