

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
Вычислительная техника

Направление подготовки 09.03.01 «Информатика и вычислительная техника»
Профиль Вычислительные машины, комплексы, системы и сети

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ
РАБОТА БАКАЛАВРА

на тему

Система для организации интернет-продаж

Студент Кузин Кузин Денис Олегович
(подпись, дата)

Руководитель Дорошенко Дорошенко И.Н.
(подпись, дата) 20.06.18

Нормоконтролёр Кучин Кучин А.В.
(подпись, дата) 17.06.18

Работа допущена к защите (протокол заседания кафедры от 25.05.2018 № 38-09-12)

Заведующий кафедрой Пашенко Пашенко Д.В.
(подпись)

Работа защищена с отметкой отм. (протокол заседания ГАК от 19.06.18 № 5)

Секретарь ГАК Гурин Е.И. Гурин
(подпись)

Пенза, 2018

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
Вычислительная техника

«Утверждаю»
Заведующий кафедрой
Д.В. Пащенко
« 20 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Кузин Денис Олегович гр. 14ВВ1 факультета вычислительной техники.
2. Тема работы «Система для организации интернет-продаж».
Тема утверждена приказом ПГУ № 398а/о от "03" мая 2018 г.
3. Руководитель работы доцент кафедры ВТ ПГУ кандидат технических наук Дорошенко И.Н.
4. Задание на работу (назначение разработки, исходные данные и т.п.).
Назначение: система для организации интернет-продаж, предназначенная для взаимодействия клиента с интернет-магазином и удобства отслеживания сделанных заказов.
Возможности системы: просмотр товаров интернет-магазина, редактирование списка товаров, оформление заказа, просмотр сделанных заказов.
5. Перечень подлежащих разработке вопросов: изучение способов организации связи между клиентским приложением и сервером, изучение характерных черт современных СУБД, определение особенностей работы с интернет-магазинами, составление общей схемы работы системы для организации интернет-продаж, а также разработка клиент-серверного приложения.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области и составление общей схемы работы системы	10	12.03.18	Дорошенко
2.	Изучение сетевой модели передачи данных TCP/IP и клиент - серверной архитектуры	10	26.03.18	Дорошенко
3.	Изучение современных серверных СУБД и общих сведений о интернет-магазине	10	9.04.18	Дорошенко
4.	Разработка серверного приложения	20	23.04.18	Дорошенко
5.	Разработка клиентских приложений	20	7.05.18	Дорошенко
6.	Составление выводов	10	14.05.18	Дорошенко
7.	Отладка и тестирование	10	21.05.18	Дорошенко
8.	Оформление выпускной работы	10	28.05.18	Дорошенко

Дата выдачи задания "5" марта 2018 г.

Руководитель бакалаврской работы Дорошенко 5.03.18 Дорошенко И.Н.
(подпись, дата)

Задание к исполнению принял студент Кузин 5.03.18 Кузин Д.О.
(подпись, дата)

Работу к защите допустить
Декан факультета Фионова 18.06.2018 Фионова Л.Р.
(подпись, дата)

РЕФЕРАТ

Бакалаврская работа 117 листов, 37 рисунков, 2 таблицы, 4 источника, 2 приложения.

ИНФОРМАЦИОННАЯ СИСТЕМА, БАЗА ДАННЫХ, ИНТЕРНЕТ-МАГАЗИН, КЛИЕНТ, СЕРВЕР

Цель работы - разработать клиент-серверное приложение, осуществляющее доступ базе данных и работу с ней, в частности, просмотр и редактирование информации.

Объектом разработки является комплекс приложений (клиент и сервер), которые должны, взаимодействуя между собой, осуществлять работу с базой данных.

В результате проведенной работы создан проект, удовлетворяющий поставленной задаче, а именно, создано клиентское приложение, имеющее интуитивно понятный интерфейс для отображения и работы с данными из базы, и серверное приложение, принимающее и направляющее SQL-запросы от клиента в базу данных. Взаимодействие приложений осуществляется по протоколу TCP/IP.

Данный проект рекомендуется использовать при необходимости контроля полученных заказов, сделанных клиентами в интернет-магазине и редактировании информации, хранящейся в БД. При создании интернет-магазина, где используется база данных, с целью упрощения работы с большим количеством информации может быть внедрен данный проект.

					ПГУ 1.090301.14.001 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Кузин Д.О.	<i>Кузин Д.О.</i>		Лит.	Лист	Листов
Провер.		Дорошенко И.Н.	<i>Дорошенко И.Н.</i>			4	117
Н. Контр.		Кучин А.В.	<i>Кучин А.В.</i>		ФВТ, каф. ВТ		
Утверд.		Павленко Д.В.	<i>Павленко Д.В.</i>		гр. 14ВВ1		

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ	7
ВВЕДЕНИЕ	8
1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ	10
1.1 Сетевая модель передачи данных TCP/IP	10
1.1.1 Схема передачи данных по протоколу TCP/IP	12
1.2 Клиент - серверная архитектура	13
1.2.1 Клиент-серверная архитектура: применение технологии.....	15
1.2.2 Преимущества архитектуры «клиент-сервер»	16
1.3 Характерные черты современных СУБД	18
1.3.1 Реализация для нескольких платформ	19
1.3.2 Административные утилиты.....	20
1.3.3 Резервное копирование данных.....	20
1.3.4 Обслуживание репликаций	21
1.3.5 Параллельная обработка данных в многопроцессорных системах	21
1.3.6 Поддержка доступа к данным с помощью Internet	22
1.4 Особенности работы интернет-магазина	22
1.4.1 Что такое интернет-магазин.....	23
1.4.2 Виды интернет-магазинов.....	24
1.4.3 Виды интернет-магазинов по представляемым товарам в каталоге:.....	27
1.4.4 Специфика интернет-магазина	28
1.4.5 Логика работы интернет-магазина	28
1.5 Движки интернет-магазинов	33
1.5.1 Бесплатные CMS	34
1.5.2 Платные CMS	39
2 РАЗРАБОТКА ПРИЛОЖЕНИЯ	42
2.1 Постановка задачи.....	42
2.2 Структура данных приложений	42
2.3 Выбор архитектуры приложения и способа организации пользовательского интерфейса.....	44
2.4 Описание разработки программы	46
2.5 Отладка и тестирование	50
3 ОПИСАНИЕ ПРОГРАММЫ	52
3.1 Описание клиентского приложения (для сотрудника)	52
3.2 Описание клиентского приложения (для покупателя)	54

3.3 Описание серверного приложения	56
3.4 Руководство пользователя	58
ЗАКЛЮЧЕНИЕ	69
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ	70
ПРИЛОЖЕНИЕ А Листинги программы	71
Приложение А.1 Файл сервера mainwindow.h.....	71
Приложение А.2 Файл сервера main.cpp	73
Приложение А.4 Файл клиента (Сотрудник) mainwindow.h.....	81
Приложение А.5 Файл клиента (Сотрудник) main.cpp	83
Приложение А.6 Файл клиента (Сотрудник) mainwindow.cpp	83
Приложение А.7 Файл клиента (Клиент) mainwindow.h	93
Приложение А.8 Файл клиента (Клиент) main.cpp	95
Приложение А.8 Файл клиента (Клиент) mainwindow.cpp	96
ПРИЛОЖЕНИЕ Б ПРЕЗЕНТАЦИЯ	111

ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ

СУБД - Система управления базами данных

БД - База данных

ИБ – информационная база

ВВЕДЕНИЕ

В настоящее время в оптовых организациях наблюдается стремительный рост количества информации, требующей обработки для принятия управленческих решений, а также решений, удовлетворяющих спрос клиентов. Следовательно, актуальной проблемой является автоматизация процесса учёта заказов клиентов. При этом одной из главных задач становится снабжение лиц, принимающих решения, актуальной информацией о заказах, поэтому задачей информационной системы является сбор и систематизация сведений о заказах.

Интернет-торговля составляет значительную часть рынка, при этом объем внутрироссийских онлайн-продаж материальных товаров продолжает расти из года в год. По данным сайта cs-cart.ru в 2018 году, по прогнозам, прирост составит 18%, объем рынка достигнет 1,115 миллиарда рублей. Для сравнения: рост рынка в 2017 году составил 18%, в 2016 году - 23%.



Рисунок 1 – Рост онлайн-продаж в России

Целью данной выпускной квалификационной работы является разработка информационной системы (ИС) по учёту заказов на примере кондитерского производства. Основными недостатками существующих ИС данного типа являются высокая стоимость разработки. Кроме того, многие известные магазины кондитерских изделий вовсе не имеют собственной ИС по учёту заказов клиентов, например, сеть магазинов кондитерских изделий «Сладкий мир», что в отрицательную сторону влияет на имидж и прибыль бренда.

Бизнес по производству и реализации кондитерских изделий является сегодня особенно актуальным и выгодным. Спрос на такую продукцию сегодня достаточно большой, ведь с одной стороны сладости любят буквально все, а с другой стороны конкуренция в данной отрасли бизнеса сегодня очень слабая. Также стоит учесть то, что продажа продуктов питания входит в десятку самых продаваемых видов товара через интернет, а прирост по объемам онлайн-продаж в 2017 году составил 31%.

Расширение клиентской базы, в том числе привлечение людей с ограниченными способностями к передвижению, делают рассматриваемую ранее разработку ИС достаточно актуальной.

В рамках данной выпускной квалификационной работы необходимо разработать систему для организации интернет продаж, которая должна совмещать в себе невысокую стоимость разработки, интерактивность процесса оформления заказов покупателями, лёгкость администрирования, многофункциональный режим работы, а также удобное взаимодействие пользователей с ИС, и как результат - быстрое оформление заказов.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Сетевая модель передачи данных TCP/IP

Взаимодействие между компьютерами в интернете осуществляется посредством сетевых протоколов, представляющих собой согласованный набор определенных правил, в соответствии с которыми разные устройства передачи данных обмениваются информацией. Существуют протоколы для форматов данных, скорости передачи, для контроля ошибок и другие виды протоколов. В глобальном межсетевом взаимодействии чаще всего используется протокол TCP/IP.

Название протокола интернета TCP/IP произошло от двух сетевых протоколов: TCP и IP. Конечно, этими двумя протоколами построение сетей не ограничивается, но они являются базовыми в том, что касается именно организации передачи данных. Фактически, TCP/IP есть набор протоколов, позволяющих индивидуальным сетям объединяться для образования глобальной сети.

Ниже приведем развернутую характеристику каждому протоколу, входящему в общее понятие TCP/IP.

- Интернет-протокол (IP) отвечает за непосредственную передачу информации в сети. Информация делится на части (другими словами, пакеты) и передается получателю от отправителя. Для точной адресации нужно задать точный адрес или координаты получателя. Такие адреса состоят из четырех байт, которые отделены друг от друга точками. Адрес каждого компьютера уникален. Однако использования одного лишь IP-протокола может быть недостаточно для корректной передачи данных, так как объем большей части пересылаемой информации более 1500 символов, что уже не вписывается в

один пакет, а некоторые пакеты могут быть потеряны в процессе передачи или присланы не в том порядке, что требуется.

- Протокол управления передачей (TCP) используется на более высоком уровне, чем предыдущий. Основываясь на способности IP-протокола переносить информацию от одного узла другому, TCP-протокол позволяет пересылать большие объемы информации. TCP отвечает также за разделение передаваемой информации на отдельные части — пакеты — и правильное восстановление данных из пакетов, полученных после передачи. При этом данный протокол автоматически повторяет передачу пакетов, которые содержат ошибки.

Управление организацией передачи данных в больших объемах может осуществляться с помощью ряда протоколов, имеющих специальное функциональное назначение. В частности, существуют следующие виды протоколов:

1. FTP (File Transfer Protocol) организует перенос файлов и используется для передачи информации между двумя узлами Internet с использованием TCP-соединений в виде бинарного или же простого текстового файла, как поименованной области в памяти компьютера. При этом не имеет никакого значения, где данные узлы расположены и как соединяются между собой.

2. UDP (User Datagram Protocol) Протокол пользовательских дейтаграмм, не зависит от подключений, он передает данные пакетами, которые называют UDP-дейтаграммами. Однако этот протокол не так надежен, как TCP, потому что отправитель не получает данных о том, был ли принят пакет в действительности.

3. ICMP (Internet Control Message Protocol) существует для того, чтобы передавать сообщения об ошибках, возникающих в процессе обмена данными в сети Internet. Однако при этом ICMP-протокол только лишь сообщает об ошибках, но не устраняет причины, которые привели к возникновению этих ошибок.

4. Telnet - сетевой протокол, который используется для реализации текстового интерфейса в сети с помощью транспорта TCP.

5. SMTP (Simple Mail Transfer Protocol) - это специальный протокол обмена электронными сообщениями, определяющий формат сообщений, которые пересылаются с одного компьютера, называемого SMTP-клиентом, на другой компьютер, на котором запущен SMTP-сервер. При этом данная пересылка может быть отложена на некоторое время до тех пор, пока не активируется работа как клиента, так и сервера.

1.1.1 Схема передачи данных по протоколу TCP/IP

1. Протокол TCP разбивает весь объем данных на пакеты и нумерует их, упаковывая в TCP-конверты, что позволяет восстановить порядок получения частей информации. При помещении данных в такой конверт происходит вычисление контрольной суммы, которая записывается потом в TCP-заголовок.

2. Далее посредством протокола IP все пакеты передаются непосредственно получателю.

3. Затем с помощью протокола TCP происходит проверка того, все ли пакеты получены. Если во время приема вычисленная заново контрольная сумма не совпадает с указанной на конверте, это свидетельствует о том, что часть информации была утеряна или искажена при передаче, протокол TCP/IP заново запрашивает пересылку этого пакета. Также требуется подтверждение прихода данных от получателя.

4. После подтверждения получения всех пакетов протокол TCP упорядочивает их соответствующим образом и собирает заново в единое целое. Протоколом TCP используются повторные передачи данных, периоды ожидания (или таймауты), что обеспечивает надежность доставки информации. Пакеты могут передаваться в двух направлениях одновременно. Тем самым

					ПГУ 1.090301.14.001 ПЗ	Лист
						12
Изм.	Лист	№ докум.	Подпись	Дата		

протокол TCP/IP снимает необходимость использования повторных передач и ожиданий для прикладных процессов (таких, как Telnet и FTP).

TCP естественным образом интегрируется в окружение клиент/сервер. Серверное приложение прослушивает поступающие запросы на соединение. Например, службы WWW, пересылки файлов или доступа с терминала, поступающие от клиентов. Коммуникации в TCP запускаются соответствующими подпрограммами, которые и инициализируют соединение с сервером.

Реально клиент может быть другим сервером. Например, почтовые серверы могут соединяться с другими почтовыми серверами для пересылки сообщений электронной почты между компьютерами.

1.2 Клиент - серверная архитектура

Компьютеры, программы и периферийные устройства являются неравноправными составляющими информационной сети. Одни владеют каким-то ресурсом, поэтому называются серверами, другие обращаются к этим ресурсам и называются клиентами. Рассмотрим, как же они взаимодействуют между собой и что собой представляет клиент-серверная архитектура.

					ПГУ 1.090301.14.001 ПЗ	Лист
						13
Изм.	Лист	№ докум.	Подпись	Дата		

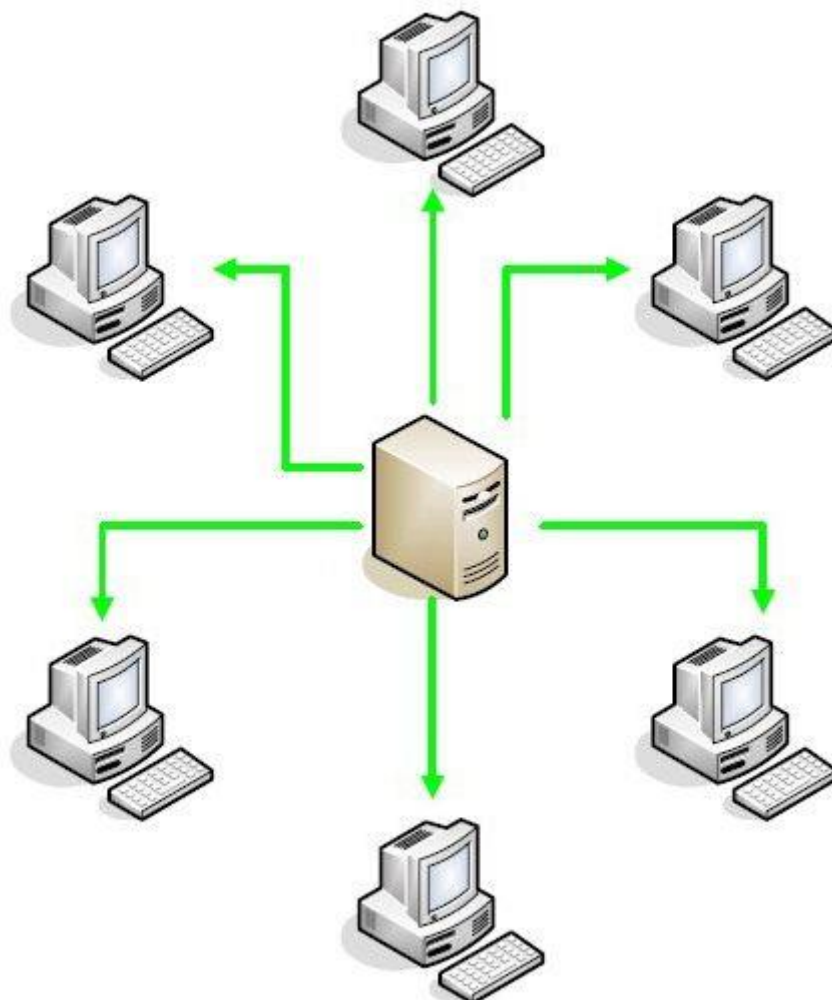


Рисунок 2 - Клиент - Серверная архитектура

Архитектура «Клиент-Сервер» представляет собой взаимодействие структурных компонентов в сети на основе определенных принципов организации данной сети, где структурными компонентами являются сервер и узлы-поставщики определенных специализированных функций (сервисов), а также клиенты, которые пользуются данным сервисом. Специфические функции принято делить на три группы на основе решения определенных задач:

1. функции ввода и представления данных предназначены для взаимодействия пользователя с системой;
2. прикладные функции - для каждой предметной области имеется собственный набор;

3. функции управления ресурсами предназначены для управления файловой системой, различными базами данных и прочими компонентами.

Автономная система, например, компьютер без сетевого подключения, представляет компоненты представления, прикладного назначения и управления на различных уровнях. Такого рода уровнями считаются операционная система, прикладное и служебное программное обеспечение, различные утилиты. Точно так же и в сети представлены все вышеуказанные компоненты. Главное - правильно обеспечить сетевое взаимодействие между этими составляющими.

Клиент-серверная архитектура наиболее часто используется для создания корпоративных баз данных, в которых информация не только хранится, но и периодически поддается обработке различными методами. Зачастую база данных является главным элементом любой корпоративной информационной системы, а на сервере располагается ядро этой базы. Так, на сервере происходят наиболее сложные операции, касающиеся ввода, хранения, обработки и модификации данных. Когда пользователь (клиент) обращается к базе данных (серверу), происходит обработка запроса: непосредственно обращение к базе данных и возврат ответа (результата обработки). Результат обработки - это сообщение сети об успешном проведении операции или ошибке. Серверные компьютеры могут обрабатывать одновременно обращение нескольких клиентов к одному и тому же файлу. Такая работа и передача данных по сети позволяет ускорить работу используемых приложений.

1.2.1 Клиент-серверная архитектура: применение технологии

Данная архитектура используется для доступа к различным ресурсам с использованием сетевых технологий: Web-серверы, серверы приложений, серверы баз данных, почтовые серверы, межсетевые экраны, прокси-серверы.

					ПГУ 1.090301.14.001 ПЗ	Лист
						15
Изм.	Лист	№ докум.	Подпись	Дата		

Разработка клиент-серверных приложений позволяет повысить безопасность, надежность и производительность используемых приложений и сети в целом. Наиболее часто клиент-серверные приложения используются для автоматизации бизнес-процессов.

1.2.2 Преимущества архитектуры «клиент-сервер»

Информационные системы, использующие архитектуру «клиент-сервер», обладают серьезными преимуществами по сравнению с их аналогами, созданными на основе сетевых версий настольных СУБД. Ниже будут рассмотрены наиболее важные из них.

Одним из важнейших преимуществ архитектуры «клиент-сервер» является снижение сетевого трафика при выполнении запросов. Например, при необходимости выбора пяти записей из таблицы, содержащей миллион записей, клиентское приложение посылает серверу запрос, который сервером анализируется на корректность и, если запрос корректен, компилируется, оптимизируется и выполняется. После этого результат запроса (те самые пять записей, а вовсе не вся таблица) передается обратно клиенту. При этом, формулируя запрос, можно не задумываться о том, есть ли в базе данных индексы, способные облегчить поиск нужных записей, — если они есть, то они будут использованы сервером, а если нет, запрос все равно будет выполнен, хотя, возможно, это займет больше времени, чем при использовании индексов. Но в любом случае - есть индексы или нет - в клиентское приложение передается только результат запроса, и в этом случае сетевой трафик не зависит ни от их наличия, ни от числа записей в таблицах, к которым произведен запрос.

Вторым преимуществом архитектуры «клиент-сервер» является возможность хранения бизнес-правил (например, правил ссылочной целостности или ограничений на значения данных) на сервере, что позволяет избежать дублирования кода в различных клиентских приложениях,

использующих общую базу данных. Кроме того, в этом случае любое редактирование данных, в том числе и редактирование средствами, не предусмотренными разработчиками информационной системы (например, различными утилитами администрирования сервера), может быть произведено только с учетом этих правил. Если последние версии некоторых настольных СУБД и способны хранить ограничения на значения данных либо тексты SQL-запросов, триггеры и хранимые процедуры в них, как правило, отсутствуют - их просто некому выполнять. Исключением здесь является только Microsoft Data Engine, но отнести к настольным эту СУБД можно весьма условно - фактически MSDE представляет собой локальный сервер баз данных, обладающий всеми характерными для серверной СУБД атрибутами, включая отдельный серверный процесс.

Для описания серверных бизнес-правил в наиболее типичных ситуациях (например, при реализации связей master/detail) нередко используются так называемые CASE-средства (CASE означает Computer-Aided System Engineering) для создания диаграмм «сущность-связь». Эти инструменты позволяют описать подобные правила на уровне логической и физической моделей данных без какого бы то ни было программирования, а затем сгенерировать код соответствующих серверных объектов - триггеров, хранимых процедур, серверных ограничений. В этом случае клиентские приложения будут избавлены от значительной части кода, связанного с реализацией бизнес-правил непосредственно в приложении. Стоит отметить, что часть кода, связанного с обработкой данных, также может быть реализована в виде хранимых процедур сервера, что позволяет еще больше «облегчить» клиентское приложение, а это означает, что требования к рабочим станциям могут быть не столь высоки. Это в конечном итоге снижает стоимость информационной системы даже при использовании дорогостоящих серверной СУБД и аппаратного обеспечения.

Помимо перечисленных выше трех преимуществ следует, что современные серверные СУБД обладают широкими возможностями управления пользовательскими привилегиями и правами доступа к различным объектам базы данных. Как правило, в базе данных хранятся сведения о ее пользователях, их паролях и привилегиях, а каждый объект базы данных принадлежит какому-либо пользователю. Владелец объекта может предоставить другим пользователям право использовать его тем или иным способом (например, позволить читать из него данные какому-либо другому пользователю). Большинство серверных СУБД поддерживает так называемые роли, представляющие собой совокупность прав на доступ к тем или иным объектам базы данных. В этом случае каждый пользователь может иметь одну или несколько ролей и соответственно определенные в этих ролях привилегии.

Современные серверные СУБД обладают также широкими возможностями резервного копирования и архивации данных, а нередко и оптимизации выполнения запросов. Они также, как правило, предоставляют возможность параллельной обработки данных, особенно в случае использования многопроцессорных компьютеров в качестве сервера баз данных.

1.3 Характерные черты современных СУБД

Выше были расписаны преимущества архитектуры «клиент-сервер», обусловленные ее базовыми принципами и не зависящие от того, какая конкретно СУБД выбрана. Также современные серверные СУБД обычно предоставляют дополнительный набор сервисов, связанных с обслуживанием хранения и обработки данных, созданием клиентских приложений, сменой СУБД или ее версии, обслуживанием нескольких баз данных, публикацией данных в Internet. Поскольку в условиях конкуренции между различными производителями СУБД каждый из них стремится к завоеванию как можно большей части рынка. Это приводит к тому, что все они пытаются перегнать

друг друга и предоставить потенциальному потребителю (а также производителям средств разработки, средств проектирования данных и генераторов отчетов) как можно больше сервисов различного назначения, так как при выборе СУБД потребитель обычно ориентируется на то, что поддерживает данная СУБД, с одной стороны, и чем она поддерживается - с другой. И, как и в случае настольных СУБД, в этой области также происходит процесс заимствования идей, решений и интерфейсов.

Ниже перечислены наиболее характерные для сегодняшнего дня сервисы, предоставляемые серверными СУБД.

1.3.1 Реализация для нескольких платформ

Почти все современные серверы баз данных существуют в нескольких версиях для различных платформ (как правило, различные коммерческие версии UNIX — Solaris, HP/UX и др., а также Windows NT Server и, с недавнего времени, Windows 2000). Многие производители серверных СУБД также выпускают версии своих серверов для Windows NT Workstation и Windows 95/98 (а иногда даже для Windows CE). В последнем случае такие серверы нередко бывают персональными (однопользовательскими), либо в их реализации отсутствуют возможности, характерные для полнофункциональных версий этих серверов. Подобные персональные серверы, как правило, используются в «переносных» системах, например на ноутбуках, применяемых для работы с базой данных отдельно от центрального офиса компании с последующей репликацией данных. Иногда такие серверы используются при создании приложений для изоляции разработчика от сервера баз данных, находящегося в процессе эксплуатации.

В последнее время многие производители серверных СУБД выпускают также версии для Linux - с этой точки зрения Linux в последние два года была весьма «модной» платформой.

Исключением из этого правила является Microsoft SQL Server. Однако для данной СУБД это вполне оправданно - компания Microsoft сама производит серверные операционные системы и в отличие от большинства других производителей серверных СУБД может себе позволить создавать серверы баз данных, тесно интегрированные с сервисами операционной системы собственного производства и поддерживающие исключительно их.

1.3.2 Административные утилиты

Наличие удобных утилит администрирования, иногда оказывается одним из решающих факторов при выборе СУБД. Именно поэтому подавляющее большинство современных СУБД обычно поставляется с подобными утилитами, и их интерфейс в последнее время напоминает интерфейс Windows Explorer.

Следует отметить, что производители СУБД, вовремя не заметившие эту тенденцию, оказались лишенными значительной части рынка или практически вытесненными с него. Весьма показателен здесь пример IB Database (InterBase Corporation) - производители этой СУБД, очень неплохой с точки зрения хранения данных и скорости обработки запросов, не позаботились вовремя об удобных утилитах администрирования (а также о средствах репликации данных), положившись на продукты третьих фирм и на продвижение этой СУБД в составе популярных средств разработки, - и в результате многие ее потенциальные пользователи сделали выбор в пользу других СУБД, а сама IB Database практически перешла в разряд некоммерческих продуктов с доступными исходными текстами.

1.3.3 Резервное копирование данных

Резервное копирование данных и журналов транзакций поддерживается всеми без исключения коммерческими серверными СУБД. Различия между

СУБД в поддержке резервного копирования заключаются в том, возможно ли производить резервное копирование в процессе работы пользователей, и если да, то какие пользовательские операции в это время нельзя выполнять. Помимо этого, в комплекте поставки некоторых СУБД могут содержаться утилиты для использования различных внешних устройств в качестве средств хранения резервных копий.

1.3.4 Обслуживание репликаций

Репликация, по существу, представляет собой гарантированное копирование информации из одной базы в несколько других. Репликации используются для разделения нагрузки между серверами в сети, для перемещения поднаборов данных на вспомогательные серверы, для синхронизации данных на нескольких серверах и многих других целей.

В том или ином виде репликации поддерживаются всеми современными серверными СУБД. Различия могут быть лишь в поддержке тех или иных конкретных сценариев репликаций (например, внесение изменений одновременно на нескольких серверах, возможность шифрования реплицируемых данных и др.). Исключение здесь составляет IB Database, в текущей версии не поддерживающая репликаций.

1.3.5 Параллельная обработка данных в многопроцессорных системах

О возможностях повышения производительности с помощью параллельной обработки запросов в многопроцессорных системах начали говорить несколько лет назад, после появления первого продукта такого класса - Oracle Parallel Server. Серверы, поддерживающие параллельную обработку, разрешают нескольким процессорам обращаться к одной базе данных, что позволяет обеспечить высокую скорость обработки транзакций.

					ПГУ 1.090301.14.001 ПЗ	Лист
						22
Изм.	Лист	№ докум.	Подпись	Дата		

В настоящее время подавляющее большинство производителей современных серверных СУБД поставляют на рынок версии, поддерживающие параллельную обработку данных. Обычно это версии для Windows NT и коммерческих версий UNIX.

1.3.6 Поддержка доступа к данным с помощью Internet

Без поддержки публикации данных в Internet или получения данных от удаленных Internet-клиентов сегодня не обходится практически ни одна коммерческая СУБД, в том числе настольные базы данных. Тем или иным способом производители серверных СУБД поддерживают Web-технологии. Чаще всего эта поддержка осуществляется с помощью Web-серверов собственного производства, либо посредством создания расширений для существующих Web-серверов, либо просто путем включения в комплект поставки утилит, генерирующих Web-страницы согласно определенному расписанию.

Исходя из всего вышесказанного можно сделать вывод о том, что клиент-серверная архитектура и база данных для хранения информации отлично подходит для реализации интернет-магазина. Ведь системы электронной коммерции должны обладать способностью координировать бизнес-транзакции в многочисленных бизнес-приложениях, уметь извлекать отдельные части информации из различных источников, своевременно и беспрепятственно доставлять клиенту необходимую информацию, - все по единственному Web-запросу пользователя.

1.4 Особенности работы интернет-магазина

Электронная коммерция (англ. e-commerce, e-comm,) стала неотъемлемой частью нашего общества. По мере интенсивного роста числа пользователей сети Интернет она становится все более мощным инструментом бизнеса,

средством реализации товаров и услуг. Интернет стал основным источником информации для огромного числа потенциальных покупателей.

Электронная коммерция - сфера мировой экономики, включающая в себя финансовые и торговые транзакции, которые проводятся при помощи компьютерных сетей, а также все виды бизнес-процессов, связанные с проведением таких транзакций.

Электронная коммерция предоставляет огромные возможности для развития своего дела. Однако для того, чтобы использовать ее действительно эффективно мало быть талантливым предпринимателем, точно так же, как и мало уметь ориентироваться в безграничном виртуальном пространстве. Необходимо еще обладать рядом высокотехнологичных инструментов для ведения электронного бизнеса. Одним из них и является интернет-магазин.

1.4.1 Что такое интернет-магазин

Интернет-магазин является одной из разновидностей электронной коммерции. Продажа собственных товаров и услуг через интернет - основной способ коммерческого использования сайтов для большинства производственных и торговых предприятий. Компания создает сайт и размещает на нем информацию о своих товарах и услугах, ценах и гарантиях для покупателей. Интернет-магазины становятся одним из необходимых и эффективных инструментов увеличения числа продаж предприятия, увеличения оборота и повышения имиджа, успешного развития. Существует несколько определений «что такое интернет-магазин».

Интернет-магазин (англ. online shop или e-shop) - специализированный вид бизнеса, основанный на дистанционном обслуживании и дающий возможность продавцу и покупателю совершать сделки по покупке-продаже того или иного вида товаров посредством сети Интернет.

Интернет-магазин - это интерактивный веб-сайт, рекламирующий товар или услугу, принимающий заказы на покупку, предлагающий пользователю выбор варианта расчета, способа получения заказа и выписывающий счет на оплату.

Интернет-магазин - это совокупность программ, работающих на Web-сайте, которые позволяют покупателю дистанционно выбрать товар из каталога и оформить его заказ.

Интернет-магазин является посредником между покупателем и складом производителя, которому достаточно иметь ресурс в Интернете и телефон. Магазин виртуального типа не нуждается в представительном офисе, складских помещениях, не несет расходы на содержание помещений, владелец магазина может управлять продажами даже из своей квартиры. Именно потому цены в таких магазинах минимальны. Купленные вещи подвозят потребителю непосредственно со склада производителя.

1.4.2 Виды интернет-магазинов

Интернет-магазины можно классифицировать по следующим видам:

- по модели бизнеса;
- по объемам продаж;
- по видам продаж;
- по способам получения дохода;
- по товарному ассортименту;
- по отношению с поставщиками;
- по представляемым товарам в каталоге;

По модели бизнеса:

1. Онлайн магазин без действующего физического магазина.

2. Совмещение оффлайн-бизнеса с онлайн-бизнесом (когда интернет-магазин был создан на основе уже действующей реальной торговой структуры). Этот вид интернет-магазина характеризуется отличительными чертами классического бизнеса: склад, купленные партии товара, сотрудники, бухгалтер, юрист и т. д.
3. На аутсорсинге. При этом виде интернет-магазина вы сами непосредственно не занимаетесь приемом, доставкой, хранением и комплектацией заказов, а передаете на аутсорс сторонней компании, решая лишь организационные бизнес-вопросы. Такая модель освобождает массу времени и в то же время позволяет получать довольно значимую прибыль (около 70% от фактической выручки интернет-магазина).
4. Продажа по системе дропшипинг (когда товара нет, а сайт магазина «продает» продукцию поставщиков, отправляя заказчику готовые заказы или покупая у него от имени своего покупателя).

По объемам продаж:

- розничная продажа (продажи на одного клиента идут в небольших объемах, зачастую по одной единице товара);
- оптовая продажа (продажи идут в больших объемах, зачастую клиент является постоянным, что говорит о систематичности покупок).

По видам продаж:

- B2B (би-ту-би или «бизнес для бизнеса») - продажа или оказание услуг иным коммерческим предприятиям;
- B2C (би-ту-си или «бизнес для клиента») - продажа или оказание услуг конечному потребителю.

По способам получения дохода:

1. Продажа через сайт товаров и услуг от производителя или официального представителя. Продажа собственных товаров и услуг через Интернет - основной способ коммерческого использования сайтов для большинства производственных и торговых предприятий. Компания создает сайт и размещает на нем информацию о своих товарах и услугах, ценах и гарантиях для покупателей.
2. Продажа товаров и услуг по партнерской программе (аффилиатские сети). Основная идея партнерских программ - это участие в чужих продажах. Суть подобных программ состоит в том, что продавец (владелец «партнерки») согласен разделить часть дохода от продажи товаров или услуг с владельцем того сайта (партнером, или аффилиатом), который прислал ему покупателя.
3. Продажа информации (контента). Для продажи информации уже нет необходимости в затратах на бумагу, тиражи, точки продаж. Любая веб-страница может быть доступна миллионам пользователей из всех стран мира. Основной способ реализации этого вида интернет-магазина - платные электронные рассылки и сайты с платным доступом. Оплата за информацию происходит таким же образом - в онлайне, как и при оплате материальных товаров в стандартном интернет-магазине.

Виды интернет-магазинов по товарному ассортименту: книги, аудио и видеокассеты, CD, DVD, компьютерная и бытовая техника, мобильные телефоны, мебель, одежда, обувь, часы и другие товары.

По отношению с поставщиками:

- имеют собственный склад (наличие реальных товарных запасов);
- работают по договорам с поставщиками (отсутствие значительных собственных запасов).

1.4.3 Виды интернет-магазинов по представляемым товарам в каталоге:

- интернет-витрины;
- интернет-магазины;
- онлайн-аукционы.

Каждый из видов интернет-магазина включает в себя несколько групп, достаточно чётко определяющих смысл существования магазина и его аудиторию.

Интернет-витрина - скорее это рекламный сервер. На витрине выкладывают информацию о товарах, которую постоянно обновляют. Затраты на ее создание и администрирование могут быть довольно низкими, а практическая польза такой витрины очевидна. Но это еще не торговля. Потенциальный покупатель, посетив витрину, должен позвонить на фирму, оплатить товар, договориться о доставке. Поэтому интернет-витрина оправдана в тех случаях, когда покупателя надо познакомить со сложной продукцией, на изучение которой в торговом зале у него уйдет слишком много времени.

Интернет-витрина может быть размещена где угодно - на собственном сервере, на сервере провайдера, на сервере, предоставляющем бесплатные страницы. Для работы с витриной достаточно иметь подключение через телефонную линию и минимум навыков работы с HTML.

Каталог интернет-магазина будет содержать многоуровневое логическое дерево разделов и подразделов, внутри которых будут находиться списки с карточками товаров. Он должен быть хорошо структурированным и очень логичным, чтобы в разнообразии товаров посетители нашли то, что им действительно нужно. Конечной точкой перемещения по каталогу должна стать карточка товара, содержащая его описание, признак наличия в продаже и цену.

Список в каталоге сортируется по признаку наличия товара на складе. Актуальные товары находятся в начале списка, снятые с продажи - в конце.

Онлайн-аукцион, или интернет-аукцион - это аукцион, проводящийся посредством сети Интернет. Он является видом интернет-магазина, так как имеет в своем функционале прием онлайн-платежей. Момент окончания интернет-аукциона заранее назначается самим продавцом при постановке товара на торги. По окончании интернет-аукциона покупатель должен перевести деньги продавцу по безналичному расчёту (реже - наличными, например - при получении товара лично), а продавец обязан выслать товар покупателю по почте, нередко в любую точку страны проведения или всего мира. Границы возможной пересылки товара указываются самим продавцом заранее.

1.4.4 Специфика интернет-магазина

Главная особенность интернет-магазина - большая часть взаимодействия продавец-покупатель происходит в онлайн. Многие этапы покупки и продажи довольно успешно проходят дистанционно. Это является большим преимуществом. Но в тоже время, именно в этом и заключается одна из трудностей, которую надо преодолеть любому интернет-магазину - внушить доверие посетителю и расположить его к совершению покупки.

1.4.5 Логика работы интернет-магазина

Посетитель заходит на сайт и набирает корзину товаров, после чего происходит оформление заказа. Если посетитель уже покупал товары на этом сайте, достаточно ввести логин, пароль и уточнить адрес доставки. Если не покупал - необходимо будет дополнительно ввести контактные данные. Затем выбирается способ доставки и формируется конечная стоимость заказа (товары + доставка).

					ПГУ 1.090301.14.001 ПЗ	Лист
						28
Изм.	Лист	№ докум.	Подпись	Дата		

Следующим этапом является выбор формы оплаты. Как правило, происходит переход на сайт платёжной системы, где пользователь авторизуется и оплачивает покупку.

После оплаты посетитель возвращается на сайт и получает подтверждение - прошла оплата или нет. Если нет - обычно предлагается повторить попытку или связаться с менеджером интернет-магазина для получения помощи.

В зависимости от выбранного способа доставки в срок, указанный на сайте или сообщенный менеджером, приходит заказ — на почту или по адресу. В случае доставки курьером иногда можно указать удобные дни и время доставки.



Рисунок 3 - Схема интернет-магазина

Способы оплаты покупок в интернет-магазине:

1. Банковская карта - безналичный вид расчёта, который часто подвергается нареканиям в плане безопасности.

2. Банковский перевод - оплата заказа производится банковским платежным поручением на расчётный счет магазина.
3. Электронные деньги - безналичный вид расчёта.
4. Терминалы моментальной оплаты - оплата производится в уличных платёжных терминалах.
5. SMS-платежи. Интернет-магазин формирует запрос средств, передает его в одну из специализированных систем электронных платежей и отправляет туда пользователя. Пользователь уже во взаимодействии с платежной системой отправляет SMS с указанными системой реквизитами и подтверждает платеж. Платежная система после подтверждения отправляет магазину уведомление об успешности перевода или наоборот.
6. Наложённый платеж - товар оплачивается на почте при получении.
7. Наличный расчет - товар оплачивается покупателем курьеру наличными деньгами при получении товара.
8. Электронные кассы - вид расчета, объединяющий практически все перечисленные выше способы оплаты. При выборе такого способа оплаты пользователю предлагается на выбор наиболее удобный способ перевода денег от пластиковой карточки до терминала и мобильного телефона.

Преимущества и риски

Интернет-магазин обладает рядом преимуществ по сравнению с обычными магазинами:

1. Интернет-магазин работает 24 часа в сутки, 365 дней в году, без перерыва на обед, без выходных и праздничных дней.
2. Необязательно иметь в наличии товар. Но в этом случае обязательно иметь сеть поставщиков, работающих по принципу «точно вовремя».

Это гарантирует доставку товара в заранее оговоренные минимальные сроки.

3. Не нужно помещение для продаж. Витрины, стеллажи, прилавки, квадратные метры и арендная плата - на всем этом можно существенно сэкономить. На чем сэкономить нельзя - так это на качественно выполненном дизайне сайта для интернет-магазина, поскольку именно внешний вид веб-страниц будет заменять покупателям и стеллажи, и витрины.
4. Срок и стоимость создания интернет-магазина несоизмеримо ниже, чем обычного магазина.
5. Свобода передвижений продавца. Поскольку интернет-магазин - это точка продаж в виртуальном пространстве, вы можете управлять им из любого уголка земного шара, где есть доступ в Интернет.
6. С интернет-магазином вы получаете возможность расширить географию своего бизнеса вплоть до мировых рынков. С оговоркой, что содержание сайта будет понятно даже иностранцам.
7. Можно не нанимать продавцов, администраторов, менеджеров, товароведов, кассиров, охранников и еще кого бы то ни было. Профессионально грамотно созданный интернет-магазин может работать полностью автономно. Практически без обслуживания.
8. Интернет-магазин дает вам возможность самим распределять свое рабочее время. Вы можете тратить на его обслуживание столько часов в сутки, сколько захотите - один, два, а то и все двадцать четыре. И когда хотите - ночью, днем, утром, вечером.
9. Для создания интернет-магазина не требуется получения многочисленных разрешений и лицензий. Его не проверяет пожарный инспектор, санэпидемстанция и другие службы.

Однако помимо преимуществ ведения торговли через интернет-магазин есть и существенные риски:

1. Атака хакеров. В «лучшем» - внесут путаницу в данные о заказах, отправлениях, расчетах, и Вам будет необходимо объясняться со всеми своими клиентами на предмет «Извините, это наша ошибка». В худшем случае у Вас могут украсть базу данных с номерами кредитных карт клиентов, а Вы погрязнете в судебных разбирательствах. Однако в Интернете есть средства, обеспечивающие безопасность электронной торговли, поэтому имеется возможность интегрировать в интернет-магазин новейшие защитные разработки.
2. «Баги». Это ошибки в программном обеспечении, сделанные самими разработчиками. Ошибки есть всегда. В любом программном обеспечении для Интернет-торговли. Это может привести к тому, что в определенный момент, при определенных условиях Ваш магазин начинает «глючить» или вообще «откажется» работать. Проблему с багами помогает решить четко функционирующая служба технической поддержки при компании-разработчике. Техническая поддержка - это гарантия того, что в случае обнаружения ошибки к Вам на помощь придут высококвалифицированные специалисты. Если таковой не имеется - подумайте семь раз прежде, чем приобретать интернет-магазин у этого производителя.
3. Покупка «кота в мешке». Покупая интернет-магазин, вы никогда сразу не поймете владельцем какого «добра» отныне Вы являетесь. Поэтому насторожитесь, если компания-разработчик отказывается предложить вам совершенно бесплатно в течение некоторого времени попользоваться своим продуктом. Или, в крайнем случае, не гарантирует возврата Ваших денег - опять же в течение определенного времени. Бесплатное тестовое использование и гарантия возврата

денег - это не рекламный ход, это возможность «примерить» программное обеспечение на свой бизнес.

4. Клиенты легко приходят и уходят. К счастью, в интернет-магазин не надо «идти» - важно знать адрес. Но, к сожалению, чтобы его покинуть, также достаточно одного щелчка «мышью». Все, чем Вы можете привлечь клиентов на свой сайт - это оригинальный дизайн, выгодные предложения, хорошие тексты-описания товаров. Никаких эмоций, никакого общения, никаких старых знакомых клиентов. И никакой быстрой реакции на ситуацию. Однако и эта проблема в последнее время становится разрешимой. Появляется такое программное обеспечение, которое позволяет буквально отслеживать посетителей вашего сайта - по полу, возрасту, профессии, месту жительства и т.п., а также добавляется интерактивность: вы можете с ними общаться! Профессиональные разработчики интернет-магазинов предусматривают возможность встраивать в свои продукты подобные программы либо производят их сами.

Тем не менее, интернет-магазины завоевывают всё больше доверия и популярность, и, несомненно, их ожидает большое будущее.

1.5 Движки интернет-магазинов

Невозможно себе представить полноценный интернет-магазин без CMS - системы управления контентом. Процесс добавления, редактирования и удаления товаров происходит постоянно, поэтому функциональная админ-панель просто необходима.

Все системы управления делятся на:

- бесплатные;
- платные.

1.5.1 Бесплатные CMS

Их популярность вызвана не только бесплатностью, но и хорошей функциональностью: на них создают достаточно качественные и профессиональные сайты. Далее они будут рассмотрены подробнее.

WordPress

Эта система сама по себе не является платформой для магазина. Однако установка специальных плагинов решает проблему. На сегодняшний день самыми известными расширениями, позволяющими превратить WordPress в интернет-магазин, являются WooCommerce, WP Shop и WP e-commerce.

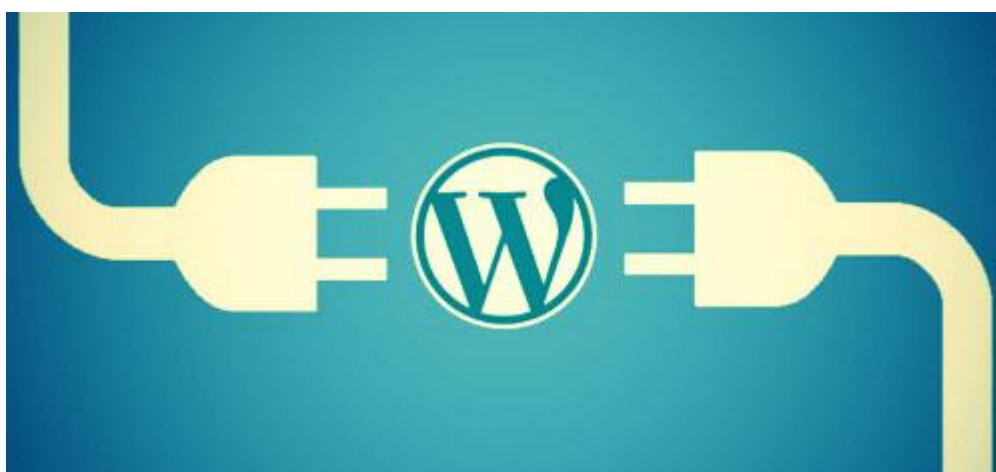


Рисунок 4 – WordPress

Плюсы:

- WordPress очень прост в освоении: можно быстро научиться им управлять. А если есть опыт работы, то настроить магазин не составит труда.
- Гибкость в настройках - ещё основное достоинство WordPress. С помощью дополнительных плагинов или изменения кода можно делать с сайтом всё что угодно.
- Движок постоянно обновляется.

Joomla

Доля Joomla в российском Интернете меньше. Однако многие с успехом её используют.

Чтобы сайт на Joomla превратился в интернет-магазин, необходимо установить специальное дополнение - VirtueMart. Благодаря ему можно создать неограниченное число категорий, добавлять сколько угодно товаров, а также реализовать Корзину.



Рисунок 5 – VirtueMart

Плюсы:

- Простота настройки и управления.
- Идеально подойдёт для небольших проектов (например, продажа цифровых товаров).

Минусы:

- Слабый уровень безопасности. По статистике, сайты на Joomla и WordPress взламываются чаще.
- Технические настройки. Мало просто установить VirtueMart - его ещё надо адаптировать к установленному шаблону. Редко случается так, что всё работает с первого раза.

Opencart

Этот движок специально создан для интернет-магазинов. В нём заложен достаточно большой функционал: корзина, различные способы доставки и оплаты, партнёрская программа, подарочные сертификаты, товары со скидками, неограниченное число товаров и рубрик.

Плюсы:

- Постоянные обновления. Периодически в открытом доступе появляются новые версии движка. Через несколько дней после выхода оригинального Opencart в Рунете появляется переведённая на русский язык сборка. Несмотря на то, что движок иностранный, проблем с переводом нет.
- Модули. За всё время существования Opencart было создано огромное число модулей, значительно расширяющих функционал. Модули дорабатываются, совершенствуются и адаптируются к каждой новой версии системы.
- Поддержка. Существует большое число форумов и сообществ, посвящённых этой CMS. Там можно задать любой интересующий вопрос и получить поддержку.
- Шаблоны. Для Opencart существует большое количество шаблонов, отличающихся своей оригинальностью. Самые последние из них адаптированы к мобильным устройствам.
- Безопасность. Эта CMS считается достаточно защищённой от взлома.
- Применимость к большинству проектов. Opencart подойдёт практически ко всем видам товаров, которые можно продавать в интернете.



Рисунок 6 – Opencart

Минусы:

Технические сложности. Для полноценной настройки движка под свои нужды придётся исправлять код. Это требует некоторых навыков или привлечения программиста.

PrestaShop

Ещё одна платформа, созданная для проектов, ориентированных на коммерцию. Используется реже, чем Opencart, и отсюда вытекают её главные минусы:

- Меньше шаблонов.
- Меньше модулей.
- Меньше специалистов, работающих с этой CMS.

Тем не менее, это рабочая система, на основе которой можно сделать качественный проект.

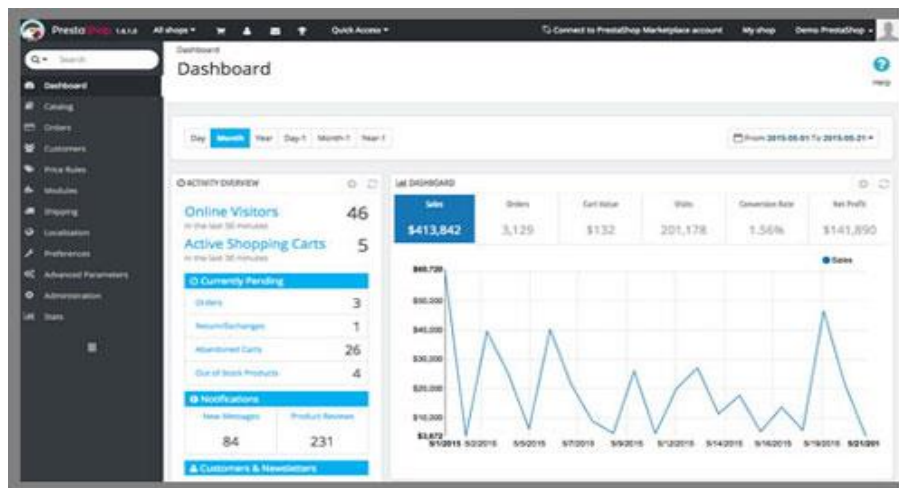


Рисунок 7 – PrestaShop

osCommerce

Несколько лет назад osCommerce был очень распространён, так как позволял отлично реализовывать проекты интернет-магазинов.

Плюсы:

- Существующее сообщество пользователей и разработчиков.
- Документация + русский язык.
- Широкий функционал.

Минусы:

Система устарела. Последнее обновление датируется июнем 2014 года. Это очень давно, учитывая постоянное развитие других CMS и возрастающие потребности пользователей. Сейчас можно использовать и более современные движки.



Рисунок 8 – osCommerce

Magento

Имеет как платные, так и бесплатные версии. Система сложна и требовательна к ресурсам, но при этом имеет очень много возможностей. Одна из ключевых – возможность управлять разными интернет-магазинами в единой админ-панели.

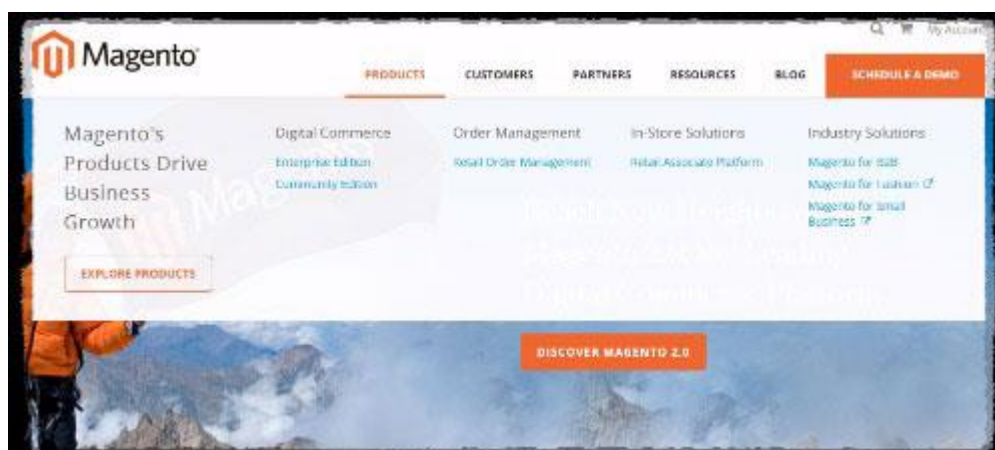


Рисунок 9 – Magento

1.5.2 Платные CMS

Bitrix

Это один из самых мощных и лучших движков, используемых для создания онлайн-магазинов. Стоимость редакции «Малый бизнес», включающей в себя функцию интернет-магазина, составляет 35900 рублей.

Электронные версии продукта	
Лицензия на «1С-Битрикс: Управление сайтом»	Цена, руб.
Лицензия на «1С-Битрикс: Управление сайтом - Старт»	5 400 руб. 
Лицензия на «1С-Битрикс: Управление сайтом - Стандарт»	15 900 руб. 
Лицензия на «1С-Битрикс: Управление сайтом - Малый Бизнес»	35 900 руб. 
Лицензия на «1С-Битрикс: Управление сайтом - Эксперт»	52 900 руб. 
Лицензия на «1С-Битрикс: Управление сайтом - Бизнес»	72 900 руб. 

Рисунок 10 - Bitrix

Плюсы:

- Техподдержка от разработчиков.
- Большое число готовых модулей.
- Безопасность на высоком уровне.

Минусы:

- Работать с Bitrix непросто. Настройка сайта и его доработка потребуют вмешательства программиста.
- Требовательность к ресурсам. Для размещения интернет-магазина на Bitrix потребуется производительный хостинг или выделенный сервер.
- Неудобство администрирования. Тем, кто никогда не работал с Bitrix, будет трудно освоить интерфейс админ-панели.

Moguta CMS

Молодой проект, созданный отечественными разработчиками. Имеет бесплатную версию, у которой достаточно небольшой набора функций. Платная версия стоит от 11 990 до 15 990 рублей.

 Moguta.CMS <small>основа вашего интернет-магазина</small>	Moguta.CMS v.6.2.2 Free Скачать бесплатно	Moguta.CMS v.6.2.3 Standart Купить сейчас за 11990 руб.	Акция! Standart +Модули Купить сейчас за 15990 руб.
ГОТОВЫЙ ИНТЕРНЕТ-МАГАЗИН		Тестовая версия	
каталог товаров	✓	✓	✓
корзина товаров	✓	✓	✓
адаптивный дизайн	✓	✓	✓
скидки и наценки	✗	✓	✓
центр обновлений	✗	✓	✓
прием оплаты по картам Visa	✗	✓	✓
дополнительные и объемные скидки (+700 руб.)	✗	✗	✓
отзывы о товарах с Яндекс.Маркет (+490 руб.)	✗	✗	✓
Посмотреть все возможности			

Рисунок 11 - Moguta CMS

Плюсы:

- Это полностью русскоязычная система.
- Имеет дружелюбную админ-панель.
- Имеет множество готовых модулей в комплекте.
- Бесплатно даётся 14 дней на тестирование.

Главный минус - Moguta не так широко распространена, как остальные системы, поэтому имеет ограниченное число дополнений. А это накладывает определённые ограничения на функционал. Если Moguta CMS будет развиваться и дальше, то ситуация, возможно, поменяется.

2 РАЗРАБОТКА ПРИЛОЖЕНИЯ

2.1 Постановка задачи

Необходимо разработать клиентское и серверное приложения, работающие по протоколу TCP/UDP. Серверное приложение должно работать с базой данных. Доступ к базе данных должен осуществляться в двух видах: открытый на просмотр и просмотр/редактирование. Полномочия зависят от клиентского приложения.

Многомодульность программы. Необходимо разбить программу на модули. Это повысит наглядность и легкость восприятия кода программы, а также облегчит поиск и исправление возможных ошибок.

Режим работы видеосистемы-графический.

Устройства ввода-клавиатура и мышь.

Пользовательский интерфейс должен быть построен на основе меню и панели инструментов. Необходимо определиться с выбором доступного и интуитивно понятного меню для возможности эффективно работать с программой.

Операционная система - MS Windows

Среда программирования - Qt

Язык программирования - C++

2.2 Структура данных приложений

Для реализации приложения был выбран объектно-ориентированный язык программирования C++, который поддерживается средой разработки Qt.

Взаимодействие с пользователем построено на использовании кнопок, выпадающих списков и текстовых полей для ввода.

					ПГУ 1.090301.14.001 ПЗ	Лист
						42
Изм.	Лист	№ докум.	Подпись	Дата		

На рисунках 12 и 13 показаны UML-диаграммы вариантов использования клиентских программ

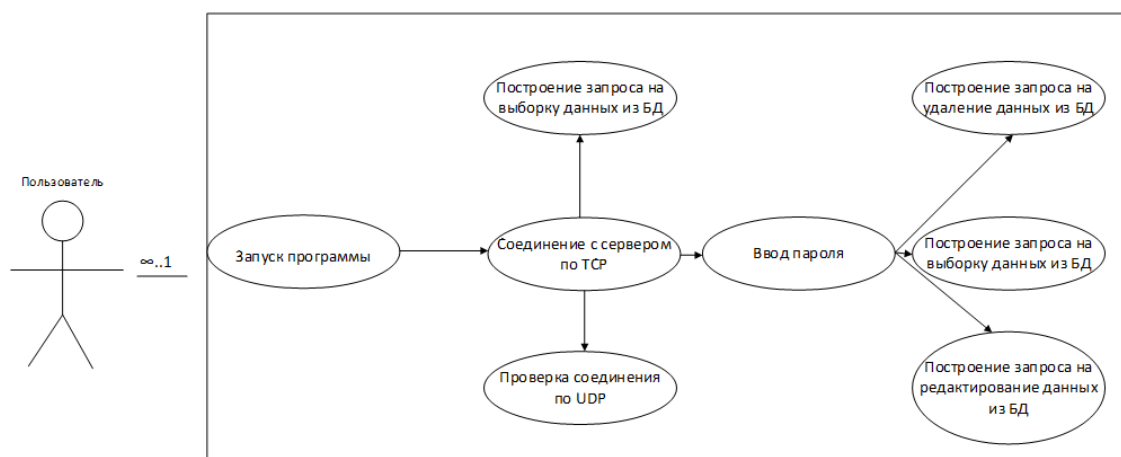


Рисунок 12 - UML-диаграмма вариантов использования клиентской программы (для сотрудника)



Рисунок 13 - UML-диаграмма вариантов использования клиентской программы (для покупателя)

Программа-клиент (для сотрудника) имеет возможность выбора портов и ip адреса, куда будут посылаются запросы на данные по протоколу TCP и для проверки соединения по протоколу UDP. После успешного соединения с сервером есть возможность выбирать таблицы и столбцы, данные которых необходимо получить, а также редактировать их.

Программа-клиент (для покупателя), после успешного соединения с сервером, позволяет просматривать список товаров, создавать и отправлять заказы.

На рисунке 14 показана UML-диаграмма вариантов использования серверной программы.

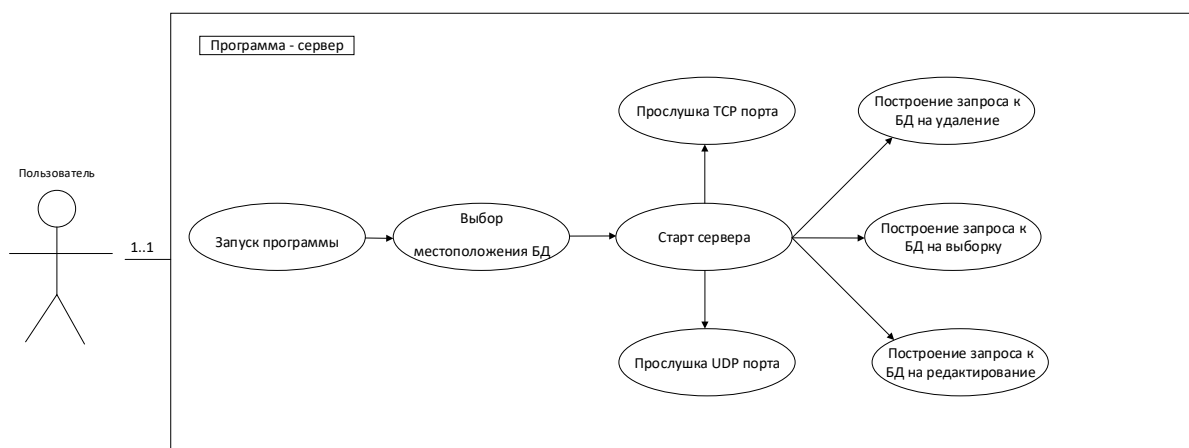


Рисунок 14 - UML-диаграмма вариантов использования серверной программы

Программа-сервер имеет возможность выбора портов для прослушки. После выбора БД и нажатия на кнопку “Начать работу” сервер будет обрабатывать запросы в автоматическом режиме.

2.3 Выбор архитектуры приложения и способа организации пользовательского интерфейса

Программа-клиент и сервер состоит из одного основного окна, которое создается при запуске программы. Окно содержит в себе элементы пользовательского интерфейса: кнопки, всплывающие меню, текстовые поля. Основные функции программы помещены в отдельные группы, что делает интерфейс интуитивно понятным.

На рисунке 15 изображена UML-диаграмма состояний программы-клиента (для сотрудника).

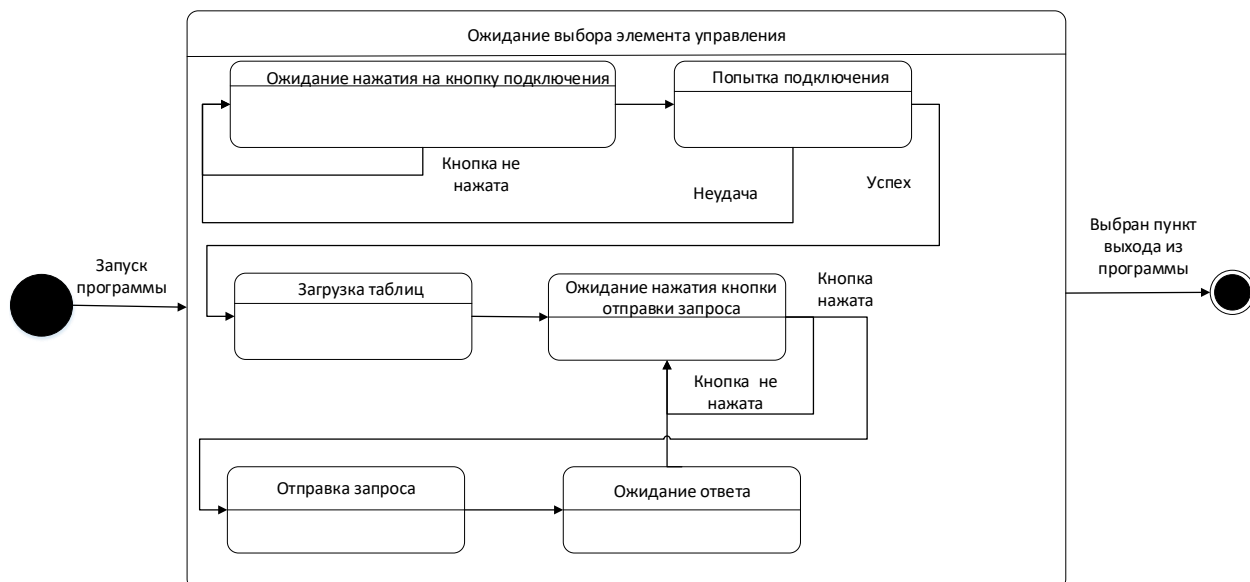


Рисунок 15 - UML-диаграмма состояний программы-клиента (для сотрудника)

На рисунке 16 изображена UML-диаграмма состояний программы-клиента (для покупателя).

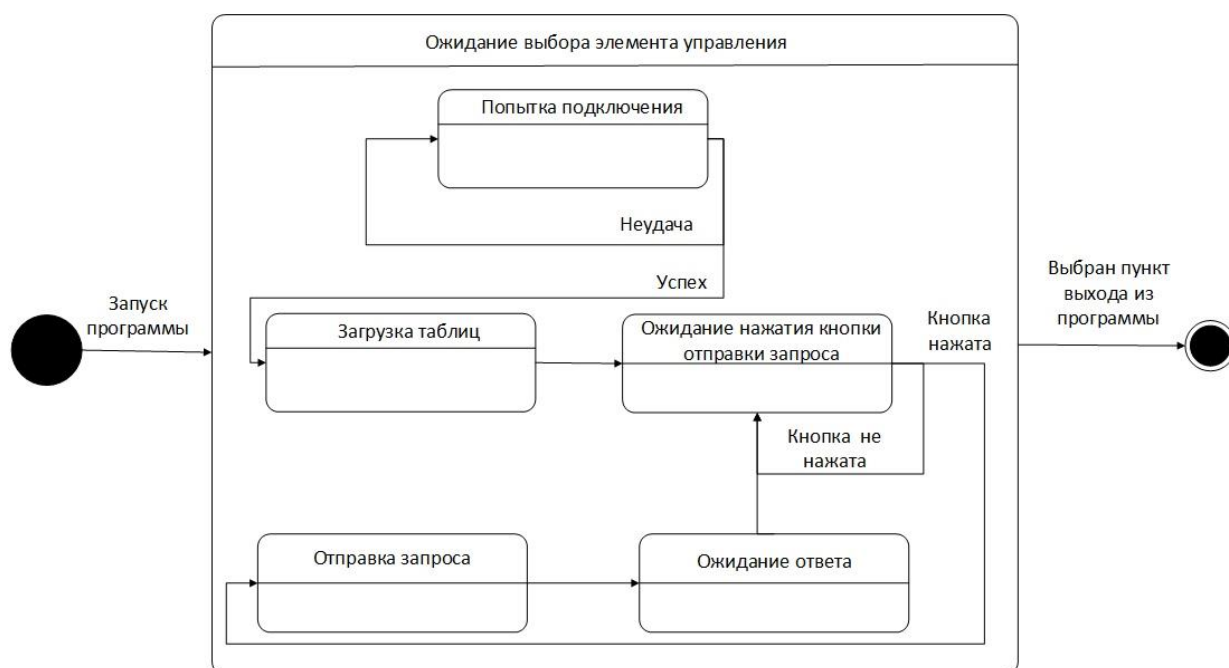


Рисунок 16 - UML-диаграмма состояний программы-клиента (для покупателя)

На рисунке 17 изображена UML-диаграмма состояний программы-сервера.

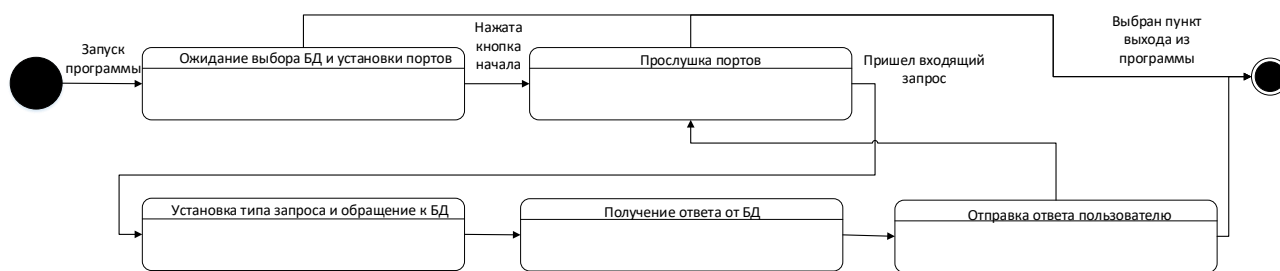


Рисунок 17 - UML-диаграмма состояний программы-сервера

2.4 Описание разработки программы

Для написания данной программы был использован язык программирования C++.

В качестве среды программирования была выбрана платформа Qt.

Интерфейс программ клиента и сервера представлен в виде меню с элементами управления, так как этот способ представления функций наиболее удобен для восприятия и может быть легко освоен даже не подготовленным пользователем.

При запуске программы-клиента (для сотрудника) на экране появляется окно с элементами настроек портов, выбора ip адреса и всплывающими списками для выбора различных типов запроса. После нажатия на кнопку “подключиться” приложение попытается подсоединиться к серверу по заданному в текстовом поле адресу сервера и номеру порта. На выбранный в поле записи числа UDP порт раз в 2 секунды начнут посылаться UDP пакеты для проверки соединения с сервером. Если в течение 5 секунд не придет ни одного ответа от сервера, сервер будет считаться недоступным и пользователь получит соответствующее сообщение. Если подсоединение к серверу прошло успешно, то программа сама загрузит список таблиц БД и выведет их в соответствующий выпадающий список.

После выбора необходимых столбцов и таблиц, а также выбора типа запроса пользователь может нажать на кнопку “Выполнить” в правой или левой

части программы. Если пользователь нажал на левую кнопку, программа отошлет запрос с выбранными параметрами на сервер и будет ждать ответа, а если на правую – программа сначала потребует ввести пароль и, если он верный, отошлет текст запроса на изменение данных в БД.

При запуске программы-клиента (для покупателя) на экране появляется окно с текстовым полем, где, после нажатия кнопки «Показать товары», появится список доступных для заказа наименований.

Далее следует ввести id и количество товара, которое необходимо приобрести и нажать кнопку «Добавить в корзину». Для редактирования списка корзины следует использовать кнопки «Редактировать» и «Удалить товар». После заполнения корзины нужными товарами следует заполнить поле с номером телефона, для связи сотрудника магазина с клиентом и нажать кнопку «Отправить заказ».

При запуске программы-сервера пользователю будет предложено выбрать путь к БД и номера портов TCP и UDP для прослушивания, после чего по нажатию на кнопку старт произойдет попытка прослушивания портов и в случае возникновения ошибки пользователь получит соответствующее сообщение.

На рисунке 18 показана UML-диаграмма классов программы-клиента (для сотрудника).

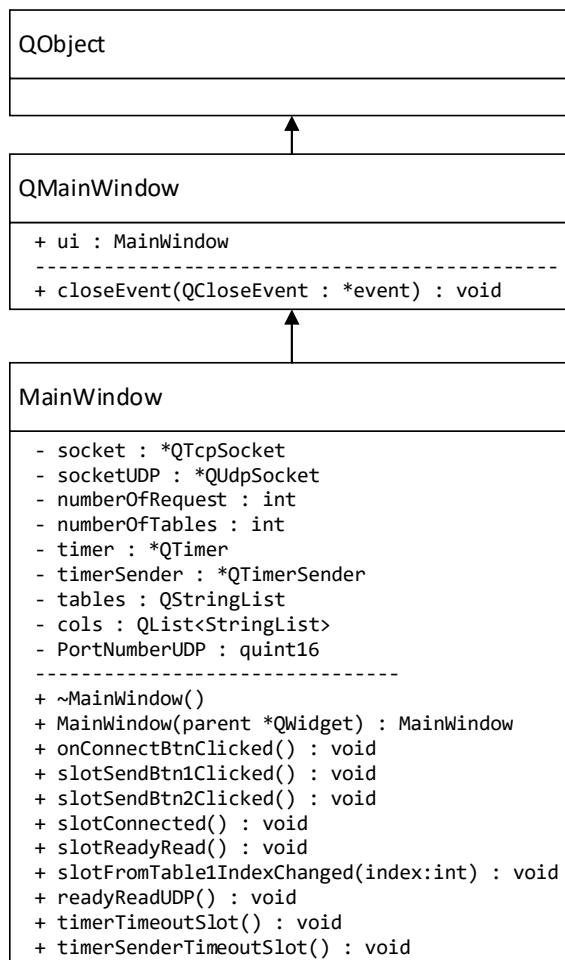


Рисунок 18 - UML-диаграмма классов программы-клиента

На рисунке 19 показана UML-диаграмма классов программы-клиента (для покупателя).

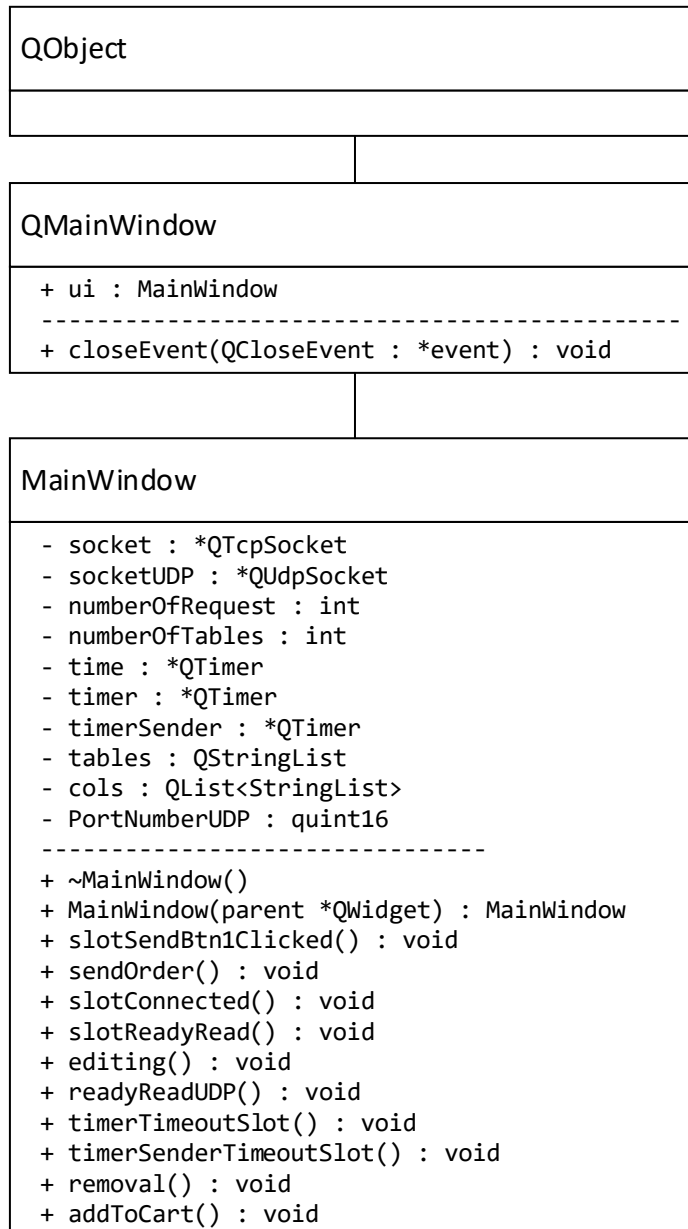


Рисунок 19 - UML-диаграмма классов программы-клиента (для покупателя)

На рисунке 20 показана UML-диаграмма классов программы-сервера.

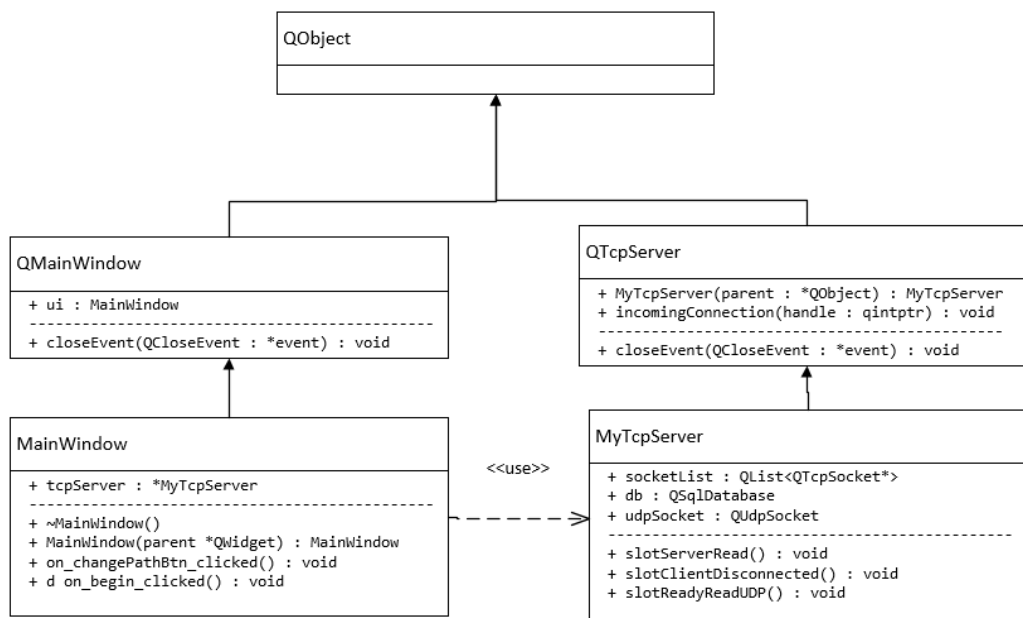


Рисунок 20 - UML-диаграмма классов программы-сервера

В среде разработки Qt класс QObject является базовым для всех остальных классов. Дополнительно были созданы классы MainWindow, который наследуется от класса QMainWindow и отвечает за отображение элементов взаимодействия формы с пользователем и обработку событий, связанных с вводом данных от пользователя и класс TcpServer, наследуемый от класса QtcpServer и отвечающий за сетевое взаимодействие и прослушку портов. Объекты классов взаимодействуют между собой посредством получения ссылок на различные переменные других классов, а также использования объектов классов внутри собственных классов.

2.5 Отладка и тестирование

В качестве среды разработки была выбрана среда Qt, которая содержит в себе все необходимые средства для разработки и отладки модулей и программ. Для отладки программы использовались различные инструменты: точка останова, пошаговое выполнение кода программы, анализ содержимого глобальных и локальных переменных, результаты работы некоторых функций, трассировка.

Тестирование проводилось в рабочем порядке, в процессе разработки и после завершения написания программы, все выявленные ошибки были устранены.

					ПГУ 1.090301.14.001 ПЗ	Лист
						51
Изм.	Лист	№ докум.	Подпись	Дата		

3 ОПИСАНИЕ ПРОГРАММЫ

3.1 Описание клиентского приложения (для сотрудника)

После запуска клиентского приложения (для сотрудника) пользователь может подключиться к серверу по его ip адресу и номеру TCP порта. Проверка соединения будет осуществляться по номеру UDP порта. Если соединение установлено, выпадающие списки с названиями таблиц и столбцов для каждой таблицы заполнятся автоматически. При выборе каждого из них можно регулировать выбираемую таблицу и столбец в ней для запроса. В правой части программы расположена группа переключателей, отвечающая за выбор вида запроса на изменение к БД. В текстовом поле условие можно написать условие для изменяемых элементов таблицы. После нажатия на кнопку Выполнить следует ввести пароль и после проверки сформированный запрос отошлется на сервер, после чего результат запроса будет показан в нижнем поле программы.

Описание всех событий клиентского приложения (для сотрудника) выполнено в таблице 1.

Таблица 1 – описание состояний программы-клиента

Событие	Действие Пользователя	Действие программы
Подключение	Нажата кнопка “Подключиться”	Ожидание ответа от сервера
Показ окна “Вы отключились от сервера!”	Нажатие на кнопку “ОК”	Оповещение пользователя о неудачном соединении с сервером
Показ текста “Поздравляю! Вы подключились к серверу”	-	Оповещение пользователя об удачном соединении с сервером

Продолжение таблицы 1

Отправка запроса серверу	Нажата кнопка “Выполнить” в левой части экрана	Ожидание ответа на запрос от сервера
Показ результата запроса в текстовом поле внизу окна	-	Ожидание действий пользователя после получения ответа от сервера
Показ формы ввода пароля	Нажата кнопка “Выполнить” в правой части экрана	Ожидание ввода пароля пользователя
Выход	Нажата кнопка "Выход"	Выход из приложения

Диаграмма активности программы-клиента (для сотрудника) приведена на рисунке 21.

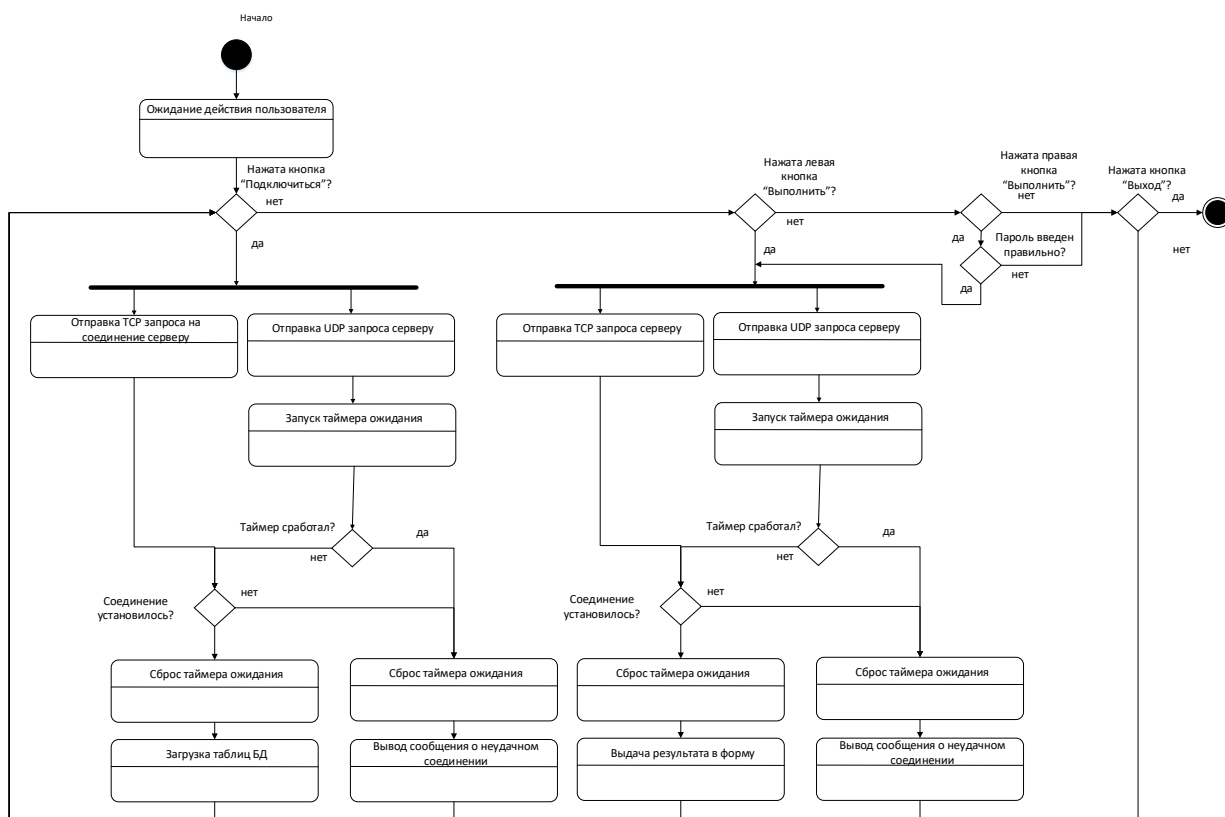


Рисунок 21 – UML- диаграмма активности программы-клиент (для сотрудника)

3.2 Описание клиентского приложения (для покупателя)

После запуска клиентского приложения (для покупателя) программа подключается к серверу по его ip адресу и номеру TCP порта. Проверка соединения будет осуществляться по номеру UDP порта. Если соединение установлено, можно просмотреть список доступных для покупки товаров. В нижней части программы доступно оформление заказа путем добавления товаров в корзину и отправки заказа.

Описание всех событий клиентского приложения (для покупателя) в таблице 2.

Таблица 2 – описание состояний программы-клиента

Событие	Действие Пользователя	Действие программы
Показ окна “Вы отключились от сервера!”	Нажатие на кнопку “ОК”	Оповещение пользователя о неудачном соединении с сервером
Отправка запроса серверу	Нажата кнопка “Показать товары” в левой части экрана	Ожидание ответа на запрос от сервера
Показ результата запроса в текстовом поле внизу окна	-	Ожидание действий пользователя после получения ответа от сервера
Показ окна «Вы не ввели номер телефона»	Нажатие на кнопку “ОК”	Оповещение пользователя о необходимости ввода своего номера телефона
Показ окна «Ваш заказ отправлен»	Нажатие на кнопку “ОК”	Оповещение пользователя о успешной отправке заказа
Отправка запроса серверу	Нажата кнопка "Отправить заказ"	Отправка запроса на сервер

Выход	Нажата кнопка "Выход"	Выход из приложения
-------	--------------------------	---------------------

Диаграмма активности программы-клиента (для покупателя) приведена на рисунке 22.

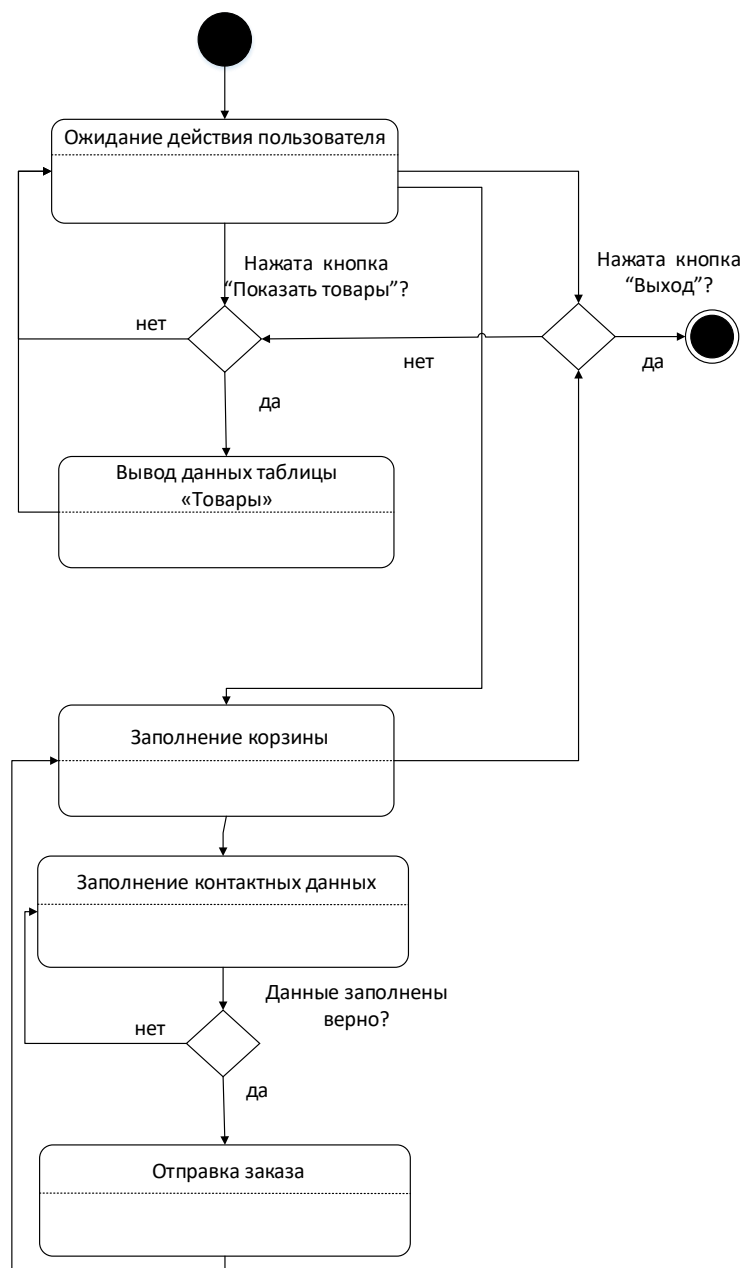


Рисунок 22 - UML- диаграмма активности программы-клиент

3.3 Описание серверного приложения

После запуска сервера пользователь может выбрать путь к БД путем нажатия на кнопку “Указать”, после чего путь к файлу БД выведется в текстовое поле. После установки нужных значений портов в полях “Порт TCP” и “Порт UDP” пользователю следует нажать на кнопку старт, и сервер войдет в состояние ожидания соединений на выбранные порты.

Описание всех событий сервера выполнено в таблице 3.

Таблица 3 - описание состояний программы-сервера

Событие	Действие Пользователя	Действие программы
Выбор файла БД	Нажата кнопка “Указать”	Ожидание выбора пользователем файла БД
Изменение значений полей ввода для портов	Ввод значений в числовые поля портов	Ожидание выбора пользователем прослушиваемых портов
Начало прослушивания портов	Нажата кнопка “Начать работу”	Начало работы сервера
Получение запроса от клиента	-	Запрос к БД
Запрос к БД выполнен	-	Отправка ответа клиенту
Выход	Нажата кнопка "Выход"	Выход из приложения

Диаграмма активности программы-сервера приведена на рисунке 23.

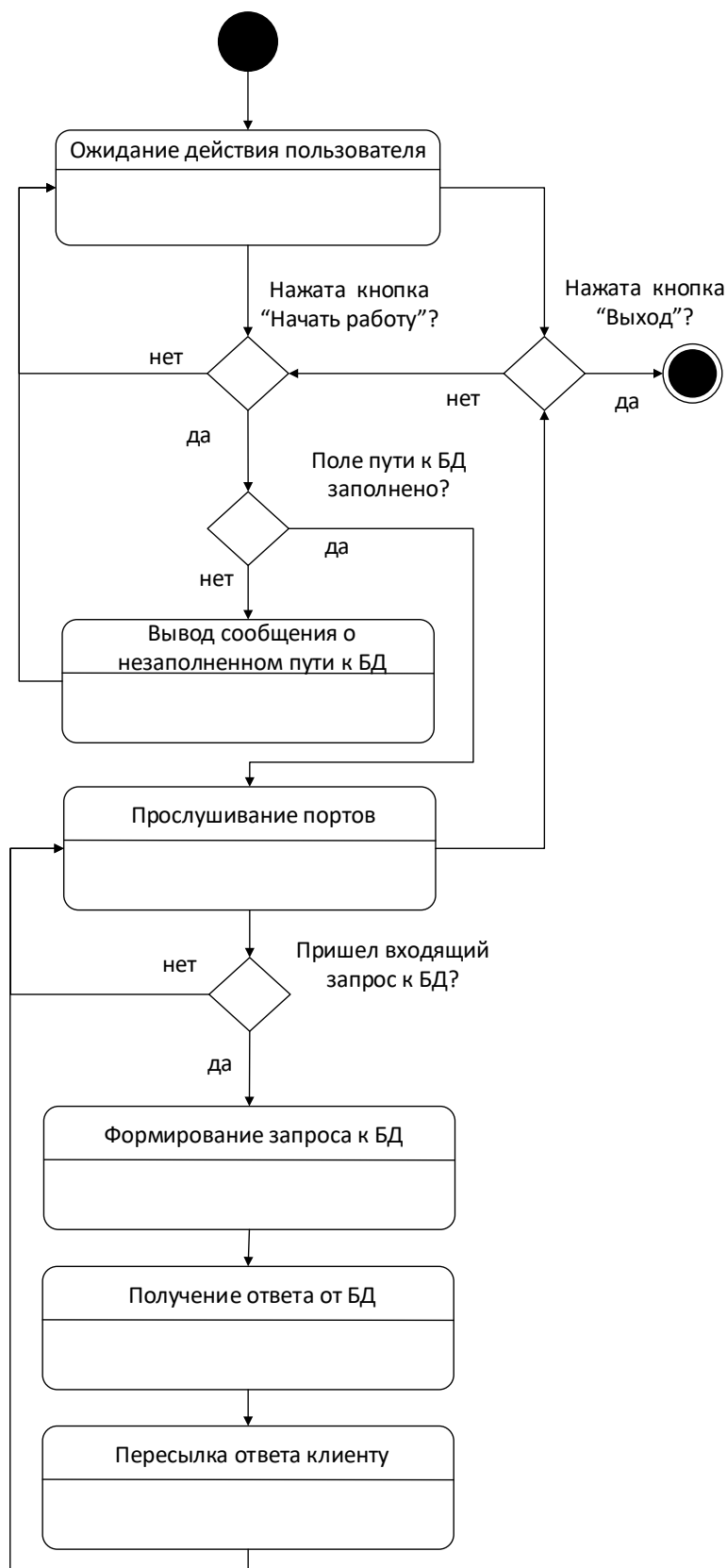


Рисунок 23 – UML- диаграмма активности программы-сервера

3.4 Руководство пользователя

При запуске клиента (Сотрудник) появляется графическое окно, которое включает кнопки “Подключиться”, “Выполнить”, выпадающие списки “Выбор столбцов” и “Выбор таблиц”, группу переключателей “Работа с данными”, текстовые поля “Запрос на выборку”, “Указатель на строку” и “Данные”, а также поля ввода чисел “Порта TCP” и “UDP порт для отправки”.

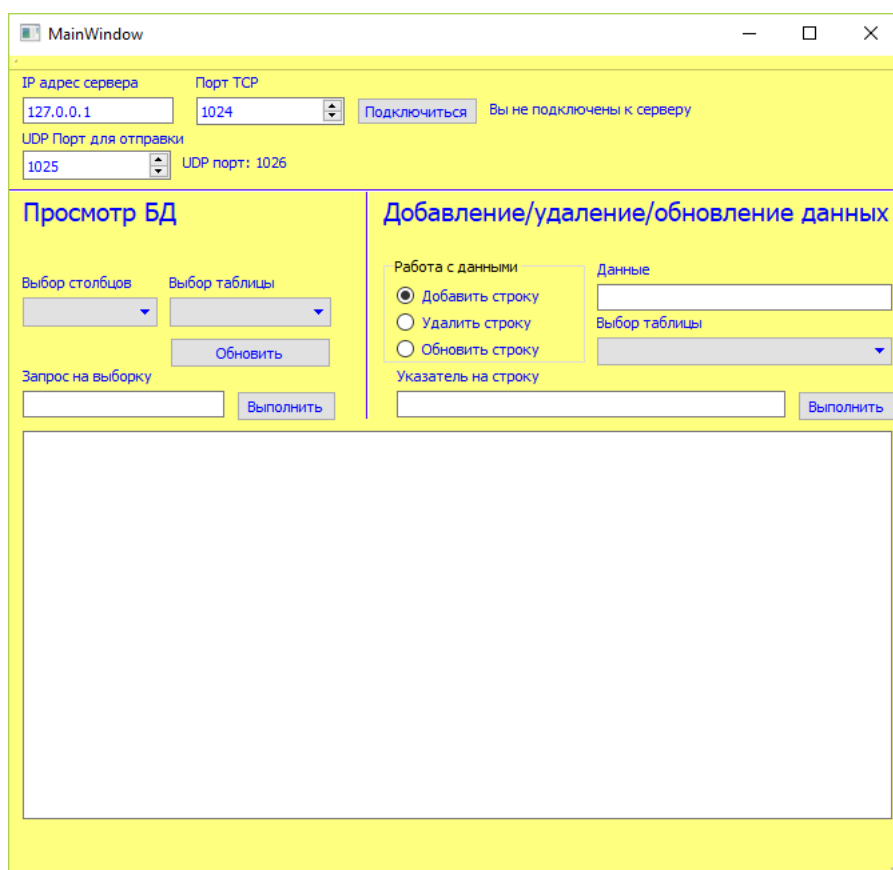


Рисунок 24 – Интерфейс клиентской программы (для сотрудника)

Для установки соединения с сервером клиент должен знать его ip адрес, номера TCP и UDP портов. После ввода необходимых значений клиент должен нажать на кнопку “Подключиться” и подождать 6 секунд. Если по истечении этого времени было показано сообщение “Вы отключились от сервера”, то необходимо изменить настройки в полях, при необходимости проверить соединение и повторить вышеуказанные действия.

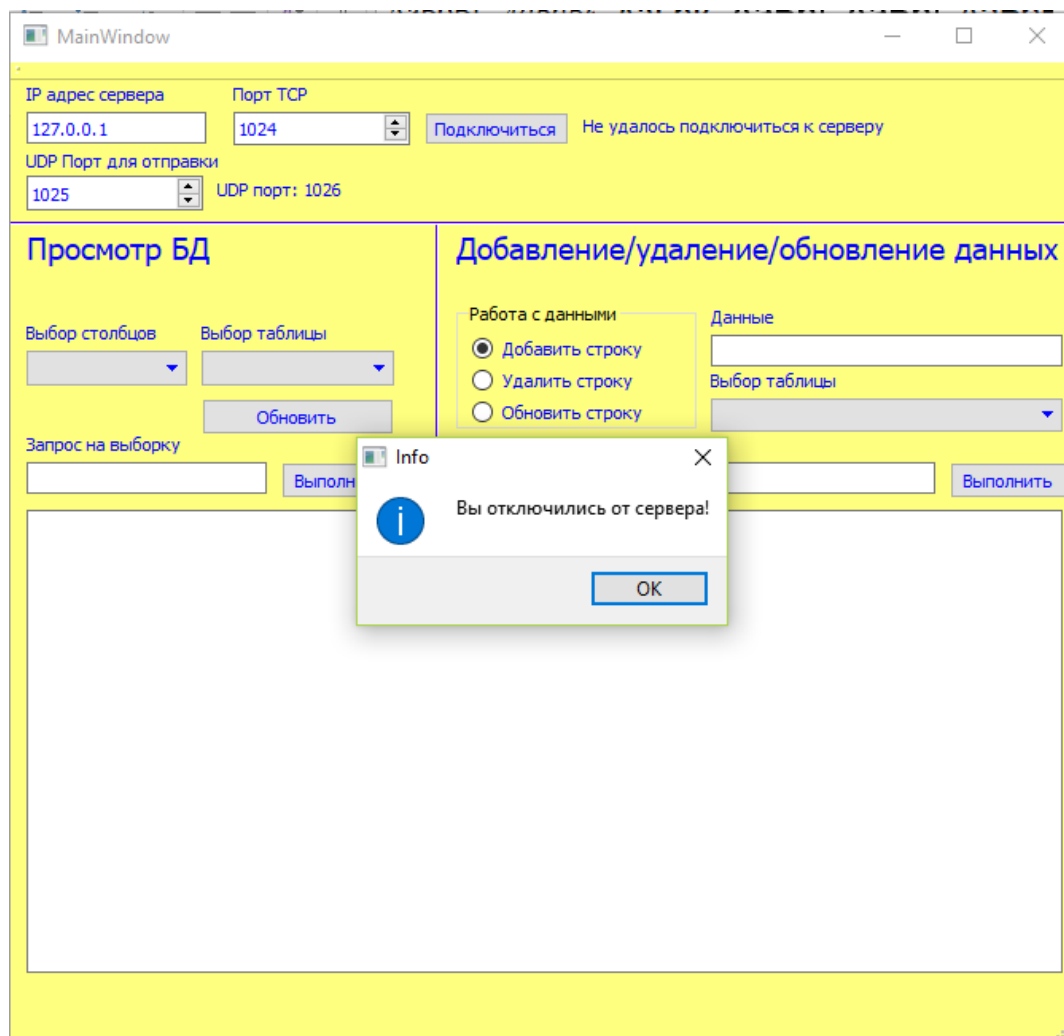


Рисунок 25 – Интерфейс клиентской программы (для сотрудника) при потере соединения с сервером

После успешного подключения к серверу статус надписи изменится на “Поздравляю! Вы подключились к серверу”, а все выпадающие списки будут заполнены значениями.

Рисунок 26 – Интерфейс клиентской программы (для сотрудника) после успешного подключения к серверу

Чтобы выбрать нужную колонку таблицы и саму таблицу, нужно выбрать соответствующие названия в выпадающих списках и при необходимости написать условие в текстовом поле “Запрос на выборку”. После чего необходимо нажать на кнопку “Выполнить” и ждать ответа в нижней части формы.

ID_Товара	НаименованиеТовара	Стоимость
1	Батон нарезной	25
2	Батон столичный	30
3	Батон с отрубями	30
4	Хлеб 'Кирпич'	25
5	Хлеб 'Круглый'	25
6	Хлеб белый	30
7	Сухари ванильные	50
8	Сухари с паприкой	55
9	Сухари с сыром	60
10	Торт 'Прага'	150
11	Торт 'Медовый'	160
12	Торт 'От Палыча'	390
13	Торт 'Наполеон'	250
14	Пирожное 'Картошка'	30
15	Пирожное 'Медовое'	50
16	Пирожное 'Эклер'	45
17	Чипсы 'Pringles с сыром'	135
18	Чипсы 'Pringles со сметаной'	135
19	Чипсы 'Lays с беконом'	60
20	Чипсы 'Lays с крабом'	60

Рисунок 27 – Интерфейс клиентской программы (для сотрудника) после выборки всех данных из таблицы “Товары”

Для отправки запроса на модификацию данных пользователю необходимо выбрать операцию модификации, и, если выбрана операция “Обновить строку”, заполнить текстовое поле “Данные” обновляемыми столбцами. После этого при необходимости пользователь может написать условие для изменения таблиц в поле “Указатель на строку” и нажать на кнопку “Выполнить”. Если в открывшуюся форму введен правильный пароль, приложение отправит текст запроса и покажет результат его выполнения в нижней части окна.

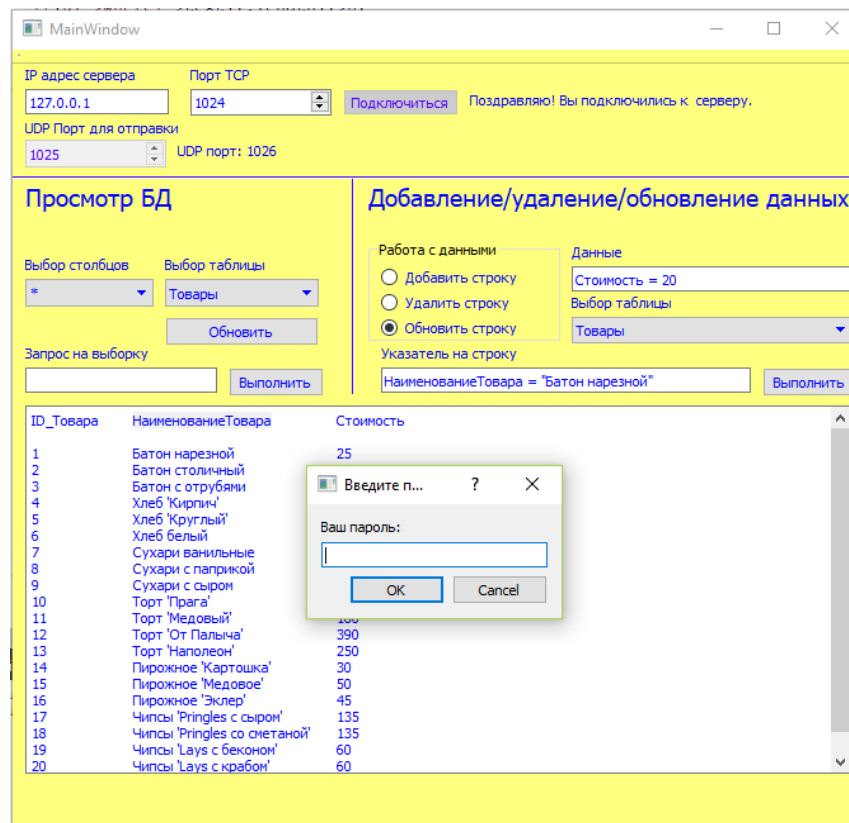


Рисунок 28 – Интерфейс клиентской программы (для сотрудника) при запросе ввода пароля для изменения данных

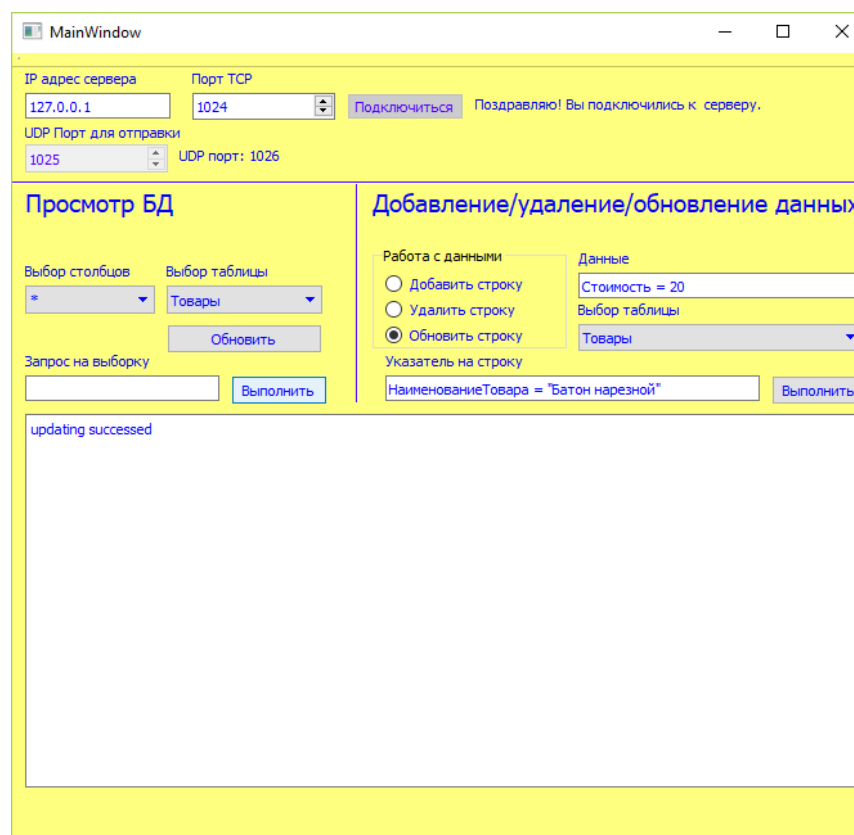


Рисунок 29 – Интерфейс клиентской программы (для сотрудника) после успешного обновления данных

При запуске клиента (Клиент) появляется графическое окно, которое включает кнопки “Показать товары”, “Редактировать”, “Добавить в корзину”, “Удалить товар”, “Отправить заказ”, текстовые поля “Поиск товаров”, “ID товара”, “Количество” и “Номер телефона”.

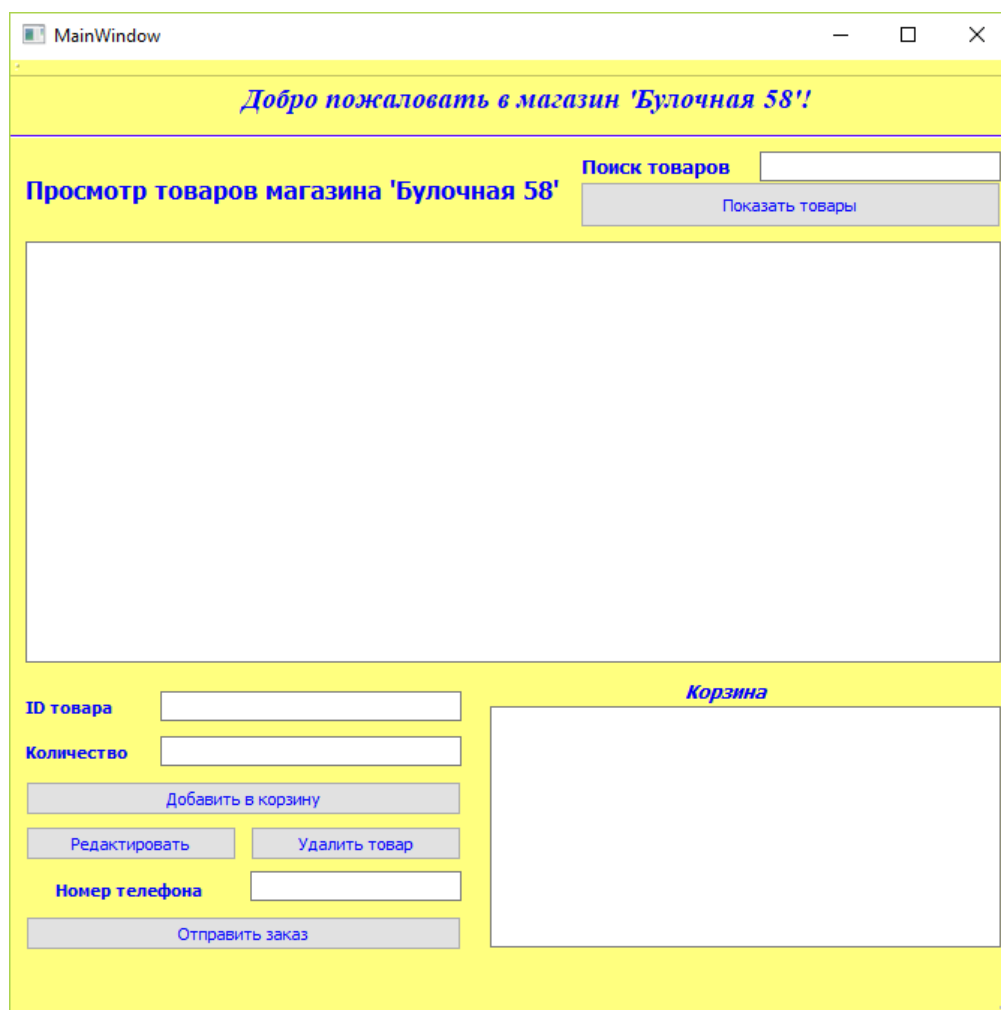


Рисунок 30 – Интерфейс клиентской программы (для покупателя)

Соединение с сервером устанавливается автоматически. Если после запуска приложения, прошло 5 секунд и появилось сообщение “ Просим прощения за неудобства, но произошла ошибка подключения”, то следует перезапустить программу. После успешного подключения к серверу, при нажатии кнопки «Показать товары» в текстовом окне появится список доступных для покупки наименований.

MainWindow

Добро пожаловать в магазин 'Булочная 58'!

Просмотр товаров магазина 'Булочная 58'

Поиск товаров

Показать товары

ID_Товара	НаименованиеТовара	Стоимость
1	Батон нарезной	20
2	Батон столичный	30
3	Батон с отрубями	30
4	Хлеб 'Кирпич'	25
5	Хлеб 'Круглый'	25
6	Хлеб белый	30
7	Сухари ванильные	50
8	Сухари с паприкой	55
9	Сухари с сыром	60
10	Торт 'Прага'	150
11	Торт 'Медовый'	160
12	Торт 'От Пальча'	390
13	Торт 'Наполеон'	250
14	Пирожное 'Картошка'	30
15	Пирожное 'Медовое'	50
16	Пирожное 'Эклер'	45
17	Чипсы 'Pringles с сыром'	135
18	Чипсы 'Pringles со сметаной'	135
19	Чипсы 'Lays с беконом'	60

ID товара

Количество

Добавить в корзину

Редактировать

Удалить товар

Номер телефона

Отправить заказ

Корзина

Рисунок 31 – Интерфейс клиентской программы (для покупателя) после нажатия кнопки “Показать товары”

Для добавления товара в корзину следует заполнить поля “ID товара”, “Количество” и нажать кнопку “Добавить в корзину”.

MainWindow

Добро пожаловать в магазин 'Булочная 58'!

Просмотр товаров магазина 'Булочная 58'

Поиск товаров

Показать товары

ID_Товара	НаименованиеТовара	Стоимость
1	Батон нарезной	20
2	Батон столичный	30
3	Батон с отрубями	30
4	Хлеб 'Кирпич'	25
5	Хлеб 'Круглый'	25
6	Хлеб белый	30
7	Сухари ванильные	50
8	Сухари с паприкой	55
9	Сухари с сыром	60
10	Торт 'Прага'	150
11	Торт 'Медовый'	160
12	Торт 'От Палыча'	390
13	Торт 'Наполеон'	250
14	Пирожное 'Картошка'	30
15	Пирожное 'Медовое'	50
16	Пирожное 'Эклер'	45
17	Чипсы 'Pringles с сыром'	135
18	Чипсы 'Pringles со сметаной'	135
19	Чипсы 'Lays с беконом'	60

ID товара

4

Количество

5

Добавить в корзину

Редактировать

Удалить товар

Номер телефона

Отправить заказ

Корзина

ID товара 4 | Количество5

Рисунок 32 – Интерфейс клиентской программы (для покупателя) после добавления товара в корзину

Если необходимо отредактировать количество товара, то следует заполнить поля “ID товара”, “Количество” и нажать кнопку “Редактировать”.

MainWindow

Добро пожаловать в магазин 'Булочная 58'!

Просмотр товаров магазина 'Булочная 58'

Поиск товаров

Показать товары

ID_Товара	НаименованиеТовара	Стоимость
1	Батон нарезной	20
2	Батон столичный	30
3	Батон с отрубями	30
4	Хлеб 'Кирпич'	25
5	Хлеб 'Круглый'	25
6	Хлеб белый	30
7	Сухари ванильные	50
8	Сухари с паприкой	55
9	Сухари с сыром	60
10	Торт 'Прага'	150
11	Торт 'Медовый'	160
12	Торт 'От Палыча'	390
13	Торт 'Наполеон'	250
14	Пирожное 'Картошка'	30
15	Пирожное 'Медовое'	50
16	Пирожное 'Эклер'	45
17	Чипсы 'Pringles с сыром'	135
18	Чипсы 'Pringles со сметаной'	135
19	Чипсы 'Lays с беконом'	60

ID товара

4

Количество

3

Добавить в корзину

Редактировать

Удалить товар

Номер телефона

Отправить заказ

Корзина

ID товара 4 | Количество3

Рисунок 33 – Интерфейс клиентской программы (для покупателя) после редактирования списка товаров в корзине

Если необходимо удалить товар, то следует заполнить поле “ID товара” и нажать кнопку “Удалить”. Для отправки заказа, нужно заполнить поле “Номер телефона” и нажать кнопку “Отправить заказ”.

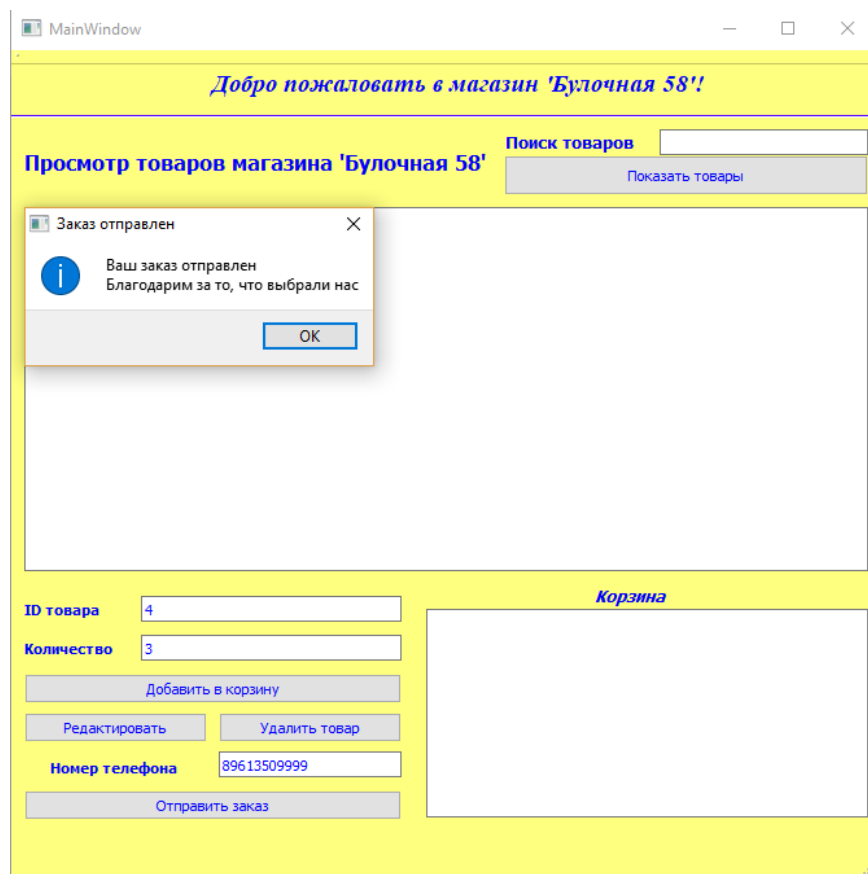


Рисунок 34 – Интерфейс клиентской программы (для покупателя) после отправки заказа

При запуске сервера также появляется окно формы, которое содержит кнопки “Указать” и “Начать работу”, поле ввода текста “Путь к Базе Данных” и поля ввода чисел “Порт TCP” и “Порт UDP”.

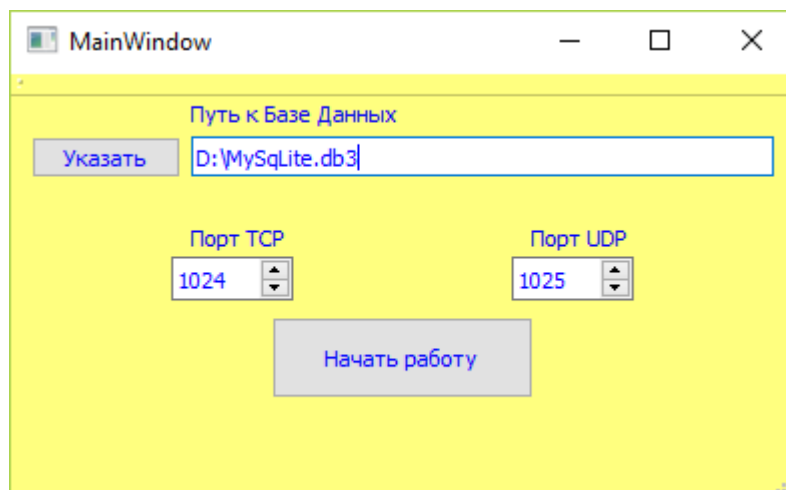


Рисунок 35 – Интерфейс серверной программы

Чтобы выбрать нужную БД пользователь должен нажать на кнопку “Указать”, после чего будет предложено выбрать необходимую БД из окна обозревателя файлов или вписать путь вручную.

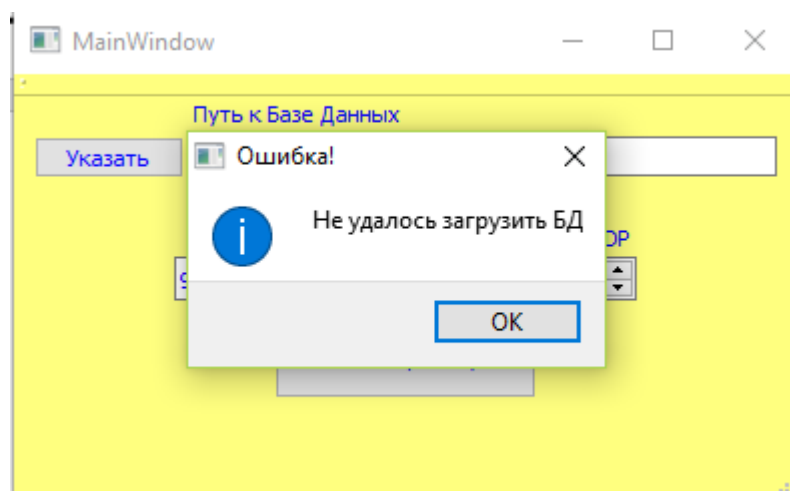


Рисунок 36 – Интерфейс серверной программы после указания неправильного пути к базе данных

Далее стоит настроить порты TCP и UDP для прослушивания и нажать кнопку “Начать работу”. После выполненных действий, если порты удалось прослушать, сервер начнет работать автоматически. Если порты прослушать не удалось, появится соответствующее окно, после чего стоит изменить значения портов для прослушивания.

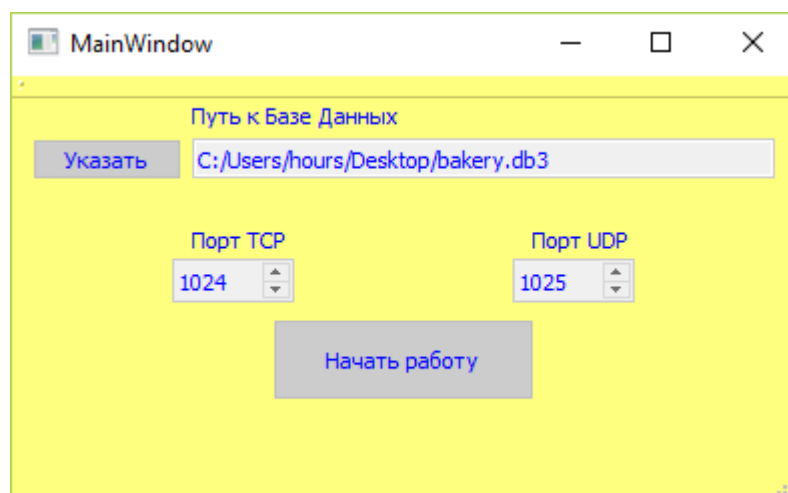


Рисунок 37 – Интерфейс серверной программы в рабочем состоянии

ЗАКЛЮЧЕНИЕ

Результатом выполненной работы является клиент-серверное приложение с интуитивно понятным пользовательским интерфейсом. Приложение позволяет просматривать информацию в базе данных, предоставляет возможность оформлять интернет-заказы, осуществлять обработку различных данных в БД. Данный проект полностью удовлетворяет поставленной цели.

Отладка и тестирование, проведенные на созданной базе данных, а также множество оформленных тестовых заказов, позволили исправить ошибки в программе и добиться корректных результатов работы.

Работа над проектом позволила расширить знания и улучшить навыки создания приложений с клиент-серверной архитектурой, разработки сетевых приложений и работы с базами данных.

В дальнейшем программа может быть улучшена в соответствии с потребностями пользователей, например, можно добавить возможность просмотра ранее созданных заказов. Кроме того, можно добавить личный кабинет пользователя, систему скидок, сохранение данных в текстовый или иной файл.

					ПГУ 1.090301.14.001 ПЗ	Лист
						69
Изм.	Лист	№ докум.	Подпись	Дата		

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Qt Documentation. URL: <http://doc.qt.io/>
2. Стендли Б. Липпман “Основы программирования на C++”
3. Статья на сайте Habr. URL: <https://habr.com/post/128836/>
4. Статья на сайте Habr. URL: <https://habr.com/post/149356/>

					ПГУ 1.090301.14.001 ПЗ	Лист
						70
Изм.	Лист	№ докум.	Подпись	Дата		

ПРИЛОЖЕНИЕ А

Листинги программы

Приложение А.1

Файл сервера mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QWidget>
#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlTableModel>
#include <QTcpServer>
#include <QTcpSocket>
#include <QUdpSocket>

namespace Ui {
class MainWindow;
class TcpServer;
}

class TcpServer : public QTcpServer{
    Q_OBJECT
public:
    explicit TcpServer(QObject * parent=0);
```

```

void incomingConnection(qintptr handle);

int startWorking(QString path,int portNumber,int portNumberUDP);

QList<QTcpSocket*> socketList;

QSqlDatabase db;

QUdpSocket udpSocket;

public slots:

    void slotServerRead();

    void slotClientDisconnected();

    void slotReadyReadUDP();

};

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

    TcpServer* tcpServer;

private:
    Ui::MainWindow *ui;

    void closeEvent(QCloseEvent *event);

private slots:

```

```

void on_changePathBtn_clicked();

void on_begin_clicked();


public slots:


};


#endif // MAINWINDOW_H

```

Приложение А.2 Файл сервера main.cpp

```

#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}

```

Приложение А.3

Файл сервера mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QDebug>
#include <QSqlError>
#include <QSqlRecord>
#include <QCloseEvent>
#include <QMessageBox>
#include <QInputDialog>
#include <QFileDialog>
```

```
TcpServer::TcpServer(QObject *parent) : QTcpServer (parent){

}
```

```
int      TcpServer::startWorking(QString      path,int      portNumber,int
portNumberUDP){
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(path);
    if(!db.open()){
        return -1;
    }
    if(!listen(QHostAddress::Any,portNumber)){
        return -2;
    }
}
```

					ПГУ 1.090301.14.001 ПЗ	Лист
						74
Изм.	Лист	№ докум.	Подпись	Дата		

```

        if(!udpSocket.bind(QHostAddress("127.0.0.1"),portNumberUDP)){
            return -3;
        }
        else{

connect(&udpSocket,SIGNAL(readyRead()),this,SLOT(slotReadyReadUDP()));

        }

        return 0;
    }

MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent),
ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ui->pathDB->setText("D:\\MySqlLite.db3");//Путь к БД по умолчанию
    tcpServer = new TcpServer(this);

    ui->portNumber->setValue(1024);//Порты по умолчанию
    ui->portNumberForUDP->setValue(1025);//Порты по умолчанию

}

MainWindow::~MainWindow()
{
    delete ui;
}

```

```

    }

void TcpServer::incomingConnection(qintptr handle)
{
    QTcpSocket* socket = new QTcpSocket(this);
    socket->setSocketDescriptor(handle);

    this->socketList.append(socket);
    connect(this->socketList[socketList.size()-
1],SIGNAL(readyRead()),this,SLOT(slotServerRead()));
    connect(this->socketList[socketList.size()-
1],SIGNAL(disconnected()),this,SLOT(slotClientDisconnected()));
}

void TcpServer::slotServerRead(){

    QSqlQuery query = QSqlQuery(db);
    QTcpSocket* clientSocket = (QTcpSocket*)sender();
    QString qq(clientSocket->readAll());
    if(!query.exec(qq)){
        clientSocket->write(query.lastError().databaseText().toUtf8());
    }
    else{
        QString outString;

        if(qq=="select * from sqlite_master where type= 'table'")//список таблиц
        {

```

```

while(query.next()){
    outString.append(query.record().value(1).toString()+",");
}
clientSocket->write(outString.remove(outString.size()-1,1).toUtf8());
return;
}

else if(qq.startsWith("pragma")){//список столбцов
    while (query.next()) {
        outString.append(query.record().value(1).toString()+",");
    }
    clientSocket->write(outString.remove(outString.size()-1,1).toUtf8());
    return;
}

else if(qq.startsWith("select")){
    for(int i=0;i<query.record().count();i++)
        outString.append(query.record().fieldName(i)+'\t');
    outString.append("\n\n");
    while(query.next()){
        for(int i=0;i<query.record().count();i++)
            outString.append(query.record().value(i).toString()+"\t");
        outString.append("\n");
    }
    clientSocket->write(outString.toUtf8());
}

else if(qq.startsWith("insert")){
    clientSocket->write(QString("inserting succeeded").toUtf8());
}

```

```

    }

    else if(qq.startsWith("delete")){

        clientSocket->write(QString("deleting succeeded").toUtf8());

    }

    else if(qq.startsWith("update")){

        clientSocket->write(QString("updating succeeded").toUtf8());

    }

}

}

void TcpServer::slotReadyReadUDP(){

    QByteArray buffer;

    buffer.resize(udpSocket.pendingDatagramSize());

    QHostAddress sender;

    quint16 senderPort;

    udpSocket.readDatagram(buffer.data(),buffer.size(),&sender,&senderPort);//получаем
    вопрос на работоспособность

    QString str(udpSocket.readAll());

    udpSocket.writeDatagram(QString("").toUtf8(),sender,senderPort);//шлем
    ответ о работоспособности

}

void TcpServer::slotClientDisconnected()//разрыв соединения с клиентом

{

    QTcpSocket* socket = (QTcpSocket*)sender();

    socket->close();

```

```

    }

void MainWindow::closeEvent(QCloseEvent *event)
{
    event->ignore();

    if (QMessageBox::Yes == QMessageBox::question(this, "Внимание!",
                                                "Вы действительно хотите закрыть
приложение?",
                                                QMessageBox::Yes|QMessageBox::No))
    {
        event->accept();
    }
}

void MainWindow::on_changePathBtn_clicked()
{
    QString fileName = QFileDialog::getOpenFileName(this,
                                                    QString::fromUtf8("Открыть файл"),
                                                    QDir::currentPath(),
                                                    "SQLite databases (*.db3);;All files (*.*)");
    ui->pathDB->setText(fileName);
}

void MainWindow::on_begin_clicked()
{
    if(ui->pathDB->text().endsWith(".db3")){

```

```

switch(tcpServer->startWorking(ui->pathDB->text(),ui->portNumber-
>value(),ui->portNumberForUDP->value())){
    case -1:
        QMessageBox::information(this,"Ошибка!","Не удалось
загрузить БД");
        return;
    case -2:
        QMessageBox::information(this,"Ошибка!","Не удалось
прослушать порт TCP");
        return;
    case -3:
        QMessageBox::information(this,"Ошибка!","Не удалось
прослушать порт UDP");
        return;
    default:
        break;
}

ui->changePathBtn->setEnabled(false);
ui->pathDB->setEnabled(false);
ui->portNumber->setEnabled(false);
ui->portNumberForUDP->setEnabled(false);
ui->begin->setEnabled(false);
}
else
    QMessageBox::information(this,"Info","Выбранный файл не является
файлом БД SQLite");

```

}

Приложение А.4 Файл клиента (Сотрудник)

mainwindow.h

```
#ifndef MAINWINDOW_H
```

```
#define MAINWINDOW_H
```

```
#include <QMainWindow>
```

```
#include <QTcpSocket>
```

```
#include <QUdpSocket>
```

```
#include <QTimer>
```

```
#include <QTime>
```

```
namespace Ui {
```

```
class MainWindow;
```

```
}
```

```
class MainWindow : public QMainWindow
```

```
{
```

```
    Q_OBJECT
```

```
public:
```

```
    explicit MainWindow(QWidget *parent = 0);
```

```
    //void incomingConnection(qintptr);
```

```
    ~MainWindow();
```

					ПГУ 1.090301.14.001 ПЗ	Лист
						81
Изм.	Лист	№ докум.	Подпись	Дата		

private:

Ui::MainWindow *ui;

QTcpSocket *socket;

QUdpSocket *socketUDP;

int numberOfRequest=0;

int numberOfTables=0;

QTimer* timer;

QTimer* timerSender;

QStringList tables;//список таблиц

QList<QStringList> cols;//всех таблиц

quint16 myPortNumberUDP;//хранит udp порт прослушиваемый этой копией проги

void *closeEvent*(QCloseEvent *event);

public slots:

void tableReadyRead();

void onConnectBtnClicked();

void slotSendBtn1Clicked();

void slotSendBtn2Clicked();

void slotConnected();

void slotReadyRead();

void slotFromTable1IndexChanged(int);

void readyReadUDP();

void timerTimeoutSlot();

void timerSenderTimeoutSlot();

```
};
```

```
#endif // MAINWINDOW_H
```

Приложение А.5 Файл клиента (Сотрудник) main.cpp

```
#include "mainwindow.h"
```

```
#include <QApplication>
```

```
int main(int argc, char *argv[])
```

```
{
```

```
    QApplication a(argc, argv);
```

```
    MainWindow w;
```

```
    w.show();
```

```
    return a.exec();
```

```
}
```

Приложение А.6 Файл клиента (Сотрудник)

mainwindow.cpp

```
#include "mainwindow.h"
```

```
#include "ui_mainwindow.h"
```

```
#include <QTcpSocket>
```

```
#include <QMessageBox>
```

```
#include <QInputDialog>
```

					ПГУ 1.090301.14.001 ПЗ	Лист
						83
Изм.	Лист	№ докум.	Подпись	Дата		

```
#include <QCloseEvent>
```

```
MainWindow::MainWindow(QWidget *parent) :
```

```
    QMainWindow(parent),
```

```
    ui(new Ui::MainWindow)
```

```
{
```

```
    ui->setupUi(this);
```

```
    timer = new QTimer(this);
```

```
    timerSender = new QTimer(this);
```

```
    timer->setInterval(6500);//интервал (5 сек)
```

```
    timerSender->setInterval(2000);//проверка подключения (2 сек)
```

```
    connect(timer,SIGNAL(timeout()),this,SLOT(timerTimeoutSlot()));
```

```
connect(timerSender,SIGNAL(timeout()),this,SLOT(timerSenderTimeoutSlot()));
```

```
    connect(ui-
```

```
>connectBtn,SIGNAL(clicked()),this,SLOT(onConnectBtnClicked()));
```

```
    connect(ui-
```

```
>sendBtn1,SIGNAL(clicked()),this,SLOT(slotSendBtn1Clicked()));
```

```
    connect(ui-
```

```
>sendBtn2,SIGNAL(clicked()),this,SLOT(slotSendBtn2Clicked()));
```

```
    socket = new QTcpSocket(this);//создание сокета
```

```
    connect(socket, SIGNAL(connected()), SLOT(slotConnected()));
```

```
    connect(socket, SIGNAL(readyRead()), SLOT(slotReadyRead()));
```

```
    connect(ui->pushButton,SIGNAL(clicked()),this,SLOT(tableReadyRead()));
```

```
    socketUDP = new QUdpSocket(this);
```

```

for(int i=1026;i<65536;i++){//ищем первый подходящий сокет
    if(!socketUDP->bind(QHostAddress("127.0.0.1"),i))
    {
        if(i==65535){
            QMessageBox::information(this,"Info","Не удалось подключиться
ни к одному UDP порту!");
            QApplication::exit(0);
        }
    }
    else{
        myPortNumberUDP=i;

connect(socketUDP,SIGNAL(readyRead()),this,SLOT(readyReadUDP()));

        ui->listenPortLabel->setText(QString("UDP порт: %1").arg(i));
        break;
    }
}

connect(ui-
>fromTable1,SIGNAL(currentIndexChanged(int)),this,SLOT(slotFromTable1IndexC
hanged(int)));

    ui->hostAdres->setText("127.0.0.1");
    ui->portNumber->setValue(1024);
    ui->portNumberUDP->setValue(1025);
}

```

```
MainWindow::~MainWindow()
```

```
{  
    delete ui;  
}
```

```
void MainWindow::tableReadyRead(){
```

```
    QString selectString("select * from sqlite_master where type=  
'table'");//список таблиц
```

```
    socket->write(selectString.toUtf8());
```

```
    ui->fromTable1->blockSignals(true);
```

```
    ui->fromTable1->clear();
```

```
    ui->fromTable1->blockSignals(false);
```

```
    ui->selectCols->clear();
```

```
    cols.clear();
```

```
    numberOfRequest = 0;
```

```
    numberOfTables = 0;
```

```
}
```

```
void MainWindow::closeEvent(QCloseEvent *event)
```

```
{
```

```
    event->ignore();
```

```
    if (QMessageBox::Yes == QMessageBox::question(this, "Внимание!",
```

```
        "Вы действительно хотите закрыть приложение?",
```

```
        QMessageBox::Yes|QMessageBox::No))
```

```
{
```

```

        socket->disconnectFromHost();

        socket->close();

        socket->deleteLater();

        event->accept();

    }

}

void MainWindow::onConnectBtnClicked()

{

    socket->connectToHost(ui->hostAdres->text(),ui->portNumber-
>value());//пытаемся подключиться к серверу

    timerSender->start();

    timer->start();

    ui->connectBtn->setEnabled(false);

    ui->portNumberUDP->setEnabled(false);

    //ui->portNumberUDP->setEnabled(false);

}

void MainWindow::slotConnected(){

    ui->statusLbl->setText("Поздравляю! Вы подключились к серверу.");

    if(numberOfRequest==0)//если это 1й запрос

    {

        QString selectString("select * from sqlite_master where type=
'table'");//список таблиц

        socket->write(selectString.toUtf8());

    }

}

```

```

    }

    void MainWindow::slotReadyRead(){

        QString qq(socket->readAll());

        if(numberOfRequest==0)//ожидаем список таблиц
        {
            numberOfRequest++;

            tables=qq.split(',');

            numberOfTables=tables.size();//количество таблиц
            for(int i=0;i<numberOfTables;++i)//заполняем таблицы
            {
                ui->fromTable1->addItem(tables[i]);
                ui->fromTable2->addItem(tables[i]);
            }

            if(numberOfTables>0){

                QString selectString("pragma table_info("+tables[0]+")");//количество
                столбцов

                socket->write(selectString.toUtf8());

                return;
            }

            else{

                QMessageBox::information(this,"Info","В выбранной БД отсутствуют
                таблицы!");

                QApplication::exit();
            }
        }
    }

```

```

    }

    return;
}

else if(numberOfRequest>0 && numberOfRequest < numberOfTables){//
    cols.append(qq.split(','));//добавляем список колонок текущей таблицы
    cols[numberOfRequest-1].append("*");

    QString                                                    selectString("pragma
table_info("+tables[numberOfRequest]+"");

    numberOfRequest++;

    socket->write(selectString.toUtf8());

}

else    if(numberOfRequest>0    &&    numberOfRequest    ==
numberOfTables){//получаем последний список столбцов

    cols.append(qq.split(','));//список колонок текущей таблицы
    cols[numberOfRequest-1].append("*");
    ui->selectCols->clear();
    for(int i=0;i<cols[0].length();i++)
        ui->selectCols->addItem(cols[0][i]);
    numberOfRequest++;
}

else{

    ui->textEdit->setText(qq);

}

}

```

```

void MainWindow:: slotSendBtn1Clicked(){
    QString selectString="select "+ui->selectCols->currentText()+" from "
        +ui->fromTable1->currentText();
    if(ui->where1->text().length()>0)
        selectString.append(" where "+ui->where1->text());
    socket->write(selectString.toUtf8());//текст запроса
}

void MainWindow:: slotSendBtn2Clicked(){
    //проверяем ввод пароля и посылаем запрос
    bool ok;

    QString password = QDialog::getText(this,"Введите пароль","Ваш
пароль:",QLineEdit::Normal,"",&ok);
    QString selectString;
    if(ok && password=="Qwerty")
    {
        if(ui->insertRB->isChecked()){//добавление строки
            selectString.append("insert into "+ ui->fromTable2->currentText()+ "
values(" + ui->values->text()+")");
        }
        else if(ui->updateRB->isChecked()){//обновление таблицы
            selectString.append("update "+ ui->fromTable2->currentText()+" set
"+ui->values->text());
            if(ui->where2->text().length()>0)
                selectString.append(" where "+ui->where2->text());//условие
обновления

```

```

    }

    else{//удаление строки

        selectString.append("delete from "+ui->fromTable2->currentText());

        if(ui->where2->text().length()>0)

            selectString.append("      where      "+ui->where2->text());//условие
удаления

    }

    socket->write(selectString.toUtf8());//текст запроса

}

}

void  MainWindow::slotFromTable1IndexChanged(int  index){//изменение
значения столбцов

    if(numberOfRequest>numberOfTables)

    {

        ui->selectCols->clear();

        for(int i=0;i<cols[index].length();i++)

            ui->selectCols->addItem(cols[index][i]);//добавление          списка
столбцов

    }

}

void  MainWindow::readyReadUDP();//пришло подтверждение что серв на
связи

{

    QByteArray buffer;

    buffer.resize(socketUDP->pendingDatagramSize());

```

```

        QHostAddress sender;

        quint16 senderPort;

        socketUDP->
>readDatagram(buffer.data(),buffer.size(),&sender,&senderPort);

        QString str(socketUDP->readAll());

        if(senderPort==ui->portNumberUDP->value()                &&
senderPort!=myPortNumberUDP){//если я получил ответ от порта сервера и не от
моего же порта то все норм

            timer->start();//сброс таймера

            return;

        }

    }

void MainWindow::timerTimeoutSlot(){

    ui->statusLbl->setText("Не удалось подключиться к серверу");

    timer->stop();

    timerSender->stop();


    ui->connectBtn->setEnabled(true);

    ui->portNumberUDP->setEnabled(true);

    QMessageBox::information(this,"Info","Вы отключились от сервера!");

}

void MainWindow::timerSenderTimeoutSlot()

{

```

```

        QByteArray datagram="";

        quint16 port=ui->portNumberUDP->value();

        socketUDP->
>writeDatagram(datagram.data(),datagram.size(),QHostAddress("127.0.0.1"),port);
    }

```

Приложение А.7 Файл клиента (Клиент) mainwindow.h

```

#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QTcpSocket>
#include <QUdpSocket>
#include <QTimer>
#include <QTime>
#include "QtSql/QtSqlDatabase"
#include "QtSql/QtSqlQuery"

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

```

public:

```
explicit MainWindow(QWidget *parent = 0);
```

```
//void incomingConnection(qintptr);
```

```
~MainWindow();
```

private:

```
Ui::MainWindow *ui;
```

```
QSqlDatabase db;
```

```
QString hostAdres = QString("127.0.0.1");
```

```
int portNumber = 1024;
```

```
int portNumberUDP = 1025;
```

```
QTcpSocket *socket;
```

```
QUdpSocket *socketUDP;
```

```
int numberOfRequest=0;
```

```
int numberOfTables=0;
```

```
QTimer* time;
```

```
QTimer* timer;
```

```
QTimer* timerSender;
```

```
QStringList tables;//список таблиц
```

```
QList<QStringList> cols;//всех таблиц
```

quint16 myPortNumberUDP;//хранит udp порт прослушиваемый этой копией проги

```
void closeEvent(QCloseEvent *event);
```

public slots:

					ПГУ 1.090301.14.001 ПЗ	Лист
						94
Изм.	Лист	№ докум.	Подпись	Дата		

```

void sendOrder1();

void editing();

void removal();

void sendOrder();

void addToCart();

//void tableReadyRead();

void slotSendBtn1Clicked();

//void slotSendBtn2Clicked();

void slotConnected();

void slotReadyRead();

//void slotFromTable1IndexChanged(int);

void readyReadUDP();

void timerTimeoutSlot();

void timerSenderTimeoutSlot();

};

#endif // MAINWINDOW_H

```

Приложение А.8 Файл клиента (Клиент) main.cpp

```

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])

{

```

					ПГУ 1.090301.14.001 ПЗ	Лист
						95
Изм.	Лист	№ докум.	Подпись	Дата		

```

    QApplication a(argc, argv);

    MainWindow w;

    w.show();

    return a.exec();
}

```

Приложение А.8 Файл клиента (Клиент)

mainwindow.cpp

```

#include "mainwindow.h"

#include "ui_mainwindow.h"

#include <QTcpSocket>
#include <QMessageBox>
#include <QInputDialog>
#include <QCloseEvent>
#include <QProcess>

#include "QtSql/QtSqlDatabase"
#include "QtSql/QtSqlQuery"
#include "QtSql/QtSqlError"

```

```

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    //Подключаем базу данных

```

```

db = QSqlDatabase::addDatabase("QSQLITE");
db.setDatabaseName("basket.db3");
db.open();
if (!db.open()) {
    qDebug() << db.lastError().text();
}

QSqlQuery query;
query.exec("create table Заказ (ID_Товара INTEGER PRIMARY KEY
NOT NULL, Количество INTEGER NOT NULL);");

time = new QTimer(this);
timer = new QTimer(this);
timerSender = new QTimer(this);
time->setInterval(500);
timer->setInterval(6500);//интервал (5 сек)
timerSender->setInterval(2000);//проверка подключения (2 сек)
connect(timer,SIGNAL(timeout()),this,SLOT(timerTimeoutSlot()));

connect(timerSender,SIGNAL(timeout()),this,SLOT(timerSenderTimeoutSlot()));

//connect(ui-
>connectBtn,SIGNAL(clicked()),this,SLOT(onConnectBtnClicked()));

connect(ui-
>sendBtn1,SIGNAL(clicked()),this,SLOT(slotSendBtn1Clicked()));

//connect(ui-
>sendBtn2,SIGNAL(clicked()),this,SLOT(slotSendBtn2Clicked()));

socket = new QTcpSocket(this);//создание сокета
connect(ui->pushButton, SIGNAL(clicked()), this, SLOT(sendOrder1()));

```

```

connect(time,SIGNAL(timeout()),this,SLOT(sendOrder()));
connect(socket, SIGNAL(connected()), SLOT(slotConnected()));
connect(socket, SIGNAL(readyRead()), SLOT(slotReadyRead()));
//connect(timerSender,SIGNAL(timeout()),this, SLOT(tableReadyRead()));
connect(ui->pushButton_3, SIGNAL(clicked()), this, SLOT(addToCart()));
connect(ui->pushButton_2, SIGNAL(clicked()), this, SLOT(editing()));
connect(ui->pushButton_4, SIGNAL(clicked()), this, SLOT(removal()));

socketUDP = new QUdpSocket(this);
for(int i=1026;i<65536;i++){//ищем первый подходящий сокет
    if(!socketUDP->bind(QHostAddress("127.0.0.1"),i))
    {
        if(i==65535){
            QMessageBox::information(this,"Ошибка","Просим прощения за
неудобства, но произошла ошибка подключения.\n"
                                "Error: Не удалось подключиться ни к
одному UDP порту!");
            QApplication::exit(0);
        }
    }
    else{
        myPortNumberUDP=i;

connect(socketUDP,SIGNAL(readyRead()),this,SLOT(readyReadUDP()));

        //ui->listenPortLabel->setText(QString("UDP порт: %1").arg(i));
        break;

```

```

    }

}

//connect(ui-
>fromTable1,SIGNAL(currentIndexChanged(int)),this,SLOT(slotFromTable1IndexC
hanged(int)));

//ui->hostAdres->setText("127.0.0.1");
//QString hostAdres = QString("127.0.0.1");
//ui->portNumber->setValue(1024);
//ui->portNumberUDP->setValue(1025);

socket->connectToHost(hostAdres, portNumber);//пытаемся подключиться
к серверу

timerSender->start();
timer->start();

}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::closeEvent(QCloseEvent *event)
{
    event->ignore();
}

```

```

if (QMessageBox::Yes == QMessageBox::question(this, "Внимание!",
        "Вы действительно хотите закрыть приложение?",
        QMessageBox::Yes|QMessageBox::No))
{
    socket->disconnectFromHost();
    socket->close();
    socket->deleteLater();
    event->accept();
}

 QSqlQuery query;
 query.exec("DROP TABLE IF EXISTS Заказ;");
}

void MainWindow::sendOrder1(){
    QString number = ui->lineEdit_3->text();
    if (number == ""){
        QMessageBox::information(this, "Внимание", "Вы не ввели номер
        телефона\n"
        "Без него с Вами не смогут связаться по
        поводу заказа");
        return;
    }

    QDate dateToday = QDate::currentDate();
    QString str;
    str = dateToday.toString("dd_MM_yy_"); //str - "03.07.07" str
    QString nameTable = str + number;

```

```

        QString selectString="create table T"+nameTable+" (ID_Товара,
Количество);";

        socket->write(selectString.toUtf8());

        time->start();
    }

```

```

void MainWindow::sendOrder()
{
    QSqlQuery query;
    QString ID;
    QString Number;
    QString number = ui->lineEdit_3->text();
    if (number == ""){
        return;
    }

    QDate dateToday = QDate::currentDate();
    QString str;
    str = dateToday.toString("dd_MM_yy_"); //str - "03.07.07" str
    QString nameTable = str + number;
    QString selectString = "insert into T"+nameTable+" (ID_Товара,
Количество) values";

    query.exec("select * from Заказ");
    ui->textEdit_2->clear();
    while (query.next()){
        ID = query.value(0).toString();
        Number = query.value(1).toString();
    }
}

```

```

        selectString.append("(" + ID + ", " + Number + "),");

    }

    selectString.append("('NaN', 'NaN');");
    socket->write(selectString.toUtf8());
    time->stop();

    QMessageBox::information(this, "Заказ отправлен", "Ваш заказ
отправлен\n"

                                "Благодарим за то, что выбрали нас");
}

void MainWindow::slotConnected(){
    ui->statusLbl->setText("Добро пожаловать в магазин
'Булочная 58!');

    if(numberOfRequest==0)//если это 1й запрос
    {
        QString selectString("select * from sqlite_master where type=
'table'");//список таблиц
        socket->write(selectString.toUtf8());
    }
}

/*void MainWindow::tableReadyRead(){
    QString selectString("select * from sqlite_master where type=
'table'");//список таблиц
    socket->write(selectString.toUtf8());
}

```

```

//ui->fromTable1->clear();

numberOfRequest = 0;

}*/

void MainWindow::slotReadyRead(){

    QString qq(socket->readAll());

    if(numberOfRequest==0)//ожидаем список таблиц
    {
        numberOfRequest++;

        tables=qq.split(',');

        numberOfTables=tables.size();//количество таблиц

        /*for(int i=0;i<numberOfTables;++i)//заполняем таблицы
        {
            //ui->fromTable1->addItem(tables[i]);

            //ui->fromTable2->addItem(tables[i]);

        }*/

        if(numberOfTables>0){

            QString selectString("pragma table_info("+tables[0]+")");//количество
            столбцов

            socket->write(selectString.toUtf8());

            return;

        }

        else{

```

```

        QMessageBox::information(this,"Ошибка","Просим прощения за
неудобства,\n но произошла при загрузке списка товаров");

        QApplication::exit();
    }

    return;
}

else if(numberOfRequest>0 && numberOfRequest < numberOfTables){//
    cols.append(qq.split(','));//добавляем список колонок текущей таблицы
    cols[numberOfRequest-1].append("*");

    QString selectString("pragma
table_info("+tables[numberOfRequest]+")");

    numberOfRequest++;

    socket->write(selectString.toUtf8());

}

else if(numberOfRequest>0 && numberOfRequest ==
numberOfTables){//получаем последний список столбцов

    cols.append(qq.split(','));//список колонок текущей таблицы
    cols[numberOfRequest-1].append("*");

    /*ui->selectCols->clear();

    for(int i=0;i<cols[0].length();i++)
        ui->selectCols->addItem(cols[0][i]);*/

    numberOfRequest++;

}

else{

    ui->textEdit->setText(qq);

```

```

    }

}

void MainWindow:: slotSendBtn1Clicked(){
    QString selectString="select * from Товары";
    if(ui->where1->text().length()>0)
        selectString.append(" where НаименованиеТовара = '"+ui->where1->text()+"");
    socket->write(selectString.toUtf8());//текст запроса
}

void MainWindow:: addToCart(){
    QSqlQuery query;
    QString ID;
    QString Number;
    /*QString selectString("select НаименованиеТовара from Товар where
ID_Товара = "+ID); //исправить
    socket->write(selectString.toUtf8());
    QString Rezult(ID+", "+qq+", "+Number);
    ui->textEdit_2->setText(qq);*/
    query.exec("insert into Заказ values('"+ui->lineEdit->text()+"', '"+ui->lineEdit_2->text()+"');");
    query.exec("select * from Заказ");
    ui->textEdit_2->clear();
    while (query.next()){
        ID = query.value(0).toString();

```

```

        Number = query.value(1).toString();

        ui->textEdit_2->insertPlainText("ID          товара          "+ID+"          |
Количество"+Number+"\n");

    }

}

```

```

void MainWindow:: editing(){

    QSqlQuery query;

    QString ID;

    QString Number;

    query.exec("update  Заказ set  Количество = "+ui->lineEdit_2->text()+"
where ID_Товара = "+ui->lineEdit->text());

    query.exec("select * from Заказ");

    ui->textEdit_2->clear();

    while (query.next()){

        ID = query.value(0).toString();

        Number = query.value(1).toString();

        ui->textEdit_2->insertPlainText("ID          товара          "+ID+"          |
Количество"+Number+"\n");

    }

}

```

```

void MainWindow:: removal(){

    QSqlQuery query;

    QString ID;

    QString Number;

```

```

query.exec("delete from Заказ where ID_Товара = "+ui->lineEdit->text());
query.exec("select * from Заказ");
ui->textEdit_2->clear();
while (query.next()){
    ID = query.value(0).toString();
    Number = query.value(1).toString();
    ui->textEdit_2->insertPlainText("ID          товара          "+ID+"          |
Количество"+Number+"\n");
}
}

/*void MainWindow:: slotSendBtn2Clicked(){
    //проверяем ввод пароля и посылаем запрос
    bool ok;

    QString password = QInputDialog::getText(this,"Введите пароль","Ваш
пароль:",QLineEdit::Normal,"",&ok);

    QString selectString;
    if(ok && password=="Qwerty")
    {
        if(ui->insertRB->isChecked()){//добавление строки
            selectString.append("insert into "+ ui->fromTable2->currentText()+ "
values(" + ui->values->text()+")");
        }
        else if(ui->updateRB->isChecked()){//обновление таблицы

```

```

        selectString.append("update  "+ ui->fromTable2->currentText()+" set
"+ui->values->text());

        if(ui->where2->text().length()>0)

            selectString.append("      where      "+ui->where2->text());//условие
обновления

        }

        else{//удаление строки

            selectString.append("delete from "+ui->fromTable2->currentText());

            if(ui->where2->text().length()>0)

                selectString.append("      where      "+ui->where2->text());//условие
удаления

        }

        socket->write(selectString.toUtf8());//текст запроса

    }

}*/

/*void  MainWindow::slotFromTable1IndexChanged(int  index){//изменение
значения столбцов

    if(numberOfRequest>numberOfTables)

    {

        ui->selectCols->clear();

        for(int i=0;i<cols[index].length();i++)

            ui->selectCols->addItem(cols[index][i]);//добавление          списка
столбцов

    }

}*/

```

```

void MainWindow::readyReadUDP()//пришло подтверждение что серв на
СВЯЗИ
{
    QByteArray buffer;
    buffer.resize(socketUDP->pendingDatagramSize());
    QHostAddress sender;
    quint16 senderPort;
    socketUDP-
>readDatagram(buffer.data(),buffer.size(),&sender,&senderPort);
    QString str(socketUDP->readAll());
    if(senderPort==portNumberUDP                                &&
senderPort!=myPortNumberUDP){//если я получил ответ от порта сервера и не от
моего же порта то все норм
        timer->start();//сброс таймера
        return;
    }
}

void MainWindow::timerTimeoutSlot(){
    /*ui->statusLbl->setText("Не    удалось    подключиться    к    серверу
магазина.\n"
                                "Приносим свои извинения");*/
    timer->stop();
    timerSender->stop();

    //ui->connectBtn->setEnabled(true);

```

```

// ui->portNumberUDP->setEnabled(true);

    QMessageBox::information(this,"Ошибка","Вы отключились от сервера
магазина.\n"

                                "Попробуйте перезапустить приложение");

}

void MainWindow::timerSenderTimeoutSlot()
{
    QByteArray datagram="";
    quint16 port=portNumberUDP;
    socketUDP-
>writeDatagram(datagram.data(),datagram.size(),QHostAddress("127.0.0.1"),port);
}

```

ПРИЛОЖЕНИЕ Б
ПРЕЗЕНТАЦИЯ

ТЕМА РАБОТЫ: СИСТЕМА ДЛЯ ОРГАНИЗАЦИИ ИНТЕРНЕТ-ПРОДАЖ

Автор работы: студент группы 14вв1 Кузин Д.О.

Руководитель: к.т.н., доцент Дорошенко И.Н.

- Цель работы - разработка информационной системы (ИС) по учёту заказов на примере кондитерского производства
- Актуальность работы заключается в автоматизации процесса учёта заказов клиентов

2

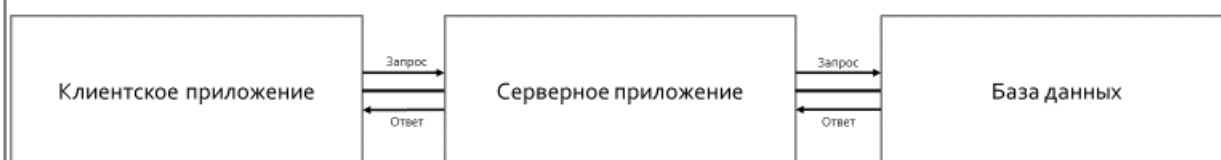
					ПГУ 1.090301.14.001 ПЗ	Лист
						111
Изм.	Лист	№ докум.	Подпись	Дата		

Рост онлайн-продаж в России



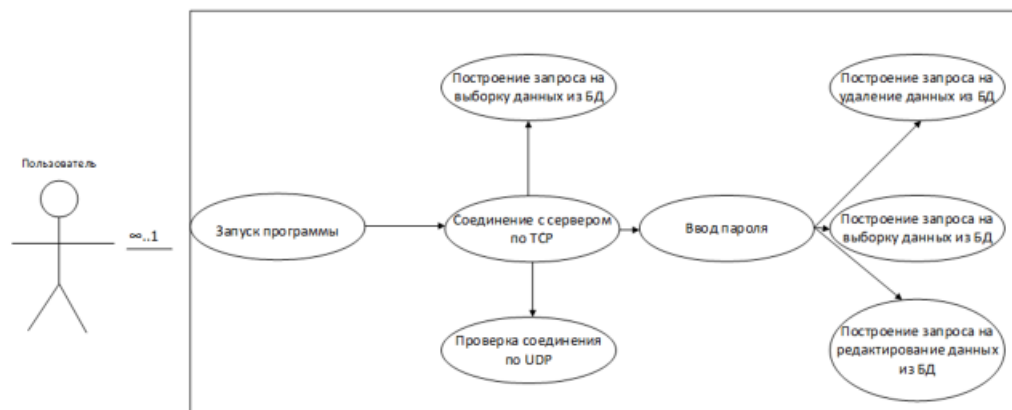
3

Взаимодействие компонентов системы



4

Взаимодействие пользователя с клиентской программой (для сотрудника)



5

Взаимодействие пользователя с клиентской программой (для клиента)



6

Диаграмма состояний клиентского приложения (для сотрудника)

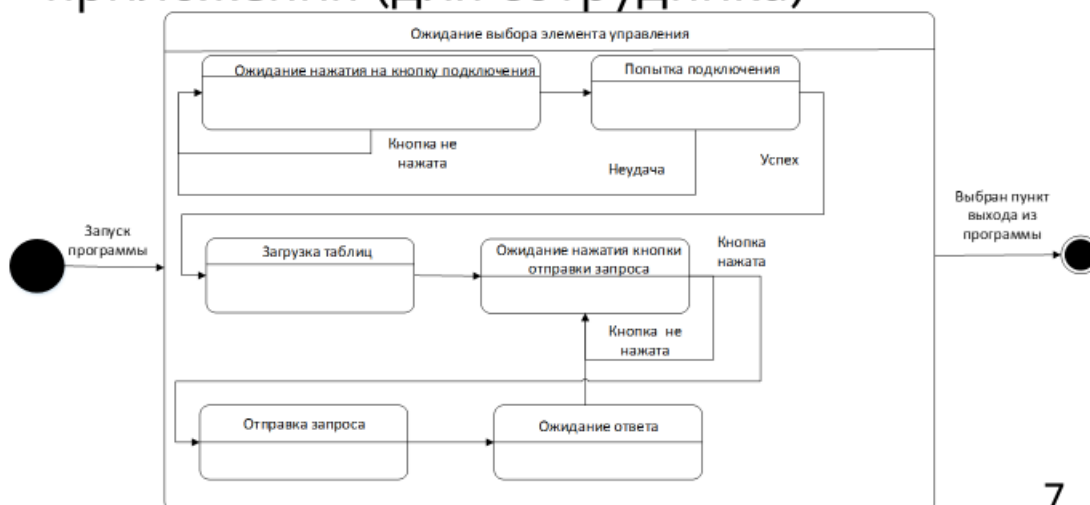
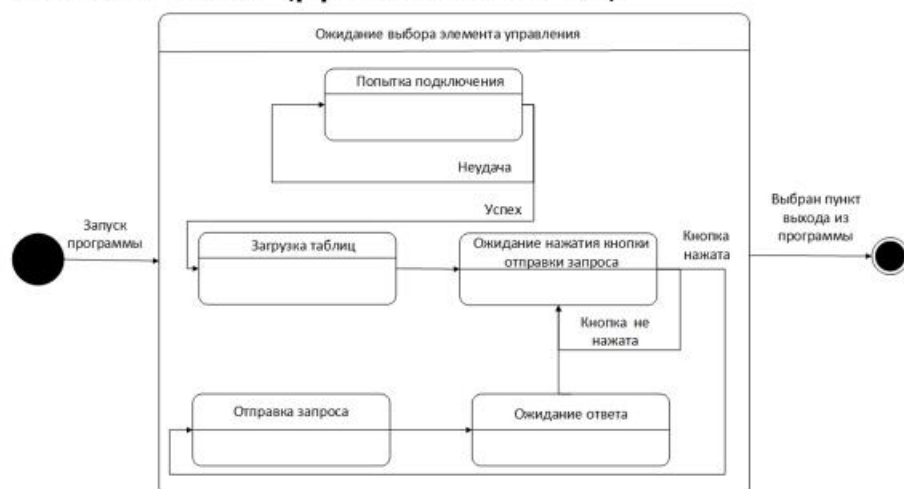
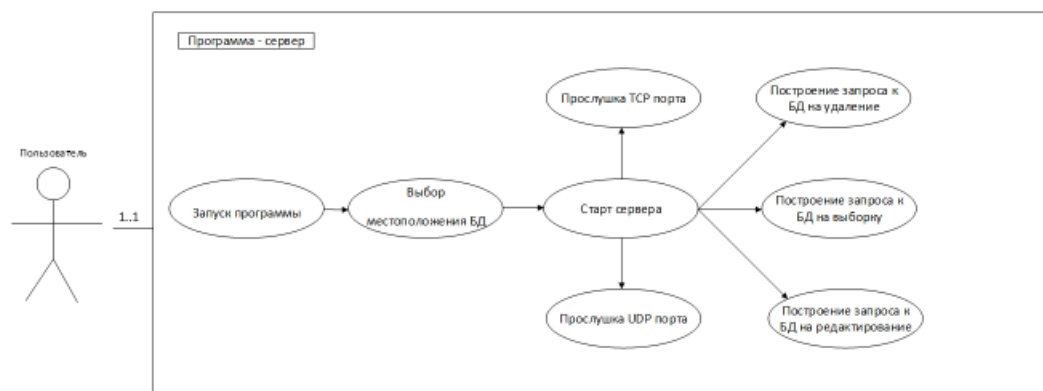


Диаграмма состояний клиентского приложения (для клиента)



Взаимодействие пользователя с серверной программой



9

Диаграмма состояний серверного приложения



10

Интерфейс серверного приложения

11

Интерфейс клиентского приложения (для сотрудника)

ID_Товара	Наименование Товара	Стоимость
1	Батон нарезной	25
2	Батон столбчатый	30
3	Батон с отрубями	30
4	Хлеб 'Корнет'	25
5	Хлеб 'Круглый'	25
6	Хлеб белый	30
7	Сухари ванильные	50
8	Сухари с корицей	55
9	Сухари с корицей	60
10	Торт 'Тирамису'	150
11	Торт 'Черничка'	160
12	Торт 'Ориental'	390
13	Торт 'Наполеон'	250
14	Пирожное 'Картошка'	30
15	Пирожное 'Мороженое'	50
16	Пирожное 'Эклер'	45
17	Чипсы 'Fritolite с сыром'	125
18	Чипсы 'Fritolite со сметаной'	125
19	Чипсы 'Lays с беконом'	60
20	Чипсы 'Lays с сырком'	60

12

Интерфейс клиентского приложения (для клиента)

Добро пожаловать в магазин 'Булочная 58'!

Просмотр товаров магазина 'Булочная 58'

Поиск товаров: Показать товары

ID Товара	Наименование Товара	Стоимость
1	Батон нарезной	20
2	Батон столбчатый	30
3	Батон с опрушкой	30
4	Колье Хрусталь	25
5	Колье Хрусталь	25
6	Колье Белый	30
7	Сладки вафельные	50
8	Сладки с пастилой	50
9	Сладки с орехом	60
10	Торт Прага	150
11	Торт Медовый	140
12	Торт 'On Пармезан'	300
13	Торт Наполеон	200
14	Пирожные 'Крем-шник'	30
15	Пирожные 'Макарусы'	50
16	Пирожные 'Эклер'	40
17	Чашки 'Pudding с сыром'	120
18	Чашки 'Pudding со сметаной'	120
19	Чашки 'Заварка с бисквитом'	60

ID товара:

Количество:

Добавить в корзину

Редактировать Удалить товар

Номер телефона:

Отправить заказ

Корзина

13

- Проект удовлетворяет поставленной цели
- Работа над проектом позволила расширить знания в области клиент-серверной архитектуры, сетевых приложений и баз данных
- Программа может быть улучшена в дальнейшем в соответствии с потребностями пользователей

14

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.14.001 ПЗ

Лист
117