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)

**ВВЕДЕНИЕ**

Наш проект называется BesMes - TCP/IP мессенджер для ПК под операционные системы Windows/Linux.

BesMes – безопасный и быстрый TCP мессенджер с открытой архитектурой и исходным кодом. Основное назначение данной системы – использование его в качестве средства связи между сотрудниками внутри одной компании.

**ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ**

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

**НАЗНАЧЕНИЕ ПРОГРАММНОГО ПРОДУКТА**

Наш проект позволяет людям общаться друг с другом

**ТРЕБОВАНИЯ**

**Требования к функциональным характеристикам**

Проект состоит из трёх компонентов:

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

1. Требования к серверному приложению

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

2. Требования к базе данных проекта

Для проекта используется БД MySql.

Необходимо реализовать следующие таблицы:

1. Таблица пользователей user. Хранит следующую информацию:

Имя, фамилия, почта,

3. 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-16.

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

Необходимо проработать команды:

1. Аутентификации
2. Регистрации
3. Отправки кода верификации
4. Синхронизации клиента с сервером(получение всех новых сообщений клиентом)
5. Отправка сообщения
6. Отправка файлов

4. Требования к средству просмотра сервера

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

5. Требования к клиентскому приложению

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

6. Требование к взаимодействию клиентского приложения с серверным приложением

Клиентское приложение обменивается с сервером строками-командами, формат которых определяется протоколом BesMesProtocol.

7. Требование к взаимодействию средства просмотра сервера с серверным приложением

Средство просмотра сервера обменивается с сервером строками-командами, формат которых оп

•требования к надежности («программа должна обеспечивать проверку корректности входных данных» и т.п.)

Программа не должна падать хотя бы первые 5 минут

Объём оперативной памяти приложения не должно превышать 50мб

•условия эксплуатации -требуемая квалификация и уровень подготовки пользователя

•требования к составу и параметрам технических средств-описание требований к hardware

•требования к информационной и программной совместимости -описание требований к software

•требования к маркировке и упаковке

•требования к транспортированию и хранению

•специальные требования

**СТАДИИ И ЭТАПЫ РАЗРАБОТКИ**

**ПОРЯДОК КОНТРОЛЯ И ПРИЁМА**

**ПРИЛОЖЕНИЕ**

Ссылки на репозитории проекта:

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

Средство просмотра сервера: <https://github.com/SyrtcevVadim/BesMesServerViewer>

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

Ссылки на макеты в Figma:

Макет средства просмотра сервера:

<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

**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:

(КАТЯ, НАПИШИ)

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.

Описание проектируемой информационной системы.

(КАТЯ НАПИШИ)

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

**Методы решения поставленных задач в аналогичных системах**

Аналитическая записка ???