# 1. Постановка и анализ задачи.

## 1.1 Описание предметной области

В наше время современный бизнес требует активного общения с клиентами в удобной для них среде. На данный момент самым удобным способом для общения люди выбирают мессенджеры. И действительно: проще написать кому-либо или оставить голосовое сообщение, а затем ответить, когда появится время и придет уведомление.

Клиенты – простые люди со своими вкусами, предпочтениями и кругом общения. И если сейчас взглянуть на список популярных мессенджеров, то мы обнаружим что их не один или два, а десятки. А отвечать только в одном – значит потерять клиентскую базу, которым просто будет неудобно написать в выбранном мессенджере. Но даже если выделить несколько наиболее популярных мессенджеров в сфере продаж и работать только с ними, то остаются следующие проблемы:

- нет возможности у некоторых популярных мессенджеров открывать несколько приложений на разных компьютерах для одного номера телефона;

- нет группировки чатов по темам;

- хранение переписки и важной информации зависит от компании, предоставляющей мессенджер.

Для решений этих проблем можно соединить всё необходимое в одном месте, в приложении - агрегаторе. На данный момент существует множество аналогов, например, Umnico, МультиЧат, ЕАДЕСК, Chat2Desk. Данные агрегаторы предоставляют возможность общаться с клиентами через различные мессенджеры и отслеживать статистику менеджеров. Для общения в них необходимо покупать аккаунт для каждого отдельного менеджера. Также для доступа к разным мессенджерам необходимо платить ежемесячную плату.

## 1.2 Диаграмма Вариантов использования

Управление агрегатором мессенджеров производиться не только непосредственно операторами, которые отвечают на сообщения клиентов. Также необходимы администраторы для контроля качества операторов, регистрации аккаунтов и других руководящих целей. Для этого разработаны два уровня доступа к данным. На диаграмме использования их можно разделить на два действующих лица:

- менеджер;

- администратор.

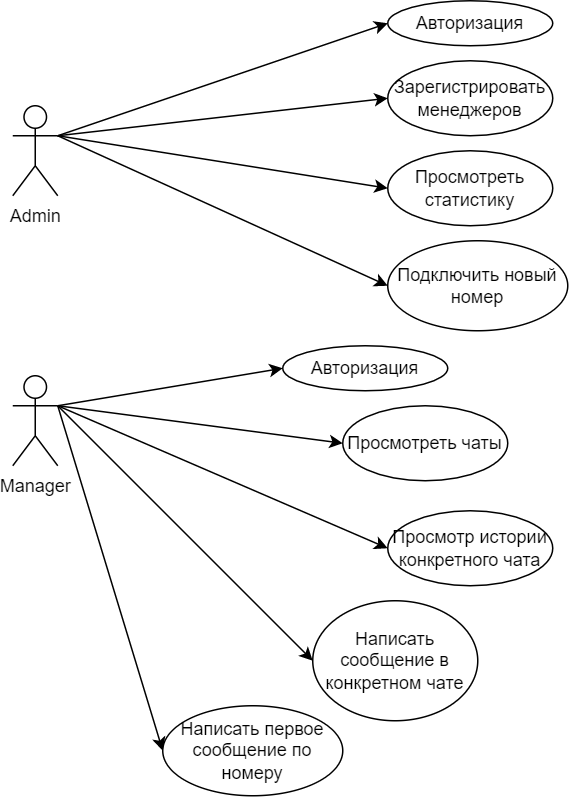


Рисунок 1 Диаграмма вариантов использования

Для работы программы требуется реализовать:

- регистрация номеров/аккаунтов мессенджеров;

- отправка/получение сообщений;

- регистрация менеджеров в системе;

- отслеживание статистики.

## 1.3 Обоснование выбора средств реализации

Для того, чтобы понять, как разрабатывать данный проект, первым делом потребуется изучить API актуальных мессенджеров. Тут ставится задача по определению возможностей API. Например, популярный мессенджер Viber предоставляет Viber REST API, который по своей сути является ботом. А значит имеет ограничение на отправку сообщений первым, то есть клиент должен написать первым, чтобы диалог состоялся. Это не всегда удобно, но другого выбора в Viber нет. Также бот имеет ограничение на отправку бесплатных сообщений в 25000, что может послужить причиной увеличения стоимости пользования этим мессенджером.

Наиболее большой выбор для разработчика предоставляет Telegram: на сайте представлено три вида API для использования. Первый и самый простой – это схожий с Viber ботом Bot API. Имеет тоже очень важное ограничение на отправку сообщений первым. Второй способ и самый сложный – это Telegram API. Этот API позволяет создать собственный клиент Telegram. Он на 100% открыт для разработчиков и на нем как раз и написан Bot API. Следующий способ TDLib. TDLib наиболее подходит для внедрения Telegram к себе в проект: оно инкапсулирует в себя возможности Telegram API и предоставляет понятный интерфейс, который можно изучить в подробной документации на сайте.

Также VK предоставляет все необходимый возможности для работы с зарегистрированной группой в виде REST.

Наименее доступным для разработки студентом является WhatsApp Business API, так как имеет абонентскую плату. Данная API рассчитана на большой бизнес и внедрять ее сразу же, не имея капитала – невозможно.

Из всего этого можно сделать вывод, что есть возможность реализовать общение с мессенджером через доступные интерфейсы в виде REST или написанной библиотеки TDLib.

Но для использования других мессенджеров можно прибегнуть к написанию самостоятельного приложения для получения информации без открытого API. Для этого будет использоваться Selenium WebDriver – драйвер, позволяющий моделировать поведение пользователя в браузере. Данное приложение будет неспособно обрабатывать асинхронно многие команды, например, такие как отправка сообщения, так как оно моделирует поведения пользователя в браузере. И выполнение команд может занимать довольно продолжительное время. Поэтому необходимо реализовать очередь команд, чтобы иметь возможность хранить их продолжительное время, а также сортировать в зависимости от важности. Для данной задачи хорошо подходит брокер сообщений RabbitMQ.

Для реализации проекта хорошо подходит микросервисная архитектура, так как различные части будут работать отдельно, использовать разные технологии.

Данный проект требует написания большого объема кода, связанного с серверной и клиентской частью. Для реализации серверной части потребуется код с возможностью:

- обрабатывать REST запросы;

- работать с web-драйвером Selenium;

- работать с брокером сообщений RabbitMQ;

- создавать микросервисную архитектуру.

С данными задачами справляется платформа разработки .NET, в частности ASP.NET, позволяющая создавать веб-приложения на платформе .NET.

Также к плюсам данной платформы можно отнести использование в качестве основного языка разработки – C#, который является языком со строгой типизацией и построен для использования преимущественно объектно-ориентированного подхода реализации программ.

Клиентская часть проекта должна быть выполнена на JasvaScript, так как обмен между браузером клиента и сервером не должен прекращаться. Клиент должен узнавать обо всех изменениях на сервере без перезагрузки страниц.

Под данную задачу был выбран фреймворк React с использованием языка программирования TypeScript. React выбран как один из самых популярных фреймворков для написания браузерных приложений с большим количеством доступного обучающего материала. TypeScript представляет язык программирования на основе JavaScript. Причины выбора TypeScript:

- строго типизированный и компилируемый язык. На выходе компилятор создает JavaScript, который затем исполняется браузером. Однако строгая типизация уменьшает количество потенциальных ошибок, которые могли бы возникнуть при разработке на JavaScript;

- TypeScript реализует многие концепции, которые свойственны объектно-ориентированным языкам, как, например, наследование, полиморфизм, инкапсуляция и модификаторы доступа и так далее;

- потенциал TypeScript позволяет быстрее и проще писать большие сложные комплексные программы, соответственно их легче поддерживать, развивать, масштабировать и тестировать, чем на стандартном JavaScript.