|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ | | | | | | | | |
|  | | |  | | |  | | |
| ФГБОУ ВО «Пермский государственный  национальный исследовательский университет» | | | | | | | | |
|  | | |  | | |  | | |
|  | | ОТЧЕТ  по лабораторной работе № 2 «Форма регистрации и аутентификации «Учебная практика по БД и СУБД» | | | | |  | |
|  | | |  | | |  | | |
|  | Работу выполнил  студент гр. \_\_\_\_\_\_\_\_\_\_\_\_\_\_  Зимин И. В.\_\_\_\_\_\_\_\_\_\_\_\_  (подпись)  «\_\_\_» \_\_\_\_\_\_\_\_\_ 2018 | | |  | Проверил  \_\_\_\_\_\_\_\_\_\_\_ кафедры МОВС  (доц., ст. преп., асс.)  Постаногов И. С.\_\_\_\_\_\_\_  (подпись)  «\_\_\_» \_\_\_\_\_\_\_\_ 2018 | | |  |
|  |  | | |  |  | | |  |
| Пермь 2018 | | | | | | | | |

1 Постановка задачи

Написать программу, которая на графическом интерфейсе пользователя позволяет зарегистрировать нового пользователя с логином и паролем (т.е. занести соответствующую запись в базу данных). Дата регистрации должна сохраняться в базу данных. Пароль должен сохраняться в виде хэш-кода, зависящего от сгенерированного для этого пользователя модификатора (соли). Создание пользователя с существующим логином запрещено.

Написать программу, которая на графическом интерфейсе пользователя позволяет аутентифицировать ранее зарегистрированного пользователя.

Провести тестирование корректности работы программы.

Написать отчёт о проделанной работе.

2 Анализ предметной области

Для работы приложения требуется всего одна таблица. Я назвал её «Users», в ней хранятся данные о пользователях:

* Логин: вводится пользователем и напрямую попадает в БД
* Пароль: вычисляется автоматически с использованием соли и введенным при регистрации паролем и хранится в виде хэш-кода.
* Соль: хранится в БД для аутентификации пользователя.
* Дата регистрации пользователя: вставляется автоматически при корректном вводе остальных данных.

Структура таблицы представлена в таблице 2.1.

Поле Login уникально, что позволяет не использовать дополнительное поле с идентификатором, оно имеет тип nvarchar(50), так как длина логина конкретного пользователя может существенно различаться по длине.

Поле Password имеет постоянную длину 40 символа, так как для хеширования пароля используется хэш-функция SHA1, выходной поток которой составляет 20 байт и после конвертации в строку принимает размер 40 символов.

Поле Salt имеет постоянную длину 40 символа и создаётся программным путём с помощью криптостойкой последовательности байтов.

Поле RegistrationDate имеет соответствующий тип date, так как поле содержит нечто иное, как конкретную дату.

Важным аспектом является то, что все поля не допускают значение NULL, так как отсутствие хоть одного из них приведёт к неправильной работе программы.

Таблица 2. - Структура таблицы «Users»

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Разрешить NULL |
| Login | nvarchar(50) | нет |
| Password | char(40) | нет |
| Salt | char(40) | нет |
| RegistrationDate | date | нет |

3 Описание функционала разработанной программы

Приложение представляет собой две маленькие формочки: регистрации и входа.

Форма регистрации содержит: два поля ввода, один для логина, другой для пароля, две кнопки: «Зарегистрироваться», и «Войти», для перехода к форме входа, присутствует возможность скрыть пароль. Данная форма представлена на рисунке 3.1

Форма входа содержит так же два поля ввода: для логина и пароля, одну кнопку «Войти» и так же присутствует возможность скрыть пароль. Данная форма изображена на рисунке 3.2.

При запуске приложения отображается форма регистрации. Кнопка «Зарегистрироваться» стает активной только тогда, когда пользователь заполнил оба поля ввода и логин ещё никем не используется. При нажатии на кнопку «Войти» открывается форма входа, где аналогичным образом, кнопка «Войти» стает активной лишь когда пользователь заполнил оба поля ввода. При вводе некорректных данных выводятся соответствующие сообщения.

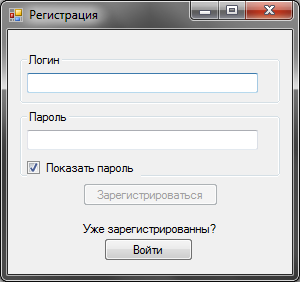


Рисунок 3.1 - Форма регистрации

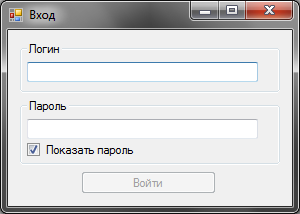


Рисунок 3.2 – Форма аутентификации

4 Описание алгоритма

1. Регистрация пользователя:

* Шаг 1. Проверяем, есть ли уже пользователь в таблице с введенным логином. Для этого, при каждом вводе символа в поле просматриваем записи в таблице. При выполнении данной проверки, должно возвращаться число – кол-во идентификаторов записей, у которых значение поля логина равняется введенном логину в приложении. Если вернулось число больше нуля, то выводим ошибку «Логин занят» и блокируем кнопку, иначе – запускаем проверку на корректность введенных данных (число символов в каждом поле больше нуля, логин не занят).
* Шаг 2. Если оба поля заполнены корректно, разблокируем кнопку «Зарегистрироваться».
* Шаг 3. Обработка нажатия кнопки «Зарегистрироваться». При нажатии на кнопку необходимо добавить в таблицу с данными новую запись, передав следующие параметры: логин, пароль, дата, соль.
  + - 1. Генерируем соль
         1. В качестве пароля передаем строку, созданную с помощью алгоритма хеширования SHA1 из конкатенации строк введенного пароля и сгенерированной соли.

1. Если запись успешно добавлена, то выводим сообщение «Вы зарегистрированы», блокируем кнопку «Зарегистрироваться», очищаем поля ввода.
2. Аутентификация пользователя

* Шаг 1. Проверяем корректность введенных данных: число символов в каждом поле больше нуля;
* Шаг 2. Если оба поля заполнены корректно, разблокируем кнопку «Войти».
* Шаг 3. Обработка нажатия кнопки «Войти». При нажатии на кнопку необходимо найти такую запись в таблице, что введенный логин равен значению поля логина в таблице, и вернуть значения хешированного пароля и соли.

Если такая запись найдена, то получаем значения пароля и соли данной записи.

Проверяем, равно ли значение хешированного пароля из таблицы строке, созданной с помощью алгоритма хеширования SHA1 из конкатенации строк введенного пароля и полученной соли из таблицы.

Если условие выполнено, то выводим сообщение «Вход выполнен», блокируем кнопку, очищаем поля ввода. Иначе же, выводим ошибку с сообщением «Неверный пароль».

Если запись не найдена, то выводим ошибку с сообщением «Пользователь с таким логином не найден».

5 Описание SQL-запросов

В программе используется три SQL-запроса. Два SELECT-запроса и один INSERT -запрос.

Для регистрации пользователя требуется осуществить 2 запроса:

Первый представляет из себя INSERT – запрос, который необходим для вставки данных, введенных пользователем, в БД. Он имеет следующий вид:

INSERT INTO Users(Login, Password, Salt, RegistrationDate)

VALUES (@curLogin, @curPassword, @curSalt, @curRegDate)

Данный запрос выполняется при нажатии кнопки «Зарегистрироваться» и вставляет в БД полученный из диалогового окна логин, сгенерированную соль, хэш пароля и текущую дату.

Второй запрос требуется для проверки корректности ввода логина, выглядит он следующим образом:

SELECT COUNT(\*) FROM Users WHERE Login = @currentLogin

Данный запрос позволяет определить количество записей в БД с данным логином и так как значение логина в БД уникально запрос может вернуть либо 1, это будет означать, что данный логин в БД уже существует и его нельзя использовать для регистрации, либо 0, это будет означать, что логина в БД нет, и им можно воспользоваться.

Для авторизации пользователя требовалось осуществить один запрос:

Этот запрос необходим для получения данных из БД для проверки правильности введенных данных: логина/пароля. Зарос имеет следующий вид:

SELECT Password, Salt FROM Users WHERE Login = @currentLogin

Данный запрос выполняется при нажатии кнопки «Войти» и в случае корректности ввода логина осуществляет выборку соли и хешированного пароля, соответствующих данному логину.

6 Реализация на C#

Формирование строки для подключения к базе данных выполняется с помощью класса SqlConnectionStringBuilder.

Далее нам нужно открыть подключение к базе данных при помощи класса SqlConnection.

Описание команды при помощи класса SqlCommand, имеющего параметры: сonnection – в нём хранится объект, над которым будут совершаться действия; commandtext – храниться запрос к базе данных. Команда используется для выборки данных.

SqlCommand Command = new SqlCommand()

Хеширование «соленого пароля» производится при помощи объекта класса SHA1CryptoServiceProvider.

SHA1CryptoServiceProvider SHA1 = new SHA1CryptoServiceProvider()

Выходной поток составляет 160 бит, после чего при помощи метода BitConverter преобразуем полученный хэш к строке, где каждый байт преобразовывается в пару шестнадцатеричных цифр, все пары разделены дефисами, в итоге получаем хэш длиной 59 символов и с помощью метода Replace удаляем дефисы. В итоге остаётся строка из 16-ричных цифр, длина которой равна 40 символам.

BitConverter.ToString(SHA1.ComputeHash(Encoding.UTF8.GetBytes(text))).Replace("-", "");

Для получения случайной соли используется байтовый массив, размерности 20, которую я выбрал исходя из длины хэша, чтобы с последующими преобразованиями данного массива к строке, получить те же 40 символов. Метод GetBytes класса RNGCryptoServiceProvider заполняет массив криптостойкой последовательностью случайных чисел. Далее полученный массив байтов преобразуется к строке и возвращается в основную программу.

RNGCryptoServiceProvider rng = new RNGCryptoServiceProvider();

byte[] buff = new byte[20];

rng.GetBytes(buff);

С помощью метода BitConverter мы преобразуем массив байтов к строке. Далее при помощи Replace убираем все символы «-».

BitConverter.ToString(buff).Replace("-", "");

Далее описаны события на нажатие кнопок Регистрации и Аутентификации на форме.

По нажатии на кнопку регистрации генерируется соль, считается хэш по строке, состоящей из введенного пароля плюс соль, все это, а также логин и дата, полученная при помощи метода DateTime.Today вставляются в БД. При удачной вставке выводится на экран сообщение об успешной регистрации.

Для считывания данных из БД используется объект SqlCommand. Для передачи параметров в SQL запрос используется метод AddWithValue.

Для защиты от SQL инъекций используется параметризированный ввод.

sCommand.Parameters.AddWithValue("curLogin", tbLogin.Text);

sCommand.Parameters.AddWithValue("curPasswordHash", CalcHash(tbPassword.Text + salt));

sCommand.Parameters.AddWithValue("curSalt", salt);

sCommand.Parameters.AddWithValue("curRegDate", DateTime.Today);

Команда выполняется методом ExecuteNonQuery(), который чтобы проверить успешна ли регистрация пользователя.

Кнопка входа передаёт сопоставляет данные из формы ввода логина и пароля с данными из БД, которые получаются при помощи SELECT-запроса при условии равенства логина. Если логин нашёлся, то считается хэш составленный из введенного пароля плюс соль из БД и если полученный хэш равен хэшу из БД, то выдаётся сообщение о успешной аутентификации. Команда, отвечающая за проверку, выполняется методом ExecuteScalar(), который возвращает результат запроса.

7 Тестирование программы

Чтобы протестировать корректность работы программы можно все тесты разбить на 2 класса.

Класс №1: Тесты для регистрации

1. Оба поля ввода пусты
2. Пусто одно из полей
3. Ввод несуществующего логина и несуществующего пароля
4. Ввод существующего логина и несуществующего пароля
5. Ввод несуществующего логина, но существующего пароля.

Класс №2: Тесты для аутентификации

1. Оба поля ввода пусты
2. Пусто одно из полей
3. Ввод верного логина и верного пароля
4. Ввод верного логина и неверного пароля
5. Ввод неверного логина и верного пароля
6. Ввод неверного логина и неверного пароля

Для проверки данного набора тестов достаточно наполнить БД одной строкой Первый ввод: введем любой логин и любой пароль, тем самым покрыв тест 3 из класса №1.

|  |  |  |  |
| --- | --- | --- | --- |
| Login | Password | Salt | Date |
| Fred | 728F5B79CD46D312D8CA  A461A33BDB28EA0C578E | 945F95BCC4BDC8F08E93  B151561D67AF84C84692 | 2018-02-02 |

В столбце Password содержится хэш, полученный из пароля «I’mFred1990» и соли из столбца Salt.

Теперь мы можем проводить тесты 4-5 из класса №1 и тесты 3-6 из класса №2

Для тестов с пустыми полями не требуется наполнять БД. Кнопки «Войти» и «Зарегистрироваться» будут неактивны.

Программа на данных тестах отработала корректно. При необходимости были выведены соответствующие сообщения.