Техническое задание

1. Введение;  
2. Основания для разработки;  
3. Назначение программного продукта;  
4. Требования к программе;  
5. Требования к программной документации;   
7. Стадии и этапы разработки;  
8. Порядок контроля и приемки;  
9. Приложение (опционально)

**1 ВВЕДЕНИЕ**

Программа называется BesMes – это мессенджер для ПК под управлением операционных систем Windows/Linux.

Основное назначение данной системы – использование его в качестве средства связи между сотрудниками внутри одной компании.

Система включает в себя три программы:

1. BesMes Server – серверное приложение.

2. BesMes Client – клиентское приложение.

3. BesMes Admin Panel – панель администратора.

Серверное приложение размещается в локальной сети или на специально выделенном серверном устройстве, которое будет ответственно за обработку всей активности, происходящей в мессенджере.

Клиентские приложения распространяются в виде установочного пакета на конечные устройства пользователей.

Панель администраторов позволяет администраторам корпоративной сети, если такие имеются,

**2 ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ**

Проект «Мессенджер BesMes» - задание преподавателя по дисциплине «Проектная деятельность».

**3 НАЗНАЧЕНИЕ ПРОГРАММНОГО ПРОДУКТА**

**Функциональное назначение**

Клиентское приложение BesMes предоставляет возможность налаживание коммуникаций в какой-либо организации посредством обмена сообщениями в рамках локальной сети.

Серверное приложение обрабатывает запросы клиентских приложений, собирает об активности пользователей различную информацию.

Панель администратора

Мессенджер BesMes предоставляет компаниям возможность организовать локальное средство для общения друг с другом. Наш проект предназначен для удобного, быстрого и самое главное безопасного обмена сообщениями и файлами.

**Эксплуатационное назначение**

Клиентское приложение позволяет пользователям отправлять сообщения другим пользователям, создавать беседы с неограниченным количеством участников, прикреплять к сообщениям файлы.

Панель администратора позволяет администраторам просматривать статистику пользователей на сервере, удалять пользователей, лишать их привилегий. С помощью панели администратор может приостановить работу серверного приложения или изменить параметры конфигурации.

**4 ТРЕБОВАНИЯ**

**4. 1 Требования к функциональным характеристикам**

Проект состоит из трёх компонентов:

1. Серверное приложение
2. Средство просмотра сервера
3. Клиентское приложение

Также в системе необходимо реализовать базу данных и протокол общения с сервером BesProtocol.

**4. 1. 1 Требования к базе данных проекта**

Для хранения данных о пользователях, сообщениях, созданных беседах используется база данных MySql.

Необходимо реализовать в базе данных следующие таблицы:

1. user – хранит зарегистрированных в системе пользователей
2. ??? что-то про chat (хранит и message

**Таблица user**

Обязательная информация, которая должна быть указана при регистрации: имя, фамилия, адрес электронной почты, пароль от аккаунта и дата регистрации.

Опциональная информация, которая может быть указана по желания пользователя: пол, день рождения.

**Таблица chat**

Каждая беседа – это новая таблица, записи которой – сообщения.

В таблице chat хранится следующая информация:

**4. 1. 2 Требования к протоколу общения BesProtocol**

Данный протокол будет использоваться клиентским приложением и средством просмотра сервера для общения с сервером и получения от него необходимых данных

Этот протокол использует кодировку UTF-8. ?

Необходимо, чтобы команды этого протокола состояли только из кириллических символов.

Команда протокола в общем случае имеет следующий вид:

*“<КОД КОМАНДЫ> SP <аргумент 1> SP <аргумент 2> SP … <аргумент N>\r\n”,*

где <КОД КОМАНДЫ> - либо символьный идентификатор команды, например, «РЕГИСТРАЦИЯ» или числовой код команды, например, 12

Необходимо проработать команды:

1. Аутентификации (вход пользователя в аккаунт)
2. Регистрации нового пользователя
3. Отправки кода верификации регистрации
4. Синхронизации клиента с сервером(получение всех новых сообщений клиентом)
5. Отправка сообщения
6. Отправка файла

Каждая команда протокола имеет список ошибок, которые могут возникнуть при выполнении команды. Каждой ошибки в соответствие ставится код ошибки.

Если команда выполнилась успешно, сервер возвращает клиенту в начале сообщения символ “+”. Если во время выполнения команды произошла ошибка, клиенту возвращается сообщение, начинающееся с “-“, за минусом идёт код ошибки.

**Типичный алгоритм общения сервера с клиентом**

(К) – Клиент, (С) - Сервер

(К) – подключается к серверу по защищённому соединению

(С): “+ Привет! Вы успешно подключились к серверу BesMesServer. Пожалуйста, пройдите аутентификацию или зарегистрируйтесь.\r\n”

(К): “ПРИВЕТ [ivanov@gmail.com](mailto:ivanov@gmail.com) 12345\r\n”

(С): “+ Вы успешно прошли аутентификацию!\r\n”

или

(С): “- 2 Не существует пользователя с такой почтой и паролем\r\n”

**Оптимизация протокола**

Для уменьшения доли передаваемого сетевого трафика каждой команде BesProtocol в соответствие ставится числовой код из интервала [0..99]. Мы предполагаем, что протокол не будет содержать более 100 команд,

**4. 1. 3 Требования к серверному приложению**

Серверное приложение – консольное приложение. Должно быть разработано с использованием фреймворка QT.

Информация для настройки сервера должна храница в файлах конфигурации:

1. Файл для настройки сервера
2. Файл для настройки подключения к базе данных
3. Файл для настройки используемых почт для отправки кодов верификации

**4.1.4 Требование к взаимодействию средства просмотра сервера с серверным приложением**

Средство просмотра сервера обменивается с сервером строками-командами, формат которых определяется протоколом BesMesProtocol.

**4.1.5 Требования к средству просмотра сервера**

Средства просмотра сервера – графическое приложение. Должно быть разработано с использованием фреймворка QT.

**4.1.6 Требование к взаимодействию клиентского приложения с серверным приложением**

Клиентское приложение обменивается с сервером строками-командами, формат которых c

Необходимо реализовать два разных подключения клиента к серверу:

1. Первое подключение будет использоваться для передачи команд на сервер и принятия ответов на эти команды

2. Второе подключение будет использоваться для отправки и получения файлов

Такое разделение «областей влияния» подключений позволит производить отправку на сервер командных запросов, даже если сервер занят передачей огромного файла.

**4. 1. 7 Требования к клиентскому приложению**

Клиентское приложение – графическое приложение. Должно быть разработано с использованием фреймворка QT. Для графической части должна использоваться технология qml.

В клиентском приложении нужно реализовать следующий функционал:

1. Регистрация нового аккаунта

2. Аутентификация существующего пользователя

3. Выход из аккаунта

4. Отправка сообщения другому пользователю

5. Создание новой беседы

6. Удаление существующей беседы

**4. 2 Требования к надежности**

**Сервер**

Сервер должен игнорировать неизвестные команды, которые не поддерживаются протоколом BesProtocol.

Во время своей работы сервер заполняет журнал сообщений, куда записывает: отладочную информацию(опцию можно отключить), ошибки (ошибка при подключении к базе данных), системная информация(информация о запуске сервера/приостановке работы сервера).

Серверное приложение должно быть максимально гибким, оно должно брать все необходимые настройки из файлов конфигурации

**Протокол**

Для каждой команды протокола BesProtocol определены коды ошибок, которые могут возникнуть при выполнении тех или иных команд.

**Передача данных**

Программа должна обеспечивать безопасную передачу данных. Этого нужно добиться, используя асинхронное шифрование, предоставляемое протоколом TLS v.3+.

Для шифрования мы выбираем алгоритм SHA256. Сгенерированный приватный ключ должен содержать 2048 символов. Это надёжно защитит данные от дешифровки.

Следует ограничить размер передаваемых файлов. Максимум – 100 Мб. Объём обусловлен требованиями корпоративного сегмента по передаче данных.

**4. 3 Условия эксплуатации**

Требования к климатическим условиям не предъявляются.

**Требуемая квалификация и уровень подготовки пользователя**

Для настройки серверного приложения требуются минимальные знания в сетевых технологиях.

Для использования клиентского приложения не требуется никакой квалификации, т.к. все элементы привычны пользователям и встречаются во многих мессенджерах

**4. 4 Требования к составу и параметрам технических средств**

Минимальные системные требования:

ОС: Windows 7 и выше, Linux, Ubuntu

Объём ОЗУ: 2 Гб

Место на диске: 1 Гб

Процессор: не менее 1ГГц

Серверное приложение ориентировано на многопоточную обработку данных, поэтому его следует разворачивать на устройствах, которые удовлетворяют следующим техническим характеристикам.

Рекомендуемые системные требования для серверного приложения:

ОС: Windows 7 и выше, Linux, Ubuntu

ОЗУ: 4 Гб

Места на диске: 5 Гб

Процессор: 4-ёх ядерный процессор, частотой не менее 3 ГГц и поддержкой технологии многопоточности (Intel HTT или AMD SMT)

**4. 5 Требования к информационной и программной совместимости**

**4.5.1 Требование к исходным кодам и языкам программирования**

Исходные коды программы должны быть написаны на C++ и JavaScript

Для разработки проекта BesMes выбраны языки C++, фреймфорк Qt5 (версия 5.12), база данных MySql.

**4.5.2**. **Требования к программным средствам, используемым программой**

Программа для хранения данных использует бесплатную СУБД MariaDB. Для организации пользовательского интерфейса используется фреймворк Qt5.

Программа разрабатывается под операционные системы Windows/Linux.

**4. 6 Требования к маркировке и упаковке**

Программа распространяется в виде установочных пакетов: мастер установки для серверного приложения, для клиентского приложения и для панели администратора.

Программное обеспечение должно иметь маркировку с обозначением наименования изделия, ФИО разработчиков и года выпуска программы

**4. 7 Требования к транспортированию и хранению**

Для отслеживания хода работы использовать систему контроля версий GitHub. Каждое приложение хранится в своём репозитории.

Ссылки на репозитории проекта:

Серверное приложение: https://github.com/SyrtcevVadim/BesMesClient

Панель администратора: https://github.com/SyrtcevVadim/BesMesAdminPanel

Клиентское приложение: https://github.com/SyrtcevVadim/BesMesServer

**4. 8 Специальные требования**

**5 ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ**

5. 1 Состав программной документации

* "Мессенджер BesMes" Техническое задание
* "Мессенджер BesMes" Конвенции наименований

5.2 Специальный требования к программной документации

Документы к программе должны быть выполнены в соответствии с ГОСТ 19.106-78

**6 СТАДИИ И ЭТАПЫ РАЗРАБОТКИ**

**7 ПОРЯДОК КОНТРОЛЯ И ПРИЁМА**

**8 ПРИЛОЖЕНИЕ**

Ссылки на репозитории проекта:

Серверное приложение: https://github.com/SyrtcevVadim/BesMesClient

Панель администратора: https://github.com/SyrtcevVadim/BesMesAdminPanel

Клиентское приложение: https://github.com/SyrtcevVadim/BesMesServer

Ссылки на макеты в Figma:

Макет средства просмотра панели администратора:

https://www.figma.com/file/NZ9f92Ofzkzak6y7SyQeSH/BesMesServer?node-id=0%3A1

Макет клиентского приложения

https://www.figma.com/file/19ifmovXJHrXFpWv1dDCpD/BesMes?node-id=0%3A1

Всё, что ниже – вспомогательные материалы

**Анализ предметной области**

Как нами было выяснено, самым правильным подходом в построении приложения для обмена сообщениями является использование клиент-серверной архитектуры.

Есть некоторый центральный сервер, который принимает и обрабатывает все запросы клиентских приложений. Сервер сохраняет всю информацию, которая к нему поступает:

1. Все отправляемые пользователями сообщения сохраняются в общую базу данных. Мы будем использовать PostgresSQL
2. Хранится информация о каждой пользовательской сессии
3. Хранится информация о каждом отправляемом на сервер запросе. Эта информация должна находиться в некотором Log-файле.

Для повышения отказоустойчивости сервера, необходимо производить сохранение состояния сервера в некотором файл.

Современное клиентское приложение должно соответствовать следующим требованиям:

1. Иметь минимально возможное время отклика
2. Иметь возможность отправлять текстовые сообщения, голосовые сообщения и файлы.

Неплохим архитектурным решением будет сделать два разных подключения клиента к серверу:

1. Первое подключение будет использоваться для передачи команд на сервер и принятия ответов на эти команды
2. Второе подключение будет использоваться для отправки и получения файлов

Такое разделение «областей влияния» подключений позволит производить отправку на сервер командных запросов даже если сервер занят передачей огромного файла.

Для уменьшения нагрузки на файловое подключение следует ограничить возможный размер передаваемых файлов. Например, популярный мессенджер ВКонтаке не позволяет передавать файлы размером более 256 Мебибайт. В нашей программе мы хотим сделать ограничения в 2 Мебибайта.

Предполагается, что в клиент-серверной архитектуре используется протокол TCP и сокеты для подключения к удалённому серверу. Более того, чтобы обеспечить безопасность передаваемых данных, необходимо использовать защищённые сокеты.

Для взаимодействия с сервером клиент должен использовать некоторый протокол. Есть множество стандартных протоколов для передачи данных, например: SMTP для передачи почтовых сообщений. Однако нет такого протокола, который удовлетворил бы все нужды нашего проекта. Поэтому целесообразно будет разработать собственный протокол. В свете грядущего импортозамещения, разумным решеним будет создать протокол с киррилическими командами, а не с латинскими. Это будет ключевой особенностчью взаимодействия клиента и сервера.

Современные мессенджеры обычно распространяются в виде мобильных приложений. Именно поэтому кроме настольного приложения нами будет разработан и мобильный аналог. Однако может повлечь существенные отличия в пользовательском интерфейсе.

Большинство современных мессенджеров поддерживают механизм создания чат-ботов.

Мы также хотим реализовать эту функцию. Для этого

необходимо сделать некоторую часть интерфейса программы открытой (Open API).

Для обеспечения приемлемой скорости работы приложения, настольная часть проекта будет реализована на языке C++. Мобильное приложение будет реализовано на Java.

Клиент не должен при каждом подключении к серверу подтягивать заново все сообщения во всех диалогах. На клиентской стороне должна работать небольшая база данных, в которой будут храниться все диалоги пользователя локально. Другим вариантом будет хранение всех сообщений в текстовом файле, однако клиентское приложение предполагает поиск сообщений, удаление и неоднократное добавление новых сообщений. Поэтому самым удачным решением будет создание локальной базы данных

Требования к программируемой системе:

1. безопасность передачи сообщений;
2. верификация регистрации аккаунтов пользователей по почте;
3. разделение приложения на серверную и клиентскую части;
4. хранение информации о пользователях и их переписки в базе данных;
5. возможность отправлять текстовые сообщения и различные типы файлов;
6. в клиентском приложении реализовать возможность создания личные чаты и беседы (чат с несколькими пользователями);
7. настройка личных данных и некоторых функций системы (например, звук, тема);
8. предоставление инструментов для создания ботов;
9. отслеживание используемых ресурсов и администрирование участников в серверной части.

Группы пользователей: пользователи, администраторы, сторонние разработчики ботов.

Диаграмма вариантов использования.

// TODO

**Глоссарий**

Open API – открытый программный интерфейс

Мебибайт – 2^20 байт

Чат - диалог между пользователями в рамках приложения

Беседа – чат с несколькими пользователями.

Командный сокет – сокет для отправки на сервер команд

Файловый сокет – сокет для отправки на сервер файлов