**Министерство науки и высшего образования РФ**

Федеральное государственное автономное

образовательное учреждение высшего образования

«Казанский (Приволжский) федеральный

университет»

ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА ТЕОРЕТИЧЕСКОЙ КИБЕРНЕТИКИ

Специальность (направление): 01.03.02 – Прикладная математика и информатика

КУРСОВАЯ РАБОТА

Разработка десктопного клиент - серверного мессенджера на C++

Работа завершена:

Студент гр. 09-812

"\_\_\_"\_\_\_\_\_\_\_\_\_ 20\_\_ г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И. Н. Садыков

Работа допущена к защите:

Научный руководитель

Доцент КТК

"\_\_\_"\_\_\_\_\_\_\_\_\_ 20\_\_ г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. Р. Байрашева

Казань – 2021

Содержание

[Введение 3](#_Toc90602277)

[Глава I. Проектирование приложения 5](#_Toc90602278)

[Основные понятия и концепции используемые при разработке приложения 5](#_Toc90602279)

[Используемые инструменты 7](#_Toc90602280)

[Глава II. Реализация приложения 9](#_Toc90602281)

[Заключение 9](#_Toc90602282)

[Список литературы 10](#_Toc90602283)

[Исходный код программы 11](#_Toc90602284)

# Введение

**Актуальность работы.** Сейчас сложно представить современный мир без сети интернет. В нем мы делаем покупки, получаем образование, развлекаемся, общаемся. Свободный обмен информацией является одним из основных преимуществ глобальной паутины. Большую часть времени в интернете люди проводят в различных социальных сетях, мессенджерах.

Основой таких приложений и всего интернета в целом является стек протоколов TCP/IP. Стандартная архитектура таких приложений является клиент – серверной. Архитектура клиент-сервер - это вычислительная модель, в которой сервер хранит и обрабатывает большую часть ресурсов и служб, которые потребляются клиентом. В архитектуре этого типа один или несколько клиентских компьютеров подключены к центральному серверу через сеть или подключение к Интернету. Поверх транспортного протокола TCP для приложения часто пишут свой более высокоуровневый протокол обмена данными, который адаптирован под конкретную бизнес логику.

Современные тенденции требуют быстрой разработки программного обеспечения, чтобы в короткие сроки его можно было доставить пользователю, заказчику и получать прибыль. В связи с этим очень выгодно разрабатывать кроссплатформенное программное обеспечение. Выпуская свое приложение на разные платформы и операционные системы, можно увеличить количество пользователей, а также сократить количество затраченных ресурсов на разработку, потому что приложение нужно разработать всего раз. Для таких целей используют интерпретируемые языки которые не завязаны на конкретной платформе и исполняют код на виртуальной машине. Но их производительность не такая хорошая как на компилируемых языках. Мой выбор при написании приложения пал на язык C++ с использованием фреймворка Qt, на языке C++ при грамотном использовании можно добиться потрясающей производительности при этом используя высокоуровневые конструкции и ООП, но его стандартная библиотека довольная небольшая и простая, поэтому прибег к использованию кроссплатформенного фреймворка в котором есть все что нужно для разработки сетевого приложения с графическим интерфейсом.

**Цель** **работы –** реализация прикладного программного обеспечения клиент – серверного кроссплатформенного мессенджера.

Задачи которые стоят для достижения цели:

* Проектирование архитектуры клиент – серверного приложения
* Определение модели данных, протокола на которых будет основан клиент сервер
* Реализация приложения в программном коде
* Тестирование

# Глава I. Проектирование приложения

## Основные понятия и концепции используемые при разработке приложения

* Приложение - это компьютерная программа, разработанная для выполнения определенной задачи, отличной от той, которая связана с работой самого компьютера.
* Клиент – программа исполняющаяся на машине пользователя, клиентская программа предоставляют интерфейс, позволяющий пользователю компьютера запрашивать услуги сервера и отображать результаты, возвращаемые сервером.
* Сервер – программа чаще всего на удаленной машине но также может быть и на локальной, ждет запросов от клиентов, а затем отвечают на них. В идеале сервер предоставляет клиентам стандартизированный прозрачный интерфейс, чтобы клиенты не знали о специфике системы (то есть аппаратного и программного обеспечения), предоставляющей услугу.
* Параллелизм - возможность декомпозиции программы на части, которые могут выполняться независимо друг от друга. Это означает, что задачи могут выполняться вне очереди, и результат будет таким же, как если бы они выполнялись по порядку.
* Кросс-платформенная разработка приложений - это создание одного приложения, которое может работать в разных операционных системах, вместо разработки разных версий приложения для каждой платформы. Кросс-платформенность может быть достигнута с помощью интерпретаторов которые переводят байт код в машинный а также фреймворков которые под высокоуровневым конструкциями способны собрать код подходящий под целевую платформу.
* Фреймворк - это абстракция, в которой реализованы общие функциональные возможности, скелет для будущего приложения, обычно это набор функций классов, которые разработчик может расширять, изменять под то поведение которое ему требуется.

## Язык С++

При выборе основного инструмента для разработки я остановился на языке C++. В проекте использую стандарт 17 года.

C++ - один из самых популярных языков программирования в мире, его можно найти в современных операционных системах, графических пользовательских интерфейсах и встроенных системах. С++ мультипарадигменный, компилируемым язык со статической типизацией, чаще всего используют объектно-ориентированный подход, что дает четкую структуру программам и позволяет повторно использовать код, снижая затраты на разработку. C++ является переносимым и может использоваться для разработки приложений, которые можно адаптировать к нескольким платформам.

## Фреймворк Qt

## Структурные особенности проекта с использованием Qt

## Архитектура приложения

# Глава II. Реализация приложения

## План проекта

## Структура проекта

## Протокол взаимодействия компонентов

## Архитектура Клиента

## Архитектура Сервера

## База данных

# Заключение

В своей курсовой работе я достиг, поставленных целей реализовав программный продукт – клиент – серверный мессенджер. В процессе проектирования и разработки использовались проверенные временем концепции и методологии. Данную программу можно развивать и дальше добавив новые функциональные возможности.

# Список литературы

1. Энтони Уильямс. Параллельное программирование на С++ в действии. Практика разработки многопоточных программ
2. Мейерс Скотт. Эффективный и современный С++: 42 рекомендации по использованию C++11 и C++14
3. Gang of Four Design Patterns
4. Andrew S. Tanenbaum. Computer Networks (5th Edition) 5th Edition
5. Qt Documentation: <https://doc.qt.io/>
6. CMake Documentation: <https://cmake.org/documentation/>
7. PostgreSQL Documentation: <https://www.postgresql.org/docs/>
8. SSL: <https://www.cloudflare.com/learning/ssl/what-is-ssl/>
9. OpenSSL Documentation: <https://www.openssl.org/docs/>

# Исходный код программы