МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно–Уральский государственный университет»
(национальный исследовательский университет)
Высшая школа экономики и управления
Кафедра «Информационные технологии в экономике»

РАБОТА ПРОВЕРЕНА
Рецензент, к.т.н., доцент
_____/А.И.Демченко /
«_____» _____2018 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.т.н., с.н.с.
_____/Б.М.Суховилов /
«____» _____2018 г.

АВТОМАТИЗАЦИИ ПОИСКА ЗАКАЗОВ ДЛЯ СТРОИТЕЛЕЙ И СТРОИТЕЛЬНЫХ ФИРМ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ЮУрГУ – 09.03.03.2018.110. ВКР

Консультант, к.т.н., доцент
_____/ О.И. Галичин /
«_____» _____2018 г.

Руководитель, к.т.н., доцент
_____/ Е.М. Сартасов /
«_____» _____2018 г.

Автор
студент группы ЭиУ-469
_____/ Г.А. Андреев/
«_____» _____2018 г.

Нормоконтролер, доцент
_____/ Е.А. Конова/
«_____» _____2018 г.

Челябинск 2018

АННОТАЦИЯ

Андреев Г.А. Автоматизации поиска заказов для строителей и строительных фирм– Челябинск: ЮУрГУ, ЭиУ-468, 58–стр, 25–ил, табл–13, библиогр. список –6 наим., 6-прил.

Выпускная квалификационная работа выполнена с целью разработки автоматизированной системы поиска заказов.

В работе проанализированы аналогичные существующие системы, выявлены их достоинства и недостатки, обоснована актуальность выбранной темы, сформулирована цель и задачи.

Разработана структура приложение учета заказов, который успешно отлажен и протестирован.

В экономической части работы рассчитаны затраты на разработку и внедрение программного продукта.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящее время существует большое количество онлайн сервисов которые предоставляют множество функций для строителей, такие как поиск заказов, поиск материалов, а также множество других функций.

Задачи дипломного проекта: анализ предметной области, создание и заполнение базы данных, выбор инструментальных средств, разработка приложения для автоматизации строительных процессов подготовки проекта.

Разработано приложение, которое позволяет осуществлять отправку строительного заказа, поиск заказа, поиск материалов, поиск фирм. Приложение имеет универсальный характер и может использоваться пользователями с разными целями.

.

1. ПОСТАНОВКА ЗАДАЧИ

1.1 Техническое задание

Одной из целей, преследуемых при создании приложения – это минимизировать функции как заказчиков, так и исполнителей, а также предоставить им информацию которая может помочь как в процессе продумывания проекта, так и во время основных работ.

Требуется разработать приложение.

Основные требования к приложению:

- авторизованный доступ;
- дружественный интерфейс;
- отправка заказа, просмотр заказов, поиск материалов, поиск фирм;
- обеспечение рассылки пользователям;
- работа без подключения к интернету;

1.2 Анализ существующих разработок и выбор стратегии автоматизации

На данный момент существует большое количество онлайн сервисов, обеспечивающих поддержу пользователей в строительной сфере, но все они требуют постоянного подключения к интернету.

1.2.1 www.stroikaural.ru

www.stroikaural.ru – специализированный сетевой ресурс, посвященный теме строительства и ремонта, работает с 2008 года. www.stroikaural.ru создан для информационной поддержки и продвижения строительного бизнеса уральского региона. Информация портала адресована как профессионалам строительного бизнеса, так и частным лицам, заинтересованным в получении актуальной информации в области ремонта, отделки, строительства.

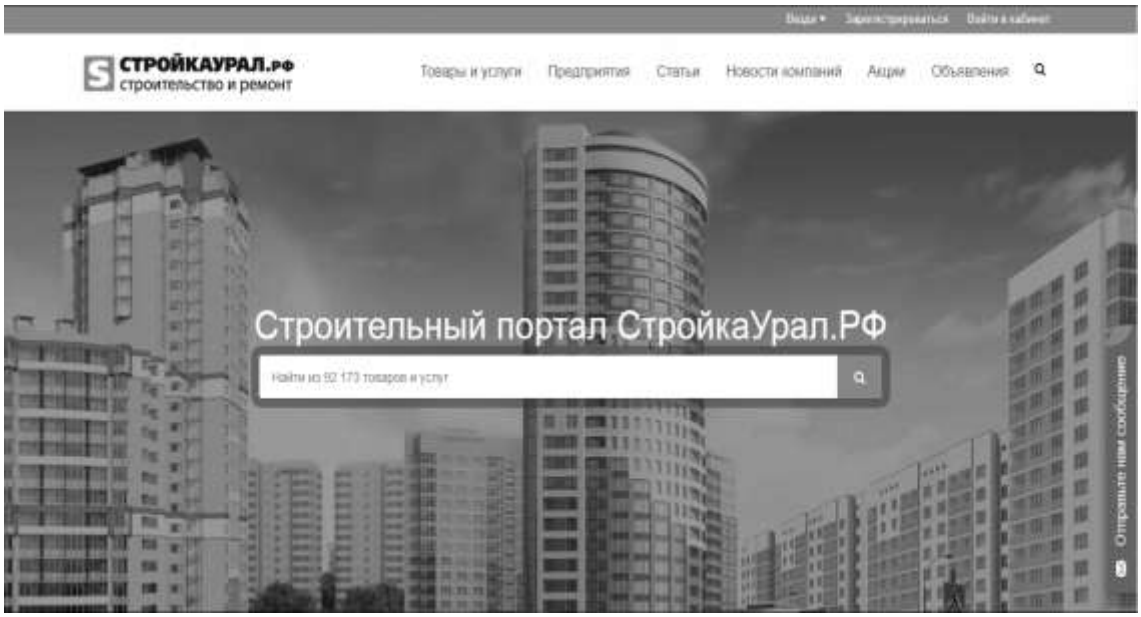

Рисунок 1.1 – Главная страница сайта www.stroikaural.ru

Данный сервис больше нацелен на поддержу строительных фирм чем заказчиков.

Основные плюсы:

- удобный Поиск фирм;
- справочная информация;
- список товаров и услуг.

Минусы:

- поиск заказов;
- взаимодействие с картой;
- требуется постоянное подключение к интернету.

1.2.2 propetrovich.ru

Главная страница сайта (propetrovich.ru) приведена на рисунке 1.2

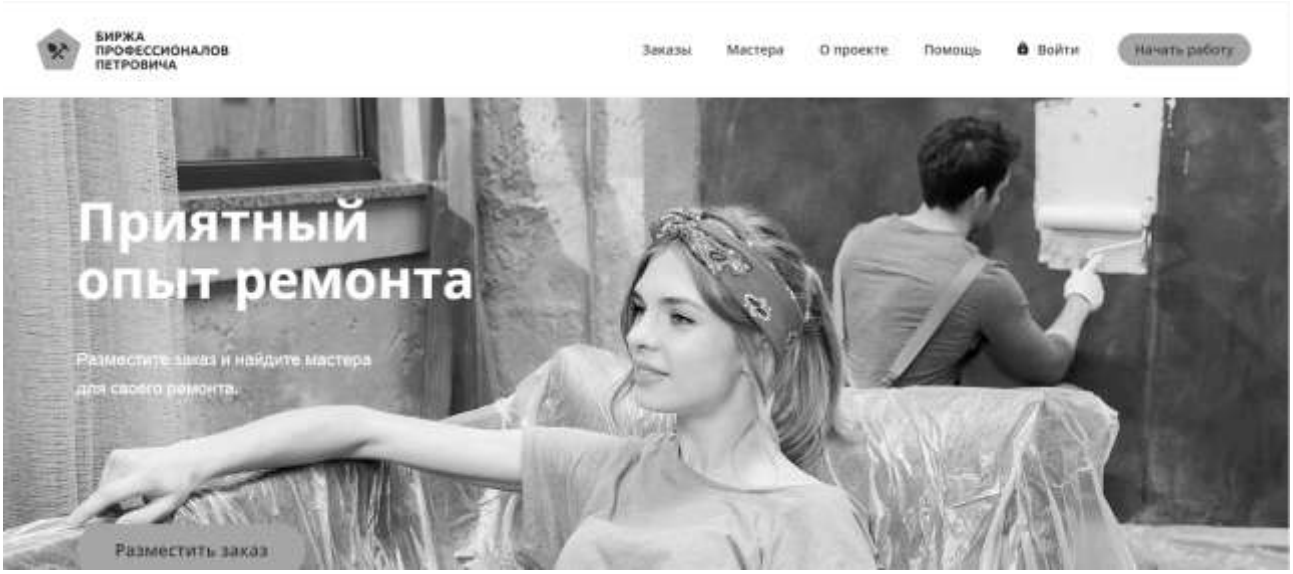

Рисунок 1.2 – Главная страница сайта propetrovich.ru

Компания Петрович уже 20 лет объединяет вокруг себя тысячи прорабов, строителей, бригадиров, мастеров широкого профиля и других специалистов в области ремонта и строительства.

Плюсы:

- предоставление интерфейса подачи заявки для заказчиков;
- интерфейс поиска заказов для строителей.

Минусы:

- взаимодействие с картой;
- нацеленность на большие города;
- требуется постоянное подключение к интернету

1.2.3 remontnik.ru

Главная страница сайта(remontnik.ru) приведена на рисунке 1.3

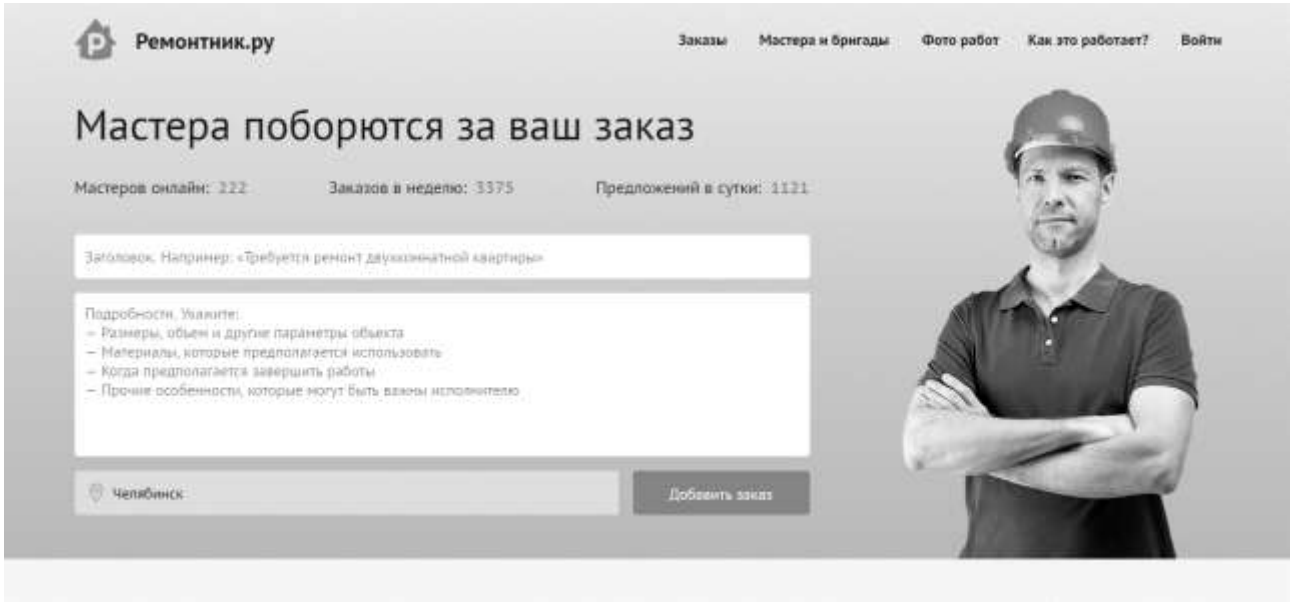

Рисунок 1.3 – Главная страница сайта(remontnik.ru)

Сайт «Ремонтник.ру. В данной системе в центре поставили не подрядчика и его объявление, а заказчика и его заказ. Сайт построен вокруг конкретных заказов, которые любой пользователь может добавить бесплатно за пару кликов: «построить дом», «положить плитку», «заменить кран».

Плюсы:

- предоставление интерфейса подачи заявки для заказчиков
- интерфейс поиска заказов для строителей

Минусы:

- требуется постоянное подключение к интернету
- отсутствует поиск материалов

1.3 Выбор программных средств

При рассмотрении среды разработки выбор сделан в пользу MS Visual Studio.

Visual Studio включает в себя. Встроенный отладчик может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и

дизайнер схемы базы данных. Основное качество – это средства автоматизации, которые существенно сокращают объем ручного кодирования, создавая заготовку и программный макет[1].

В качестве языка программирования для написания клиентского приложения использован язык C# на платформе Net.

Основой языка является, несомненно, C++. Однако, некоторые детали были позаимствованы из языка Java, как пространства имен etc. Еще одно перекочевавшее из Java достоинство, это система автоматического управления памятью. На систему возложены функции выделения памяти для различных классов, и ее освобождения после прекращения использования классов. Реализация интерфейсов в C# идентична реализации в языке Java[5].

Следующие возможности были взяты из старых языков программирования, например, из Паскаля. Первая – четкая типизация переменных. При описании переменной необходимо указать ее тип, чего не было в C++. Вторая – автоматическая инициализация переменных. Вот этой возможности и нет в Java[5].

Технологии COM+ и Windows API полностью поддерживаются C#. Так как язык C# сам не содержит библиотек классов, то в нем появилась возможность использования библиотек других систем программирования, разработанных фирмой Microsoft, как, например, Visual Basic[2].

Именно поэтому C# выбран языком для разработки.

В качестве системы управления базами данных предлагается использовать Microsoft Access.

Основная причина выбора данной системы стала ее переносимость, таким образом возможности обновление БД.

В отличие от других настольных СУБД, Access хранит все данные в одном файле, хотя и распределяет их по разным таблицам, как и положено реляционной СУБД[6].

Особенности MS Access, отличающиеся от представления об «идеальной» реляционной СУБД.

Создание многопользовательской БД Access и получение одновременного доступа нескольких пользователей к общей базе данных возможно в локальной одно ранговой сети или в сети с файловым сервером. Сеть обеспечивает аппаратную и программную поддержку обмена данными между компьютерами. Access следит за разграничением доступа разных пользователей к БД и обеспечивает защиту данных. При одновременной работе. Так как Access не является клиент серверной СУБД, возможности его по обеспечению многопользовательской работы несколько ограничены. Обычно для доступа к данным по сети с нескольких рабочих станций, файл БД Access (с расширением *.mdb) выкладывается на файловый сервер. При этом обработка данных ведется в основном на клиенте – там, где запущено приложение, в силу принципов организации файловых СУБД. Этот фактор ограничивает использование Access для обеспечения работы множества пользователей (более 15–20) и при большом количестве данных в таблицах, так как многократно возрастает нагрузка не сеть[6].

В плане поддержки целостности данных Access отвечает только моделям БД небольшой и средней сложности. В нем отсутствуют такие средства как триггеры и хранимые процедуры, что заставляет разработчиков возлагать поддержание бизнес логики БД на клиентскую программу[6].

В отношении защиты информации и разграничения доступа Access не имеет надежных стандартных средств. В стандартные способы защиты входит защита с использованием пароля БД и защита с использованием пароля пользователя. Снятие такой защиты не представляет сложности для специалиста[6].

Однако, при известных недостатках MS Access обладает большим количеством преимуществ по сравнению с системами подобного класса[6].

В первую очередь можно отметить распространенность, которая обусловлена тем, что Access является продуктом компании Microsoft, программное обеспечение и операционные системы которой использует большая часть пользователей персональных компьютеров. MS Access полностью совместим с операцион-

ной системой Windows, постоянно обновляется производителем, поддерживает множество языков[6].

В целом MS Access предоставляет большое количество возможностей за сравнительно небольшую стоимость. Также необходимо отметить ориентированность на пользователя с разной профессиональной подготовкой, что выражается в наличии большого количества вспомогательных средств. Эти средства облегчают проектирование, создание БД и выборку данных из нее[6].

Получается, что Access, обладая всеми чертами СУБД, предоставляет и дополнительные возможности. Это не только гибкая и простая в использовании СУБД, но и система для разработки работающих с базами данных приложений и именно поэтому выбрана эта система.

Вывод по разделу один

В разделе один представлены примеры существующих проектов, определены их возможности, выявлены преимущества и недостатки. Представлен выбор инструментальных средств для разработки.

2. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

2.1 Проектирование и применение базы данных

Для хранения данных, используемых в клиентском приложении, применяется база данных Access.

Проектирование БД – одна из наиболее сложных и ответственных задач, связанных с созданием информационной системы. В результате решения этой задачи должны быть определены содержание БД, эффективный для всех её будущих пользователей способ организации данных и инструментальные средства управления данными[6].

В крупных системах проектирование БД требует особой тщательности, поскольку цена допущенных на этой стадии просчётов и ошибок особенно велика. Некоторые ошибки проектирования можно скорректировать позже в процессе эксплуатации с помощью средств реструктуризации и реорганизации БД, но такие операции являются весьма трудоемкими и дорогостоящими.

База данных – это совокупность структурированных и взаимосвязанных данных и методов, обеспечивающих добавление, выборку и отображение данных[6].

Схема разработанной базы данных представлена на рисунке 2.1.1

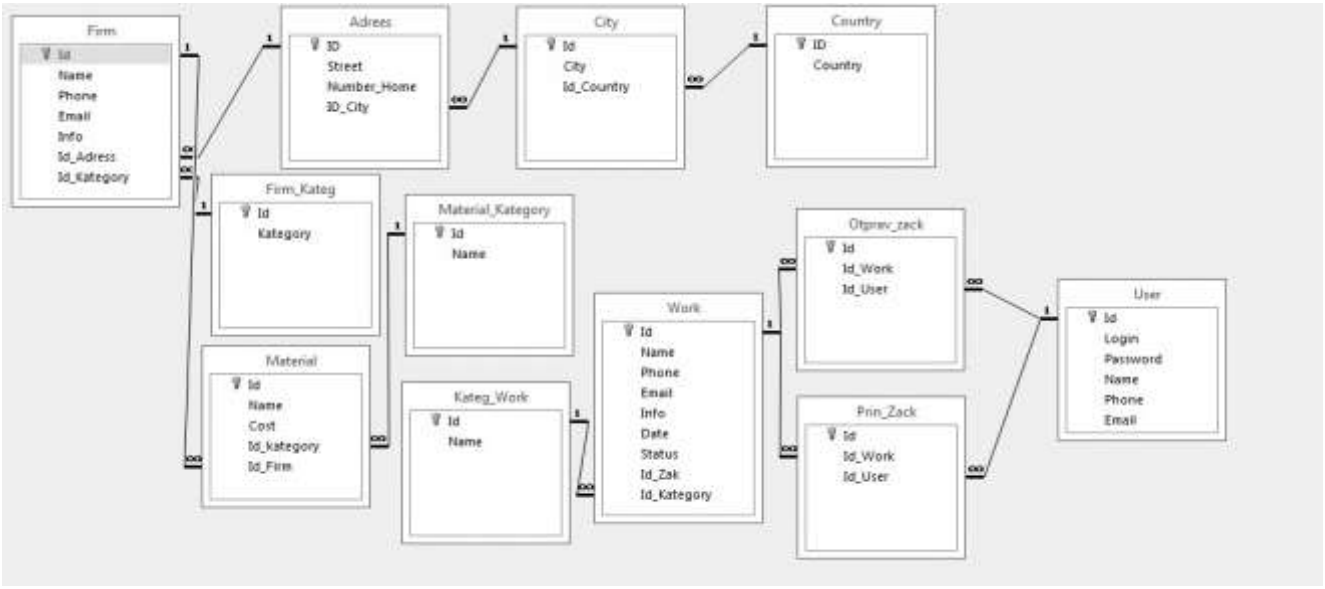

Рис 2.1.1– Схема базы данных

2.2 Описание таблиц базы данных

Для реализации проекта разработана база данных, она состоит из 12-ти взаимосвязанных таблиц. В таблицах 2.1 – 2.12 приведены описания полей таблиц базы данных.

Таблица Material содержит в себе перечень материалов для строительных работ.

Описание полей таблицы Material приведено в таблице 2.1.

Таблица 2.1 – Описание полей таблицы Material

Название поля	**Описание**	**Тип поля**
Id	Уникальный код товара	Счетчик
Name	Название Товара	Текстовое
Cost	Цена	Числовое
Id_Kategory	Код Категории	Числовое
Id_Firm	Код Фирмы	Числовое

Таблица Firm содержит в себе информация о фирмах и категориях их работ.

Описание полей таблицы Firm приведено в таблице 2.2

Таблица 2.2 – Описание полей таблицы Firm

Название поля	Описание	Тип поля
Id	Уникальный код	Счетчик
Name	Название фирмы	Текстовое
Phone	Телефон фирмы	Числовое
Info	Информация о фирме	Текстовое
Email	E-mail Адрес	Текстовое
Id_Addres	Код Адреса фирмы	Числовое
Id_Kategory	Код Категории	Числовое

Таблица User содержит в себе информация о фирмах и категориях их работ.

Описание полей таблицы User приведено в таблице 2.3

Таблица 2.3 – Описание полей таблицы User

Название поля	Описание	Тип поля
Id	Уникальный код пользователя	Счетчик
Login	Логин	Текстовое
Password	Пароль	Текстовое
Name	ФИО пользователя	Текстовое
Phone	Телефон	Текстовое
Email	E-mail Адрес	Текстовое

Таблица Work содержит в себе информацию о доступных заявках для работников.

Описание полей таблицы Work приведено в таблице 2.4.

Таблица 2.4 – Описание полей таблицы Work

Название поля	Описание	Тип поля
Id	Уникальный код	Счетчик
Name	ФИО заказчика	Текстовое
Phone	Телефон заказчика	Числовое
Email	E-mail Адрес заказчика	Текстовое
Info	Информация о работе	Текстовое
Date	Дата заявки	Дата/время
Id_Zak	Код заказчика	Числовое
Id_Kategory	Код Категории	Числовое

Таблица Country содержит в себе список стран, требующихся для таблицы Firm.

Описание полей таблицы Country приведено в таблице 2.5

Таблица 2.5 – Описание полей таблицы Country

Название поля	Описание	Тип поля
Id	Уникальный код Страны	Идентификатор
Country	Наименование страны	Текстовое

Таблица City содержит в себе список городов, требующихся для таблицы Firm. Описание полей таблицы City приведено в таблице 2.6

Таблица 2.6 – Описание полей таблицы City

Название поля	Описание	Тип поля
Id	Уникальный код Города	Идентификатор
City	Наименование города	Текстовое
Id_Country	Id Страны	Числовое

Таблица Adrees содержит в себе список адресов. Описание полей таблицы БД Adrees приведено в таблице 2.7

Таблица 2.7 – Описание полей таблицы Adrees

Название поля	Описание	Тип поля
Id	Уникальный код	Идентификатор
Street	Наименование Улицы	Текстовое
Number_Home	Номер дома	Текстовое
ID_City	Id Города	Числовое

Таблица Firm_Kateg содержит в себе список Категорий фирм, требующихся для таблицы Firm.

Описание полей таблицы Firm_Kateg приведено в таблице 2.8

Таблица 2.8 – Описание полей таблицы Firm_Kateg

Название поля	Описание	Тип поля
Id	Уникальный код	Идентификатор
Country	Наименование Категории	Текстовое

Таблица Material_Kategory содержит в себе список категорий материалов, требующихся для таблицы Material.

Описание полей таблицы Material_Kategory приведено в таблице 2.9

Таблица 2.9 – Описание полей таблицы Material_Kategory

Название поля	Описание	Тип поля
Id	Уникальный код	Идентификатор
Country	Наименование Категории	Текстовое

Таблица Kateg_Work содержит в себе список Категорий Работ, требующихся для таблицы Work.

Описание полей таблицы Kateg_Work приведено в таблице 2.10

Таблица 2.10 – Описание полей таблицы Kateg_Work

Название поля	Описание	Тип поля
Id	Уникальный код	Идентификатор
Country	Наименование Категории	Текстовое

Таблица Otprav_Zack содержит в себе список отправленных заказов определённым пользователем.

Описание полей таблицы Otprav_Zack приведено в таблице 2.11

Таблица 2.11 – Описание полей таблицы Otprav_Zack

Название поля	Описание	Тип поля
Id	Уникальный код	Идентификатор
Id_User	Код пользователя	Числовое
Id_Work	Код работы	Числовое

Таблица Prin_Zack содержит в себе список принятых заказов определённым пользователем.

Описание полей таблицы Prin_Zack приведено в таблице 2.12

Таблица 2.12 – Описание полей таблицы Prin_Zack

Название поля	Описание	Тип поля
Id	Уникальный код	Идентификатор
Id_User	Код пользователя	Числовое
Id_Work	Код работы	Числовое

2.3 Структура информационной системы

Диаграмма прецедентов представлена на рисунке 2.3.1. На диаграмме показано взаимодействие пользователей и администратора с приложением.

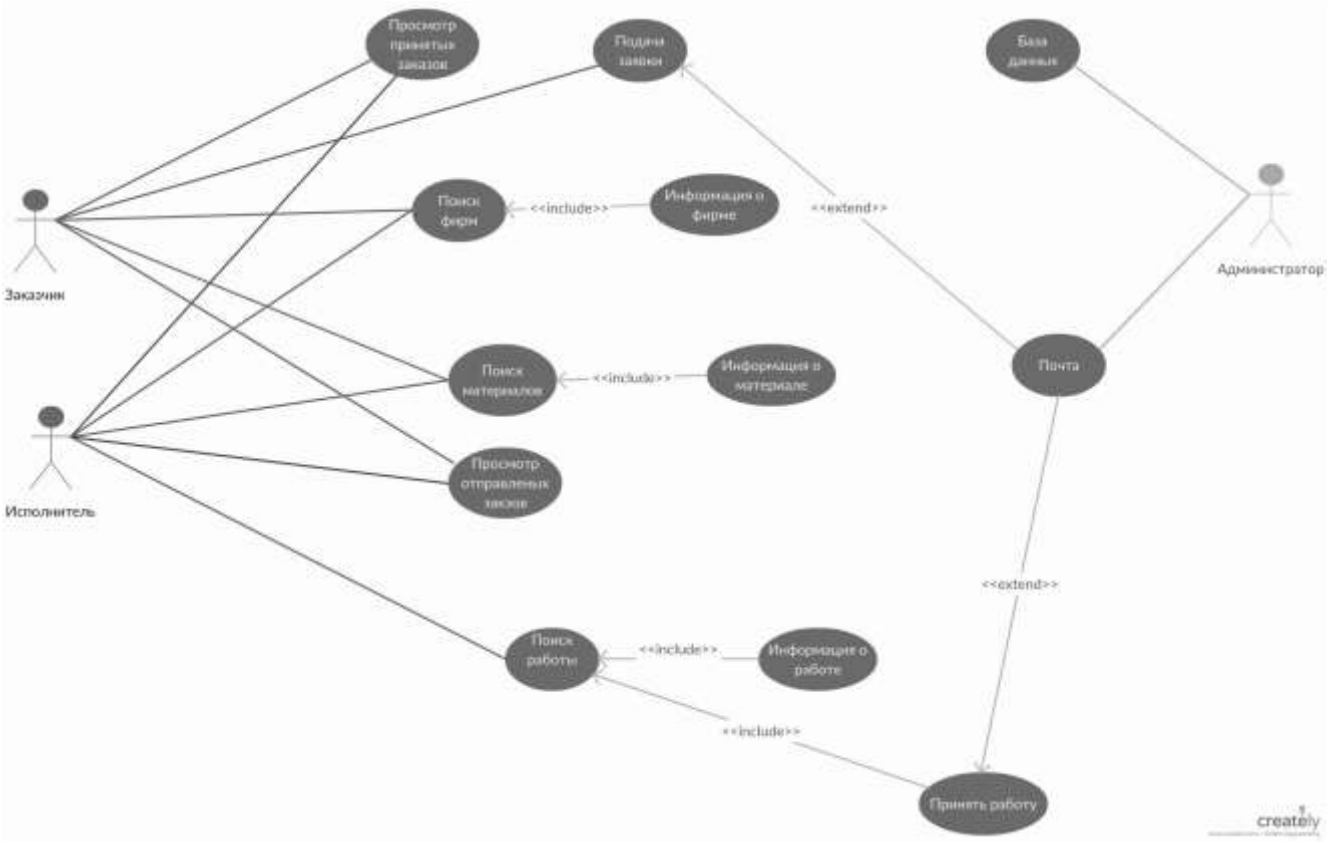

Рисунок 2.3.1 – Диаграмма прецедентов

На диаграмме показано последовательность основных 2-х процессов приложения.

1. Отправка заявки на работу.
2. Принятие заявки.

Диаграмма последовательности представлена на рисунке 2.3.2

.

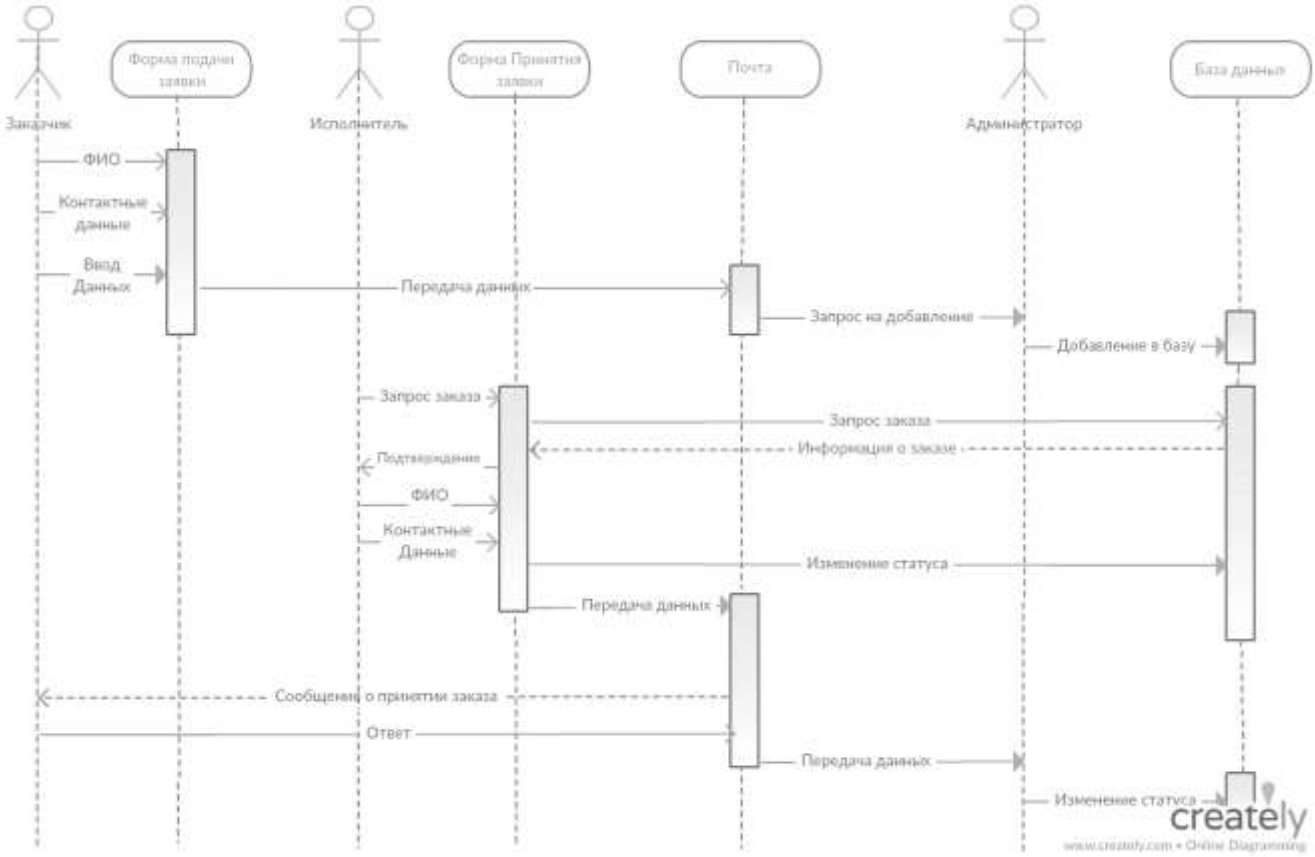

Рисунок 2.3.2 – Диаграмма последовательности

На диаграмме потоков данных показано передача и хранение данных в приложении.

Диаграмма потоков данных представлена на рисунке 2.3.3.

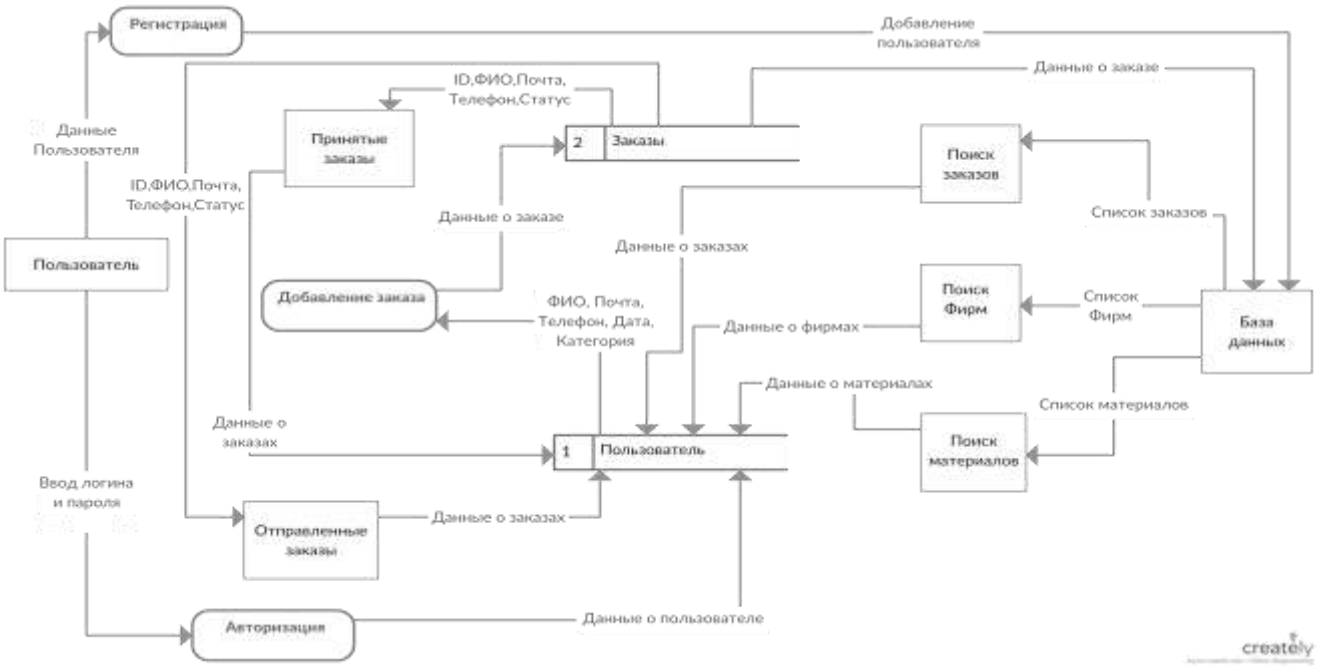

Рисунок 2.3.3 – Диаграмма потоков данных

Из всех прошлых диаграмм сформирована информационная модель программы, на которой показано взаимодействие разных пользователей с приложением.

Информационная модель программы представлена на рисунке 2.3.4

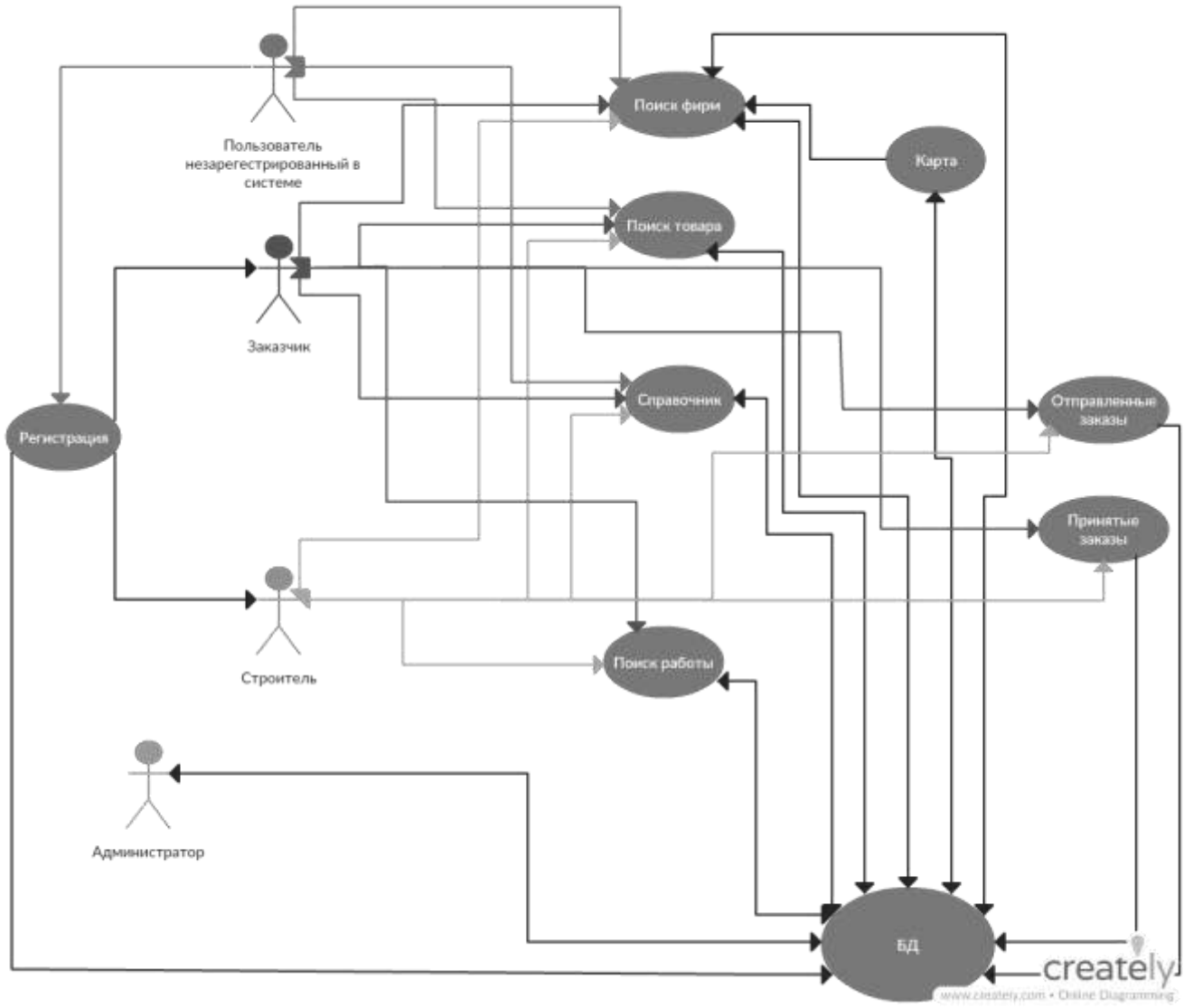

Рисунок 2.3.4 – Информационная модель

2.4 Описание разработки приложения

Файлы, которые содержит в себе проект «StroiChel74», представлены на рисунке 2.4.1

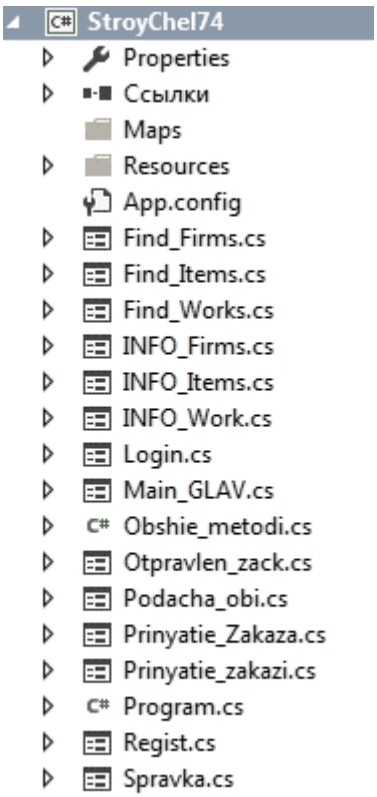

Рисунок 2.4.1 – файлы проекта «StroiChel74»

Проект содержит в себе 14 форм, 1 ссылку на ресурс обеспечивавший подключение к карте, 1 файл исходного кода с расширением cs, который включает в себя общий метод отправки письма на почту.

Файл Obshie_metodi.cs содержит в себе метод для отправки письма на указанную почту[3]

Код файла Obshie_metodi.cs Представлен в ПРИЛОЖЕНИИ Ж

Файл YandexAPI.dll обеспечивает подключение к сервису Яндекс.Карты и взаимодействие с ним[4].

При запуске программы загружается форма «Вход».

Код формы «Вход» представлен в ПРИЛОЖЕНИИ А.

Форма «Вход» представлена на рисунке 2.4.2.

Рисунок 2.4.2 – Форма «Вход»

При нажатии на кнопку «Регистрация» будет вызвана форма Регистрация, которая добавляет пользователя, заполнившего обязательные поля в базу и отправляет письмо на почту для оповещения администратора.

Форма «Регистрация» представлена на рисунке 2.4.3.

Рисунок 2.4.3 – Форма «Регистрация»

Код формы «Регистрация» представлен в ПРИЛОЖЕНИИ Б

При нажатии на надпись «Войти без регистрации» на форме «Вход» Пользователь будет направлен на основную форму «StroiChel74», на которой будут за-

блокированы кнопки перехода на формы в которой содержится информация о заказах и отправки заказа, рисунок 2.4.4.

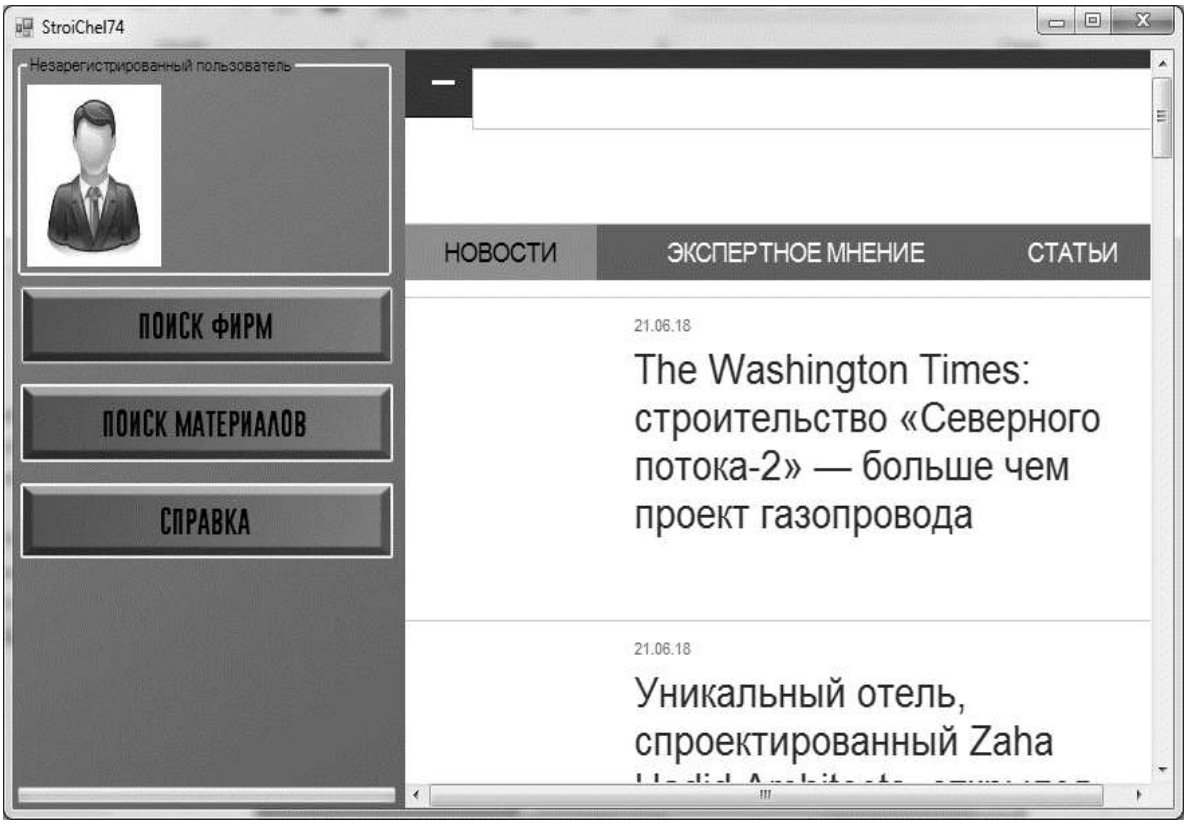

Рисунок 2.4.4 – Форма «StroiChel74» (вход без логина)

При вводе логина и пароля на форме «Вход» производиться поиск пользователя с заданным логином и заданным для него паролем с помощью запроса, после данной процедуры будет доступен полный функционал приложения[5].

В Форму «StroiChel74» передаются два параметра: Логин и Id пользователя; для дальнейших взаимодействий с программой.

Форма «StroiChel74» с полным функционалом представлена на рисунке 2.4.5.

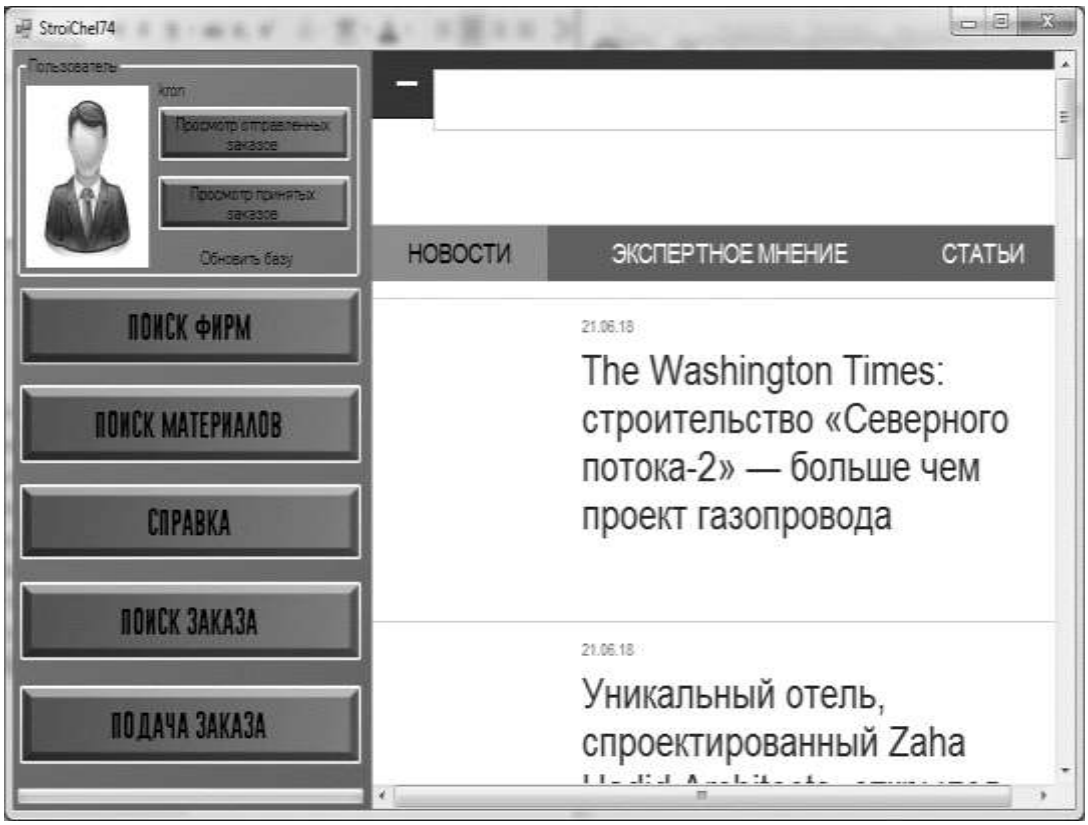

Рисунок 2.4.5 – Форма «StroiChel74»

На форме «StroiChel74» идет разделение на три секции.

Секция Пользователя, представленная на рисунке 2.4.6, содержит в себе Логин пользователя и кнопки перехода на формы: «Принятые заказы» – содержит список заказов, которые принял пользователь; «Отправленные заказы» – содержит список заказов, которые оставил пользователь.

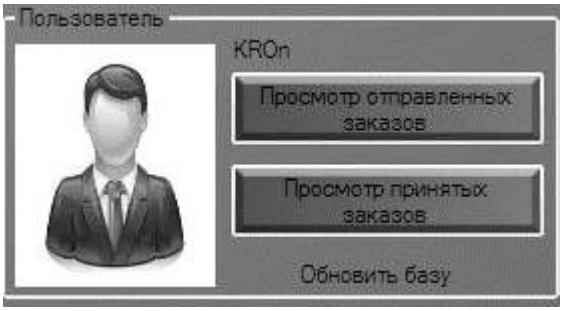

Рисунок 2.4.6 – Секция пользователя

Секция Веб-интерфейса отображает новости в сфере строительства загружая через интернет информацию по указанной ссылке.

Секция Основного меню представленная на рисунке 2.4.7 содержит в себе основные кнопки для перехода.

Рисунок 2.4.7 – Секция основного меню

На форме «Поиск фирм» представлена таблица со списком фирм строительной сферы. С помощью меню выбора категории фирм можно отсортировать таблицу для более удобного поиска требуемых фирм, список загружается из базы.

Код формы «Поиск фирм» представлен в ПРИЛОЖЕНИИ В.

Форма «Поиск фирм» представлена на Рисунке 2.4.8.

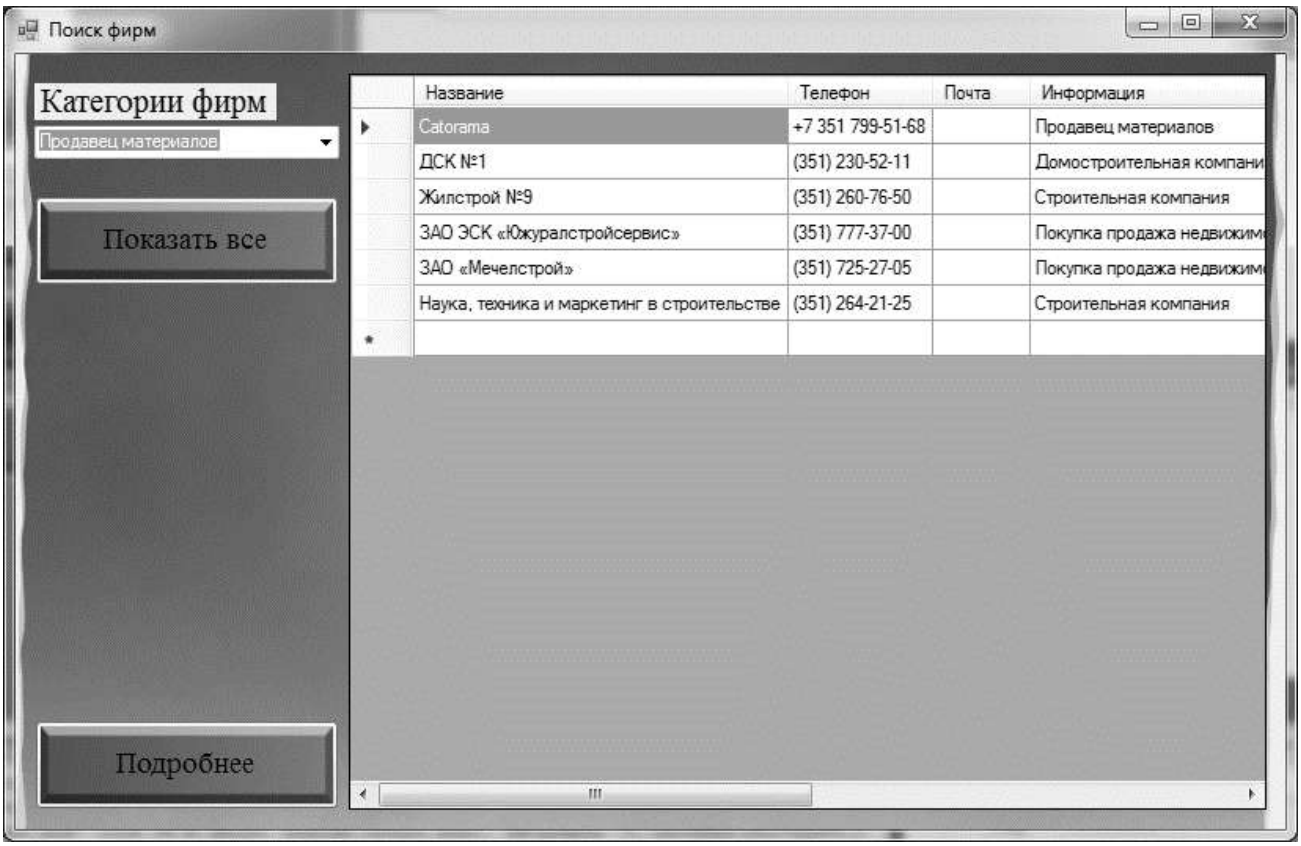

Рисунок 2.4.8 – Форма «Поиск фирм»

Нажав на кнопку «Подробнее» запуститься форма «Информация о фирме» в которую передаются данные с выделенной строки для более удобного просмотра.

С помощью подключаемого файла YandexAPI.dll происходит показ карты. Принцип работы YandexAPI.dll: получая сведения о адресе происходит подключение к сервису Яндекс карт после этого выводиться статическое изображение с обозначением на карте[4].

Форма «Информация о фирме» представлена на рисунке 2.4.9.

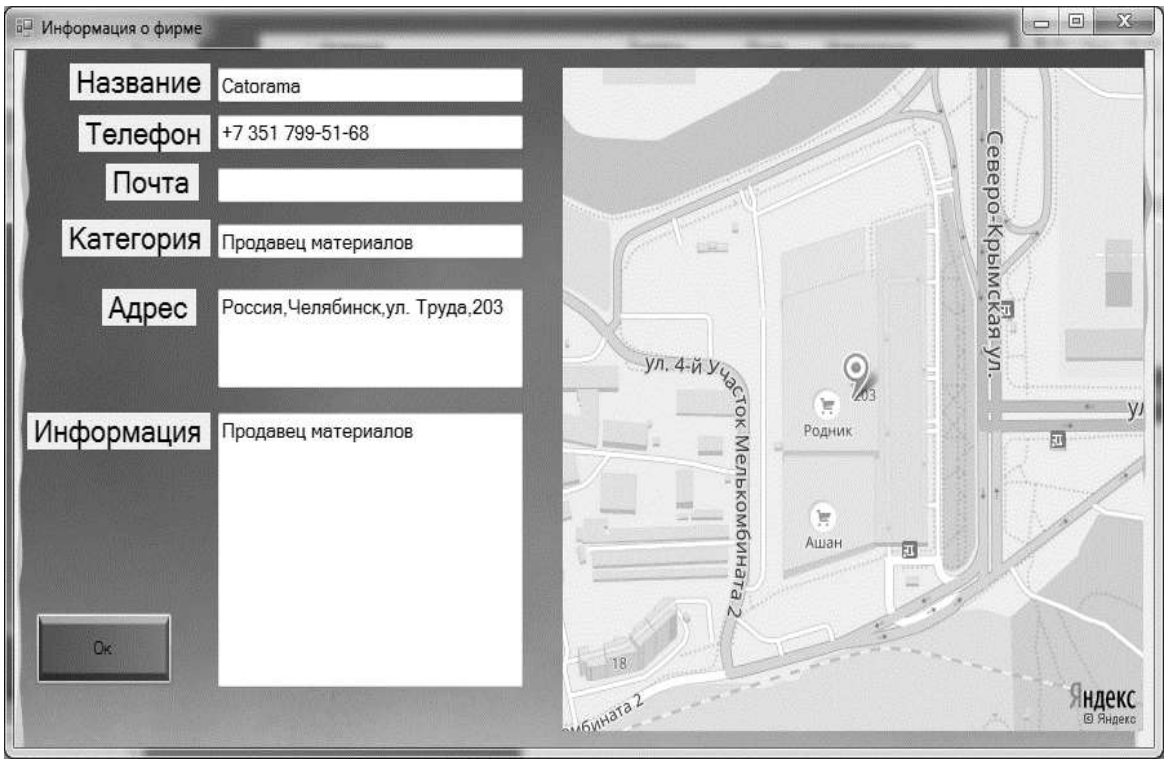

Рисунок – 2.4.9 – Форма «Информация о фирме»

Форма «Поиск материалов» Содержит в себе список материалов, поиск по таблице, фильтры которые позволяют сортировать записи по категориям и стоимости, применяемые для таблиц. Поиск и фильтрация по таблице происходит путем запросов к таблице на форме.

Форма «Поиск материалов» представлена на рисунке 2.4.10.

Код формы «Поиск материалов» Представлен в ПРИЛОЖЕНИИ Г.

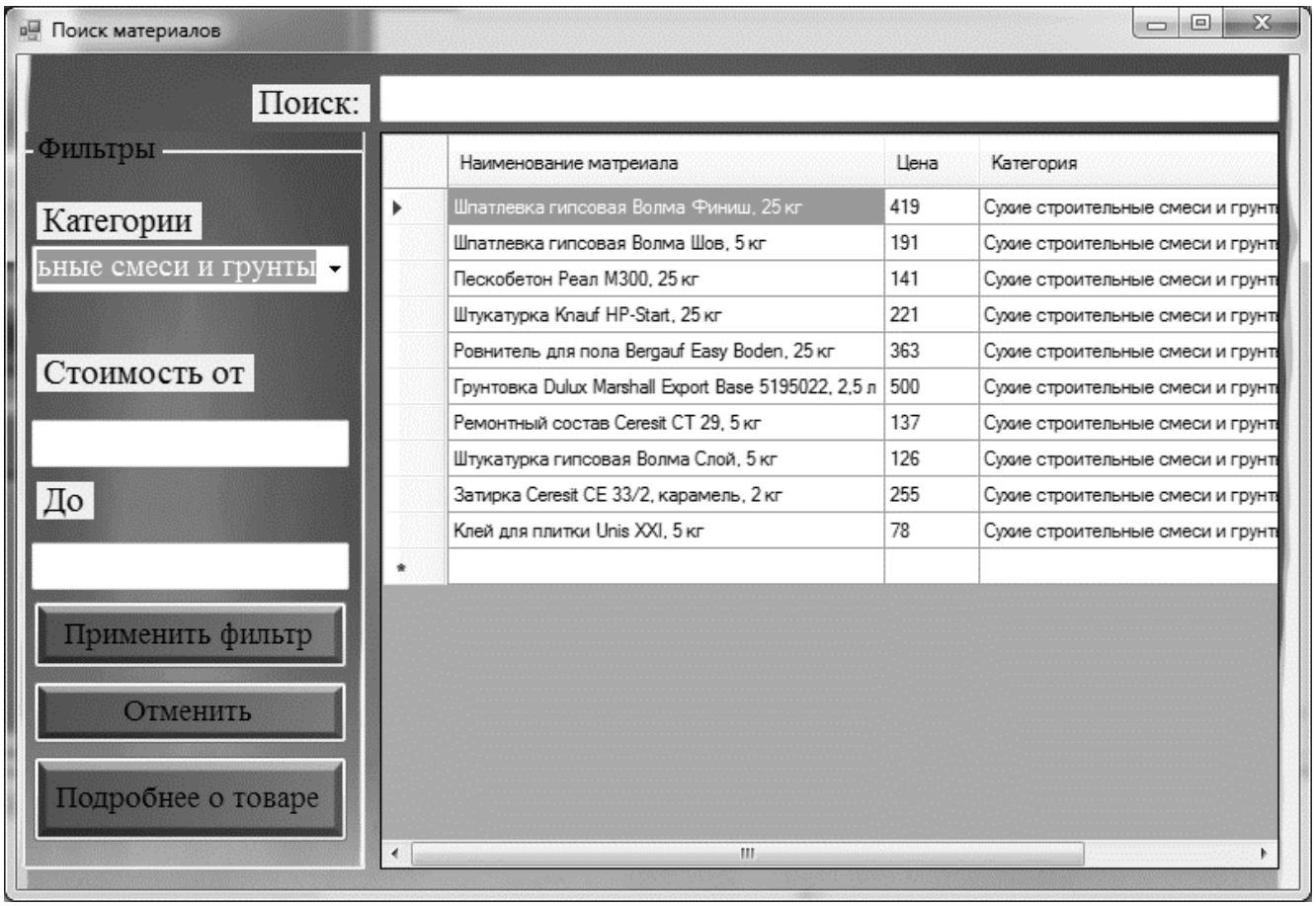

Рисунок 2.4.10 Форма – «Поиск материалов»

По кнопке «Подробнее о товаре» соответственно вызывается форма в которой содержится информация о товаре, для более удобного просмотра информации о товаре.

Форма отправки заявки принимает информацию о логине и Id пользователя, при заполнении всех полей и нажатию кнопки отправить происходит отправка информации для администратора, после проверки всех данных администратор добавляет заказ в базу данных.

Форма отправки заявки представлена на рисунке 2.4.11

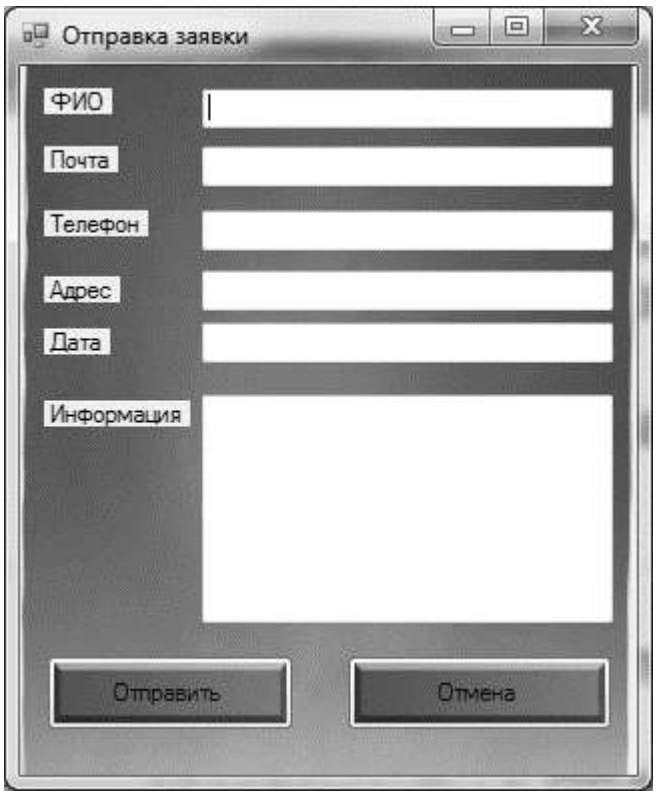

Рисунок 2.4.11 – Форма отправки заявки

Форма «Поиск заказа» выводит список всех заказов. Фильтрация по таблице производиться по пунктам категории и статус, в зависимости от статуса заказа строка в таблице выделяется определенными понятными для всех цветами:

Статус свободен зеленным, статус в обработке желтым и соответственно закрыт красным, при нажатии кнопки «Подробнее о работе» происходит открытие формы «Информация о работе».

Форма «Поиск заказа» представлена на рисунке 2.4.12

Код формы «Поиск заказа» представлен в ПРИЛОЖЕНИИ Д

Форма «Информация о работе» представлена на рисунке 2.4.13

На форме «Информация о работе» выводиться информация о заказе в более удобном виде.

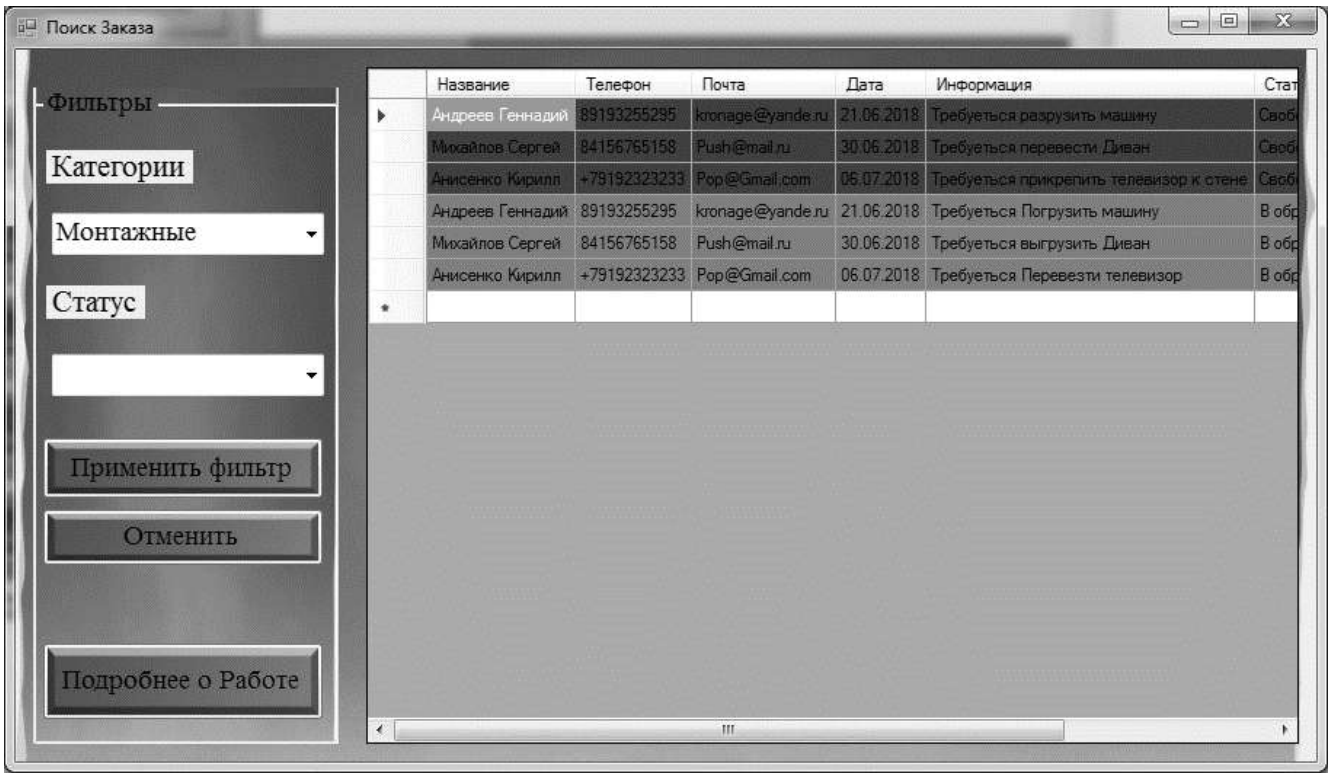

Рисунок 2.4.12 – Форма «Поиск заказа»

Рисунок 2.4.13 Форма – «Информация о работе»

При нажатии на кнопку принять заказ загружается форма «Принятие заказа» рисунок 2.4.14

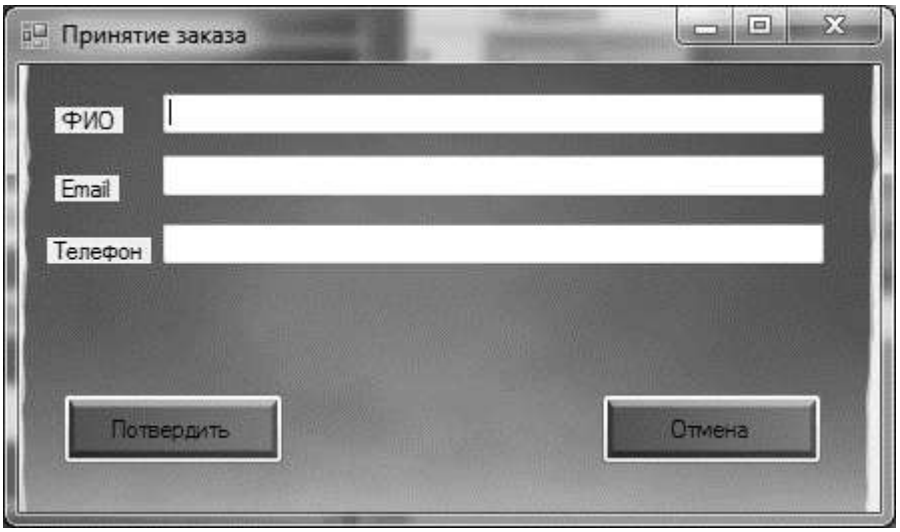

Рисунок 2.4.14 – форма «Принятие заказа»

При заполнении всех данных и подтверждении заказа происходит отправка письма заказчику и администратору.

Код формы «Принятие заказа» представлен в ПРИЛОЖЕНИИ Е.

На форме «Отправленные заказы» выводиться таблица с данными о заказах которые отправил пользователь авторизовавшийся в системе, форма представлена на рисунке 2.4.15.

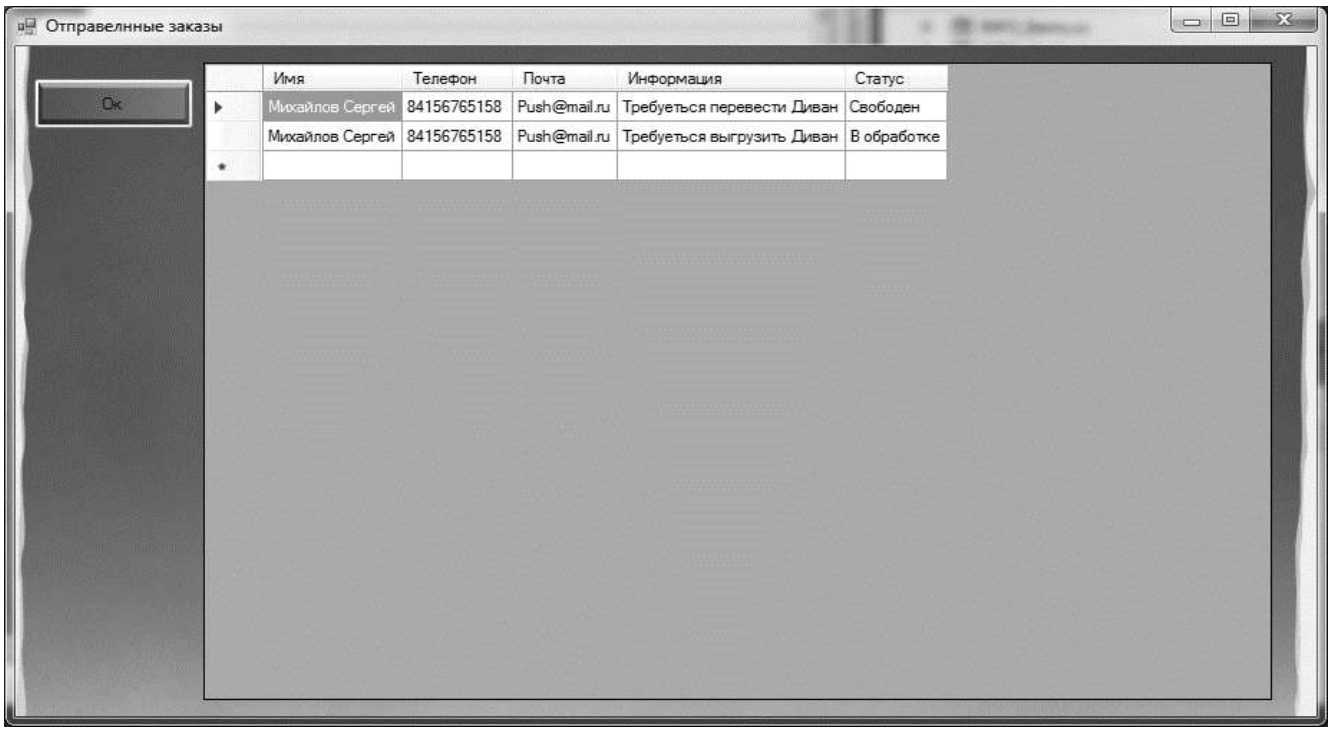

Рисунок 2.4.15 – Форма «Отправленные заказы»

На форме «Принятые заказы» выводиться таблица с данными о заказах которые принял пользователь авторизовавшийся в системе.

Форма «Принятые заказы» представлена на рисунке 2.4.16.

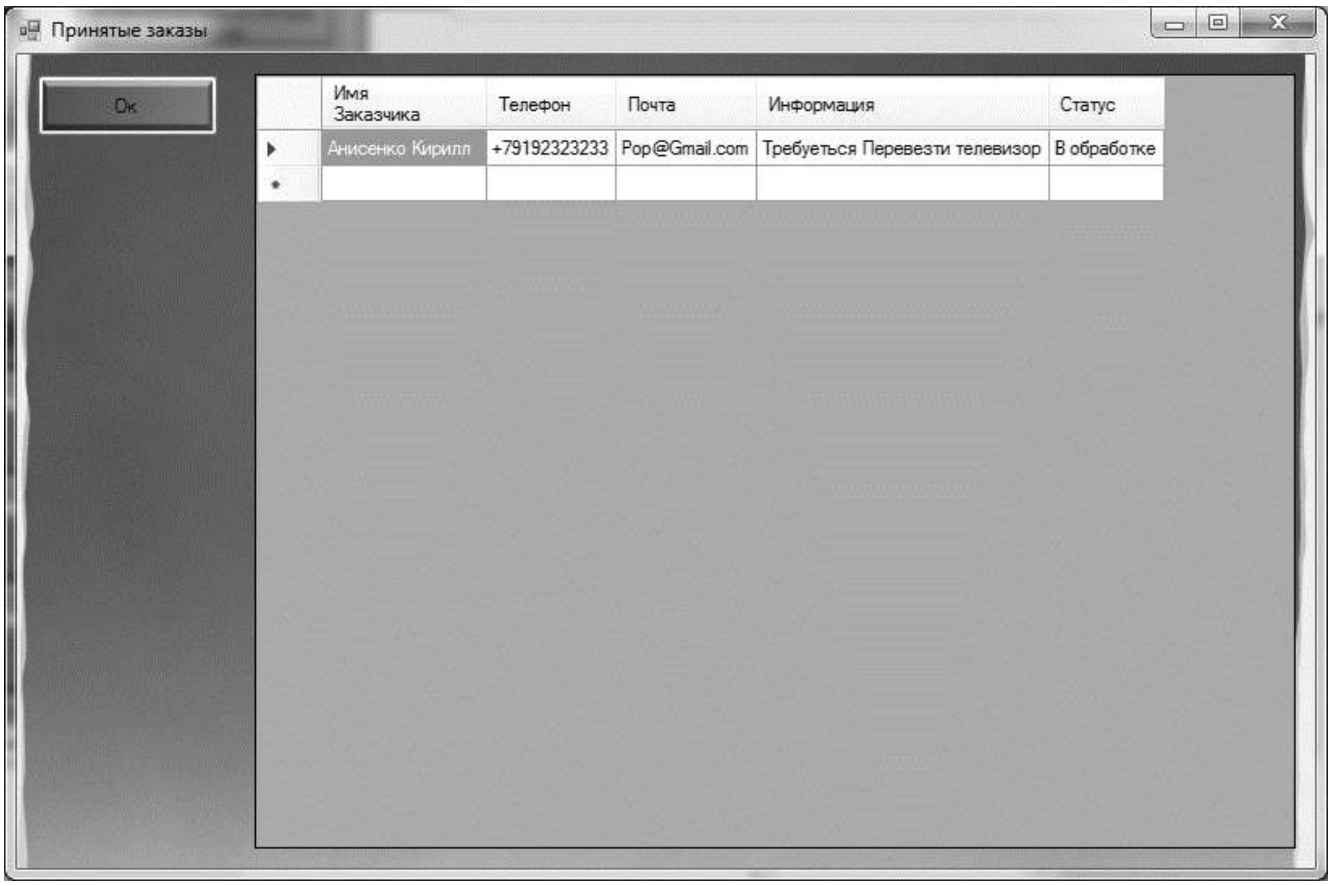

Рис 2.4.16 – Форма «Принятые заказы»

На форме «Справка» выводиться информация о приложении, краткое руководство о принятие заказа, и почта для связи с администратором.

Форма «Справка» представлена на рисунке 2.4.17.

Рис 2.4.17 – Форма «Справка»

Вывод по разделу два

В разделе описаны таблицы базы данных, используемые при разработке клиентского приложения. Отображена структура разработанного приложения и приведено описание работы в программе, а именно описаны все формы. Приведены примеры их использования.

3. ОРГАНИЗАЦИОННО-ЭКОНОМИЧЕСКИЙ РАЗДЕЛ

При принятии решения о создании и внедрении приложения необходимо рассчитать экономическую эффективность и целесообразность внедрения разработки

3.1 Расчет затрат

Затраты по покупку лицензионного программного обеспечения и ПК отражены в таблице 3.1.

Таблица 3.1 – Затраты на ПО и ПК

Наименование ПО	**Стоимость, руб**
Microsoft Visual Studio 2015	28 820
Microsoft Access 2013 32-bit/x64 Russian	9500
ПК	22 000
Итого:	60320

Расчет заработной платы работников, разрабатывающих и внедряющих проект, приведен в таблице 3.2.

Таблица 3.2 – Расчет заработной платы

Должность	**Оклад, руб./мес.**	**Оплата, руб./день**	**Продолжительность работ, дни**	**Итого, руб.**
Программист	25 000	1137	20	22 740
Итого:				22 740
Дополнительная заработная плата (20% от основной)				4 548
Основная и дополнительная заработная плата				27 288
Социальные платежи (26% от основной и дополнительной заработной платы)				7 094,88

В результате затраты на разработку и внедрение программного продукта составляют 94 702,88руб.

3.2 Расчет цены продукта

Было подсчитано, что в затраты на создание и внедрение программы в общей сумме составят 94 702,88руб. После внедрения программы требуется оплачивать сопровождение приложения в размере 10000 рублей в месяц администратору.

Выводы по разделу три

В организационно-экономическом разделе описан ожидаемый экономический эффект от разработки и внедрения программы, проанализированы и рассчитаны статьи затраты на разработку и внедрение.

ЗАКЛЮЧЕНИЕ

В процессе выполнения выпускной квалификационной работы поставлены цели и задачи, обоснована актуальность и необходимость выбранной темы.

В основном разделе работы, рассмотрены выбранные средства разработки. Информационная системы была разработана на языке программирования C# на платформе .NET при помощи Microsoft Visual Studio 2015, а в качестве СУБД была использована программа Microsoft Office Access 2013.

Кроме того, спроектирована и разработана база данных и описаны используемые в программе таблицы БД.

Что касается ведущего клиентского приложения, разработана его структура и описана работа отдельных частей программы.

В экономической части данной работы подсчитаны затраты на разработку и внедрение программного продукта.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Официальное руководство по среде Visual Studio: – https://docs.microsoft.com/ru-ru/visualstudio/ .

2. Справочный центр разных языков программирования: – https://ru.stackoverflow.com.

3. Ресурс для IT-Специалистов: – https://habr.com/sandbox/72340/.

4. Официальная документация по технологиям Яндекс: – https://tech.yandex.ru/maps/doc/jsapi/2.1/dg/concepts/load-docpage/.

5. Официальная документация по C#: – https://msdn.microsoft.com/ru-ru/library/67ef8sbd(v=vs.120).aspx.

6. Фуллер, Л.У. Access 2010 для чайников / Л.У. Фуллер, К. Кук. – М.: Диалектика, 2010. – 384 с.

ПРИЛОЖЕНИЕ А

Код формы «Вход»

```
public partial class Login : Form
    {
        public Login()
        {
            InitializeComponent();
            textBoxPassword.Text = "";
            textBoxPassword.PasswordChar = '*';
            textBoxPassword.MaxLength = 14;
        }
        private void button2_Click(object sender, EventArgs e)
        {
            Regist Reg = new Regist();
            Reg.Show();
        }
        private void label3_Click(object sender, EventArgs e)
        {
            int id = 0;
            Main_GLAV M = new Main_GLAV(" ",id);
            M.Show();
            this.Hide();
        }
        private void Button1_Click(object sender, EventArgs e)
        {
            if (textBoxLogin.Text!="" && textBoxPassword.Text!="")
            {
                OleDbConnection con = new
OleDbConnection("provider=Microsoft.Jet.OLEDB.4.0;" +
                "data source= DiplomS.mdb;Persist Security Info=True;Jet
OLEDB:Database Password=tg2120324");
              OleDbDataAdapter ada = new OleDbDataAdapter("Select * From
[User] where [Login] = '" + textBoxLogin.Text + "'and [Password] = '" +
textBoxPassword.Text + "'", con);
```

```
            DataTable dt = new DataTable();
            ada.Fill(dt);
            int id = Convert.ToInt32(dt.Rows[0][0].ToString());
            if (dt.Rows.Count > 0)
            {
                MessageBox.Show("Вы успешно вошли в систему");
                Main_GLAV M = new Main_GLAV(textBoxLogin.Text, id);
                M.Show();
                this.Hide();
            }
            else
            {
                MessageBox.Show("Неправильный логин или пароль!!!");
            }
            con.Close();
        }
        else
        {
            MessageBox.Show("Заполните поля Логин и Пароль");
        }
    }
}
```

ПРИЛОЖЕНИЕ Б

Код формы «Регистрация»

```
public partial class Regist : Form
    {
        public Regist()
        {
            InitializeComponent();
            Mail_textBox.LostFocus += TextBox6_LostFocus1;
            Pass_textBox.Text = "";
            Pass_textBox.PasswordChar = '*';
            Pass_textBox.MaxLength = 20;
            Pass2_textBox.Text = "";
            Pass2_textBox.PasswordChar = '*';
            Pass2_textBox.MaxLength = 20;
        }

        private void TextBox6_LostFocus1(object sender, EventArgs e)
        {
            string str = Mail_textBox.Text;
            if (IsValidEmail(str) == false)
            {
                MessageBox.Show("Неправильно введена почта");
            }
            else
            {
                MessageBox.Show("все Хороошо");
            }
        }
       bool invalid = false;
        public bool IsValidEmail(string strIn)//Проверка почты
        {
          invalid = false;
            if (String.IsNullOrEmpty(strIn))
```

```
                return false;
            try
            {
                strIn = Regex.Replace(strIn, @"(@)(.+)$",
this.DomainMapper,
                                      RegexOptions.None,
TimeSpan.FromMilliseconds(200));
            }
            catch (RegexMatchTimeoutException)
            {
                return false;
            }
            if (invalid)
                return false;
            try
            {
                return Regex.IsMatch(strIn,
                      @"^(?("")(""".+?(?<!\\)""@)|(([0-9a-z]((\.(?!\.))|[-
!#\$%&'\*\+/=\?\^`\{\}\|~\w])*)(?<=[0-9a-z])@))" +
                      @"(?(\[)(\[(\d{1,3}\.){3}\d{1,3}\])|(([0-9a-z][-
\w]*[0-9a-z]*\.)+[a-z0-9][\-a-z0-9]{0,22}[a-z0-9]))$",
                      RegexOptions.IgnoreCase,
TimeSpan.FromMilliseconds(250));
            }
            catch (RegexMatchTimeoutException)
            {
                return false;
            }
        }
        private string DomainMapper(Match match)//проверка почты
        {
            IdnMapping idn = new IdnMapping();
            string domainName = match.Groups[2].Value;
            try
            {
```

```
                domainName = idn.GetAscii(domainName);
            }
            catch (ArgumentException)
            {
                invalid = true;
            }
            return match.Groups[1].Value + domainName;
        }
        private void button2_Click(object sender, EventArgs e)
        {
            string connectionString =
              "provider=Microsoft.Jet.OLEDB.4.0;" +
            "data source = DiplomS.mdb;Persist Security Info=True;Jet
OLEDB:Database Password=tg2120324";
            OleDbConnection myOleDbConnection = new
OleDbConnection(connectionString);
            OleDbCommand myOleDbCommand =
myOleDbConnection.CreateCommand();
            if (Login_textBox.Text != "" && Pass_textBox.Text != "")
            {
                try
                {
                    myOleDbCommand.CommandText = "INSERT INTO [User]" +
"([Login],[Password],[Name],[Phone],[Email])" + "VALUES('" + Log-
in_textBox.Text + "','" + Pass_textBox.Text + "','" + Name_textBox.Text +
"','" + Phone_textBox.Text + "','" + Mail_textBox.Text + "')";
                    myOleDbConnection.Open();
                    myOleDbCommand.ExecuteNonQuery();
                    Close();
                    Obshie_metodi r = new Obshie_metodi();
Obshie_metodi.SendMail("smtp.mail.ru", "stroichel174@mail.ru", "tg2120324",
"Stroichel74@yandex.ru", "Регистрация", "/nВ приложении зарегистрировался"
+ Name_textBox.Text + "\nЛогин" + Login_textBox + "\nПароль" +
Pass_textBox.Text + "\nПочта-" + Mail_textBox.Text + "\nТелефон-" +
Phone_textBox.Text + "требуется добавить в базу");
                } catch
                {
```

```
            MessageBox.Show("Данный Логин Уже используется)");
        }
    }
    else
    {
        MessageBox.Show("Вы не заполнили поля Логин или пароль");
    }
}
private void textBox3_KeyUp(object sender, KeyEventArgs e)
{
    ToolTip pod = new ToolTip();
    string a = Pass_textBox.Text;
    string b = Pass2_textBox.Text;
    if (a != b)
    {
        pod.Show("Неправильно", Pass2_textBox, 3000);
    }
    if(a == b)
    {
        pod.Show("правильно", Pass2_textBox, 3000);
    }
}
private void button1_Click(object sender, EventArgs e)
{
    Close();
}
```

ПРИЛОЖЕНИЕ В

Код формы «Поиск фирм»

```
public partial class Find_Firms : Form
    {
public Find_Firms()
        {
            InitializeComponent();
        }
DataTable dt = new DataTable("Firm_Viw");
DataTable dts = new DataTable("Firm_Kategory");
private void Find_Firms_Load(object sender, EventArgs e)
        {
            try
            {
using (OleDbConnection con = new OleDbConnection("provider = Mi-
crosoft.Jet.OLEDB.4.0; " +
"data source = DiplomS.mdb;Persist Security Info=True;Jet OLEDB:Database
Password=tg2120324"))
                {
if (con.State == ConnectionState.Closed)
                    {
con.Open();
OleDbDataAdapter das = new OleDbDataAdapter("select *from [Firm_Kateg]",
con);
das.Fill(dts);
this.comboBox1.DataSource = dts;
this.comboBox1.DisplayMember = "Kategory";
this.comboBox1.ValueMember = "Id";
using (OleDbDataAdapter da = new OleDbDataAdapter("select *from
[Firm_Viw]", con))
                        {
da.Fill(dt);
dataGridView1.DataSource = dt;
dataGridView1.Columns[0].HeaderText = "Название";
```

```
dataGridView1.Columns[1].HeaderText = "Телефон";
dataGridView1.Columns[2].HeaderText = "Почта";
dataGridView1.Columns[3].HeaderText = "Информация";
dataGridView1.Columns[4].HeaderText = "Категория";
dataGridView1.Columns[5].HeaderText = "Страна";
dataGridView1.Columns[6].HeaderText = "Город";
dataGridView1.Columns[7].HeaderText = "Улица";
dataGridView1.Columns[8].HeaderText = "Номер дома";
                        }
                    }
                }
dataGridView1.AutoResizeColumns();
            }
catch (Exception ex)
            {
MessageBox.Show(ex.Message, "Сообщение", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }
private void button1_Click_1(object sender, EventArgs e)
        {
String Name_Firms = dataGridView1.CurrentRow.Cells[0].Value as String;
String Phone_Firms = dataGridView1.CurrentRow.Cells[1].Value as String;
String Addres_Firms = dataGridView1.CurrentRow.Cells[5].Value as String +
dataGridView1.CurrentRow.Cells[6].Value as String +
dataGridView1.CurrentRow.Cells[7].Value as String +
dataGridView1.CurrentRow.Cells[8].Value as String;
String Email_Firms = dataGridView1.CurrentRow.Cells[2].Value as String;
String Info_Firms = dataGridView1.CurrentRow.Cells[3].Value as String;
String Kategory_Firms = dataGridView1.CurrentRow.Cells[4].Value as String;
            INFO_Firms F = new INFO_Firms(Addres_Firms);
            F.Name_Firms = Name_Firms;
            F.Phone_Firms = Phone_Firms;
            F.Addres_Firms = Addres_Firms;
            F.Mail_Firms = Email_Firms;
```

```
            F.Info_Firms = Info_Firms;
            F.Kategory_Firms = Kategory_Firms;
            F.Show();
        }
private void comboBox1_MouseClick(object sender, MouseEventArgs e)
        {
if (comboBox1.Text != "")
            {
DataView dv = dt.DefaultView;
dv.RowFilter = string.Format("[Kategory] = '{0}'", comboBox1.Text);
dataGridView1.DataSource = dv.ToTable();
            }
        }
private void button2_Click(object sender, EventArgs e)
        {
DataView dv = dt.DefaultView;
dv.RowFilter = String.Empty;
dataGridView1.DataSource = dv.ToTable();
        }
    }
```

ПРИЛОЖЕНИЕ Г

Код формы «Поиск Материалов»

```
public partial class Find_Items : Form
    {
public Find_Items()
        {
InitializeComponent();
        }
DataTable dt = new DataTable("Material_Viw");
DataTable dts = new DataTable("Material_Kategory");
private void Find_Items_Load(object sender, EventArgs e)
        {
try
            {
using (OleDbConnection con = new OleDbConnection("provider = Mi-
crosoft.Jet.OLEDB.4.0; " +"data source=DiplomS.mdb;Persist Security In-
fo=True;Jet OLEDB:Database Password=tg2120324"))
                {
if (con.State == ConnectionState.Closed)
                    {
con.Open();
OleDbDataAdapter das = new OleDbDataAdapter("select *from [Materi-
al_Kategory]", con);
das.Fill(dts);
this.comboBox1.DataSource = dts;
this.comboBox1.DisplayMember = "Name";
this.comboBox1.ValueMember = "Id";
using (OleDbDataAdapter da = new OleDbDataAdapter("select *from [Materi-
al_Viw]", con))
                        {
da.Fill(dt);
dataGridView1.DataSource = dt;
dataGridView1.Columns[0].Visible = false;
dataGridView1.Columns[1].HeaderText = "Наименование матреиала";
```

```
dataGridView1.Columns[2].HeaderText = "Цена";
dataGridView1.Columns[3].HeaderText = "Категория";
dataGridView1.Columns[4].HeaderText = "Фирма продавец";
                        }
                    }
                }
dataGridView1.AutoResizeColumns();
            }
catch (Exception ex)
            {
MessageBox.Show(ex.Message, "Сообщение", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }
private void textBox_Search_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (e.KeyChar == (char)13)
            {
                DataView dv = dt.DefaultView;
            dv.RowFilter = string.Format("[Material.Name] like '%{0}%'",
textBox_Search.Text);
                dataGridView1.DataSource = dv.ToTable();
            }
        }
        private void button1_Click(object sender, EventArgs e)
        {
            if (comboBox1.Text != "" && textBox1.Text != "" &&
textBox2.Text != "")
            {
                DataView dv = dt.DefaultView;
dv.RowFilter = string.Format("[Cost] < '{1}' and [Cost] > '{0}' and [Mate-
rial_Kategory.Name] = '{2}'  ", textBox1.Text, textBox2.Text,
comboBox1.Text);

                dataGridView1.DataSource = dv.ToTable();
```

```
            }
            else
            {
                if (comboBox1.Text != "")
                {
                    DataView dv = dt.DefaultView;
           dv.RowFilter = string.Format("[Material_Kategory.Name] = '{0}'",
comboBox1.Text);
                    dataGridView1.DataSource = dv.ToTable();
                }
                if (textBox1.Text != "")
                {
                    DataView dv = dt.DefaultView;
           dv.RowFilter = string.Format("[Cost] > '{0}'", textBox1.Text);
                    dataGridView1.DataSource = dv.ToTable();
                }
                if (textBox2.Text != "")
                {
                    DataView dv = dt.DefaultView;
dv.RowFilter = string.Format("[Cost] < '{0}'", textBox2.Text);
                    dataGridView1.DataSource = dv.ToTable();
                }
                if (textBox1.Text != "" && textBox2.Text != "")
                {
                    DataView dv = dt.DefaultView;
dv.RowFilter = string.Format("[Cost] < '{1}' and [Cost] > '{0}' ",
textBox1.Text, textBox2.Text);
                    dataGridView1.DataSource = dv.ToTable();
                }
            }
```



```
        }
        private void button2_Click(object sender, EventArgs e)
        {
```

```
Int32 Id_Item = Convert.ToInt32(dataGridView1.CurrentRow.Cells[0].Value);
String Name_Item = dataGridView1.CurrentRow.Cells[1].Value as String;
Int32 Cost_item = Convert.ToInt32(dataGridView1.CurrentRow.Cells[2].Value);
String Kategory_Material = dataGridView1.CurrentRow.Cells[3].Value as
String;
String Firma = dataGridView1.CurrentRow.Cells[4].Value as String;
            Info_Items F = new Info_Items();
            F.Name_Item = Name_Item;
            F.Cost_item = Cost_item;
            F.Kategory_Item = Kategory_Material;
            F.Firm_Item = Firma;
            F.Show();
        }
        private void button3_Click(object sender, EventArgs e)
        {
            DataView dv = dt.DefaultView;
            dv.RowFilter = String.Empty;
            dataGridView1.DataSource = dv.ToTable();
        }
    }
```

ПРИЛОЖЕНИЕ Д

Код Формы «Принятие зааказа»

```
public partial class Prinyatie_Zakaza : Form
    {
        string A = "";
        string B = "";
        int C = 0;
        public int ID_User = 0;
        public Prinyatie_Zakaza(string Name_Zakazchika, string
Email_Zakazchika, int Status,int id ) : base()
        {
            InitializeComponent();
            A = Name_Zakazchika;
            B = Email_Zakazchika;
            C = Status;
            ID_User = id;
        }
        private void button2_Click(object sender, EventArgs e)
        {
            Close();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            if (textBox1.Text == "")
            {
                MessageBox.Show("Вы не заполнили поле имя");
            }
            else
            {
                if (textBox2.Text == "" || textBox3.Text == "")
                {
                    MessageBox.Show("Вы не заполнили поля для связи с За-
казчиком");
```

```
                }
                else
                {
Obshie_metodi.SendMail("smtp.mail.ru", "stroichel174@mail.ru", "tg2120324",
B, "Заказ","Здраствуйте" +A+ "Ваш заказ хочет Принять"+"Имя-" +
textBox1.Text + "\nПочта-" + textBox2.Text + "\nТелефон-" +
textBox3.Text+"\nПодтвердите что согласны отправив письмо на наш Адрес");
                    MessageBox.Show("Заказ отправлен на обработку");
                    OleDbConnection one = new OleDbConnection("provider =
Microsoft.Jet.OLEDB.4.0; " +
                    "data source= DiplomS.mdb ;Persist Security In-
fo=True;Jet OLEDB:Database Password=tg2120324");
                    OleDbCommand add = new OleDbCommand($"UPDATE [Work] SET
[Work].[Status] = 'В обработке' WHERE [Work].[Id]=" + C , one);
                    OleDbCommand addd = new OleDbCommand("INSERT
INTO[Prin_Zack]" + "([id_User],[id_Work])" + "VALUES('" + ID_User + "','" +
C + "'", one);
                    one.Open();
                    add.Prepare();
                    add.ExecuteNonQuery();
                    addd.Prepare();
                    addd.ExecuteNonQuery();
                }
            }
            Close();
        }
    }
```

ПРИЛОЖЕНИЕ Ж

Код файла «Obshie_metodi.cs»

```
class Obshie_metodi
    {
public static void SendMail(string smtpServer, string from, string pass-
word, string mailto, string caption, string message, string attachFile =
null)
        {
            try
            {
                MailMessage mail = new MailMessage();
                mail.From = new MailAddress(from);
                mail.To.Add(new MailAddress(mailto));
                mail.Subject = caption;
                mail.Body = message;
                if (!string.IsNullOrEmpty(attachFile))
                    mail.Attachments.Add(new Attachment(attachFile));
                SmtpClient client = new SmtpClient();
                client.Host = smtpServer;
                client.Port = 587;
                client.EnableSsl = true;
  client.Credentials = new NetworkCredential(from.Split('@')[0], password);
                client.DeliveryMethod = SmtpDeliveryMethod.Network;
                client.Send(mail);
                mail.Dispose();
            }
            catch (Exception e)
            {
                throw new Exception("Mail.Send: " + e.Message);
            }
        }
    }
```