МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Кафедра Прикладная математика и информатика
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки, специальности)

Бизнес-информатика
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка информационной системы для обработки заявок пользователей

Студент Р.В. Татаринов
(И.О. Фамилия) (личная подпись)

Руководитель к.э.н., доцент, Т.А. Раченко
(ученая степень, звание, И.О. Фамилия)

Тольятти 2024

Аннотация

Выпускная квалификационная работа посвящена разработке информационной системы для обработки заявок пользователей. Работа состоит из 73 страниц печатного текста, 24 рисунков, 3 таблиц и 27 источников информации.

Объект исследования – деятельность компании, работающей в сфере промышленно гражданского строительства.

Предмет исследования – автоматизация процесса обработки заявок пользователей.

Цель выпускной квалификационной работы – разработка информационной системы для обработки заявок пользователей компании ООО «Гарант».

Работа содержит три главы с описанием хода выполнения работы, введения, заключения о результатах выполнения работы и списка литературы.

Характеристика компании и ее деятельность за последние годы, а также бизнес-процессы, которые осуществляются подразделениями представлено в первой главе работы. Бизнес-процессы представлены в виде диаграмм на разных уровнях декомпозиции до и после автоматизации.

Вторая глава описывает шаги, которые были предприняты при концептуальном проектировании информационной системы. Рассмотрены документы, которые поступают в отдел стратегического развития и методы их обработки.

В третьей главе описывается архитектура проекта и способы реализации. Представлен контрольный пример информационной системы и проведен расчет экономической эффективности проекта.

Заключение демонстрирует все выполненные задачи во время выполнения выпускной квалификационной работы.

Abstract

The final qualifying work is devoted to the development of an information system for processing user requests. The work consists of 73 pages of printed text, 24 figures, 3 tables and 27 sources of information.

The object of the study is the activities of a company operating in the field of industrial civil engineering.

The subject of the research is the automation of the process of processing user requests.

The purpose of the final qualifying work is to develop an information system for processing applications from users of the company Garant LLC.

The work contains three chapters describing the progress of the work, introduction, conclusion on the results of the work and a list of references.

The characteristics of the company and its activities in recent years, as well as the business processes carried out by the divisions, are presented in the first chapter of the work. Business processes are presented in the form of diagrams at different levels of decomposition before and after automation.

The second chapter describes the steps that were taken in the conceptual design of the information system. The documents that are received by the strategic development department and methods for their processing are considered.

The third chapter describes the project architecture and implementation methods. A test example of an information system is presented and the economic efficiency of the project is calculated.

The conclusion demonstrates all the tasks completed during the final qualifying work.

Содержание

Введение 5
1 Функциональное моделирование деятельности ооо «Гарант» 6
1.1 Обзор предметной области 7
1.2 Концептуальное проектирование предметной области 9
1.3 Анализ существующих разработок 15
1.4 Постановка задачи на разработку системы 17
2 Логическое проектирование автоматизированной информационной системы обработки заявок пользователей 19
2.1 Логическая модель информационной системы обработки заявок пользователей 19
2.2 Проектирование базы данных системы обработки заявок пользователей . 23
2.3 Выбор архитектуры информационной системы 26
3 Физическое проектирование автоматизированной информационной системы обработки заявок пользователей 29
3.1 Выбор инструментов разработки 29
3.2 Разработка системы 32
3.3 Разработка информационной системы обработки заявок компании ООО «Гарант» 33
3.4 Тестирование информационной системы 39
Заключение 43
Список используемой литературы и используемых источников 44
Приложение А Листинг программы 47

Введение

Данная выпускная квалификационная работа актуальна в современном рынке строительных услуг, где эффективная обработка заявок пользователей является ключевым фактором успеха.

Актуальность работы - эффективная обработка заявок пользователей, которая является ключевым фактором успеха в строительной компании. В условиях конкуренции и высоких требований клиентов, автоматизация этого процесса становится необходимостью для сокращения ошибок, улучшения качества обслуживания и повышения профессионализма сотрудников.

Централизованное хранение информации учета заявок облегчит доступ и поиск данных, а постоянное развитие информационных технологий делает разработку информационной системы для обработки заявок пользователей неотъемлемой и актуальной задачей.

Разработанная информационная система для обработки заявок от пользователей позволит снизит количество ошибок, допускаемых менеджерами при обработке заявок, и увеличит скорость обслуживания пользователей, также позволит просматривать готовые заявки, прикрепленные к письму при отправлении на обработку.

Целью данной работы будет автоматизация процесса обработки заявок пользователей путем создания информационной системы.

Объект исследования – деятельность компании, работающей в сфере промышленно гражданского строительства.

Предмет исследования – автоматизация процесса обработки заявок пользователей.

Задачи выпускной квалификационной работы:

– провести исследование деятельности строительной фирмы, включая анализ ее бизнес-процессов и основных операций;

– идентифицировать процессы, требующие автоматизации, и определить наиболее приоритетные с точки зрения повышения эффективности и сокращения ошибок;

– провести концептуальное проектирование информационной системы, включающее определение функциональных требований, структуры базы данных и взаимодействия пользователей с системой;
– реализовать информационную систему для обработки заявок пользователей строительной компании;
– рассчитать экономическую эффективность разработки и внедрения информационной системы.

В ходе работы применялись различные программы для проектирования и моделирования, для проектирования организационной структуры был использован бесплатный сервис для моделирования draw.io, проектирование диаграмм процессов и их декомпозиция осуществлялась с помощью программы AllFusion ERwin Data Modeler, программный продукт StarUML для проектирования диаграмм на основе объектно-ориентированного подхода.

Работа включает три главы в которых описаны все пункты по изучению, проектированию и разработке информационной системы обработки заявок пользователей.

Характеристика компании и ее деятельность за последние годы, а также бизнес-процессы, которые осуществляются подразделениями представлено в первой главе работы. Бизнес-процессы представлены в виде диаграмм на разных уровнях декомпозиции до и после автоматизации.

Вторая глава описывает шаги, которые были предприняты при концептуальном проектировании информационной системы. Рассмотрены документы, которые поступают в отдел стратегического развития и методы их обработки.

В третьей главе описывается архитектура проекта и способы реализации. Представлен контрольный пример информационной системы и проведен расчет экономической эффективности проекта.

Заключение демонстрирует все выполненные задачи во время выполнения выпускной квалификационной работы.

1 Функциональное моделирование деятельности ООО «Гарант»

1.1 Обзор предметной области

ООО «Гарант» - является строительной организацией, которая осуществляет по заказу общестроительные работы в промышленно-гражданском строительстве.

Цель предприятия – получение максимальной прибыли от выполняемых работ в сфере строительства.

Задачи предприятия – эффективное использование технических (машин и механизмов), производственных и трудовых ресурсов, совершенствование управление предприятием с целю обеспечения качественного выполнения работ.

Основные направления деятельности:

– строительство жилых и нежилых зданий;

– строительство автомобильных дорог и автомагистралей;

– строительство прочих инженерных сооружений, не включённых в другие группировки;

– разборка и снос зданий;

– производство земляных работ;

– разведочное бурение;

– производство электромонтажных работ;

– производство санитарно-технических работ, монтаж отопительных систем и систем кондиционирования воздуха;

– производство прочих строительно-монтажных работ;

– производство штукатурных работ;

– работы столярные и плотничные;

– работы по устройству покрытий полов и облицовке стен;

– производство малярных и стекольных работ;

– производство прочих отделочных и завершающих работ;

– производство кровельных работ;

– работы строительные, специализированные прочие, включённые в другие группировки;

– работы гидроизоляционные.

В состав организационной структуры предприятия входит отдел стратегического развития информационных технологий.

Основная цель отдела производственно-технического отдела – обеспечение строительного производства, оценка текущего строительного рынка, подготовка закупочной документации, сопровождение строительного производства до ввода объекта в эксплуатацию. Стратегическое планирование и развитие, поддержание предприятия в сфере строительства и повышение качества строительного производства для заказчика.

Как и все современные строительные компании ООО «Гарант» внедряет новейшие информационные технологии для оптимизации строительной деятельности, управления заказами и подготовке отчётной, и исполнительной документации в рамках исполнения договоров и контрактов, предусмотренных законодательством Российской Федерации.

На рисунке 1 показана организационная структура компании ООО «Гарант».

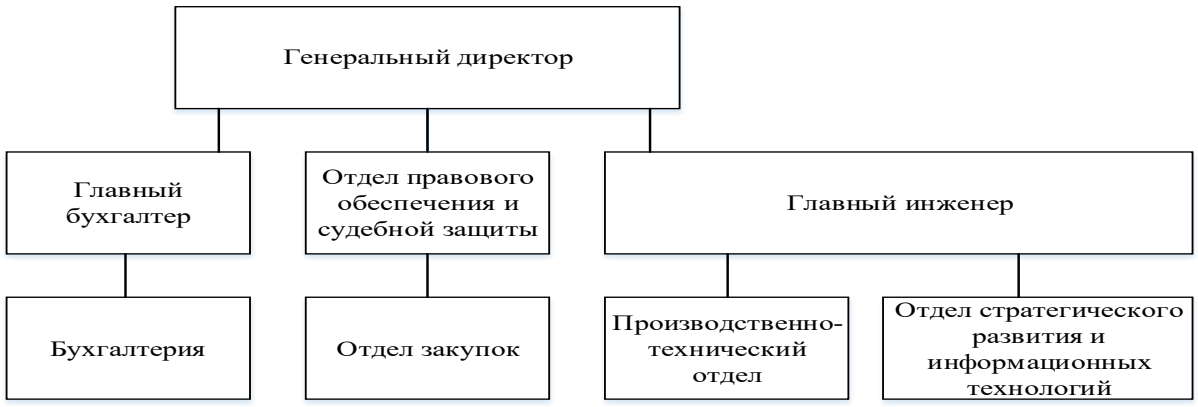

Рисунок 1 – Организационная структура предприятия ООО «Гарант»

Объектом рассмотрения в данной организационной схеме является – отдел стратегического развития и информационных технологий (далее – Отдел).

Субъектом управления является непосредственно руководитель отдела, и опосредованно – главный инженер и генеральный директор.

Особенности функционирования рассматриваемого объекта. В процессе выполнения основной функции Отдел взаимодействует с производственно-техническим отделом. Главный инженер компании осуществляет управление отделом и при необходимости задействование смежных отделов через генерального директора.

Основным технико-экономическими показателями бизнес-процесса как объекта управления являются извлечение прибыли от выполняемых работ в сфере строительства и максимальная автоматизация бизнес-процессов.

1.2 Концептуальное проектирование предметной области

Концептуальное моделирование деятельности компании или отдельного процесса подразумевает создание абстрактной модели процессов компании. На основе моделей можно выделить входящие и исходящие документы описываемого процесса.

Концептуальное проектирование затрагивает выполнение определенных моделей, которые выполняются в различных программах. Опишем самые распространённые и наиболее применяемое программные продукты для данного моделирования: BPMN, ARIS, Microsoft Visio.

BPMN (Business Process Model and Notation) довольно старенькая разработка, которая позволяет проводить моделирование процессов в разных нотациях. Программа позволяет создавать графические диаграммы, проводить декомпозиции процессов и формировать на основе их различные отчеты.

ARIS Express представляет собой удобный инструмент для моделирования различных диаграмм и бизнес-процессов, которые допускаются в программе.

Microsoft Visio имеет большой ассортимент инструментов для моделирования различных диаграмм и процессов. Инструменты программы довольно легко применяются на практике и готовые рисунки легко импортируются.

Процессы компании не представляют какую-то сложную структуру, поэтому для описания их и моделирования достаточно будет применения нотации IDEF0.

Набор методологий IDEF самый распространенный из инструментов моделирования и широко применяется обычными пользователями не имеющих опыт в моделировании.

Для анализа процесса построим диаграммы в нотации IDEF0, которая наглядно показывает какие процессы выполняются при обработке заявок пользователей, также можно определить какой из подпроцессов тормозит весь процесс и разработать методы по усовершенствованию процесса.

Первоначально будет создана контекстная диаграмма процесса обработки заявок «как есть», компании ООО «Гарант». Все диаграммы будут разработаны в программе BpWin 7.0, с помощью которой будет проведено моделирование процесса на разных уровнях и показаны все участники процесса. Рисунок 2 демонстрирует контекстную диаграмму процесса обработки заявок пользователей.

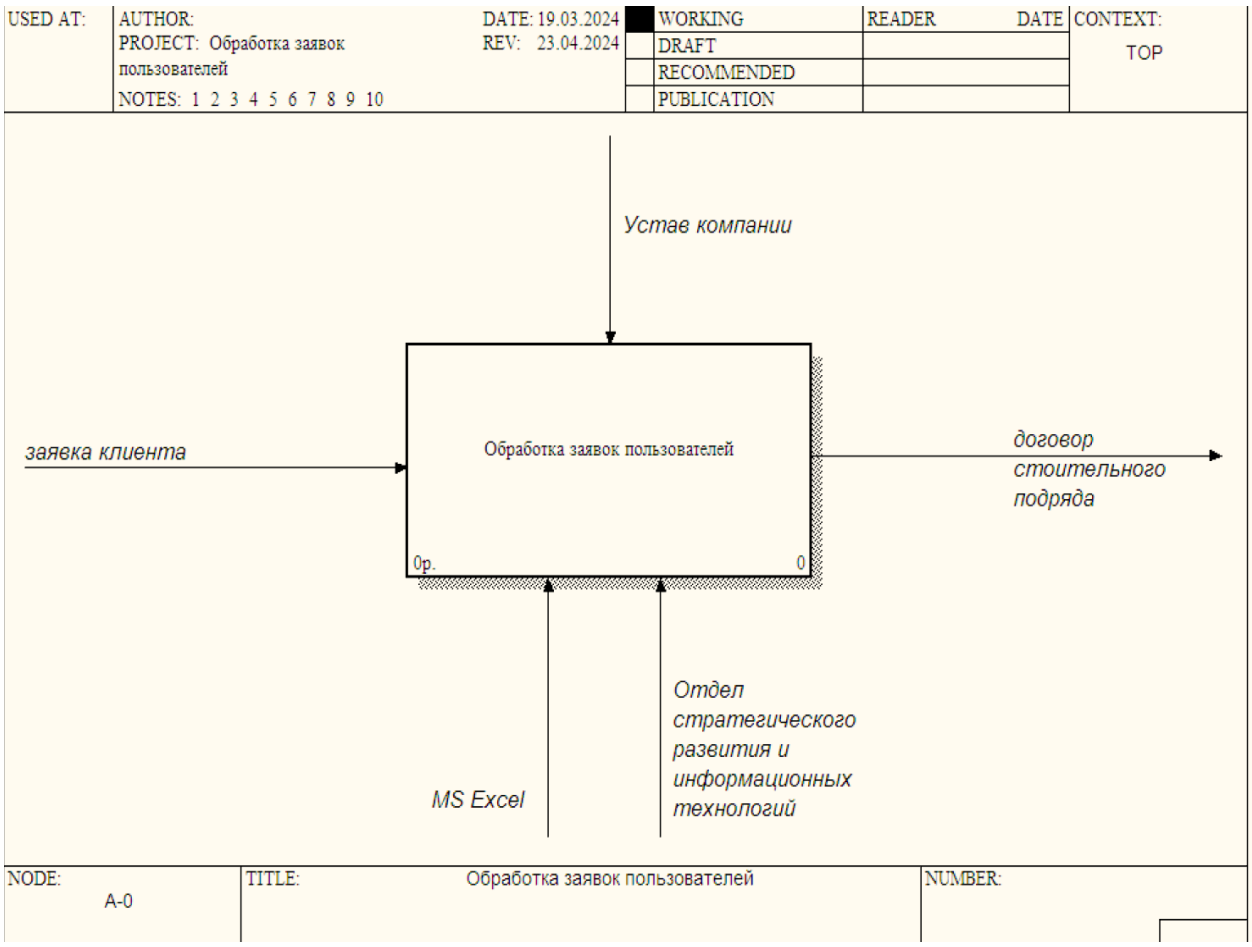

Рисунок 2 – Контекстная диаграмма процесса «Обработка заявок пользователей»

Из диаграммы можно увидеть, что обработка заявок пользователей начинает выполняться после того, как поступает заявка клиента. Клиентами могут быть как физические, так и юридические лица. Итогом процесса будет заключенный договор строительного подряда с клиентом. Все процессы будут выполняться в отделе стратегического развития и информационных технологий компании ООО «Гарант».

Выполним декомпозицию процесса, разделив процесс обработки заявок на подпроцессы:

– первоначальная обработка;

– формирование заказа;

– передача предложения;

– обсуждение заявки;

– фиксирование обязательств.

На рисунке 3 представлена диаграмма декомпозиции процесса обработки заявок пользователей в компании.

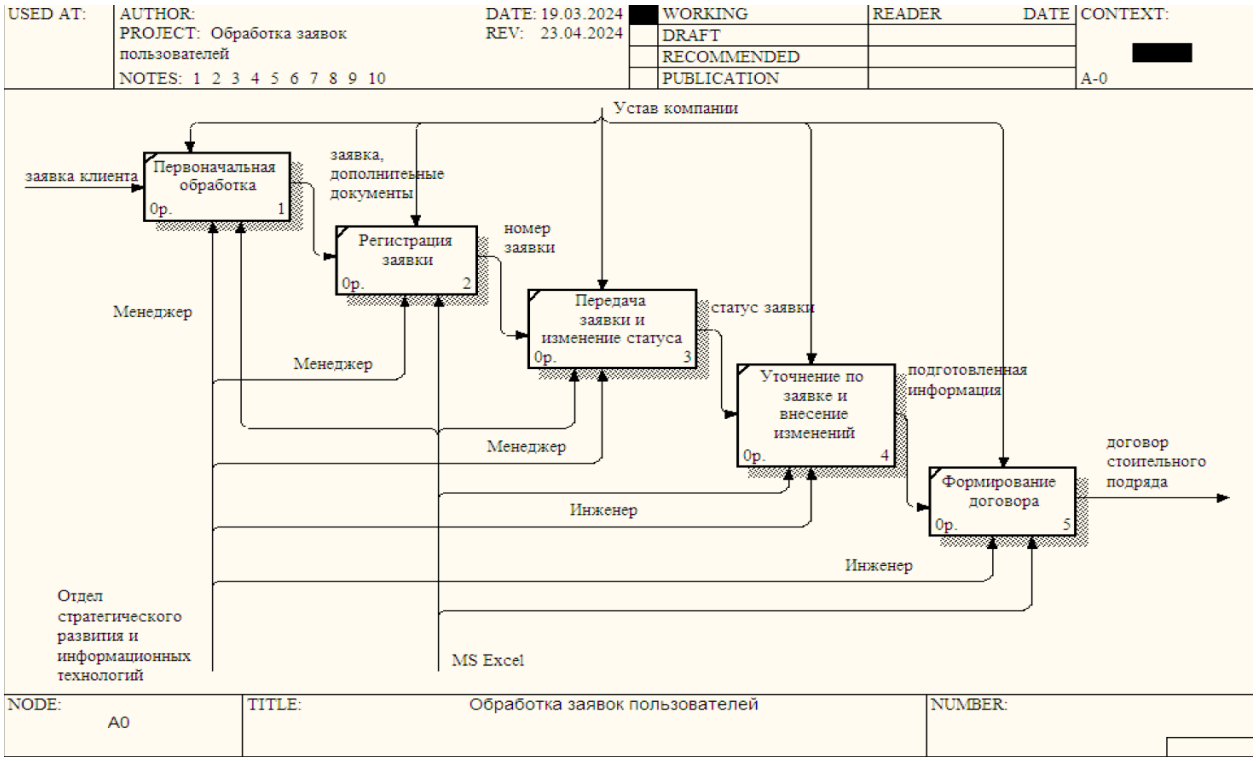

Рисунок 3 – Диаграмма декомпозиции процесса «Обработка заявок пользователей» до автоматизации

Как видно процесс обработки заявок пользователей проходит несколько этапов прежде, чем будет заключен договор строительного подряда с клиентом. После первоначальной обработки происходит формирование заказа, предоставляется дополнительная информация клиентам по услугам компании и обсуждаются условия, требования и бюджет заявки.

На основании вышеизложенных пунктов подписывается договор строительного подряда в двух экземплярах.

Так как сейчас обработка заявок происходит в ручном режиме на которую затрачивается очень много времени. Также такая обработка допускает наличие

ошибок при расчетах и формировании документов. Информационная система обеспечит быструю и эффективную обработку заявок, позволит хранить данные в одном месте и получать актуальные данные в любой момент.

На рисунке 4 показана контекстная диаграмма процесса обработки заявок пользователей после автоматизации.

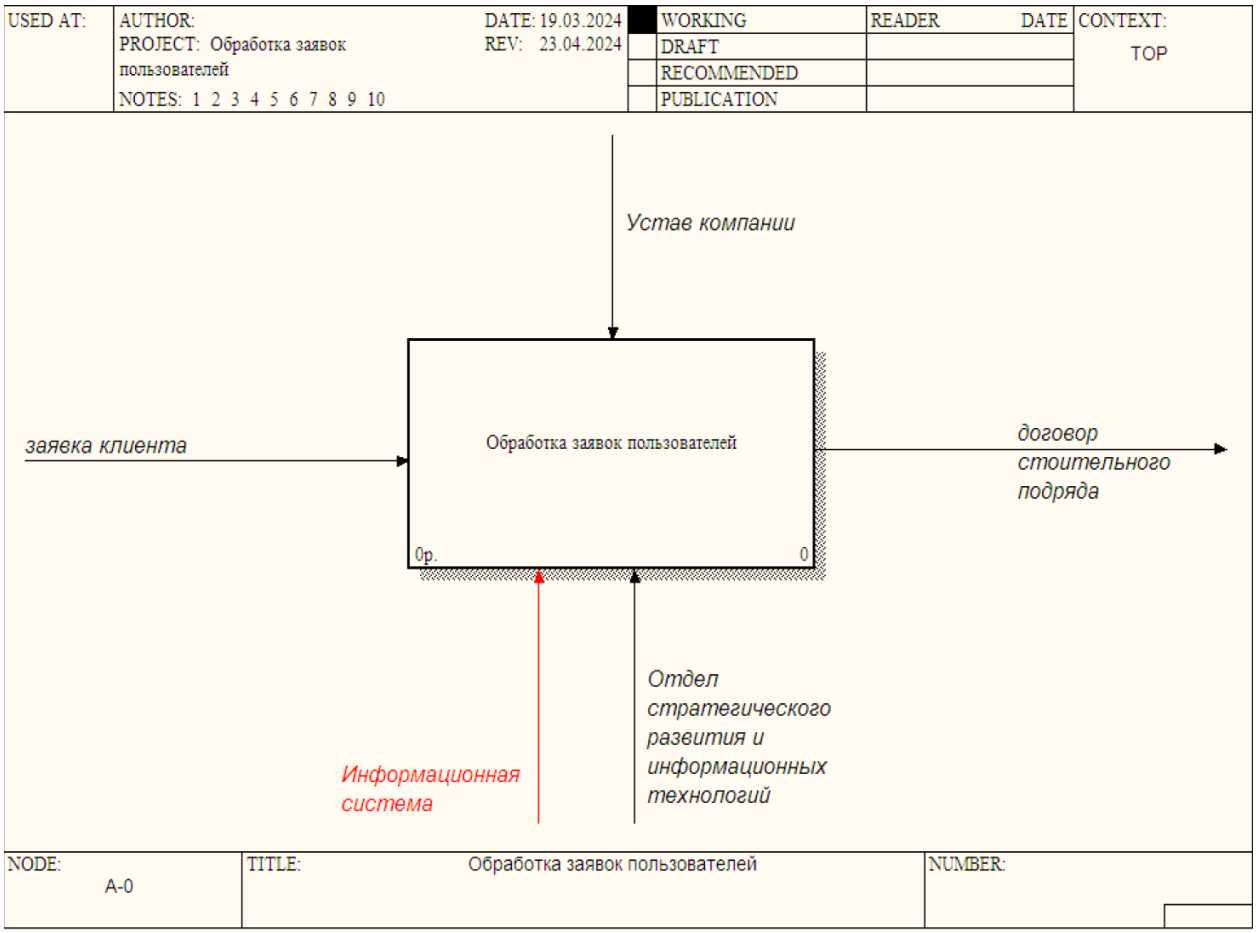

Рисунок 4 – Контекстная диаграмма процесса «Обработка заявок пользователей»

На рисунке 4 видно, что в процессе появилась информационная система, в которой и будет осуществляться процесс обработки заявок пользователей.

Покажем диаграмму декомпозиции этого процесса после автоматизации, в соответствии с рисунком 5.

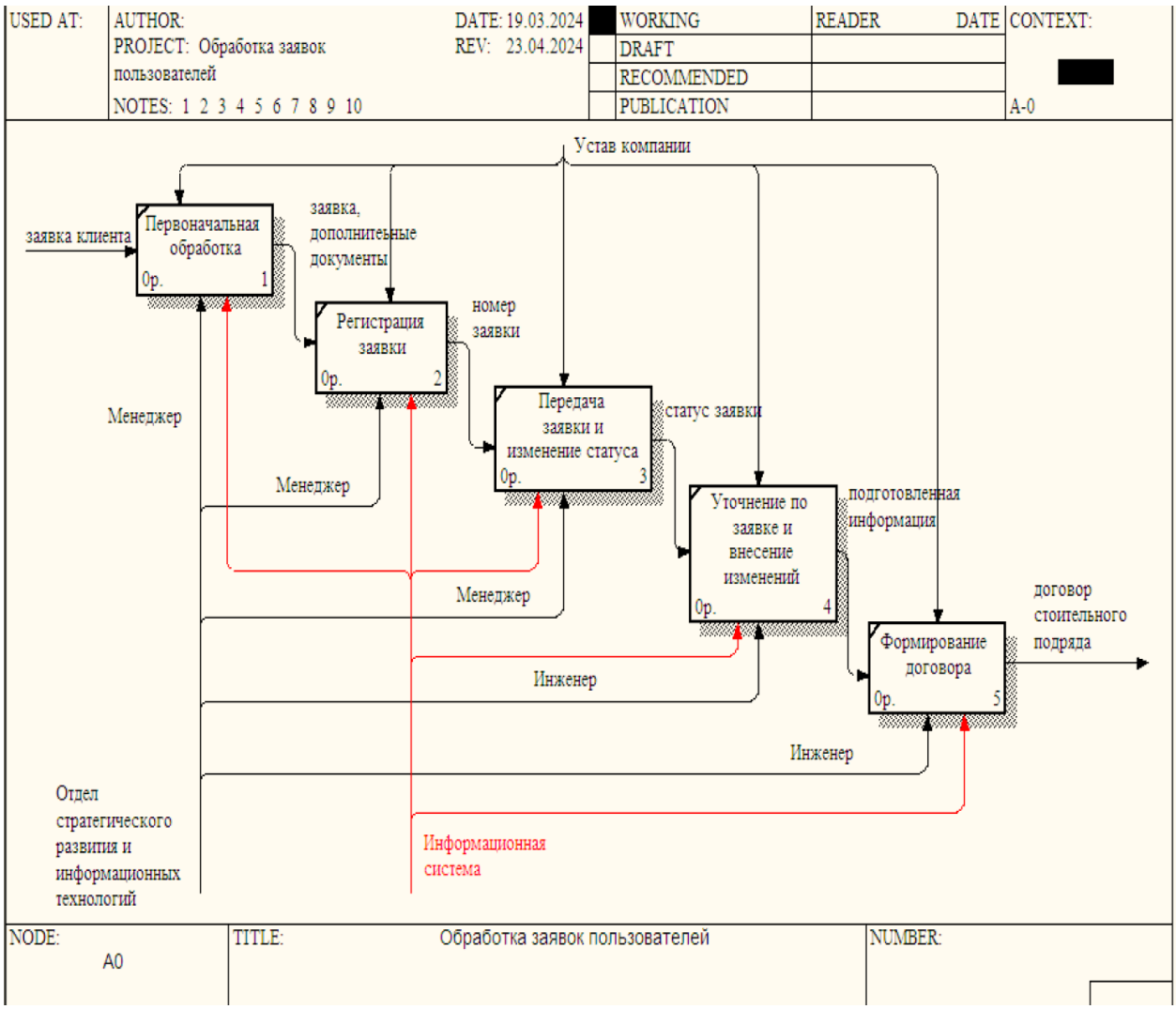

Рисунок 5 – Диаграмма декомпозиции процесса «Обработка заявок пользователей» после автоматизации

На диаграмме декомпозиции процесса обработки заявок пользователей изменились подпроцессы с учетом автоматизации. После первоначальной обработки заявки, можно прикреплять дополнительные документы к заявке прямо в системе и регистрировать заявку, чтобы она никуда не пропала. Также заявке можно будет менять статус и отправлять дальше для обработки и формирования необходимых документов дял заключения договора с клиентом.

Процесс обработки заявок, поступивших от клиентов компании занимает достаточно много времени у менеджера компании, а затем и инженера, который занимается обработкой заявок.

Заявки, которые поступают на бумажных носителях имеет риск не дойти до исполнителя или потеряться на этапе перераспределения заявки внутри отдела. Также заявка не имеет истории обработки, при желании просмотра деталей заявки нет такой возможности и просмотра дополнительной информации.

После обработки заявки могли утилизироваться по истечении некоторого времени, чтобы не загружать архив компании, автоматизация данного процесса позволит сохранять все заявки в едином хранилище данных, доступ которому будет открыт круглосуточно.

Пользователи системы получат возможность контролировать процесс обработки заявками и оповещать клиента при запросе выполнения заявки имея данные о ее статусе выполнения.

Функции системы должны обеспечивать выполнение действий:

– добавление новых данных о заявках клиентов;

– добавление документа к оформленной заявке;

– сохранение документов;

– доступ к изменению статуса заявки.

Нефункциональные требования:

– наличие интерфейса;

– меню на русском языке;

– отклик системы не более 5 с.

Эта лишь основные требования, которыми система должна обладать чтобы эффективно выполнять процесс обработки заявок пользователей компании.

1.3 Анализ существующих разработок

Сравним три аналогичные информационные системы, которые выполняют функции по обработке заявок пользователей в разных сферах.

Salesap. Новый российский программный продукт, который позволяет провести комплексную работу по наведению порядка в бизнесе – начиная от создания базы клиентов и заканчивая финансовым учетом. На первоначальных этапах программа была разработана для сотрудников call-центров, позже программу расширили дополнительными функциями, которые позволили проводить анализ и формирование отчетов по процессам компании.

Основные характеристики:

– количество контактов до 100;

– создание хранилищ;

– учет и распределение задач на сотрудников;

– учет всех продаж компании;

– учет прибыли;

– корпоративный портал.

Плюсы и минусы:

– доступное меню для пользователя;

– база знаний;

– готовые формы для ввода данных;

– существует бесплатная и платная версия.

Pyrus еще одна достойная система с бесплатной версией.

Пользователям дают настроить маршрут оформления отдельной заявки до конкретного менеджера или разделять задачи под разных сотрудников.

Функции системы:

– создание списка задач;

– создание новых пользователей;

– сохранение информации;

– совместимость с другими система;

– составление отчетов.

Плюсы и минусы:

– ограниченная бесплатная версия;

– нет консультаций при установке ПО.

Итилиум — это программное решение для управления ИТ-услугами на платформе 1С, поставляемое с открытым исходным кодом.

Итилиум автоматизирует функции ключевых процессов:

– добавление и удаление услуг;

– отслеживание и реагирование на прецеденты;

– управление запросами на обслуживание.

Рассмотренные выше информационные системы показали, что имеют множество функций и некоторые из них мобильное приложение, однако для автоматизации процесса обработки заявок пользователей в строительной компании не требуется многофункциональная система с мобильным приложением. Поэтому целесообразно будет создать собственную систему для решения задач отдела стратегического развития и информационных технологий.

1.4 Постановка задачи на разработку системы

Автоматизированная система создается как система обработки заявок пользователей. Данная система представляет собой набор данных, которые позволяют осуществлять прием заявки, формирование заказа, просматривать список заявок, корректировать заявки. Разрабатываемая система позволяет пользователям системы быстро находить нужную информацию по заказам и вести их учет. Используется система в сфере услуг.

Основная цель разрабатываемой системы – упростить рабочий процесс сотрудников отдела стратегического развития и информационных технологий.

Система предоставляет возможность хранить данные о заявках, клиентах, закрытых заказах, договорах. Кроме того, система позволяет учитывать и обрабатывать все поступившие заявки.

Автоматизированная система будет создаваться как основная система для обработки заявок в отделе стратегического развития и IT- технологий. Система позволит работать одновременно нескольким пользователям. Пользователь,

который будет обрабатывать заявки получит возможность видеть все заявки, которые поступили, изучить детали заявки, посмотреть документы, которые будут прикреплены к заявке и обработать ее, присваивая ей статус на всем этапе обработки.

Система будет обладать функциями:

– учет поступающих заявок в отдел;

– регистрация данных клиента;

– заполнения данных по договору;

– ведение заявки на всем этапе до закрытия.

Интерфейс пользователя должен быть доступным для сотрудников и иметь навигацию в виде кнопок, которые должны обеспечивать открытия форм или отчетов по клику пользователя.

Основными пользователями будут сотрудники отдела, администратор системы.

Взаимодействие между пользователем системы и ее модулями необходимо сетевое оборудование быстрое и надежное.

Операционная система должна поддерживать Windows Server.

Для написания приложения может быть выбран любой язык программирования высокого уровня.

Интерфейс пользователя должен быть понятным обычному пользователю не имеющего специального образования или навыков.

Архитектура приложения должна позволять масштабировать приложения на основе потребностей строительной компании, в частности отдела стратегического развития и информационных технологий.

Вывод по первой главе.

В первой главе проведено изучение деятельности и основных бизнес-процессов компании ООО «Гарант». Данное изучение помогло правильно провести моделирование процессов и провести их декомпозицию с помощью нотации IDEF0. Диаграммы были проанализированы и выделены процессы, которые необходимо улучшить с помощью автоматизации.

2 Логическое проектирование автоматизированной информационной системы обработки заявок пользователей

2.1 Логическая модель информационной системы обработки заявок пользователей

Существует несколько технологий логического моделирования, рассмотрим несколько из них и выберем которое больше подходит для моделирования приложения.

С помощью UML можно построить различные диаграммы такие как диаграммы классов, вариантов использования и многие другие, отражающие логику программного обеспечения.

Entity-Relationship Diagrams применяется в большей степени при моделировании базы данных их сущностей и связей [1].

BPMN больше используется для моделирования бизнес-процессов организации, такой способ моделирования больше отражает шаги бизнес-процесса и взаимодействие участников в них [7].

Логическое моделирование позволит представить работу системы изнутри и показать взаимодействие пользователей с системой, определить границы системы и функциональность ее.

Создадим диаграмму прецедентов, показывающую весь спектр функций системы. В системе будет два основных пользователя это менеджер и инженер, оба пользователя будут участвовать в процессе обработки заявок в соответствии с рисунком 6.

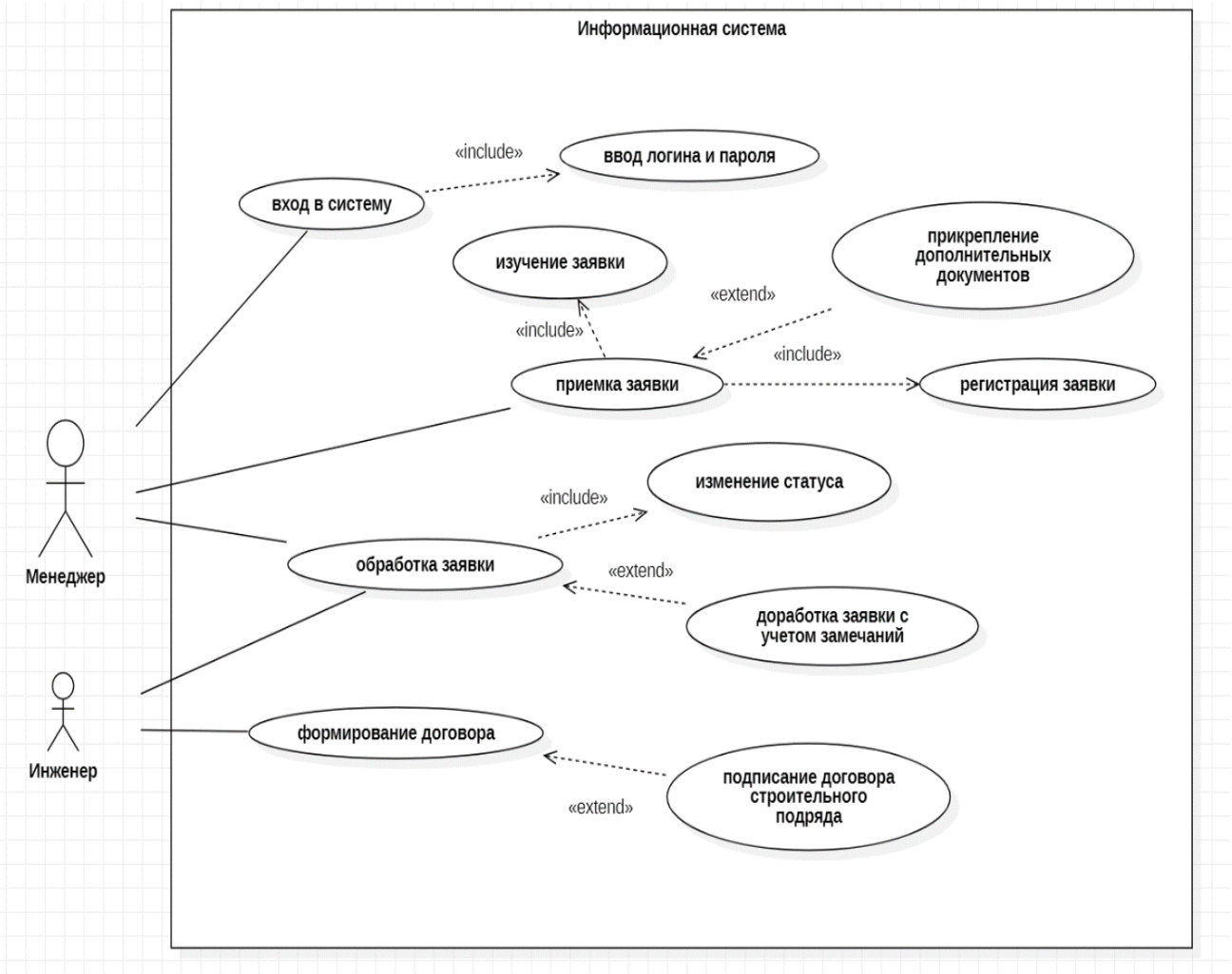

Рисунок 6 – Диаграмма вариантов использования

Функциональные требования:

– менеджер и инженер должны иметь возможность войти в систему;

– менеджер должен иметь возможность внести данные заявки;

– менеджер должен иметь возможность изучить заявку;

– менеджер и инженер должны иметь возможность внести изменения в заявку;

– менеджер должен иметь возможность зарегистрировать заявку;

– менеджер должен иметь возможность прикреплять дополнительные документы к заявке;

– менеджер и инженер должны иметь возможность изменять статус заявки после ее обработки;

– инженер должен иметь возможность формировать договора строительного подряда.

Спроектируем диаграмму состояний для прецедента «Обработка заявки», чтобы понимать какие состояния заявка сможет принимать на всем жизненном этапе. Диаграмма состояния для процесса «Обработка заявки» показана на рисунке 7.

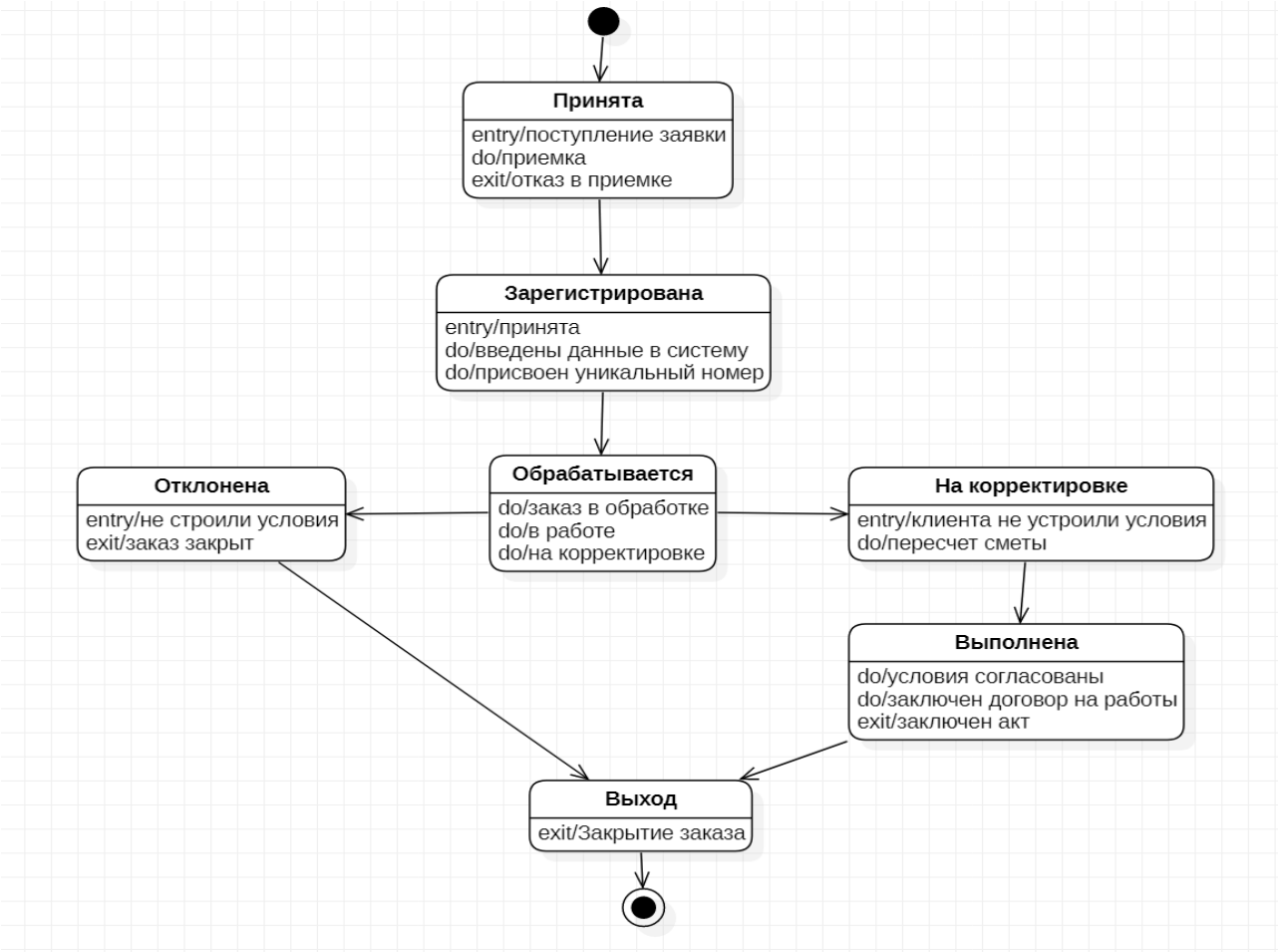

Рисунок 7 – Диаграмма состояния для процесса «Обработка заявки»

Теперь рассмотрим диаграмму последовательности, данная диаграмма позволяет рассмотреть различные варианты использования разрабатываемой системы.

При обработке заявки от клиента, сотрудник проверяет данные заявки, вводит данные по заявке и клиенту в информационную систему. Затем проверяет данные и прикрепляет новые документы (при необходимости), меняет статус заявки и отправляет дальше на обработку. После прохождения и согласования заявки с клиентом, формируется договор строительного подряда. Диаграмма последовательности процесса обработки заявки клиента показана в соответствии с рисунком 8.

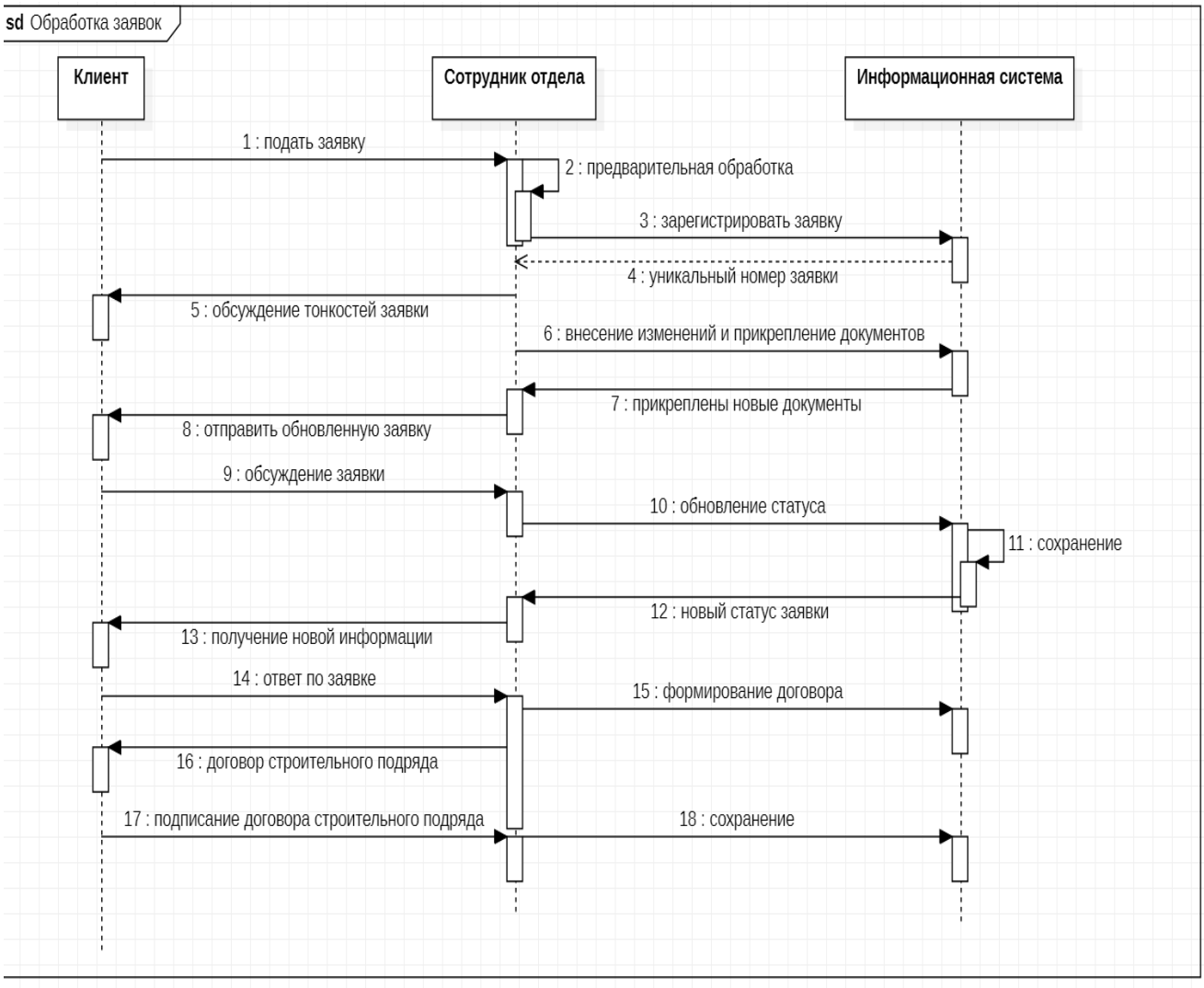

Рисунок 8 – Диаграмма последовательности для процесса «Обработка заявки»

При разработке информационной системы будет учитываться логика системы и функции которые она будет исполнять на логическом уровне.

2.2 Проектирование базы данных системы обработки заявок пользователей

Существует несколько технологий проектирования базы данных:

– ER-моделирование;

– реляционное моделирование;

– нормализация данных.

В работе применяется логическое моделирование на основе технологий реляционного моделирования, нормализации данных и ER-моделирования.

При разработке используются требования к данным и результаты анализа для формирования логической модели данных. Логическую модель приводят к третьей нормальной форме, и проверяет ее на соответствие модели процессов [21].

Для создания концептуальной и логической модели необходимо определить сущности, которые будут хранить данные информационной системы.

На рисунке 9 показана логическая модель данных.

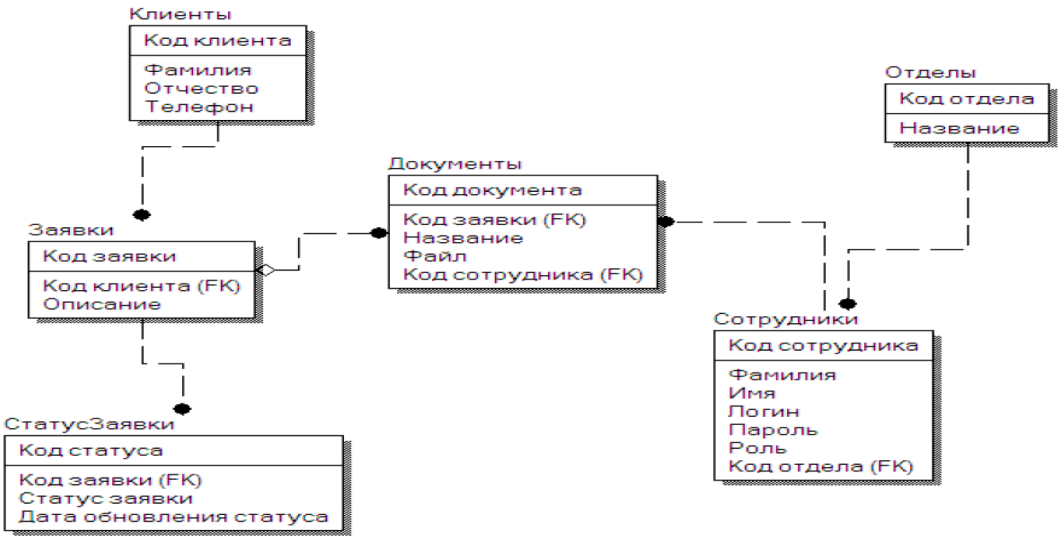

Рисунок 9 — Логическая модель данных

Информационная система обработки заявок пользователей должна содержать базу данных для хранения всей информации, которая поступает и обрабатывается в отделе стратегического развития и информационных технологий [2].

Выделим основные сущности для проектирования базы данных:

– клиенты;

– заявки;

– документы;

– статус документа;

– сотрудники;

– отделы.

В компании в одном отделе могут работать несколько сотрудников.

Клиент может с компанией заключить несколько договоров, куда могут входить несколько заявок.

Клиент может оформлять несколько заявок в одной компании.

Одна заявка может иметь нескольких статусов с периодичностью во времени.

Одну заявку могут обрабатывать несколько сотрудников, поочередно под своим логином и паролем.

В таблице 1 отражены реквизиты сущностей и их ограничения.

Таблица 1 — Реквизиты сущностей и их ограничения

Имя сущности	Описание	Тип
Отдел	Код отдела	Числовой
	Название	25 символов
Статус заявки	Код статуса	Числовой
	Статус заявки	Текст
	Дата обновления статуса	Краткий формат даты
	Заявка	Числовой
Сотрудник	Код сотрудника	Числовой
	Фамилия	25 символов
	Имя	25 символов
	Отдел	Числовой
	Логин	25 символов
	Пароль	25 символов
	Роль	25 символов

Продолжение таблицы 1

Заявка	Код заявки	Числовой
	Клиент	Числовой
	Описание	255 символов
	Статус	Числовой
Клиент	Код клиента	Числовой
	Фамилия	25 символов
	Имя	25 символов
	Отчество	25 символов
	Телефон	11 символов
Документы	Код документа	Числовой
	Название	25 символов
	Файл	
	Заявка	Числовой

В таблице 1 описаны все сущности, которые будут содержаться в базе данных, также видно, что все таблицы будут связаны между собой ключами, которые будут способствовать передачи данных.

Рассмотрим правильность связей между таблицами.

Сущность Клиент и Заявки имеют связь один-ко-многим. Связь является обязательной, то есть, заявка сама по себе не может быть создана, должна обязательно создаваться клиентом или на основе беседы с ним [3]. Один клиент может создавать несколько заявок и отправлять или передавать в компанию для обработки.

Сущность Заявки и Документы имеют связь один-ко-многим. Связь является не обязательной, так как не обязательно к заявке должны прикрепляться дополнительные документы. Но если все-таки документы прикрепляются, то их может быть несколько к одной заявке.

Сущность Заявки и Статус заявки имеют связь один-ко-многим. Связь является обязательной, то есть, заявка обязательно после того, как принята должна быть зарегистрирована в системе и соответственно иметь статус. Одна заявка может иметь несколько статусов на этапе обработки ее сотрудниками.

Сущность Сотрудники и Документы имеют связь один-ко-многим. Связь является обязательной, так как документ не может быть прикреплен без участия сотрудника. Один сотрудник может прикрепить к заявке несколько документов.

Сущность Сотрудники и Отделы имеют связь многие-к-одному. Связь является обязательной, так как отдел обязательно должен иметь сотрудников. Один отдел может содержать несколько сотрудников, так же, как и несколько сотрудников могут работать в одном отделе.

После создания моделей для базы данных можно приступать к физическому проектированию базы данных и реализации информационной системы.

2.3 Выбор архитектуры информационной системы

Все данных и документы, поступающие в отдел стратегического планировании компании, должны храниться в одном хранилище данных для общего доступа сотрудников. Объем хранилища должен позволять хранение информации при накоплении ее в случае расширения потребностей отдела или компании в целом [4].

Данные получаемые пользователем системы будут выдаваться по средствам SQL запросов.

Операционная система должна поддерживать Windows Server.

Для написания приложения может быть выбран любой язык программирования высокого уровня [6].

Архитектура приложения должна быть гибкой и доступной для вносимых изменений при расширении системы [5].

Рассмотрим несколько распространенных архитектур приложения, чтобы определить какая из них подойдет для приложения, которое будет разрабатываться для компании.

Монолитная архитектура характеризуется единым монолитным кодом, который развертывается единым приложением [8]. Архитектура имеет достоинства такие как простота разработки и тестирование.

Микросервисная архитектура подразумевает что приложение состоит из нескольких автономных сервисов, каждый из которых отвечает за определенную бизнес-функциональность [9].

Преимущества данной архитектуры заключаются в высокой гибкости и масштабируемости. Недостатки тоже есть, возможные проблемы согласованности данных между сервисами, а также дополнительные затраты на управление инфраструктурой.

Клиент-серверная архитектура имеет клиента и сервер.

Функциональность клиент-серверной архитектуры позволяет обрабатывать запросы от клиентов, управлять ресурсами и выполнять операции, необходимые для клиентов [10].

На рисунке 10 показана клиент-серверная архитектура.

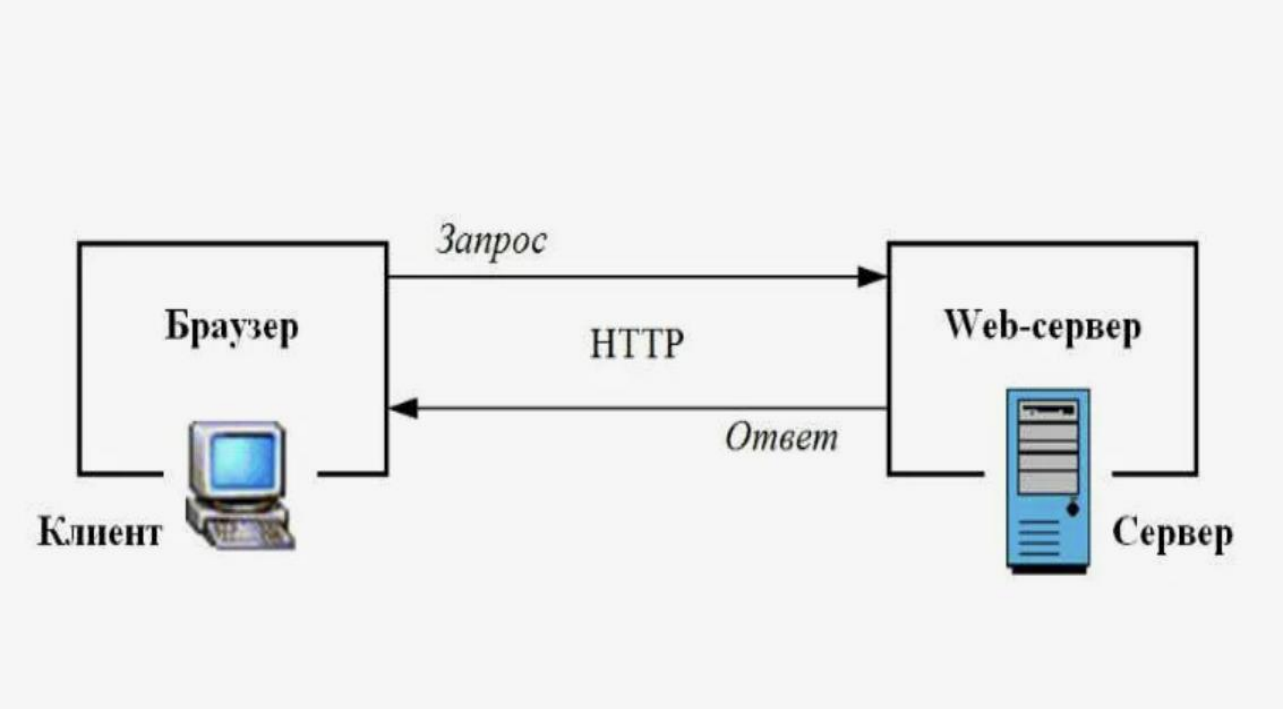

Рисунок 10 — Клиент-серверная архитектура

Клиент-серверная архитектура самая подходящая для реализации системы обработки заявок сотрудников компании ООО «Гарант».

Выводы по второй главе.

Вторая глава работы была ориентирована на логическое проектирование информационной системы обработки заявок пользователей компании ООО «Гарант».

На основе выявленных слабых мест в процессе обработки заявок пользователей было предложено автоматизировать данный процесс.

Для создания информационной системы были рассмотрены различные архитектуры и определена клиент-серверная архитектура, которая позволяет обрабатывать запросы клиентов компании ООО «Гарант».

Была спроектирована диаграмма прецедентов, показывающая основные функции разрабатываемой системы.

Более подробно процесс обработки заявок был представлен на диаграмме последовательности.

Итогом работы во второй главе было выделение основных сущностей для создания базы данных и разработка логической модели данных.

3 Физическое проектирование автоматизированной информационной системы обработки заявок пользователей

3.1 Выбор инструментов разработки

Рассмотрим C++, C# и Java, наиболее распространенные.

C++ язык, содержащий средства создания эффективных программ [13]. Однако недостатки не подходят нам для разработки, такие как синтаксис, провоцирующий ошибки, препроцессор, унаследованный от C, очень примитивен, плохая поддержка модульности.

C# много перенял у Java и C++ [12]. C# изначально предназначался для веб-разработки, и примерно 75% его синтаксических возможностей такие же, как у Java [14].

Так как в приложении будет создаваться графический интерфейс и база данных, то этот язык подходит со своей функциональностью и гибкостью.

Язык Java зародился как часть проекта создания передового программного обеспечения для различных бытовых приборов.

В Java используются практически идентичные соглашения для объявления переменных, передачи параметров, операторов и для управления потоком выполнением кода [16]. В Java добавлены все хорошие черты C++.

В таблице 2 отражены результаты сравнения языков программирования.

Таблица 2 – Результаты сравнения языков программирования

Основные отличия	C++	C#	Java
1	2	3	4
Понятный синтаксис	-	+	+
Интеграция с базой данных	+	+	+
Скорость обработки данных	+	+	-
Функциональность и гибкость языка	-	+	-
Интеграция с фрэймворками	-	+	+

Для создания системы выберем инструмент разработки, система будет небольшой и создаваться в учебных целях, поэтому создадим пользовательский интерфейс и выберем СУБД, где будет храниться информация по заявкам и клиентам.

Для сравнения возьмем наиболее распространенные СУБД такие как: SQL Server, Access и PostgreSQL [17].

Access можно быстро изучить, любой пользователь найдет много информации как в ней работать, поэтому она доступна для пользователей с низкой квалификацией [11].

Достоинства:

– понятный интерфейс;

– наличие русской версии;

– возможность работать в конструкторе.

Недостатки СУБД:

– отсутствие серьезной защиты информации;

– ограниченный объем информации;

– отсутствие собственного языка программирования;

– отсутствие скорости при обработке больших данных.

PostgreSQL имеет большой функционал и имеет большую надежность хранения данных от несанкционированного доступа [15]. Огромным плюсом является то, что СУБД имеется в открытом доступе и имеет открытый код.

Ниже представлены несколько функциональных возможностей СУБД:

– вложенные запросы;

– представления;

– ссылочная целостность – внешние ключи.

Однако есть и минусы, как и любой другой системы управления данными. Главным минусом является плохая генерация с языками программирования [18].

СУБД SQL Server разработана Microsoft и расширяет свои возможности по сей день. Объем хранения данных, может расширяться по мере наполнения информацией, без заметного уменьшения быстродействия операций с записями в многопользовательском режиме [23].

Пользователь получает ответы от SQL Server путем отправки запроса на сервер и получения данных в ответ. Как правило такие СУБД выбирают для обработки данных в сети интернет [19].

На основе рассмотренных выше систем управления данными можно сделать вывод и выбрать оптимальный вариант для приложения. СУБД Microsoft SQL Server подойдет больше всех, она имеет все доступные средства для разработки приложения [24]. В таблице 3 отражены результаты сравнения баз данных.

Таблица 3 – Результаты сравнения баз данных

Основные отличия	Access	MS SQL	PostgreSQL
1	2	3	4
Доступность	+	+	+
Хранение большого объема данных	-	+	+
Скорость обработки операций с большими данными	-	+	-
Защита от несанкционированного доступа	-	+	+
Простота использования	+	-	-

Сравнительный анализ СУБД показал, что с применением языка программирования C# подходящей будет СУБД Microsoft SQL Server.

3.2 Разработка системы

Ранее была выбрана СУБД Microsoft SQL Server, создадим в ней таблицы, спроектированные выше для информационной системы обработки заявок от клиентов компании. На рисунке 11 показана модель данных, созданная в СУБД.

Созданы будут таблицы под названием:

– клиенты;

– сотрудники:

– заявки;

– статус заявки;

– документы;

– отделы.

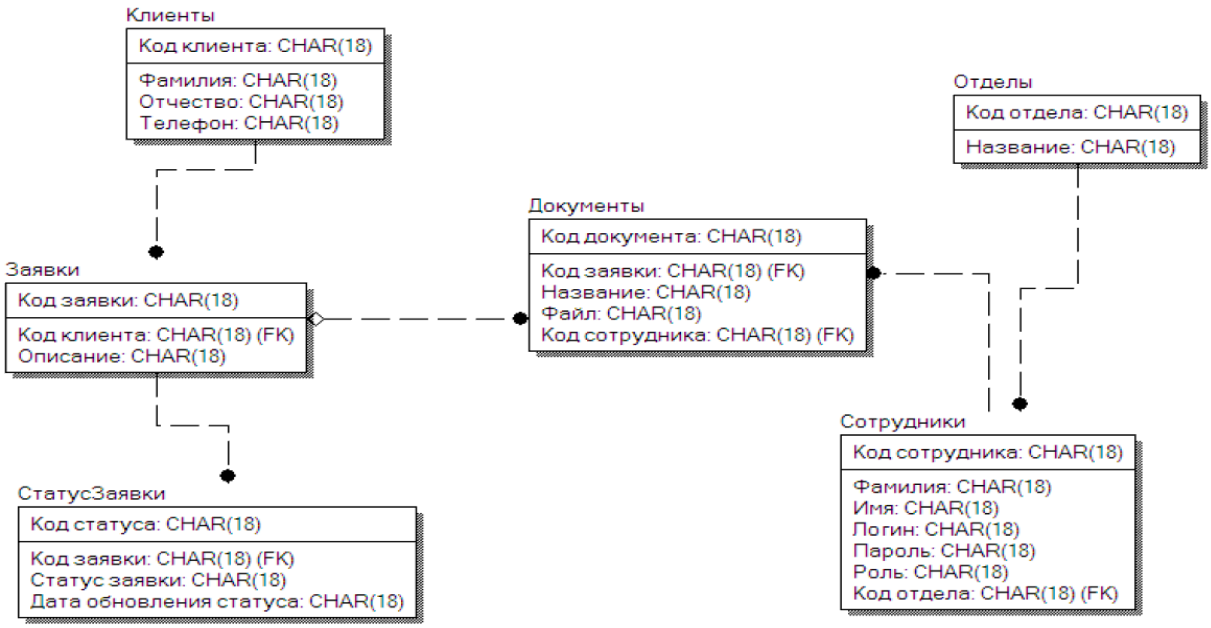

Рисунок 11 – Физическая модель данных

В модели представлены таблицы на латинском языке, так же, как и поля. Из рисунка 11 видно, что заявки будут иметь статус и историю. Сотрудник,

который будет открывать заявку сможет посмотреть историю ее поступлению начиная от даты поступления до последнего обновления статуса заявки [20].

3.3 Разработка информационной системы обработки заявок компании ООО «Гарант»

Приложение разработано в среде Visual Studio 2022. Среда позволяет создавать код на выбранном языке программирования C#. Программный продукт во многом помогает разработчику создавать правильный код без ошибок, подсвечивая нужную информацию как подсказку для пользователя [21].

При работе с информационной системой она будет запрашивать у пользователя при входе ввести логин и пароль, который будет присваивать администратор системы, в соответствии с рисунком 12.

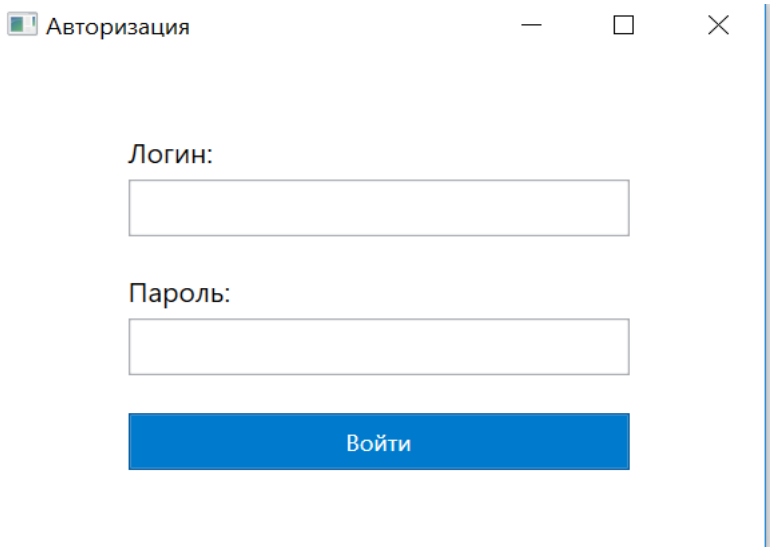

Рисунок 12 – Авторизация в системе

После внесения логина и пароль сотрудник компании (менеджер, инженер) входит в систему согласно своей роли, в соответствии с рисунком 13.

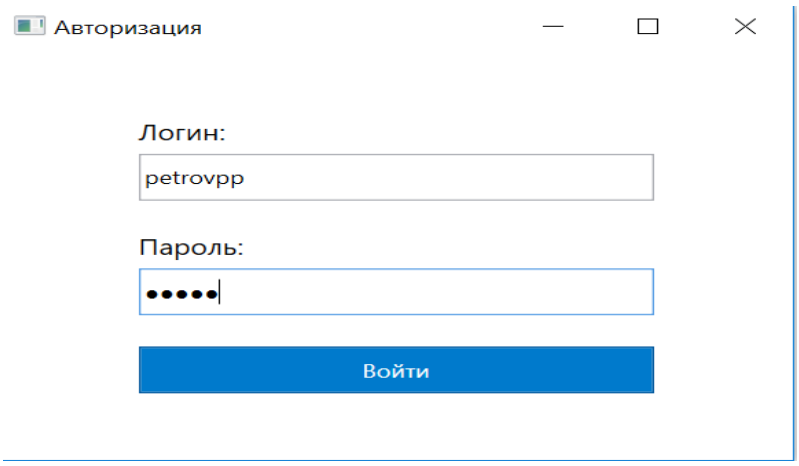

Рисунок 13 – Вход в систему

После входа в систему сотрудник может зарегистрировать заявку, для этого в системе предусмотрена готовая форма, в соответствии с рисунком 14.

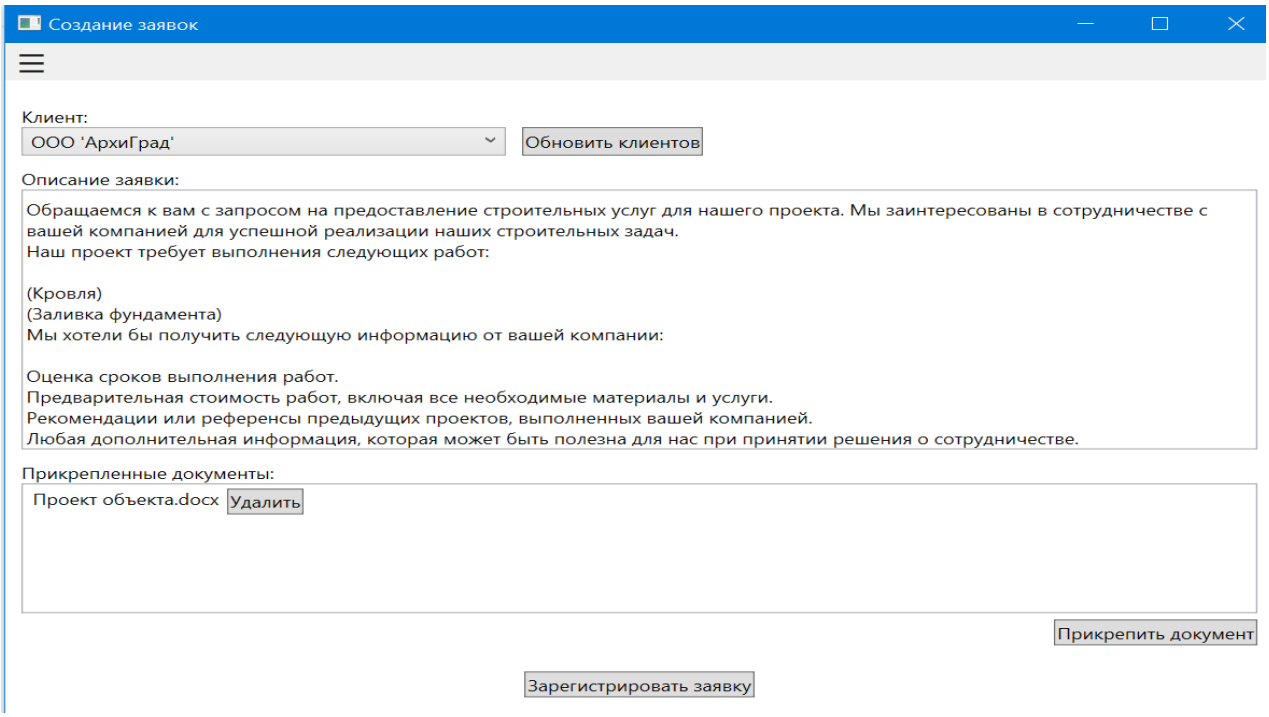

Рисунок 14 – Форма создания заявок

Форма позволяет выбирать из списка уже зарегистрированных ранее клиентов либо зарегистрировать нового клиента, если он обращается в компанию впервые, в соответствии с рисунком 15.

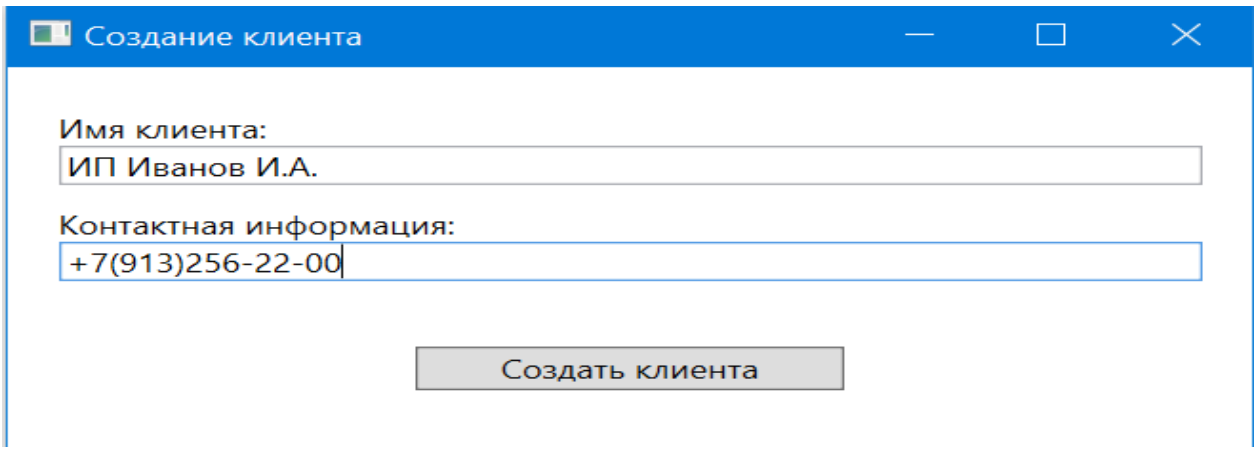

Рисунок 15 – Форма создания клиента

Когда клиент добавлен или уже есть в системе, можно приступать к оформлению заявки согласно требованиям клиента. Форма позволяет выбрать клиента из базы данных, внести описание заявки, прикрепить дополнительные документы, к примеру план дома, после чего заявка регистрируется, в соответствии с рисунком 16.

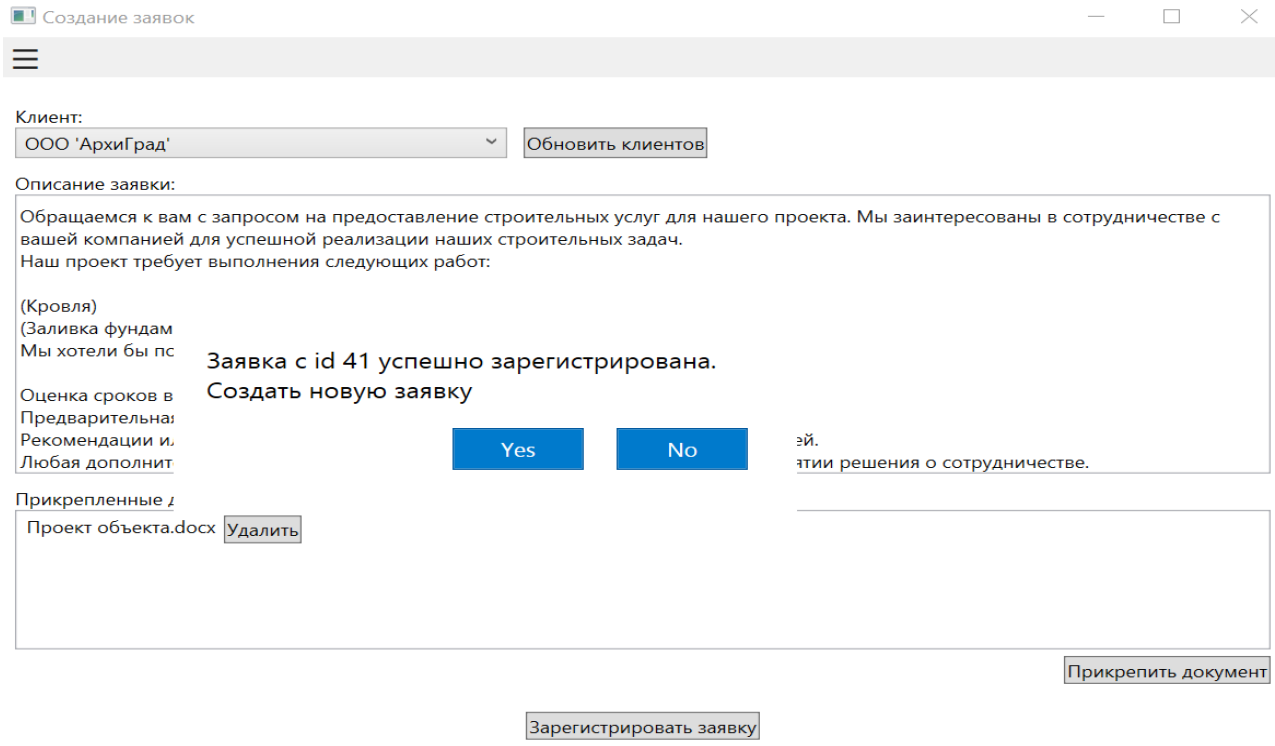

Рисунок 16 – Создание и регистрация заявки

После чего заявка появляется в списке заявок в системе в левой стороне окна пользователя, где он может выбрать любую из списка и просмотреть информацию. Посмотрим заявку, которую создали ранее, в соответствии с рисунком 17.

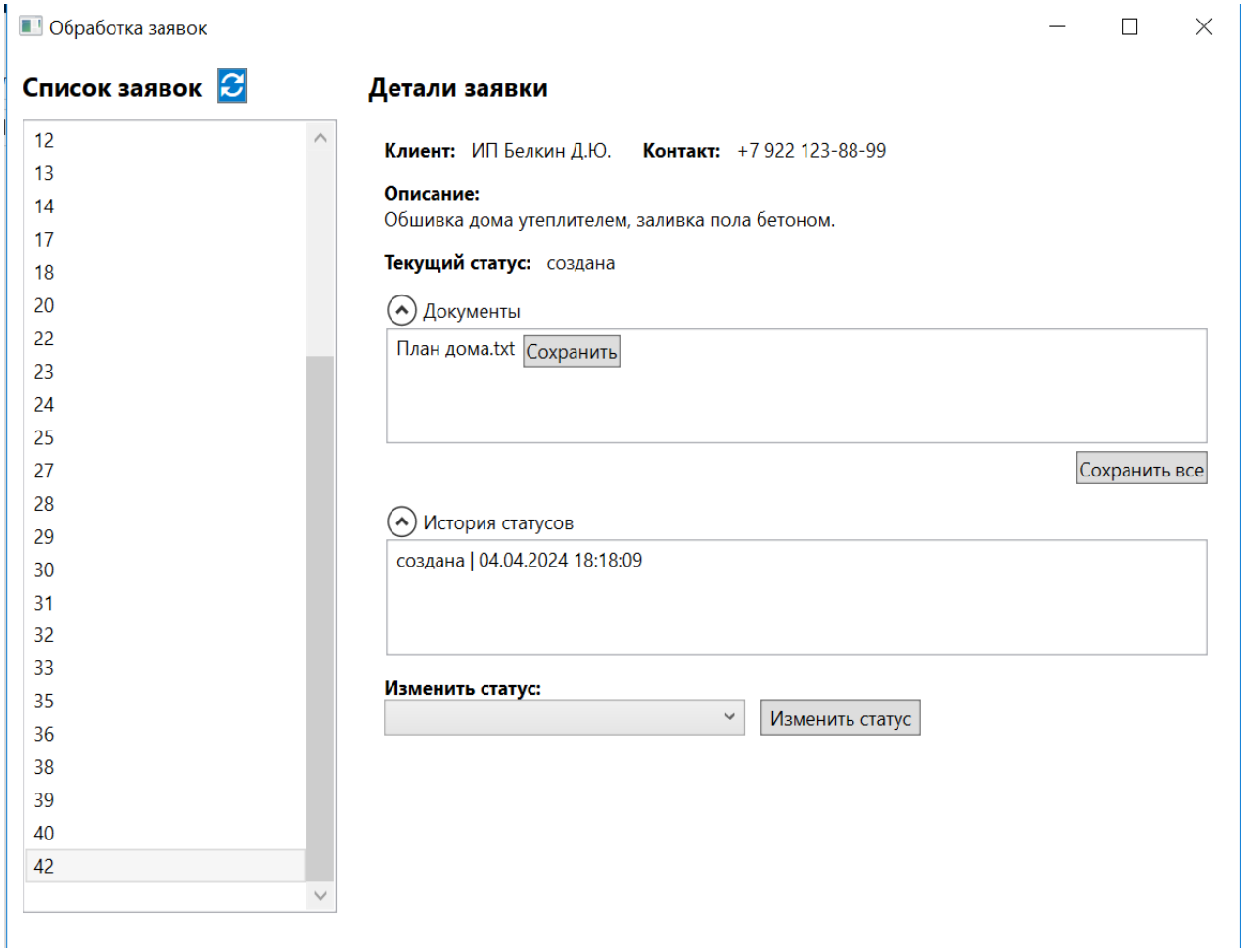

Рисунок 17 – Обработка заявок

На форме показана вся информация о заявке, которая поступила в систему для обработки. Как видно в списке заявок у каждой заявки присвоен номер, который не повторяется. Также видно кто является клиентом заявки, его контакты и описание заявки. Текущий статус заявки стоит создана, этот статус проставляется автоматически при регистрации заявки и ставится текущая дата регистрации [22]. В окне Документы показаны все документы, которые были

прикреплены к заявке, их может быть как несколько, так и один, их можно сохранить на рабочий стол, к примеру. История статусов показывает какие статусы были у заявки и в каком статусе она находится на данным момент [27]. После обработки сотрудником заявки он меняет статус заявки, в соответствии с рисунком 18.

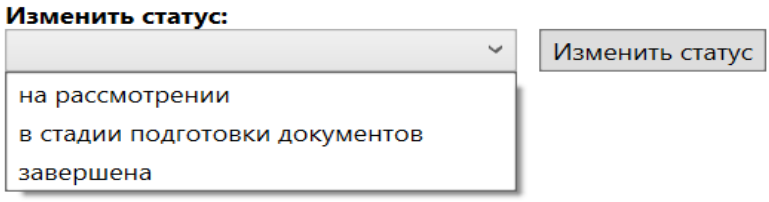

Рисунок 18 – Статусы заявок

После того как статус меняется система выдает сообщение о том, что статус заявки успешно изменен, в соответствии с рисунком 19.

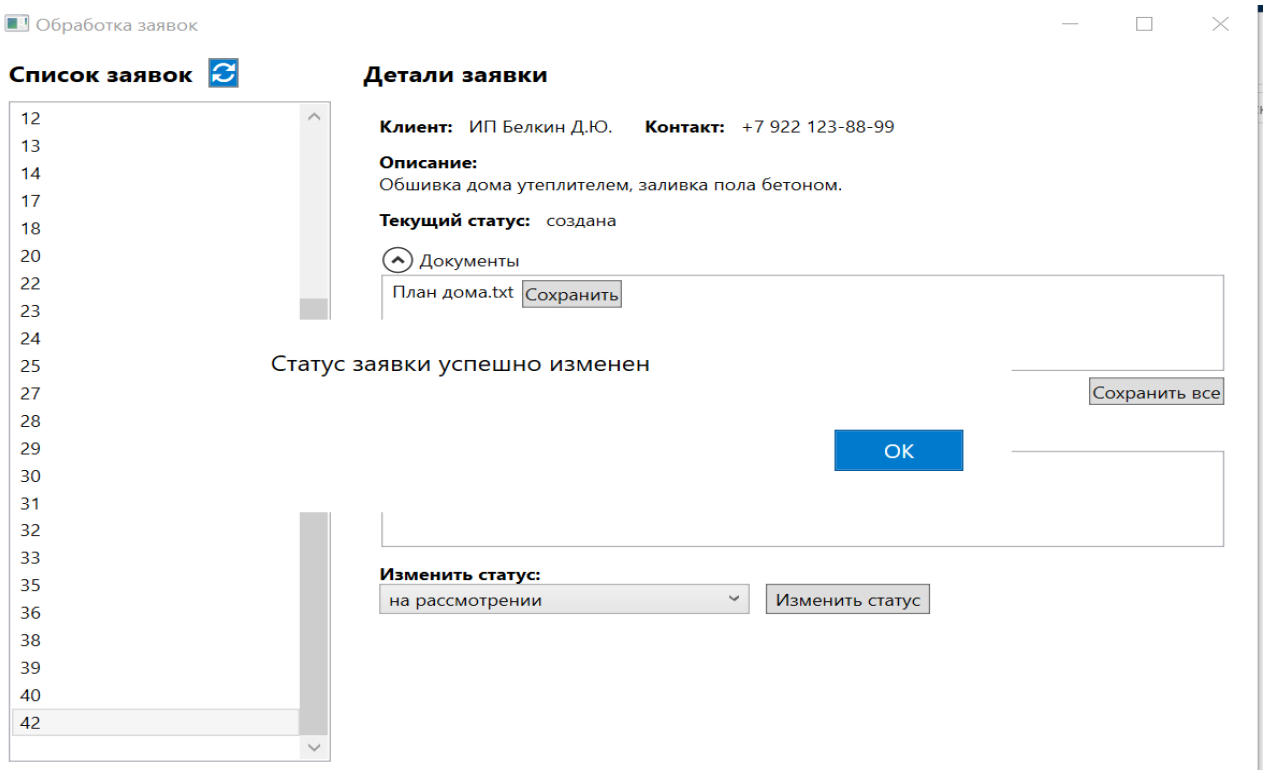

Рисунок 19 – Изменение статуса

После того как заявка будет обработана всеми участниками процесса, заявка поменяет статус на «завершена», после чего заявка исчезнет из списка заявок, в соответствии с рисунком 20.

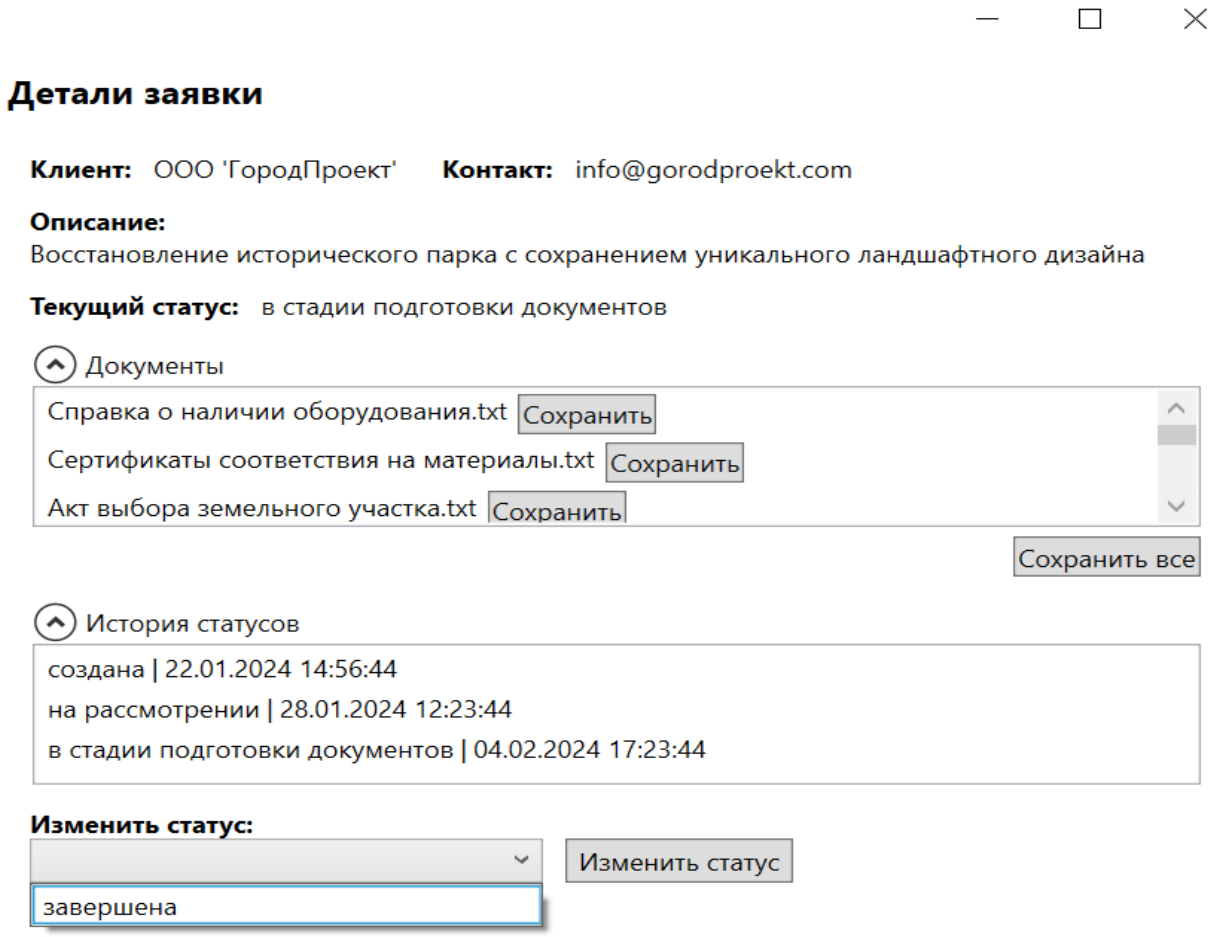

Рисунок 20 – Завершение заявки

Таким образом будет проходить автоматизация обработки заявок пользователей. Одну заявку сможет видеть не один сотрудник, а несколько одновременно, однако менять статус сможет только один и только в правильном хронологическом порядке [25]. Если второй сотрудник одновременно захочет поменять статус заявки, система выдаст ошибку и сообщение о том, что заявка уже обрабатывается.

3.4 Тестирование информационной системы

Проведем тестирование системы на проверку корректности работы и вывода информации об ошибках пользователю.

Менеджер, работающий в системе, который работает непосредственно клиентами компании, вводит новые данные в форму, которая должна сохранять данные в базу данных. Ожидания от системы, что при вводе всех данных о клиенте которые необходимо предоставить в готовой форме, после создания нового клиента, система должна показать оповещение о том, что клиент создан в соответствии с рисунком 21.

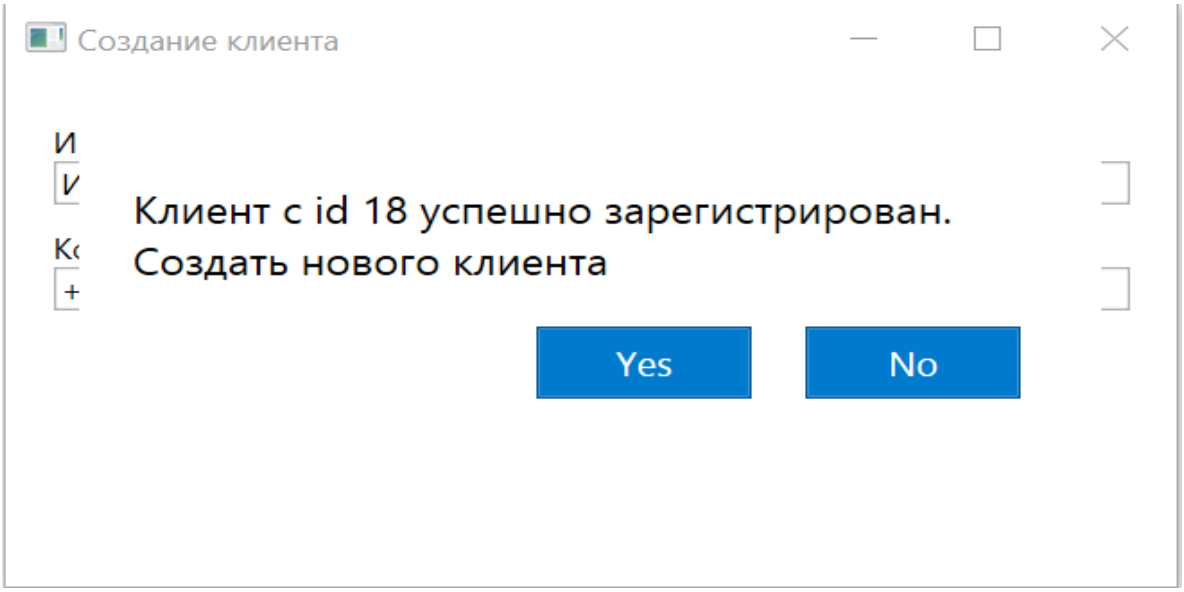

Рисунок 21 – Оповещение о создании нового клиента

Если пользователь не вводит какие-то данные о клиенте, то система не дает ему сохранять данные и выводит сообщение о том, были введены не все данные о клиенте, в соответствии с рисунком 22.

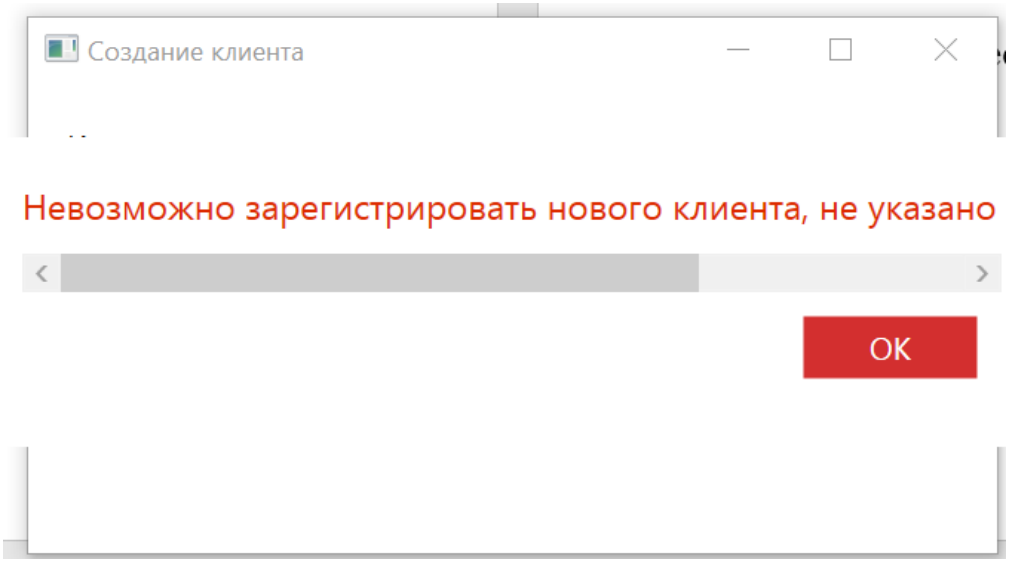

Рисунок 22 – Оповещение об ошибке

Другой пользователь, который непосредственно обрабатывает поступившие заявки, может увидеть в деталях заявки прикрепленные дополнительные документы [26] как показано на рисунке 20 и может их скачать, однако если допущена ошибка и не указан путь сохранения, система выдаст ошибку и оповещение пользователю, в соответствии с рисунком 23.

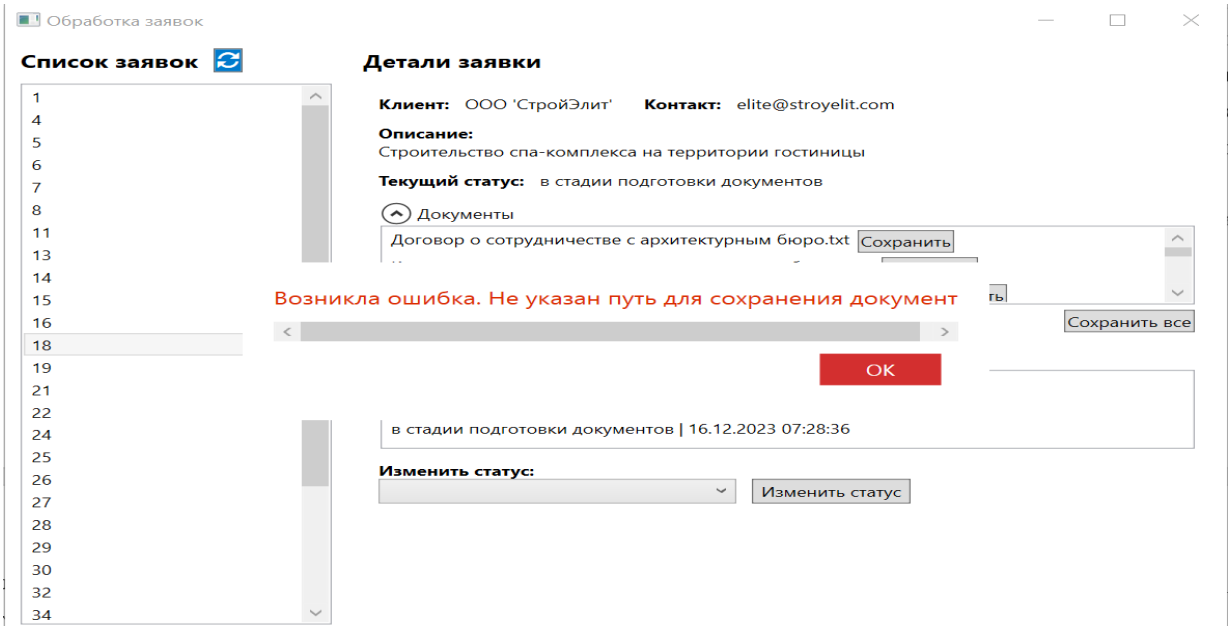

Рисунок 23 – Оповещение об отсутствии пути для сохранения документа

При создании заявки пользователем система также предоставляет готовую форму для заполнения, где должны быть заполнены все поля, если по какой-то причине пользователь не вводит данные в какое-то поле заявки [26], то система не даст сохранить эту заявку и выдаст сообщение об ошибке, в соответствии с рисунком 24.

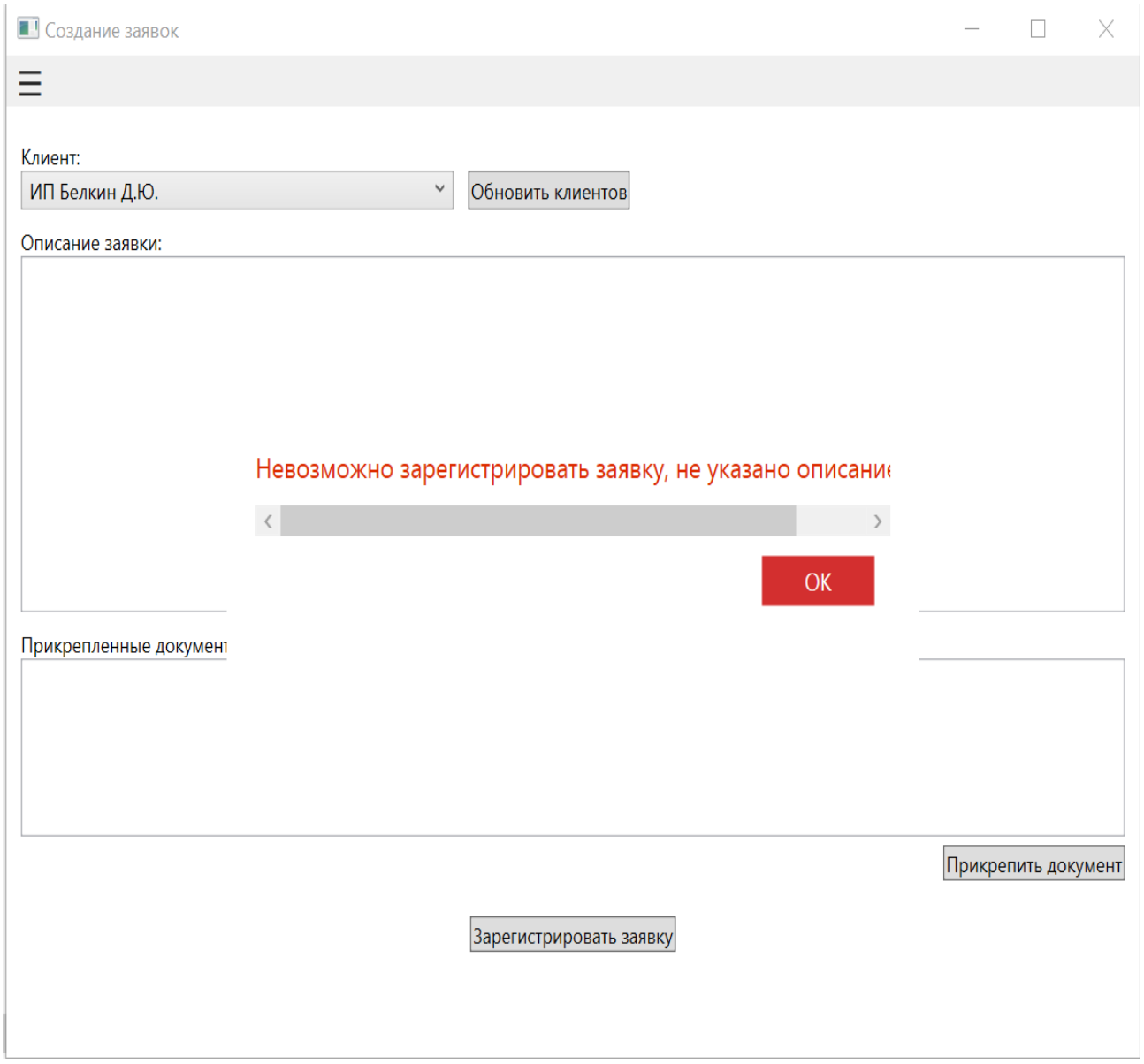

Рисунок 24 – Оповещение о пустых полях

Таким образом было проведено тестирование основных функций системы обработки заявок пользователей.

Выводы по третьей главе.

В заключительной главе работы была проведена физическая реализация информационной системы обработки заявок пользователей компании ООО «Гарант».

В работе представлены основные формы, с которыми будут работать пользователи согласно своей роли, также представлен интерфейс системы и проведено тестирование функциональности системы.

На основе проведенного тестирования можно сделать вывод, что информационная система отвечает всем поставленным требованиям заказчика и корректно выполняет функции обработки заявок.

Заключение

В ходе выполнения выпускной квалификационной работы описана разработка информационной системы для автоматизации обработки заявок пользователей в строительной компании ООО «Гарант».

На первом этапе работы была рассмотрена структура и проведено изучение деятельности компании, выявлении бизнес-процессов, которые не автоматизированы и нуждаются в автоматизации с помощью внедрения информационных технологий.

По результатам анализа было выяснено, что в автоматизации нуждается отдел стратегического развития и информационных технологий, который занимается приемкой и обработкой заявок от пользователей.

Проведен сравнительный анализ существующих разработок сторонних компаний, что показало все недостатки этих программных продуктов.

Для реализации приложения были выбраны инструменты разработки C# и СУБД Microsoft SQL, как наиболее подходящие и эффективные для реализации системы.

Было проведено тестирование информационной системы. По результатам тестирования было установлено, что система выполняет все заложенные в нее функции и при наличии ошибок оповещает об этом пользователя.

Разработанная информационная система увеличит эффективность работы сотрудников, а также позволит сотрудникам отдела стратегического развития и информационных технологий принимать и обрабатывать заявки пользователей за более короткий срок с наименьшими ошибками, что и является целью выпускной квалификационной работы.

По результатам выполненной работы можно считать, что задачи ВКР выполнены в полном объеме, цель достигнута.

Список используемой литературы и используемых источников

1. Волк В. К. Базы данных. Проектирование, программирование, управление и администрирование: учебник /В. К. Волк. — Санкт-Петербург: Лань, 2020. — 244 с.: ил.— (Учебники для вузов. Специальная литература). — Текст: непосредственный.

2. Как использовать Entity Framework. [Электронный ресурс]. – URL: https://skillbox.ru/media/code/entity_framework/ (дата обращения: 01.04.2024)

3. Концептуальное логическое и физическое моделирование данных. [Электронный ресурс]. – URL: https://frameworx.ru/SID/datamodelling.html (дата обращения: 01.04.2024).

4. Моделирование систем с использованием информационных технологий: учебн. пособие / В. Г. Лисиенко, Н. Г. Дружинина, О. Г. Трофимова, С. П. Трофимов. – Екатеринбург: УГТУ-УПИ, 2009. – 440 с.

5. Онлайн курс обучения программированию: методологии разработки. [Электронный ресурс]. – URL: https://javarush.ru/groups/posts/647-metodologii-razrabotki-po (дата обращения: 01.04.2024)

6. Отличия, достоинства и недостатки базы данных PostgreSQL: что такое PostgreSQL. [Электронный ресурс]. – URL: https://oracle-patches.com/common/3214-что-такое-postgresql (дата обращения: 01.04.2024)

7. Попова-Коварцева, Д. А. Основы проектирования баз данных: учеб. пособие / Д.А. Попова-Коварцева, Е.В. Сопченко. – Самара: Изд-во Самарского университета, 2019. – 112 с.

8. C GUI на C#: Кроссплатформенное приложение [Электронный ресурс]. - URL: https://skillbox.ru/media/code/ne_windows_(дата обращения: 01.04.2024)

9. Тамре Л. Введение в тестирование программного обеспечения /пер. с англ. М.: Вильямс, 2018. 368 с.

10. Этапы и методологии проектирования баз данных. [Электронный ресурс]. – URL: https://studfile.net/preview/2674691/page:4/ (дата обращения: 09.10.2023).

11. Access: База данных. [Электронный ресурс]. – URL: https://www.microsoft.com/ru-ru/microsoft-365/access(дата обращения: 01.04.2024)

12. GeekBrains – Язык программирования C#. [Электронный ресурс]. - URL: https://geekbrains.ru/posts/yazyk-programmirovaniya-c-sharp-istoriya-specifika-mesto-na-rynke (дата обращения: 01.04.2024)

13. Helpiks.org: Достоинства и недостатки языка [Электронный ресурс]. - URL: https://helpiks.org/6-21879.html (дата обращения: 01.04.2024)

14. Java-энциклопедия языков программирования: Java. [Электронный ресурс]. – URL: http://progopedia.ru/language/java/ (дата обращения: 01.04.2024)

15. SoftClipper: что такое SQL Server. [Электронный ресурс]. – URL: https://softclipper.net/foxpro-i-sql/sravnenie-baz-dannykh-microsoft-sql-server-i-microsoft-visual-foxpro.html (дата обращения: 01.04.2024)

16. SQL Server: руководство по SQL Server. [Электронный ресурс]. – URL: https://docs.microsoft.com/ru-ru/sql/sql-server/tutorials-for-sql-server-2016?view=sql-server-ver15 (дата обращения: 01.04.2024)

17. SQL Server: программное обеспечение. [Электронный ресурс]. – URL: https://www.microsoft.com/ru-ru/sql-server/sql-server-downloads (дата обращения: 01.04.2024)

18. C#. [Электронный ресурс]. – URL: https://dotnet.microsoft.com/en-us/languages/csharp (дата обращения: 01.04.2024)

19. The C Sharp (C#) Beginner’s Guide [Электронный ресурс]. – URL: https://medium.com/c-sharp-language/the-c-beginners-guide-6a14af03ed85 (дата обращения: 01.04.2024)

20. Visual Studio 2022: программное обеспечение. [Электронный ресурс]. – URL: https://visualstudio.microsoft.com/ru/vs/ (дата обращения: 01.04.2024)

21. Visual Studio 2022. [Электронный ресурс]. – URL: https://www.techspot.com/downloads/7493-visual-studio-2022.html (дата обращения: 01.04.2024)

22. Visual Studio tutorials | C#. [Электронный ресурс]. – URL: https://learn.microsoft.com/en-us/visualstudio/get-started/csharp/?view=vs-2022l (дата обращения: 01.04.2024)

23. Choy, K. L. Development of an intelligent customer-supplier relationship management system: the application of case-based reasoning / K. L. Choy, Kenny K H Fan, Victor Lo // Industrial Management & Data Systems. – 2003. – Vol. 103, No. 4. – P. 263-274.

24. Law, Y. F. D. An Integrated Case-Based Reasoning Approach for Intelligent Help Desk Fault Management / Y. F. D. Law, B. F. Sew, S. E. J. Kwan // Expert Systems with Applications. – 1997. – Vol. 13, No. 4. – P. 265-274. – EDN AJGJSH.

25. Twidale, M. B. Browsing is a collaborative process / M. B. Twidale, D. M. Nichols, C. D. Paice // Information Processing & Management. – 1997. – Vol. 33, No. 6. – P. 761-783. – EDN AITGCJ.

26. Mark, G. Conventions and Commitments in Distributed CSCW Groups / G. Mark // Computer Supported Cooperative Work. – 2002. – Vol. 11, No. 3-4. – P. 349-387. – EDN BBXQUN.

27. Simazu, H. ExpertGuide: A Conversational Case-Based Reasoning Tool for Developing Mentors in Knowledge Spaces / H. Simazu, A. Shibata, K. Nihei // Applied Intelligence. – 2001. – Vol. 14, No. 1. – P. 33-48. – EDN AMODOB.

Приложение А

Листинг программы

```
namespace BuildCompanyRequestManager.WpfApplication.Services
{
   public class FilePathRequestor : IFilePathRequestor
   {
      private readonly INavigationService _navigationService;

      public FilePathRequestor(INavigationService navigationService)
      {
         _navigationService = navigationService ?? throw new ArgumentNullException(nameof(navigationService));
      }

      public string? Request(string fileName)
      {
         try
         {
            SaveFileDialog saveFileDialog = new SaveFileDialog();
            saveFileDialog.Filter = "All files (*.*)|*.*";
            saveFileDialog.FileName = fileName;
            if (saveFileDialog.ShowDialog() == true)
            {
               return saveFileDialog.FileName;
            }
            return null;
         }
         catch (Exception e)
```

```
{
            _navigationService.ShowErrorView($"Возникла ошибка {e.Message}");
            return null;
        }
    }
  }
}
namespace BuildCompanyRequestManager.WpfApplication.Views
{
  /// <summary>
  /// Логика взаимодействия для ConfirmWindow.xaml
  /// </summary>
  public partial class ConfirmWindow : Window
  {
    public ConfirmWindow()
    {
      InitializeComponent();
    }

    public string Question
    {
      set { QuestionTextBlock.Text = value; }
    }

    private void Yes_Button_Click(object sender, RoutedEventArgs e)
    {
```

```
this.DialogResult = true;
this.Close();
        }

        private void No_Button_Click(object sender, RoutedEventArgs e)
        {
            this.DialogResult = false;
            this.Close();
        }
    }
}
namespace BuildCompanyRequestManager.Application
{
    public class RegisterNewClientUseCase : IRegisterNewClientUseCase
    {
        private readonly IClientRepository _clientRepository;
        private readonly INavigationService _navigationService;

        public RegisterNewClientUseCase(
            IClientRepository clientRepository,
            INavigationService navigationService)
        {
            _clientRepository = clientRepository ?? throw new
ArgumentNullException(nameof(clientRepository));
            _navigationService = navigationService ?? throw new
ArgumentNullException(nameof(navigationService));
        }
```

```
public async Task ExecuteAsync(string? name, string? contactInfo)
{
    try
    {
        if (string.IsNullOrWhiteSpace(name))
        {
            _navigationService.ShowErrorView($"Невозможно зарегистрировать нового клиента, " +
            $"не указано имя клиента");
            return;
        }

        if (string.IsNullOrWhiteSpace(contactInfo))
        {
            _navigationService.ShowErrorView($"Невозможно зарегистрировать нового клиента, " +
            $"не указано контактная информация");
            return;
        }

        var client = new Client(0, name, contactInfo);
        var registeredClient = await _clientRepository.AddAsync(client);

        if (registeredClient is null)
        {
            _navigationService.ShowErrorView($"При регистрации клиента " +
            $"возникла ошибка");
```

```
return;
}
            var again = _navigationService.ConfirmView($"Клиент с id " +
                $"{registeredClient.Id} успешно зарегистрирован. " +
                $"Создать нового клиента");
            if (!again)
            {
                _navigationService.CloseView<IRegistratorClientView>();
                _navigationService.ShowView<IOperatorView>();
                return;
            }
            _navigationService.ReOpenView<IRegistratorClientView>();
        }
        catch (Exception e)
        {
            _navigationService.ShowErrorView($"При регистрации клиента " +
                $"возникла ошибка {e.Message}");
        }
    }
  }
}
namespace BuildCompanyRequestManager.WpfApplication.ViewModels
{
  internal class OperatorWindowViewModel : ViewModel
  {
```

```
private readonly IGetAllClientsUseCase _getAllClientsUseCase;
    private readonly IAddDocumentsToListDocumentsUseCase
_addDocumentsToListDocumentsUseCase;
    private readonly IRegisterApplicationUseCase _saveApplicationUseCase;

private readonly INavigateToRegisterNewClientUseCase
_navigateToRegisterNewClientUseCase;

    #region Заголовок окна
    private string? _title = "Создание заявок";

    /// <summary>
    /// Заголовок окна
    /// </summary>
    public string? Title
    {
       get => _title;
       set => Set(ref _title, value);
    }
    #endregion

    #region Данные загружены?
    private bool _isDataLoaded;

    /// <summary>
    /// Данные загружены?
```

```
/// </summary>
        public bool IsDataLoaded
        {
            get => _isDataLoaded;
            set
            {
                Set(ref _isDataLoaded, value);
                CallCanExecuteForAllCommand();
            }
        }
        #endregion

        #region Клиенты
        private IEnumerable<Client>? _clients;

        /// <summary>
        /// Клиенты
        /// </summary>
        public IEnumerable<Client>? Clients
        {
            get => _clients;
            set => Set(ref _clients, value);
        }
        #endregion

        #region Выбранный клиент
```

```
private Client? _client;

        /// <summary>
        /// Выбранный клиент
        /// </summary>
        public Client? Client
        {
            get => _client;
            set => Set(ref _client, value);
        }
        #endregion

        #region Описание заявки
        private string? _description;

        /// <summary>
        /// Описание заявки
        /// </summary>
        public string? Description
        {
            get => _description;
            set => Set(ref _description, value);
        }
        #endregion

        #region Разрешено удалять все документы?
```

```
private bool _isEnabledRemoveAllDocuments;

        /// <summary>
        /// Разрешено удалять все документы?
        /// </summary>
        public bool IsEnabledRemoveAllDocuments
        {
            get => _isEnabledRemoveAllDocuments;
            set => Set(ref _isEnabledRemoveAllDocuments, value);
        }
        #endregion

        #region Список документов
        private IEnumerable<Document>? _documents;

        /// <summary>
        /// Список документов
        /// </summary>
        public IEnumerable<Document>? Documents
        {
            get => _documents;
            set
            {
                Set(ref _documents, value);
                if (value?.Count() > 1)
                    IsEnabledRemoveAllDocuments = true;
```

```
else
                IsEnabledRemoveAllDocuments = false;
            CallCanExecuteForAllCommand();
        }
    }
    #endregion

    #region Выбранный документ
    private Document? _document;

    /// <summary>
    /// Выбранный документ
    /// </summary>
    public Document? Document
    {
        get => _document;
        set => Set(ref _document, value);
    }
    #endregion

    public OperatorWindowViewModel(
        IGetAllClientsUseCase getAllClientsUseCase,
        IAddDocumentsToListDocumentsUseCase
addDocumentsToListDocumentsUseCase,
        IRegisterApplicationUseCase saveApplicationUseCase,
        INavigateToRegisterNewClientUseCase
navigateToRegisterNewClientUseCase)
```

```
{
        _getAllClientsUseCase = getAllClientsUseCase ?? throw new
System.ArgumentNullException(nameof(getAllClientsUseCase));
        _addDocumentsToListDocumentsUseCase =
addDocumentsToListDocumentsUseCase ?? throw new
ArgumentNullException(nameof(addDocumentsToListDocumentsUseCase));
        _saveApplicationUseCase = saveApplicationUseCase ?? throw new
ArgumentNullException(nameof(saveApplicationUseCase));
        _navigateToRegisterNewClientUseCase =
navigateToRegisterNewClientUseCase;
        _ = InitializeWithTimeoutAsync();
    }

    #region AddDocumentCommand
    private LambdaCommand? _addDocumentCommand;

    /// <summary>
    /// Добавить документ
    /// </summary>
    public ICommand AddDocumentCommand
    {
      get
      {
        if (_addDocumentCommand == null)
        {
          _addDocumentCommand = new
LambdaCommand(OnAddDocumentCommandExecuted,
CanAddDocumentCommandExecute);
        }
```

```
return _addDocumentCommand;
        }
    }

    private bool CanAddDocumentCommandExecute(object? p)
    {
        return true;
    }

    private void OnAddDocumentCommandExecuted(object? p)
    {
        Task.Run(() =>
        {
            try
            {
                if (Documents is null)
                    Documents = new List<Document>();
                Documents = _addDocumentsToListDocumentsUseCase.Execute(Documents.ToList());
            }
            catch (Exception e)
            {
                MessageBox.Show(System.Windows.Application.Current.MainWindow, $"Возникла ошибка {e.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);
            }
        });
```

```
}
    #endregion

    #region RemoveDocumentCommand
    private LambdaCommand? _removeDocumentCommand;

    /// <summary>
    /// Удалить документ
    /// </summary>
    public ICommand RemoveDocumentCommand
    {
      get
      {
        if (_removeDocumentCommand == null)
        {
          _removeDocumentCommand = new
LambdaCommand(OnRemoveDocumentCommandExecuted,
CanRemoveDocumentCommandExecute);
        }
        return _removeDocumentCommand;
      }
    }

    private bool CanRemoveDocumentCommandExecute(object? p)
    {
      return true;
    }
```

```
private void OnRemoveDocumentCommandExecuted(object? p)
{
    if (p is null)
        return;

    if (Documents is null)
        return;

    Task.Run(() =>
    {
        try
        {
            if (p is Document document)
            {
                var documents = Documents.ToList();
                documents.Remove(document);
                Documents = documents;
            }
        }
        catch (Exception e)
        {
            System.Windows.Application.Current.Dispatcher.Invoke(() =>
            {
                MessageBox.Show(
                    System.Windows.Application.Current.MainWindow,
                    $"Возникла ошибка при удалении документа {e.Message}",
```

```
"Ошибка",
                    MessageBoxButton.OK,
                    MessageBoxImage.Error);
            });
        }
    });
}
#endregion

#region RemoveAllDocumentCommand
private LambdaCommand? _removeAllDocumentCommand;

/// <summary>
/// Удалить все документы
/// </summary>
public ICommand RemoveAllDocumentCommand
{
    get
    {
        if (_removeAllDocumentCommand == null)
        {
            _removeAllDocumentCommand = new
LambdaCommand(OnRemoveAllDocumentCommandExecuted,
CanRemoveAllDocumentCommandExecute);
        }
        return _removeAllDocumentCommand;
    }
```

```
}

    private bool CanRemoveAllDocumentCommandExecute(object? p)
    {
        return IsEnabledRemoveAllDocuments;
    }
    private void OnRemoveAllDocumentCommandExecuted(object? p)
    {
        if (Documents is null)
            return;

        Task.Run(() =>
        {
            try
            {
                Documents = new List<Document>();
            }
            catch (Exception e)
            {
                System.Windows.Application.Current.Dispatcher.Invoke(() =>
                {
                    MessageBox.Show(
                        System.Windows.Application.Current.MainWindow,
                        $"Возникла ошибка при удалении документов {e.Message}",
                        "Ошибка",
                        MessageBoxButton.OK,
```

```
MessageBoxImage.Error);
                });
            }
        });
    }
    #endregion

    #region RegisterApplicationCommand
    private LambdaCommand? _registerApplicationCommand;

    /// <summary>
    /// Зарегистрировать заявку
    /// </summary>
    public ICommand RegisterApplicationCommand
    {
        get
        {
            if (_registerApplicationCommand == null)
            {
                _registerApplicationCommand = new
LambdaCommand(OnRegisterApplicationCommandExecuted,
CanRegisterApplicationCommandExecute);
            }
            return _registerApplicationCommand;
        }
    }
    private bool CanRegisterApplicationCommandExecute(object? p)
```

```
{
    return true;
}

private void OnRegisterApplicationCommandExecuted(object? p)
{
    Task.Run(async() =>
    {
        try
        {
            await _saveApplicationUseCase.ExecuteAsync(
                Client,
                Description,
                Documents,
                DateTime.Now);
        }
        catch (Exception e)
        {
            System.Windows.Application.Current.Dispatcher.Invoke(() =>
            {
                MessageBox.Show(
                    System.Windows.Application.Current.MainWindow,
                    $"Возникла ошибка {e.Message}",
                    "Ошибка",
                    MessageBoxButton.OK,
                    MessageBoxImage.Error);
```

```
});
            }
        });
    }
    #endregion

    #region RegisterNewClientCommand
    private LambdaCommand? _registerNewClientCommand;

    /// <summary>
    /// Зарегистрировать нового клиента
    /// </summary>
    public ICommand RegisterNewClientCommand
    {
        get
        {
            if (_registerNewClientCommand == null)
            {
                _registerNewClientCommand = new
LambdaCommand(OnRegisterNewClientCommandExecuted,
CanRegisterNewClientCommandExecute);
            }
            return _registerNewClientCommand;
        }
    }

    private bool CanRegisterNewClientCommandExecute(object? p)
```

```
{
        return true;
    }

    private void OnRegisterNewClientCommandExecuted(object? p)
    {
        Task.Run(() =>
        {
            try
            {
                _navigateToRegisterNewClientUseCase.Navigate();
            }
            catch (Exception e)
            {
                System.Windows.Application.Current.Dispatcher.Invoke(() =>
                {
                    MessageBox.Show(
                        System.Windows.Application.Current.MainWindow,
                        $"Возникла ошибка {e.Message}",
                        "Ошибка",
                        MessageBoxButton.OK,
                        MessageBoxImage.Error);
                });
            }
        });
    }
```

```
#endregion

    #region ReinitializeCommand
    private LambdaCommand? _reinitializeCommand;

    /// <summary>
    /// Обновить контекст
    /// </summary>
    public ICommand ReinitializeCommand
    {
        get
        {
            if (_reinitializeCommand == null)
            {
                _reinitializeCommand = new
LambdaCommand(OnReinitializeCommandExecuted,
CanReinitializeCommandExecute);
            }
            return _reinitializeCommand;
        }
    }

    private bool CanReinitializeCommandExecute(object? p)
    {
        return true;
    }
```

```
private void OnReinitializeCommandExecuted(object? p)
        {
            Task.Run(() =>
            {
                try
                {
                    _ = InitializeWithTimeoutAsync();
                }
                catch (Exception e)
                {
                    System.Windows.Application.Current.Dispatcher.Invoke(() =>
                    {
                        MessageBox.Show(
                            System.Windows.Application.Current.MainWindow,
                            $"Возникла ошибка {e.Message}",
                            "Ошибка",
                            MessageBoxButton.OK,
                            MessageBoxImage.Error);
                    });
                }
            });
        }
        #endregion

        private async Task InitializeWithTimeoutAsync()
        {
```

```
using var cts = new CancellationTokenSource(TimeSpan.FromSeconds(30));
            var token = cts.Token;
            await Task.Run(async () =>
            {
                await InitializeAsync(token);
            }, token);
        }
        private async Task InitializeAsync(CancellationToken cancellationToken)
        {
            try
            {
                IsDataLoaded = false;
                await InitializeClientsAsync(cancellationToken);
                IsDataLoaded = true;
            }
            catch (OperationCanceledException)
            {
                System.Windows.Application.Current.Dispatcher.Invoke(() =>
                {
                    MessageBox.Show(
                        System.Windows.Application.Current.MainWindow,
                        "Инициализация не завершена за установленное время, приложение будет закрыто",
                        "Ошибка",
                        MessageBoxButton.OK,
                        MessageBoxImage.Error);
                    System.Windows.Application.Current.Shutdown();
```

```
});
            }
            catch (Exception)
            {
                System.Windows.Application.Current.Dispatcher.Invoke(() =>
                {
                    MessageBox.Show(
                        System.Windows.Application.Current.MainWindow,
                        "Возникла ошибка при инициализации, приложение буде закрыто",
                        "Ошибка",
                        MessageBoxButton.OK,
                        MessageBoxImage.Error);
                    System.Windows.Application.Current.Shutdown();
                });
            }
        }
        private async Task InitializeClientsAsync(CancellationToken cancel = default)
        {
            Clients = await _getAllClientsUseCase.ExecuteAsync(cancel);
        }
    }
}
namespace BuildCompanyRequestManager.Application.Interfaces
{
    public interface IDocumentsRequestor
    {
```

```
IEnumerable<Document>? Request();
    }
}
namespace BuildCompanyRequestManager.Application.Interfaces
{
    public interface INavigationService
    {
        void CloseView<TView>() where TView : IView;
        void ShowView<TView>() where TView : IView;
        void ShowErrorView(string error);
        void ShowInfoView(string info);
        bool ConfirmView(string question);
        void ReOpenView<TView>() where TView : IView;
        void HideView<TView>() where TView : IView;
    }
}
namespace BuildCompanyRequestManager.Application
{
    public class PasswordVerificationUseCase : IPasswordVerificationUseCase
    {
        private readonly IEmployeeRepository _employeeRepository;
        private readonly PasswordService _passwordService;

        public PasswordVerificationUseCase(IEmployeeRepository
employeeRepository, PasswordService passwordService)
        {
```

```
_employeeRepository = employeeRepository ?? throw new ArgumentNullException(nameof(employeeRepository));
        _passwordService = passwordService ?? throw new ArgumentNullException(nameof(passwordService));
    }

    public UserRole VerifyPassword(string username, string password)
    {
        var user = _employeeRepository.FindByUsername(username);
        if (user == null)
            throw new AuthenticationException($"Пользователь с именем '{username}' не найден или пароль неверен.");
        var isPasswordIncorrect = _passwordService.VerifyPassword(password, user.PasswordHash);
        if (!isPasswordIncorrect)
            throw new AuthenticationException($"Пользователь с именем '{username}' не найден или пароль неверен.");
        return user.UserRole;
    }

    public async Task<UserRole> VerifyPasswordAsync(string username, string password, CancellationToken cancel = default)
    {
        var user = await _employeeRepository.FindByUsernameAsync(username.ToLower(), cancel);
        if (user == null)
            throw new AuthenticationException($"Пользователь с именем '{username}' не найден или пароль неверен.");
```

```
var isPasswordIncorrect = _passwordService.VerifyPassword(password, user.PasswordHash);
            if (!isPasswordIncorrect)
                throw new AuthenticationException($"Пользователь с именем '{username}' не найден или пароль неверен.");
            return user.UserRole;
        }
    }
}
```