Geekbrains

Дипломная работа

Система управления записями малого бизнеса с использованием баз данных и фреймворка Qt

Программа: Разработчик C++. Специалист

ФИО Байлук А.А.

Санкт-Петербург

2025

СОДЕРЖАНИЕ

[1 ТЕОРЕТИЧЕСКИЕ ОСНОВЫ 1](#_Toc205874116)

[1.1 Архитектура клиент-серверных приложений 1](#_Toc205874117)

[1.2 Различие TCP и UDP 2](#_Toc205874118)

[1.3 Возможности фреймворка Qt для разработки GUI 3](#_Toc205874119)

[1.4 Обзор современных систем управления записями 4](#_Toc205874120)

[2 Практическая часть 7](#_Toc205874121)

[2.1 Архитектура базы данных 7](#_Toc205874122)

[2.2 Разработка библиотеки DbManager 10](#_Toc205874123)

[2.3. Разработка серверной части проекта 15](#_Toc205874124)

[2.4 Разработка клиентской части проекта: 16](#_Toc205874125)

[ЗАКЛЮЧЕНИЕ 17](#_Toc205874126)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 18](#_Toc205874127)

ВВЕДЕНИЕ

Современные малые предприятия в сфере услуг всё чаще сталкиваются с необходимостью автоматизации бизнес-процессов. В салонах красоты, парикмахерских и косметологических кабинетах ручное ведение записей в блокнотах или таблицах Excel приводит к ошибкам, дублированию, утечкам информации и снижению качества обслуживания. При этом коммерческие решения зачастую дороги, требуют подписки и выхода в интернет, что делает их недоступными для небольших компаний.

В связи с этим актуальной задачей становится создание бесплатного и простого в использовании приложения, способного эффективно управлять записями клиентов без зависимости от внешних сервисов. Разработка подобного ПО позволяет повысить уровень сервиса, минимизировать человеческий фактор и обеспечить централизованное хранение данных.

Целью дипломного проекта являетсяразработка и реализация клиентского приложения для автоматизации процесса записи клиентов в салон красоты, включающего:

* Удобный графический интерфейс с поддержкой вкладок;
* Формирование и редактирование записей;
* Ведение анкеты клиента;
* Административную панель для отслеживания посещений;
* Отправку данных на сервер по протоколу TCP для централизованного хранения.

Выполнение дипломного проекта осуществляется в несколько этапов:

1. Анализ предметной области — изучение существующих решений, определение требований к системе.
2. Проектирование архитектуры приложения — разработка структуры интерфейса, определение классов и взаимодействий.
3. Реализация графического интерфейса — создание вкладок, таблиц, диалоговых окон без использования Qt Designer.
4. Разработка функционала — добавление, редактирование и удаление записей.
5. Интеграция сетевого взаимодействия — реализация TCP-клиента для отправки данных на сервер.
6. Тестирование и отладка — проверка корректности работы всех компонентов.
7. Оформление результатов — подготовка документации, приложений и выводов.

Проект решает проблемуотсутствия простого, автономного и недорогого решения для автоматизации записи клиентов в малом бизнесе. Предлагаемое приложение позволяет:

* Заменить бумажные журналы и Excel-таблицы;
* Предотвратить конфликты по времени;
* Централизованно хранить данные;
* Обеспечить наглядность и удобство работы для администратора и мастера.

Для выполнения проекта были использованы знания и навыки, полученные в ходе изучения следующих дисциплин:

* Программирование на C++
* Объектно-ориентированное проектирование
* Сетевые технологии
* Графические интерфейсы пользователя

В ходе реализации проекта используются следующие инструменты и технологии:

* Язык программирования: C++17
* Фреймворк: Qt 6
* Среда разработки: Qt Creator / CMake
* Сетевое взаимодействие: QTcpSocket, протокол TCP/IP
* Управление проектом: Git

Проект выполнен в индивидуальном порядке. Автор выполнил все этапы разработки самостоятельно, выступая в следующих ролях:

* Аналитик — анализ требований и проектирование функционала;
* Разработчик (C++/Qt) — реализация интерфейса и логики приложения;
* Сетевой инженер — настройка TCP-взаимодействия;
* Тестировщик — проверка корректности работы всех модулей;
* Документатор — оформление технической документации и приложений.

Хороший пример командной работы: ранее участвовал в учебном проекте по разработке CRM-системы в роли C++-разработчика. В команде также были: руководитель проекта, тестировщик и технический писатель. Полученный опыт позволил эффективно организовать самостоятельную работу над данным дипломным проектом.

# **1 ТЕОРЕТИЧЕСКИЕ ОСНОВЫ**

## **1.1 Архитектура клиент-серверных приложений**

Современные информационные системы всё чаще строятся на основе клиент-серверной архитектуры — модели взаимодействия программных компонентов, при которой одна часть системы (сервер) предоставляет ресурсы или услуги, а другая (клиент) запрашивает и использует их. Эта архитектура является фундаментальной для построения распределённых приложений и широко применяется в веб-сервисах, базах данных, системах автоматизации и других областях информационных технологий.

Клиент-серверная архитектура (англ. Client-Server Architecture) — это модель вычислительной сети, в которой клиент инициирует запрос на выполнение определённого действия, а сервер принимает этот запрос, обрабатывает его и возвращает результат. Клиент и сервер могут находиться на одном компьютере (локально) или в разных узлах сети.

* Клиент — приложение, с которым взаимодействует пользователь. В рамках данного проекта клиентом выступает разрабатываемое настольное приложение на C++ с графическим интерфейсом.
* Сервер — программа или устройство, отвечающее за хранение данных, обработку запросов и обеспечение доступа к ресурсам. В проекте используется простой TCP-сервер, принимающий данные о записях и сохраняющий их для дальнейшего использования.

Взаимодействие между клиентом и сервером осуществляется по заранее определённому протоколу, который задаёт формат сообщений, порядок обмена данными и правила обработки ошибок.

Использование клиент-серверной модели предоставляет ряд существенных преимуществ по сравнению с автономными или одноранговыми (P2P) решениями:

1. Централизованное хранение данных  
   Все данные сосредоточены на сервере, что исключает их дублирование и несогласованность. Это особенно важно в условиях, когда несколько сотрудников работают с одной базой записей. Например, администратор и мастер могут использовать разные клиентские приложения, но получать актуальную информацию с одного сервера.
2. Масштабируемость  
   Систему можно легко масштабировать: добавлять новых клиентов (рабочие места), увеличивать производительность сервера, внедрять резервное копирование. Архитектура позволяет отделить логику от данных, что упрощает модернизацию.
3. Безопасность  
   Доступ к данным контролируется на уровне сервера. Можно реализовать аутентификацию, шифрование, ограничение прав доступа. Даже если клиентское устройство будет утеряно, данные на сервере останутся защищёнными.
4. Упрощённое обновление и сопровождение  
   Обновление логики или структуры данных производится на сервере, и все клиенты автоматически получают изменения при следующем подключении. Это исключает необходимость обновления программного обеспечения на каждом рабочем месте.
5. Надёжность и отказоустойчивость  
   Сервер может быть настроен на резервирование, логирование и автоматическое восстановление, что повышает отказоустойчивость системы в целом.

В рамках данного проекта используется TCP/IP как наиболее подходящий протокол для локальной автоматизированной системы. Он позволяет организовать надёжное соединение между клиентским приложением и сервером без необходимости в интернете, что идеально подходит для салонов красоты, где важна автономность и стабильность работы.

В дипломном проекте реализована клиент-серверная архитектура:

* Клиент — настольное приложение на C++ с использованием фреймворка Qt;
* Сервер — простой TCP-сервер;

Каждое действие пользователя (создание, редактирование, удаление записи) приводит к отправке структурированного JSON-объекта на сервер. Это позволяет:

* Централизованно хранить все записи;
* Обеспечить резервное копирование данных;
* В перспективе — подключить несколько клиентов к одному серверу;
* Организовать логирование событий.

## **1.2 Различие TCP и UDP**

При разработке клиент-серверных приложений, особенно в условиях локальной сети (LAN), выбор сетевого протокола передачи данных играет ключевую роль в обеспечении надёжности, производительности и целостности информации. Наиболее широко используемыми протоколами транспортного уровня модели OSI являются TCP (Transmission Control Protocol) и UDP (User Datagram Protocol). Оба протокола работают поверх IP (Internet Protocol), но существенно различаются по принципам функционирования, что определяет их применимость в различных типах приложений.

TCP (Transmission Control Protocol) — это надёжный, ориентированный на соединение протокол, обеспечивающий упорядоченную и достоверную передачу данных между двумя узлами сети.

Основные особенности TCP:

* Установление соединения (three-way handshake): перед передачей данных клиент и сервер устанавливают логическое соединение.
* Контроль доставки пакетов: каждый пакет подтверждается (acknowledgment), и при потере — автоматически повторяется.
* Упорядоченная доставка: пакеты нумеруются, и на приёмной стороне восстанавливаются в правильной последовательности.
* Контроль потока (flow control): предотвращает переполнение буфера приёмника.
* Контроль перегрузки сети (congestion control): адаптирует скорость передачи в зависимости от состояния сети.

UDP (User Datagram Protocol) — это ненадёжный, без установления соединения протокол, ориентированный на скорость и минимальные накладные расходы.

Основные особенности UDP:

* Отсутствие соединения: данные отправляются без предварительного handshake.
* Нет подтверждения доставки: отправитель не узнаёт, получил ли получатель пакет.
* Нет гарантии порядка: пакеты могут приходить в произвольной последовательности или теряться.
* Минимальные накладные расходы: заголовок UDP — 8 байт против 20+ байт у TCP.
* Высокая скорость передачи: за счёт отсутствия контроля и повторных отправок.

## **1.3 Возможности фреймворка Qt для разработки GUI**

Одним из ключевых факторов успешной реализации настольного приложения является выбор подходящего инструмента для разработки графического пользовательского интерфейса (GUI). В рамках данного проекта в качестве основного фреймворка выбран Qt — мощная, кроссплатформенная библиотека для разработки приложений на языке C++. Qt зарекомендовал себя как надёжное и гибкое решение, широко используемое как в промышленной, так и в учебной разработке. Ниже рассматриваются ключевые возможности Qt, которые были использованы при создании системы управления записями в салоне красоты.

Одним из главных преимуществ Qt является его высокая кроссплатформенность. Приложение, написанное с использованием Qt, может быть скомпилировано и запущено на различных операционных системах без необходимости переписывания кода:

* Windows (7, 10, 11)
* Linux (Ubuntu, Fedora, Debian и др.)
* macOS

Это достигается за счёт абстрагирования от низкоуровневых API операционных систем: Qt предоставляет единый интерфейс для работы с окнами, графикой, шрифтами, событиями ввода и другими компонентами. В условиях проекта это означает, что разработанное приложение может быть использовано в салоне красоты независимо от используемой ОС, что значительно расширяет его практическую применимость.

Кроме того, Qt поддерживает различные системы сборки — как qmake, так и CMake, что позволяет интегрировать проект в современные среды разработки и CI/CD-процессы.

Qt предоставляет обширную библиотеку графических компонентов (виджетов), которые позволяют быстро и эффективно создавать сложные интерфейсы. В рамках проекта были использованы следующие ключевые виджеты:

* QTabWidget — упрощает восприятие интерфейса и позволяет логически разделить функционал.
* QTableWidget — используется для отображения табличных данных. Поддерживает редактирование, выделение, сортировку и настройку размеров столбцов.
* QDateEdit и QTimeEdit — удобные элементы для выбора даты и времени.
* QLineEdit — поля ввода.
* QComboBox — выпадающий список.
* QPushButton — кнопки действий.

Все эти компоненты легко настраиваются, поддерживают стилизацию через стилевые таблицы (Qt Style Sheets) и могут быть вложены в различные компоновки (QVBoxLayout, QHBoxLayout, QFormLayout), что обеспечивает гибкость при проектировании макета.

Помимо GUI, Qt предоставляет мощные инструменты для сетевого программирования. В частности, класс QTcpSocket позволяет реализовать TCP-клиент без использования сторонних библиотек. Он интегрирован в цикл событий Qt и работает асинхронно, что исключает блокировку интерфейса при отправке или приёме данных.

В проекте QTcpSocket используется для:

* Подключения к локальному серверу (127.0.0.1:1122);
* Отправки JSON-сообщений о создании, редактировании и удалении записей;
* Обработки ответов от сервера (например, подтверждения приёма данных).

Сигналы connected, readyRead, errorOccurred позволяют гибко управлять состоянием соединения и обрабатывать ошибки в реальном времени.

Хотя Qt предлагает визуальный редактор Qt Designer для создания .ui-файлов, в данном проекте был выбран **подход** ручного программирования интерфейса на C++. Это позволило:

* Полностью контролировать структуру и иерархию виджетов;
* Избежать привязки к внешним файлам;
* Упростить отладку и сопровождение кода;
* Лучше понять архитектуру Qt (компоновки, сигналы, владение объектами).

Одной из уникальных особенностей Qt является система сигналов и слотов — механизм, позволяющий объектам взаимодействовать без жёсткой привязки. Сигнал генерируется при наступлении события (например, нажатие кнопки), а слот — это функция, которая его обрабатывает.

Сочетание объектно-ориентированного C++ **и** гибкого механизма взаимодействия делает Qt особенно удобным для создания сложных, но хорошо структурированных приложений.

Фреймворк Qt оказался оптимальным выбором для реализации клиентской части системы управления записями. Его кроссплатформенность**,** богатая библиотека виджетов**,** встроенная поддержка сетевого взаимодействия **и** гибкий механизм сигналов/слотов позволили создать функциональное, надёжное и удобное приложение с минимальными временными и ресурсными затратами.

Использование чистого C++ без Qt Designer способствовало более глубокому пониманию архитектуры GUI-приложений и обеспечило полный контроль над поведением интерфейса. Таким образом, Qt не только ускорил разработку, но и стал эффективным инструментом для демонстрации практических навыков программирования на C++ в рамках дипломного проекта.

# 2 Практическая часть

## **2.1 Архитектура базы данных**

Рассмотрим проектную модель классов:

## **2.2 Разработка библиотеки DbManager**

На этапе проектирования любой

## **2.3. Разработка серверной части проекта**

Одной из ключевых целей проведения

## **2.4 Разработка клиентской части проекта:**

1. Одной из ключевых целей проведения

# ЗАКЛЮЧЕНИЕ

В ходе выполнения

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Алексеева О.Г. Методические указания по экономическому обоснованию выпускных квалификационных работ бакалавров: Метод. указания, СПб.: Изд-во СПбГЭТУ “ЛЭТИ”, 2013. с.
2. Горожанина, Е.И. Имитационное моделирование: учебник / Е.И. Горожанина, Е.А. Богданова. – 2-3 изд. – Самара: ПГУТИ, 2023. -300 с.
3. Емельянов, А.А. Имитационное моделирование экономических процессов / А.А. Емельянов, Е.А. Власова, Р.В. Дума – М.: Финансы и статистика, 2002. – 368 с.
4. Кельтон, В. Имитационное моделирование. В. Кельтон, А. Лоу. – СПб.: Питер , 2004. – 848 с.
5. Кутузов О.В. Имитационное моделирование систем массового обслуживания. Практикум на AnyLogic: учебное пособие для вузов/ О.И. Кутузов, Т.М. Татарникова – Санкт-Петербург : Лань, 2025. – 168 с.
6. Лимановская, О.В. Имитационное моделирование в AnyLogic 7: в 2 частях: Часть 1: учебное пособие / О.В. Лимановская – Электрон. текстовые дан. – Екатеринбург: Издательство Уральского университета, 2017
7. Мешечкин, В.В. Имитационное моделирование: учебное пособие / В.В. Мешечкин, М.В. Косенкова; Кемеровский государственный университет. – Кемерово, 2012. – 116 с.
8. Пимонов А.Г., Веревкин С.А., Прокопенко Е.В. Имитационное моделирование: электронное учебное пособие для студентов направления подготовки 38.04.02 (080200ю68) «Менеджмент» / А.Г. Пимонов, С.А. Веревкин, Е.В. Прокопенко; КузГТУ. – Кемерово, 2015.
9. Шеннон, Р. Имитационное моделирование систем – искусство и наука. Пер. с англ. [Текст] / Р. Шеннон. – М.: Мир, 1978. – 420 с.
10. Эльберг, М.С. Имитационное моделирование: учебное пособие / М.С. Эльберг, Н.С. Цынанков; Сибирский Федеральный университет. – Красноярск: СФУ, 2017. – 128 с.
11. Документация по языку AnyLogic [Электронный ресурс]. – URL: <https://anylogic.help/ru/> – (дата обращения: 28.04.2025)
12. Сайт вакансий dreamjob.ru [Электронный ресурс]. – URL: <https://dreamjob.ru/salary/razrabotchik> – (дата обращения: 28.04.2025)