МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ 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, зав. сектором веб-проектов
ОГСТ УИ ЮУрГУ.
_____/ А. И. Сапожников /
«_____» _____20____ г.

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой, д.т.н., с.н.с.
_____/ Б.М. Суховилов /

«_____»_____20____ г.

Разработка системы автоматизации учёта заявок в цехе по добыче нефти и газа

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.03.2019.1881.ВКР

Руководитель, к.т.н., доц.
_____/ В. А. Конов /
«_____»_____20____ г.

Автор, студент группы ЭУ-530
_____/ Н. Г. Лильбок /
«_____» _____20____ г.

Нормоконтролер, доц.
_____/ Е. А. Конова /
«_____»_____20____ г.

Челябинск 2019

АННОТАЦИЯ

Лильбок Н.Г. Разработка системы автоматизации учёта заявок в цехе по добыче нефти и газа для ООО «Лукойл-Западная Сибирь» – Челябинск: ЮУрГУ, ЭУ-530, 43 с., 10 ил., библиогр. список – 10 наим..

Дипломный проект выполнен с целью разработки системы автоматизации учёта заявок в цехе по добыче нефти и газа для ООО «Лукойл-Западная Сибирь»

В дипломном проекте проанализирована организационная структура предприятия, экономическая характеристика организации. Проведён анализ проблем цеха по добыче нефти и газа № 3 в области документооборота.

С помощью диаграммы прецедентов и схемы данных созданы графические модели приложения, позволяющие анализировать, документировать и планировать изменения.

Созданы настольное и Web приложения, проанализированы результаты внедрения дипломного проекта.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Развитие и активное внедрение электронного документооборота началось в 90-е годы, когда на российском рынке появилось большое количество программ по автоматизации делопроизводства. С тех пор эта отрасль активно развивается. Все больше организаций стремятся внедрить у себя систему электронного документооборота, чтобы повысить эффективность использования рабочего времени и свести к минимуму затраты на ручную обработку документов. Но не стоит забывать, что каждая компания, каждый цех индивидуальны. Поэтому максимальной эффективности автоматизации делопроизводства в них можно добиться только при индивидуальном подходе.

Целью выпускной квалификационной работы является усовершенствование работы внутрицеховых процессов в организации ООО «ЛУКОЙЛ-Западная Сибирь» Цех по добыче нефти и газа № 3. В частности процесса создания, рассылки и регистрации заявок, путем внедрения новой информационной системы, которая сможет обеспечить решение ряда следующих поставленных задач:

- избавиться от бумажного журнала заявок;
- снизить время создания и получения заявок, поступающих от геолого-технологической службы и пульта управления автоматикой и телемеханикой бригадам;
- снизить время обратной связи по заявкам;
- ввести единую базу данных заявок.

Для внедрения системы, удовлетворяющей поставленным задачам, необходимо:

- проанализировать рынок продуктов данного класса;
- провести сравнительный анализ нескольких выбранных систем;
- определить недостающий функционал в анализируемых системах;
- создать систему, реализующую все поставленные задачи.

После внедрения нужно произвести отладку системы и обучить сотрудников цеха работе с ней.

Тема данной выпускной квалификационной работы является особенно актуальной в современных условиях. Важно ускорять процесс принятия решений, повышать мобильность контакта геолого-технологической службы, пульта управления автоматики и телемеханики и бригад, и при этом грамотно и своевременно осуществлять мониторинг внутренних процессов цеха с целью достижения большей эффективности работы. Безусловно, автоматизация здесь выступает ключевым звеном.

Выпускная квалификационная работа состоит из трех частей.

В первой части отражена аналитическая часть проекта, в которой представлены технико-экономическая характеристика ООО «Лукойл Западная Сибирь» и существующего на предприятии процесса выдачи и регистрации заявок от геолого-технологической службы, пульта управления автоматики и телемеханики бригадам; устранение недостатков существующих систем путем создания компьютерной системы, и обоснование необходимости автоматизации описанного процесса в ООО «Лукойл Западная Сибирь»; анализ существующих разработок.

Вторая часть проекта содержит непосредственно проектную часть, которая состоит из разработки проекта автоматизации регистрации заявок от геолого-технологической службы, пульта управления автоматики и телемеханики, информационного и программного обеспечения автоматизации описанного процесса, а также описания контрольного примера реализации проекта.

Третья работы включает в себя разработку и создание приложения автоматизации регистрации заявок от геолого-технологической службы, пульта управления автоматики и телемеханики.

1 АНАЛИТИЧЕСКАЯ ЧАСТЬ

1.1 Технико-экономическая характеристика объекта

1.1.1 Экономическая характеристика организации

ООО «ЛУКОЙЛ-Западная Сибирь» добывает более 40% углеводородов Группы «ЛУКОЙЛ» [1]. Предприятие образовано в 1995 году, является 100% дочерним обществом Компании.

ЛУКОЙЛ-Западная Сибирь работает на территории крупных северных энергетических регионов России: Тюменской области, Ханты-Мансийского автономного округа — Югры, Ямало-Ненецкого автономного округа и на полуострове Таймыр в Красноярском крае в пределах 112 лицензионных участков, суммарная площадь которых составляет почти 76 тысяч квадратных километров.

В состав ООО «ЛУКОЙЛ-Западная Сибирь» входят шесть территориально-производственных предприятий: «Лангепаснефтегаз», «Урайнефтегаз», «Когалымнефтегаз», «Повхнефтегаз», «Покачевнефтегаз», «Ямалнефтегаз», представительство в г. Тюмени, управление производственно-технического обеспечения и комплектации оборудованием. На предприятии работают более 11 тыс. сотрудников.

Основной задачей предприятия является реализация масштабных нефтегазодобывающих проектов с использованием инновационных технологий и современного оборудования, наращивание ресурсной базы, сохранение благоприятной окружающей среды, рациональное использование природных ресурсов, укрепление экономической стабильности и развитие социальной инфраструктуры в регионах деятельности.

1.1.2 Организационная структура управления внутри цеха по добыче нефти и газа

На рисунке 1.1 представлена схема структуры управления внутри цеха.

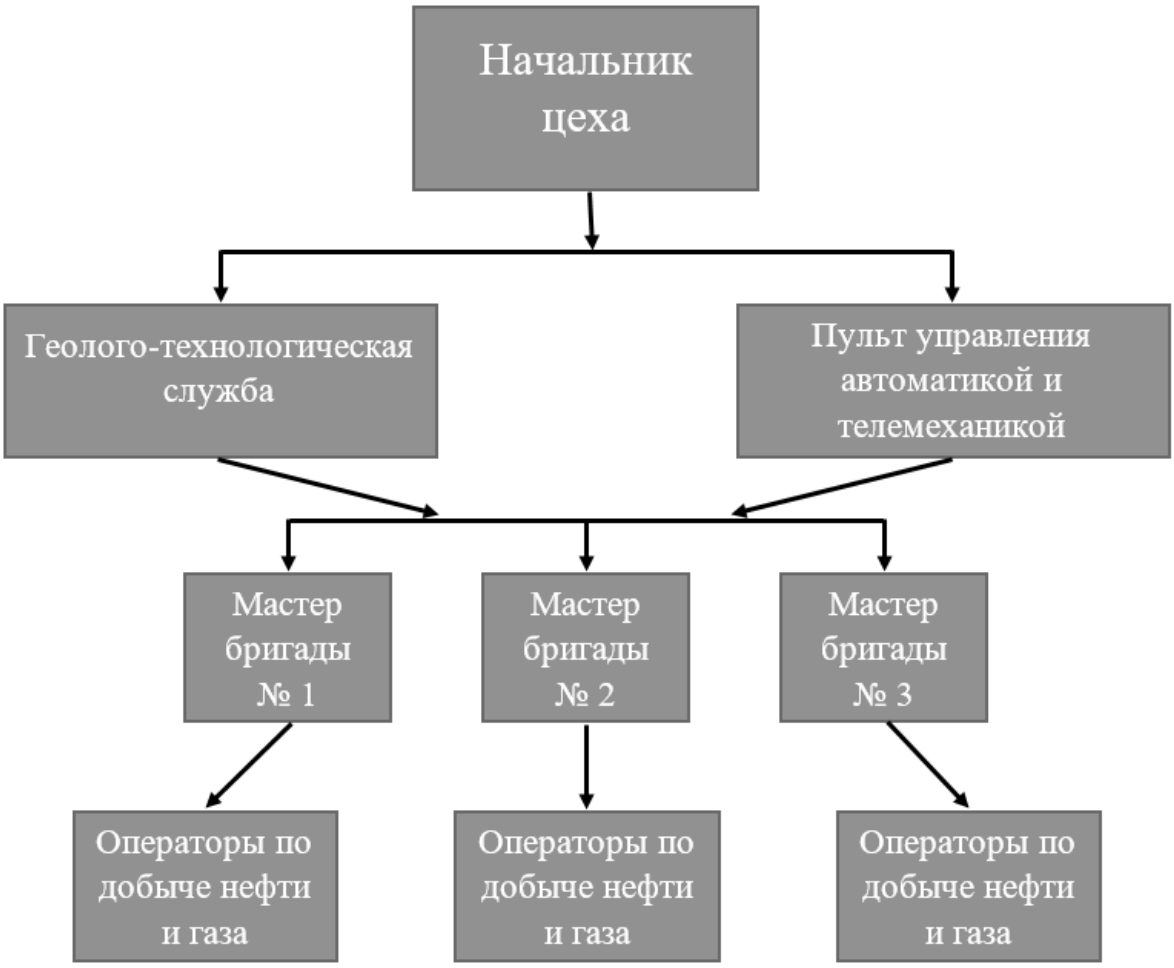

Рисунок 1.1 — Организационная структура
цеха по добыче нефти и газа № 3

В цехе по добыче нефти и газа № 3 ООО «Лукойл Западная Сибирь» работой руководит начальник цеха. Он ставит задачи перед геолого-технологической службой и операторами пульта управления автоматики и телемеханики.

Геолого-технологическая служба и пульт управления автоматики и телемеханики, в свою очередь, ставят задачи перед мастерами бригад. Происходит это посредством заполнения бумажного журнала заявок, который относят мастерам. Мастера ознакомляют исполнителей, т.е. операторов по добыче нефти и газа с заявками, которые последние должны выполнить.

1.2 Анализ текущей деятельности и постановка задачи

1.2.1 Текущее положение

На текущий момент в цехе по добыче нефти и газа используется бумажный журнал выдачи и регистрации заявок.

Геолого-технологическая служба и пульт управления автоматики и телемеханики в системе телеконтроля изучают состояние скважин, насосного оборудования, спущенного в скважины, дебит скважин и по этим данным составляют заявки для работы операторов.

Например, технолог видит, что замер по скважине, которая должна добывать 100 $м^3$ / сутки нефти, упал до 30 $м^3$/ сутки. Он составляет заявку, что нужно проверить эту скважину и произвести ручной замер. Заявка записывается в бумажный журнал и его относят в бригаду мастеру. Мастер изучает заявку и ставит задачу оператору по добыче нефти и газа. После выполнения заявки, оператор заполняет журнал, и его относят обратно в геолого-технологическую службу.

Этот метод выдачи и регистрации заявок имеет множество недостатков. Заявки на день составляются ночью либо рано утром, чтобы к началу смены журнал успели отнести в бригады мастерам (геолого-технологическая служба, пульт управления автоматики и телемеханики и бригады территориально удалены друг от друга). С утра, в начале смены, операторы по добыче нефти и газа ознакомляются с заявками и в течение дня работают по ним. Заполняется журнал операторами вечером, в конце рабочей смены, после чего его уносят обратно в геолого-технологическую службу и на пульт управления автоматикой и телемеханикой.

Если же в течение рабочей смены геолого-технологическая служба или пульт управления автоматикой и телемеханикой решают подать ещё одну или несколько заявок для работы, то им приходится связываться с мастерами по телефону либо отправлять письмо на почту. После выполнения этих заявок они чаще всего теряются и не записываются в журнал.

Поскольку журнал возвращается в геолого-технологическую службу и на пульт управления автоматики и телемеханики только в конце рабочей смены, то посмотреть и проанализировать выполненные заявки в течение рабочей смены невозможно. Геолого-технологической службе и пульту управления автоматикой и телемеханикой приходится связываться с мастерами, а то и с операторами по телефону и узнавать состояние заявок.

Заявки для оператора, работающего в ночную смену, в журнале не записываются вообще, а передаются по почте или телефону, поскольку смена заканчивается и журнал некому унести в бригаду.

Основные недостатки бумажного журнала выдачи и регистрации заявок:

- часть заявок теряется и не попадает в журнал;
- затрачивается время для того чтобы переносить журнал между геолого-технологической службой, пультом управления автоматики и телемеханики и бригадами;
- невозможность отследить по журналу состояние заявок в течение рабочей смены;
- большое количество времени уходит на заполнение журнала вручную;
- затруднения в поиске необходимой информации в журнале.

1.2.2 Постановка задачи

Исходя из недостатков существующей системы выдачи и регистрации заявок, бумажного журнала, возникает необходимость внедрения компьютерной системы, которая сократит время на написание заявок как операторами, так и геолого-технологической службой и операторами пульта управления автоматики и телемеханики, сведёт к нулю потери времени на перенос бумажного журнала по территории цеха, позволит просматривать выполнение заявок в течение рабочей смены, уменьшит количество заявок, которые не попадают в журнал, создаст комфортные условия для просмотра заявок и поиска необходимой информации.

1.3 Анализ существующих систем

1.3.1 HelpDesk AstroSoft

Система предназначена для регистрации и учета заявок, поступающих от пользователей.

Возможности HelpDesk AstroSoft:

- стандартный способ регистрации и выдачи заданий специалистам;
- контроль за последовательностью исполнения работ, потраченным временем и ресурсами;
- отчётность по затратам времени и средств на выполнение запросов;
- гибкая система маршрутизации. Под маршрутизацией понимается передача заявки от одного отдела к другому;
- создание единой Базы Знаний;
- легкий Web-интерфейс, который понятен каждому пользователю ПК;

Прием заявок от пользователей производится тремя способами:

- с помощью электронной почты MS Outlook;
- при помощи телефонного звонка;
- непосредственно через заведение заявки самим пользователем в системе AstroSoft.

Недостатком такой обработки является:

- большое количество времени, затраченное на обработку одной заявки;
- простой в работе у конечного пользователя.

1.3.2 Учет заявок для 1С

По почте ответственных сотрудников, участников Система учета заявок для 1С на платформе «Цифровой Кот» предназначена для автоматизации входящих обращений от клиентов и партнеров. Она представляет собой встроенный в 1С helpdesk, работающий в режиме реального времени.

Заявки размещаются через Интернет. Для каждого клиента настраиваются уровни доступа.

Система учета заявок для 1С позволяет правильно выстроить работу с заявками и отладить технологические процессы на предприятии. Область применения - для любых компаний с большим потоком поступающей информации, где требуется учесть и качественно обработать каждую заявку.

Возможности программы «Учет заявок для 1С»:

- работа в локальной сети и через Интернет;
- приоритет заявок;
- хранение всех заявок с начала работы функционала;
- настройка уведомлений процесса;
- доступность с мобильных устройств.

1.3.3 Регистрация заявок LAN

«Регистрация заявок LAN» - HelpDesk система для учета и отслеживания заявок и вопросов от пользователей в сети.

Предположим, что у пользователя возникла какая-либо проблема или вопрос, который он хочет направить IT-персоналу или администратору. С помощью программы «Клиент» он создает заявку, которая поступает на сервер и обрабатывается программой «Сервер». Клиент может также отправлять заявки выбранным исполнителям.

Администратор (диспетчер), у которого установлена программа «Администратор», получает уведомление о поступившей заявке и обрабатывает её. Он может назначить исполнителя и направить заявку ему. Исполнитель в свою очередь получает об этом уведомление. Администратор может удалять заявки, направлять их исполнителям или изменять статус, а исполнитель - только изменять статус.

Пользователь, который подал заявку, имеет возможность просматривать изменение её статуса, в котором может быть указан срок выполнения заявки, исполнитель или ответ на вопрос.

Таким образом, все возникшие вопросы и проблемы держатся на контроле как у администраторов и исполнителей, так и у пользователей.

Программа состоит из 3-х модулей: программа «Сервер», программа «Администратор» и программа «Клиент».

Программа «Сервер» (rrLAN_server.exe) содержит в себе программу, которая обрабатывает заявки, а также сервер баз данных Firebird. Программа обязательно устанавливается на 1 компьютер. Программа является сервером, который принимает заявки от пользователей. Если, по какой-либо причине, программа будет отключена, заявки будут находиться в режиме ожидания, и поступят на обработку сразу после включения программы.

Программа «Администратор» (rrLAN_admin.exe) содержит в себе программу, с помощью которой можно просматривать и редактировать заявки, делать отчеты, создавать задачи и многое другое. Устанавливается на любое число компьютеров всем администраторам, исполнителям или наблюдателям.

Программа «Клиент» (rrLAN_client.exe) содержит программу, с помощью которой создаются заявки. Устанавливается всем пользователям, которые будут отправлять заявки.

Основной набор функций:

- регистрация и отслеживание заявок от пользователей в локальной сети TCP/IP (или в VPN-сети). Программа-сервер работает в постоянном режиме и принимает заявки от пользователей;
- редактируемая база знаний;
- возможность изменения в описании заявок и создания своей структуры заявок;
- программу можно использовать в любых других целях, если условия задачи связаны с отправкой заявок и их обработкой;
- 3 уровня описания заявок: общее название, уточнение и подробное описание.
- Клиент-Серверная технология (3 программных модуля). Клиенты — это пользователи сети, Сервер — главный администратор или диспетчер, которому поступают заявки или жалобы от пользователей;

- администратор или диспетчер может работать с программой-сервером, а может подключиться удаленно с помощью программы-администратора;
- возможность отправлять заявки напрямую исполнителю;
- поддержка многоязычности. В программе доступны 2 языка: русский и английский. Есть возможность добавлять конфигурации с любыми другими языками;
- программа обладает простым интерфейсом, который будет понятен любому пользователю и не требует специального обучения. Администратору и исполнителям предоставляется удобный механизм контроля поступивших заявок, а каждому пользователю – возможность отслеживать состояние своей заявки.

1.3.4 Заявки ИТ

Заявки ИТ - это система с WEB-интерфейсом и Windows-клиентами, позволяющая упростить и автоматизировать учет и обработку заявок поступающих в отдел ИТ. Обладает мощным генератором отчётов FastReport, который может составить отчёт любой сложности.

Возможности программы «Заявки ИТ»:

- интуитивно понятный интерфейс;
- регистрация и отслеживание заявок от пользователей в локальной сети;
- настраиваемый WEB-интерфейс для регистрации заявок от пользователей;
- уведомления о новых заявках в модулях «Руководитель» и «Сотрудник»;
- рассылка уведомлений о назначении заявок;
- возможность установки приоритетов и сроков выполнения заявок;
- цветовая расцветка приоритетов и сроков выполнения;
- широкий временной диапазон обновления заявок;
- поиск, сортировка и настраиваемый фильтр заявок;
- редактируемые справочники: Сотрудники и Инциденты;

- мощная система формирования отчетов на основе полученных заявок;
- возможность вкладывать в заявки неограниченное количество файлов.

1.3.5 Обоснование разработки собственной системы

На предприятии в цехе по добыче нефти и газа № 3 ведется бумажный журнал регистрации заявок, этот метод документооборота морально устарел и требует замены на электронную систему [3].

–

нсовых затрат предприятию ООО «Лукойл Западная Сибирь». Выбрав готовое программное обеспечение, появляются жёсткие рамки, установленные разработчиком этого программного обеспечения, и в лучшем случае, предприятие может претендовать на незначительные доработки этого продукта. Также, выбором узконаправленного программного продукта нет возможности дальнейшего расширения информационной системы [6].

Плюсы разработки собственной автоматизированной системы это то, что программа создается конкретно под потребности и задачи предприятия ООО «Лукойл Западная Сибирь», то есть в ней нет ничего лишнего, и при этом есть всё что нужно.

Вывод по первому разделу

Выполнена постановка задачи, проведён сравнительный анализ систем документооборота, принято решение о разработке собственного программного обеспечения.

2 ПРОЕКТНАЯ ЧАСТЬ

2.1 Описание системы с использованием языка моделирования UML

Унифицированный язык моделирования (UML) является стандартным инструментом для создания «чертежей» программного обеспечения. С помощью UML можно визуализировать, специфицировать, конструировать и документировать артефакты программных систем.

UML пригоден для моделирования любых систем: от информационных систем масштаба предприятия до распределенных Web-приложений и даже встроенных систем реального времени. Это очень выразительный язык, позволяющий рассмотреть систему со всех точек зрения, имеющих отношение к ее разработке и последующему развертыванию. Несмотря на обилие выразительных возможностей, этот язык прост для понимания и использования. Изучение UML удобнее всего начать с его концептуальной модели, которая включает в себя три основных элемента: базовые строительные блоки, правила, определяющие, как эти блоки могут сочетаться между собой, и некоторые общие механизмы языка.

UML - это графическое представление набора элементов, изображаемое чаще всего в виде связанного графа с вершинами (сущностями) и ребрами (отношениями). Диаграммы рисуют для визуализации системы с разных точек зрения. Диаграмма - в некотором смысле одна из проекций системы. Как правило, за исключением наиболее тривиальных случаев, диаграммы дают свернутое представление элементов, из которых составлена система. Один и тот же элемент может присутствовать во всех диаграммах, или только в нескольких (самый распространенный вариант), или не присутствовать ни в одной (очень редко).

В этой работе будет использоваться диаграмма прецедентов.

Диаграммы прецедентов — это один из пяти типов диаграмм, применяемых в UML для проектирования информационных систем. Диаграммы прецедентов играют основную роль в моделировании поведения системы, подсистемы или класса. Каждая такая диаграмма показывает множество

прецедентов, актеров и отношения между ними. Диаграммы прецедентов применяются для моделирования вида системы с точки зрения прецедентов (или вариантов использования). Чаще всего это предполагает моделирование контекста системы, подсистемы или класса либо моделирование требований, предъявляемых к поведению указанных элементов. Диаграммы прецедентов имеют большое значение для визуализации, специфицирования и документирования поведения элемента. Они облегчают понимание систем, подсистем или классов, представляя взгляд извне на то, как данные элементы могут быть использованы в соответствующем контексте. Кроме того, такие диаграммы важны для тестирования исполняемых систем в процессе прямого проектирования и для понимания их внутреннего устройства при обратном проектировании.

2.1.1 Диаграмма прецедентов создаваемой системы

На рисунке 2.1 представлена диаграмма прецедентов создаваемой системы по учёту и регистрации заявок.

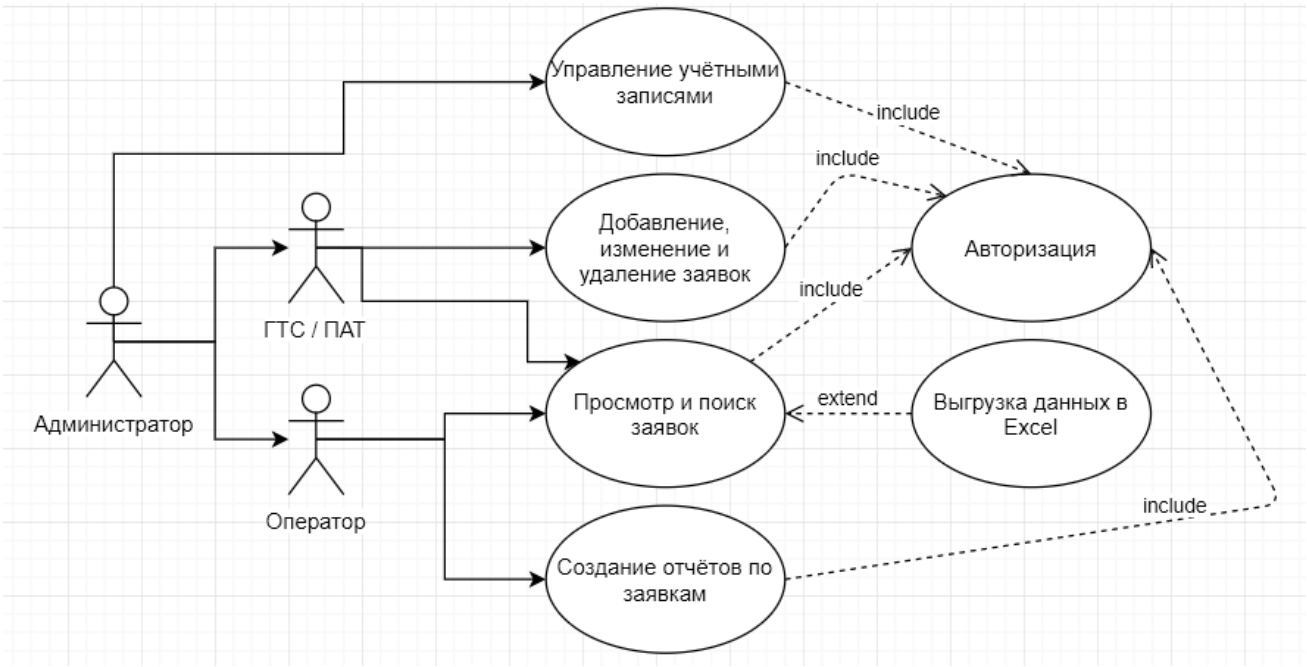

Рисунок 2.1 — Диаграмма прецедентов создаваемой системы

На диаграмме прецедентов видно, что для того, чтобы выполнить какое-либо действие в системе всем пользователям необходимо авторизоваться. После

авторизации всем пользователям будет доступен просмотр заявок и выгрузка таблицы заявок в Excel.

Пользователь «Оператор» может создавать отчёты по выполненным заявкам.

Пользователь «ГТС / ПАТ» может управлять заявками, т.е. добавлять их, изменять и удалять.

Пользователь «Администратор» имеет доступ ко всем функциям системы.

2.2 Схема данных

Схема данных является графическим образом БД. Она используется для определения связей между несколькими таблицами. Например, при создании формы, содержащей данные из нескольких взаимосвязанных таблиц, схема данных обеспечивает автоматический согласованный доступ к полям этих таблиц [8]. Она же обеспечивает целостность взаимосвязанных данных при корректировке таблиц.

Связь между таблицами устанавливает отношения между совпадающими значениями в ключевых полях, обычно между полями, имеющими одинаковые имена в обеих таблицах. В большинстве случаев с ключевым полем одной таблицы, являющимся уникальным идентификатором каждой записи, связывается внешний ключ другой таблицы.

Обязательным условием при установлении связи является совпадение связываемых полей по типу и формату.

На рисунке 2.2 представлена схема данных создаваемой системы.

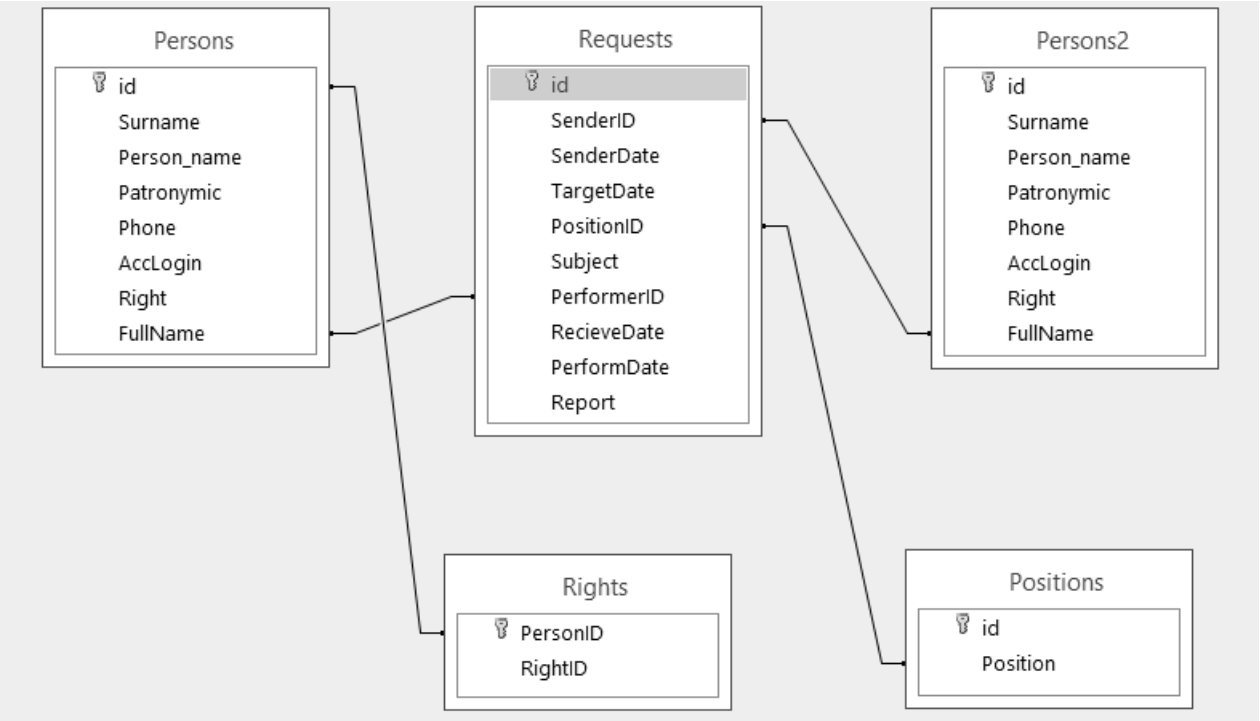

Рисунок 2.2 — Схема данных создаваемой системы

На схеме данных показана связь между данными в таблицах системы учёта и регистрации заявок. Таблица Persons2 является копией таблицы Persons. Таблицы Persons и Persons2 связаны с таблицей Requests полями FullName с полями SenderID и PerformerID. Таблица Positions связана полем Position с полем PositionID в таблице Requests. Таблица Rights связана полем PersonID с id в таблице Persons. Схема данных приведена в третьей нормальной форме.

Вывод по второму разделу

Разработана инфологическая модель приложения. Построена диаграмма прецедентов. Разработана схема данных.

3 РАЗРАБОТКА И СОЗДАНИЕ ПРИЛОЖЕНИЯ

3.1 Обоснование выбора среды разработки

Для реализации поставленных задач по разработке системы учета заявок была выбрана среда разработки Microsoft Visual Studio 2017 и язык программирования C#.

Microsoft Visual Studio — линейка продуктов компании Майкрософт, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом [10].

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. 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. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода).

Главным преимуществом Visual Studio 2017 является производительность. Обеспечивает возможность создания разнообразных приложений на основе одного набора навыков.

C# — объектно-ориентированный язык программирования.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию,

поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Основные достоинства C#:

- C# создавался параллельно с каркасом Framework .Net и в полной мере учитывает все его возможности — как FCL, так и CLR;
- C# является полностью объектно-ориентированным языком, где даже типы, встроенные в язык, представлены классами;
- C# является мощным объектным языком с возможностями наследования и универсализации;
- C# является наследником языков C/C++ [2], сохраняя лучшие черты этих популярных языков программирования. Общий с этими языками синтаксис, знакомые операторы языка облегчают переход программистов от C++ к C#;
- сохранив основные черты своего великого родителя, язык стал проще и надежнее. Простота и надежность, главным образом, связаны с тем, что на C# хотя и допускаются, но не поощряются такие опасные свойства C++ как указатели, адресация, разыменование, адресная арифметика;
- благодаря каркасу Framework .Net, ставшему надстройкой над операционной системой, программисты C# получают те же преимущества работы с виртуальной машиной, что и программисты Java. Эффективность кода даже повышается, поскольку исполнительная среда CLR представляет собой компилятор промежуточного языка, в то время как виртуальная Java-машина является интерпретатором байт-кода;
- мощная библиотека каркаса поддерживает удобство построения различных типов приложений на C#, позволяя легко строить Web-службы, другие виды компонентов, достаточно просто сохранять и получать информацию из базы данных и других хранилищ данных;

- реализация, сочетающая построение надежного и эффективного кода, является немаловажным фактором, способствующим успеху C#.

В качестве базы данных приложения был выбран Microsoft SQL Server.

Microsoft SQL Server — система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

3.1 Реализация приложения

Реализация программного обеспечения – это процесс перевода системной спецификации в работоспособную систему [7]. Разработка приложений подсистемы осуществлялась на языке C# [9] с использованием платформы .Net Framework и входящих в нее библиотек.

Итогом реализации приложения является работоспособная информационная система. Разрабатываемая информационная система будет являться приложением клиент серверного типа [5].

3.1.1. База данных Microsoft SQL Server

В Microsoft SQL Server создана база данных Request [4]. Она включает в себя четыре таблицы:

- Persons, таблица, в которой хранятся id пользователя, его фамилия, имя, отчество, телефон и логин аккаунта;
- Positions, таблица, содержащая поля — id и место назначения;
- Requests, таблица, имеющая поля — id, используемый как номер заявки, SenderID — идентификатор отправителя заявки, SenderDate, TargetDate — дата и время отправления и назначения заявки соответственно, PositionID — идентификатор места назначения, PerformerID — идентификатор исполнителя

заявки, RecieveDate, PerformDate — дата и время получения и выполнения заявки и Report — отчёт по заявке;

- Rights, таблица, содержащая поля — PersonID – идентификатор пользователя и RightID – идентификатор прав доступа.

3.1.2 Авторизация в программе

В классе Program происходит вход в приложение (код приведён в листинге 3.1). Выделяется память под основную форму и форму авторизации. Если авторизация успешна – происходит дальнейшее выполнение кода – открытие основной формы.

Листинг 3.1 — Вход в приложение

```
static class Program
{
   static void Main()
   {
       Application.EnableVisualStyles();
       Application.SetCompatibleTextRenderingDefault(false);
       Data.CheckIni();

       MainForm mainForm = new MainForm();
       Authorization_Form form_A = new Authorization_Form();

       form_A.sqlConnection = mainForm.Connection;

       if (form_A.ShowDialog() != DialogResult.OK)
       {
           Application.Exit();
           return;
       }

       Application.Run(mainForm);
    }
}
```

В классе Data (код приведён в листинге 3.2) описываются вспомогательные функции для настройки строки подключения к базе данных, создания .ini файла с настройками подключения.

Функция WritePrivateProfileString (код приведён в листинге 3.2) записывает строку в специальный раздел файла инициализации (.ini файл).

Функция GetPrivateProfileString (код приведён в листинге 3.2) извлекает строку из указанного раздела файла инициализации.

Листинг 3.2 — Класс Data

```
public static class Data
{
     [DllImport("kernel32")]
     private static extern long WritePrivateProfileString(string
     section, string key, string val, string filePath);
     [DllImport("kernel32")]
     private static extern int GetPrivateProfileString(string
     section, string key, string def, StringBuilder retVal,
     int size, string filePath);

     public static string ConnectionString;
     public static string UserID;

     public static string IniFileName…
     public static void CheckIni()…
     public static void SaveConnectString()…
     public static string CutPassword(string cs)…
     public static string GetUserID(string cs)…
     public static string CreateConnectionString(string cs,
     string uid, string psw)…
}
```

IniFileName (код приведён в листинге 3.3) возвращает полный путь к текущему месторасположению программы (.exe файл) и меняет его расширение на .ini.

Листинг 3.3 — Поиск пути файла

```
public static string IniFileName
{
     get
          {
               string fileName = Application.ExecutablePath;
               fileName = Path.ChangeExtension(fileName, ".ini");
              return fileName;
          }
}
```

CheckIni() (код приведён в листинге 3.4) проверяет файл инициализации на существование, если его нет, то создаёт его с заданной строкой подключения. Если такой файл есть, то считывает из него данные и убирает из них пароль с помощью функции CutPassword. В переменную UserID записывается, с помощью функции GetUserID, имя пользователя из строки подключения, если такое имеется.

Листинг 3.4 — Проверка и создание файла инициализации

```
Public static void CheckIni()
     {
          stringfileName = IniFileName;

          if (!File.Exists(fileName))
          {
               Data.ConnectionString =
               "DataSource=localhost\\SQLEXPRESS;
               InitialCatalog=Request;";
               WritePrivateProfileString("Server",
               "ConnectionString", Data.ConnectionString, fileName);
          }
          else
          {
               StringBuilder temp = new StringBuilder(255);
               int i = GetPrivateProfileString("Server",
               "ConnectionString", "", temp, 255, fileName);
               Data.ConnectionString = CutPassword(temp.ToString());
          }
          UserID = GetUserID(Data.ConnectionString);
     }
```

SaveConnectString (код приведён в листинге 3.5) вызывает функцию записи строки подключения в файл инициализации.

Листинг 3.5 — Сохранение строки подключения

```
public static void SaveConnectString()
     {
          WritePrivateProfileString("Server", "ConnectionString",
          Data.ConnectionString, IniFileName);
     }
```

Функция CutPassword (код приведён в листинге 3.6) выполняет поиск пароля во входящей строке подключения. Если пароль в строке подключения есть, функция его вырезает.

Листинг 3.6 — Функция вырезки пароля

```
public static string CutPassword(string cs)
     {
          string result = String.Empty;

          if (cs.Length > 0)
          {
              int pos = cs.IndexOf("PASSWORD", 0, cs.Length,
              StringComparison.CurrentCultureIgnoreCase);
              if (pos >= 0)
```

```
            {
               int ps = cs.IndexOf(';', pos);
               if (ps > pos)
               {
                   result = cs.Remove(pos, ps - pos + 1);
               }
               else
               {
                   result = cs.Remove(pos, cs.Length - pos);
               }
            }
            else result = cs;
        }

        return result;
    }
```

Функция GetUserID (код приведён в листинге 3.7) выполняет поиск имени пользователя во входящей строке подключения и возвращает найденное значение.

Листинг 3.7 — Функция получения User id

```
public static string GetUserID(string cs)
    {
        string result = String.Empty;

        if (cs.Length > 0)
        {
            int pos = cs.IndexOf("USER ID", 0, cs.Length,
            StringComparison.CurrentCultureIgnoreCase);
            if (pos >= 0)
            {
                pos += 7;
                int pe = cs.IndexOf('=', pos);
                if (pe >= pos)
                {
                    int ps = cs.IndexOf(';', pe);
                    if (ps < 0)
                    {
                        if (cs.Length > (pe + 1)) result =
                       cs.Substring(pe + 1, cs.Length - pe -
                       1);
                    }
                    else
                    {
                          result = cs.Substring(pe + 1, ps - pe
                          - 1);
                    }
                    result = result.Trim();
```

```
                }
            }
        }
        return result;
    }
```

CreateConnectionString (код приведён в листинге 3.8) во входящей строке подключения ищет имя пользователя. Если оно есть, то заменяет его на входящий параметр имени, если его нет, то добавляет. Также добавляет на место пароля входящий параметр пароля.

Листинг 3.8 — Создание строки подключения

```
public static string CreateConnectionString(string cs, string uid,
string psw)
        {
            string result = cs.TrimEnd();

            if (result.Length > 0)
            {
                int pos = result.IndexOf("USER ID", 0,
                result.Length,
                StringComparison.CurrentCultureIgnoreCase);
                if (pos >= 0)
                {
                    pos += 7;
                    int pe = result.IndexOf('=', pos);
                    if (pe >= pos)
                    {
                        int ps = result.IndexOf(';', pe);
                        if (ps < 0)
                        {
                            result = result.Remove(pe + 1,
                            result.Length - pe - 1) + uid;
                        }
                        else
                        {
                            result = result.Remove(pe + 1, ps - pe -
                            1);
                            result = result.Insert(pe + 1, uid);
                        }
                    }
                }
                else
                {
                    if (result[result.Length - 1] != ';') result =
                    result + ';';
                    result = result + "User ID=" + uid;
```

```
            }
        }
        else result = result + "User ID=" + uid;

        if (result[result.Length - 1] != ';') result = result +
        ';';
        result = result + " Password=" + psw;
        return result;
    }
```

На рисунке 3.1 представлена форма авторизации пользователя.

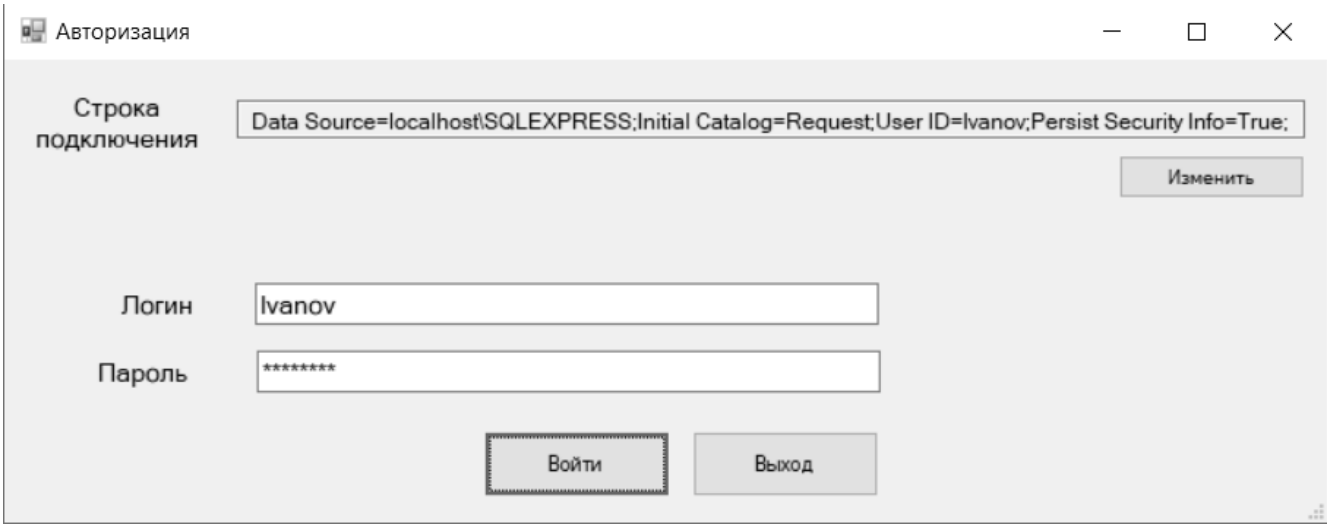

Рисунок 3.1 — Форма авторизации пользователя

Для успешной авторизации пользователю необходимо ввести свои логин и пароль и нажать на кнопку «Войти» (код приведён в листинге 3.9).

Из полей «Строка подключения», «Логин», «Пароль» формируется строка подключения. Объекту sqlConnection.ConnectionString присваивается сформированная строка подключения.

Выполняется попытка подключения, и если она успешна, то из строки подключения вырезается пароль, и если новая строка подключения отличается от старой, то файл инициализации перезаписывается с новыми параметрами. После успешного подключения форма авторизации закрывается и открывается основная форма.

Если же в процессе подключения возникли ошибки, например, был неверно введён пароль или логин, программа выдаст сообщение об ошибке.

При необходимости изменить строку подключения, пользователю необходимо нажать кнопку «Изменить», которая находится под строкой подключения.

Листинг 3.9 — Обработчик события нажатия на кнопку «Войти»

```
private void btnOK_Click(object sender, EventArgs e)
        {
            string cs =
            Data.CreateConnectionString(tbConnection.Text,
            tbLogin.Text, tbPassword.Text);

            sqlConnection.ConnectionString = cs;
            try
            {
                sqlConnection.Open();
            }
            catch (SqlException x)
            {
                MessageBox.Show(x.Message);
                DialogResult = DialogResult.None;
            }
            if (sqlConnection.State == ConnectionState.Open)
            {
                cs = Data.CutPassword(cs);
                if (Data.ConnectionString != cs)
                {
                    Data.ConnectionString = cs;
                    Data.SaveConnectString();
                }
                DialogResult = DialogResult.OK;
            }
        }
```

Обработка нажатия кнопки «Изменить» (код приведён в листинге 3.10). При успешном изменении, в строку соединения в форме авторизации записывается новое значение.

Листинг 3.10 — Обработчик события нажатия на кнопку «Изменить»

```
private void btnChange_Click(object sender, EventArgs e)
        {
            Connection_Form form = new Connection_Form();
            form.ConnectionString = tbConnection.Text;
            if (form.ShowDialog() == DialogResult.OK)
            {
                tbConnection.Text =
                Data.CutPassword(form.ConnectionString);
            }
        }
```

Свойство формы установки строки соединения имеет два параметра: возвратить значение строки соединения и установить значение строки соединения (код приведён в листинге 3.11).

На рисунке 3.2 представлена форма установки строки соединения.

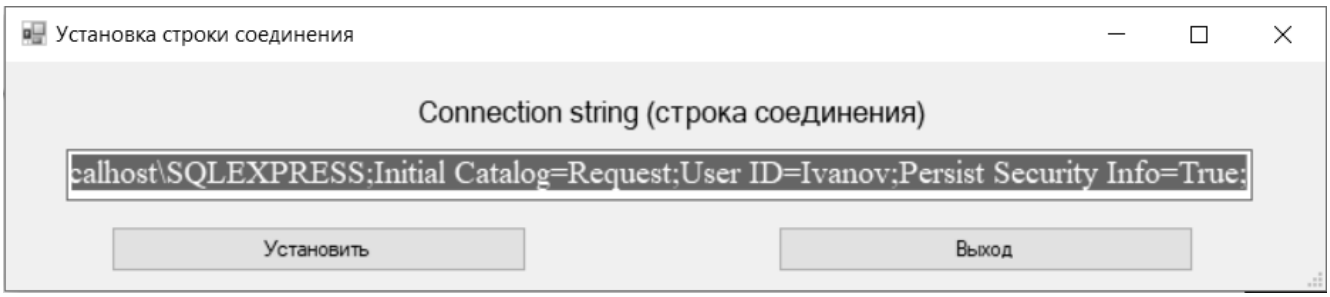

Рисунок 3.2 — Форма установки строки соединения

Листинг 3.11 — Свойство формы установки соединения

```
public string ConnectionString
        {
            get
            {
                return tbConnection.Text;
            }
            set
            {
                tbConnection.Text = value;
            }
        }
```

3.1.3 Работа с заявками

После успешной авторизации, пользователю, в зависимости от его прав доступа, открывается основная форма работы с заявками (рисунок 3.3).

Существует 3 вида прав доступа:

1. Администратор;
2. ГТС / ПАТ;
3. Оператор.

Права доступа «Администратор» открывают возможность ко всему функционалу программы. Главной отличительной чертой является возможность добавления, изменения и удаления пользователей системы.

Права доступа «ГТС / ПАТ» (геолого-технологическая служба / пульт управления автоматикой и телемеханикой) включают в себя возможность добавлять, изменять и удалять заявки.

Права доступа «Оператор» включают в себя возможность создавать отчёты по выполненным заявкам.

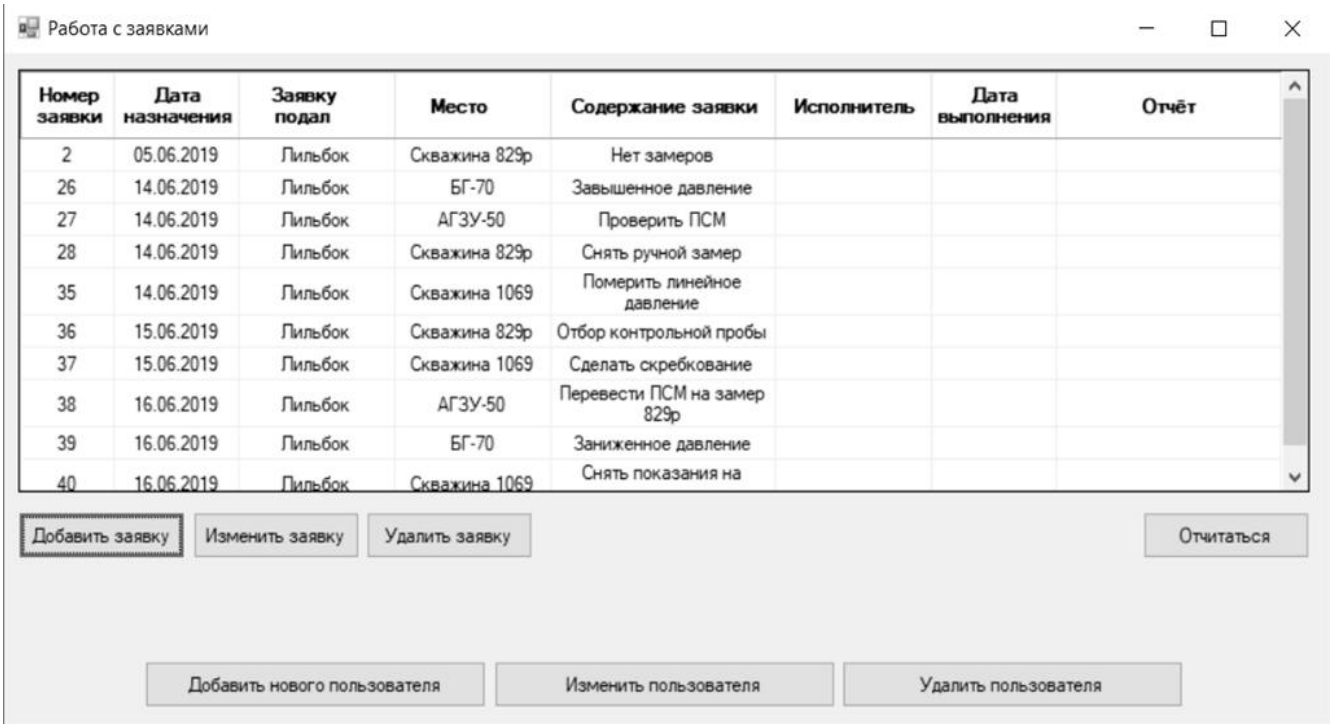

Рисунок — 3.3 Форма работы с заявками

На рисунке 3.3 представлена форма такой, какой видит её пользователь с правами администратора.

Рассмотрим функционал геолого-технологической службы и операторов пульта управления автоматикой и телемеханикой (ГТС и ПАТ). ГТС и ПАТ доступны такие функции как:

- кнопка «Добавить заявку», которая открывает форму добавления заявок (рисунок 3.4) и предлагает создать заявку;
- кнопка «Изменить заявку, которая открывает форму изменения заявки (рисунок 3.5) и предлагает изменить выбранную заявку;
- кнопка «Удалить заявку», которая удаляет одну или несколько выбранных заявок;

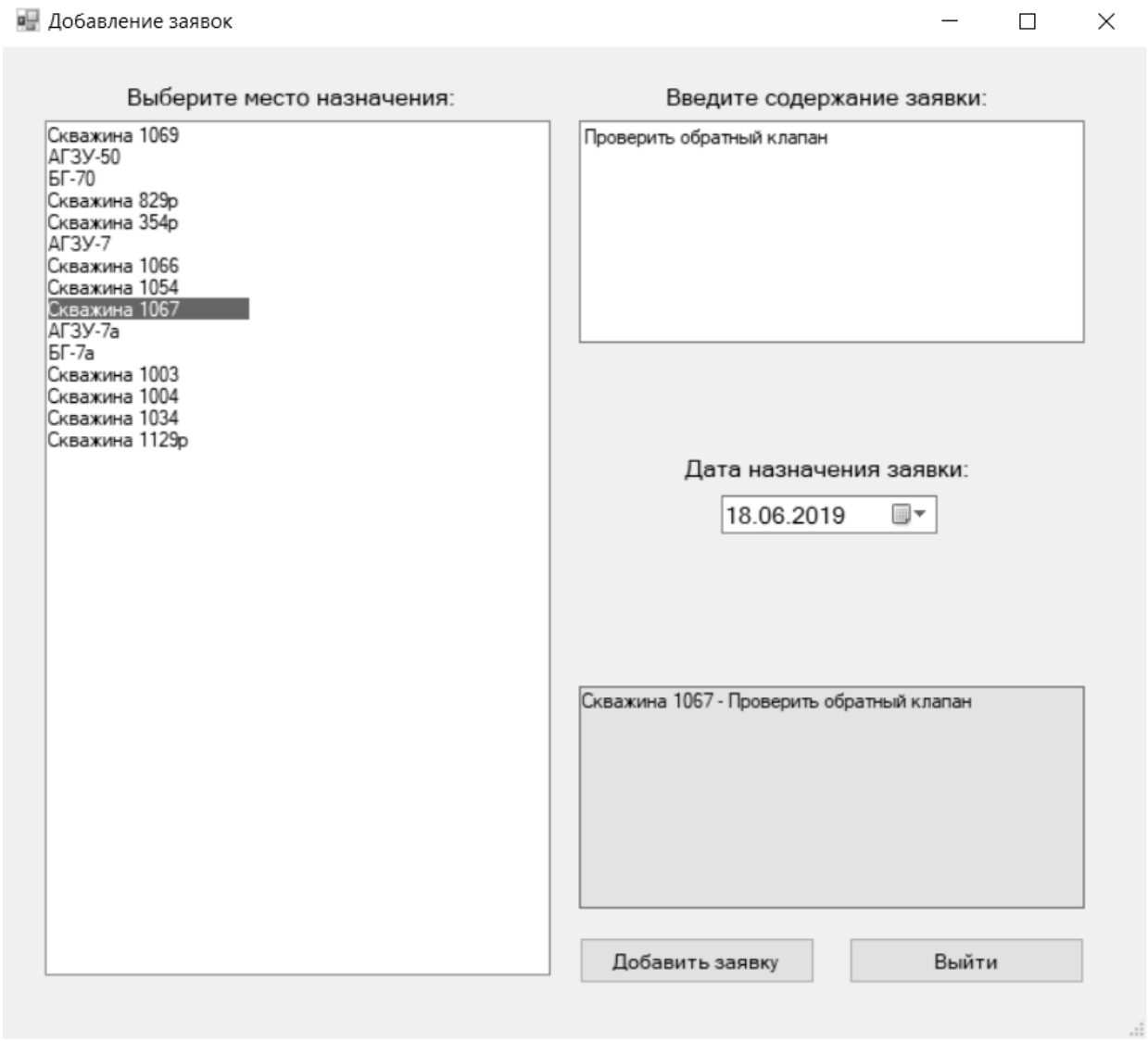

Рисунок 3.4 — Форма добавления заявок

В форме добавления заявок пользователю требуется выбрать место назначения, ввести текст содержания заявки и выбрать дату назначения заявки, после чего нажать кнопку «Добавить заявку», часть кода, отвечающая за добавление, которой приведена в листинге 3.12.

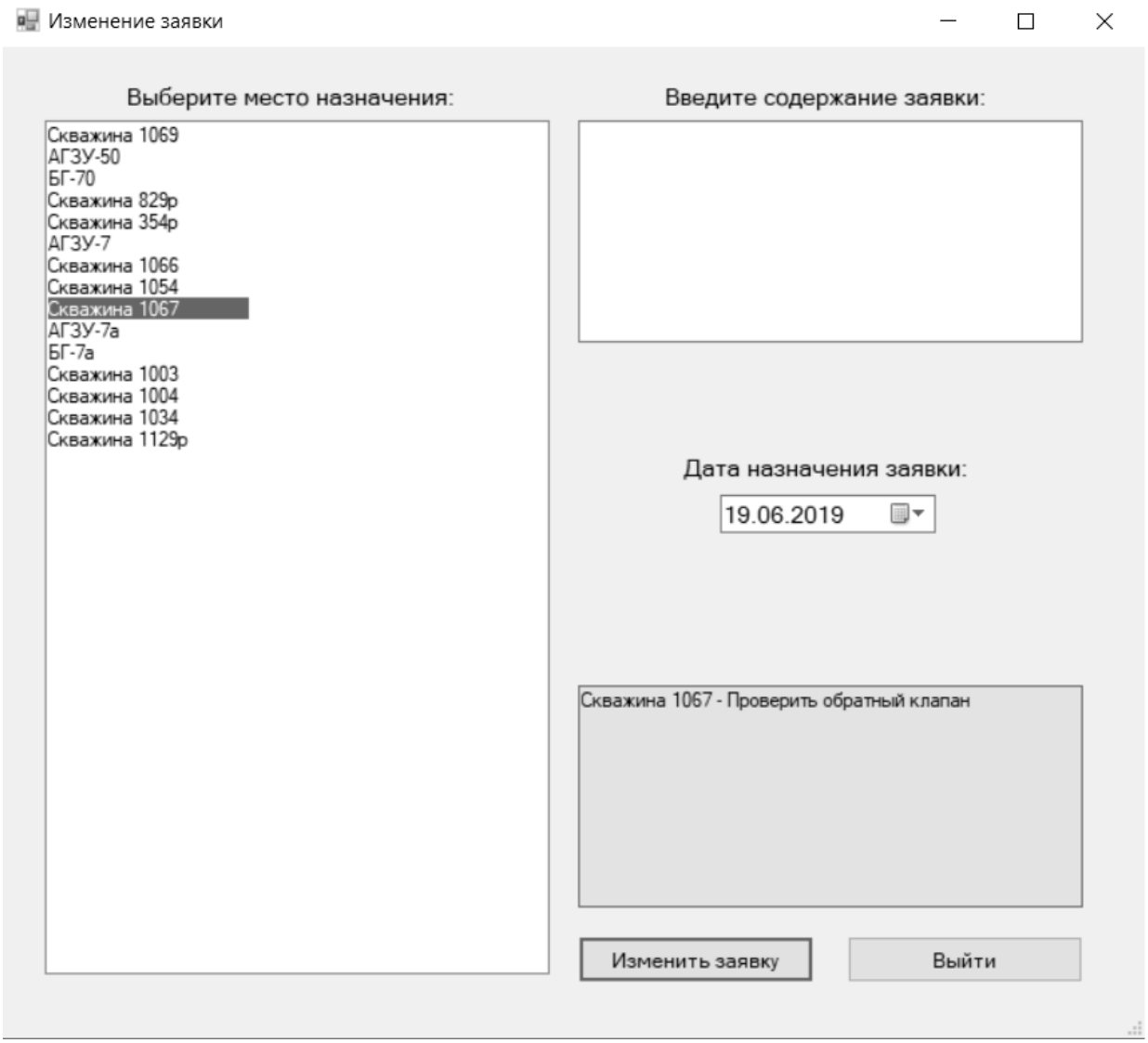

Рисунок 3.5 — Форма изменения заявки

В форме изменения заявки пользователю выводится выбранная в основной форме заявка, которую он может изменить. Для изменения пользователю необходимо изменить хотя бы один из параметров: место назначения, текст содержания или дату назначения, после чего нажать кнопку «Изменить заявку».

Листинг 3.12 — Добавление заявки

```
str = textBox_Subject.Text;
date = DateTime.Now;

if ((listBox_Positions.SelectedIndex < 0) |
(String.IsNullOrEmpty(textBox_Subject.Text)))
  {
     MessageBox.Show("Необходимо корректно заполнить заявку!",
     "Ошибка добавления заявки", MessageBoxButtons.OK,
```

```
    MessageBoxIcon.Error);
    return;
  }

  AddRequest(senderID, date, dateTimePicker1.Value, idx, str);
  str = "";
  listBox_Positions.ClearSelected();
  textBox_Subject.Text = "";
  textBox_Result.Text = "";
  labelAlert.Visible = true;
  timer.Start();
```

В листинге 3.12 видно, что добавление заявки происходит с помощью вызова функции AddRequest. Параметры этой функции:

1. senderID — id пользователя, добавляющего заявку, необходим для получения фамилии;
2. senderDate — текущая дата и время формирования заявки;
3. targetDate — дата назначения заявки;
4. positionID — индекс места назначения;
5. subject — содержание заявки.

Сама функция является хранимой процедурой на SQL сервере. Код процедуры приведён в листинге 3.13.

Изменение заявки выполняется с помощью похожей на AddRequest функции UpdateRequest (код приведён в листинге 3.14), которая имеет следующие параметры:

1. senderDate;
2. targetDate;
3. positionID;
4. subject;
5. id — индекс выбранной заявки в форме работы с заявками.

Листинг 3.13 — Хранимая процедура SQL по добавлению заявки

```
USE [Request]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
```

```
GO
ALTER PROCEDURE [db_worktables].[AddRequest]
     @senderID int,
     @senderDate DateTime,
     @targetDate DateTime,
     @positionID int,
     @subject nvarchar(max)
AS
     INSERT INTO View_requests_2(SenderID, SenderDate, TargetDate,
     PositionID, Subject)
     VALUES (@senderID, @senderDate, @targetDate, @positionID,
     @subject)
```

Процедура по добавлению заявки AddRequest заносит значения в представление View_requests_2, код которого представлен в листинге 3.15.

Листинг 3.14 — Хранимая процедура SQL по изменению заявки

```
USE [Request]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [db_worktables].[UpdateRequest]
     @senderDate DateTime,
     @targetDate DateTime,
     @positionID int,
     @subject nvarchar(max),
     @id int
AS
     UPDATE db_worktables.Requests SET SenderDate = @senderDate,
            TargetDate = @targetDate, PositionID = @positionID,
            Subject = @subject WHERE id = @id;
```

Процедура UpdateRequest обновляет таблицу db_worktables.Request введёнными значениями переменных.

Листинг 3.15 — Представление SQL db_worktables.View_requests_2

```
SELECT r.id, r.SenderDate, r.TargetDate, r.SenderID,
TRIM(pr.Surname) AS Sender, r.PositionID, ps.Position, r.Subject,
r.PerformerID, TRIM(pr1.Surname) AS Performer, r.RecieveDate,
r.PerformDate, r.Report
FROM db_worktables.Requests AS r INNER JOIN db_worktables.Persons AS
pr ON r.SenderID = pr.id INNER JOIN db_worktables.Positions
```

```
AS ps ON r.PositionID = ps.id LEFT OUTER JOIN db_worktables.Persons
AS pr1 ON r.PerformerID = pr1.id
```

В представлении View_requests_2 получаем следующую информацию (рисунок 3.6): id, SenderDate, TargetDate, SenderID, Sender, PositionID, Position, Subject, PerformerID, Performer, RecieveDate, PerformDate, Report.

	id	SenderDate	TargetDate	SenderID	Sender	PositionID	Position	Subject	PerformerID	Performer	RecieveDate	PerformDate	Report
1	2	2019-06-23 18:33:12.947	2019-06-05 21:26:03.000	1	Лильбок	4	Скважина 829р	Нет замеров	NULL	NULL	NULL	NULL	NULL
2	26	2019-06-23 18:29:57.453	2019-06-14 21:36:06.000	1	Лильбок	3	БГ-70	Завышенное давление	NULL	NULL	NULL	NULL	NULL
3	27	2019-06-23 18:30:54.020	2019-06-14 10:38:15.000	1	Лильбок	2	АГЗУ-50	Проверить ПСМ	NULL	NULL	NULL	NULL	NULL
4	28	2019-06-23 18:31:21.063	2019-06-14 10:38:15.000	1	Лильбок	4	Скважина 829р	Снять ручной замер	NULL	NULL	NULL	NULL	NULL
5	35	2019-06-23 18:31:52.400	2019-06-14 13:06:30.000	1	Лильбок	1	Скважина 1069	Померить линейное давление	NULL	NULL	NULL	NULL	NULL
6	36	2019-06-23 18:32:25.417	2019-06-15 17:13:00.000	1	Лильбок	4	Скважина 829р	Отбор контрольной пробы	NULL	NULL	NULL	NULL	NULL
7	37	2019-06-23 18:32:48.580	2019-06-15 17:13:00.000	1	Лильбок	1	Скважина 1069	Сделать скребкование	NULL	NULL	NULL	NULL	NULL
8	38	2019-06-23 18:33:52.033	2019-06-16 17:13:00.000	1	Лильбок	2	АГЗУ-50	Перевести ПСМ на замер 829р	NULL	NULL	NULL	NULL	NULL
9	39	2019-06-23 18:34:22.063	2019-06-16 17:13:00.000	1	Лильбок	3	БГ-70	Заниженное давление	NULL	NULL	NULL	NULL	NULL
10	40	2019-06-23 18:35:03.287	2019-06-16 17:13:00.000	1	Лильбок	1	Скважина 1069	Снять показания на регистратор	NULL	NULL	NULL	NULL	NULL
11	43	2019-06-23 19:35:50.983	2019-06-18 19:33:32.000	1	Лильбок	9	Скважина 1067	Проверить обратный клапан	NULL	NULL	NULL	NULL	NULL

Рисунок 3.6 — Представление View_requests_2

Для удаления одной или нескольких заявок пользователю нужно выделить необходимые заявки в основной форме работы с заявками и нажать кнопку «Удалить заявку». Удаление заявок выполняется также с помощью хранимой процедуры SQL (код представлен в листинге 3.16).

Листинг 3.16 — Хранимая процедура SQL по удалению заявок

```
USE [Request]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [db_worktables].[DeleteRequest]
     @id int
AS
     DELETE FROM Requests WHERE (@id = id)
```

Процедура имеет лишь один параметр id, который получает при выделении заявок.

Если было выделено несколько заявок на удаление, процедура удаления вызывается в цикле для каждой заявки.

В функционал пользователя с правами доступа «Оператор» входит возможность отчитаться по выполненным заявкам.

Для того, чтобы создать отчёт по заявке, пользователю необходимо нажать кнопку «Отчитаться». Отчёт добавляется с помощью хранимой процедуры SQL (код приведён в листинге 3.17).

Листинг 3.17 — Хранимая процедура отчёта по выполненным заявкам

```
USE [Request]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [db_worktables].[AddReport]
     @report nvarchar(max),
     @recieve_date DateTime,
     @perform_date DateTime,
     @performerID int,
     @idx int
AS
     UPDATE db_worktables.Requests SET Report = @report, PerformDate
= @perform_date, RecieveDate = @recieve_date,
PerformerID = @performerID
           WHERE id = @idx;
```

В листинге 3.17 видно, что процедура обновляет таблицу заявок такими данными как, отчёт, дата принятия и выполнения заявки, фамилией исполнителя.

После того, как отчёт по выполненной заявке сделан, основная форма работы с заявками пополняется информацией об исполнителе, дате выполнения заявки и самим отчётом (рисунок 3.7).

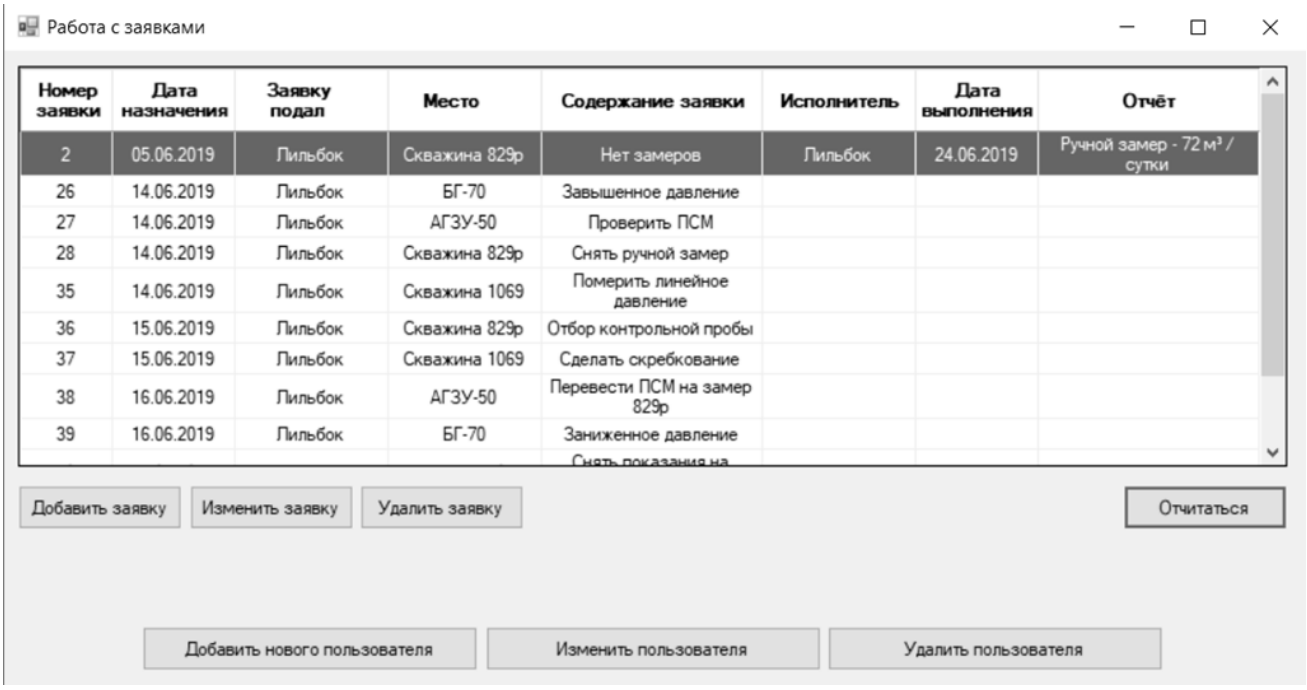

Рисунок 3.7 — Выполненная заявка

3.1.4 Создание Web-системы

С помощью ASP.NET MVC создана Web-система для внедрения возможности удалённо (с мобильных устройств) создавать отчёты по заявкам.

Система связана с той же базой данных, которая используется в десктопном приложении.

Для использования системы достаточно иметь мобильное устройство с выходом в Интернет. Получить доступ к системе возможно с помощью аккаунта, созданного администратором системы и любого браузера.

После успешной авторизации, пользователю выводится информация по заявкам, точно та же, что и в созданном десктопном приложении. Пользователю доступны две функции:

- просмотр заявок;
- создание отчётов по заявкам.

С течением времени база данных будет пополняться и увеличиваться, поэтому в Web-системе реализована постраничная загрузка информации, чтобы уменьшить нагрузку на сеть пользователя.

Идея создания Web-системы заключается в следующем — операторы (исполнители заявок) работают по заявкам на выезде, далеко от компьютеров. Возможность создать отчёт по выполненным заявкам с одним десктопным приложением у них будет лишь в обеденный перерыв и вечером, в конце рабочей смены. Web-система решает эту проблему тем, что оператор, после выполнения заявки, может зайти в систему со своего мобильного телефона и отправить отчёт по заявке, прямо на месте работ.

Web-система значительно снизит время составления отчётов по выполненным заявкам, даст возможность геолого-технологической службе и операторам пульта управления автоматики и телемеханики просматривать отчёты сразу после выполнения заявок, а не дожидаться, пока оператор вернётся на базу и составит отчёт. Web-система также уменьшит количество случаев, когда

оператор, не своевременно получая заявку, будет вынужден ехать в одно и то же место назначения несколько раз за рабочую смену. Например, оператор выезжает по заявке на скважину № 1069, выполняет заявку и уезжает на другую удалённую скважину, после чего ему звонит мастер и сообщает, что поступила ещё одна заявка на скважину № 1069, оператор будет вынужден терять время и ехать туда ещё один раз. При использовании Web-системы оператор сможет раньше увидеть новую заявку и спланировать свой маршрут и свою работу.

Вывод по третьему разделу

Создано приложение по учёту и регистрации заявок. Создана Web-система, которая значительно экономит время на создание отчётов и просмотра информации. Десктопное приложение и Web-система полностью решают все поставленные задачи.

ЗАКЛЮЧЕНИЕ

В результате выпускной квалификационной работы разработана система учета заявок на предприятии ООО «Лукойл-Западная Сибирь» в цехе по добыче нефти и газа № 3.

Для достижения цели данной работы решены следующие задачи: изучение структуры работы цеха, выявление его основных задач и функций, построение логической и физической структуры, разработка базы данных.

Поставленные задачи решены.

Разработка позволила достичь следующих результатов:

- уменьшение времени, затрачиваемого на поиск информации о заявках;
- уменьшение времени, затрачиваемого на создание отчетов по проделанной работе;
- контроль выполнения заданий специалистом.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. [Электронный ресурс]// ООО «ЛУКОЙЛ-Западная Сибирь» - Режим доступа: http://zs.lukoil.ru/ru/
2. Панюкова Т. А., Панюков А. В. Языки и методы программирования. Путеводитель по языку C++; Либроком - Москва, 2014. - 216 с.;
3. Петров В.Н. Информационные системы - СПб.: Питер, 2003. - 688 с.;
4. Дюбуа П. MySQL, 2004, - 165 с.;
5. Емельянова Н.З. Основы построения автоматизированных информационных систем: учеб. пособие для сред. проф. образования./ Н.З. Емельянова, Т.Л.;
6. Партыка, И.И. Попов.- м.: ФОРУМ-ИНФРА-М, 2005. – 416;
7. Орлов С. Технологии разработки программного обеспечения: учебник/ - СПБ.: ПИТЕР, 2002.- 647 С.
8. Кузин А. В., Левонисова С. В. Базы данных; Академия - Москва, 2010. - 320 с.;
9. Справочник по C# [Электронный ресурс]//Руководство по языку C#. - Режим доступа:https://docs.microsoft.com/ru-ru/ (06.062019);
10. Основы программирования на C# [Электронный ресурс]// Полное руководство по языку программирования C#. – Режим доступа: https://metanit.com/sharp/tutorial/ (1.06.2019).