



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии» (ИУ7)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

**Микросервис универсального
мессенджера**

Студент ИУ7-63Б
(Группа)

Сиденко А. Г.
(Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта

Бекасов Д. Е.
(Подпись, дата) (И.О.Фамилия)

Консультант

Бабарыкин Д. С.
(Подпись, дата) (И.О.Фамилия)

2020 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ИУ7
(Индекс)
И.В.Рудаков
(И.О.Фамилия)
« ____ » _____ 20__ г.

ЗАДАНИЕ на выполнение курсового проекта

по дисциплине Базы данных

Студент группы ИУ7-63Б

Сиденко Анастасия Генадьевна
(Фамилия, имя, отчество)

Тема курсового проекта Микросервис универсального мессенджера

Направленность КП (учебный, исследовательский, практический, производственный, др.)
учебный

Источник тематики (кафедра, предприятие, НИР) кафедра

График выполнения проекта: 25% к 4 нед., 50% к 7 нед., 75% к 11 нед., 100% к 14 нед.

Задание Разработать микросервис универсального мессенджера. ПО может быть использовано как микросервис для создания полноценного корпоративного мессенджера или любого другого приложения, где необходим функционал мгновенного обмена сообщениями. Также необходимо разработать тестовое клиентское приложение.

Оформление курсового проекта:

Расчетно-пояснительная записка на 40-45 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

К защите должны быть подготовлены презентация и доклад, отражающие суть выполненной работы, содержание и методы решения основных задач, а также полученные результаты.

Дата выдачи задания « 01 » марта 2020 г.

Руководитель курсового проекта

Д.Е. Бекасов
(Подпись, дата) (И.О.Фамилия)

Студент

А. Г. Сиденко
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

РЕФЕРАТ

Курсовой проект представляет собой микросервис универсального мессенджера.

Ключевые слова: микросервис, мессенджер, универсальный, PostgreSQL, ASP.NET Core, Entity Framework Core, REST.

Данный микросервис реализуется на языке программирования C# с использованием платформы ASP.NET Core и технологии Entity Framework Core. Для взаимодействия между клиентом и сервером используется архитектурный стиль REST.

Полученное в результате работы ПО может быть использовано как микросервис для создания полноценного корпоративного мессенджера или любого другого приложения, где необходим функционал мгновенного обмена сообщениями.

Отчёт содержит 45 страниц, 23 рисунка, 3 таблицы, 14 источников.

СОДЕРЖАНИЕ

Введение.....	6
1 Аналитический раздел.....	8
1.1 Постановка задачи.....	8
1.2 Обзор и анализ существующих решений, обоснование необходимости разработки.....	8
1.2.1 Slack.....	9
1.2.2 Microsoft Teams.....	10
1.2.3 Twist.....	10
1.2.4 Discord.....	11
1.2.5 Rocket.Chat.....	12
1.3 Анализ типов и выбор СУБД.....	14
1.3.1 Типы СУБД.....	15
1.3.1.1 Реляционная модель.....	15
1.3.1.2 Иные подходы.....	15
1.3.2 Выбор реляционной СУБД.....	16
1.3.2.1 MSSQL.....	16
1.3.2.2 MySQL.....	17
1.3.2.3 PostgreSQL.....	18
1.3.2.4 Oracle.....	18
1.4 Вывод.....	19
2 Конструкторский раздел.....	20
2.1 Функциональная модель.....	20
2.2 Сценарий использования.....	21
2.3 База данных.....	24
2.4 Проектирование архитектуры.....	26

2.5 Вывод.....	29
3 Технологический раздел.....	30
3.1 Выбор технологий.....	30
3.2 Модель базы данных.....	31
3.3 UML-диаграммы классов	32
3.4 Тестовое клиентское приложение	38
3.5 Администрирование БД.....	41
3.6 Вывод.....	42
Заключение.....	43
Список использованных источников	44

ВВЕДЕНИЕ

На сегодняшний день более 4,5 миллиарда людей пользуются интернетом, а аудитория социальных сетей перевалила за отметку в 3,8 миллиарда [1].

Получается, что общение посредством мессенджеров является неотъемлемой частью жизни современного человека.

Практически все компании сталкиваются с вопросом: «Какой использовать корпоративный мессенджер?». Он должен объединять все внутренние и внешние коммуникации в одно пространство. Сегодня рынок корпоративных мессенджеров предлагает десятки вариантов с разным функционалом, интеграцией с другими сервисами и ценовой политикой. Такие как, Slack, Microsoft Teams, Twist, Discord и другие [2].

Однако данные приложения имеют ряд недостатков:

- сотрудники могут пользоваться различными мессенджерами;
- в контактах могут присутствовать люди, не имеющие отношения к работе, что служит отвлекающим фактором;
- недостаточная функциональность;
- безопасность.

Для удобства коммуникации и работы внутри компании разрабатываются свои корпоративные внутренние средства общения, которые компенсируют вышеописанные недостатки.

Таким образом можно сделать вывод, что создание универсального мессенджера является актуальной темой, так как данное приложение будет ориентировано на специфику конкретной компании.

Целью данной работы является реализация микросервиса универсального мессенджера. Функциональное назначение микросервиса — хранение и работа с данными.

Для достижения поставленной цели необходимо решить следующие задачи.

1. Анализ существующих решений.
2. Проектирование микросервиса мессенджера.
3. Реализация микросервиса.
4. Разработка тестового клиентского приложения.

1 Аналитический раздел

Целью работы является создание микросервиса для универсального корпоративного мессенджера. В данном разделе рассматриваются существующие решения, производится выбор СУБД.

1.1 Постановка задачи

Требуется разработать микросервис универсального корпоративного мессенджера, который обладает следующей функциональностью:

- создание, удаление, изменение параметров пользователей, хранение истории изменений;
- создание, удаление, изменение параметров чата, хранение истории изменений;
- добавление, удаление, изменение параметров участников чата, хранение истории изменений;
- отправка сообщений, файлов, хранение истории изменений;
- возможность добавления сообщения в избранное, закрепление сообщений в чатах, хранение истории изменений.

Система должна иметь интерфейс для демонстрации работы микросервиса. Микросервис осуществляет работу с базой данных, обрабатывает и обслуживает запросы пользователей.

1.2 Обзор и анализ существующих решений, обоснование необходимости разработки

В настоящее время существует большое число различных корпоративных мессенджеров, которые решают вопрос коммуникации сотрудников внутри компании, а также поддерживает связь с клиентами, подрядчиками и партнерами [3].

1.2.1 Slack

Slack [4] – первый корпоративный мессенджер, предложил решение, позволяющее избавиться от многочисленных писем в почте и структурировать коммуникацию внутри команды. Далее разбивается на отдельные каналы по проектам или отделам. Имеет огромный функционал.

В бесплатной версии Slack поддерживает неограниченное число пользователей, интеграцию с 10 внешними сервисами и поиск в архиве до 10 тысяч сообщений. В платных тарифах – 6,67\$ (Standard) и 12,5\$ (Plus) за пользователя в месяц – снимаются ограничения и появляются дополнительные возможности для управления правами доступа.

На рисунке 1.1 показан интерфейс мессенджера Slack.

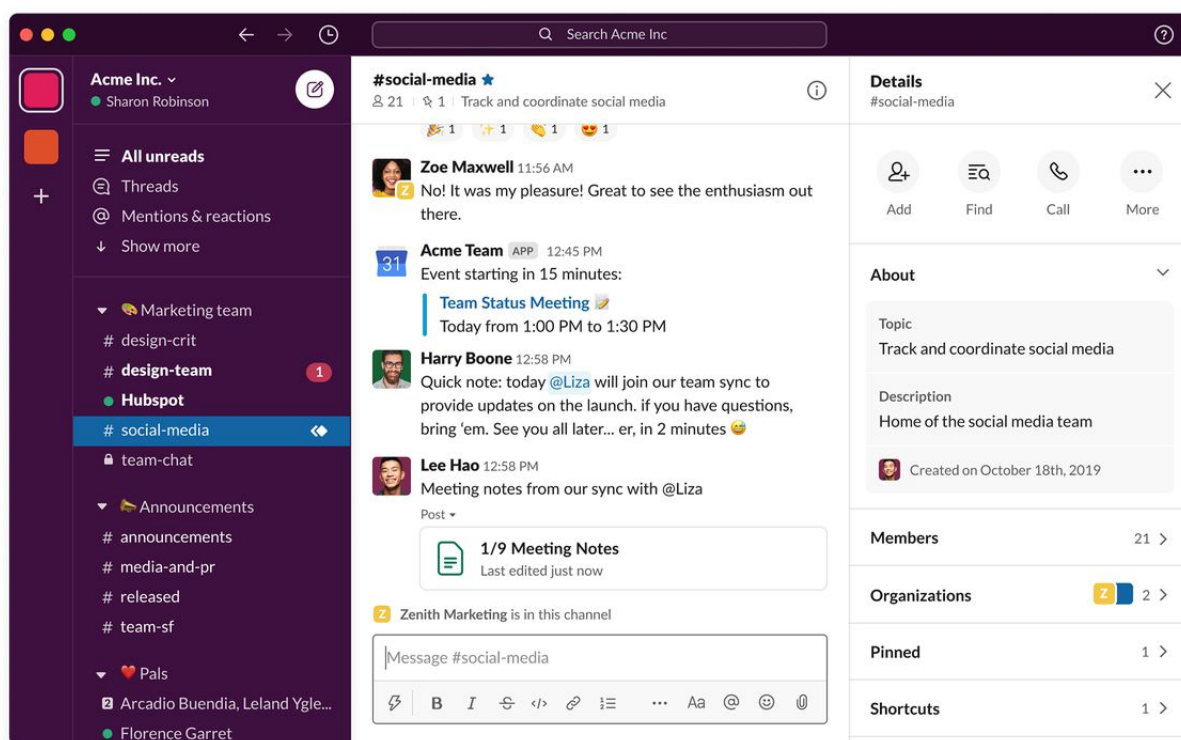


Рисунок 1.1 — Интерфейс Slack.

1.2.2 Microsoft Teams

Microsoft Teams [5] является альтернативой Slack. Возможность создания команд, общения посредством чата. Удобная интеграция со всеми сервисами Office 365. Однако, по сравнению со Slack, всего около 200 внешних сервисов (в Slack порядка тысячи).

Ценовая политика лояльнее. Так, в бесплатный план входит неограниченная история сообщений, конференции до 250 человек, совместное использование экрана. Самый доступный платный план для Microsoft Teams – \$ 5 с пользователя в месяц.

На рисунке 1.2 показан интерфейс мессенджера Microsoft Teams.

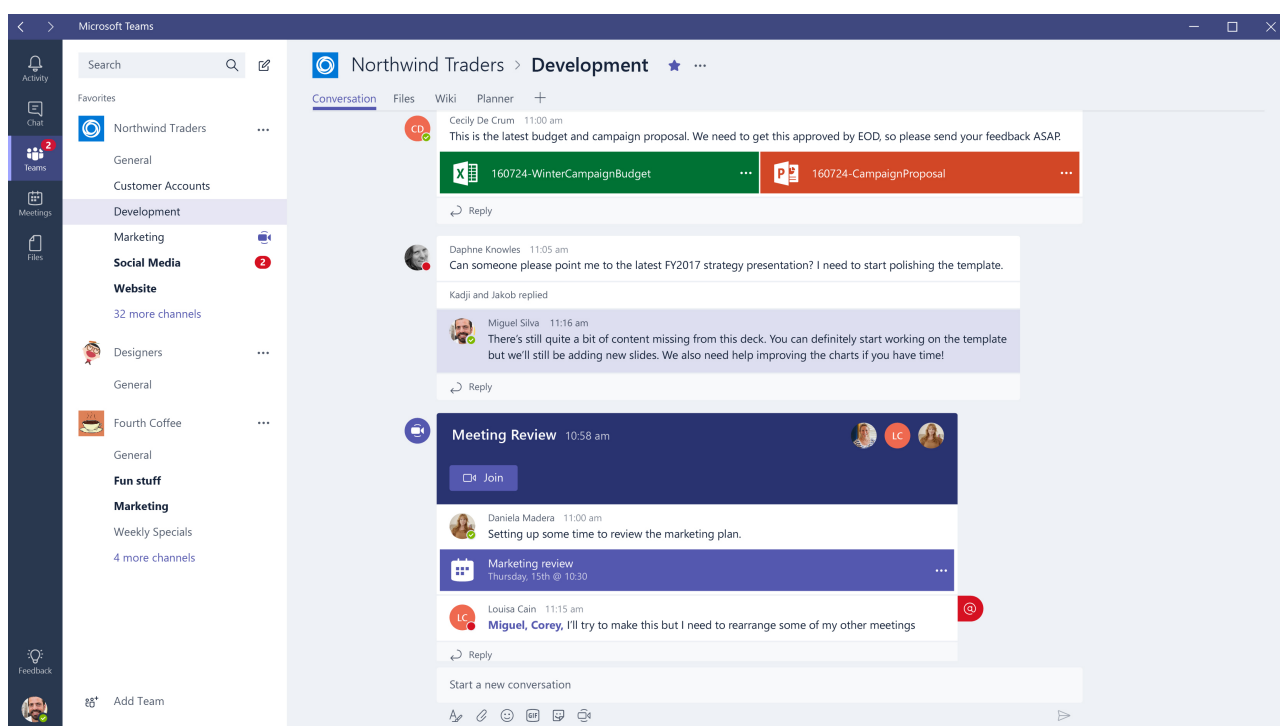


Рисунок 1.2 — Интерфейс Microsoft Teams.

1.2.3 Twist

Twist [6] – это успешная альтернатива Slack. Основное отличие в том, чтобы больше времени работать, и меньше отвлекаться на постоянные уведомления и чаты. Работа в мессенджере построена по принципу обсуждений – для каждой задачи создаётся отдельный чат, в который можно пригласить членов команды. Обсуждения можно разделить по ка-

налам – в соответствии с общей темой. Таким образом, нет необходимости долго искать нужное обсуждение.

Как и Slack, использование мессенджера бесплатно для команд, которым не нужен доступ к полной истории сообщений. Для остальных стоимость составит 329 Р в месяц за человека.

На рисунке 1.3 показан интерфейс мессенджера Twist.

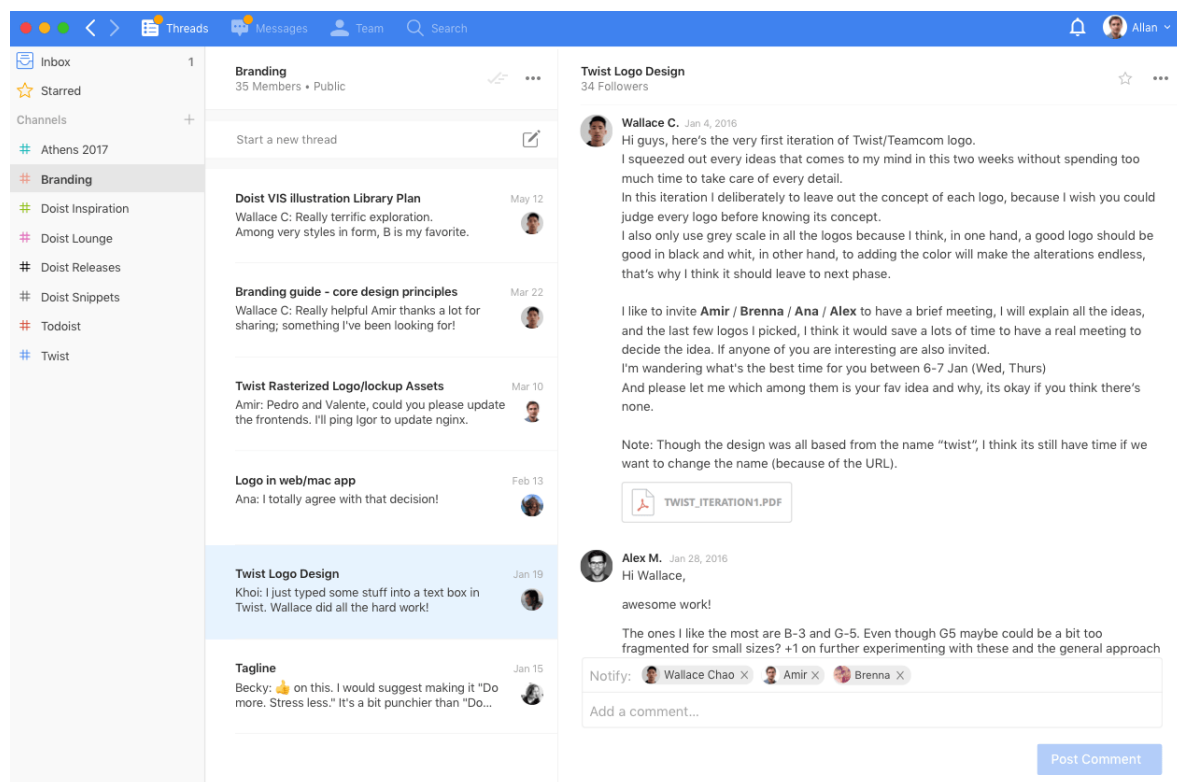


Рисунок 1.3 — Интерфейс Twist.

1.2.4 Discord

Discord [7] – игровой мессенджер. У пользователя в нём есть свой аккаунт, через который он может общаться с друзьями из списка контактов, а может присоединиться к неограниченному количеству команд. Команды создаются бесплатно и в них также может быть неограниченное количество участников. Внутри всё устроено наподобие Slack: есть каналы, есть приватные каналы, есть возможность общаться напрямую.

На рисунке 1.4 показан интерфейс мессенджера Discord.

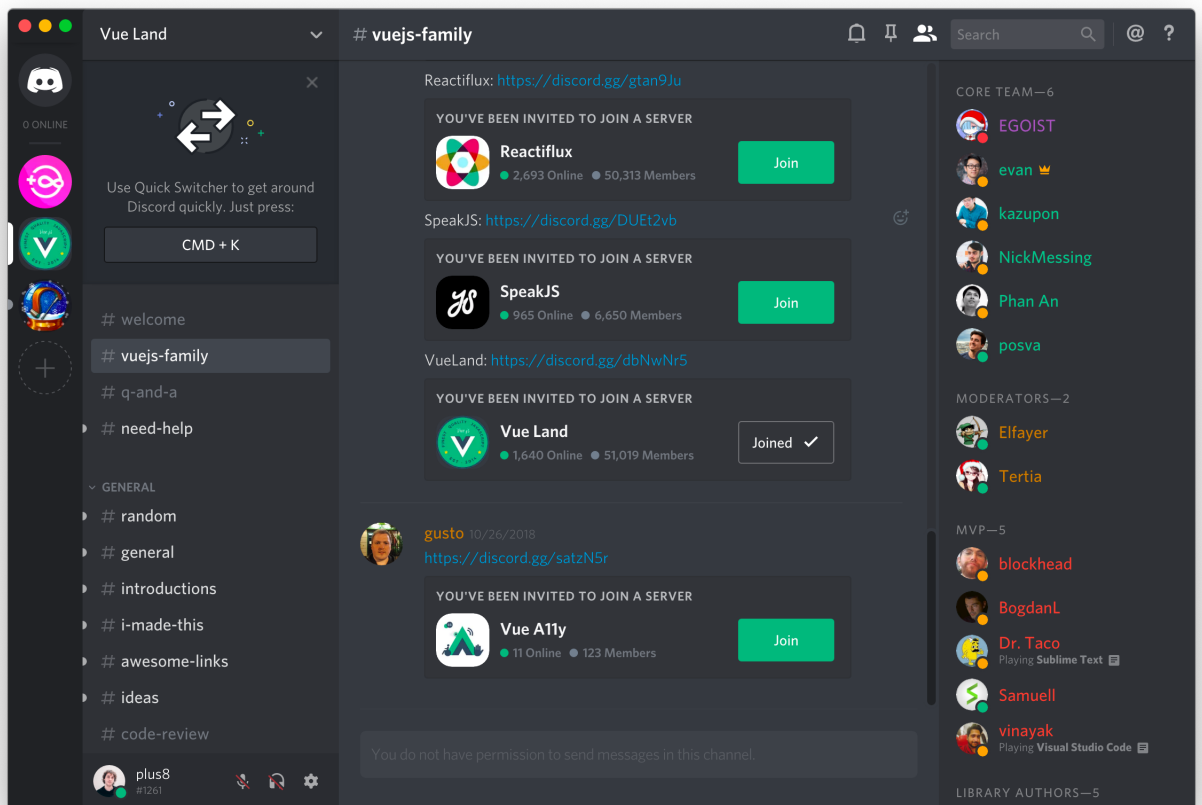


Рисунок 1.4 — Интерфейс Discord.

1.2.5 Rocket.Chat

Rocket.Chat [8] — это мессенджер с открытым исходным кодом, который поддерживает групповые чаты, обмен файлами, видеоконференции, ботов и многое другое. Rocket.Chat можно также установить на собственный сервер, а затем общаться, используя веб-интерфейс, персональный компьютер или мобильное устройство. Или использовать платную версию для облачного хранения.

Можно добавлять новую функциональность.

На рисунке 1.5 показан интерфейс мессенджера Rocket.Chat.

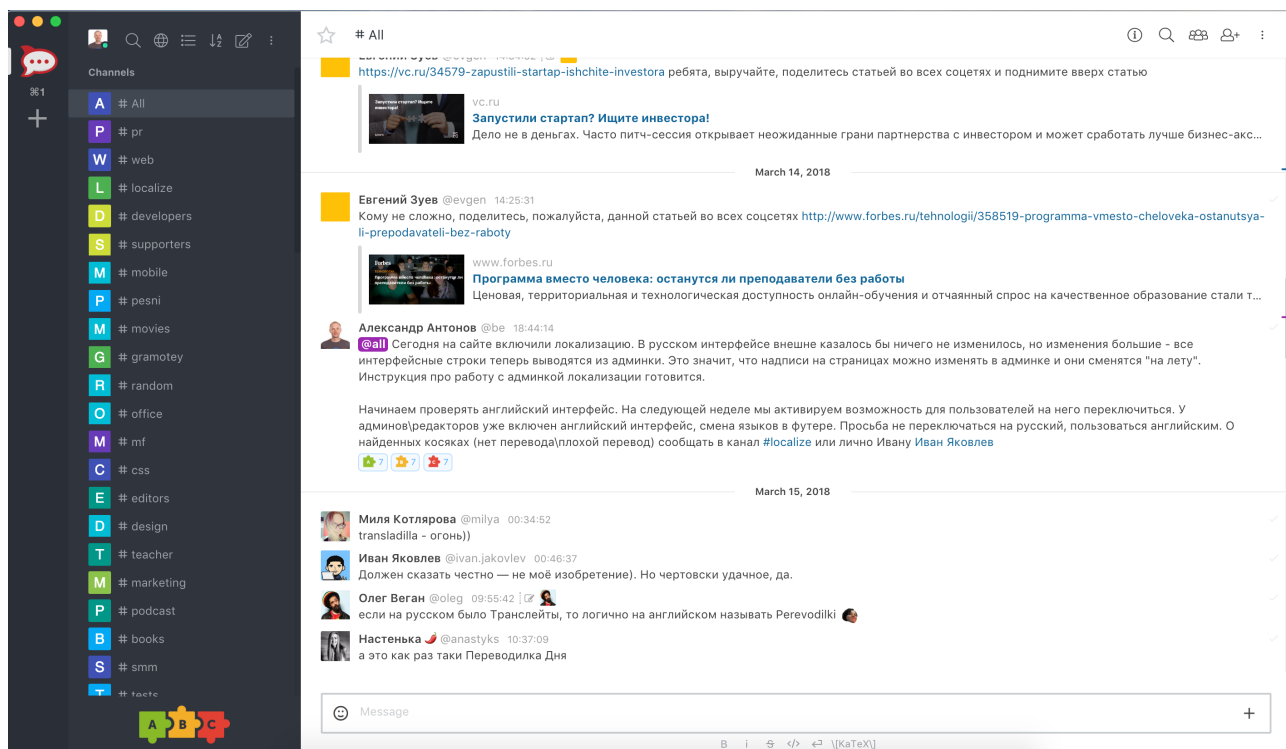


Рисунок 1.5 — Интерфейс Rocket.Chat.

На основе аналогов и поставленных цели и задач можно выделить следующие критерии сравнения:

- создание обсуждения задач;
- несколько закрепленных сообщения;
- избранные сообщения для отдельных пользователей;
- хранение всей истории изменений (чаты, сообщения, пользователи);
- разные имена пользователей в различных чатах;
- наличие различных прав у пользователей;
- возможность хранения на собственном сервере.

В таблице 1.1 представлено сравнение мессенджеров по вышепредставленным параметрам.

Таблица 1.1 — Сравнение мессенджеров.

	Slack	Microsoft Teams	Twist	Discord	Rocket.Chat
Создание обсуждения задач	+ (прям в чате)	+	+	-	-
Несколько закрепленных сообщения	+	-	-	+ (ограничение 50)	+
Избранные сообщения для отдельных пользователей	+	-	-	-	+
Хранение всей истории изменений (чаты, сообщения, пользователи)	-	-	-	-	-
Разные имена пользователей в различных чатах	-	+	-	+	+
Наличие различных прав у пользователей	-	+	+	+	+
Возможность хранения на собственном сервере	-	-	-	-	+

Актуальность темы обусловлена отсутствием универсального мессенджера с полной историей изменения чатов, пользователей, сообщений. А ещё у рассмотренных готовых решений вся переписка хранится на облачных серверах мессенджера. Данный факт не устраивает большое количество компаний, которым желательно хранить все данные на своих серверах.

1.3 Анализ типов и выбор СУБД

Цели и задачи поставлены необходимо определиться с СУБД.

1.3.1 Типы СУБД

Системы управления базами данных (СУБД) – это высокоуровневое программное обеспечение, работающее с низкоуровневыми API. Для решения проблем создавались различные виды СУБД. Два основных направления – реляционные (SQL) и нереляционные (NoSQL) СУБД [9]. СУБД основаны на моделях баз данных.

1.3.1.1 Реляционная модель

Представленная в 70-х, реляционная модель предлагает структурированное хранение данных. Отношения дают возможность группировки данных как связанных наборов, представленных в виде таблиц, содержащих упорядоченную информацию и соотносящих значения и атрибуты. РСУБД работают производительно и надёжно. Несмотря на строгие принципы формирования и обработки данных, РСУБД могут быть гибкими.

1.3.1.2 Иные подходы

NoSQL-способ структуризации данных заключается в избавлении от ограничений при хранении и использовании информации. Базы данных NoSQL, используя неструктуризированный подход, предлагают много эффективных способов обработки данных в отдельных случаях.

В таблице 1.2 представлено сравнение 2 этих подходов.

Таблица 1.2 — Сравнение моделей БД.

	SQL	NoSQL
Структура и тип хранящихся данных	Однозначно определённая структура хранения данных	Ограничений нет
Запросы	Получение данных при помощи языка SQL	Каждая NoSQL база данных реализует свой способ работы с данными
Масштабируемость	Не подходит для постоянно меняющихся структур данных	Подходит для постоянно меняющихся структур данных
Поддержка	Развитая СУБД, наличие большого сообщества вокруг неё, множество примеров	Только недавно стала популярной
Объединение таблиц	Применени join	Отсутствие join

Так как, данные имеют четкую структуру и заранее определены, выбор в пользу реляционной базы данных.

1.3.2 Выбор реляционной СУБД

Одними из самых популярных РСУБД сейчас являются MSSQL, MySQL, PostgreSQL, Oracle.

1.3.2.1 MSSQL

Microsoft SQL Server – разработанная корпорацией майкрософт, система управления реляционными базами данных.

Преимущества

- простота: MSSQL легко устанавливается и используется;

- возможность регулировки и отслеживания уровня производительности, чтобы уменьшить загрузку;
- возможность интеграции с другими продуктами Microsoft;
- визуализация на мобильных устройствах.

Недостатки

- высокая стоимость продукта для юридических лиц;
- возможны проблемы в работе служб интеграции импорта файлов;
- высокая ресурсоемкость SQL Server.

1.3.2.2 MySQL

MySQL – это самая популярная из всех крупных серверных БД. Разобраться в ней просто, большое количество информации.

Преимущества

- простота: MySQL легко устанавливается и используется;
- много функций: MySQL поддерживает большую часть функционала SQL;
- безопасность: в MySQL встроено много функций безопасности;
- мощность и масштабируемость: MySQL может работать с действительно большими объёмами данных, и неплохо подходит для масштабируемых приложений;
- скорость: пренебрежение некоторыми стандартами позволяет MySQL работать производительнее.

Недостатки

- ограничения функциональности, так как не полностью реализованы SQL-стандарты;
- надёжность: некоторые операции реализованы менее надёжно, чем в других РСУБД;

— застой в разработке: хотя MySQL и является open-source продуктом, работа над ней сильно заторможена.

1.3.2.3 PostgreSQL

PostgreSQL – это РСУБД, ориентирующаяся в первую очередь на полное соответствие стандартам и расширяемость.

Преимущества

- полная SQL-совместимость;
- сообщество: PostgreSQL поддерживается опытным сообществом 24/7;
- поддержка сторонними организациями;
- расширяемость: PostgreSQL можно программно расширить за счёт хранимых процедур.

Недостатки

- производительность: в простых операциях чтения PostgreSQL может уступать своим аналогам.

1.3.2.4 Oracle

Oracle – это объектно-реляционная система управления базами данных.

Преимущества

- поддержка огромных баз данных и большого числа пользователей;
- быстрая обработка транзакций;
- большой и постоянно развивающийся функционал.

Недостатки

- высокая стоимость;
- требуются значительные вычислительные ресурсы.

PostgreSQL выигрывает по многим параметрам и бесплатно распространяется. Поэтому было принято решение использовать именно эту РСУБД.

1.4 Вывод

В данном разделе были рассмотрены существующие решения, была выбрана СУБД и поставлена задача реализации микросервиса мессенджера с хранением истории, возможностью добавлять сообщения в избранное и закреплять в чатах.

2 Конструкторский раздел

Можно разбить задачу проектирования микросервиса на подзадачи:

- описание функциональности и поведения;
- выделение сущностей предметной области;
- разработка структуры БД;
- проектирование компонентов микросервиса.

В данном разделе рассматривается структура микросервиса мессенджера, каждого из его компонентов.

2.1 Функциональная модель

На рисунке 2.1 изображена функциональная модель, отображающая структуру и функции системы.

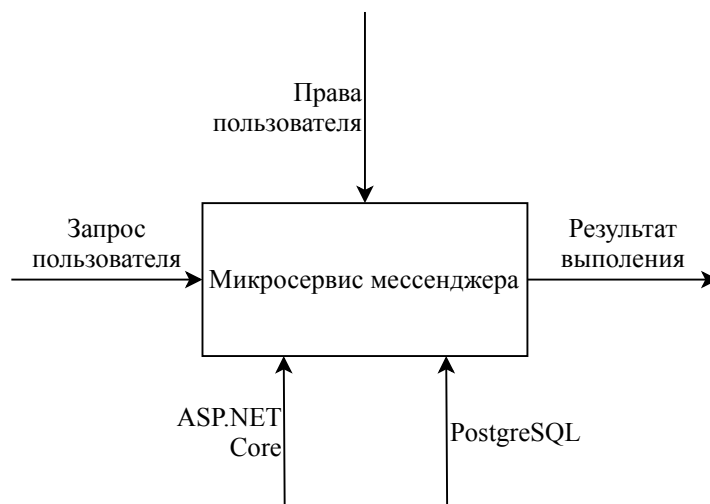


Рисунок 2.1 — Функциональная модель микросервиса мессенджера.

2.2 Сценарий использования

В данном разделе необходимо построить Use Case Diagram (диаграмму прецедентов). Она состоит из графической диаграммы, описывающей действующие лица и прецеденты – конкретные действия, которые выполняет пользователь при работе с системой.

Данная диаграмма предназначена для определения функциональных требований, действующие лица не будут делиться по ролям и правам, так как за это отвечает внешняя система.

На рисунке 2.2 представлена Use Case диаграмма для действий с чатами.

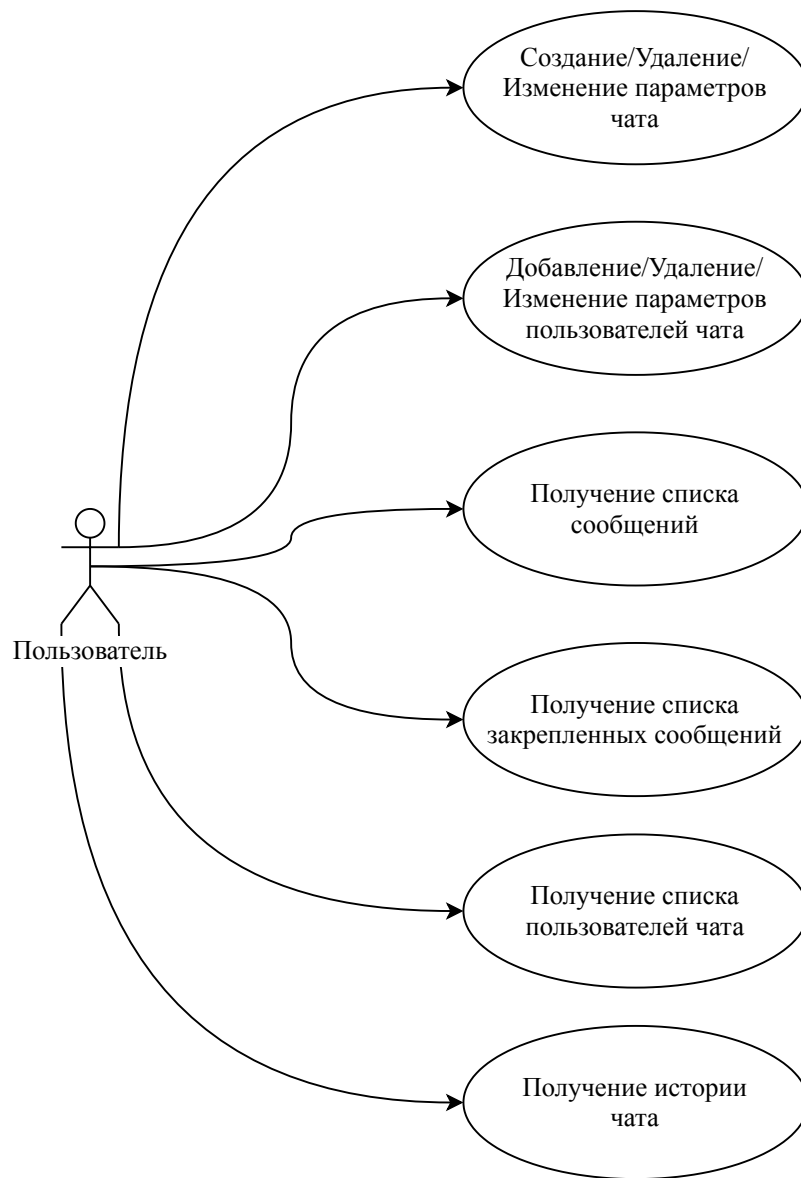


Рисунок 2.2 — Use Case диаграмма для действий с чатами.

На рисунке 2.3 представлена Use Case диаграмма для действий с пользователями.

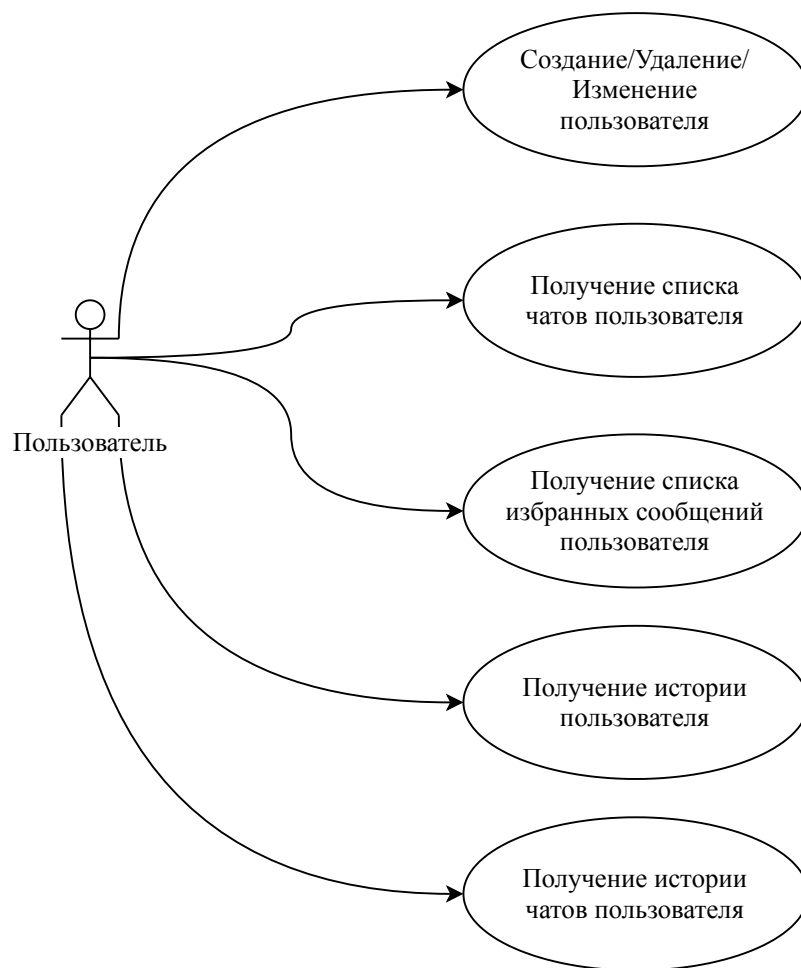


Рисунок 2.3 — Use Case диаграмма для действий с пользователями.

На рисунке 2.4 представлена Use Case диаграмма для действий с сообщениями.

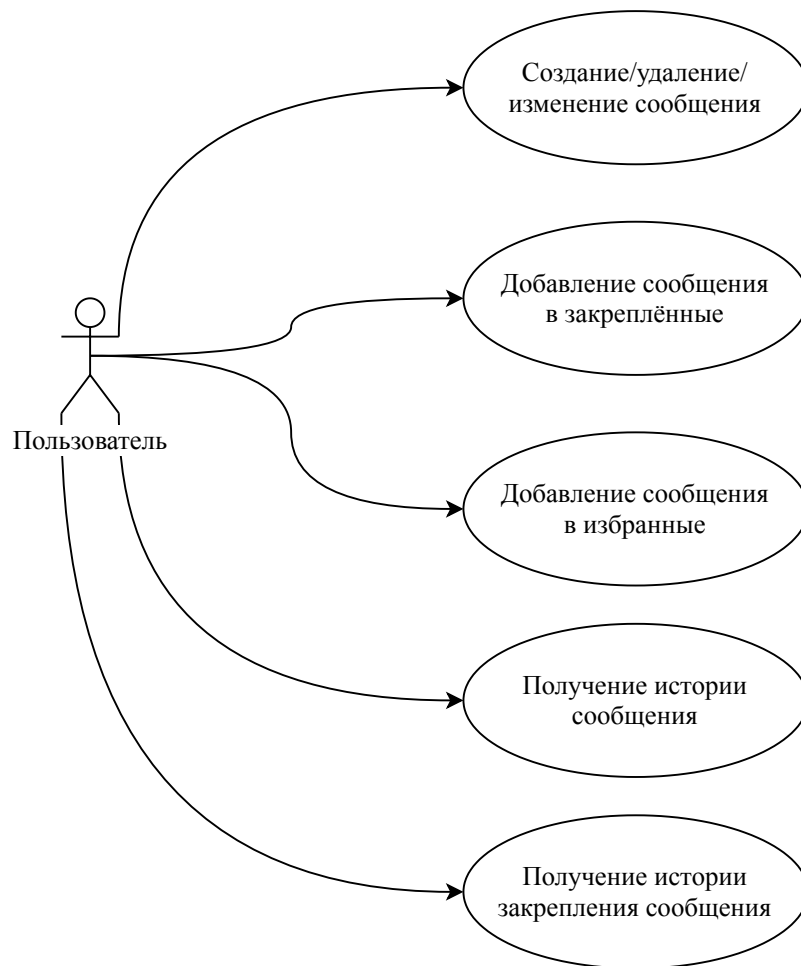


Рисунок 2.4 — Use Case диаграмма для действий с сообщениями.

2.3 База данных

Далее были выделены сущности предметной области и построена ER-диаграмма, которая содержит 14 таблиц. Диаграмма представлена на рисунке 2.5. Таблица 2.1 содержит описание сущностей базы данных.

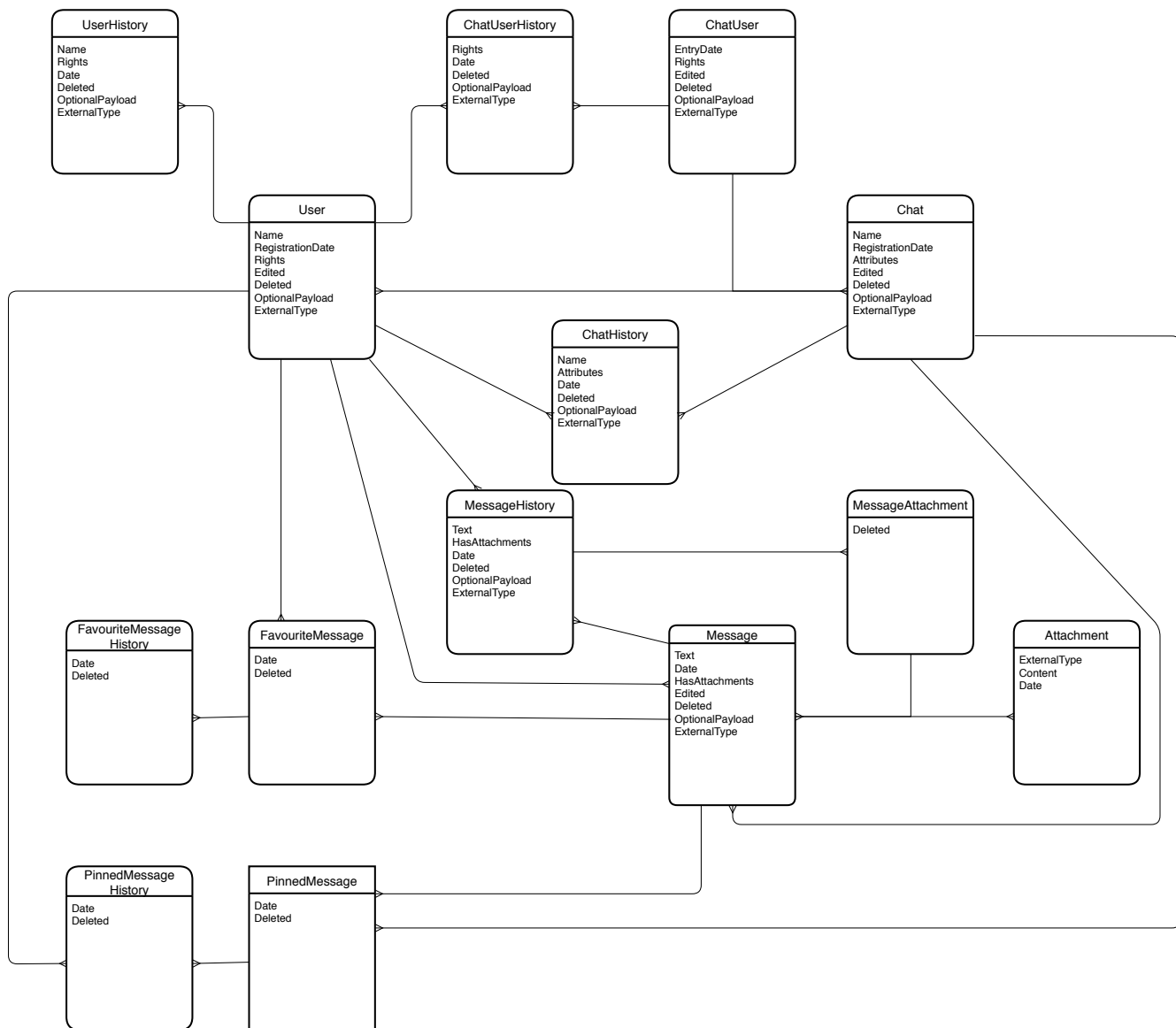


Рисунок 2.5 — ER-диаграмма базы данных.

Таблица 2.1 — Описание сущностей базы данных.

Название таблицы	Описание
User	Пользователи
UserHistory	История изменений пользователей
Chat	Чаты
ChatHistory	История изменений чатов
ChatUser	Пользователи чатов (соединительная таблица между пользователями и чатами)
ChatUserHistory	История изменений пользователей чата
Message	Сообщения
MessageHistory	История изменений сообщений
Attachment	Вложения
MessageAttachment	Вложения сообщений (соединительная таблица между вложениями и сообщениями)
FavouriteMessage	Избранные сообщения пользователя
FavouriteMessage History	История изменений избранных сообщений пользователя
PinnedMessage	Закрепленные сообщения в чатах
PinnedMessage History	История изменений закрепленных сообщений в чатах

2.4 Проектирование архитектуры

Необходимо спроектировать абстрактный микросервис мессенджера. На рисунке 2.6 представлена диаграмма компонентов микросервиса.

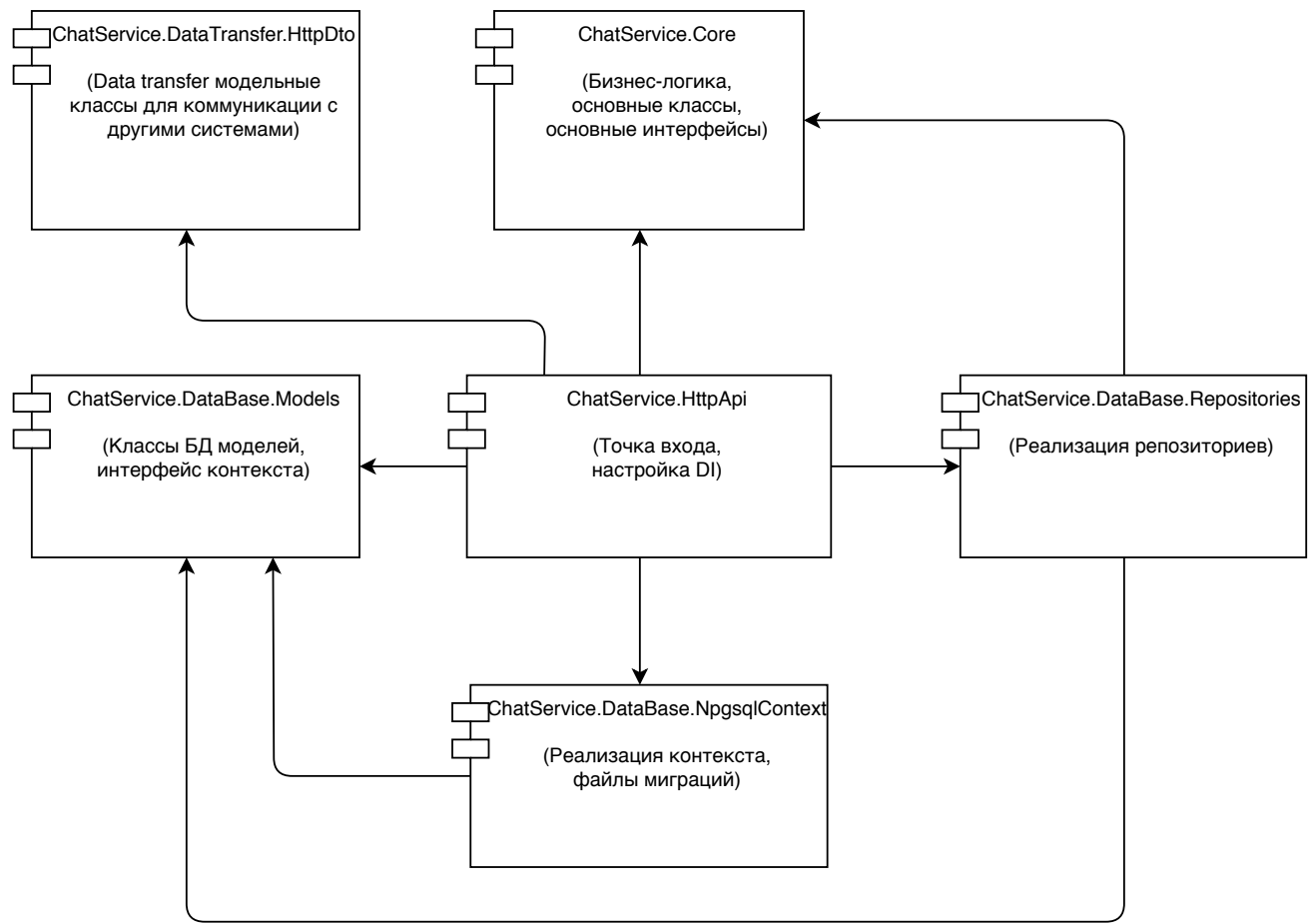


Рисунок 2.6 — Диаграмма компонентов микросервиса.

1. ChatService.HttpApi

`ChatService.HttpApi` – точка входа в проект, инициализирует и конфигурирует IoC контейнер, здесь происходит настройка DI.

Смысл IoC (Inversion of Control/инверсия управления) заключается в том, что модули верхнего уровня не должны зависеть от реализации модулей нижнего уровня [10]. То есть, например, компонент в приложении, который отвечает за модели базы данных, не должен зависеть от конкретной используемой СУБД. Вместо этого передается интерфейс.

IoC контейнер знает о всех интерфейсах и их реализациях в системе и умеет их сопоставлять. Делается это через DI.

DI (Dependency Injection/внедрение зависимостей) – это процесс предоставления внешней зависимости программному компоненту. Перед началом работы с контейнером необходимо зарегистрировать известные типы и их сопоставления (интерфейс \rightarrow реализация).

2. ChatService.Database.Models

В компоненте ChatService.Database.Models описываются классы сущностей базы данных из рисунка 2.5 и интерфейс IChatServiceContext.

3. ChatService.Database.NpgsqlContext

ChatServiceContext – класс, который позволяет взаимодействовать с базой данных используя сущностную модель классов. Класс контекста позволяет вам создавать и выполнять запросы, отслеживать изменения в объектах и отображать эти изменения на базу данных

4. ChatService.Database.Repositories

В ChatService.Database.Repositories описываются реализации репозитория.

Паттерн Репозиторий позволяет абстрагироваться от конкретных подключений к источникам данных, с которыми работает программа, и является промежуточным слоем между классами, непосредственно взаимодействующими с данными, и основными классами программы.

Преимущества использования данного паттерна:

- разделение логики (обращаемся только к тем данным, которые нужны);
- абстрагирование от способа хранения данных;
- легкость тестирования (можно передавать заглушку репозитория при тестировании бизнес-логики). Конкретная реализация репозитория регистрируется в IoC.

5. ChatService.Core

ChatService.Core – компонент, содержащий всю бизнес-логику микросервиса мессенджера. Также, в него включены интерфейсы репозитория и собственные модельные классы.

Функциональность данного класса:

- создание, удаление, изменение параметров пользователей;

- создание, удаление, изменение параметров чата;
- добавление, удаление, изменение параметров участников чата;
- отправка сообщений, файлов;
- возможность добавления сообщения в избранное, закрепление сообщений в чатах;
- получение различной доступной информации.

6. **ChatService.DataTransfer.HttpDto**

`ChatService.DataTransfer.HttpDto` – компонент используется для передачи данных между различными приложениями (в данном случае между микросервисом мессенджера и, например, клиентом). Это паттерн `Data Transfer Object`, который может хранить всю необходимую для вызова информацию. Он должен быть сериализуемым для удобной передачи по сети.

2.5 Вывод

Структура микросервиса мессенджера разработана, переход к реализации.

3 Технологический раздел

В соответствии с выбранной задачей – реализация микросервиса мессенджера. Необходимо выбрать средства реализации, создать модули и интерфейс, описать ограничения и порядок работы программы.

3.1 Выбор технологий

Для выполнения проекта был выбран язык программирования C# [11]. Язык является постоянно развивающимся, объектно-ориентированным. Он относится к семье языков с C-подобным синтаксисом, разрабатывался как язык программирования прикладного уровня для CLR (Common Language Runtime, общезыковая исполняющая среда).

Для разработки используется платформа ASP.NET Core [12], предназначенная для создания различного рода веб-приложений: от небольших веб-сайтов до крупных веб-порталов и веб-сервисов. С помощью ASP.NET Core мы можем создавать кросс-платформенные приложения. Благодаря модульности фреймворка все необходимые компоненты веб-приложения могут загружаться как отдельные модули через пакетный менеджер Nuget.

Для доступа к данным используется технология Entity Framework Core [13]. EF Core является ORM-инструментом (object-relational mapping – отображения данных на реальные объекты). То есть EF Core позволяет работать базами данных, но представляет собой более высокий уровень абстракции: EF Core позволяет абстрагироваться от самой базы данных и ее таблиц и работать с данными независимо от типа хранилища.

Entity Framework Core поддерживает множество различных систем баз данных. Таким образом, мы можем через EF Core работать с любой СУБД, если для нее имеется нужный провайдер. Для работы с PostgreSQL в проект необходимо добавить через Nuget пакет Npgsql.EntityFrameworkCore.PostgreSQL.

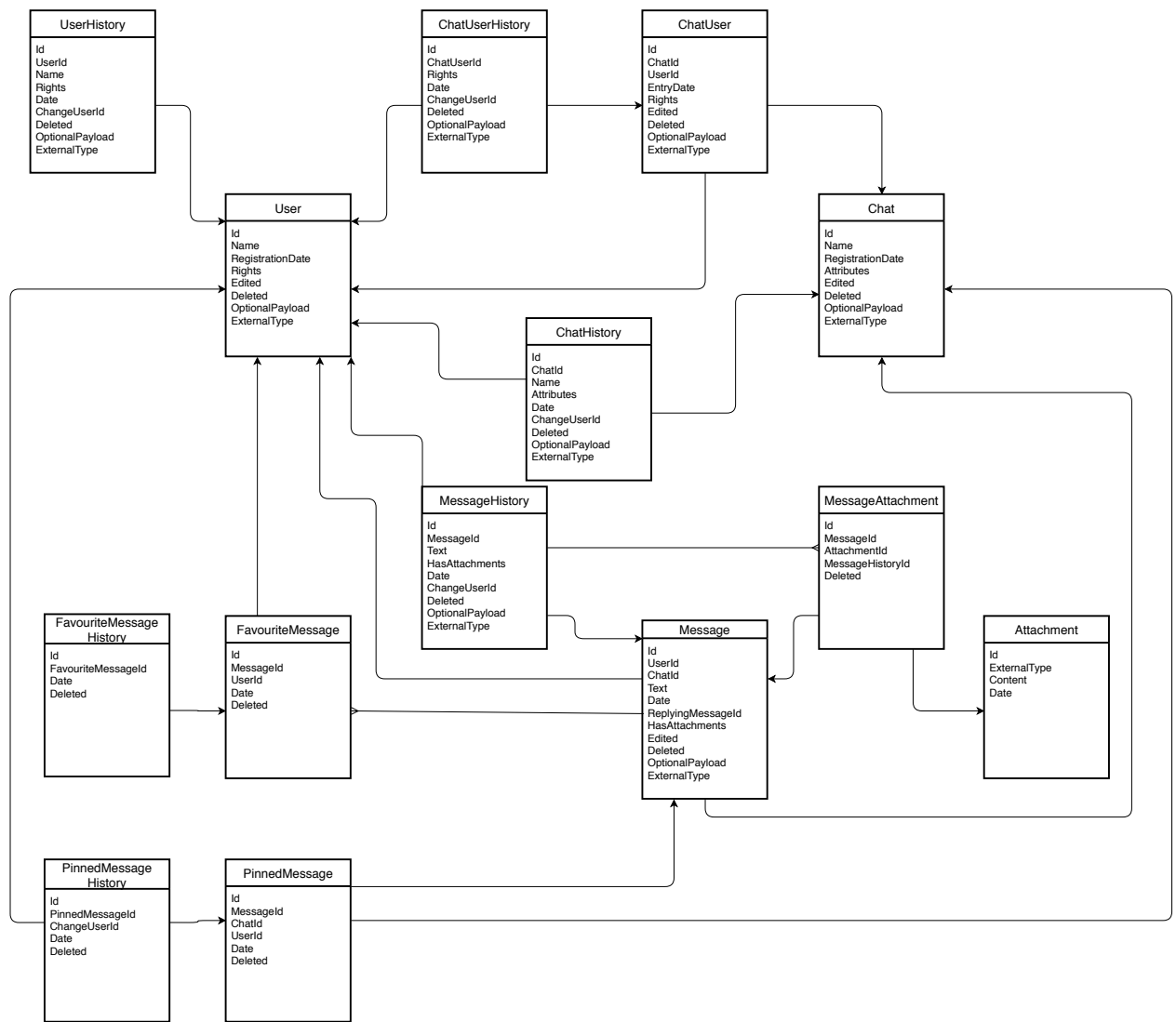
Для взаимодействия между клиентом и сервером используется архитектурный стиль REST [14]. API-интерфейс может считаться RESTful только в том случае, если соблюдены все требования. При создании микросервиса мессенджера были учтены данные требования, и поэтому веб-приложение является RESTful.

Требования REST:

- модель клиент-сервер;
- отсутствие состояния (сервис не хранит информацию о состоянии клиента);
- кэширование;
- единообразие интерфейса (идентификация ресурсов, манипуляция ресурсами через представление, «самоописываемые» сообщения);
- многослойность сервера;
- код по требованию (необязательно).

3.2 Модель базы данных

На рисунке 3.1 представлена модель базы данных.



3.3 UML-диаграммы классов

На рисунке 3.2 представлена UML-диаграмма классов компонента ChatService.Database.Models.

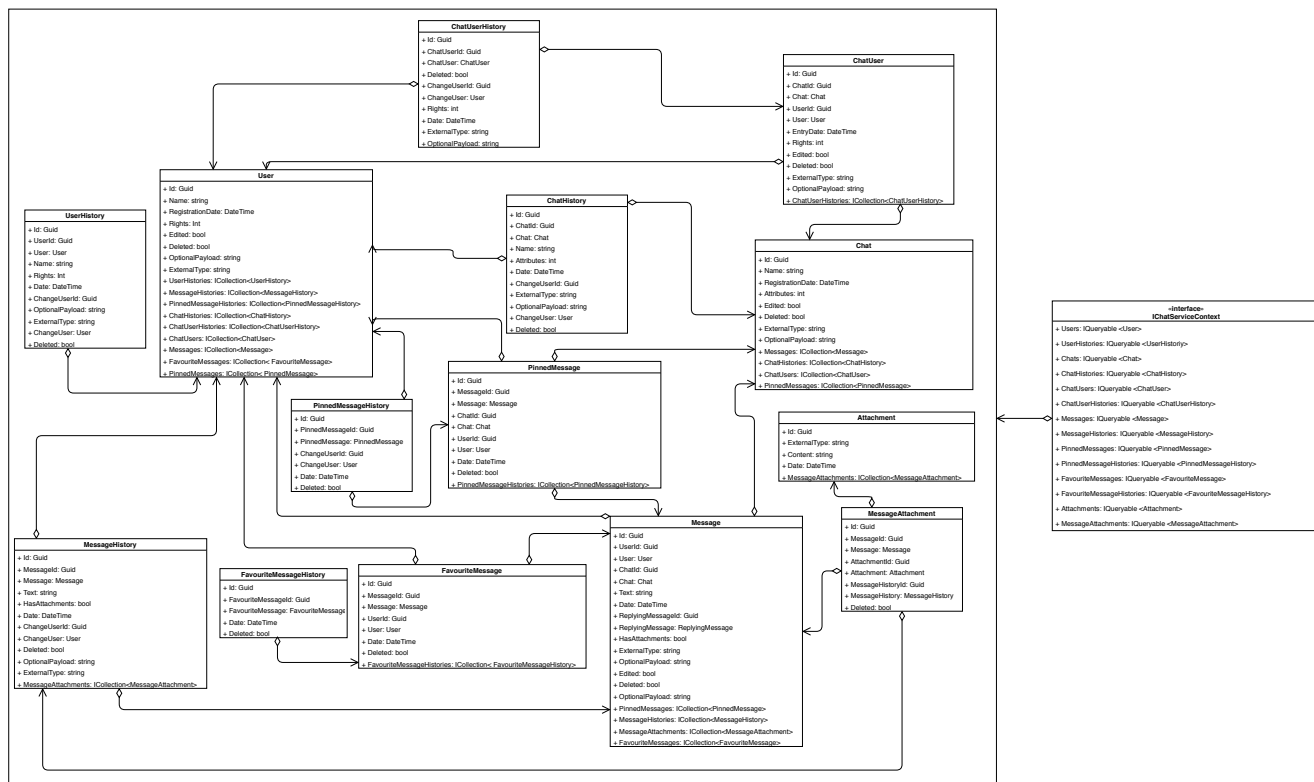


Рисунок 3.2 — UML-диаграмма классов компонента
ChatService.Database.Models.

На рисунке 3.3 представлена UML-диаграмма классов компонента
ChatService.Database.NpgsqlContext.

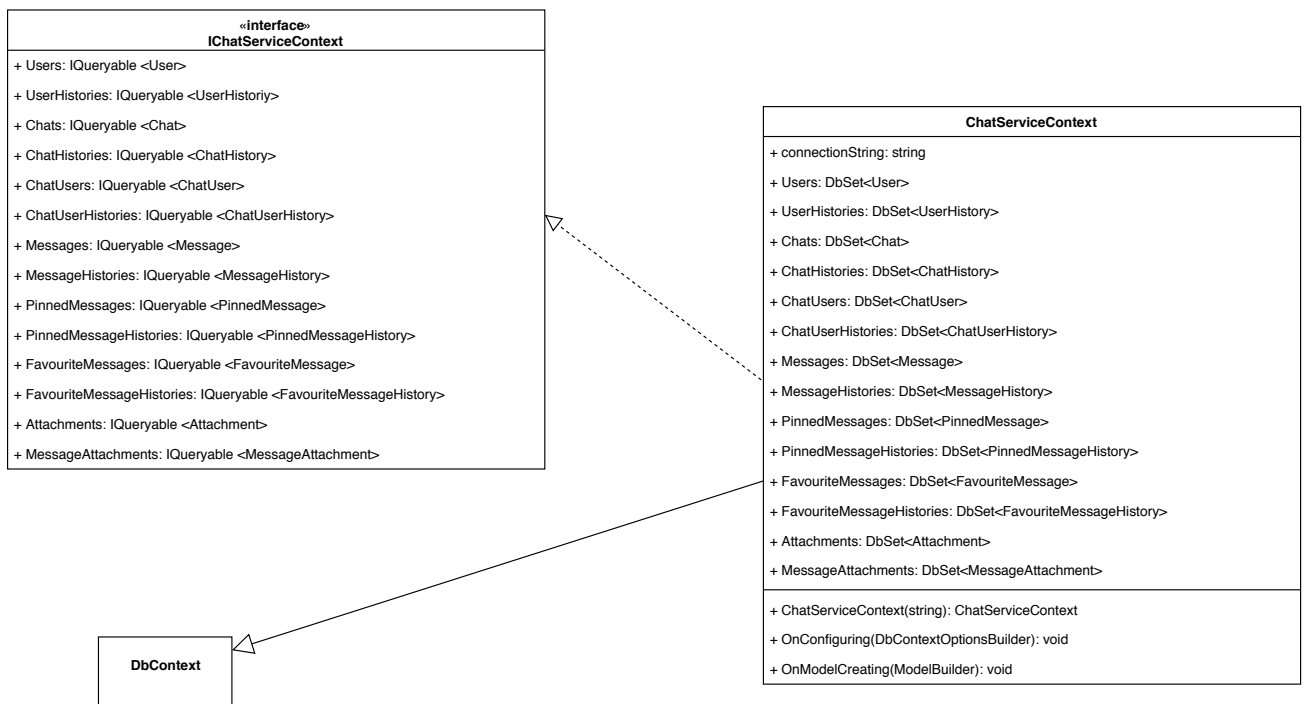


Рисунок 3.3 — UML-диаграмма классов компонента `ChatService.Database.NpgsqlContext`.

На рисунке 3.4 представлена UML-диаграмма классов компонента `ChatService.Core`.

- void DeleteUser(Guid userId, Guid changeUserId) - удаление пользователя
- void ChangeUserName(Guid userId, string newName, Guid changeUserId) - изменение имени пользователя
- void ChangeUserRights(Guid userId, int newRights, Guid changeUserId) - изменение прав пользователя
- IEnumerable<Chat> GetUserChats(Guid userId) - получение списка чатов пользователя
- IEnumerable<Message> GetUserPinnedMessages(Guid userId) - получение списка избранных сообщений пользователя
- IEnumerable<UserHistory> GetUserHistory(Guid userId) - получение всей истории пользователя
- IEnumerable<ChatUserHistory> GetUserChatsHistory(Guid userId) - получение истории чатов пользователя
- void CreateChat(string name, int attributes, string description, Guid changeUserId) - добавление чата
- void DeleteChat(Guid chatId, Guid changeUserId) - удаление чата
- void ChangeChatName(Guid chatId, string newName, Guid changeUserId) - изменение имени чата
- void ChangeChatDescription(Guid chatId, string newDescription, Guid changeUserId) - изменение описания чата
- void ChangeChatAttributes(Guid chatId, int newAttributes, Guid changeUserId) - изменение атрибутов чата
- IEnumerable<Message> GetChatMessages(Guid chatId) - получение сообщений чата
- IEnumerable<Message> GetChatPinnedMessages(Guid chatId) - получение припinnedных сообщений чата

- `IEnumerable<ChatHistory> GetChatHistory(Guid chatId)` - получение истории чата
- `void CreateChatUser(Guid chatId, Guid userId, string name, int rights, Guid changeUserId)` - добавление пользователя в чат
- `void DeleteChatUser(Guid chatUserId, Guid changeUserId)` - удаление пользователя из чата
- `void ChangeChatUserName(Guid chatUserId, string newName, Guid changeUserId)` - изменение имени пользователя чата
- `void ChangeChatUserRights(Guid chatUserId, int rights, Guid changeUserId)` - изменение прав пользователя чата
- `IEnumerable<User> GetChatUsers(Guid chatId)` - получение пользователей чата
- `IEnumerable<ChatUserHistory> GetChatUsersHistory(Guid chatId)` - получение истории пользователей чата
- `void CreateMessage(Guid userId, Guid chatId, string text, Guid replyingMessageId, string type, List<Attachment> messageAttachments)` - добавление сообщения
- `void DeleteMessage(Guid messageId, Guid changeUserId)` - удаление сообщения
- `void ChangeMessageText(Guid messageId, string newText, Guid changeUserId)` - изменение текста сообщения
- `void ChangeMessageAttachments(Guid messageId, IEnumerable<Attachment> newMessageAttachments, Guid changeUserId)` - изменение вложений сообщения
- `void ChangeMessagePinnedForUser(Guid userId, Guid changeUserId)` - припинивание/отпинивание сообщения для пользователя
- `void ChangeMessagePinnedForChat(Guid chatId, Guid changeUserId)` - припинивание/отпинивание сообщения для чата

- IEnumerable<MessageHistory> GetMessageHistory(Guid messageId)
- получение истории сообщения
- IEnumerable<PinnedMessageHistory>
GetMessagePinnedHistory(Guid messageId) - получение истории припинивания сообщения

3.4 Тестовое клиентское приложение

Реализовано с помощью

Запуск приложения представлен на рисунке 3.5.

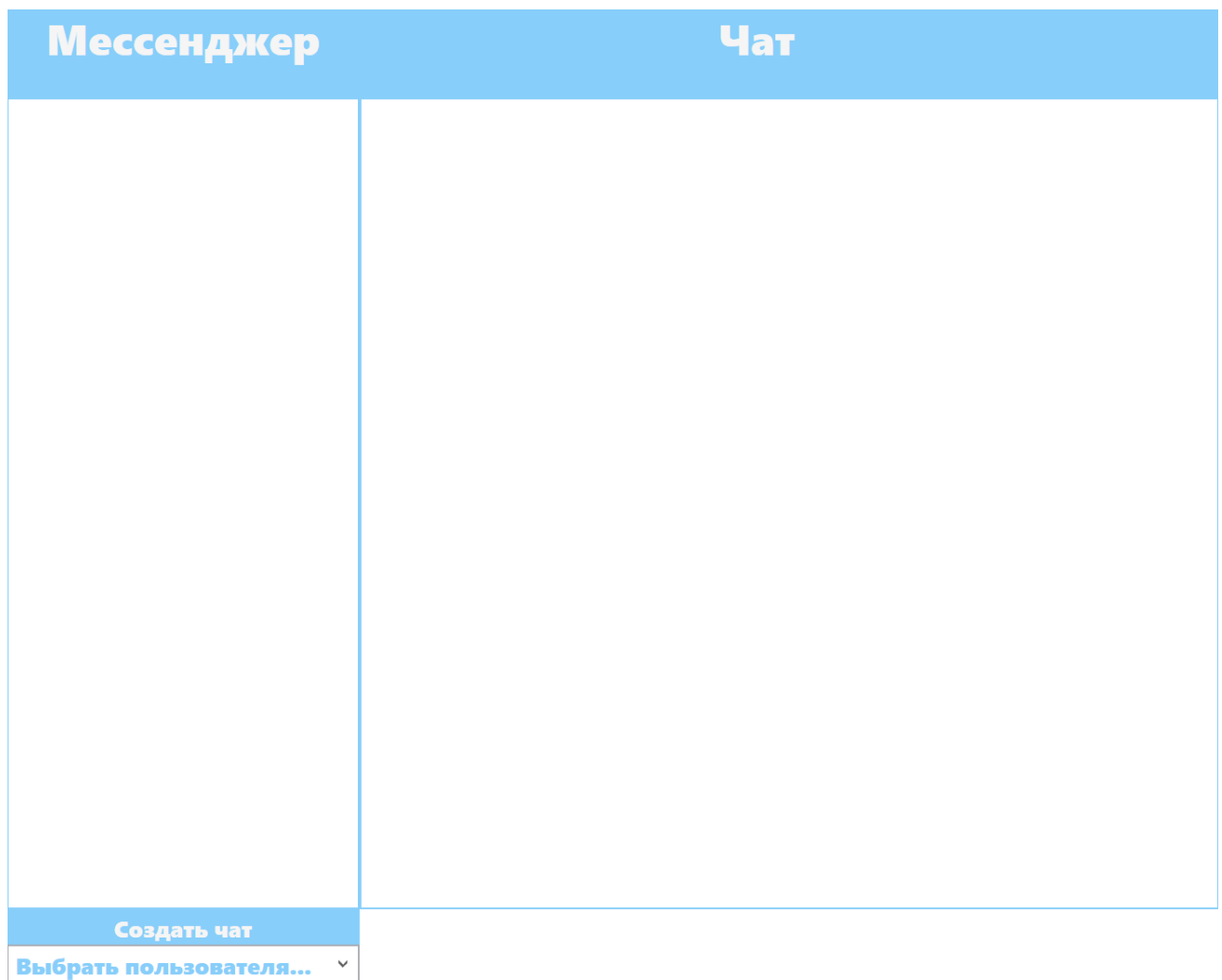


Рисунок 3.5 — Начальный экран приложения.

Далее можно выбрать текущего пользователя, от имени которого будут происходить дальнейшие действия, рисунок 3.6.

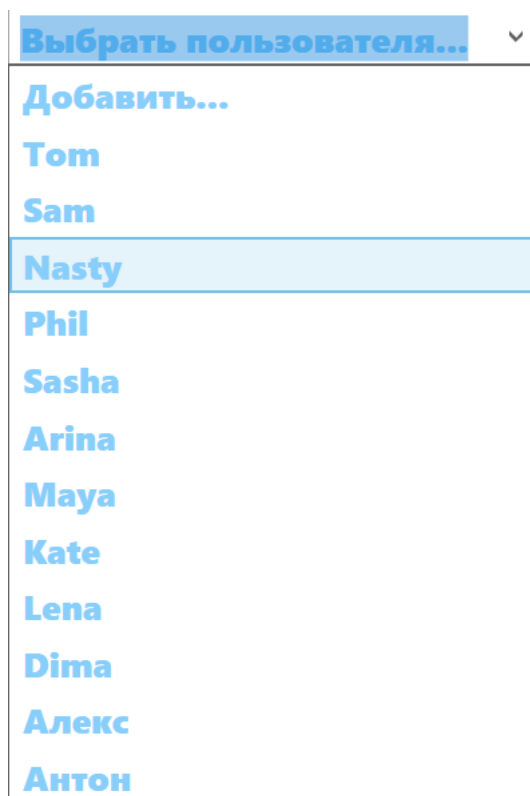


Рисунок 3.6 — Выбор пользователя.

Далее можно выбрать чат, рисунок 3.7.

Сообщения чата обновляются каждую секунду. Отправить сообщение можно либо по нажатию на кнопку, либо по нажатию на клавишу "Enter". Отправка пустого сообщения невозможна.

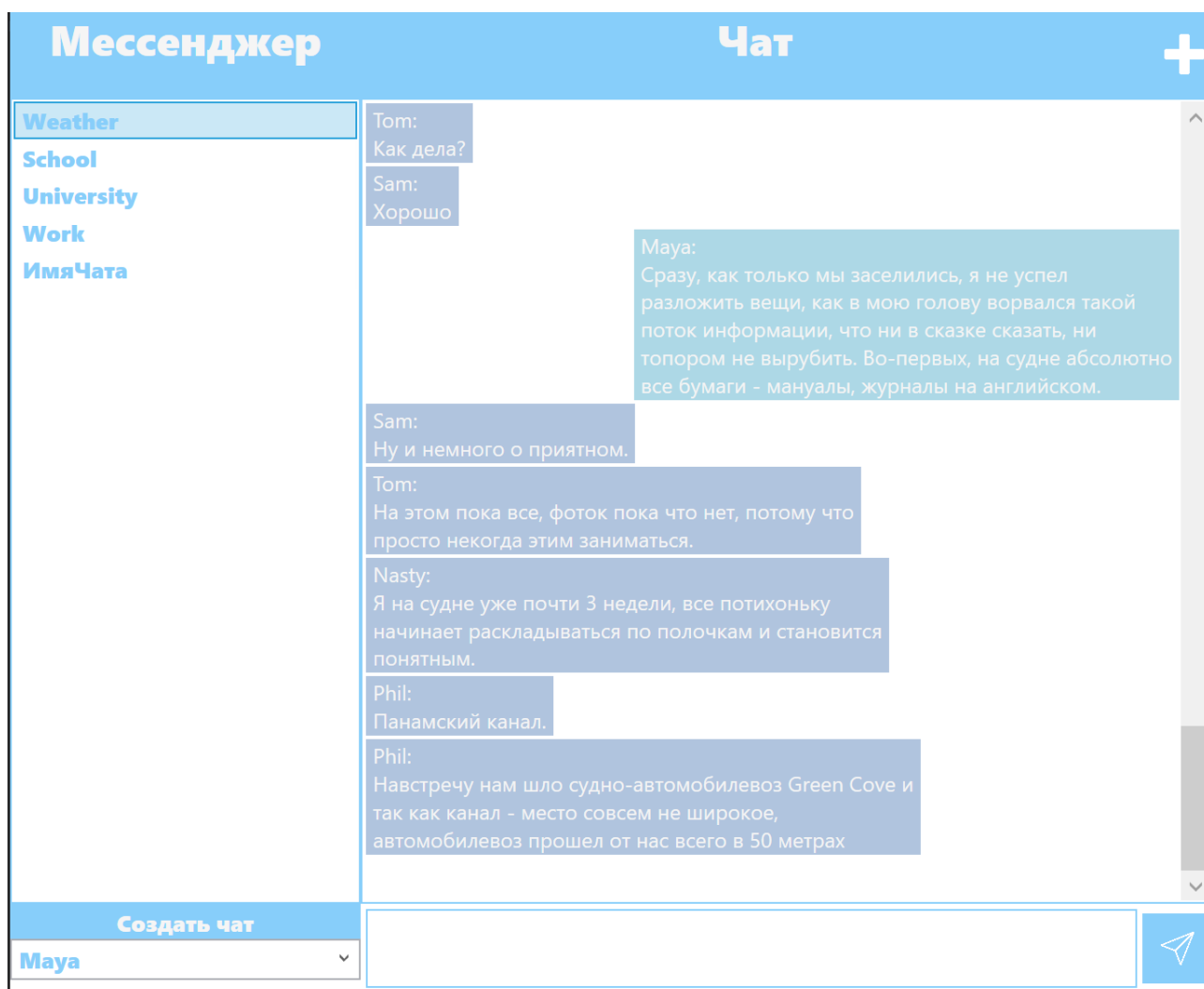


Рисунок 3.7 — Выбор чата, отображение сообщений.

Есть возможность из интерфейса создать новый чат (3.8), нового пользователя 3.9 или добавить пользователей (одного или нескольких) в уже существующий чат 3.10.

Введите имя чата:

Имя

ОК
Отмена

Рисунок 3.8 — Создание чата.

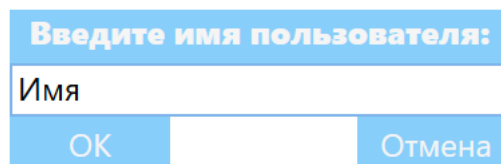


Рисунок 3.9 — Создание пользователя.

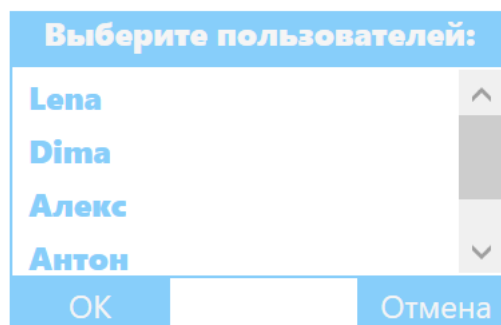


Рисунок 3.10 — Добавление пользователя/пользователей чат.

3.5 Администрирование БД

С использованием pgAdmin (программное обеспечение, предоставляющее графический интерфейс для работы с базой данных) было проверено корректное создание базы данных с необходимыми ключами.

На рисунке 3.11 показано существование таблицы в БД.

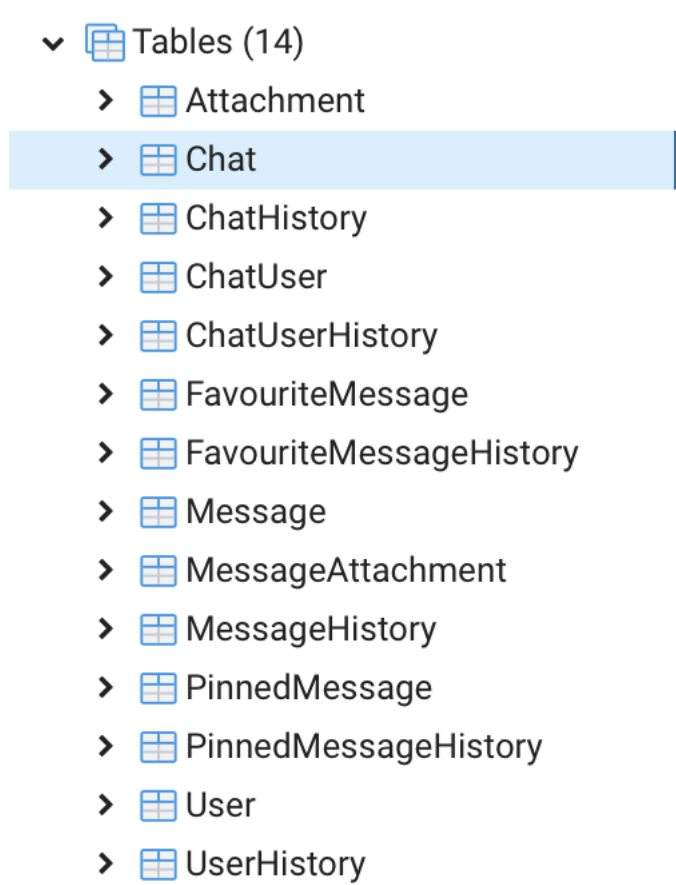


Рисунок 3.11 — Таблицы базы данных.

На рисунке 3.12 показан пример корректности ключей для таблицы чатов.

Type	Name	
Primary Key	public.PK_Chat	auto
Foreign Key	public.ChatHistory.FK_ChatHistory_Chat_ChatId	normal
Foreign Key	public.ChatUser.FK_ChatUser_Chat_ChatId	normal
Foreign Key	public.Message.FK_Message_Chat_ChatId	normal
Foreign Key	public.PinnedMessage.FK_PinnedMessage_Chat_ChatId	normal

Рисунок 3.12 — Ключи таблицы чатов.

3.6 Вывод

Реализовано спроектированное ПО, представлены результаты.

ЗАКЛЮЧЕНИЕ

В результате проделанной работы выполнены следующие задачи:

- а) проанализированы существующие решения;
- б) спроектирован микросервис мессенджера;
- в) реализовано спроектированное ПО.

Достигнута цель проекта – реализация микросервиса универсального мессенджера.

Реализован микросервис на языке программирования C# с использованием платформы ASP.NET Core и технологии Entity Framework Core.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Вся статистика интернета на 2020 год — цифры и тренды в мире и в России. — Режим доступа: <https://www.web-canape.ru/business/internet-2020-globalnaya-statistika-i-trendy/> (дата обращения: 25.04.2020).

2. Корпоративные мессенджеры: 20 вариантов для командной работы. — Режим доступа: <https://mentamore.com/biznes/korporativnye-messendzhery.html> (дата обращения: 25.04.2020).

3. Подборка командных мессенджеров. — Режим доступа: <https://vc.ru/services/54447-podborka-komandnyh-messendzherov-2018-god1> (дата обращения: 25.04.2020).

4. Slack. — Режим доступа: <https://slack.com/> (дата обращения: 25.04.2020).

5. Microsoft Teams. — Режим доступа: <https://www.microsoft.com/ru-ru/microsoft-365/microsoft-teams/free> (дата обращения: 25.04.2020).

6. Twist. — Режим доступа: <https://twist.com/ru/home/> (дата обращения: 25.04.2020).

7. Discord. — Режим доступа: <https://discord.com/> (дата обращения: 25.04.2020).

8. Rocket.Chat. — Режим доступа: <https://rocket.chat> (дата обращения: 25.04.2020).

9. Understanding SQL and NoSQL Databases and Different Database Models. — Режим доступа: <https://www.digitalocean.com/community/tutorials/understanding-sql-and-nosql-databases-and-different-database-models> (дата обращения: 25.04.2020).

10. SOLID. — Режим доступа: <https://www.geeksforgeeks.org/solid-principle-in-programming-understand-with-real-life-examples/> (дата обращения: 10.05.2020).
11. Документация по C#. — Режим доступа: <https://docs.microsoft.com/ru-ru/dotnet/csharp/> (дата обращения: 10.05.2020).
12. ASP.NET Core. — Режим доступа: <https://metanit.com/sharp/aspnet5/1.1.php> (дата обращения: 10.05.2020).
13. Entity Framework Core. — Режим доступа: <https://metanit.com/sharp/entityframeworkcore/1.1.php> (дата обращения: 10.05.2020).
14. REST. — Режим доступа: <https://www.restapitutorial.com/> (дата обращения: 10.05.2020).