МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технологический университет «СТАНКИН»
(ФГБОУ ВО «МГТУ «СТАНКИН»)

Институт информационных систем и технологий

Кафедра информационных технологий и вычислительных систем

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ №1

ПО ДИСЦИПЛИНЕ

«Защита информации»

СТУДЕНТА 4 КУРСА Бакалавриата ГРУППЫ ИДБ-20-02

Ердогана Дениза Ердаловича

НА ТЕМУ

Разработка программы разграничения полномочий пользователей на основе парольной аутентификации

Вариант № 8

Направление:	09.03.01 Информатика и вычислительная техника
Профиль подготовки:	«Программное обеспечение средств вычислительной техники и автоматизированных систем»

Отчет сдан «_____» _____ 2023 г.

Оценка _____

Преподаватель Шевляков К.А

(подпись)

МОСКВА 2023

Задание

1. Программа должна обеспечивать работу в двух режимах: администратора (пользователя с фиксированным именем ADMIN) и обычного пользователя.

2. В режиме администратора программа должна поддерживать следующие функции (при правильном вводе пароля):

- смена пароля администратора (при правильном вводе старого пароля);
- просмотр списка имен зарегистрированных пользователей и установленных для них параметров (блокировка учетной записи, включение ограничений на выбираемые пароли) – всего списка целиком в одном окне или по одному элементу списка с возможностью перемещения к его началу или концу;
- добавление уникального имени нового пользователя к списку с пустым паролем (строкой нулевой длины);
- блокирование возможности работы пользователя с заданным именем;
- включение или отключение ограничений на выбираемые пользователем пароли (в соответствии с индивидуальным заданием, определяемым номером варианта);
- завершение работы с программой.

3. В режиме обычного пользователя программа должна поддерживать только функции смены пароля пользователя (при правильном вводе старого пароля) и завершения работы, а все остальные функции должны быть заблокированы.

4. После своего запуска программа должна запрашивать у пользователя в специальном окне входа ввод его имени и пароля. При вводе пароля его символы всегда должны на экране заменяться символом ‘*’.

5. При отсутствии введенного в окне входа имени пользователя в списке

зарегистрированных администратором пользователей программа должна выдавать соответствующее сообщение и предоставлять пользователю возможность повторного ввода имени или завершения работы с программой.

6. При неправильном вводе пароля программа должна выдавать соответствующее

сообщение и предоставлять пользователю возможность повторного ввода.

При трехкратном вводе неверного пароля работа программы должна завершаться.

7. При первоначальном вводе пароля (обязательном при первом входе администратора или пользователя с зарегистрированным ранее администратором именем) и при дальнейшей замене пароля программа должна просить пользователя подтвердить введенный пароль путем его повторного ввода.

8. Если выбранный пользователем пароль не соответствует требуемым ограничениям (при установке соответствующего параметра учетной записи пользователя), то программа должна выдавать соответствующее сообщение и предоставлять пользователю возможность ввода другого пароля, завершения работы с программой (при первом входе данного пользователя) или отказа от смены пароля.

9. Информация о зарегистрированных пользователях, их паролях, отсутствии

блокировки их работы с программой, а также включении или отключении

ограничений на выбираемые пароли должна сохраняться в специальном файле. При первом запуске программы этот файл должен создаваться автоматически и содержать информацию только об администраторе, имеющем пустой пароль.

10. Интерфейс с программой должен быть организован на основе меню, обязательной частью которого должно являться подменю «Справка» с

командой «О программе». При выборе этой команды должна выдаваться информация об авторе программы и выданном индивидуальном задании. Интерфейс пользователя программы может также включать панель управления с дублирующими команды меню графическими кнопками и строку состояния.

11. Для реализации указанных в пунктах 2-3 функций в программе должны использоваться специальные диалоговые формы, позволяющие пользователю (администратору) вводить необходимую информацию.

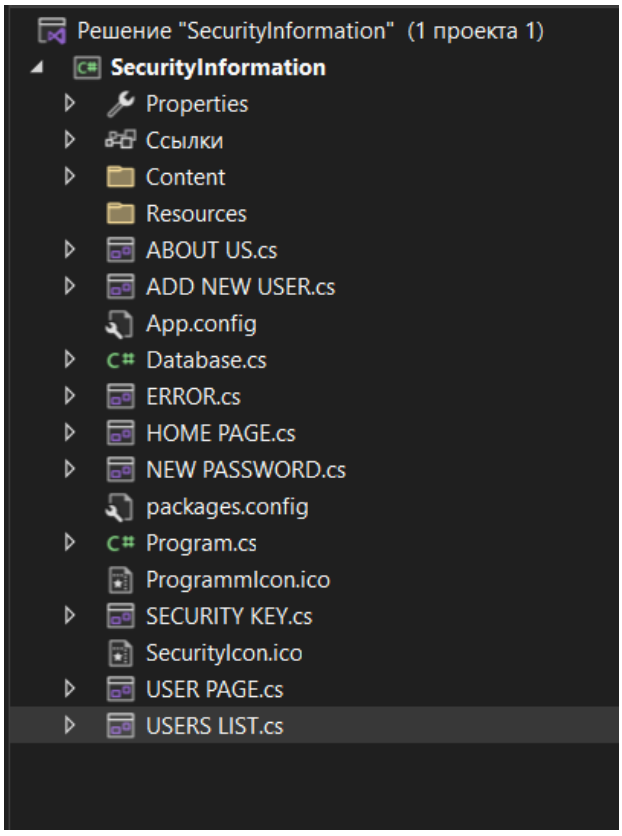
Индивидуальное задание

Наличие в пароле латинских букв и символов кириллицы.

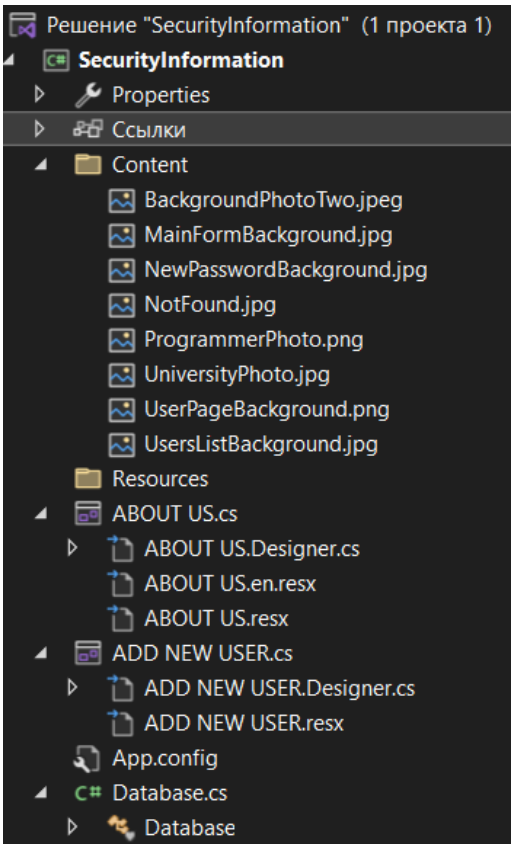
Работа выполнена на языке C# с использованием Windows Forms, SQL, MAMP.

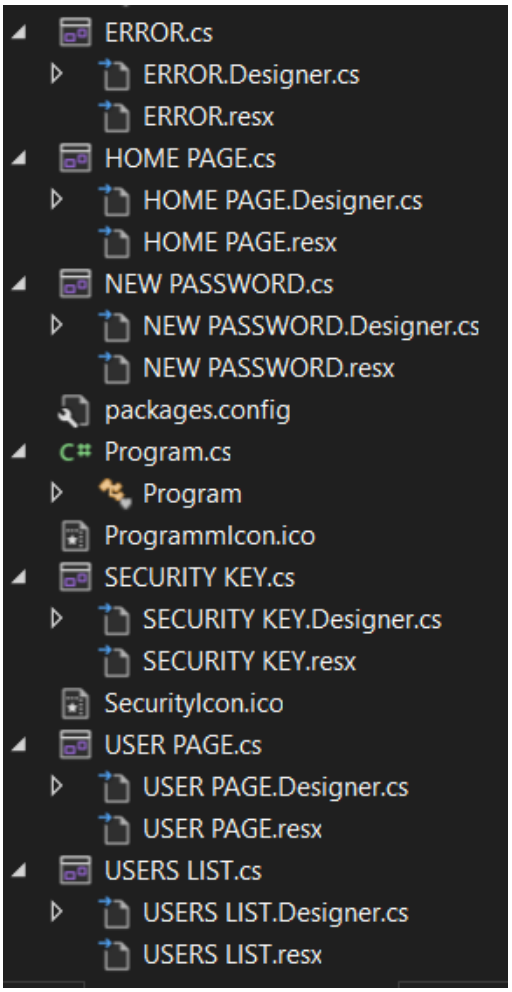
Структура проекта

Общий вид:



Расширенный вид:





Код программы с пояснением

1) Класс USER LIST.cs – список пользователей:

Необходимые библиотеки:

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Data;
using System.Windows.Forms;
```

Структура данных “UsersNode” (пользователь):

```
public struct UsersNode  // NODE FOR LIST
{
    public string userLogin;
    public string userPassword;
    public bool userLimits;
    public bool userBlocked;
    public bool userRules;
};
```

Инициализация необходимых переменных:

```
readonly List <UsersNode> usersList = new List<UsersNode>();

readonly int minUserIndex = 0;
private int maxUserIndex;
private int currentUserIndex = 0;
bool blockBox;
bool limitsBox;
```

Загрузка начального состояние страницы пользователей:

```
private void UsersList_Load(object sender, EventArgs e)  // PREPARE FORM
{
    this.TopMost = true;
    this.UserNameTextBox.ReadOnly = true;
    this.SaveButton.Enabled = false;

    UsersNode user = new UsersNode();
    Database database = new Database();
    DataTable table = new DataTable();
    MySqlDataAdapter adapter = new MySqlDataAdapter();
    MySqlCommand getTable = new MySqlCommand("SELECT * FROM `users`", database.GetConnection());

    adapter.SelectCommand = getTable;
    adapter.Fill(table);
    maxUserIndex = table.Rows.Count - 2;

    foreach (DataRow row in table.Rows)  // ADD USERS TO LIST
    {
        user.userLogin = row["login"] as string;

        if (String.Equals(user.userLogin, "ADMIN"))
            continue;

        user.userPassword = row["password"] as string;
        user.userLimits = FromStringToBool(row["limits"].ToString());
        user.userBlocked = FromStringToBool(row["blocked"].ToString());
        user.userRules = FromStringToBool(row["root"].ToString());

        usersList.Add(user);
    }

    SetCurrentUserData();
}
```

Метод для преобразования типов:

```
public bool FromStringToBool(string value)  // METHOD FOR BOOL CELLS OF SQL
{
    if (String.Equals("True", value))
        return true;
    else
        return false;
}
```

Начальные настройки окна при генерации:

```
public void GetData(System.Drawing.Point Location)  // SET FORM REFERENCES
{
    this.StartPosition = FormStartPosition.Manual;
    this.Location = Location;
}
```

Вывод значений пользователей из структуры:

```
public void SetCurrentUserData()  // SET USER LIST DATA
{
    blockBox = UserBlockedCheckBox.Checked;
    limitsBox = UserLimitsCheckedBox.Checked;

    this.PreviousButton.Enabled = true;
    this.NextButton.Enabled = true;

    if (maxUserIndex < 0)  // WHEN WE HAVE ONLY ADMIN OR EMPTY DATABASE
    {
        this.NextButton.Enabled = false;
        this.PreviousButton.Enabled = false;
        this.SaveButton.Enabled = false;
        this.UserBlockedCheckBox.Enabled = false;
        this.UserLimitsCheckedBox.Enabled = false;
    }
    else
    {
        UserNameTextBox.Text = usersList[currentUserIndex].userLogin;
        UserBlockedCheckBox.Checked = usersList[currentUserIndex].userBlocked;
        UserLimitsCheckedBox.Checked = usersList[currentUserIndex].userLimits;

        if (currentUserIndex >= maxUserIndex)
            this.NextButton.Enabled = false;

        if (currentUserIndex <= minUserIndex)
            this.PreviousButton.Enabled = false;
    }

    this.SaveButton.Enabled = false;
}
```

Получение индекса пользователя:

```
public int GetNodeIndex(string userLogin)  // FOR UPDATING USER DATA
{
    int nodeIndex = 0;

    foreach (UsersNode node in usersList)
    {
        nodeIndex += 1;

        if (String.Equals(userLogin, node.userLogin))
            return nodeIndex - 1;
    }

    return -1;
}
```

Нажатие кнопки “PREVIOUS” (предыдущий пользователь):

```
private void PreviousButton_Click(object sender, EventArgs e)  // CHECK PREVIOUS USER
{
    this.SaveButton.Enabled = false;

    currentUserIndex -= 1;
    SetCurrentUserData();
}
```

Нажатие кнопки “NEXT” (следующий пользователь):

```
private void NextButton_Click(object sender, EventArgs e)  // CHECK NEXT USER
{
    this.SaveButton.Enabled = false;

    currentUserIndex += 1;
    SetCurrentUserData();
}
```

Нажатие кнопки “SAVE” (сохранение изменений об пользователе):

```
private void SaveButton_Click(object sender, EventArgs e)  // UPDATE USERS LIST AND ACCLAIM CHANGES
{
    string status = "";

    string connectionString = "server=localhost;user=root;database=usersdatabase;password=root;";
    MySqlConnection connection = new MySqlConnection(connectionString);
    connection.Open();

    string updateLimitsCommand = $"UPDATE `users` SET `limits` = '{Convert.ToInt32(UserLimitsCheckedBox.Checked).ToString()}' WHERE `users`.`login` = '{UserNameTextBox.Text}';";
    MySqlCommand commandOne = new MySqlCommand(updateLimitsCommand, connection);
    commandOne.ExecuteScalar();

    if (UserLimitsCheckedBox.Checked != limitsBox)
        status += "LIMITS WAS CHANGED ";

    string updateBlockedCommand = $"UPDATE `users` SET `blocked` = '{Convert.ToInt32(UserBlockedCheckBox.Checked).ToString()}' WHERE `login` = '{UserNameTextBox.Text}';";
    MySqlCommand commandTwo = new MySqlCommand(updateBlockedCommand, connection);
    commandTwo.ExecuteScalar();
    connection.Close();

    if (UserBlockedCheckBox.Checked != blockBox)
        status += "USER WAS BLOCKED!";

    if (!string.IsNullOrEmpty(status))
        MessageBox.Show(status);

    UsersNode node = new UsersNode
    {
        userLogin = usersList[GetNodeIndex(UserNameTextBox.Text)].userLogin,
        userPassword = usersList[GetNodeIndex(UserNameTextBox.Text)].userPassword,
        userBlocked = UserBlockedCheckBox.Checked,
        userLimits = UserLimitsCheckedBox.Checked,
        userRules = usersList[GetNodeIndex(UserNameTextBox.Text)].userRules
    };

    usersList[GetNodeIndex(UserNameTextBox.Text)] = node;
    SetCurrentUserData();

    this.SaveButton.Enabled = false;
}
```

Нажатие кнопки “CANCEL” (закрыть страницу):

```
private void CancelButton_Click(object sender, EventArgs e)  // CLOSE THIS WINDOW
{
    this.Close();
}
```

Нажатие галочки “BLOCK USER” (блокировка пользователя):

```
private void UserBlockedCheckBox_CheckedChanged(object sender, EventArgs e)  // SWITCH BLOCK CELL
{
    this.SaveButton.Enabled = true;
}
```

Нажатие галочки “USER LIMITS” (установление ограничений пользователю):

```
private void UserLimitsCheckedBox_CheckedChanged(object sender, EventArgs e)  // SWITCH LIMITS CELL
{
    this.SaveButton.Enabled = true;
}
```

2) Класс USER PAGE.cs – меню пользователя:

Необходимые библиотеки:

```
using MySql.Data.MySqlClient;
using System;
using System.Windows.Forms;
```

Необходимые переменные:

```
string userLogin, userPassword;
```

Установка начальных значений для страницы:

```
public void GetData(string login, string password, System.Drawing.Point Location)  // SET FORM REFERENCES
{
    this.userLogin = login;
    this.userPassword = password;
    this.StartPosition = FormStartPosition.Manual;
    this.Location = Location;

    UserLoginLabel.Text += " " + userLogin.ToUpper();

    if (String.Equals(userLogin.ToLower(), "admin"))
        UserStatusLabel.Text += " ADMINISTRATOR";
    else
    {
        UserStatusLabel.Text += " USER";
        CheckUsersButton.Hide();
        AddUserButton.Hide();
    }
}
```

Нажатие кнопки “CHECK USERS” (список пользователей):

```
private void ShowUsersList(object sender, EventArgs e)  // OPEN USER LIST FORM
{
    string connectionString = "server=localhost;user=root;database=usersdatabase;password=root;";
    MySqlConnection connection = new MySqlConnection(connectionString);
    connection.Open();

    string updateLimitsCommand = $"SELECT COUNT(*) FROM  `users`;";
    MySqlCommand commandOne = new MySqlCommand(updateLimitsCommand, connection);
    string result = commandOne.ExecuteScalar().ToString();

    if (String.Equals(result, "1"))
    {
        MessageBox.Show("YOU DON'T HAVE USERS");
    }
    else
    {
        UsersList usersList = new UsersList();
        usersList.GetData(this.Location);
        usersList.ShowDialog();
    }
}
```

Нажатие кнопки “CLOSE” (закрыть окно):

```
private void ExitButton_Click(object sender, EventArgs e)  // EXIT
{
    HomePage newHomePage = new HomePage();
    newHomePage.Show();
    this.Close();
}
```

Нажатие кнопки “ADD NEW USER” (добавить нового пользователя):

```
private void AddUserButton_Click(object sender, EventArgs e)  // OPEN FORM ADD USER
{
    AddNewUserForn newUser = new AddNewUserForn();
    newUser.GetData(this.Location);
    newUser.ShowDialog();
}
```

Нажатие кнопки “CHANGE PASSWORD” (изменить пароль пользователя):

```
private void ChangePassword_Click(object sender, EventArgs e)  // CHANGE PASSWORD BUTTON PRESSED
{
    NewPasswordForm newAdminPassword = new NewPasswordForm();
    newAdminPassword.GetData(userLogin, userPassword, this.Location);
    newAdminPassword.ShowDialog();

    string connectionString = "server=localhost;user=root;database=usersdatabase;password=root;";
    MySqlConnection connection = new MySqlConnection(connectionString);
    connection.Open();

    string sqlCommand = $"SELECT `password` FROM `users` WHERE `login` = '{userLogin}';";
    MySqlCommand command = new MySqlCommand(sqlCommand, connection);
    userPassword = command.ExecuteScalar().ToString();
    connection.Close();
}
```

3) Класс SET NEW PASSWORD.cs – установка нового пароля:

Необходимые библиотеки:

```
using System;
using System.Windows.Forms;
using MySql.Data.MySqlClient;
```

Необходимые переменные:

```
string userLogin, userPassword;
```

Инициализация страницы:

```
public NewPasswordForm()  // FORM CONSTRUCTOR
{
    this.TopMost = true;
    InitializeComponent();
}
```

Проверка пароля на индивидуальное задание:

```
public bool CheckPasswordLimits(string password)  // TASK FUNCTION
{
    string RusLetters = "абвгдеёжзийклмнопрстуфхцчшщьыьэюя";
    string EngLetters = "abcdefghijklmnopqrstuvwxyz";
    bool EngFlag = false;
    bool RusFlag = false;

    if (!String.IsNullOrEmpty(password))  // IF STRING EXIST
    {
        foreach (char letter in RusLetters)  // CHECK RUS LETTERS
            RusFlag |= (password.ToLower().IndexOf(letter) != -1);

        foreach (char letter in EngLetters)  // CHECK ENG LETTERS
            EngFlag |= (password.ToLower().IndexOf(letter) != -1);
    }

    return RusFlag & EngFlag;
}
```

Скрытие полей при первой установке пароля:

```
public void FirstPassword()  // SITUATION WHEN USER DON'T HAS PASSWORD AND WE SET IT THE FIRST TIME
{
    this.OldPasswordBox.Hide();
    this.OldPasswordLabel.Hide();
}
```

Установка старого пароля аккаунта

```
public void SetOldPassword(string password)  // SET OLD PASSWORD TO CELL
{
    OldPasswordBox.Text = password;
}
```

Загрузка данных о пользователя из БД для сравнения с введённым:

```
public void GetData(string login, string password, System.Drawing.Point Location)  // PREPARE FORM
{
    this.userLogin = login;
    this.userPassword = password;
    this.StartPosition = FormStartPosition.Manual;
    this.Location = Location;

    string connectionString = "server=localhost;user=root;database=usersdatabase;password=root;";
    MySqlConnection connection = new MySqlConnection(connectionString);
    connection.Open();

    string findUserLimits = $"SELECT `limits` FROM `users` WHERE `login` = \"{userLogin}\";";
    MySqlCommand commandOne = new MySqlCommand(findUserLimits, connection);
    string userLimits = commandOne.ExecuteScalar().ToString();

    LimitsLabel.Text +=  " " + userLimits;
}
```

Нажатие кнопки “SET” (изменение или установка пароля):

```
private void ChangePasswordButton(object sender, EventArgs e)  // ATTEMPT TO CHANGE PASSWORD
{
    string connectionString = "server=localhost;user=root;database=usersdatabase;password=root;";
    MySqlConnection connection = new MySqlConnection(connectionString);
    connection.Open();

    string findUserLimits = $"SELECT `limits` FROM `users` WHERE `login` = \"{userLogin}\";";
    MySqlCommand commandOne = new MySqlCommand(findUserLimits, connection);
    string userLimits = commandOne.ExecuteScalar().ToString();

    String oldPassword = OldPasswordBox.Text;
    String newPassword = newPasswordBox.Text;
    String confirmPassword = confirmPasswordBox.Text;


    if (string.IsNullOrWhiteSpace(oldPassword) & OldPasswordBox.Visible == true)  // INPUT OLD PASSWORD
    {
        ErrorPage emptyOldPassword = new ErrorPage();
        emptyOldPassword.GetData(this.Location);
        emptyOldPassword.ChangeLabelText("INPUT OLD PASSWORD!");
        emptyOldPassword.ShowDialog();
    }
    else if (string.IsNullOrWhiteSpace(newPassword))  // INPUT NEW PASSWORD
    {
        ErrorPage emptyNewPassword = new ErrorPage();
        emptyNewPassword.GetData(this.Location);
        emptyNewPassword.ChangeLabelText("INPUT NEW PASSWORD!");
        emptyNewPassword.ShowDialog();
    }
    else if (string.IsNullOrWhiteSpace(confirmPassword))  // CONFIRM NEW PASSWORD
    {
        ErrorPage emptyConfirmPassword = new ErrorPage();
        emptyConfirmPassword.GetData(this.Location);
        emptyConfirmPassword.ChangeLabelText("CONFIRM NEW PASSWORD!");
        emptyConfirmPassword.ShowDialog();
    }
    else if (newPassword != confirmPassword)  // INPUT CORRECT NEW PASSWORDS
    {
        ErrorPage nonequalPasswords = new ErrorPage();
        nonequalPasswords.GetData(this.Location);
        nonequalPasswords.ChangeLabelText("YOU NEW PASSWORDS AREN'T EQUAL!");
        nonequalPasswords.ShowDialog();
    }
    else if (String.Equals(newPassword, oldPassword))  // INPUT THE SAME PASSWORD
```

```
    else if (String.Equals(newPassword, oldPassword))  // INPUT THE SAME PASSWORD
    {
        ErrorPage samePassword = new ErrorPage();
        samePassword.GetData(this.Location);
        samePassword.ChangeLabelText("YOU INPUT THE SAME PASSWORD!");
        samePassword.ShowDialog();
    }
    else if (String.Equals(userLimits, "True") & !CheckPasswordLimits(newPassword))  // CHECK SUCCESS LIMITS
    {
        ErrorPage incorrectLimits = new ErrorPage();
        incorrectLimits.GetData(this.Location);
        incorrectLimits.ChangeLabelText("PASSWORD DOESN'T PASS LIMITS!");
        incorrectLimits.ShowDialog();
    }
    else if (String.Equals(oldPassword, userPassword))  // CHANGE PASSWORD
    {
        string changePasswordCommand = $"UPDATE `users` SET `password` = '{newPassword}' WHERE `login` = '{userLogin}';";
        MySqlCommand commandTwo = new MySqlCommand(changePasswordCommand, connection);

        commandTwo.ExecuteScalar();
        this.Close();
        MessageBox.Show("PASSWORD WAS SET!");
    }
    else  // INCORRECT OLD PASSWORD
    {
        ErrorPage incorrectPassword = new ErrorPage();
        incorrectPassword.GetData(this.Location);
        incorrectPassword.ChangeLabelText("INCORRECT PASSWORD!");
        incorrectPassword.ShowDialog();
    }

    connection.Close();
}
```

Нажатие кнопки закрыть страницу:

```
private void CloseButton_Click(object sender, EventArgs e)  // CLOSE BUTTON
{
    this.Close();
}
```

4) Класс HOME PAGE.cs – начальная страница:

Необходимые модули:

```
using System;
using System.Windows.Forms;
using MySql.Data.MySqlClient;
using System.Security.Cryptography;
using System.Text;
using System.Data;
using System.Collections.Generic;
```

Необходимые переменные:

```
public int incorrectInputs = 0;
public string securityPassword = "123";
readonly List<SecureNode> usersList = new List<SecureNode>();
private readonly System.Windows.Forms.Timer timer = new System.Windows.Forms.Timer();
```

Загрузка начальной страницы:

```
private void HomePage_Load(object sender, EventArgs e)  // OPEN HOME PAGE AND CHECK TABLE
{
    this.TopMost = true;
    string connectionString = "server=localhost;user=root;database=usersdatabase;password=root;";
    MySqlConnection connection = new MySqlConnection(connectionString);
    connection.Open();
    string defineUsers = $"SELECT COUNT(*) FROM `secureusers`;";
    MySqlCommand commandOne = new MySqlCommand(defineUsers, connection);

    try  // CHECK THAT TABLE IS EXIST
    {
        commandOne.ExecuteScalar();
        string dropTable = "DROP TABLE `usersdatabase`.`secureusers`";
        MySqlCommand commandTwo = new MySqlCommand(dropTable, connection);
        commandTwo.ExecuteScalar();
    }
    catch  // CREATE TABLE AND ADD ADMIN IF THERE AREN'T
    {
        string createTable = $"CREATE TABLE `usersdatabase`.`users` ( `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT , " +
            $"`login` VARCHAR(64) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL , `password` VARCHAR(32) CHARACTER " +
            $"SET utf8 COLLATE utf8_general_ci NOT NULL , `limits` TINYINT(1) NOT NULL , `blocked` TINYINT(1) NOT NULL , `root` " +
            $"TINYINT(1) NOT NULL DEFAULT '0' , PRIMARY KEY (`login`), INDEX (`id`)) ENGINE = InnoDB;";
        MySqlCommand commandTwo = new MySqlCommand(createTable, connection);
        commandTwo.ExecuteScalar();

        string addAdmin = $"INSERT INTO `users` (`id`, `login`, `password`, `limits`, `blocked`, `root`) VALUES (NULL, 'ADMIN', '', '0', '0', '1');";
        MySqlCommand commandThree = new MySqlCommand(addAdmin, connection);
        commandThree.ExecuteScalar();
    }

    connection.Close();
}
```

Нажатие кнопки “EXIT” (закрытие страницы):

```
private void HomePage_FormClosed(object sender, FormClosedEventArgs e)  // ENCRYPTION BY THE END OF PROGRAMM
{
    AddEncryption();
}
```

Функция блокировки на время при трёхкратном неверном вводе пароля:

```
void TimerOn(object sender, System.EventArgs e)  // TURN ON BLOCK TIMER
{
    SignInButton.Enabled = true;
    timer.Stop();
}
```

Проверка наличия ограничений на пароль:

```
public bool CheckPasswordLimits(string password)  // TASK FUNCTION
{
    string RusLetters = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя";
    string EngLetters = "abcdefghijklmnopqrstuvwxyz";
    bool EngFlag = false;
    bool RusFlag = false;

    if (!String.IsNullOrEmpty(password))  // IF STRING EXIST
    {
        foreach (char letter in RusLetters)  // CHECK RUS LETTERS
            RusFlag |= (password.ToLower().IndexOf(letter) != -1);

        foreach (char letter in EngLetters)  // CHECK ENG LETTERS
            EngFlag |= (password.ToLower().IndexOf(letter) != -1);
    }

    return RusFlag & EngFlag;
}
```

Нажатие “OPTIONS” (опции):

```
private void AboutUsToolStripMenuItem_Click(object sender, EventArgs e)  // MENU PAGE
{
    AboutUs aboutUs = new AboutUs();
    aboutUs.ShowDialog();
}
```

Нажатие кнопки “SIGN IN” (входа в аккаунт):

```
private void SignIn_Click(object sender, EventArgs e)  // CLICK BUTTON SIGN IN
{
    String userLogin = UserNameTextBox.Text;
    String userPassword = UserPasswordTextBox.Text;

    if (string.IsNullOrEmpty(userLogin))  // IF USER DON'T INPUT USER NAME
    {
        ErrorPage emptyInputUserLogin = new ErrorPage();
        emptyInputUserLogin.GetData(this.Location);
        emptyInputUserLogin.ChangeLabelText("YOU DON'T INPUT USER LOGIN!");
        emptyInputUserLogin.ShowDialog();
    }
    else  // CHECK THAT USER EXIST IN DATABASE
    {
        string connectionString = "server=localhost;user=root;database=usersdatabase;password=root;";
        MySqlConnection connection = new MySqlConnection(connectionString);
        connection.Open();

        string findUserCommand = $"SELECT COUNT(*) FROM `users` WHERE `login` = \"{userLogin}\";";
        MySqlCommand commandOne = new MySqlCommand(findUserCommand, connection);
        string userFind = commandOne.ExecuteScalar().ToString();

        if (!String.Equals(userFind, "1"))  // IF USER WASN'T FOUND
        {
            ErrorPage userDoesntExist = new ErrorPage();
            userDoesntExist.GetData(this.Location);
            userDoesntExist.ChangeLabelText("USER DOESN'T EXIST!");
            userDoesntExist.ShowDialog();
            connection.Close();
        }
        else
        {
            string checkUserPasswordCommand = $"SELECT `password` FROM `users` WHERE `users`.`login` = \"{userLogin}\";";
            MySqlCommand commandTwo = new MySqlCommand(checkUserPasswordCommand, connection);
            string userPasswordFind = commandTwo.ExecuteScalar().ToString();

            if (String.Equals(userPasswordFind, userPassword))  // IF PASSWORD CORRECT
            {
                string checkUserBlockCommand = $"SELECT `blocked` FROM `users` WHERE `users`.`login` = \"{userLogin}\";";
                MySqlCommand commandThree = new MySqlCommand(checkUserBlockCommand, connection);
                string userBlockFind = commandThree.ExecuteScalar().ToString();

                string checkUserStatus = $"SELECT `limits` FROM `users` WHERE `users`.`login` = \"{userLogin}\";";
                MySqlCommand commandFour = new MySqlCommand(checkUserStatus, connection);
                string userLimitsFind = commandFour.ExecuteScalar().ToString();
```

```
                if (String.Equals(userBlockFind, "True"))
                {
                    ErrorPage userBlocked = new ErrorPage();
                    userBlocked.GetData(this.Location);
                    userBlocked.ChangeLabelText("USER HAS BEEN BLOCKED!");
                    userBlocked.ShowDialog();
                }
                else if (string.IsNullOrEmpty(userPassword))  // IF USER DON'T HAVE PASSWORD
                {
                    MessageBox.Show("YOU DON'T HAVE PASSWORD! PLEASE SET IT.");
                    NewPasswordForm setFirstPassword = new NewPasswordForm();
                    setFirstPassword.GetData(userLogin, userPassword, this.Location);
                    setFirstPassword.FirstPassword();
                    setFirstPassword.ShowDialog();
                }
                else if (String.Equals("True", userLimitsFind) & !CheckPasswordLimits(userPassword))  // IF USER HAVE LIMITS WE NEED TO CHECK HIS PASSWORD
                {
                    MessageBox.Show("YOUR PASSWORD DOESN'T PASS LIMITS!");

                    NewPasswordForm setNewPassword = new NewPasswordForm();
                    setNewPassword.GetData(userLogin, userPassword, this.Location);
                    setNewPassword.ShowDialog();
                }
                else
                {
                    UserForm userPage = new UserForm();
                    userPage.GetData(userLogin, userPassword, this.Location);
                    this.UserNameTextBox.Clear();
                    this.UserPasswordTextBox.Clear();
                    userPage.ShowDialog();
                    this.UserNameTextBox.Clear();
                    this.UserPasswordTextBox.Clear();
                }

                connection.Close();
            }
            else
            {
                ++incorrectInputs;
                ErrorPage incorrectPassword = new ErrorPage();
                incorrectPassword.GetData(this.Location);
                incorrectPassword.ChangeLabelText("INCORRECT PASSWORD!");
                incorrectPassword.ShowDialog();
```

```
                    if (incorrectInputs > 2)
                    {
                        incorrectInputs = 0;
                        timer.Interval = 45000;  // HERE A TIME IN MILISECONDS
                        MessageBox.Show($"YOU HAVE BEEN BLOCKED ON {timer.Interval / 1000} SECONDS!");
                        timer.Tick += TimerOn;
                        timer.Start();
                        SignInButton.Enabled = false;
                    }
                }
            }
        }
    }
}
```

5) Класс ERROR.cs – класс ошибка:

Необходимые библиотеки:

```
using System;
using System.Windows.Forms;
```

Установка текста ошибки:

```
public void ChangeLabelText(string text)  // SET ERROR TEXT
{
    this.ErrorTextLabel.Text = text;
}
```

Нажатие кнопки “CLOSE” (закрыть окно ошибки):

```
private void CloseButton_Click(object sender, EventArgs e)  // CLOSE BUTTON
{
    this.Close();
}
```

6) Класс Database.cs – класс базы данных:

Подключение к базе данных:

```
public void OpenConnection()
{
    if (sqlSqlConnection.State == System.Data.ConnectionState.Closed)
        sqlSqlConnection.Open();
}
```

Отключение от базы данных:

```
public void CloseConnection()
{
    if (sqlSqlConnection.State == System.Data.ConnectionState.Open)
        sqlSqlConnection.Close();
}
```

7) Класс ADD NEW USER.cs – класс добавления нового пользователя:

Необходимые библиотеки:

```
using MySql.Data.MySqlClient;
using System;
using System.Windows.Forms;
```

Сбор данных об активном пользователе:

```
public void GetData(System.Drawing.Point Location)  // SET PREVIOUS FORM LOCATION
{
    this.StartPosition = FormStartPosition.Manual;
    this.Location = Location;
}
```

Нажатие клавиши “ADD” (добавление пользователя):

```
private void AddButton_Click(object sender, EventArgs e)  // TRY TO ADD NEW USER
{
    string newUserLogin = UserLoginTextBox.Text;

    if (string.IsNullOrWhiteSpace(newUserLogin))  // IF USER DON'T INPUT ANYTHING
    {
        ErrorPage emptyInputLogin = new ErrorPage();
        emptyInputLogin.ChangeLabelText("YOU DON'T INPUT ANYTHING!");
        emptyInputLogin.GetData(this.Location);
        emptyInputLogin.ShowDialog();
    }
    else
    {
        string connectionString = "server=localhost;user=root;database=usersdatabase;password=root;";
        MySqlConnection connection = new MySqlConnection(connectionString);
        connection.Open();

        string commandOne = $"SELECT COUNT(*) FROM `users` WHERE `login` = \"{newUserLogin}\";";
        string commandTwo = $"INSERT INTO `users`(`id`, `login`, `password`, `limits`, `blocked`, `root`) VALUES (NULL, '{newUserLogin}', '', '0', '0', '0');";
        MySqlCommand commandFind = new MySqlCommand(commandOne, connection);
        string result = commandFind.ExecuteScalar().ToString();

        if (String.Equals(result, "1"))  // CHECK THAT USER IS ALREADY EXIST
        {
            ErrorPage existUser = new ErrorPage();
            existUser.GetData(this.Location);
            existUser.ChangeLabelText("USER IS ALREADY EXIST!");
            existUser.ShowDialog();
        }
        else  // ADD USER
        {
            MySqlCommand commandAdd = new MySqlCommand(commandTwo, connection);
            commandAdd.ExecuteScalar();
            MessageBox.Show("USER WAS ADDED!");
        }

        connection.Close();
        this.Close();
    }
}
```

Нажатие кнопки “CLOSE” (закрытие окна):

```
private void CloseButton_Click(object sender, EventArgs e)  // CLOSE FORM
{
    this.Close();
}
```

8) Класс ABOUT US.cs – класс меню опций:

Необходимые библиотеки:

```
using System.Diagnostics;
using System.Windows.Forms;
```

Загрузка страницы:

```
public AboutUs()  // SET PAGE PREFERENCE
{
    this.TopMost = true;
    _ = new OpenFileDialog();
    InitializeComponent();
}
```

Нажатие на фото программиста:

```
private void PhotoButton_Click(object sender, System.EventArgs e)  // PROGRAMMER GITHUB LINK
{
    Process.Start("https://github.com/Denzi33");
}
```

Нажатие на логотип университета:

```
private void UniversityPicture_Click(object sender, System.EventArgs e)  // UNIVERSITY LINK
{
    Process.Start("https://stankin.ru/");
}
```

Скриншоты работы программы

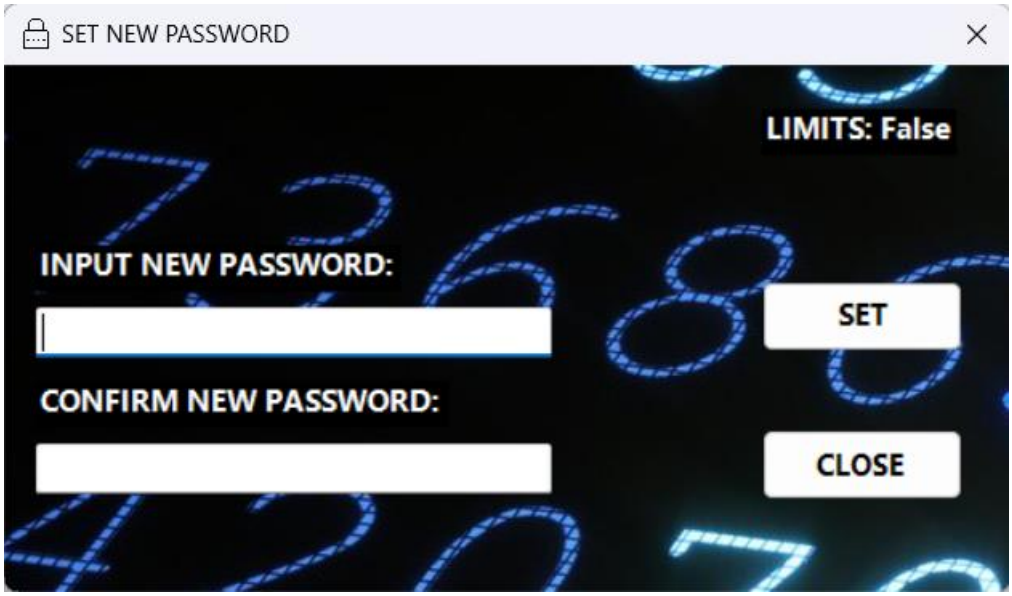


Рис. 1 Форма первой установки пароля администратором

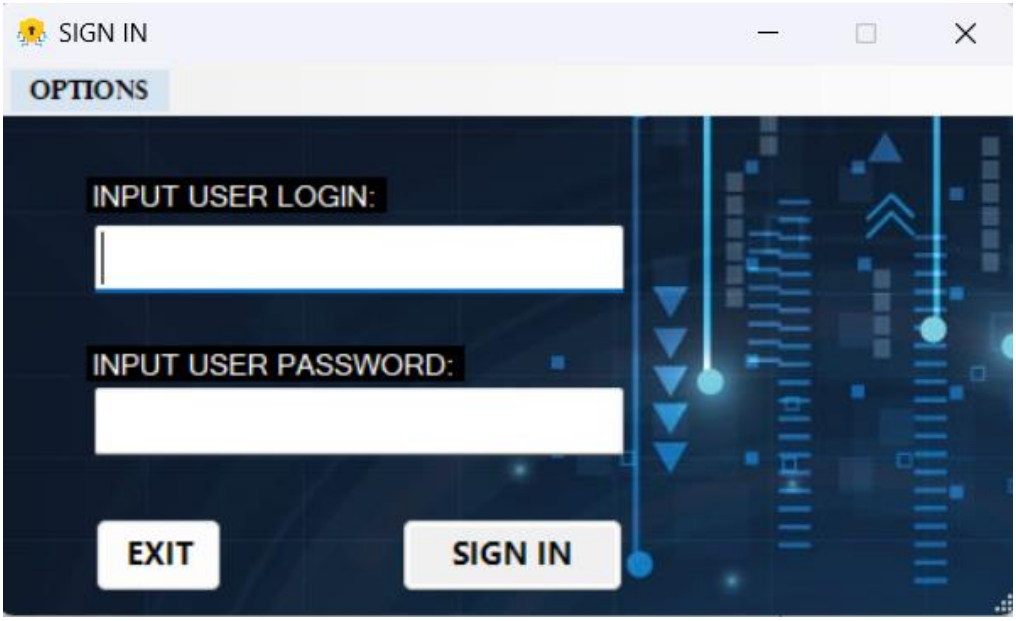


Рис. 2 Форма авторизации

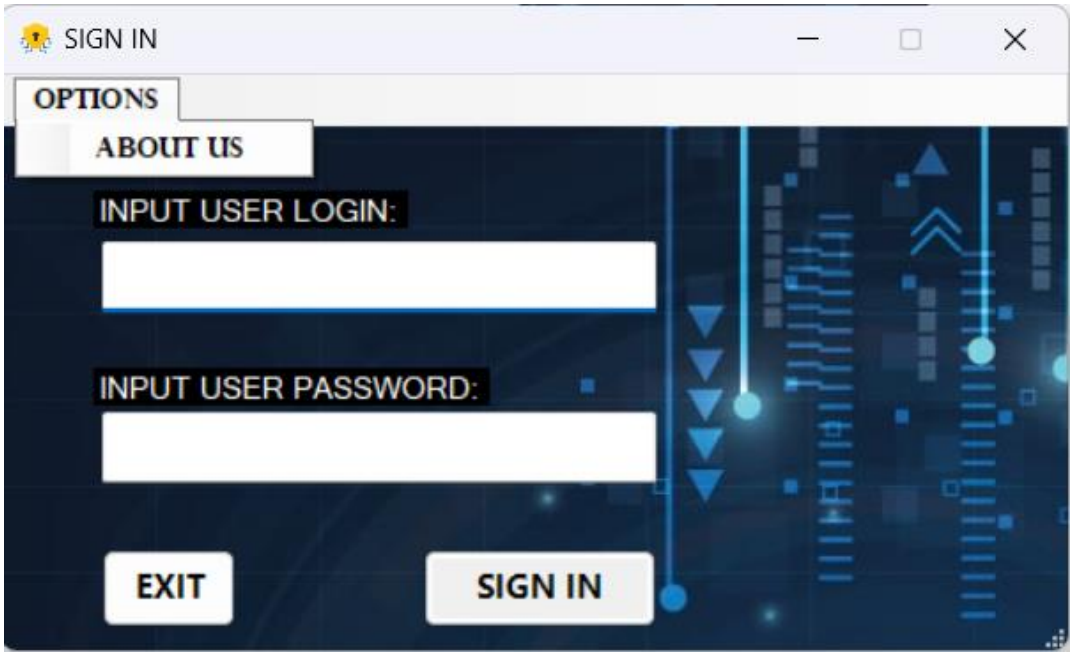


Рис. 3 Открытие меню

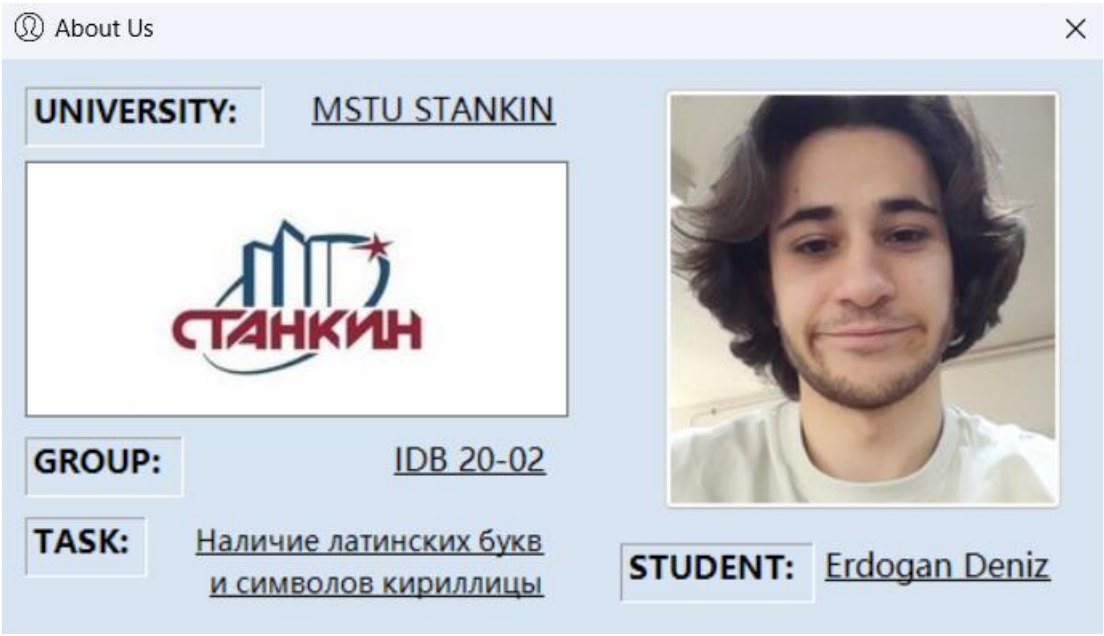


Рис. 4 Открытие меню

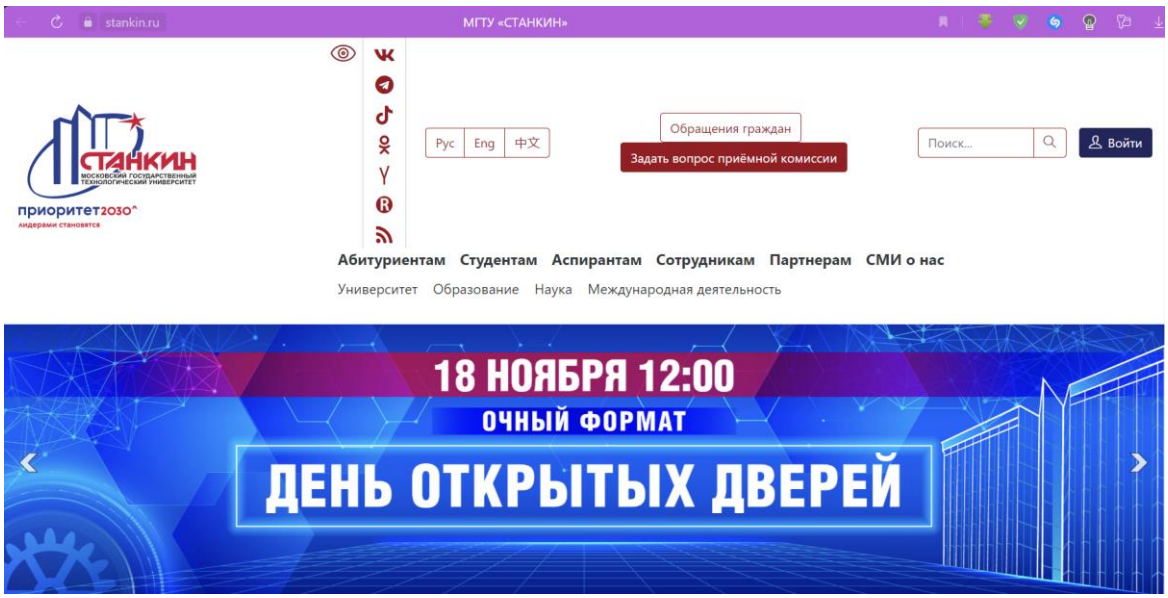


Рис. 5 Нажатие на логотип университета

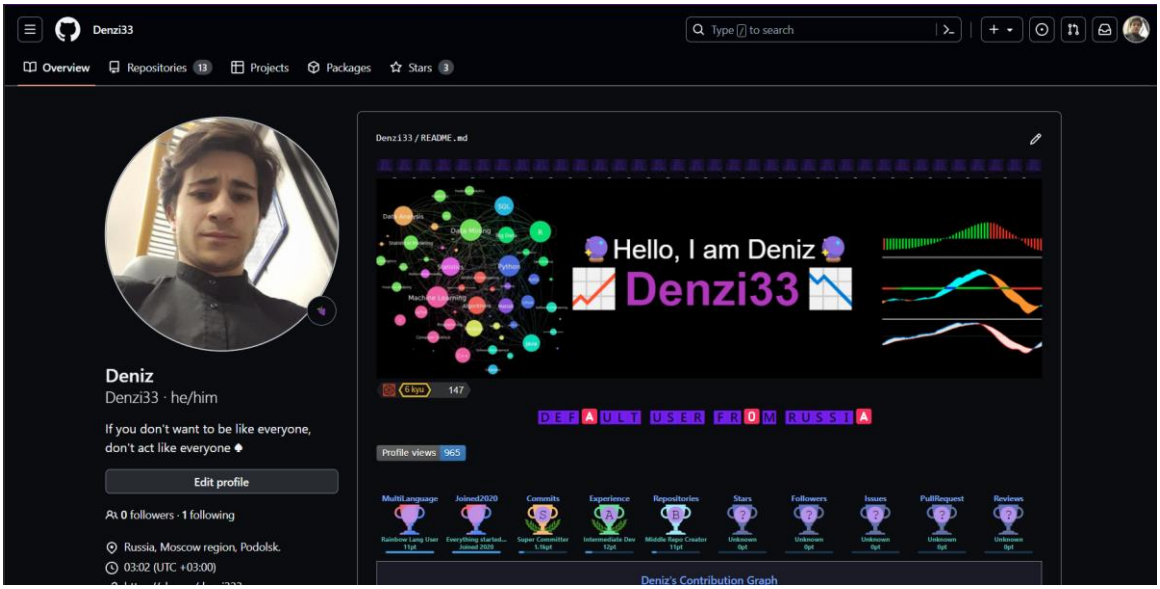


Рис. 6 Нажатие на фото автора

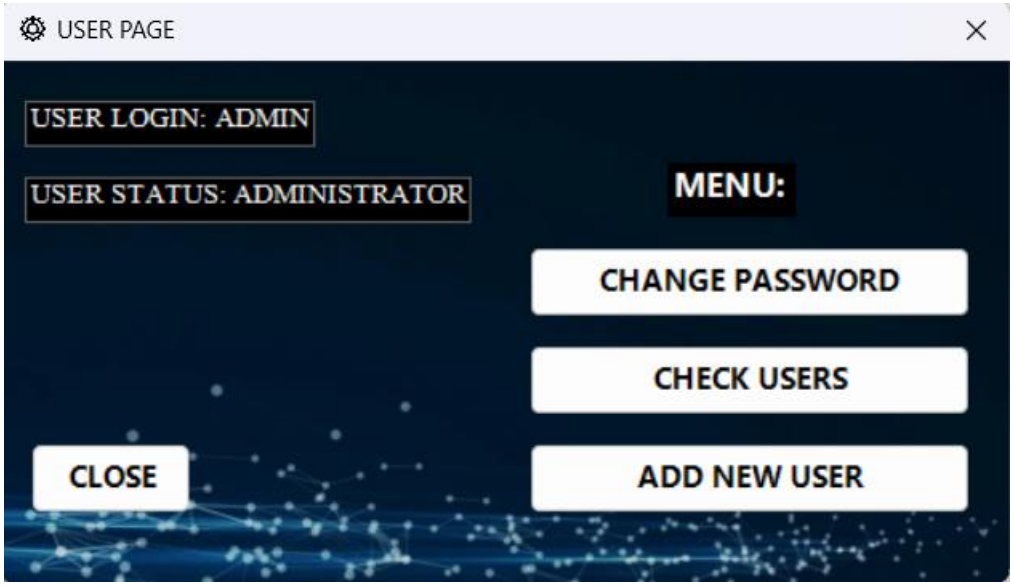


Рис. 7 Меню администратора

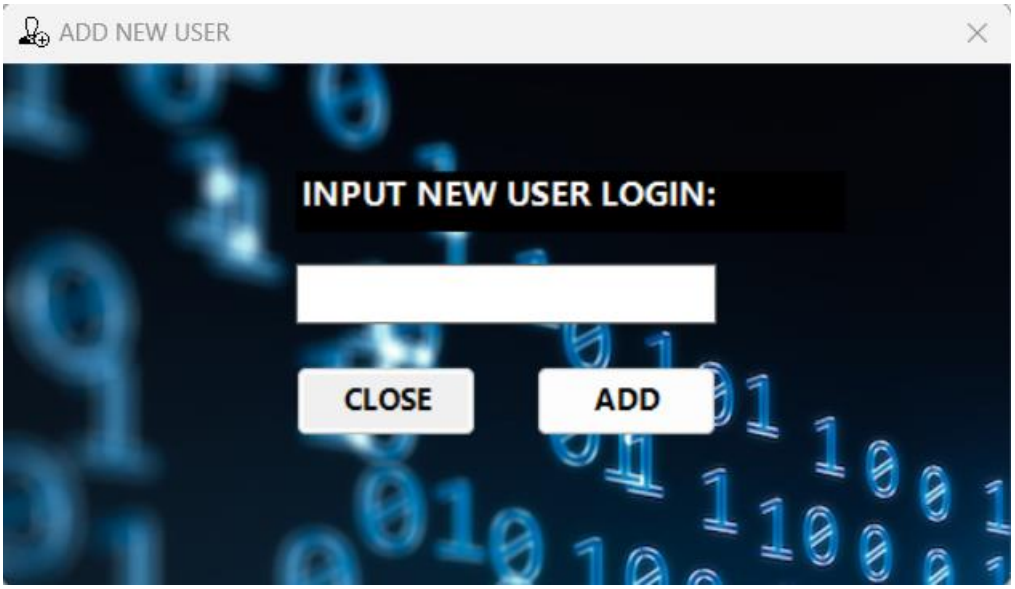


Рис. 8 Добавление нового пользователя

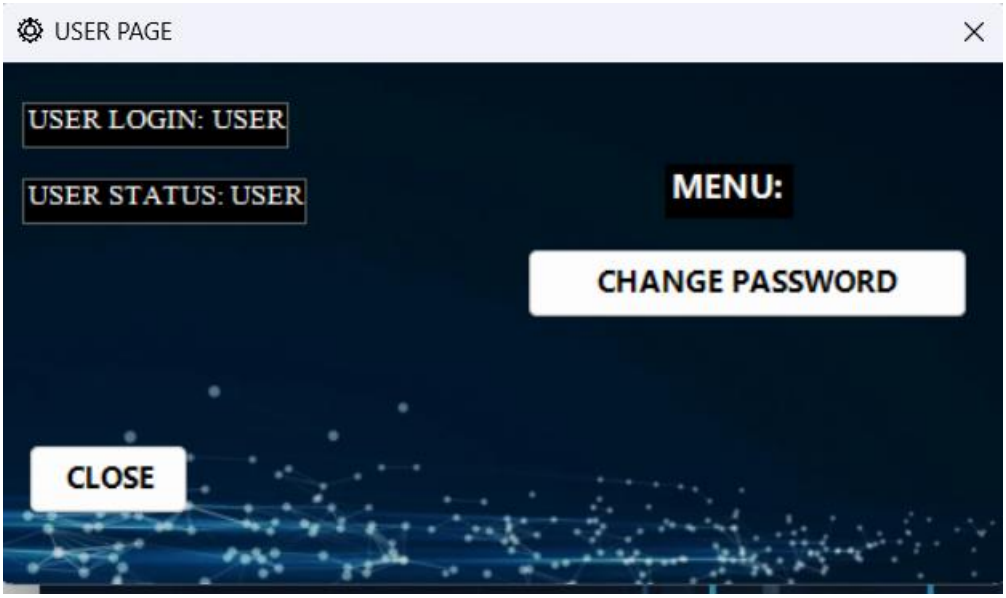


Рис. 9 Меню пользователя

Рис. 10 Просмотр пользователей

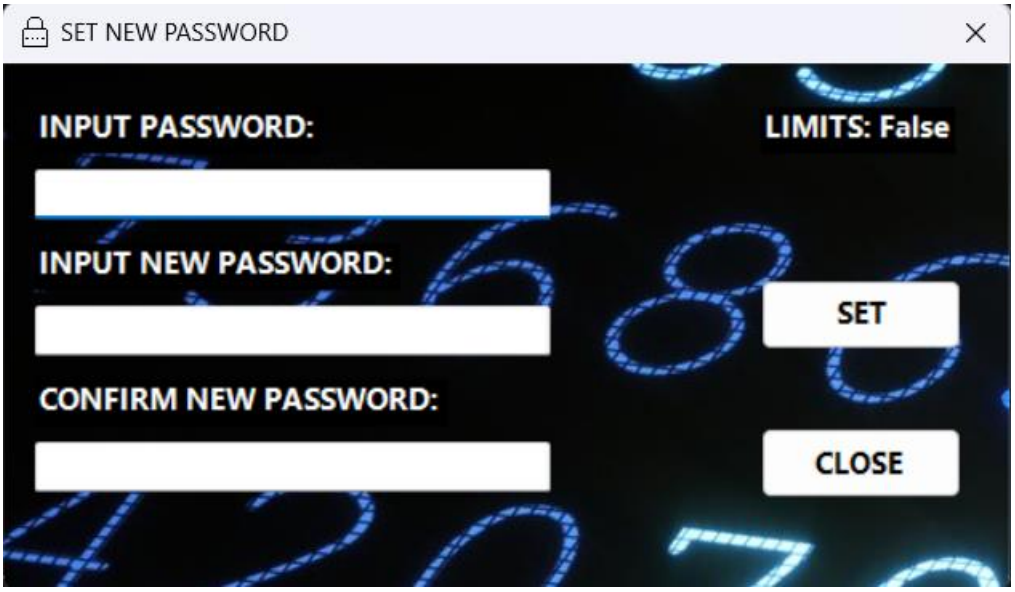


Рис. 11 Смена пароля

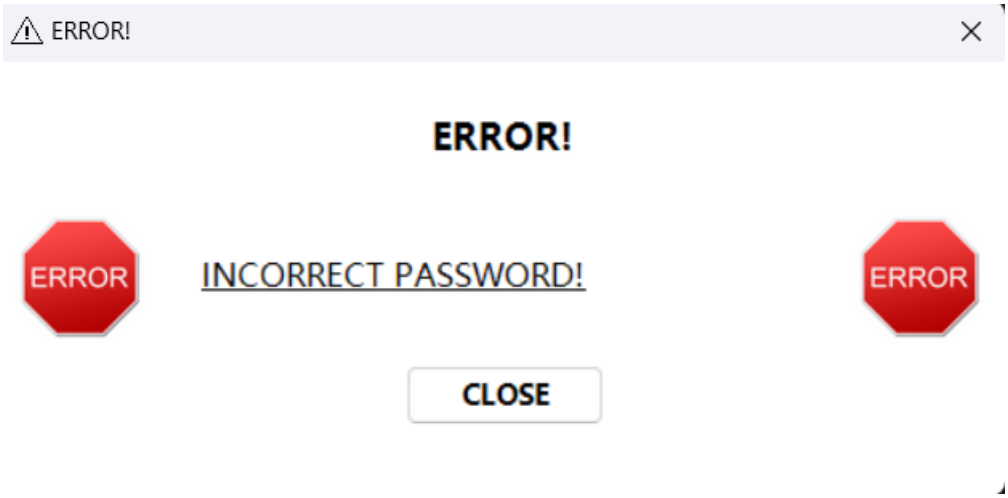


Рис. 12 Неверный ввод пароля

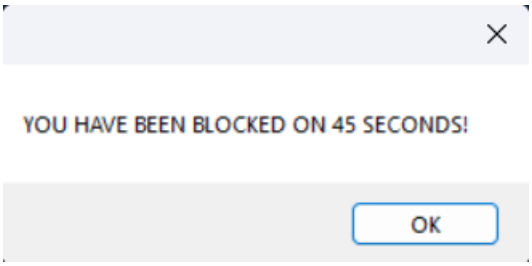


Рис. 13 Блокировка при трёхкратном неверном вводе пароля

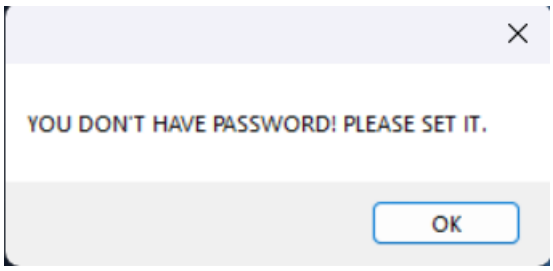


Рис. 14 Сообщение при отсутствии пароля пароля

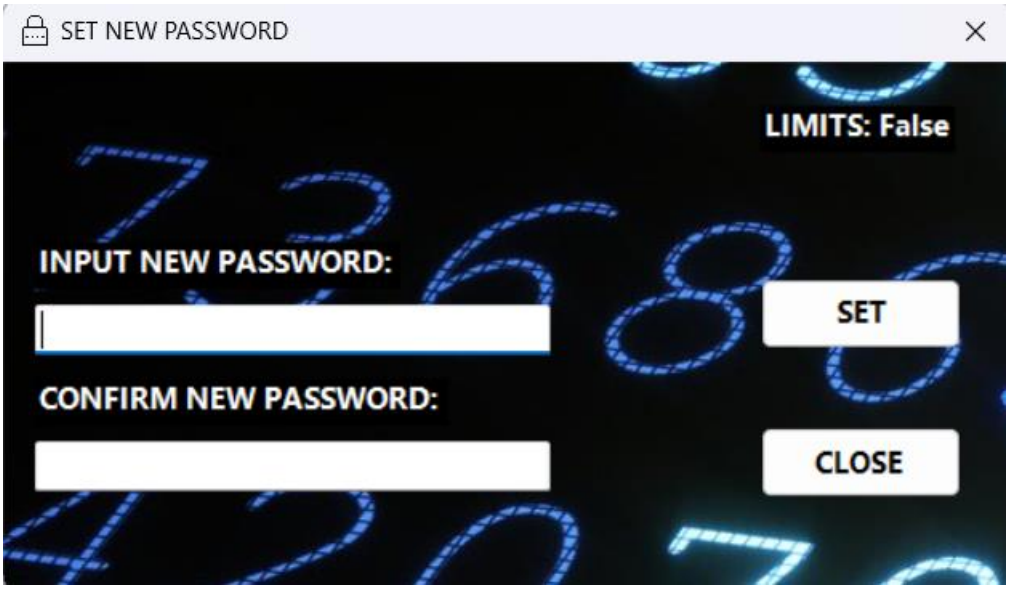


Рис. 15 Установка пароля

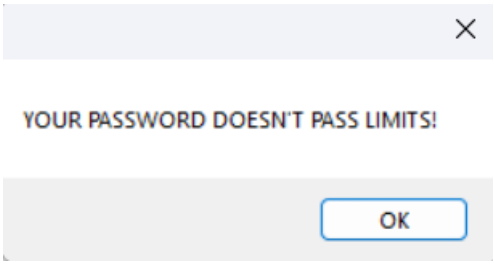


Рис. 16 Несоответствие ограничениям пароля

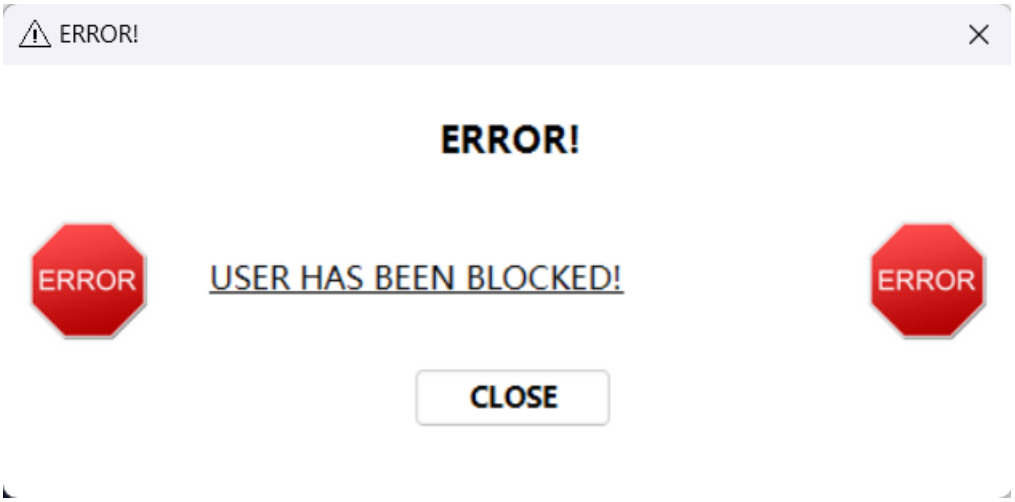


Рис. 17 Попытка входа при заблокированном аккаунте

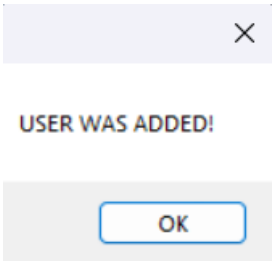


Рис. 18 Успешное добавление пользователя

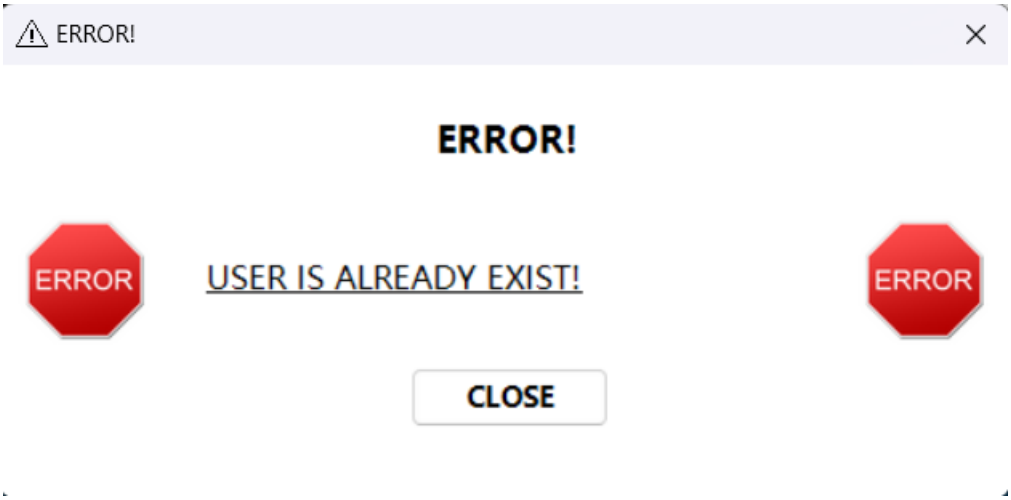


Рис. 19 Попытка добавить уже существующего пользователя

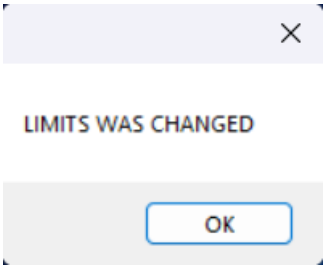


Рис. 20 Установка ограничений пользователю

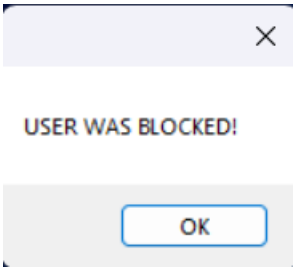


Рис. 21 Блокировка пользователя

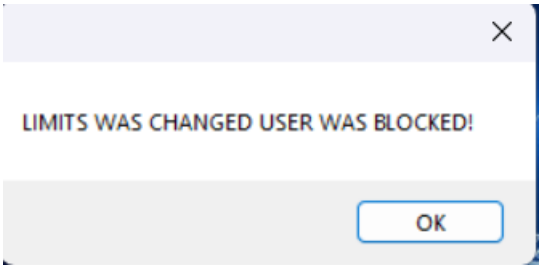


Рис. 22 Блокировка и установка ограничений пользователю

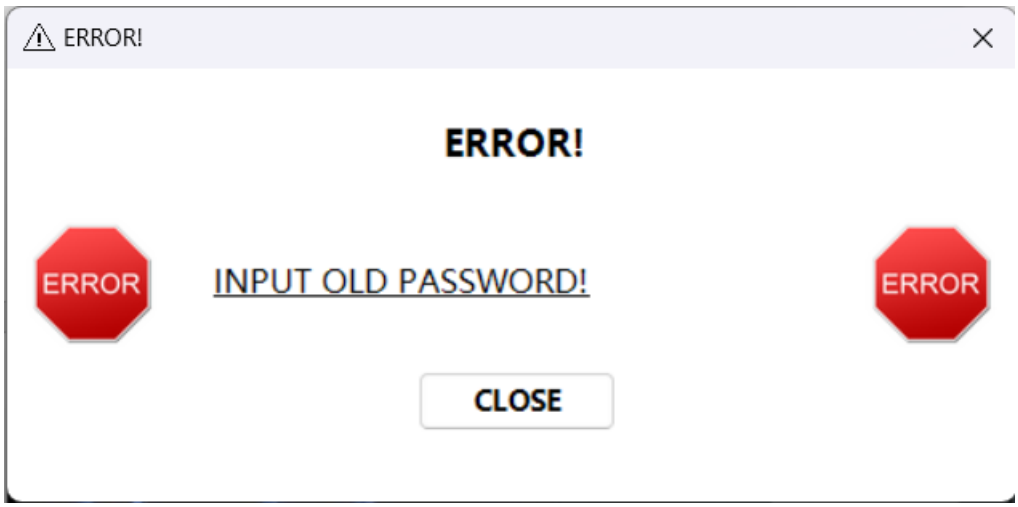


Рис. 23 Пустое поле старого пароля при смене пародя

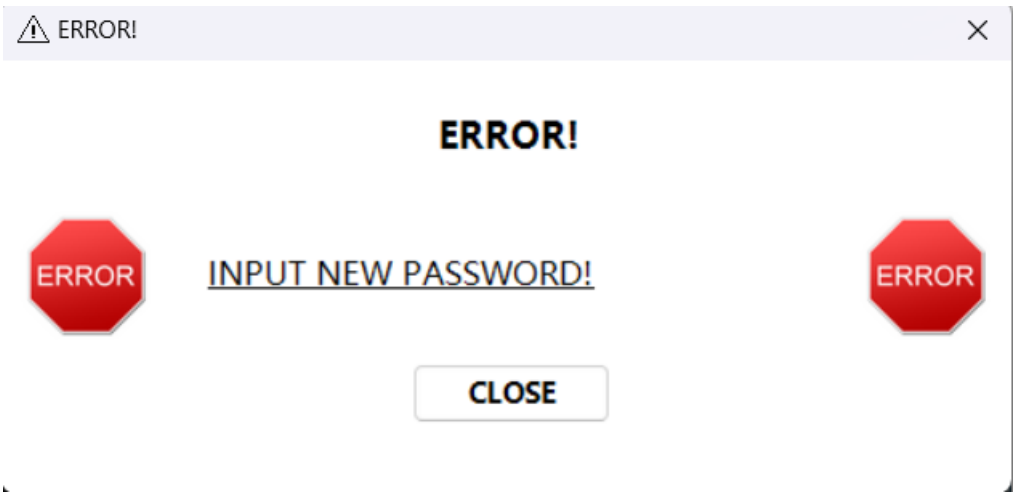


Рис. 24 Пустое поле нового пароля при смене пароля

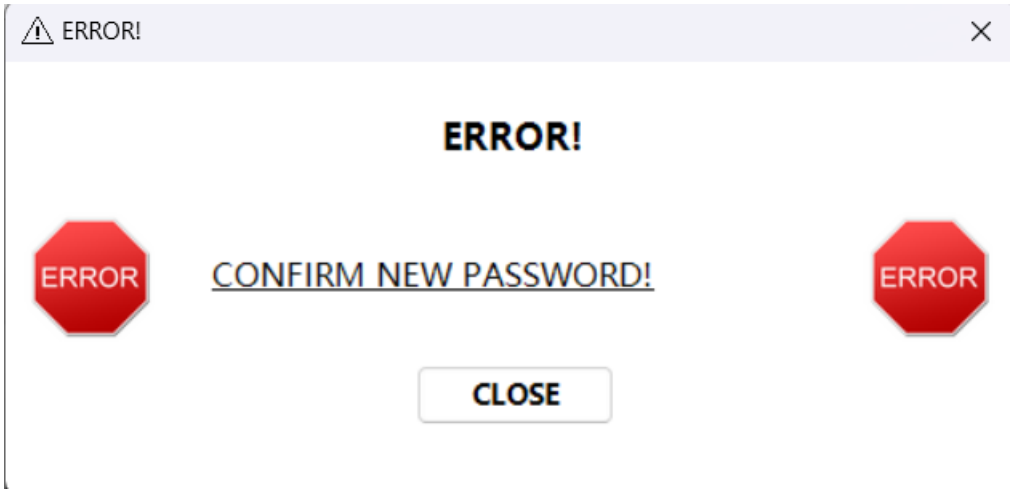


Рис. 25 Пустое поле подтверждения пароля при смене пародя

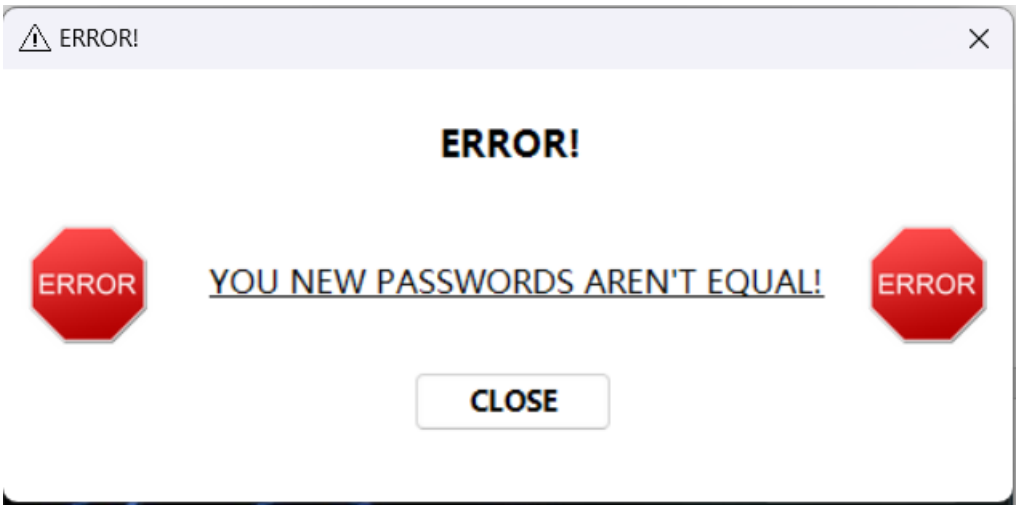


Рис. 26 Неправильное подтверждение нового пароля при смене пароля

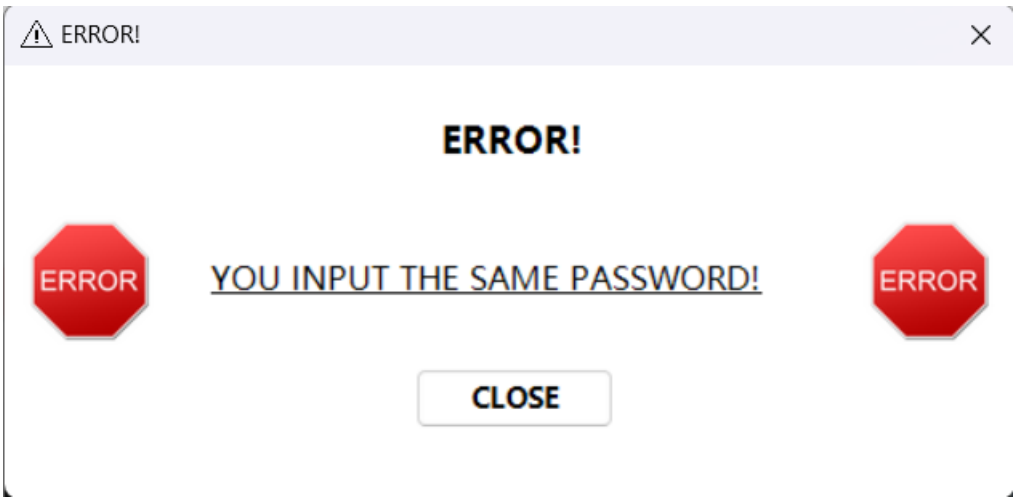


Рис. 27 Неправильный ввод старого пароля

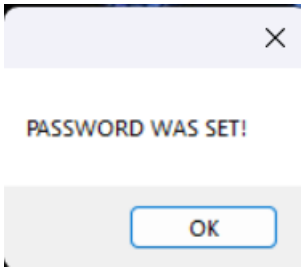


Рис. 28 Успешная смена пароля

Рис. 29 Попытка войти в несуществующего пользователя

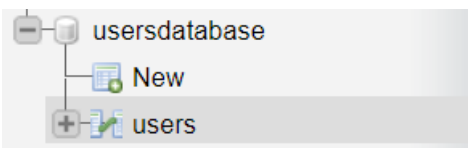


Рис. 30 База данных с пользователями

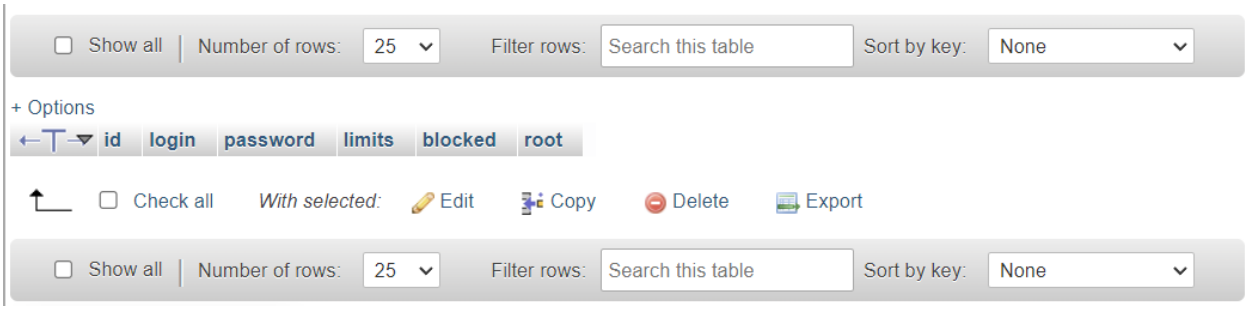


Рис. 31 База данных до запуска программы

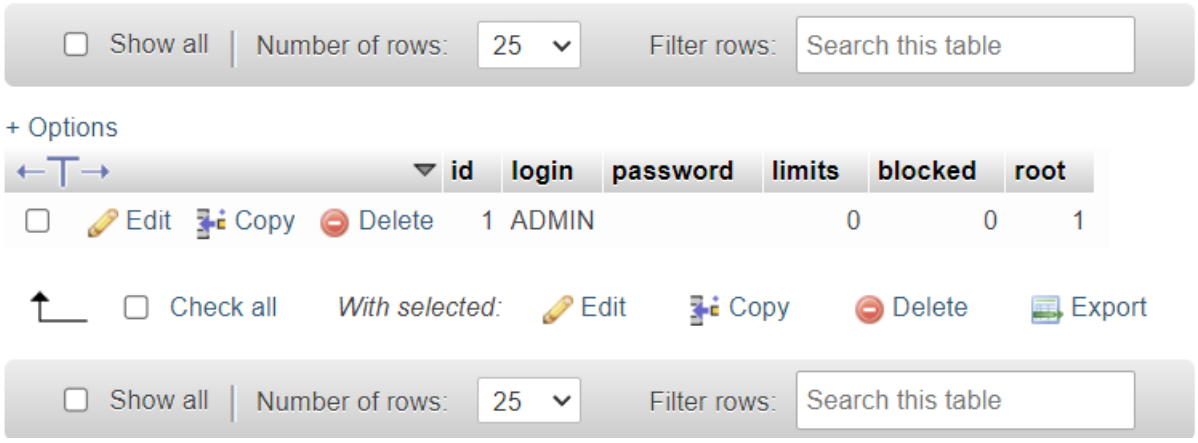


Рис. 32 База данных при первом запуске

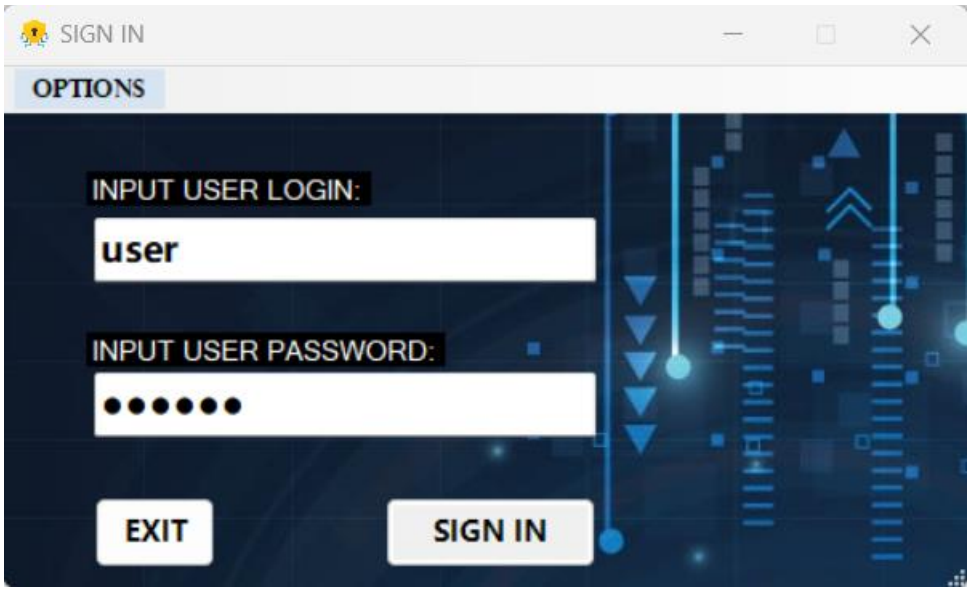


Рис. 33 Заменя пароля * при вводе

Рис. 34 Иконка приложения