|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ  Пермский государственный национальный исследовательский университет | | | | | | |
|  | Отчет  по лабораторной работе №8 «Автоматизированное рабочее место»  по дисциплине «Учебная практика по базам данных» | | | | |  |
|  | |  | | |  | |
| Работу выполнил студент гр. ПМИ-1,2  Вахрушев А.О. | | |  | Проверил  Старший преподаватель кафедры математического  обеспечения вычислительных систем  Постаногов И.С. | | |
|  | |  | | |  | |
|  | | Пермь 2019 | | |  | |

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc6925782)

[2 Анализ предметной области 4](#_Toc6925783)

[3 Схема базы данных 5](#_Toc6925784)

[4 Описание алгоритма 7](#_Toc6925785)

[5 Реализация на C# 8](#_Toc6925786)

[6 Интерфейс 11](#_Toc6925787)

[7 Тестирование 17](#_Toc6925788)

1 Постановка задачи

Написать программу, которая на графическом интерфейсе пользователя позволяет просматривать, добавлять, удалять, изменять записи и осуществлять поиск минимум 2 стандартных запросов в 4 связанных таблицах из спроектированной ранее базы данных предметной области без необходимости задания и отображения пользователю искусственных ключей. Между указанными таблицами должно присутствовать как отношение типа 1:М, так и отношение типа М:М.

При запуске программы пользователю должна отображаться форма аутентификации по логину и паролю, допускающая гостевой вход. Из формы аутентификации должна быть возможность перейти на форму регистрации нового логина и пароля для оператора АРМ.

Пользователь, зашедший в приложение под учётной записью администратора АРМ, должен иметь возможность просматривать, добавлять, удалять, изменять записи в 4 используемых таблицах из спроектированной ранее базы данных предметной области и в таблице учётных записей пользователей. Пользователь, зашедший в приложение под учётной записью оператора АРМ, должен иметь возможность просматривать, добавлять, удалять, изменять записи только в 4 используемых таблицах из спроектированной ранее базы данных предметной области. Пользователь, зашедший в приложение под гостевой учётной записью АРМ, должен иметь возможность только просматривать записи в 4 используемых таблицах из спроектированной ранее базы данных предметной области.

В пользовательском интерфейсе модификации должна предоставляться возможность контекстного пополнения связанных сущностей.

Средства стандартного поиска доступны всем категориям пользователей.

Провести тестирование корректности работы программы.

2 Анализ предметной области

В качестве предметной области используются 4 таблицы из лабораторной работы по нормализации: site, news, content\_for\_site, content. А также дополнительная таблица пользователей: users.

Есть 3 вида пользователей: администратор, оператор, гость.

Гость может только просматривать все таблицы, кроме users.

Оператор может просматривать, добавлять, изменять, удалять записи во всех таблицах, кроме users.

Администратор имеет все возможности оператора, а также возможность просмотра, добавления, изменения, удаления записей в таблице users.

Нельзя удалить последнего администратора.

3 Схема базы данных

База данных (БД) состоит из пяти таблиц.

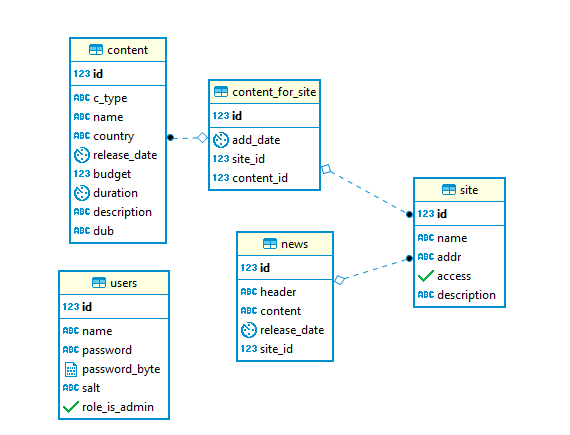


Рисунок 3.1 – Схема базы данных.

Таблица 3.1 – структура таблицы «site».

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название столбца | Тип | Not null | Unique | Primary key |
| id | serial | + | + | + |
| name | varchar(50) | + |  |  |
| addr | varchar(230) | + |  |  |
| access | bool | + |  |  |
| description | varchar(1500) | + |  |  |

Таблица 3.2 – структура таблицы «content\_for\_site».

| Название столбца | Тип | Not null | Unique | Primary key |
| --- | --- | --- | --- | --- |
| id | serial | + | + | + |
| add\_date | date | + |  |  |
| site\_id | int4 | + |  |  |
| content\_id | int4 | + |  |  |

Таблица 3.3 – структура таблицы «content».

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название столбца | Тип | Not null | Unique | Primary key |
| id | serial | + | + | + |
| name | varchar(50) | + |  |  |
| content\_type | varchar(50) | + |  |  |
| country | varchar(230) | + |  |  |
| release\_date | date | + |  |  |
| budget | int4 | + |  |  |
| duration | intreval | + |  |  |
| description | varchar(1500) | + |  |  |
| dub | varchar(500) | + |  |  |

Таблица 3.4 – структура таблицы «news».

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название столбца | Тип | Not null | Unique | Primary key |
| id | serial | + | + | + |
| header | varchar(50) | + |  |  |
| content | varchar(1500) | + |  |  |
| release\_date | date | + |  |  |
| site\_id | int4 | + |  |  |

Таблица 3.5 – структура таблицы «users».

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название столбца | Тип | Not null | Unique | Primary key |
| id | serial | + | + | + |
| name | varchar(50) | + |  |  |
| password | varchar | + |  |  |
| password\_byte | bytea | + |  |  |
| salt | varchar | + |  |  |
| role\_is\_admin | bool | + |  |  |

4 Описание алгоритма

Необходимые действия:

1. Описания форм.
2. Создания отображения БД в коде.
3. Настройка связи с БД.
4. Описание методы получения, добавления, изменения, удаления данных из БД.
5. Установка связи формы с данными из БД.
6. Установка связи действий на форме с методами для работы с БД.
7. Тестирование.

5 Реализация на C#

Создадим новое WPF приложение.

Подключим пакет для работы с БД, Npgsql.

Подключим пакет для шифрования паролей, BouncyCastle.

С помощью языка разметки xaml зададим формы и элементы на них. С помощью ключевого слова Binding устанавливаем связь элементов формы с логикой.

Создадим классы для элементов таблиц с соответствующими полями.

Создадим классы таблиц, в которых будут храниться работа с таблицей, чтение, добавление, изменение, удаление данных. Каждый этот класс в конструкторе принимает строку подключения.

Строка подключения имеет вид:

"Host=localhost;Port=5432;Database=andrey\_bd2;Username=postgres;Password=postgres"

И передается объекту класса NpgsqlConnection из пакета, который мы подключили для работы с БД.

Чтение данных.

Для получения нужных нам данных мы формируем SQL запрос к нашей БД. Код данного запроса представлен на листинге 5.1.

select id, c\_type, name, country, release\_date, budget, duration, description, dub from content;

Листинг 5.1 – получение данных.

В select мы перечисляем столбцы, значения которых хотим получить из БД. В from мы указываем таблицу, с нашими значениями.

Эта команда передается в объект класса NpgsqlCommand, который добавлен в подключенном пакете. Далее создаем объект для чтения. Будем читать построчно в режиме ExecuteReader().

Добавление данных.

Для добавления нужных нам данных мы формируем SQL запрос к нашей БД. Код данного запроса представлен на листинге 5.2.

insert into content (c\_type, name, country, release\_date, budget, duration, description, dub)

values (@c\_type, @name, @country, @release\_date, @budget, @duration, @description, @dub) returning id;

Листинг 5.2 – добавление данных.

После insert into мы указываем таблицу для вставки, затем в скобках перечисляем столбцы, после в values перечисляем значение, которое вставляем в БД, с помощью @ мы гарантируем, что переданные в запрос значения будут восприняты, как переменные, а не команды. С помощью returning мы возвращаем присвоенное данной записи значение id. Чтобы считать его нам хватит чтения просто ячейки с помощью ExecuteScalar().

Изменение данных.

Для изменения нужных нам данных мы формируем SQL запрос к нашей БД. Код данного запроса представлен на листинге 5.3.

update content set c\_type = @c\_type, name = @name, country = @country, release\_date = @release\_date, budget = @budget, duration = @duration, description = @description, dub = @dub where id = @id;

Листинг 5.3 – обновление данных.

После update мы указываем имя таблицы, записи в которой мы будем обновлять, после set мы указываем имена столбцов и значения, которые им присваиваем, после where мы указываем условие для обновляемой записи. Так как нам не важно возвращаемое значение используем ExecuteNonQuery().

Удаление данных.

Для удаления нужных нам данных мы формируем SQL запрос к нашей БД. Код данного запроса представлен на листинге 5.4.

delete from content where id = @id;

Листинг 5.4 – удаление данных.

После delete from мы указываем таблицу, из которой будем удалять запись, в where указываем условие, по которому записи будут удалены. Аналогично предыдущему используем ExecuteNonQuery().

Создадим класс, который будет хранить в себе работу с БД. В конструкторе он принимает строку подключения. Он будет иметь публичные свойства, по одному на каждую таблицу. Каждая таблицы содержит соответствующие функции чтения, добавления, изменения, удаления данных.

Создадим необходимые структуры данных в коде каждой формы. Также у форм установим интерфейс INotifyPropertyChanged, создадим необходимое событие и инициализируем его, это необходимо, чтобы класс описывающий логику формы, мог сообщать ей об изменениях своих свойств.

Для того, чтобы выполнение действий на форме приносило результат, создадим класс, RelayCommand, наследующий интерфейс ICommand. У каждой кнопки на форме зададим команду, которую она выполняет. В коде описывающим логику создадим соответствующие объекты класса RelayCommand.

Отдельные особенности реализации:

Проверка уникальности логина происходит просмотром всех пользователей в БД. Если бы мы не выгружали БД, мы бы могли использовать select и посчитать число записей полученных через ExecuteNonQuery().

Генерация хеш-кода пароля происходит с помощью класса DigestUtilities из подключенного пакета для шифрования. У него мы используем функцию CalculateDigest.

Проверка сложности пароля при регистрации из формы, а не через создание пользователя админом организована через linq запрос.

Все текстовые поля сравниваются и передаются в БД после обработки функцией .Trim(), чтобы избавиться от крайних пробелов.

Выполним тестирование программы.

Контекстное пополнение организовано через вызов уже созданных для обычного добавления форм.

Чтобы не создавать лишние формы было создано перечисление Mode, которое содержало обозначения Add и Edit, экземпляр этого перечисления передавался в конструктор каждой формы.

Заранее заготовленные запросы реализованы через select с условием where.

Проверка ввода пользователя организуется с помощью встроенных в WPF механизмов. Так как в WPF мы привязываем поля к свойствам в логике, то даже в случае, если в TextBox будет введена строка, когда ожидалось увидеть в нем число, то ошибка не возникнет, а сам TextBox будет подсвечен. Если же проигнорировать это сообщение, то внесенные изменения просто будут игнорироваться. В случае, если в поле для даты вводить строку, то встроенные функции WPF заменят эту строку на значение, установленное по умолчанию и ошибки также не будет. Ввод строк размером больше, чем выделенный для них диапазон в БД контролируется через свойство MaxLength.

6 Интерфейс

Представим пользовательский интерфейс.

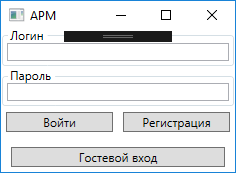
Форма регистрации и аутентификации представлена на рисунке 6.1.

Рисунок 6.1 – форма входа в АРМ.

Сам АРМ представлен на рисунках 6.2 – 6.5.

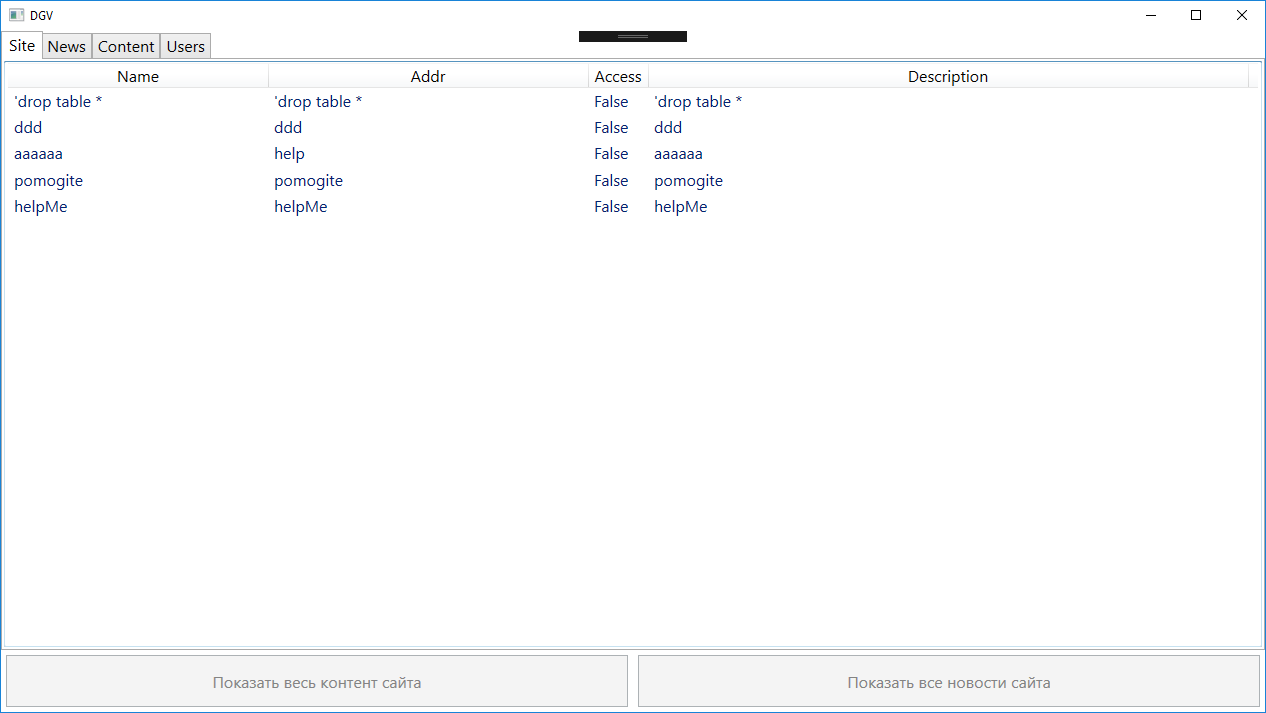


Рисунок 6.2 – представление таблицы сайтов.

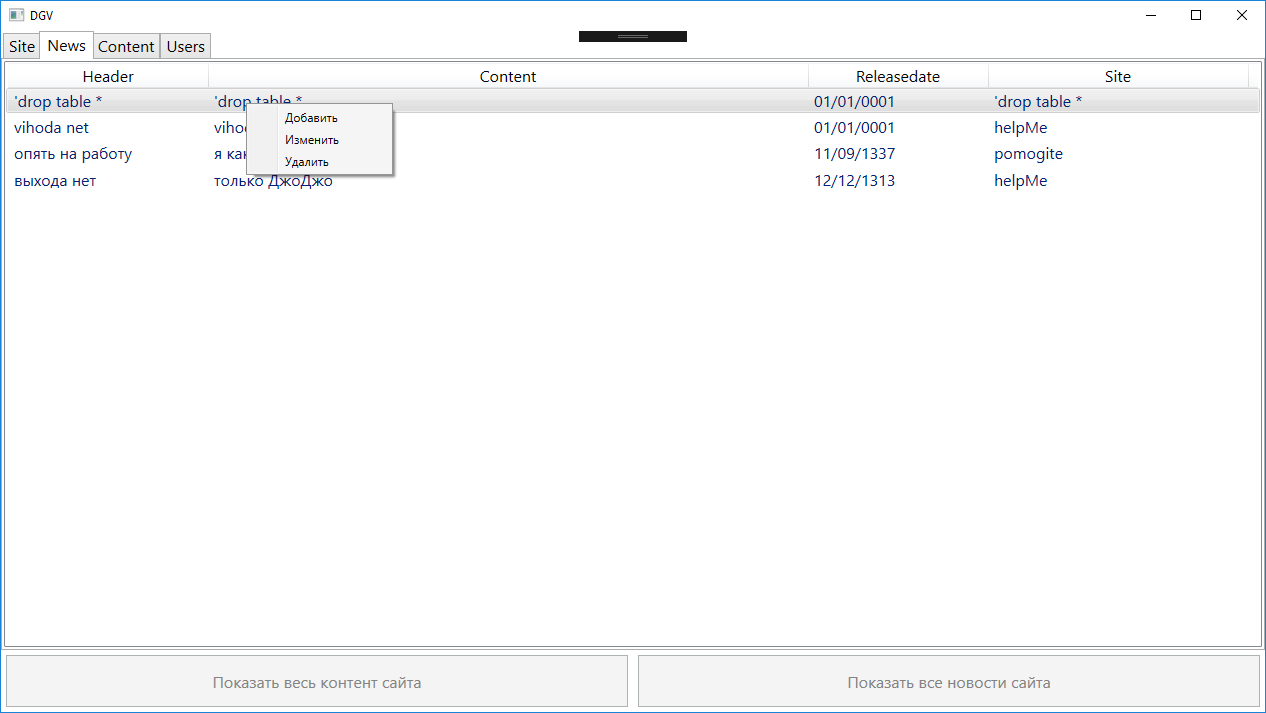


Рисунок 6.3 – представление таблицы новостей, а также доступные оператору и администратору функции.

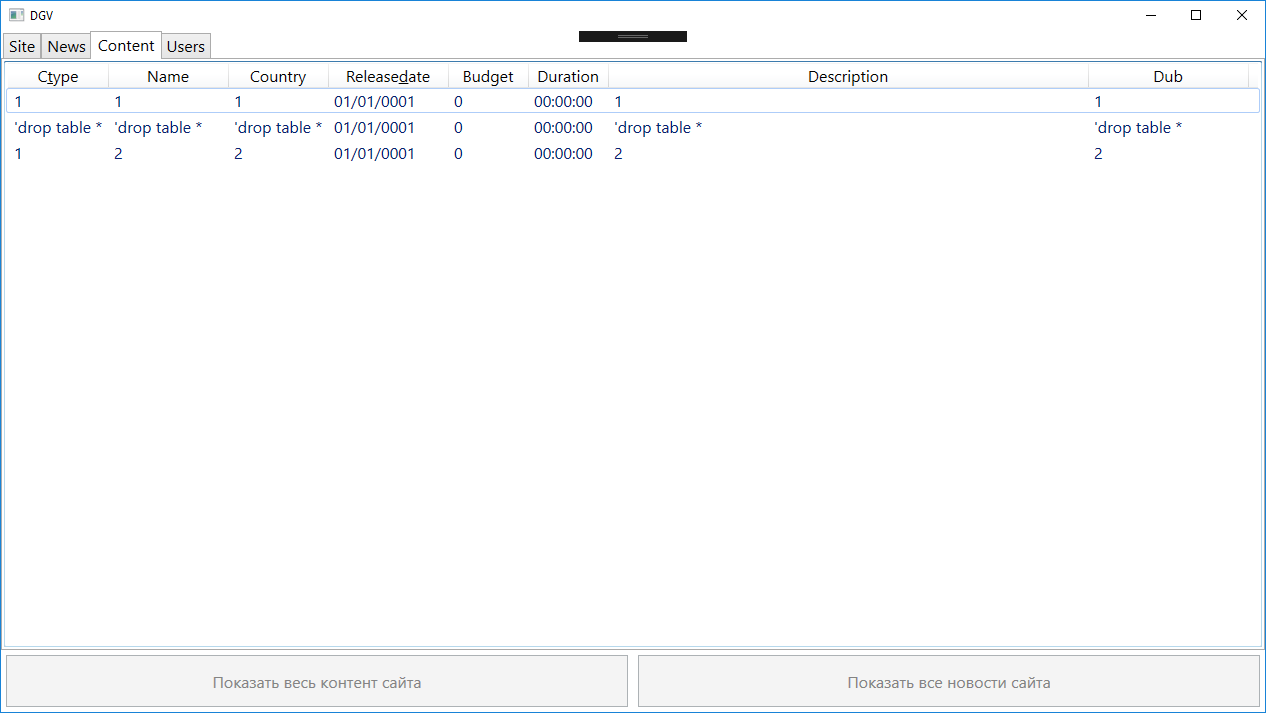


Рисунок 6.4 –представление таблицы с контентом.

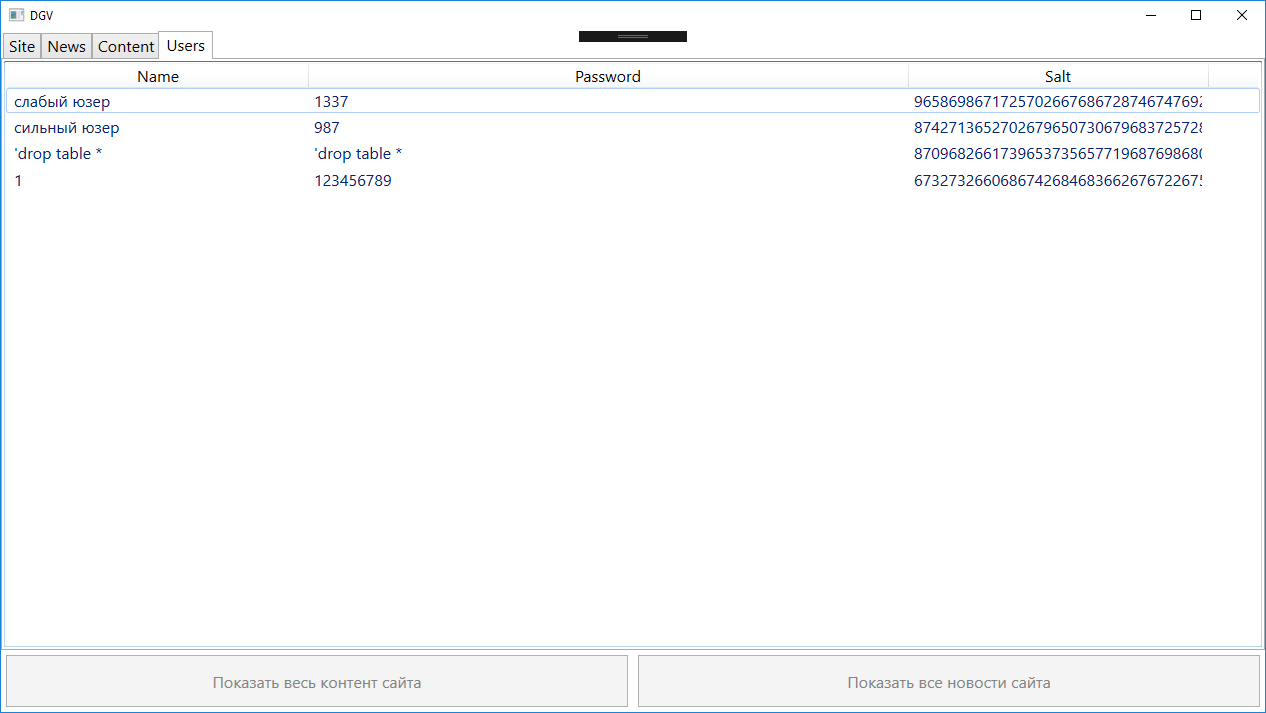


Рисунок 6.5 – представление таблицы пользователей.

Если выбрать какой-либо сайт в списке, то кнопки выполнения запроса станут доступны.

Формы добавления представлены на рисунках 6.6 – 6.10.

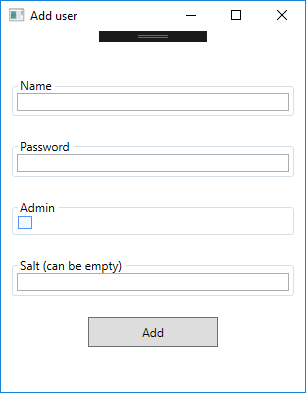


Рисунок 6.6 – форма добавления новых пользователей, в случае изменения данных о пользователе будет вызвана она, но с другими значениями полей.

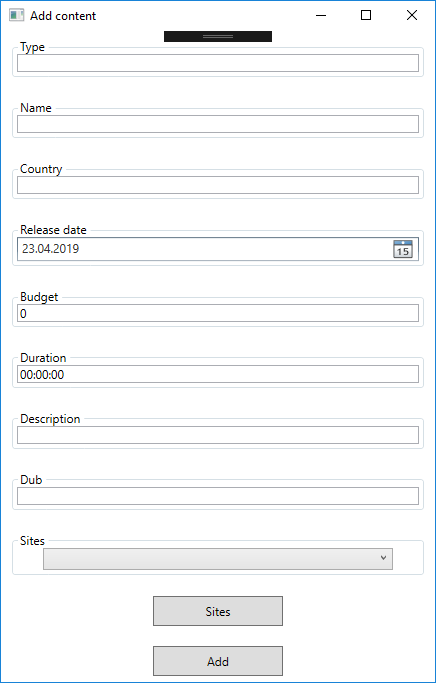


Рисунок 6.7 – форма добавления нового или изменения контента.

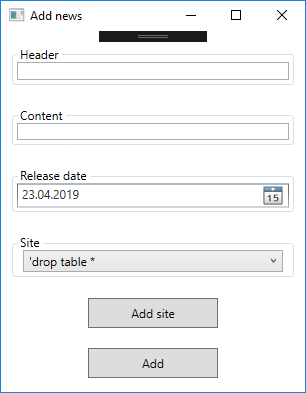


Рисунок 6.8 – форма добавления и изменения новостей.

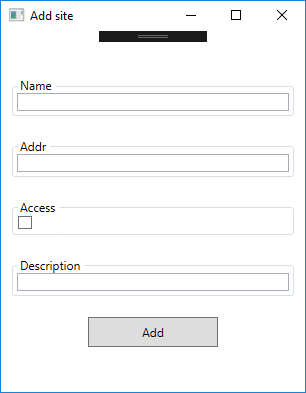


Рисунок 6.9 – форма добавления и изменения сайтов.

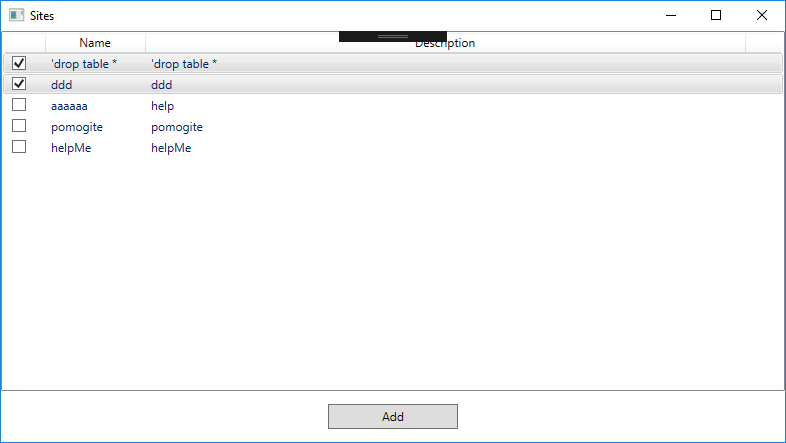


Рисунок 6.10 – форма добавления сайтов размещения для контента.

7 Тестирование

Проверим программу на следующих тестах:

1. Гостевой вход. Должно быть доступно только выполнение запросов и просмотр всех таблиц, кроме таблицы пользователей.
2. Вход с неправильными данными. Должно выводится сообщение об ошибке.
3. Регистрация с неправильными данными, должно выводиться сообщение об ошибке. Пользователи с одинаковым логином не должны создаваться, пробелы не должны влиять.
4. Регистрация с правильными данным. Ожидаем сообщение об успешной регистрации, появление в БД нового пользователя-оператора.
5. Вход с корректными данными оператора. Ожидаем открытие АРМа без таблицы пользователя, но со всем функционалом.
6. Вход с корректными данными админа. Ожидаем открытие АРМа со всеми таблицами и полным функционалом.
7. Удаление пользователей. Ожидаем удаление из списка на форме и в БД. Также запрет на удаление последнего админа.
8. Изменение пользователя. Ожидаем отображение всех изменений в АРМе и БД, а также то, что с последнего админа нельзя снять полномочия.
9. Добавление пользователя. Ожидаем добавление на форме и в БД.
10. Создание сайта. Ожидаем добавление на форме и в БД.
11. Создание новости. Ожидаем добавление на форме и в БД, не зависимо от выбранного для размещения сайта, а также обновление результата запроса. Также не зависимо от того, создан ли был сайт во время создания новости.
12. Создание контента. Ожидаем добавление на форме и в БД, не зависимо от выбранных для размещения сайтов, а также обновление результата запроса. Также не зависимо от того, создан ли был сайт во время создания контента.
13. Удаление сайта. Ожидаем то, что сайт пропадет с формы, из БД, из предлагаемых для размещения.
14. Удаление новости. Ожидаем удаление с формы, из БД, из результата запроса.
15. Удаление контента. Ожидаем удаление с формы, из БД, из результата запроса, удаление всех соответствующих записей из таблицы связки.
16. Изменение сайта. Ожидаем изменение на форме и в БД.
17. Изменение новости. Ожидаем изменение на форме, в БД, а также изменение результата запроса.
18. Изменение контента. Ожидаем изменение на форме, в БД, а также изменение результата запроса.
19. Проверим ввод не корректных данных, ввод строк в поля для даты или для чисел. Ожидаем, что программа не упадет, а также корректное отображение или исправление ошибок. Такое поведение ожидается от всех полей.

Программа прошла тестирование, данные при отображении не противоречили данным, записанным в базе данных.