МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет-приложений)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированные технологии программирования и стандарты проектирования»

Тема Программное средство «Мессенджер»

Исполнитель

студент 2 курса группы 4 Мозолевский Александр Дмитриевич

(Ф.И.О.)

Руководитель работы ассистент Пахолко А.С.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель Пахолко А.С

(подпись)

Минск 2022

**Содержание**

[Введение 3](#_Toc103263942)

[1. Аналитический обзор литературы 4](#_Toc103263943)

[2. Анализ требований к программному средству и разработка функциональных требований 6](#_Toc103263944)

[3. Проектирование программного средства 9](#_Toc103263945)

[4. Создание (реализация) программного средства 12](#_Toc103263946)

Введение

Мессенджер – это программа для смартфона или персонального компьютера, которая позволяет мгновенно обмениваться сообщениями с другими пользователями.

Более ранние версии мессенджеров работали по следующему принципу: всё, что печатал пользователь, отображалось сразу у его собеседника, это напоминало реальный телефонный разговор. Со временем их стали обновлять, сообщения стали появляться на экране собеседника только после их отправки, что способствовало нововведениям в содержании сообщений. Современные мессенджеры имеют большую функциональность, такую как обмен фотографиями, звонки, отправка голосовых сообщений, онлайн-видеочаты, создание бесед с различными пользователями и так далее.

Мессенджеры вытеснили SMS-сообщения и стали превосходной альтернативой голосовым звонкам, так как большинство из них совершенно бесплатные и более удобны в использовании. Единственное условие использования мессенджеров – это подключение к мобильному интернету или Wi-Fi. Однако полностью отказаться от мобильной связи в угоду мессенджерам не получится в связи с тем, что большинство мессенджеров привязаны к мобильному телефону пользователя.

В наше время имеется множество различных мессенджеров, которые отличаются не только интерфейсом, но и функционалом или степенью защиты сообщений. Самым популярным мессенджером на 2022 год является WhatsApp, за ним идут такие приложения, как Facebook Messenger, WeChat, QQ, Telegram, Snapchat и другие. Между различными мессенджерами нет прямой связи, таким образом пользователь WhatsApp не сможет отправить сообщение пользователю Telegram, однако ничто не мешает быть одновременно пользователем нескольких сетей.

Цель моего курсового проекта – разработать собственный мессенджер для моментального обмена сообщениями между пользователями. Каждый пользователь имеет свой собственный профиль, где он может указать информацию о себе и добавить аватар.

При использовании приложения, каждый пользователь сможет обмениваться сообщениями с другими пользователями в режиме реального времени. Язык разработки проекта – C#. При выполнении курсового проекта будут использованы принципы и приёмы ООП, а также использоваться технология Windows Presentation Foundation (WPF).

1. Аналитический обзор литературы

При написании курсового проекта была использована дополнительная литература. Основная информация для реализации курсового проекта была взята с сайтов https://metanit.com/, https://proffesorweb.ru/ и официального сайта <https://docs.microsoft.com/en-us/dotnet/csharp/>. Также использовались некоторые статьи с сайта https://habr.com/ru/.

В качестве аналога для своего курсового проекта была выбрана десктопная версия Telegram, так как она имеет приятный дизайн и функционал, который реализуется в моём приложении.

После входа под своей учетной записью открывается основное окно приложения, которое представлено на рисунке 1.1.

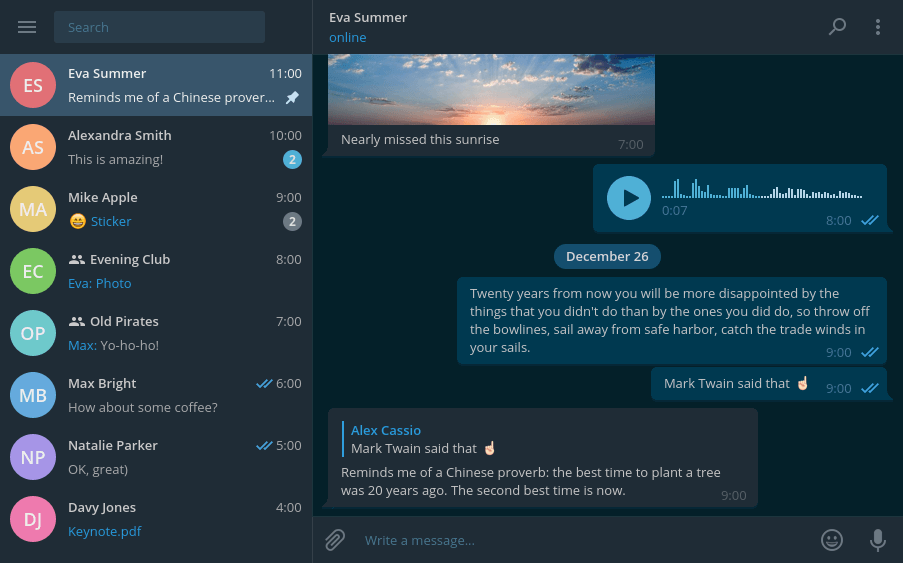


Рисунок 1.1 – Основная страница приложения

На основной странице виден список чатов пользователя и возможность просмотра сообщений при нажатии на определённый чат. Каждый чат из списка отображает аватар пользователя, название чата и последнее сообщение. В окне чата сообщения представлены разным цветом, чтобы было удобно определять свои сообщения и чужие. Также сверху есть блок, при нажатии на который отображается информация чата.

При выборе поиска в пункте меню отображается окно, в котором находятся карточки пользователей и удобная фильтрация по их имени и фамилии. Окно поиска представлено на рисунке 1.2.

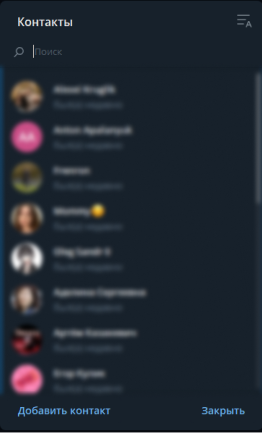


Рисунок 1.2 – Окно поиска пользователей

При выборе нескольких пользователей появляется возможность создать групповой чат, который отображается в основном окне.

При нажатии на пункт «изменить профиль» открывается страница с данными об аккаунте, которая представлена на рисунке 1.3.

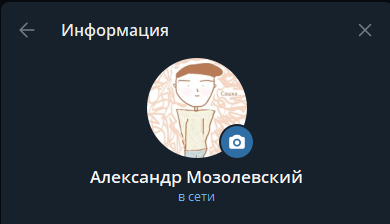


Рисунок 1.3 – Окно изменения профиля

На этом окне есть возможность изменить аватар профиля, нажав на него и выбрав файл с компьютера.

2. Анализ требований к программному средству и разработка функциональных требований

Задачей курсового проекта является разработка приложения, предназначенного для коммуникации пользователей, а именно отправки текстовых сообщений или изображений. Пользователи могут отправлять жалобы на сообщения, затем администратор имеет возможность скрыть их.

Программный продукт реализован на объектно-ориентированном языке C# с использованием технологий WPF и WCF.

Windows Communication Foundation (WCF) – программный фреймворк, используемый для обмена данными между приложениями, входящий в состав .NET Framework.

Windows Presentation Foundation (WPF) — система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML.

В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом возможностей современного графического оборудования. WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (eXtensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление.

Графической технологией, лежащей в основе WPF, является DirectX, в отличие от Window Forms, где используется GDI/GDI+. Производительность WPF выше, чем у GDI+ за счёт использования аппаратного ускорения графики через DirectX.

База данных была разработана на SQL Server в приложении DataGrip.

SQL Server является одной из наиболее популярных систем управления базами данных (СУБД) в мире.

SQL Server характеризуется такими особенностями как:

Производительность. SQL Server работает очень быстро.

Надежность и безопасность. SQL Server предоставляет шифрование данных.

Простота. С данной СУБД относительно легко работать и вести администрирование.

Для организации баз данных MS SQL Server использует реляционную модель, которая предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта.

Для взаимодействия с базой данных применяется язык SQL (Structured Query Language). Клиент (например, внешняя программа) отправляет запрос на языке SQL должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения.

Основной используемый язык запросов – Transact-SQL – реализован на структурированном языке запросов (SQL) с расширениями.

Для осуществления связи между базой данных и приложением на C# необходим посредник, которым будет являться технология Entity Framework. Она основана на платформе .NET Framework и предназначена для работы с данными.

Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Эта технология предоставляет нам набор классов, через которые мы можем отправлять запросы к базам данных, устанавливать подключения, получать ответ от базы данных и производить ряд других операций.

Entity Framework предполагает три возможных способа взаимодействия с базой данных:

* Database first: Entity Framework создает набор классов, которые отражают модель конкретной базы данных;
* Model first: сначала разработчик создает модель базы данных, по которой затем Entity Framework создает реальную базу данных на сервере;
* Code first: разработчик создает класс модели данных, которые будут храниться в базе данных, а затем Entity Framework по этой модели генерирует базу данных и ее таблицы.

В курсовом проектировании для разработки был выбран третий подход.

Приложение необходимо реализовать с использованием шаблона проектирования MVVM и с организацией взаимодействия с базой данных. В качестве клиентской части необходимо реализовать оконное приложение с использованием стандартных и иных библиотек. Приложение должно иметь возможность запускаться без использования интегрированных средств разработки.

Для разработанного продукта необходимо предусмотреть следующие задачи:

* Главное поле для отображения данных;
* Интуитивно понятный интерфейс;
* Механизм поиска пользователей по их имени;
* Возможность просмотра и редактировании информации о себе;
* Возможность просмотра сообщений в чатах;

По итогу работы над продуктом, его необходимо протестировать и отладить, проанализировать его возможности и, по необходимости, внести дополнения или улучшения проекта, с возможностью дальнейшего расширения проекта.

Программное средство предоставляет пользователю следующие функциональные возможности:

* Регистрация пользователей;
* Авторизация пользователей;
* Поиск пользователей;
* Изменение аватара пользователя;
* Создание чатов;

Отправка сообщений;

Отправка изображений.

Администратор чата может удалять или добавлять пользователей чата;

Администратор имеет следующие возможности:

Просмотр активности пользователей;

Получать жалобы от пользователей;

Скрывать сообщения.

Наглядно это видно на UML-схеме, изображённой на рисунке 2.1.

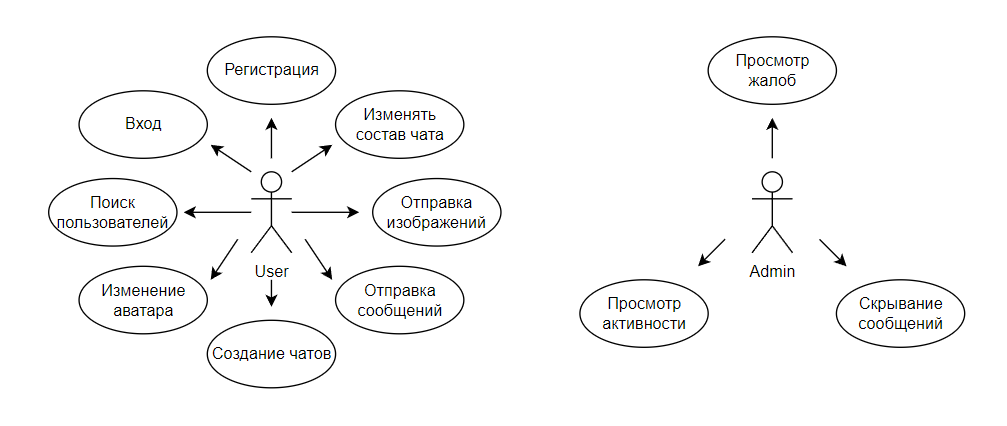


Рисунок 2.1 – «UML-диаграмма возможностей»

3. Проектирование программного средства

Общая схема приложения со стороны клиента представлена на рисунке 3.1.

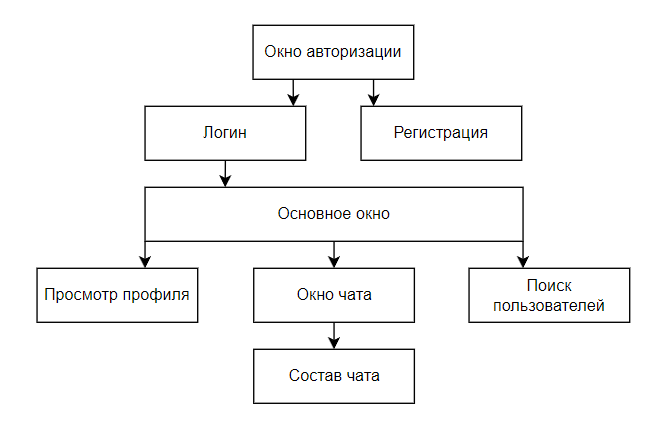


Рисунок 3.1 – Общая схема приложения

При запуске приложения открывается окно авторизации, в котором есть 2 вкладки: логин и регистрация. Чтобы войти на главное окно необходимо выполнить авторизацию. В случае, если у вас нету учётной записи, её можно создать на странице авторизации, где необходимо заполнить данные о себе. После успешной регистрации необходимо ввести данные, указанные при регистрации.

После авторизации открывается окно, на котором изображения информация о текущем пользователе. Справа находится панель навигации, а слева ваши чаты.

На окне просмотра пользователя вы можете сменить ваш аватар, выбрав изображения на компьютере.

Если вы перейдёте на страницу поиска пользователей, можно совершать их поиск по имени и фамилии. При нажатии на пользователя, его имя и фамилия будут отображены справа, разрешено выбирать неограниченное количество пользователей. Затем можно выбрать изображение и имя чата. После нажатия на кнопку создания чата его данные перенесутся на сервер, где далее занесутся в базе данных. У каждого подключённого пользователя сработает callback-функция, которая добавит чат в панели слева.

При нажатии на любой чат откроется соответствующее окно, где будут отображены название чата, сообщения и панель написания сообщений. На панели есть две кнопки: для отправки изображения и текстового сообщения. При отправке изображения, открывается file dialog, где вы можете выбрать нужную картинку. При отправке текстового сообщения вам необходимо ввести сам текст и нажать на кнопку отправки или клавишу enter на клавиатуре. При отправке сообщения данные переносятся на сервер и заносятся в базу данных. Каждому пользователю, который на данный момент находится в приложении, отправится запрос на вызов callback-функции, которая выведет полученное сообщение на экран.

При нажатии на название чата вам откроется информация об его участниках. Если вы создатель чата, то вы можете добавить в него новых участников, а также удалять пользователей, которые в нём состоят. Если пользователь, которого вы удаляете, на данный момент находится в окне чата, будет перенесён на страницу информации об аккаунте, а чат удалится из его списка.

В списке сообщений чата вы можете отправить жалобу на сообщения, которые содержат нарушения. При жалобе она отправляется на сервер, заносится в БД и вызывается callback-функция у администратора, которая добавляет жалобу в его окне.

Если вы зашли под аккаунтом администратора, вам доступная схема приложения, которая представлена на рисунке 3.2.

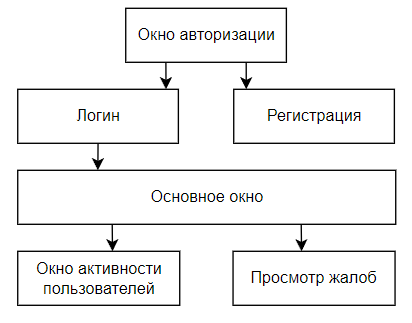


Рисунок 3.2 – Общая схема для администратора

На основном окне сразу отображается страница с активностью пользователей, где вы можете просматривать входы и выходы из приложения от обычных пользователей.

Справа находится навигация, если вы нажмёте на страницу жалоб, вам откроется список, каждый элемент которого отображает жалобу. Для каждого элемента есть две кнопки: удаления и принятия жалобы. При нажатии на удаление, данные жалобы удалятся из базы данных и на окне админа. При принятии жалобы, жалоба также удалится из базы данных, но сообщение изменит своё содержимое: картинка удалится, а текст сообщения изменится на «this message was hidden by admin».

Структура проекта части описана в таблицах 3.3 – 3.5.

Таблица 3.3 – Описание структурных пакетов проекта клиентской части

|  |  |
| --- | --- |
| Имя пакета | Описание |
| Model | Здесь описаны модели, которые используются во ViewModel: Admin, Chat, Client, Login, Message, Registration, User. |
| ViewModel | Содержит логику, которая позволяет получить данные при помощи View, обработать их, используя при этом Model, после чего передать на сервер. |
| View | Содержит все представления, которые позволяют пользователю работать с приложением. Они описывают графическую составляющую приложения. |
| Themes | Файлы тем приложения. |
| Languages | Файлы языка приложения. |
| Resources | Файлы ресурсов приложения. |
| App.config | Файл конфигурации приложения. |
| App.xaml | Основная задача данного файла состоит в определении ресурсов, общих для приложения. |

Таблица 3.4 – Описание структурных пакетов проекта серверной части

|  |  |
| --- | --- |
| Имя пакета | Описание |
| DTO | Описаны модели, на основе которых происходит генерация и работа с базой данных. |
| DataBase | Содержит класс контекста подключения к базе данных. |
| Repository | Содержит интерфейс и класс для работы с данными базы данных. |
| Unit | Содержит класс, который включает все репозитории и класс контекста. |
| Service | Содержит интерфейс и класс, в котором описаны запросы к базе данных. |
| app.config | Файл конфигурации. |

Таблица 3.5 – Описание структурных пакетов проекта хоста

|  |  |
| --- | --- |
| Имя пакета | Описание |
| Program | Содержит файл запуска сервера. |
| App.config | Файл конфигурации приложения. |

4. Создание (реализация) программного средства

При создании приложения использовался паттерн проектирования MVVM. Он заключается в разделении представления от бизнес-логики. Это достигается за счёт ввода новой логической конструкции ViewModel. Он связывает представление и бизнес-логику приложения.

При проектировании базы данных курсового проекта использовался Entity Framework. Он представляет специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работаем с объектами.

Отличительной чертой Entity Framework является использование запросов LINQ для выборки данных из БД. С помощью LINQ мы можем не только извлекать определенные строки, хранящие объекты, из бд, но и получать объекты, связанные различными ассоциативными связями.

Нередко все сущности (модель) наследует интерфейс INotifyPropertyChanged или INotifyCollectionChanged, которые позволяют отлавливать изменения и правильно на их реагировать и сохранять.

Для того, чтобы реализовать интерфейс INotifyPropertyChanged, нужно объявить свойство PropertyChanged и метод OnPropertyChanged().

Для того, чтобы начать отслеживание свойства, необходимо вызывать метод OnPropertyChanged(“Имя свойства”).

Для удобной работы с данными, используется паттерн Repository. Репозиторий позволяет абстрагироваться от конкретных подключений к источникам данных, с которыми работает программа, и является промежуточным звеном между классами, непосредственно взаимодействующими с данными, и остальной программой.

Чтобы не создавать отдельный класс репозитория для каждой модели базы данных, можно его обобщить, затем в классе UnitOfWork указать все обобщенные репозитории. На рисунке 4.1 показана структура для работы с базой данных.

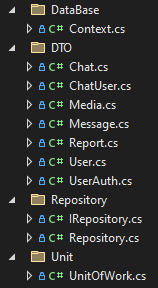


Рисунок 4.1 – структура проекта для работы с базой данных

В приложении есть пакет View, который содержит все представления. Он делится еще на 3 пакета: для авторизации, пользователя и админа. В них хранятся файлы представлений на языке разметки xaml. Структура пакета View представлена на рисунке 4.2.

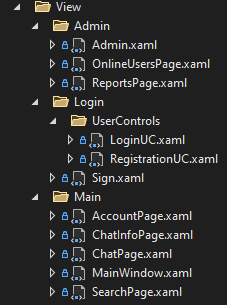


Рисунок 4.2 – структура проекта для View

Пакет ViewModel содержит классы, которые настраивают логику отображения моделей на слое представления. Классы ViewModel также разделены на три отдельных пакета: авторизации, пользователя и админа. Во ViewModel обычно содержится модель, к которой создаются привязки. Структура пакетов для Model и ViewModel представлена на рисунках 4.3 и 4.4.

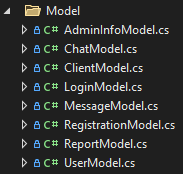


Рисунок 4.3 – структура проекта для Model

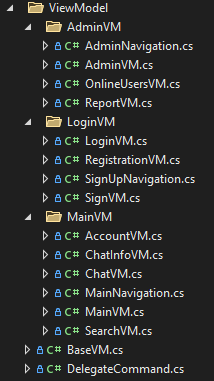


Рисунок 4.4 – структура проекта для ViewModel

На рисунке 4.5 изображена диаграмма таблиц со связями в базе данных.

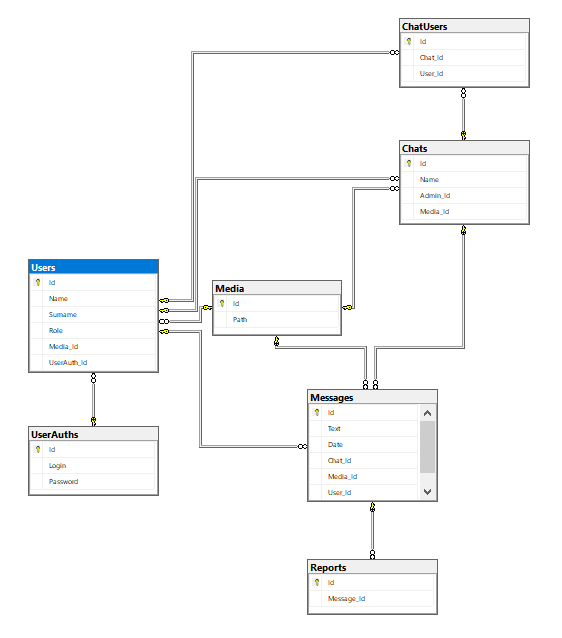
  
Рисунок 4.5 – Диаграмма таблиц

Таблица 4.6 – описание таблицы UserAuth

|  |  |
| --- | --- |
| Id | Идентификатор |
| Login | Логин |
| Password | Пароль |

Таблица 4.7 – описание таблицы User

|  |  |
| --- | --- |
| Id | Идентификатор |
| Name | Имя |
| Surname | Фамилия |
| Role | Роль |
| Media\_Id | Идентификатор аватара |
| UserAuth\_Id | Идентификатор данных для авторизации |

Таблица 4.8 – описание таблицы Reports

|  |  |
| --- | --- |
| Id | Идентификатор |
| Message\_Id | Идентификатор сообщения |

Таблица 4.9 – описание таблицы Messages

|  |  |
| --- | --- |
| Id | Идентификатор |
| Text | Текст сообщения |
| Date | Дата отправки |
| Chat\_Id | Идентификатор чата |
| Media\_Id | Идентификатор изображения |
| User\_id | Идентификатор отправителя |

Таблица 4.10 – описание таблицы Media

|  |  |
| --- | --- |
| Id | Идентификатор |
| Path | Путь к изображению |

Таблица 4.11 – описание таблицы Chats

|  |  |
| --- | --- |
| Id | Идентификатор |
| Name | Название |
| Admin\_Id | Идентификатор создателя |
| Media\_Id | Идентификатор аватара |

Таблица 4.12 – описание таблицы ChatUsers

|  |  |
| --- | --- |
| Id | Идентификатор |
| Chat\_Id | Идентификатор чата |
| User\_Id | Идентификатор пользователя |