# 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.

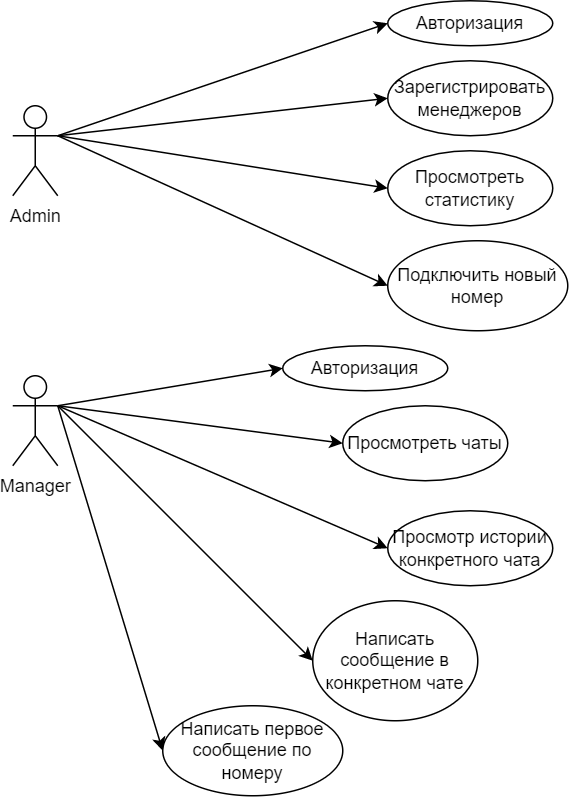


Рисунок 1 Диаграмма вариантов использования

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.

# 2 Анализ данных

Данные, с которыми работает система можно разделить на несколько групп. Входные данные – данные поступающие от пользователя в систему. Промежуточные данные – это данные, используемые системой во время работы. Выходные данные – выводимая системой информация.

В качестве входной информации в приложении выступают следующие сведения:

- данные для авторизации и аутентификации пользователя;

- данные для передачи в виде сообщений (текст, картинки, документы).

К выходным данным можно отнести:

- различные страницы для отображения диалогов с клиентами;

- данные переданные от клиентов.

## 2.1 Логическая структура данных

Одним из преимуществ при использовании приложения является хранение всех данных непосредственно в базе данных. Для этого была разработана структура для хранения всех сообщений и их данных различных мессенджеров. Помимо сообщений база данных содержит информацию о каждом диалоге и техническую информацию.

При разработке был структуры данных был применен подход Code-First. Он позволяет сначала описать структуры в виде классов, с которыми придется работать программисту, а затем уже сгенерировать таблицы в базе данных. Так как в конечном итоге в приложении работают структуры под управлением СУБД – можно представить данные в виде ER диаграммы. Описание каждой таблицы и её свойств находится в приложении А. На рисунке 2 показаны связи между структурами данных.

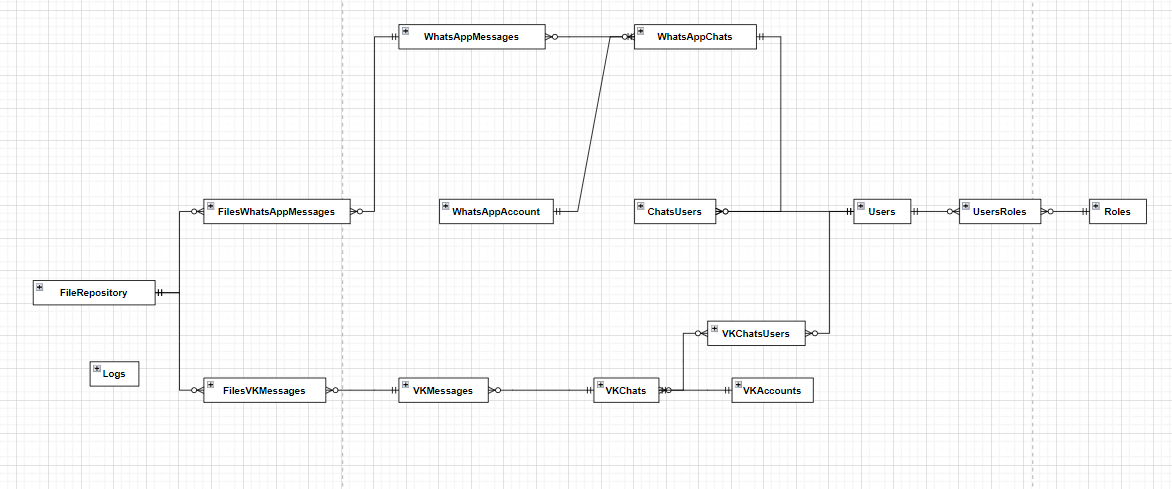


Рисунок 2 Связи между структурами данных

# 3 Программная реализация

## 3.1 Архитектура приложения

Для разработки приложения использовалась микросервисная архитектура. Приложение состоит из следующих сервисов:

- клиентское приложение;

- сервис для хранения управления данными приложения (ядро);

- сервис для управления WhatsApp номерами.

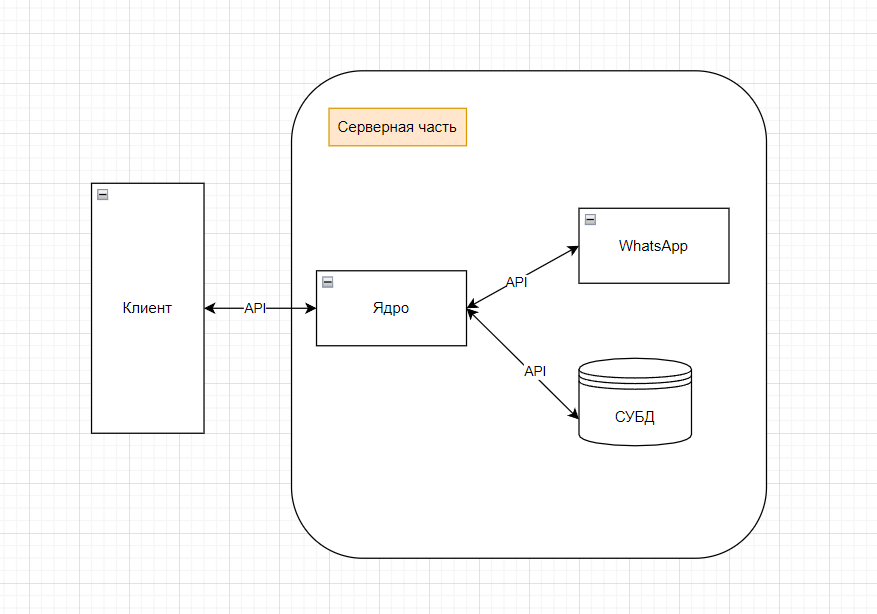


Рисунок 3Архитектура веб-приложения

Клиентское приложение – это программа, запускаемая в браузере пользователя, которая предоставляет удобный интерфейс для взаимодействия управления чатами.

Ядро – набор сервисов для управления, хранения и организации данных от различных мессенджеров. С ядром «общаются» все остальные сервисы. Клиентская часть получает необходимые данные только от ядра, а сервис для работы с номерами WhatsApp отправляет все необходимые данные ядру. Модуль для общения с VK API находится также в этом сервисе.

WhatsApp сервис – это приложение для обработки данных номеров WhatsApp в режиме реального времени.

СУБД – это система управления базами данных, которая хранит все необходимые данные. Описание таблиц находится в приложении А.

## 3.2 Реализация клиентской части

Для клиентской части была выбрана Jasvascript-библиотека пользовательских интерфейсов React. Библиотека опирается на создании инкапсулированных компонентов с собственным состоянием и их объединении в сложные пользовательские интерфейсы.

Для клиентской части были разработаны следующие компоненты:

- ProtectedRoute – маршрутизатор, отвечающий за предоставление информации на основе jwt-токенов (ролей пользователей);

- Chat – компонент чата;

- ChatList – список чатов доступных для данного пользователя

- Message – компонент сообщения в чате;

- MessagesList – список сообщений чата;

- LoginForm – форма входа в приложение;

- InitializeCard – форма инициализации номера;

- AccountInit – компонент инициализации аккаунта;

- ChatInit – компонент инициализации отдельного чата.

## 3.3 Реализация серверной части

Под серверной частью приложения понимается набор сервисов работающих удаленно от компьютера пользователя на одном или нескольких серверных компьютерах. Эти сервисы следует рассмотреть отдельно.

Для реализации ядра использовалась платформа разработки веб-приложений ASP.NET. Данная платформа позволяет легко создавать REST API для работы с данными и работать с gRPC – системой удаленного вызова процедур.

Ядро выполняет несколько похожих задач:

- общение с клиентской частью через REST API для данных, не требующих постоянного обновления;

- отправка уведомлений клиентской части в режиме онлайн;

- получение/отправка данных сервису WhatsApp;

- аутентификация и авторизация пользователей.

Для доступа к данным ядро производит проверку авторизации. Авторизация производится на основе jwt-токенов, позволяющих определить роли доступа к информации, которую предоставляет ядро.

Для решения первой задачи был написан RESTful интерфейс, позволяющий получить данные авторизированным пользователям.

Клиентская часть находится в браузере, что делает невозможным постоянный обмен информацией между сервером и клиентом без использования javascript. Поддержания постоянных обновлений в таком случае может быть реализовано тремя способами:

- websockets – протокол связи поверх TCP-соединения, предназначенный для обмена между браузером и сервером;

- server-side events – технологи отправки уведомлений сервером поверх HTTP;

- long polling – дословно «длинные опросы», постоянные опросы сервера, на предмет наличия обновления.

Для облегчения решения данной задачи была использована библиотека SignalR Core, инкапсулирующую в себе все три приведенных выше подхода.

Задачу общения с сервисом WhatsApp можно разделить на две подзадачи. Первая - отправка команд для выполнения, которые должны быть организованы в очередь и выполняться сервисом WhatsApp в том числе и по их приоритету. Вторая - получение данных от сервиса WhatsApp в асинхронном режиме.

Для решения первой подзадачи выбран программный брокер сообщений RabbitMQ, позволяющий организовывать очереди с приоритетами и отвечающий за обязательную доставку сообщения.

Для второй – система удаленного вызова процедур gRPC, использующую в качестве транспорта HTTP/2, что делает данный способ более быстрым решение нежели RESTful сервис.

# 4 Тестирование

В процессе разработки программного продукта параллельно производится его тестирование и отладка. Первый тест был направлен на способность приложения стабильно работать при некорректно введенных данных пользователем. Для обработки таких ситуаций была разработана система toast-оповещений, которая помогает пользователю понять, что именно он сделал неправильно.

Второй тест был направлен на ограничение множественных запросов от пользователя пока сервер обрабатывает поступившие данные. Для реализации такого ограничения было принято решение делать кнопки запросов неактивными на время пока сервер не даст ответ.

Третий тест был направлен на ограничение по типу файла, который пользователь может выбрать при отправки его на сервер. Для реализации такого ограничения в файловый проводник был установлен фильтр на типы файлов, которые пользователь может выбрать.

На текущий момент система находится в активной стадии разработки. При добавлении новых функций в систему производится тестирование и отладка добавленного функционала, а также производится поиск и устранение уязвимых мест в уже реализованной системе.

ПРИЛОЖЕНИЕ А

(обязательное)

Описание таблиц базы данных

Таблица А.1 – Таблицы базы данных

|  |  |
| --- | --- |
| Название таблицы | Описание |
| Roles | Хранит роли пользователей |
| Users | Информация о пользователях |
| WhatsAppChats | Информация о чатах whatsapp |
| WhatsAppMessages | Информация о сообщениях whatsapp |
| WhatsAppAccounts | Информация об аккаунтах whatsapp |
| VKChats | Информация о чатах VK |
| VKMessages | Информация о сообщениях VK |
| VKAccounts | Информация об аккаунтах VK |
| FileRepository | Информация о сохраненных файлах |
| Logs | Логи системы |

Таблица А.2 – Описание таблицы Roles

|  |  |
| --- | --- |
| Название | Описание |
| ID | Первичный ключ |
| Name | Название роли |

Таблица А.3 – Описание таблицы Users

|  |  |
| --- | --- |
| Название | Описание |
| ID | Первичный ключ |
| Email | Email пользователя |
| Password | Пароль |

Продолжение приложения А

Таблица А.4 – Описание таблицы WhatsAppChats

|  |  |
| --- | --- |
| Название | Описание |
| ID | Первичный ключ |
| Title | Заголовок чата |
| InitStatus | Статус инициализации |
| InitDate | Дата последнего инициализированного сообщения |

Таблица А.5 – Описание таблицы WhatsAppMessages

|  |  |
| --- | --- |
| Название | Описание |
| ID | Первичный ключ |
| DataId | Уникальный идентификатор в приложении WhatsApp |
| Date | Дата |
| IsOutgoing | Исходящее ли сообщение |
| IsReaded | Прочитано ли пользователем |
| ChatId | Внешний ключ WhatsAppChats |
| Text | Текст сообщения |
| DocumentName | Название документа прикрепленного к сообщению |

Таблица А.6 – Описание таблицы WhatsAppAccounts

|  |  |
| --- | --- |
| Название | Описание |
| ID | Первичный ключ |
| Number | Номер аккаунта |
| InitState | Статус инициализации аккаунта |

Продолжение приложения А

Таблица А.7 – Описание таблицы VKChats

|  |  |
| --- | --- |
| Название | Описание |
| ID | Первичный ключ |
| VKID | Идентификатор в БД VK |
| Title | Заголовок чата |

Таблица А.8 – Описание таблицы VKMessages

|  |  |
| --- | --- |
| Название | Описание |
| ID | Первичный ключ |
| VKID | Идентификатор в БД VK |
| Text | Текст сообщения |

Таблица А.9 – Описание таблицы VKAccounts

|  |  |
| --- | --- |
| Название | Описание |
| ID | Первичный ключ |
| Token | Уникальны токен группы |

Таблица А.10 – Описание таблицы FileRepository

|  |  |
| --- | --- |
| Название | Описание |
| ID | Первичный ключ |
| Hash | Хэш файла (sha256) |
| Name | Название файла в хранилище |
| OridnaryName | Обычное название файла вместе с расширением |

Продолжение приложения А

Таблица А.11 – Описание таблицы Logs

|  |  |
| --- | --- |
| Название | Описание |
| ID | Первичный ключ |
| Text | Текст лога |
| Date | Дата |