|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ | | | | | | | | | | |
|  | | | |  | | |  | | | |
| ФГБОУ ВПО «Пермский государственный национальный исследовательский университет» | | | | | | | | | | |
|  | |  | | | | | | |  | |
|  | | ОТЧЕТ  по лабораторной работе «Использование форм для модификации данных» по дисциплине «Учебная практика по базам данных» | | | | | | |  | |
|  | | |  | | | | |  | | |
|  | Работу выполнил  студент гр. ПМИ-1,2  Васильевых М.Г. \_\_\_\_\_\_\_\_\_\_  (подпись)  «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2021 | | | |  | Проверил  ассистент кафедры МОВС  Турова И.А. \_\_\_\_\_\_\_\_\_\_  (подпись)  «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2021 | | | |  |
|  | |  | | | | | | |  | |
|  | | Пермь 2021 | | | | | | |  | |

Содержание

[1 Постановка задачи 3](#_Toc83312391)

[2 Описание предметной области и базы данных 4](#_Toc83312392)

[3 Описание функционала разработанной программы 5](#_Toc83312393)

[3.1. Внешний вид и функционал приложения 5](#_Toc83312394)

[3.2 Общий алгоритм решения задачи 7](#_Toc83312395)

[4 Реализация программы 8](#_Toc83312396)

[5 Тестирование программы 13](#_Toc83312397)

# 1 Постановка задачи

Написать (доработать) программу, которая позволяет просматривать, добавлять, изменять и удалять записи в таблице зарегистрированных пользователей. Для просмотра должен использоваться элемент управления ListView. Для добавления и изменения записей должны использоваться формы. Формы должны быть защищены от SQL-инъекций.

Запрещено использовать пустые логин и пароль (добавление, редактирование).

При редактировании пользователя запрещено менять логин пользователя на логин другого пользователя системы. При изменении пароля новый пароль и его повтор должны совпадать. Кнопка «Сохранить» должна быть активна только при отсутствии ошибок. Если пользователь не изменял пароль, хэш его пароля в базе не должен быть изменён.

Должна быть возможность удаления нескольких выбранных пользователей.

# 2 Описание предметной области и базы данных

Для работы приложения нам таблица из базы данных «Пользователи» - рисунок 1.1

В базе данных хранится информация о зарегистрированных пользователях, логин, пароль (в виде хеш строки), дата регистрации пользователя. Представленная база данных реляционного типа.

Id имеет тип integer – целочисленный

Login имеет тип text – строка без ограничений по размеру

Password имеет тип char(60) – символьный массив длинной 60 символов так как алгоритм хэширования подразумевает хэш блинной строго 60 символов

Time имеет тип timestap – с установленным значение по умолчанию дата и время сейчас

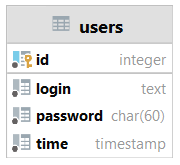


Рисунок 1.1 – Используемая БД

# 3 Описание функционала разработанной программы

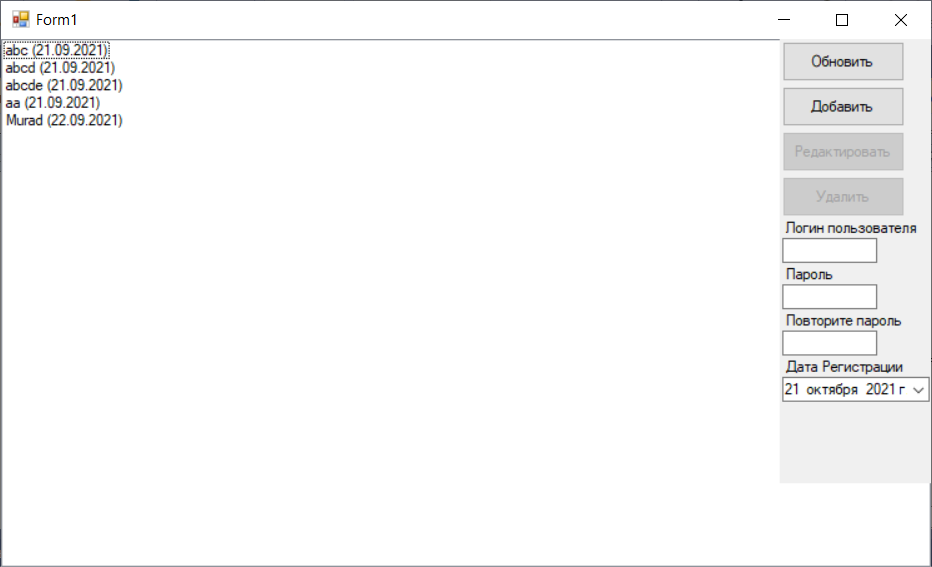
3.1. Внешний вид и функционал приложения

При запуске приложения отображается главная форма. На ней расположены элементы управления: ListViev (для отображения записей и выбора для взаимодействия с ними), Button (для подтверждения действия с записями), label(отображение информации о действии), DataTimePicker, textbox(получения данных для записи)

Если пользователь не выбрал ни одну из записей то блокируются кнопки «удалить», «изменить».

Если пользователь ни ввел ничего в поля для ввода или ввёл некорректные данные то поля блокируются кнопки «добавить», «редактировать»

Внешний вид приложения представлен на рис. 1.



## 3.2 Общий алгоритм решения задачи

Данную задачу можно разделить на 5 подзадач Добавление, Удаление, Изменение, Отображение полей, Допуск к работе с данными

1)Отображение

Для отображения полей мы создаём запрос в БД, на получение всех записей из БД. И добавляем в поля ListViev информацию о записях соответственно каждому пользователю(его логин и дату регистрации)

2) Добавление

Сначала получил логин и пароль нового пользователя. С использование таблицы подготовим выборку в соответствии логинам идентичным введенным пользователем.

Если при проверке в таблице не оказалось таких логинов, переходим к проверке соответствия пароля заданным стандартам безопасности, проверяя наличие хотя бы одного символа из каждого необходимого множества при помощи операции пересечения множеств.

Если оба условия выполнены, выполняем шифрование пароля, и передаём значение о новом пользователе в таблицу, время регистрации устанавливается по умолчанию в БД.

3)Удаление

Выбираем на поле ListViev группу пользователей которых хотим удалить и последовательно выполняем запрос на удаления ссылаясь на id удаляемого пользователя.

4)Изменение

Выбираем на поле ListViev пользователя данные о котором мы хотим изменить. Создаём запрос на проверку наличие введённого логина в БД, и запрос на получения пароля изменяемого пользователя(по id) и если все условия выполняются то формируем запрос на изменение данных о пользователе.

5)Допуск

Привязываем событие нажатия мышкой на часть рядом с вводимыми полями для проверки введенных данных. А именно: совпадает ли пароль с его повтором, имеется ли введенный логин в БД, корректный ли пароль, выделено ли значение в ListViev.

# 4 Реализация программы

**using** **System**;

**using** **System.Collections.Generic**;

**using** **System.ComponentModel**;

**using** **System.Data**;

**using** **System.Drawing**;

**using** **System.Linq**;

**using** **System.Text**;

**using** **System.Threading.Tasks**;

**using** **System.Windows.Forms**;

**using** **Npgsql**;

**namespace** **Модификация**

{

**public** **partial** **class** **Form1** : Form

{

**private** **string** connStr = **new** NpgsqlConnectionStringBuilder // поле для создания подключения к БД

{

Host = "localhost",

Port = **5432**,

Database = "laba\_2",

Username = "postgres",

Password = "postgres"

}.ConnectionString;

**public** **Form1**()

{

InitializeComponent();

btnEdit.Enabled = **false**; // ограничения на доступ к кнопкам работы с данными в начале работы приложения

btnDelete.Enabled = **false**;

btnAdd.Enabled = **false**;

ShowUsersList(); // вывод данных из БД

}

**private** **void** **ShowUsersList**() // метод для вывода данных из БД

{

lvUsers.Items.Clear();

**using** (**var** conn = **new** NpgsqlConnection(connStr))

{

conn.Open();

**using** (**var** sqlCommand = **new** NpgsqlCommand

{

Connection = conn,

CommandText = @"SELECT id, login, time FROM users" // запрос на получение данных из БД

})

{

**var** reader = sqlCommand.ExecuteReader();

**while** (reader.Read())

{

**var** id = (**int**)reader["id"]; // читаем id

**var** login = (**string**)reader["login"]; // читаем логин

**var** registration\_date = ((DateTime)reader["time"]).ToString("d"); // читаем время регистрации

**var** lvuser = **new** ListViewItem($"{login} ({registration\_date})") // в поле tag обьекта ListViev для пользователя кладём id

{

Tag = id

};

lvUsers.Items.Add(lvuser); // добавляем пользователя

}

conn.Close();

}

}

}

**private** **void** **btnRefresh\_Click**(**object** sender, EventArgs e) // обработка кнопки обновления списка данных в форме

{

ShowUsersList();

btnEdit.Enabled = **false**; // блокируем кнопки тк выделение слетает при обновлении формы

btnDelete.Enabled = **false**;

}

**private** **void** **btnEdit\_Click**(**object** sender, EventArgs e) // изменени полей о пользователе

{

**var** login = tbLogin.Text;

**var** reg\_date = dtpRegDate.Value;

**var** password = tbPassword.Text;

**var** password\_repeat = tbRepeatedPassword.Text;

**var** base\_pas = "";

**var** user\_id = (**int**)lvUsers.SelectedItems[**0**].Tag;

**using** (**var** conn = **new** NpgsqlConnection(connStr))

{

conn.Open();

**using** (**var** sqlCommand = **new** NpgsqlCommand // формируем завпрос получения пароля пользователя

{

Connection = conn,

CommandText = @"SELECT password

FROM users

WHERE id=@id"

})

{

sqlCommand.Parameters.AddWithValue("id", user\_id); // используем параметр дабы защитится от sql иньекций

**var** reader = sqlCommand.ExecuteReader();

**if** (reader.Read())

{

base\_pas = (**string**)reader["password"];

}

**else**

{

base\_pas = "";

state\_label.Text = "Ошибка, пользователь не найден";

conn.Close();

**return**;

}

conn.Close();

}

}

**if** (!BCrypt.Net.BCrypt.Verify(password, base\_pas) && password!="") // проверяем отличается ли введеный пароль от хранящегося в бд

{

base\_pas = BCrypt.Net.BCrypt.HashPassword(password); // если отличается то получаем новую хэш строку для него

}

**using** (**var** conn = **new** NpgsqlConnection(connStr)) // формируем новый запрос на изменение данных( сновым подключением)

{

conn.Open();

**using** (**var** sqlCommand = **new** NpgsqlCommand

{

Connection = conn,

CommandText = @"UPDATE users

SET login = @login, password = @password, time = @reg\_date

WHERE id= @id"

})

{

sqlCommand.Parameters.AddWithValue("id", user\_id);// используем параметр дабы защитится от sql иньекций

sqlCommand.Parameters.AddWithValue("login", login);

sqlCommand.Parameters.AddWithValue("reg\_date", reg\_date);

sqlCommand.Parameters.AddWithValue("password", base\_pas);

**if** (sqlCommand.ExecuteNonQuery() == **0**)

{

state\_label.Text = "Ошибка, пользователь не найден";

conn.Close();

}

conn.Close();

}

}

}

**private** **void** **lvUsers\_SelectedIndexChanged**(**object** sender, EventArgs e) // проверка выделеных пользователей в форме

{

**var** state\_select\_items = lvUsers.SelectedItems.Count > **0**; // проверка выделили ли мы хоть кого нибудь

btnDelete.Enabled = state\_select\_items;

**if** (state\_select\_items) // если да

{

**int** user\_id = (**int**)lvUsers.SelectedItems[**0**].Tag; // получаем id из тэга

**using** (**var** conn = **new** NpgsqlConnection(connStr))

{

conn.Open();

**using** (**var** sqlCommand = **new** NpgsqlCommand // формируем запрос на получение логина и времени

{

Connection = conn,

CommandText = @"SELECT login, time

FROM users

WHERE id=@id"

})

{

sqlCommand.Parameters.AddWithValue("id", user\_id);// используем параметр дабы защитится от sql иньекций

**var** reader = sqlCommand.ExecuteReader();

**if** (reader.Read()) // мы их читали что бы потом сунуть в текст-бокс что бы пользователь мог понять с какими именно данными имеет дело

{

tbLogin.Text = (**string**)reader["login"];

dtpRegDate.Value = (DateTime)reader["time"];

}

conn.Close();

}

}

}

}

**private** **void** **btnAdd\_Click**(**object** sender, EventArgs e) // добавление пользователя

{

**using** (**var** conn = **new** NpgsqlConnection(connStr))

{

conn.Open();

**var** login = tbLogin.Text;

**using** (**var** sqlCommand = **new** NpgsqlCommand // создаём запрос на проверку введенного логина, то есть есть ли такой логин в базе

{

Connection = conn,

CommandText = @"SELECT COUNT(\*)

FROM users

WHERE login = @userLogin"

})

{

sqlCommand.Parameters.AddWithValue("@userLogin", login);// используем параметр дабы защитится от sql иньекций

**if** ((**long**)sqlCommand.ExecuteScalar() > **0**)

{

state\_label.Text = "Логин уже занят";

**return**;

}

state\_label.Text = "";

}

**var** password = tbPassword.Text;

**var** passwordHash = BCrypt.Net.BCrypt.HashPassword(password); // хэшируем пароль

**using** (**var** sqlCommand = **new** NpgsqlCommand // запрос на вставку пользователя в БД

{

Connection = conn,

CommandText = @"INSERT INTO users (login, password)

VALUES (@login, @passwordHash)"

})

{

sqlCommand.Parameters.AddWithValue("@login", login);// используем параметр дабы защитится от sql иньекций

sqlCommand.Parameters.AddWithValue("@passwordHash", passwordHash);

**if** (sqlCommand.ExecuteNonQuery() == **1**)

{

state\_label.Text = "Пользователь успешно зарегистрирован";

}

**else**

{

state\_label.Text = "Ошибка регистрации. Попробуйте позже";

}

}

conn.Close();

}

}

**private** **void** **flowLayoutPanel1\_MouseClick**(**object** sender, MouseEventArgs e) // защита от невалидных значений

{

**if**(tbLogin.Text == "") // проверка на то что все поля не пустые

{

state\_label.Text = "Заполните все поля";

btnAdd.Enabled = **false**;

btnEdit.Enabled = **false**;

**return**;

}

**if**(!**string**.Equals(tbPassword.Text,tbRepeatedPassword.Text)) //пароли совпадают

{

state\_label.Text = "Пароль не совпадает с повторением";

btnAdd.Enabled = **false**;

btnEdit.Enabled = **false**;

**return**;

}

//проверка валидностьи пароля

**var** password1 = tbPassword.Text;

**var** legal\_elem1 = "qwertyuiopasdfghjklzxcvbnm";

**var** legal\_elem2 = "QWERTYUIOPASDFGHJKLZXCVBNM";

**var** legal\_elem3 = "1234567890";

**var** legal\_elem4 = "/|!@#$%^&\*(){}[]<>.,`~;:'+-\_";

**var** a = legal\_elem1.Except(password1);

**var** b = legal\_elem2.Except(password1);

**var** c = legal\_elem3.Except(password1);

**var** d = legal\_elem4.Except(password1);

**if** (!(a.Count() < legal\_elem1.Count() && b.Count() < legal\_elem2.Count() && c.Count() < legal\_elem3.Count()

&& d.Count() < legal\_elem4.Count()) && tbPassword.Text!="")

{

state\_label.Text = "В пароле должны присутствовать прописные и заглавные латинские буквы цифры и знаки /|!@#$%^&\*(){}[]<>.,`~;:'+-\_";

**return**;

}

**using** (**var** conn = **new** NpgsqlConnection(connStr)) // создание запроса на проверку был ли такой логин

{

conn.Open();

**using** (**var** sqlCommand = **new** NpgsqlCommand

{

Connection = conn,

CommandText = @"SELECT COUNT(\*)

FROM users

WHERE login=@login"

})

{

sqlCommand.Parameters.AddWithValue("login", tbLogin.Text);// используем параметр дабы защитится от sql иньекций

**if** ((**long**)sqlCommand.ExecuteScalar() > **0**)

{

state\_label.Text = "Логин занят";

btnAdd.Enabled = **false**;

btnEdit.Enabled = **false**;

**return**;

}

state\_label.Text = "";

conn.Close();

}

}

**if** (tbPassword.Text == "" || tbRepeatedPassword.Text == "")

{

state\_label.Text = "Заполните все поля";

btnAdd.Enabled = **false**;

}

**else**

{

btnAdd.Enabled = **true**;

}

**if** (lvUsers.SelectedItems.Count > **0**) // проверка на наличие выделенных полей

{

btnEdit.Enabled = **true**;

}

**else**

{

**return**;

}

state\_label.Text = "";

}

**private** **void** **btnDelete\_Click**(**object** sender, EventArgs e) //удаление

{

**using** (**var** conn = **new** NpgsqlConnection(connStr))

{

conn.Open();

**for** (**int** i = **0**; i < lvUsers.SelectedItems.Count; i++) //последовательно кидаем запросы на удаление всех выделенные записи

{

**using** (**var** sqlCommand = **new** NpgsqlCommand

{

Connection = conn,

CommandText = @"DELETE

FROM users

WHERE id=@id"

})

{

sqlCommand.Parameters.AddWithValue("id", lvUsers.SelectedItems[i].Tag);// используем параметр дабы защитится от sql иньекций

**if** (sqlCommand.ExecuteNonQuery() != **0**)

{

state\_label.Text = "Пользователь успешно удалён";

}

**else**

{

state\_label.Text = "Произошла ошибка";

}

}

}

conn.Close();

}

}

}

}

# 5 Тестирование программы

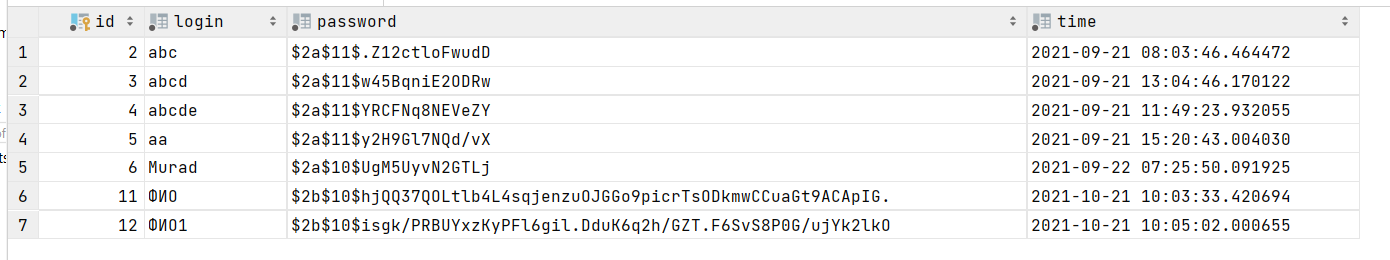
Тестирование программы можно разделить на 7 случаев:

1. Регистрация пользователя
2. Изменение полей пользователя
3. Отображение полей при изменении БД
4. Удаление пользователя
5. Удаление нескольких пользователей
6. Попытка ввести невалидные параметры при добавлении
7. Попытка ввести невалидные параметры при изменении
8. Попытка создания sql инъекции.

Тест 1(тип 1)

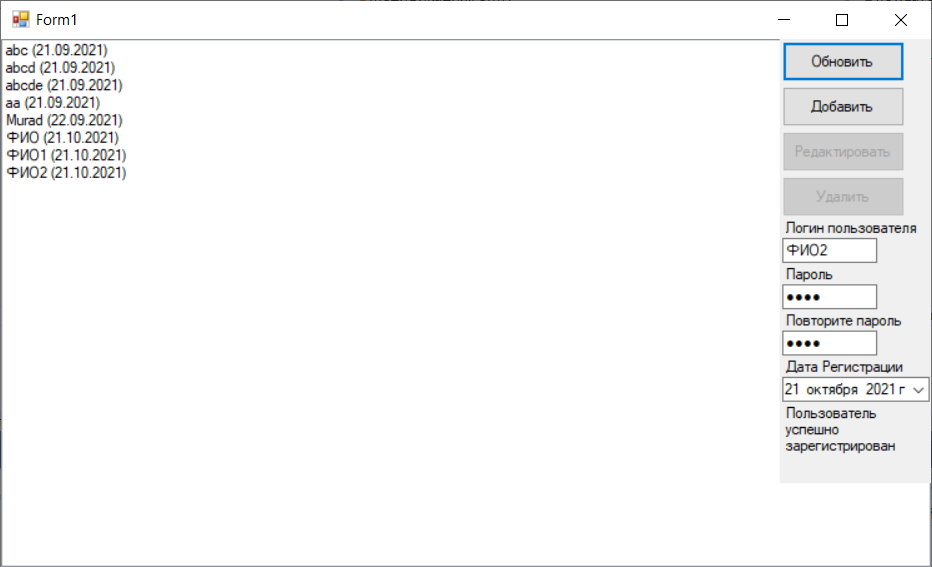
Входные данные:  
Логин: ФИО2

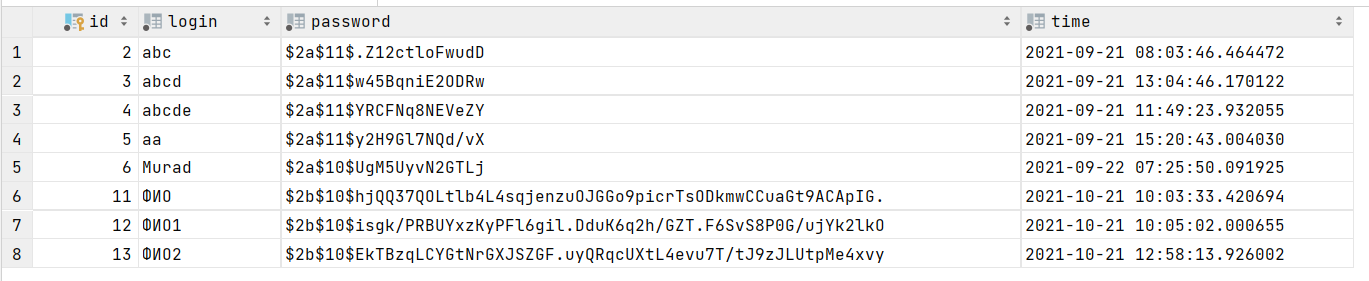
Пароль:A1+a



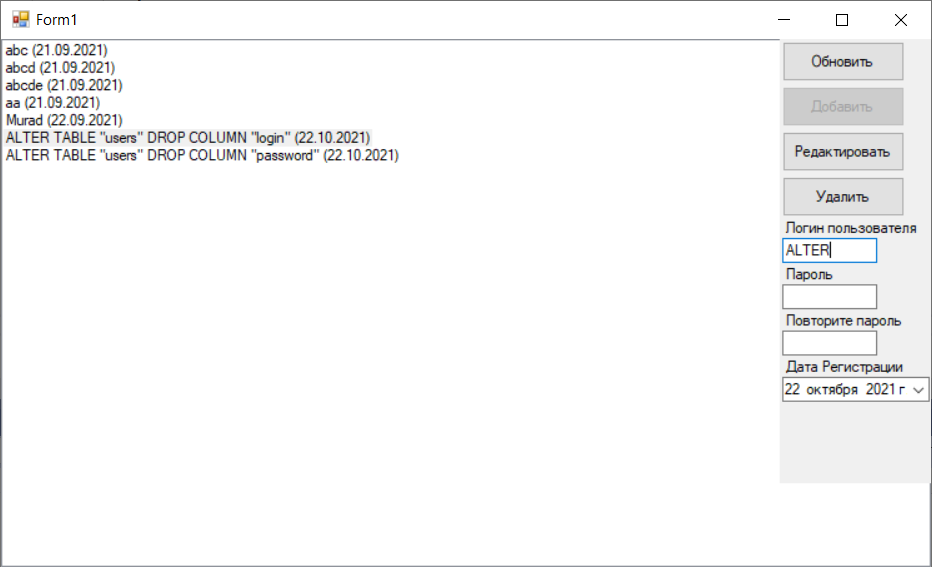
Ожидаемый результат: регистрация нового пользователя

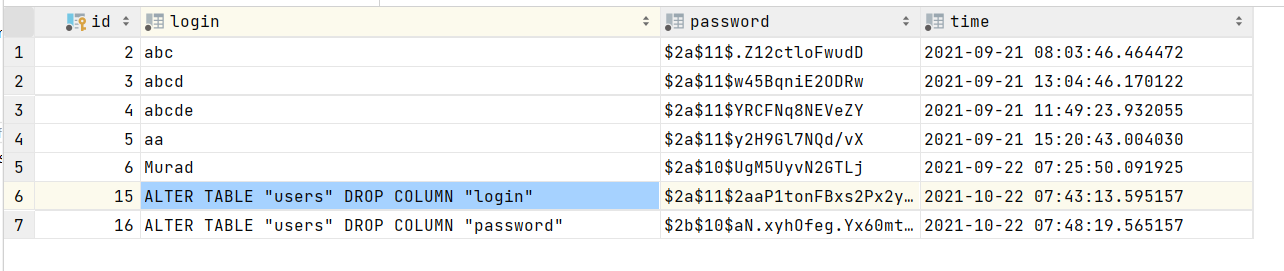
Выходной результат:





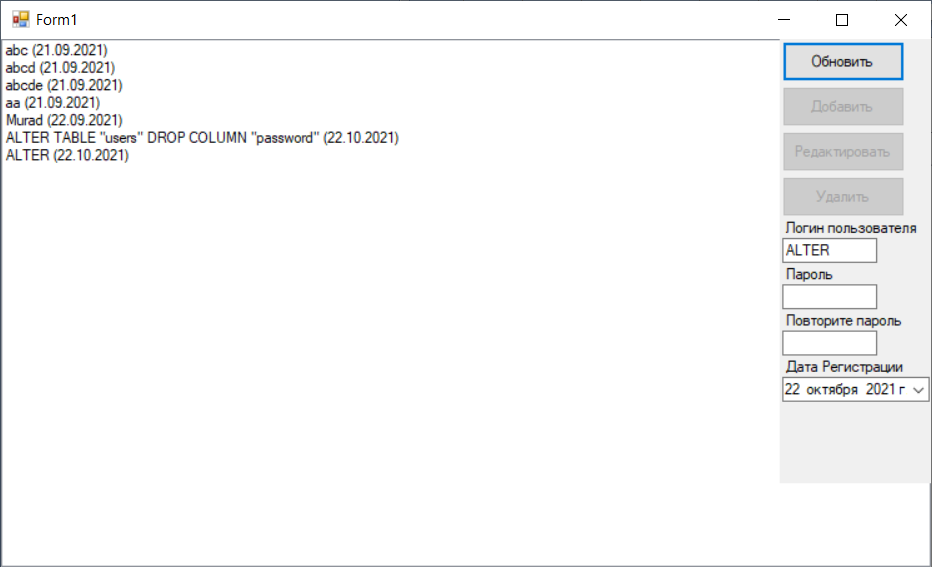
Тест 2(тип 2)  
Входные данные:

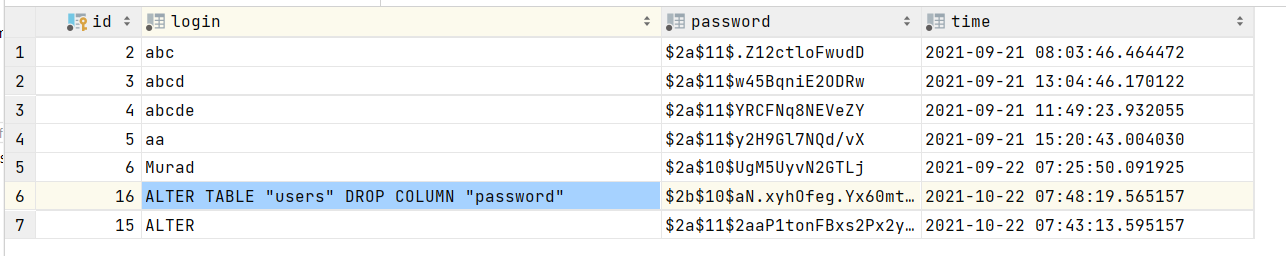




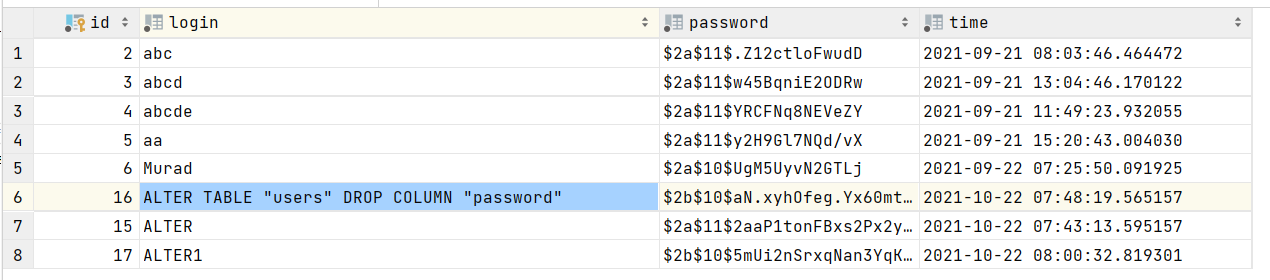
Ожидаемый результат: изменение данных пользователя

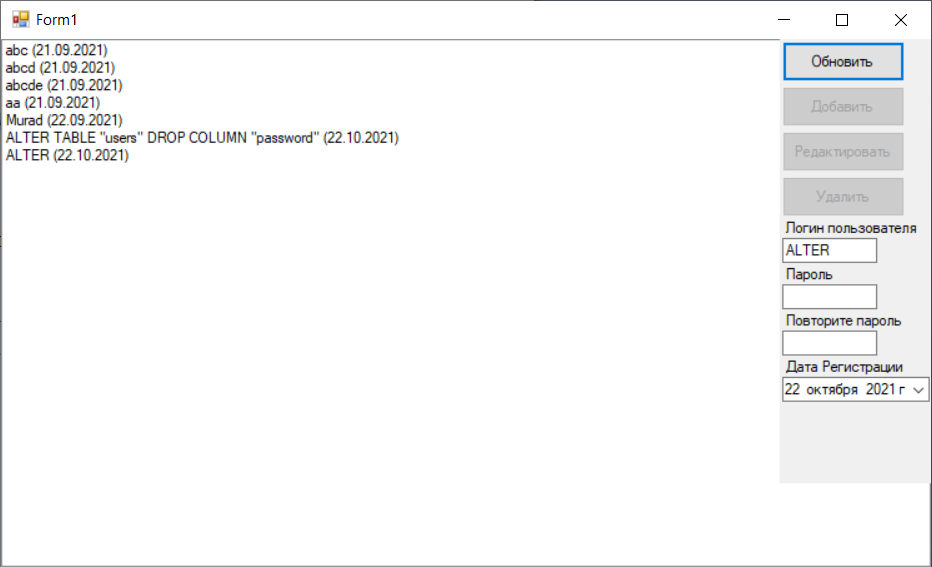
Выходной результат:





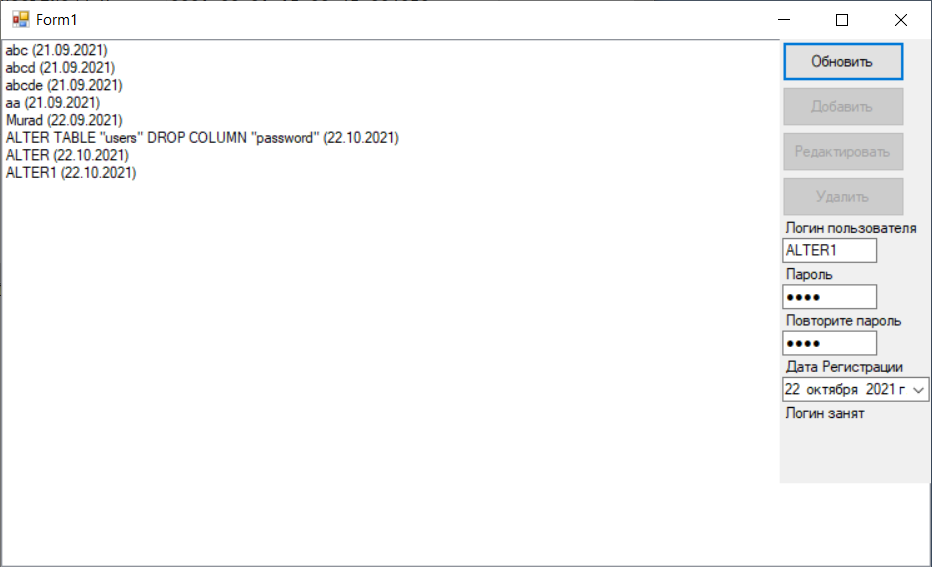
Тест 3(тип 3)  
Входные данные:





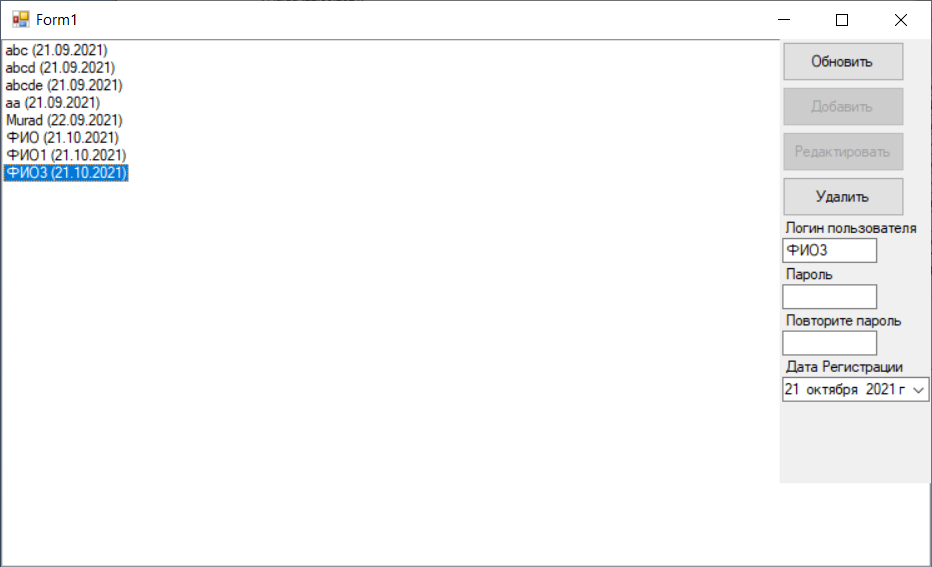
Ожидаемый результат: обновление списка пользователей.

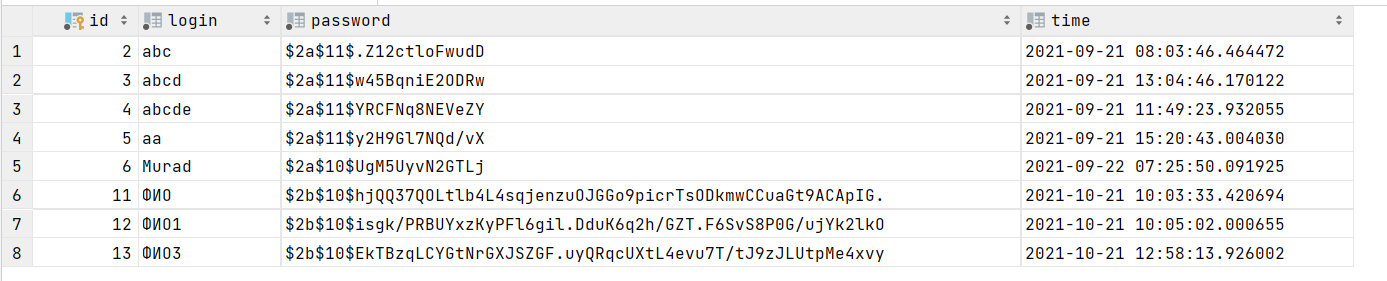
Выходной результат:



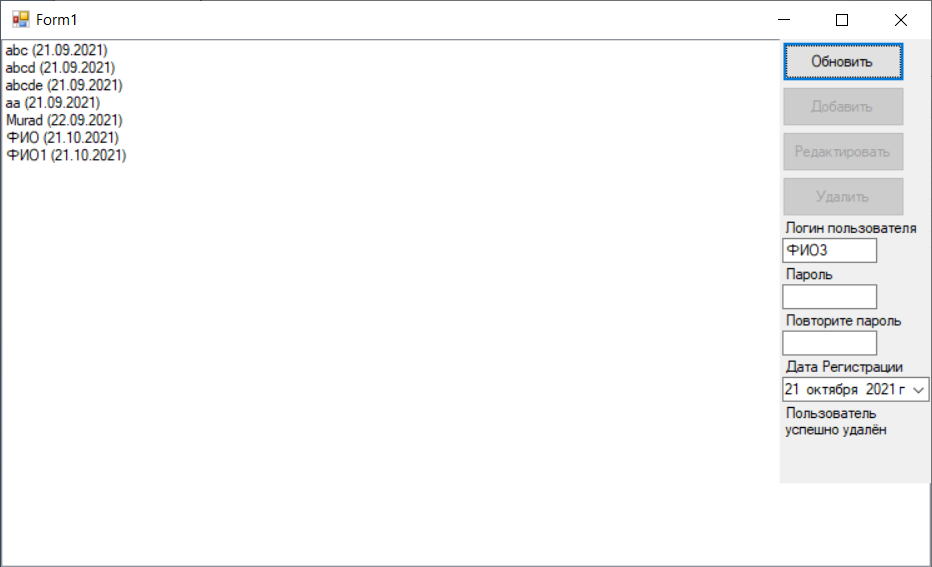
Тест 4(тип 4)

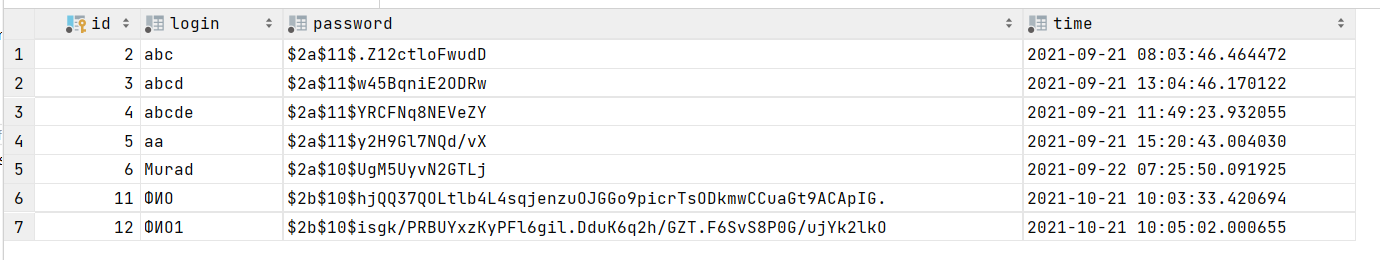
Входные данные:



Ожидаемый результат: успешное удаление пользователя.

Выходной результат:



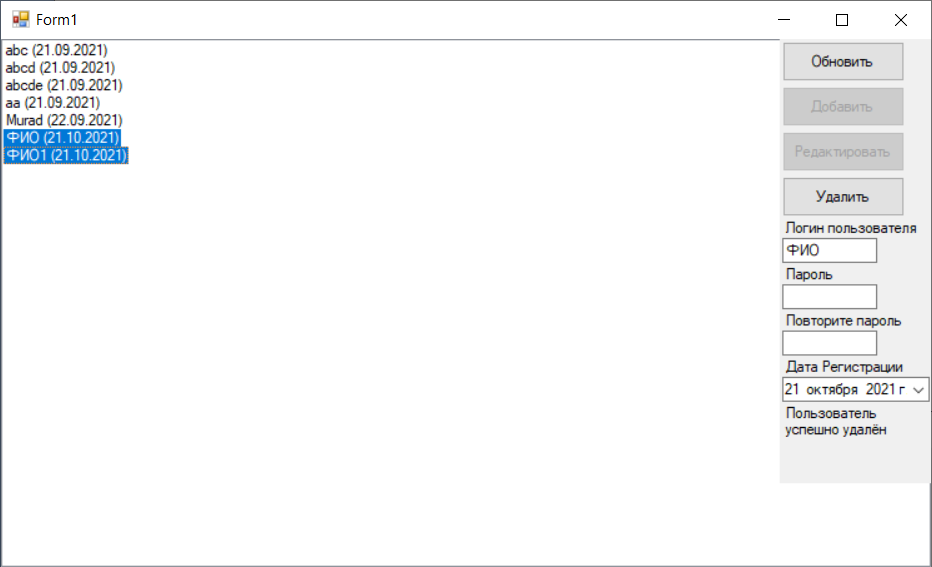


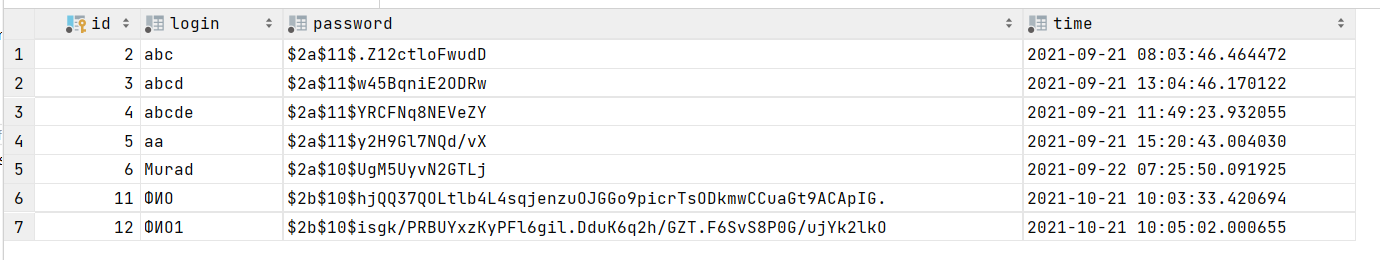
Тест 5(тип 5)

Входные данные:

Логин:

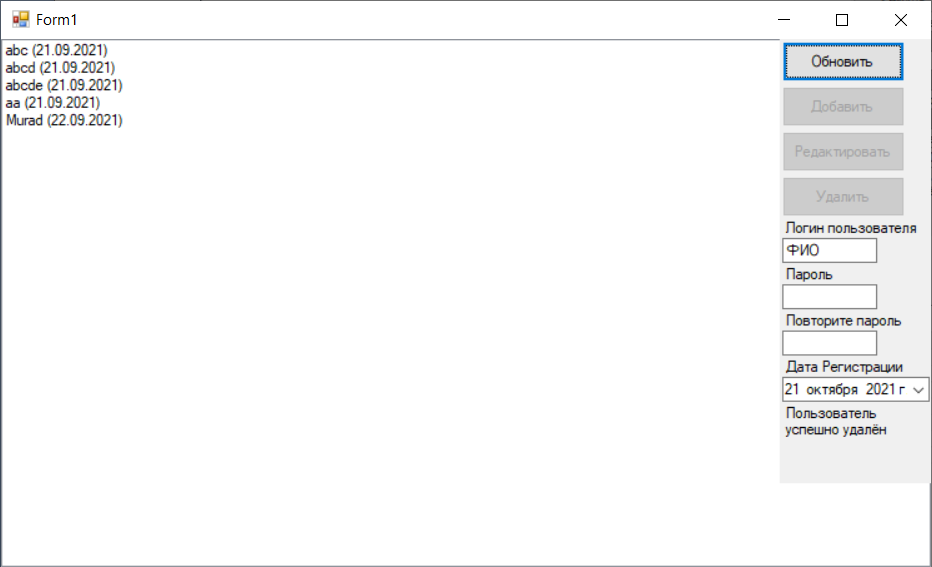
Пароль:





Ожидаемый результат: Удаление нескольких пользователей.

Выходной результат:





Тест 6(тип 6)

Входные данные:

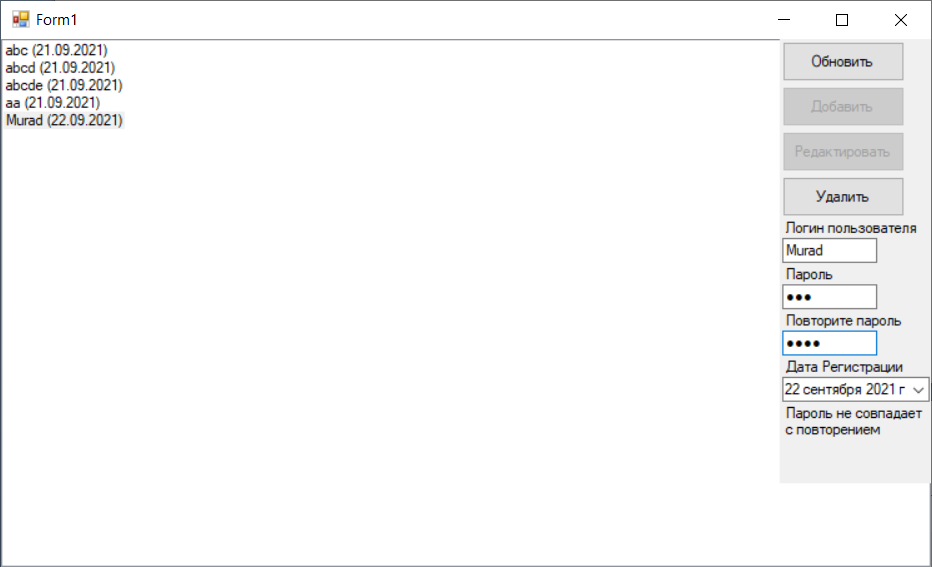
Логин:

Пароль: ффф

Повторение пароля: фффф

Ожидаемый результат: сообщение о невалидностьи параметров

Выходной результат:



Тест 7(тип 7)

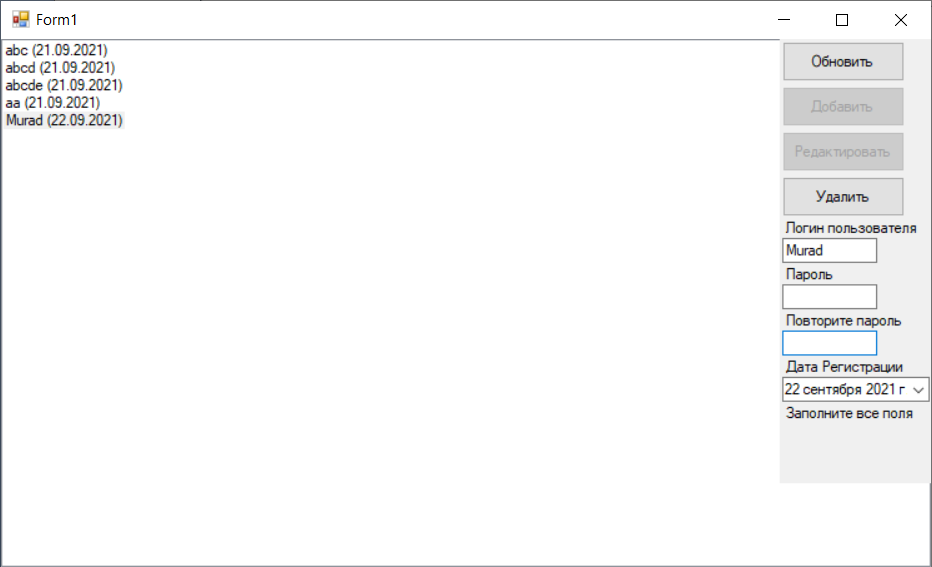
Входные данные:

Логин:

Пароль:

Ожидаемый результат: сообщение о пустых значениях.

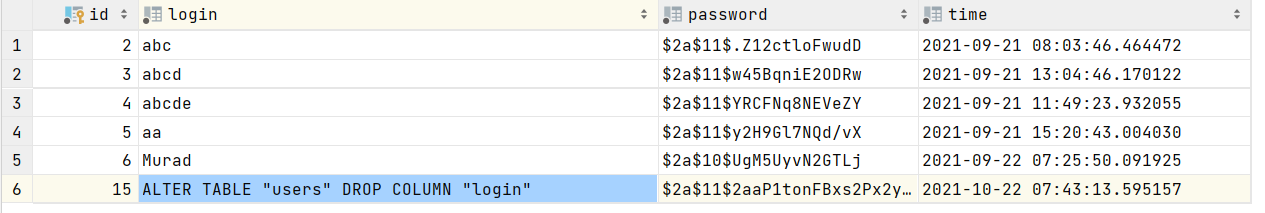
Выходной результат:



Тест 8(тип 8)

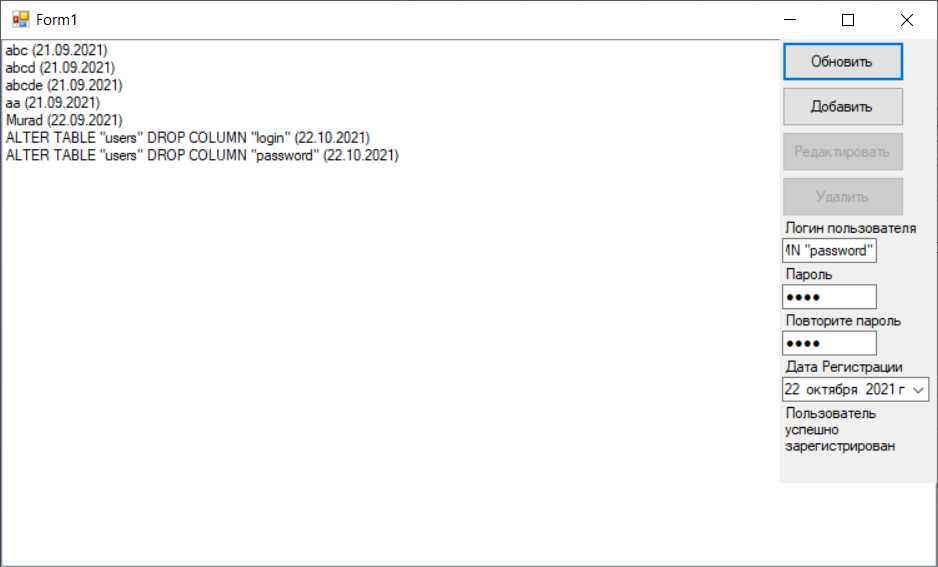
Входные данные:  
Логин: ALTER TABLE "users" DROP COLUMN "password"

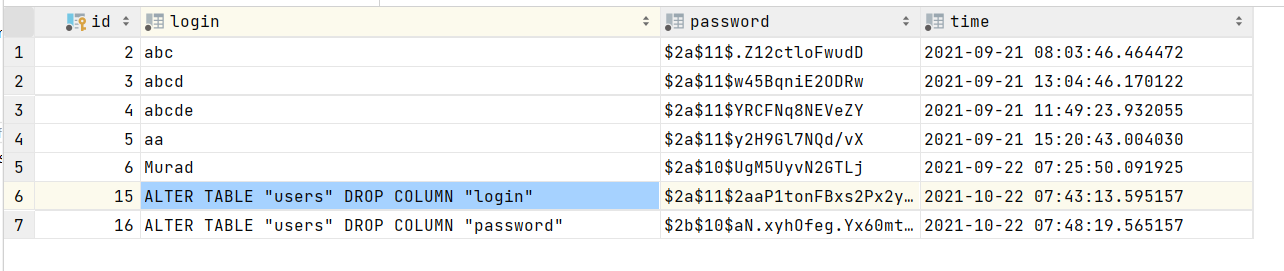
Пароль:A1+a



Ожидаемый результат: регистрация нового пользователя

Выходной результат:





Вывод: Программа успешно прошла тестирование, работает корректно.