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

Институт математики, физики и информационных технологий
(наименование института полностью)

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 сопровождения заказов (на примере сервисного центра Ref:Port)»

Обучающийся А. Г. Крдян
(Инициалы Фамилия) (личная подпись)

Руководитель Н. Н. Казаченок
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2022

Аннотация

Темой бакалаврской работы является «Разработка информационной системы сопровождения заказов (на примере сервисного центра Ref:Port)».

Цель работы – автоматизация учета и сопровождения заказов в сервисном центре Ref:Port.

Во введении определена актуальность темы бакалаврской работы, выявлены объект и предмет исследования, определены задачи для достижения поставленной цели.

В первой главе проанализирована технико-экономическая характеристика сервисного центра Ref:Port, проведено моделирование процессов учета и сопровождения заказов.

Во второй главе представлена характеристика комплекса задач и обоснование необходимости автоматизации, определены требования к разрабатываемой системе.

В третьей главе представлено физическое проектирование информационной системы сопровождения заказов.

В выполненной работе проведенные расчеты подтверждают эффективность предложенных мероприятий по улучшению учета заказов в сервисном центре Ref:Port и повышению эффективности их использования.

Оглавление

Введение

Для того чтобы на предприятии принимались верные решения, чтобы предприятие могло эти решения принимать достаточно быстро, необходимо наладить учёт сопровождения заказов. Именно от скорости получения необходимой информации зависит управление фирмой.

Чтобы автоматизировать процессы учета и сопровождения заказов необходимо разработать информационную систему. Ведь она сможет обеспечить ввод, хранение, редактирование и получение данных для формирования документации и отчетности.

Актуальность ВКР обусловлена тем, что в сервисном центре Ref:Port сотрудники ведут учет сопровождения заказов с помощью Excel и Word. Однако у данного общества с ограниченной ответственностью есть особенности, которые нужно учитывать во время ведения тех или иных сделок через систему учета сопровождения заказов.

Объект исследования – деятельность отдела продаж сервисного центра Ref:Port.

Предмет исследования – автоматизация процесса учета сопровождения заказов в отделе продаж сервисного центра Ref:Port.

Цель бакалаврской работы – автоматизация учета сопровождения заказов в сервисном центре Ref:Port. Это обеспечит легкость и быстроту выполнения начисления стоимости заказов, а также оформления и получения данных о заказах клиентов.

Задачи бакалаврской работы:

- анализ предметной области;
- анализ возможных методов проектирования решения поставленных задач;
- анализ существующих программных средств;
- проектирование информационной системы учета сопровождения заказов в сервисном центре Ref:Port;

- разработка информационной системы учета сопровождения заказов в сервисном центре Ref:Port;
- тестирование и внедрение информационной системы учета сопровождения заказов в сервисном центре Ref:Port.

Метод исследования – изучение текущего состояния бизнес-процессов в сфере продаж продукции и услуг в сервисном центре Ref:Port, изучение литературы.

Работа состоит из введения, трех разделов, заключения, списка используемой литературы и используемых источников, включает 58 страниц, 44 рисунка, 14 таблиц.

Глава 1 Технико-экономическая характеристика сервисного центра Ref:Port

1.1 Описание деятельности сервисного центра Ref:Port

Ref:Port – это сервисный центр, который принадлежит ООО «Валди».

Виды деятельности Ref:Port:

- ремонт компьютерной техники;
- ремонт смартфонов;
- продажа аксессуаров и комплектующих для оборудования;
- продажа техники Apple.

Процесс ремонта техники в ремонт осуществляется следующим образом:

- клиент сдает свою технику в сервисный центр;
- выдается договор оказания услуг, а также акт приема-передачи техники;
- администратор выдает технику мастеру;
- мастер проводит диагностику, выявляет недостатки, составляет план работ, необходимых для их устранения;
- администратор связывается с клиентом для согласования суммы ремонта;
- при положительном решении проводятся ремонтные работы и техника возвращается клиенту в срок, указанный в договоре;
- при отрицательном решении техника возвращается клиенту без проведения ремонтных работ в срок, указанный в договоре.

Сроки ремонта в каждом случае индивидуальны, однако они не могут превышать 45 дней. Поэтому мастер должен руководствоваться порядком, установленным нормативными документами сервисного центра Ref:Port, а также нормально-правовыми актами РФ.

В сервисном центре Ref:Port сделки заключаются с физическими лицами.

Система управления Ref:Port основана на положениях Устава и должностных инструкциях.

1.2 Организационная структура управления сервисным центром Ref:Port и ее характеристика

Схема организационной структуры Ref:Port представлена на рисунке 1.

Рисунок 1 – Организационная структура сервисного центра Ref:Port

На схеме показано линейное построение структурных подразделений организации, а также распределение функций управления между данными подразделениями.

В сервисном центре Ref:Port реализуется принцип единоначалия [26, с. 15].

Основные уровни управления в системе, которая подходит для сервисного центра Ref:Port:

– в высшем уровне деятельность руководителя обуславливается стратегиями и целями развития организации;

– средний уровень объединяет руководителей звена с их аппаратом. Менеджеры среднего уровня решают задачи функциональной специфики;
– низший уровень объединяет руководителей нижнего уровня. В их обязанность ходит ответственность за работу с людьми.

При линейно-функциональной структуре управления руководитель (директор) является высшим руководящим звеном [25, с. 35].

Ниже перечислены должностные инструкции сотрудников отдела по ремонту техники:

– запись клиентов на ремонт техники;
– прием техники от клиентов для последующего ремонта;
– ремонт техники;
– выдача техники клиентам после ремонта;
– предоставление ежедневных, еженедельных, ежемесячных отчетов по заказам клиентов сервисного центра;
– оформление договоров с клиентами;
– оформление документов на оплату;
– передача документов на оплату в бухгалтерию;
– сопровождение заказов, а также информирование клиента об изменении статуса заказа и ходе его выполнения;
– ведение отчетной документации;
– ведение информационной базы данных о заказах и клиентах.

В отделе ремонта ведут деятельность инженер (мастер) по гарантийному обслуживанию, сервисный инженер (мастер), менеджер (администратор). К должностным обязанностям менеджера (администратора) относится:

– консультирование клиента по вопросам ремонта (лично или по телефону);
– прием техники в ремонт, согласование сроков ремонта, стоимости;
– оформление документов при приеме и возврате техники клиента;

– продажа аксессуаров, деталей и техники.

К должностным обязанностям сервисного инженера относится:

– выполнение диагностики и обслуживание техники клиента по регламенту производителя;
– ремонт, сборка/разборка техники клиента;
– подготовка документации о выполненных работах;
– консультация клиентов.

К должностным обязанностям инженера гарантийного обслуживания относится:

– гарантийное обслуживание техники клиентов;
– выставление гарантийных требований и получение возмещения за ремонт;
– отчетность по итогам работ.

Сервисный центр Ref:Port имеет следующее оборудование:

– 3 стационарных компьютера;
– 2 МФУ;
– 1 рабочий телефон;
– 1 копировальный аппарат;
– паяльное оборудование.

Основные компоненты рабочей станции:

– процессор AMD FX-8350;
– видеокарта Radeon RX 560 4gb;
– ОЗУ 8gb;
– дисковые накопители WD Black 1TB;
– монитор MSI MAG 27CQ;
– МФУ лазерное Pantum M6800FDW;
– блок питания Aerocool 600w.

Также на рабочих станциях установлены такие программные продукты как:

– Антивирус 360 Total;
– Правовая система «Гарант»;
– Microsoft Office 2007;
– Adobe Reader 8;
– Yandex;
– WinRAR;
– Hdd sentinel;
– Hdd regenerator;
– Aida64;
– 1С Предприятие 8.3.

В качестве операционной системы используется Windows 10.

1.3 Структурно-функциональная диаграмма организации бизнес-процесса учета заказов «Как есть» и ее описание

При построении модели бизнес-процессов, как правило, используется одна из следующих нотаций [1-4]:

– Business Process Model and Notation (BPMN);
– Event-Driven Process Chain (EPC);
– Integrated Computer Aided Manufacturing Definition (IDEF0).

BPMN предназначена для отображения алгоритма действия процесса. Помимо задач, выполняемых при реализации процесса, на схеме отображаются документы, события, выбор альтернативных потоков.

Особенностью EPC-нотации является чередование событий и функций. С одной стороны, это позволяет представить подробное описание процесса, с другой – на схемах часто присутствуют тривиальные события.

Нотация IDEF0 предполагает построение иерархических моделей. На верхнем слое иерархии процесс обозначается как единое целое, далее следует

детализация как самого процесса, так и его подпроцессов. Глубина детализации ограничивается решаемой задачей [23, с. 382].

Для построения модели информационной системы сервисного центра выбрана нотация BPMN, так как данная нотация позволяет продемонстрировать подробную реализацию моделируемого процесса.

При моделировании бизнес-процесса учета заказов необходимо построить следующие модели:

– «как есть» – отображает реализацию процесса до внедрения информационной системы;
– «как должно быть» – демонстрирует реализацию процесса с использованием информационной системы.

Схема процесса учета заявок указана на рисунке 2.

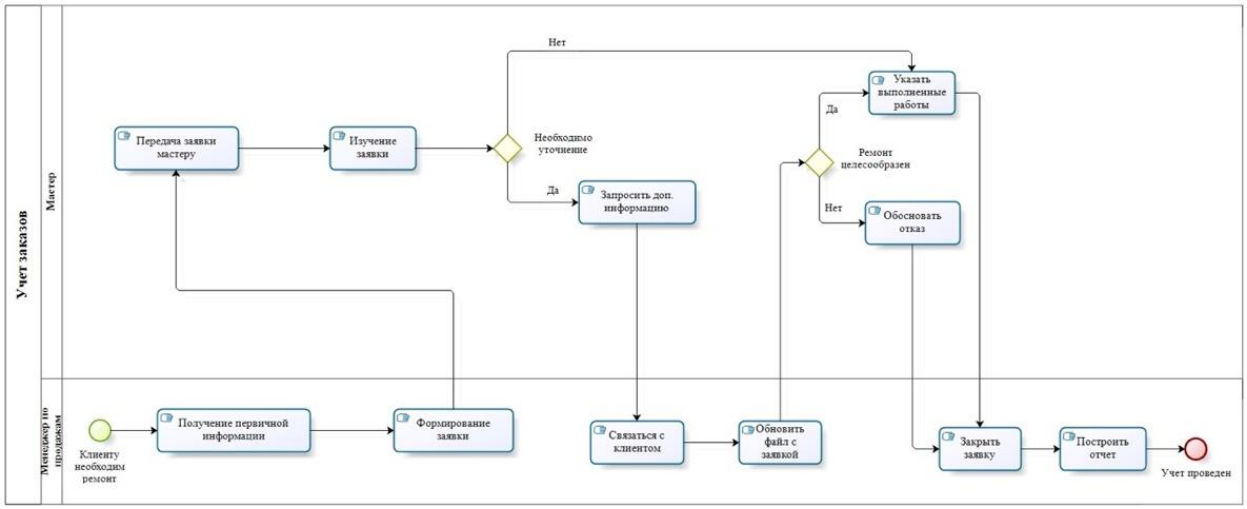


Рисунок 2 – Схема процесса учета заявок («Как есть»)

В настоящее время для реализации процесса учета заказов применяются инструменты из пакета MS Office: Word и Excel. Хранение заявок в файлах имеет следующие недостатки:

– происходит дублирование файлов, в которых содержится разная информация, из-за чего выполнение заказа задерживается. Это вызывает недовольство клиентов;

– поиск по файлам занимает большое количество времени, необходимо использовать дополнительные программные средства, либо искать по файлам вручную;
– проведение анализа файлов с целью построения отчетов также занимает большое количество времени.

Для реализации указанных выше недостатков необходимо использование единой системы для хранения и обработки данных по заявкам.

1.4 Анализ существующих разработок для автоматизации комплекса задач

Для автоматизации комплекса задач на сегодняшний день разработаны различные системы, которые позволяют хранить данные о заказах предприятия и его клиентах.

Безусловно, наиболее востребованной считается «1С: Управление торговлей 8». Данная система создана с целью повышения эффективности бизнеса компании. Удобство данной информационной системы в том, что она рассчитана на различные торговые операции [21, с. 37].

Немало известным является CRM-система «Битрикс24», которым пользуется большое количество предприятий. Данная система удобна своей простотой, более того, она может управлять всеми процессами, которые происходят в торговой компании. Учет заказов – одно из главных направлений «Битрикс24». Система имеет ряд преимуществ, однако ими можно пользоваться лишь при оплате определенного тарифа.

Новой, но в то же время, уже конкурентной является информационная система, представленная для клиентов малого бизнеса СберБизнесом. «SberCRM» подходит именно для небольших предприятий, которые нуждаются в ведении учета заказов. Стоит учесть, что данная система является «конструктором», который можно без особых усилий подстроить под свою компанию либо отдельный филиал/подразделение.

Безусловно, существует множество информационных систем, однако с этими системами удалось выявить особенности и недостатки каждой из них.

Ниже проведена сравнительная характеристика, которая позволит оценить сильные и слабые стороны рассмотренных информационных систем. При оценивании учтены следующие функции систем:

- ввод, хранение данных клиентов и их заказах;
- ввод, хранение данных об услугах;
- возможность отследить этап заказа;
- возможность поиска данных о заказе по фильтрам;
- возможность учета скидок и расходов при выведении итоговой суммы заказа;
- создание, хранение, печать документов по заказу;
- возможность редактирования сделки;
- стоимость;
- наличие приложения для смартфона;
- эффективность технической поддержки.

При оценивании данных критериев использована следующая шкала:

- 0 – функция отсутствует, неисправна;
- 1 – функция реализована частично, цена не оправдывает возможности системы, неудобное пользование системой;
- 2 – функция реализована практически полностью, цена средняя, некоторые задачи не поддерживаются, пользование системой не очень удобное;
- 3 – функция реализована, пользование системой удобное, цена выгодная.

Результаты сравнения готовых программных решений отражены в таблице 1.

Таблица 1 – Сравнительная характеристика программных решений

Необходимые функции информационной системы	Программные решения		
	1С «Торговля и склад»	CRM-система «Битрикс24»	CRM-система «SberCRM»
Ввод, хранение данных клиентов и их заказах	3	3	3
Ввод, хранение данных об услугах	2	2	3
Возможность отследить этап заказа	2	3	3
Возможность поиска данных о заказе по фильтрам	2	2	3
Возможность учета скидок и расходов при выведении итоговой суммы заказа	2	2	3
Создание, хранение, печать документов по заказу	3	2	2
Возможность редактирования сделки	2	3	3
Стоимость	2	2	3
Наличие приложения для смартфона	3	2	0
Эффективность технической поддержки	3	1	3
Общая оценка системы	24	22	26

По данным сравнительной таблицы максимальную оценку получила программа «SberCRM». Эта система является комплексной, в ней содержатся функции, которые работают исправно и достаточно хорошо. Однако даже эта система не набрала наивысшее количество баллов (30 баллов), соответственно, даже она далеко не идеальна.

Результаты сравнения – основание для разработки новой информационной системы, исходя из потребностей сервисного центра Ref:Port.

1.5 Постановка задачи на разработку информационной системы сопровождения заказов

Выявлена необходимость в разработке информационной системы для учета заказов. Приобретать одно из готовых решений для данной организации является нецелесообразным решением, так как оно повлечет за собой финансовые потери. Система сопровождения заказов должна как помочь просто вести деятельность организации, так и стать помощником для

расширения клиентской базы посредством взаимодействия с уже наработанной клиентской базой.

Информационная система предназначена для автоматизации бизнес-процесса по ведению учета заявок сервисного центра. Структура системы должна быть построена по архитектуре «клиент-сервер». При реализации информационной системы необходимо использовать объектно-ориентированный подход.

Разрабатываемая информационная система должна соответствовать следующим функциональным требованиям:

- авторизация с использованием персональной учетной записи;
- добавление, изменение, удаление данных по заявкам, услугам, ремонтируемым устройствам;
- редактирование списка услуг отдельной заявки;
- построение отчета, содержащего результаты расчета статистических параметров, по данным за выбранный период;
- отчет должен содержать результаты расчета следующих величин: общее количество заявок за период, количество отмененных заявок, средний срок обработки заявки (дни), среднюю стоимость заявки, среднее количество оказанных услуг.

Система должна обладать визуальным интуитивно-понятным интерфейсом. Все элементы интерфейса должны иметь стандартное назначение. Любое действие пользователя должно сопровождаться ответной реакцией информационной системы.

Необходимо реализовать проверку корректности входных данных. При вводе некорректных значений необходимо отображать сообщение об ошибке, в котором содержится подсказка к исправлению возникшей проблемы. Выходные данные должны быть представлены в удобном для восприятия виде.

1.6 Разработка модели бизнес-процесса учета заказов «Как должно быть»

Сервисный центр Ref:Port занимается ремонтом компьютерной техники, телефонов. Компания выполняет ремонт следующих видов:

- гарантийный – ремонт реализованного покупателю товара, который не отработал указанный производителем гарантийный срок;
- послепродажный – ремонт техники, отработавшей гарантийный период, но не выработавшей срок службы.

Алгоритм процесса по учету заказов (модель «как должно быть») с использованием информационной системы приведен на рисунке 3.

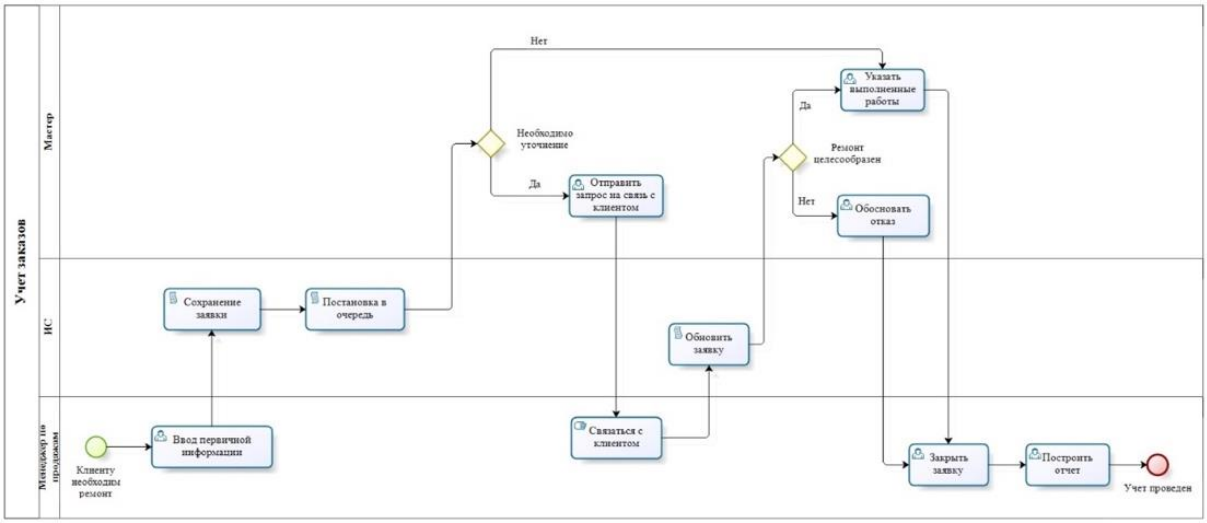


Рисунок 3 – Схема процесса учета заявок («Как должно быть»)

Как видно из рисунка 3, выполнение процесса начинается с обращения клиента в сервисный центр. Менеджер получает от клиента первичную информацию: находится ли устройство на гарантии, суть проблемы. После создания, заявка ставится в очередь на выполнение.

В ходе выполнения ремонта у мастера может возникнуть необходимость что-либо уточнить у клиента. В таком случае, мастер направляет запрос на уточнение менеджеру, а менеджер связывается с клиентом. После получения

дополнительной информации происходит обновление заявки. Затем мастер либо выполняет ремонт, либо обосновывает его нецелесообразность или невозможность после получения новой информации.

Для получения статистической информации о проводимых ремонтах, менеджеры по продажам строят отчет по выполненным заказам за выбранный период. Отчет содержит общее количество заявок, средний срок обработки заявки, среднюю стоимость заявки, количество выполненных ремонтов, количество отказов.

Выводы по главе 1

Описана деятельность сервисного центра Ref:Port с указанием ее организационной структуры управления. Проведен анализ существующих разработок, а также поставлены задачи на разработку новой информационной системы сопровождения заказов в Ref:Port.

Глава 2 Логическое проектирование информационной системы сопровождения заказов

2.1 Выбор технологии логического моделирования информационной системы сопровождения заказов

Для построения логической модели информационной системы сервисного центра оптимальным вариантом является методология UML, так как она позволяет построить модель проектируемой системы в рамках объектно-ориентированной концепции проектирования.

Данная нотация предназначена для моделирования процессов и систем любого типа и масштаба. UML полностью реализует объектно-ориентированную концепцию проектирования систем. Диаграммы языка моделирования разделяются на:

- структурные, отображают модули всей системы, либо подсистемы, а также взаимосвязь модулей;
- поведенческие, отображают функциональные возможности системы или подсистемы.

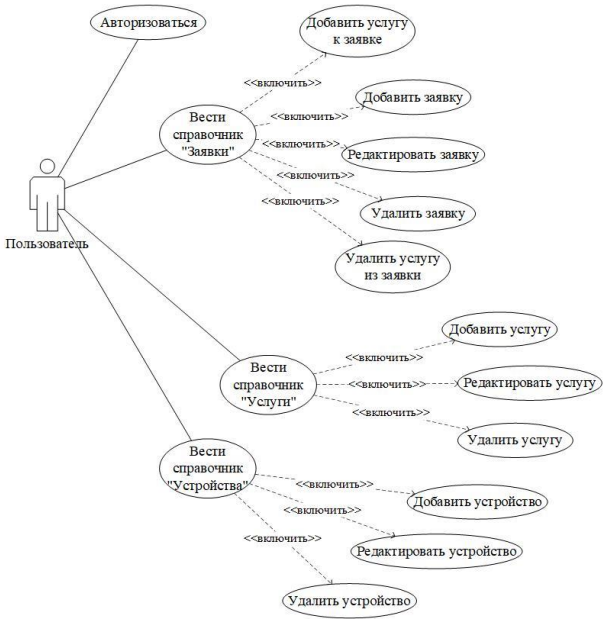


Рисунок 4 – Логическая модель информационной системы сервисного центра

Как видно из рисунка 4, пользователю информационной системы доступно ведение справочников с данными, построение статистического отчета, авторизация под персонально учетной записью. Пользователем информационной системы является менеджер сервисного центра, работающий на приеме заявок от клиентов.

2.2 Информационное обеспечение информационной системы сопровождения заказов

2.2.1 Используемые классификаторы и системы кодирования

Для однозначного определения экземпляров сущностей предметной области, используемых для решения пользовательских задач, применяются идентификаторы. Идентификаторы относятся к порядковой системе кодирования. Характеристики кодируемых объектов представлены в таблице 2.

Таблица 2 – Коды для объектов

Название объекта кодируемого множества	Значимость кода	Система кодирования
ID заявки	7	порядковая
ID услуги	3	порядковая
ID сотрудника	3	порядковая
ID устройства	7	порядковая
ID пользователя	3	порядковая

Идентификаторы позволяют создать уникальную отметку для каждого экземпляра сущности.

Входными данными являются: ФИО и телефон клиента, дата создания заявки, название устройства, описание проблемы. Для эффективной обработки заявки в ней необходимо указать данные клиента, описание проблемы с устройством. Описание должно быть кратким, но, при этом, содержать всю необходимую для мастера информацию.

Характеристики реквизитов заявки приведены в таблице 3.

Таблица 3 – Реквизиты для заявки

Название поля	Идентификатор	Длина
ФИО	name	300
телефон	phone	20
дата создания	opendate	15
название	title	255
проблема	comment	3000

Заявка от клиента содержит контактные данные клиента, название ремонтируемого устройства, краткое описание проблемы.

2.2.2 Характеристика выходной информации

После создания заявки ее необходимо поставить в очередь на исполнение.

Выходной информацией является статистический отчет. Макет статистического отчета представлен на рисунке 5.

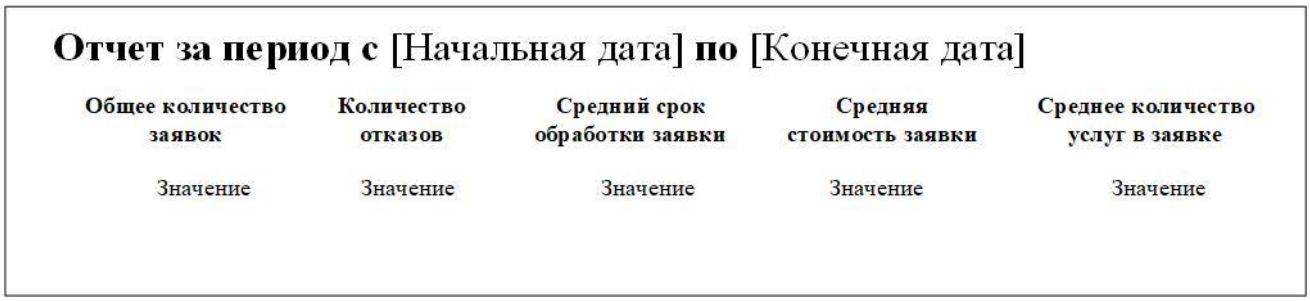


Рисунок 5 – Макет отчета

Отчет содержит временной диапазон, за который взяты данные, а также набор статистических характеристик, рассчитанных на основе исходных данных. Реквизиты отчета представлены в таблице 4.

Таблица 4 – Структура отчета

Название поля	Идентификатор	Длина
Начальная дата	Start_date	15
Конечная дата	Close_date	15
общее количество заявок	total	8
средний срок обработки заявки	Avg_time	3
среднюю стоимость заявки	Avg_price	6
количество выполненных ремонтов	Total_maked	7
количество отказов	Total_refused	7

Отчет содержит набор статистических характеристик, позволяющих оценить деятельность сервисного центра.

2.3 Проектирование базы данных информационной системы сопровождения заказов

2.3.1 Выбор технологии проектирования базы данных информационной системы сопровождения заказов

В своей книге В. И. Дубейковский обозначил, что «проектирование базы данных включает в себя следующие этапы: построение концептуальной схемы по объектам предметной области, построение логической модели базы данных. Концептуальную схему целесообразно строить с использованием нотации Чена» [7, с. 382]. Для построения логической модели данных оптимальным вариантом является нотация IDEF1X [8-12]. Объекты логической модели:

- сущность: объект предметной области, используемый для решения пользовательских задач;
- отношение: связи между сущностями, имеет кратность, указывающую на количество экземпляров сущностей, задействованных в связи с каждой из сторон;
- атрибут: параметр, позволяющий идентифицировать сущность.

Существуют следующие виды связей между сущностями, используемые в нотациях Чена и IDEF1X:

– один к одному: с обеих сторон в связи участвует по одному экземпляру сущностей, обе сущности являются равноправными;
– один ко многим: со стороны главной сущности в связи участвует один экземпляр, а со стороны зависимой сущности участвует несколько экземпляров;
– многие ко многим: сущности равноправны, с каждой из сторон участвует несколько экземпляров [5-6].

Благодаря данной нотации на схеме можно отобразить основные сущности и связи между ними.

2.3.2 Разработка концептуальной модели данных информационной системы сопровождения заказов

Концептуальная схема предназначена для определения основных сущностей, необходимых для решения задач. На рисунке 6 представлена концептуальная схема для базы данных проектируемой системы.

В ходе анализа предметной области выделены следующие сущности:

– Пользователь – пользователь информационной системы по ведению учета заявок;
– Сотрудник – сотрудник сервисного центра;
– Устройство – телефон, компьютер или другое устройство, переданное клиентом для ремонта;
– Заявка – заявка на ремонт устройства;
– Услуга – ремонт или обслуживание устройства.

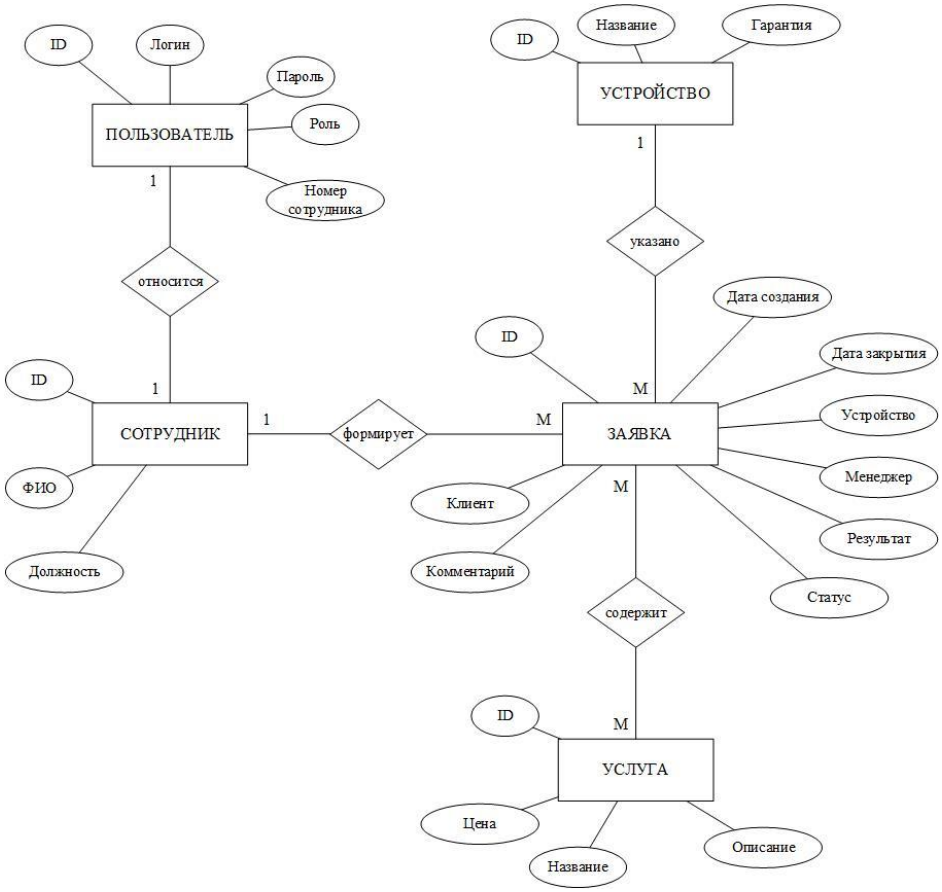


Рисунок 6 – Концептуальная схема

Концептуальная схема является основой для построения логической модели данных информационной системы сервисного центра.

2.3.3 Обоснование вида логической модели

Прежде чем выполнять построение логической модели данных, необходимо выбрать тип базы данных. Основными типами являются:

Иерархическая модель: связанная структура данных, согласно которой каждый элемент связан только с одним родителем. Иерархическая база данных имеет древовидную структуру. Данной модели соответствует ряд ограничений, которые затрудняют ее использование [13-16].

Сетевая модель: расширение для иерархической модели, согласно которому количество связей для узлов неограниченно. Данная модель является сложной в реализации и требует большое количество аппаратных ресурсов.

Реляционная модель: вся информация хранится в реляционных отношениях, имеющих предопределенную структуру. Каждое отношение соответствует одному объекту предметной области. По данным статьи голландского ученого Will van der Aalst «реляционная модель является наиболее востребованной при создании хранилищ данных на основе СУБД» [28, с. 21].

Реляционная модель данных выбрана для построения базы данных информационной системы сервисного центра. Данная модель представляет собой наиболее оптимальный вариант.

Одним из важных моментов при построении реляционной модели базы данных является проведение нормализации модели. Нормализация – это процесс приведения таблиц базы к оптимальной структуре, в которой исключено наличие дубликатов атрибутов, сложных атрибутов, включающих в себя несколько параметров (например, ФИО при нормализации разделяется на фамилию, имя и отчество), транзитивных зависимостей между неключевыми атрибутами таблиц.

Существует пять нормальных форм, однако в большинстве случаев достаточно приведение базы к третьей нормальной форме. Нормализация может быть выполнена на любом этапе проектирования базы данных.

2.3.4 Разработка логической модели данных информационной системы сопровождения заказов

Логическая модель разрабатываемой информационной системы сервисного центра приведена на рисунке 7. Данная модель является ненормализованной. Нормализация позволяет исключить дублирование, противоречивость данных [5, 17-19]. Для приведения модели к третьей нормальной форме необходимо внести следующие изменения:

- разделить атрибут ФИО на фамилию, имя и отчество;
- разделить связь «многие-ко-многим» между сущностями «Заявка» и «Услуга».

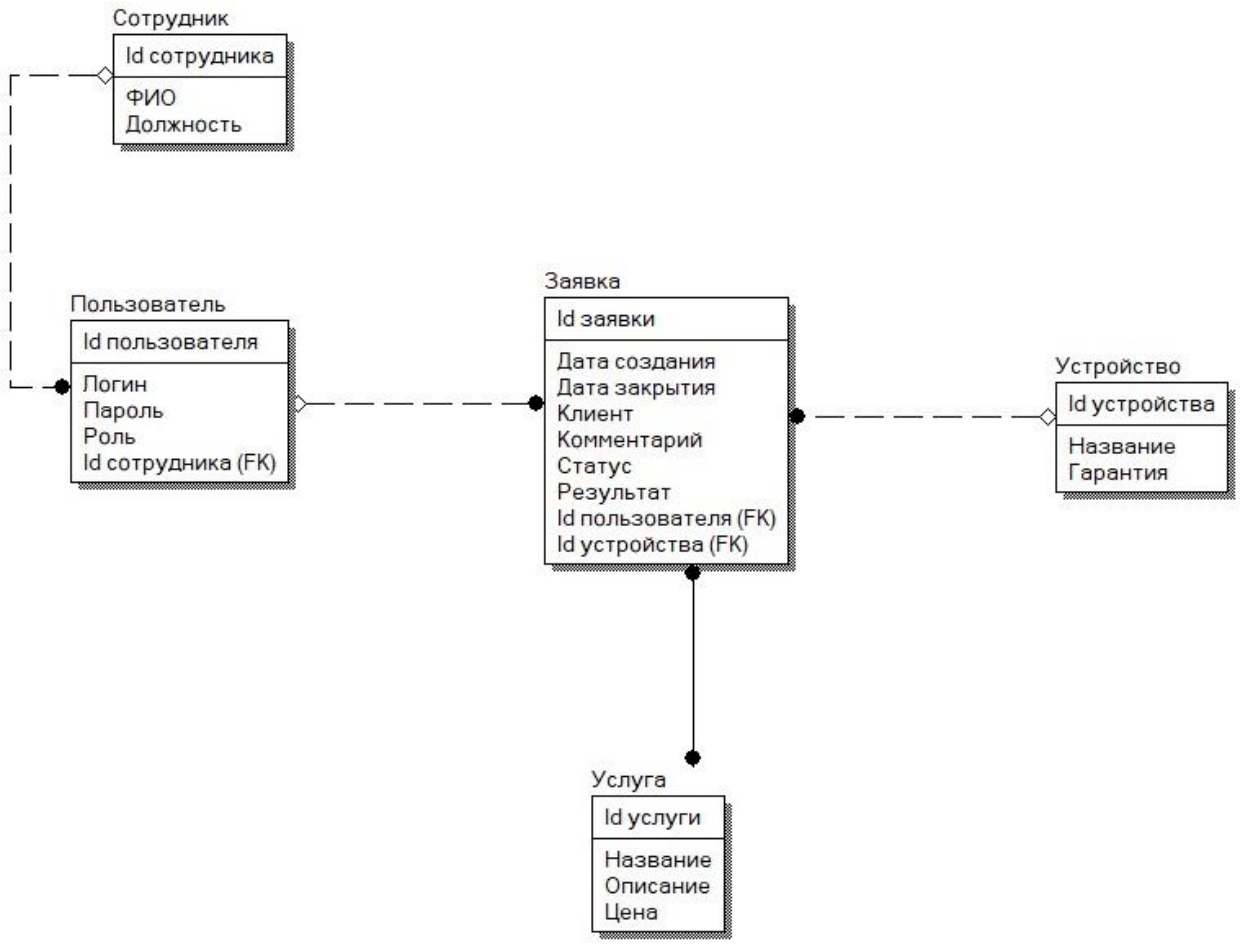


Рисунок 7 – Логическая модель (без нормализации)

Нормализованная логическая модель приведена на рисунке 8.

Якобсон А. отмечает, что «Реляционные отношения соответствуют первой нормальной форме (1НФ), так как все атрибуты имеют атомарные значения. Реляционные отношения находятся во второй нормальной форме (2НФ), так как они находятся в 1НФ и все неключевые атрибуты неприводимо зависят от ключевых. Реляционные отношения находятся в третьей нормальной форме (3НФ), так как они находятся в 2НФ и отсутствуют транзитивные зависимости между неключевыми атрибутами» [22, с 294].

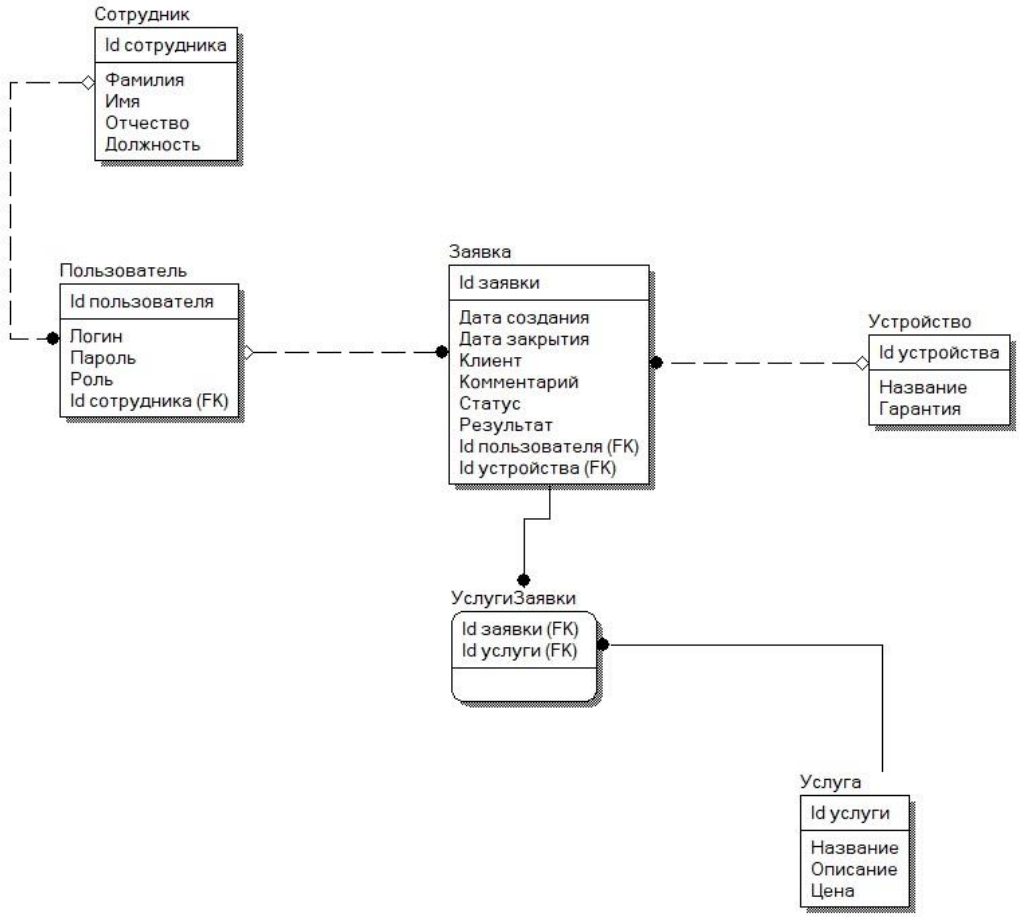


Рисунок 8 – Нормализованная логическая модель

На основе логической модели строится физическая модель данных в выбранной СУБД. В данной работе для построения базы данных будет выбрана реляционная СУБД MS SQL Server.

2.4 Требования к аппаратно-программному обеспечению информационной системы сопровождения заказов

Для эффективной работы информационной системы сервисного центра необходимо использование аппаратного обеспечения, обладающего определенными техническими характеристиками.

Для развертывания приложения необходимо наличие следующих компонент:

– сервер баз данных, на котором будет расположена база сервисного центра;
– клиентские компьютеры по числу сотрудников, работающих с информационной системой;
– локальная сеть, связывающая рабочие места персонала с сервером.

Серверное оборудование должно соответствовать следующим техническим характеристикам:

– частота процессора от 3 ГГц;
– количество ядер не менее 4;
– объем оперативной памяти не менее 10 ГБ;
– жесткие диски не менее2 шт;
– скорость вращения от 10000 об/м.

Рабочие места сотрудников должны соответствовать следующим требованиям:

– процессор с частотой от 1.6 ГГц;
– количество ядер от 2 шт;
– объем оперативной памяти не менее 4 Гб;
– размер жесткого диска от 512 Мб.

Количество рабочих мест сотрудников определяется по штатному расписанию сервисного центра.

Выводы по главе 2

Обоснован выбор технологий моделирования бизнес-процесса по сопровождению заявок, а также моделирования базы данных. Построены концептуальная схема и логическая модель данных информационной системы учета заявок в Ref:Port.

Глава 3 Физическое проектирование информационной системы сопровождения заказов

3.1 Выбор архитектуры информационной системы сопровождения заказов

Существуют следующие типы клиент-серверной архитектуры:

- файл-сервер;
- n-звенный клиент-сервер.

Файл-серверная архитектура устроена следующим образом:

- данные размещаются на отдельном компьютере (сервер);
- через сеть с сервером связываются клиентские компьютеры.

Файловый сервер не имеет инструментов для обработки данных. Клиенты скачивают данные и затем проводят их обработку. В следствии этого по сети передаются большие объемы данных, что повышает нагрузку на сеть.

Использование N-звенного клиент-сервера означает, что в сети могут быть несколько серверов, выполняющих различные функции. Даже если сервер один, то он не только хранит данные, но и проводит их обработку. Таким образом, клиенту передаются не исходные данные, а результаты их обработки. Недостатком данной архитектуры является сложная структура сети [18-19].

При разработке информационной системы для сервисного центра выбрана архитектура «2-звенный клиент-сервер». А. С. Ходарев в своих научных статьях отмечает: «В сети организации расположен один сервер, на котором хранятся данные. Клиентские компьютеры сотрудников отправляют на сервер запросы с помощью приложения. Обработка данных выполняется на сервере, средствами СУБД. Клиент получает результаты обработки. Данная архитектура обеспечит минимальную нагрузку на сеть, позволит применить оптимальную по сложности конфигурацию сети».

Для защиты компьютеров, объединенных в сеть, целесообразно использовать антивирусное программное обеспечение. Антивирус должен быть

установлен как на клиентском компьютере, так и на сервере. Также целесообразно применение других инструментов защиты, таких как брандмауэр. Помимо программных средств защиты целесообразно использовать организационные меры, выполняемые всеми пользователями информационной системы.

3.2 Выбор технологии разработки программного обеспечения информационной системы сопровождения заказов

Рассмотрим современные языки программирования и их особенности.

Java – высокоуровневый объектно-ориентированный язык программирования. Особенность языка Java заключается в том, что его код не компилируется в команды конкретного процессора, а интерпретируется в набор команд виртуальной машины Java Virtual Machine (JVM). «Именно виртуальная машина выполняет код, используя процессор. Использование виртуальной машины делает программы, написанные на Java универсальными, работающими на любой платформе» [27].

Python – язык программирования высокого уровня. Интерпретируемый, скриптовый язык, поддерживающий объектно-ориентированную концепцию. Как правило, применяется для решения узкоспециализированных задач: автоматизация и анализ данных.

PHP – язык программирования, поддерживающий как процедурный, так и объектно-ориентированный подходы. Активно используется для веб-разработки, при написании серверной части приложений (back-end). Обладает большим набором инструментов для решения различных задач, таких как подключение к базе данных, манипуляция данными из базы с помощью запросов, проверки вводимых данных и т.д.

C# представляет собой объектно-ориентированный язык, работающий под управлением платформы .NET. Интерпретируется для выполнения на

виртуальной машине. Может быть использован для создания приложений любого типа. Является продолжением линейки C→C++→C#.

В данной работе, для разработки приложения, выбран язык программирования C#. Так как данный язык полностью поддерживает объектно-ориентированную концепцию программирования, обладает большим набором инструментов для разработки приложений различного типа.

3.3 Выбор системы управления базами данных информационной системы сопровождения заказов

Фаулер М. утверждает, что «СУБД Oracle представляет собой объектно-реляционную систему управления базами данных. Является одной из наиболее распространенных на рынке СУБД как на платформе Windows, так и среди UNIX-подобных систем. Одна из причин широкого распространения Oracle – высокие эксплуатационные характеристики системы» [17, с. 428].

Сервера баз данных, выпускаемые Oracle, поставляются в одном из четырех вариантов в зависимости от размеров информационной системы, в которой предполагается использовать СУБД.

MySQL – реляционная СУБД. Ее разработчик – дочерняя компания Oracle. СУБД MySQL написана на C, C++. MySQL распространяется под GNU General Public License, а также имеет собственную коммерческую лицензию. Кроме указанных вариантов распространения, разработчики создают системы, имеющие заказанную клиентом функциональность. Наиболее полезный механизм, созданный по заказу – механизм репликации.

СУБД MS SQL Server. Основные функции:

- сбор и хранение данных, организация их обработки;
- обеспечение целостности хранимых данных;
- очистка данных с целью устранения дубликатов;
- проведение бизнес-аналитики данных;

– построение различных отчетов, контроль целостности исходных для отчета данных;
– синхронизация нескольких версий базы данных;
– проведение транзакций;
– наличие собственного клиентского приложения для работы с базой данных.

Решение от компании Microsoft распространяется как по коммерческой, так и свободной лицензии (Express-версия). СУБД рассчитана на системы любого типа и размера.

Для разработки базы данных выбрана реляционная СУБД MS SQL Server, так как данная СУБД подходит для небольших и средних проектов, обладает одной из самых высоких скоростей по обработке данных.

3.4 Разработка физической модели данных информационной системы сопровождения заказов

Физическая модель данных предназначена для отображения таблиц, входящих в базу данных, их атрибутов, а также связей между таблицами.

На физической модели представлены следующие таблицы:

– Users – пользователи информационной системы;
– Employees – сотрудники сервисного центра;
– Services – услуги сервисного центра;
– Orders – заявки от клиентов;
– Items – устройства, ремонтируемые в сервисном центре;
– ServiceOrders – услуги для каждой из заявок.

Физическая модель данных представлена на рисунке 9.

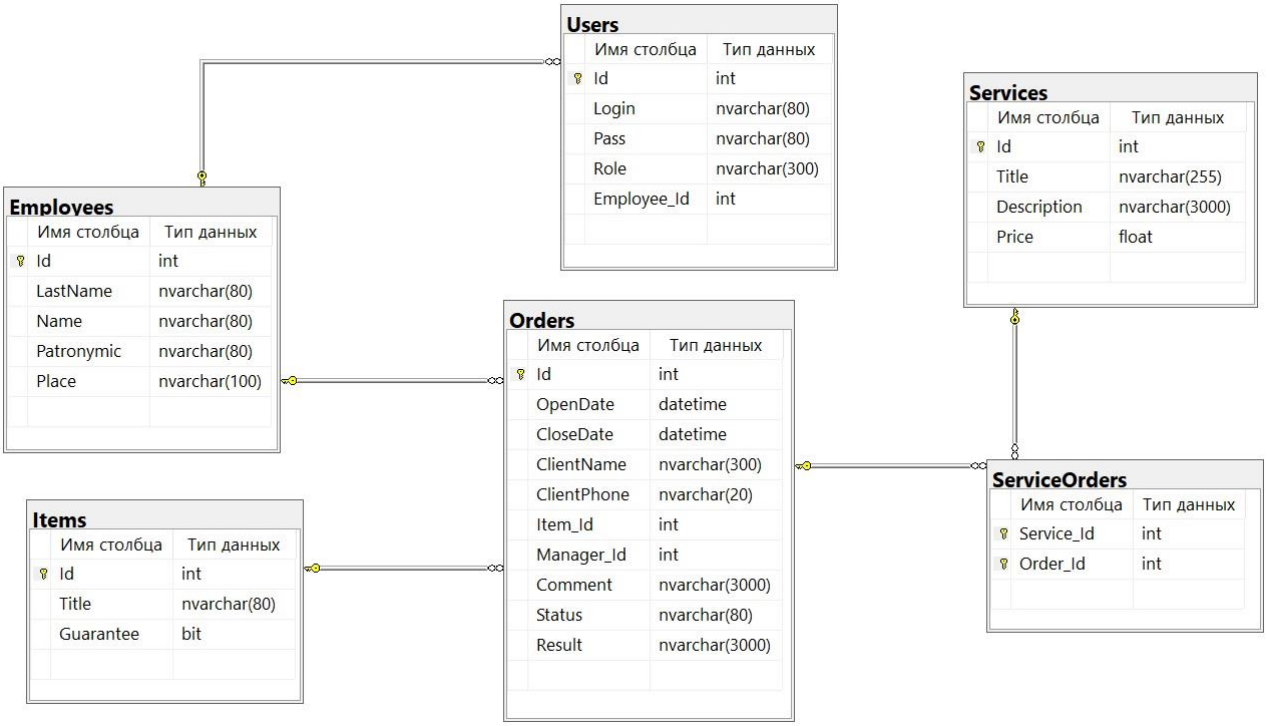


Рисунок 9 – Физическая модель данных

Характеристики полей реляционных отношений приведены в таблицах 5 - 10.

Таблица 5 – Employees (Сотрудники)

Атрибут	Тип данных	Характеристика
Id	int	Идентификатор сотрудника, PK
LastName	nvarchar(80)	Полное имя сотрудника
Name	nvarchar(80)	
Patronymic	nvarchar(80)	
Place	nvarchar(100)	Должность сотрудника

Таблица 6 – Items (Устройства)

Атрибут	Тип данных	Характеристика
Id	int	Идентификатор устройства, PK
Title	nvarchar(80)	Название устройства
Guarantee	bit	Находится ли устройство на гарантии

Таблица 7 – Users (Пользователи)

Атрибут	Тип данных	Характеристика
Id	int	Идентификатор пользователя, PK
Login	nvarchar(80)	Данные учетной записи пользователя
Pass	nvarchar(80)	
Role	nvarchar(300)	Тип учетной записи пользователя
Employee_Id	int	Номер сотрудника, которому принадлежит учетная запись, FK

Таблица 8 – Orders (Заявки)

Атрибут	Тип данных	Характеристика
Id	int	Идентификатор заявки, PK
OpenDate	datetime	Дата создания заявки
CloseDate	datetime	Дата закрытия заявки
ClientName	nvarchar(300)	Имя клиента
ClientPhone	nvarchar(20)	Номер телефона клиента
Item_Id	int	Номер устройства, FK
Manager_Id	int	Номер сотрудника, оформившего заявку, FK
Comment	nvarchar(3000)	Описание проблемы
Status	nvarchar(80)	Статус заявки
Result	nvarchar(3000)	Результат выполнения заявки

Таблица 9 – Services (Услуги)

Атрибут	Тип данных	Характеристика
Id	int	Идентификатор услуги, PK
Title	nvarchar(255)	Название услуги
Description	nvarchar(3000)	Описание услуги
Price	float	Цена услуги

Таблица 10 – ServiceOrders (Услуги Заявки)

Атрибут	Тип данных	Характеристика
Service_Id	int	Номер услуги, FK
Order_Id	int	Номер заявки, FK

Для разделения связей типа «многие-ко-многим» используются вспомогательные сущности.

3.5 Разработка программного обеспечения информационной системы сопровождения заказов

3.5.1 Схема взаимосвязи модулей приложения информационной системы сопровождения заказов

Для функций информационной системы сервисного центра (рисунок 10) используется разделение на следующие группы:

– служебные: обеспечение корректной работы приложения;

– основные: решение пользовательских задач.

К основным функциям относятся:

– сбор данных по заявкам, услугам и ремонтируемым устройствам;

– обработка данных: поиск записей, редактирование, построение отчетов;

– представление данных: отображение данных о заявках, устройствах, услугах.

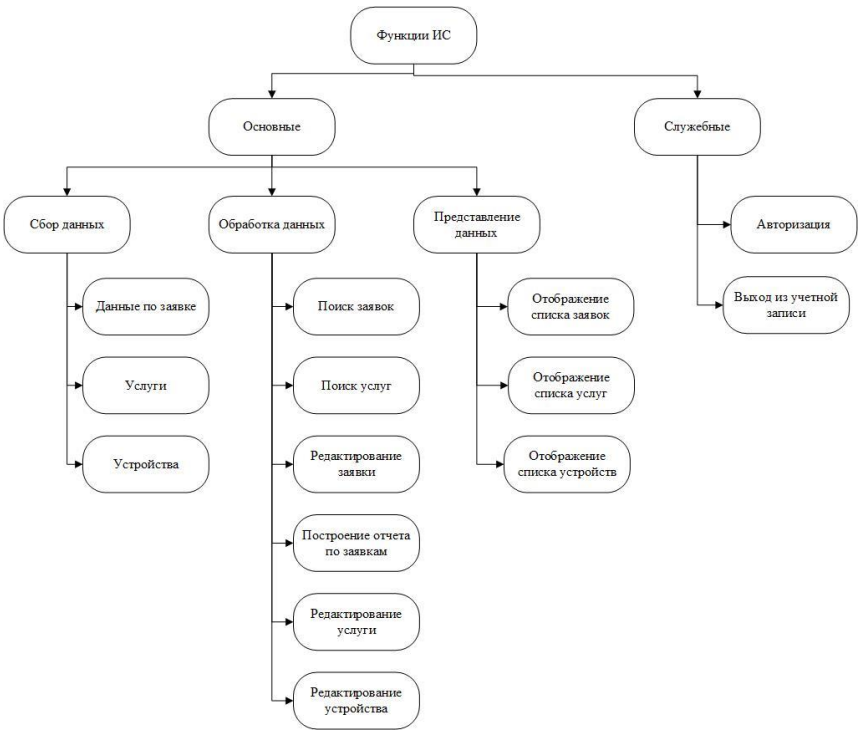


Рисунок 10 – Дерево функций

Служебными функциями являются авторизация и выход из учетной записи.

3.5.2 Описание модулей приложения информационной системы сопровождения заказов с примерами программного кода

Разработанное приложение состоит из следующих модулей:

- Models – классы-модели, на их основе формируются таблицы базы данных;
- Views – классы-представления, являются оконным интерфейсом приложения;
- ViewModels – классы, реализующие логику по обработке данных из моделей и передачу результатов обработки в представления;
- Commands – команды, используются для реализации связей между элементами интерфейса и логикой по обработке пользовательских действий;
- Controllers – классы-контроллеры, предназначены для реализации навигации в приложении.

Диаграмма пакетов представлена на рисунке 11.

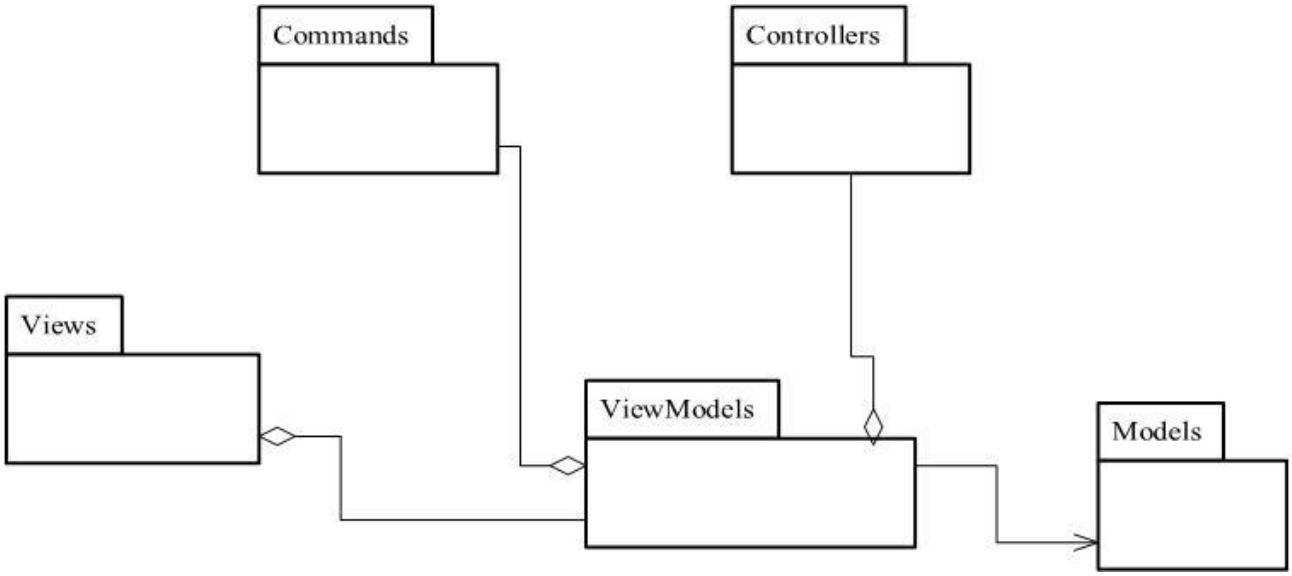


Рисунок 11 – Диаграмма пакетов

Исходный код разработанного программного продукта приведен в Приложении А и код бизнес-логики приведен в Приложении Б.

3.6 Описание функциональности информационной системы сопровождения заказов

Перед началом работы в приложении, пользователю необходимо войти в учетную запись. Форма для входа показана на рисунке 12.

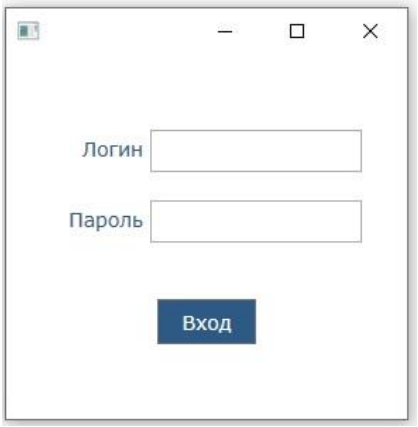


Рисунок 12 – Вход в учетную запись

Если пользователь укажет неверные данные учетной записи, то приложение отобразит следующую ошибку (рисунок 13):

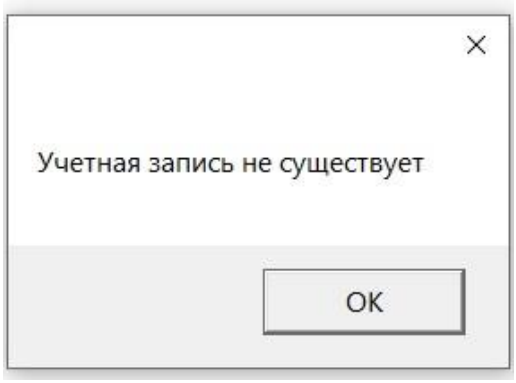


Рисунок 13 – Сообщение о неверном логине/пароле

В случае ввода корректных логина и пароля пользователь перемещается на главное окно приложения (рисунок 14).

В верхней части окна расположена таблица с заявками от клиентов. Ниже данной таблицы находится поле с дополнительной информацией, которое заполняется данными по выбранной заявке при нажатии на кнопку «Инфо».

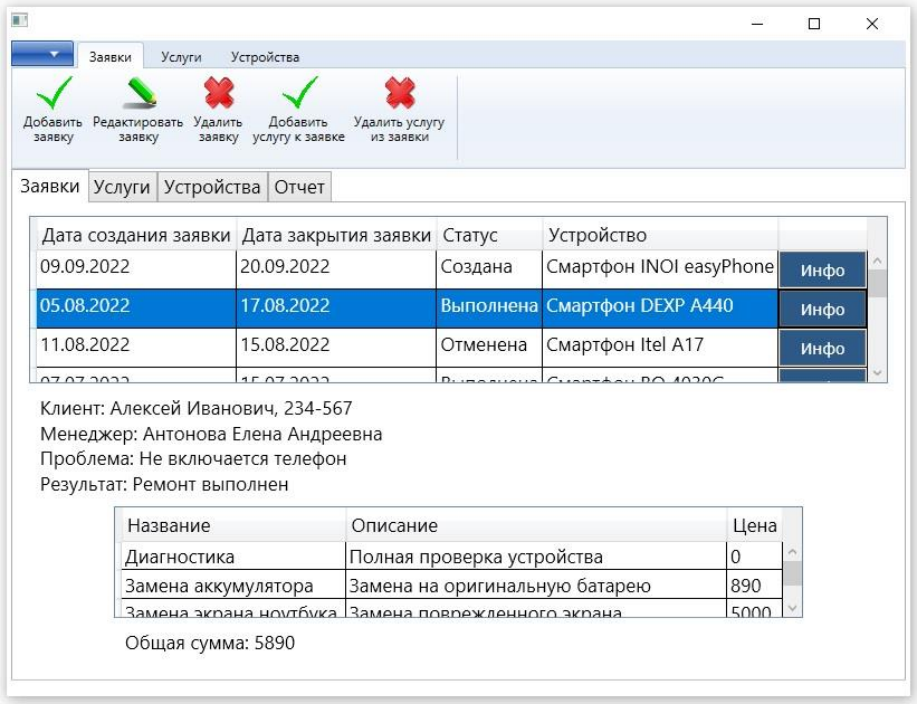


Рисунок 14 – Главная форма приложения, вкладка «Заявки»

В нижней части окна находится таблица с услугами для выбранной заявки и общая стоимость услуг.

Пункт меню «Добавить заявку» открывает окно для создания новой заявки (рисунок 15).

Для создания заявки необходимо указать даты создания и завершения заявки. Дата завершения может быть примерной, впоследствии скорректированной на дату фактического выполнения заявки.

Устройство выбирается из списка, следовательно, на момент создания заявки оно должно присутствовать в базе данных системы.

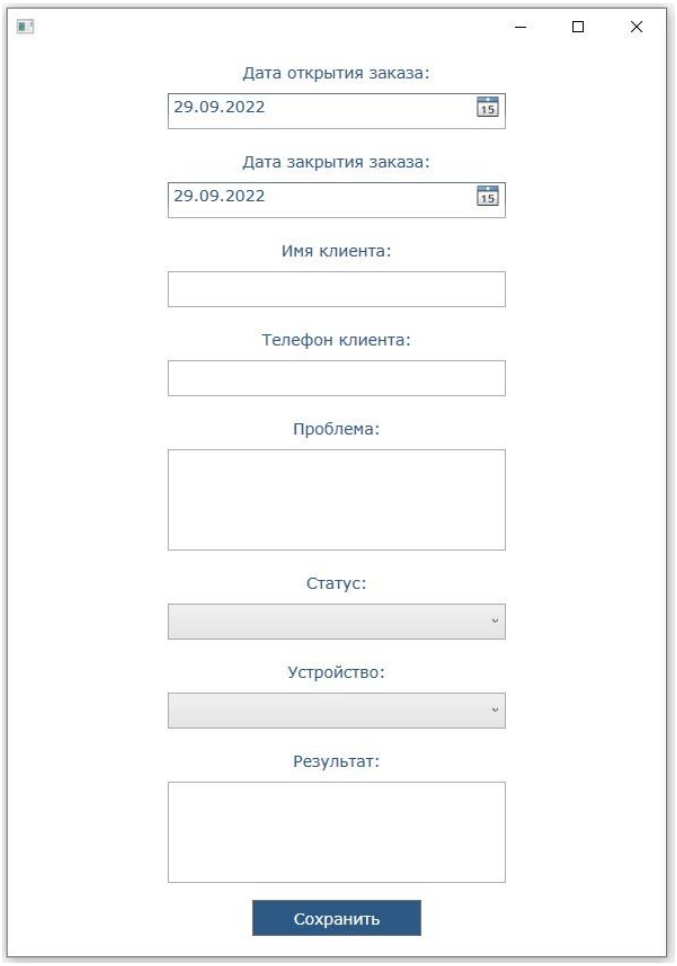


Рисунок 15 – Форма для добавления новой заявки

Все поля формы, кроме поля «Результат», являются обязательными для заполнения. Если хотя бы одно из обязательных полей останется незаполненным, то система отобразит следующее сообщение об ошибке (рисунок 16):

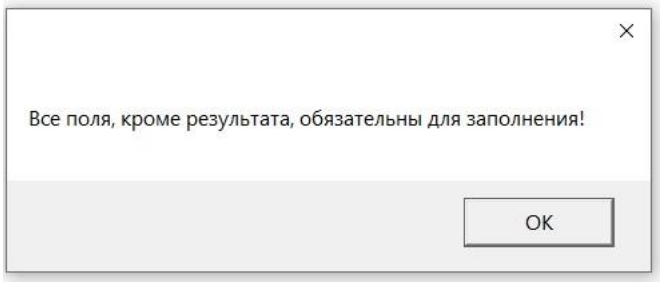


Рисунок 16 – Ошибка при создании заявки

Если все обязательные поля заполнены и даты выставлены корректно, то система отобразит следующее сообщение (рисунок 17):

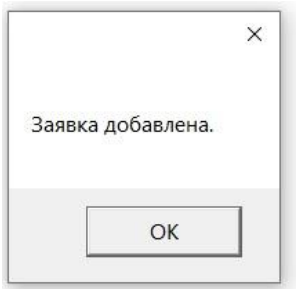


Рисунок 17 – Сообщение о создании новой заявки

Для того чтобы изменить заявку, необходимо выбрать одну из записей в верхней таблице, а затем нажать на пункт меню «Редактировать заявку». Если заявка для редактирования не выбрана, то приложение отобразит следующую ошибку (рисунок 18):

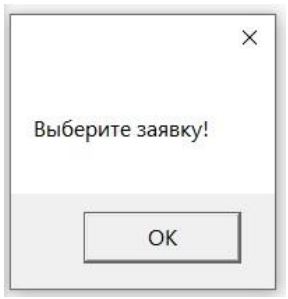


Рисунок 18 – Ошибка при редактировании заявки

При редактировании записи действуют те же проверки на корректность данных, что и при создании записи. Сообщение об успешном изменении данных заявки показано на рисунке 19.

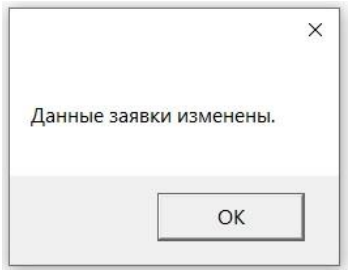


Рисунок 19 – Запись о заявке изменена

На рисунке 20 показана форма редактирования выбранной заявки.

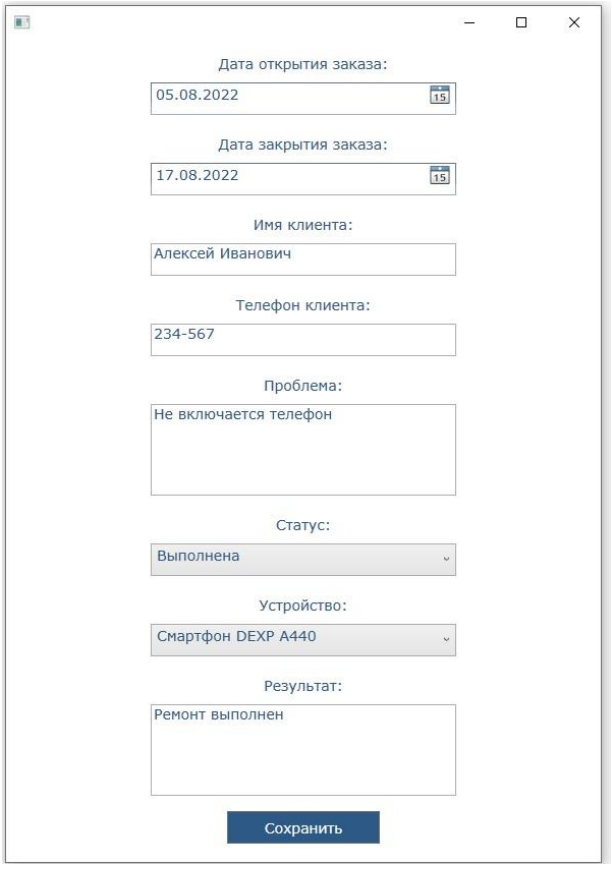


Рисунок 20 – Редактирование заявки

Пункт меню «Удалить заявку» позволяет удалить выбранную в таблице запись. В случае успешного удаления, пользователю будет отображено следующее сообщение (рисунок 21):

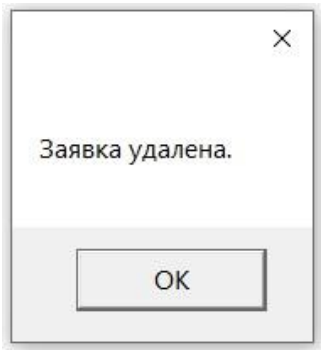


Рисунок 21 – Сообщение об удалении заявки

Для редактирования перечня услуг, входящих в выбранную заявку, используются пункты меню «Добавить услугу к заявке» и «Удалить услугу из заявки». Форма добавления услуги показана на рисунке 22.

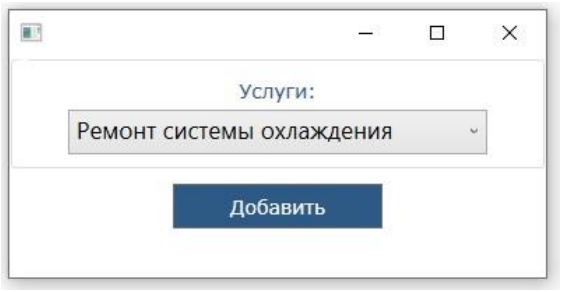


Рисунок 22 – Форма добавления новой услуги к заявке

После добавления услуги приложение отобразит сообщение об успешном выполнении операции (рисунок 23):

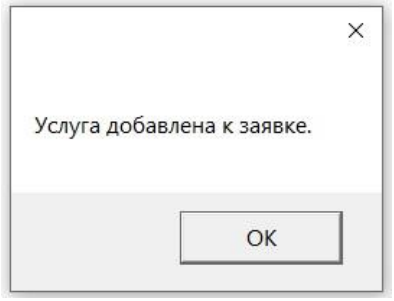


Рисунок 23 – Услуга добавлена к заявке

Для удаления услуги необходимо сначала выбрать заявку в верхней таблице, затем удаляемую услугу в нижней таблице и нажать на пункт меню «Удалить услугу из заявки». Сообщение о выполнении задачи показано на рисунке 24.

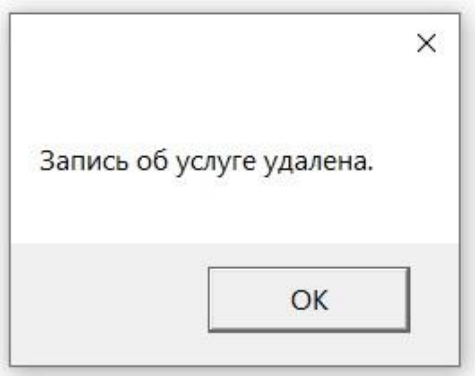
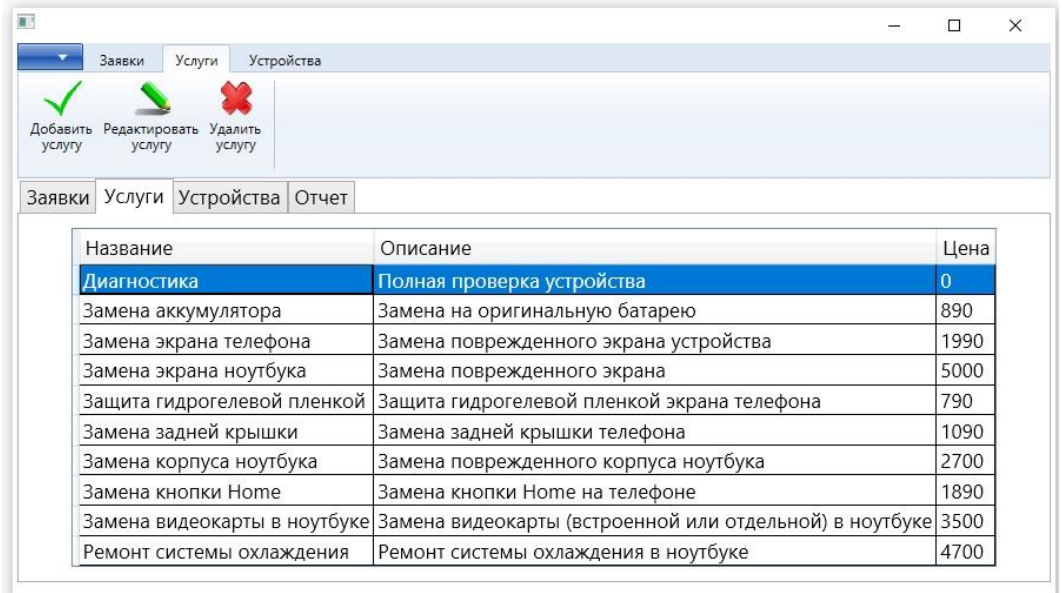


Рисунок 24 – Сообщение об удалении услуги из заявки

На рисунке 25 показана вкладка «Услуги».



Название	Описание	Цена
Диагностика	Полная проверка устройства	0
Замена аккумулятора	Замена на оригинальную батарею	890
Замена экрана телефона	Замена поврежденного экрана устройства	1990
Замена экрана ноутбука	Замена поврежденного экрана	5000
Защита гидрогелевой пленкой	Защита гидрогелевой пленкой экрана телефона	790
Замена задней крышки	Замена задней крышки телефона	1090
Замена корпуса ноутбука	Замена поврежденного корпуса ноутбука	2700
Замена кнопки Home	Замена кнопки Home на телефоне	1890
Замена видеокарты в ноутбуке	Замена видеокарты (встроенной или отдельной) в ноутбуке	3500
Ремонт системы охлаждения	Ремонт системы охлаждения в ноутбуке	4700

Рисунок 25 – Вкладка «Услуги»

Соответствующий раздел меню позволяет добавлять услуги, редактировать и удалять. Форма для добавления новой услуги показана на рисунке 26.

Рисунок 26 – Добавление новой услуги

Поля название, описание и цена являются обязательными для заполнения. Для цены выполняется дополнительная проверка: цена должна быть положительным числом. Сообщения об ошибках, отображаемые при некорректном заполнении полей, показаны на рисунках 27 и 28.

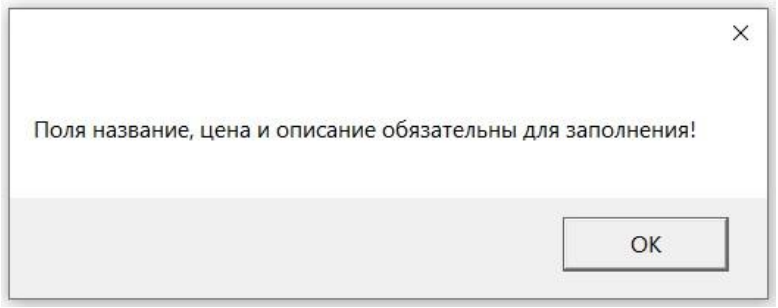


Рисунок 27 – Ошибка при отсутствии данных в обязательных полях

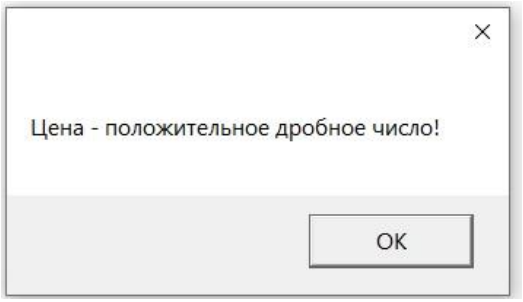


Рисунок 28 – Указана некорректная цена

Сообщение об успешном создании услуги показано на рисунке 29.

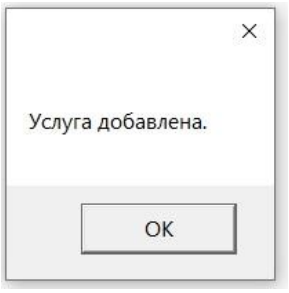


Рисунок 29 – Сообщение об успешном добавлении услуги

При редактировании услуги необходимо выбрать запись для изменения. Если запись не выбрана, то приложение отобразит следующее сообщение об ошибке (рисунок 30):

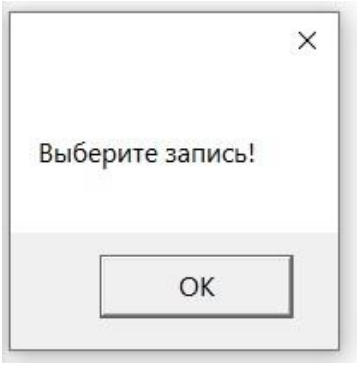


Рисунок 30 – Ошибка редактирования услуги

Форма редактирования записи показана на рисунке 31.

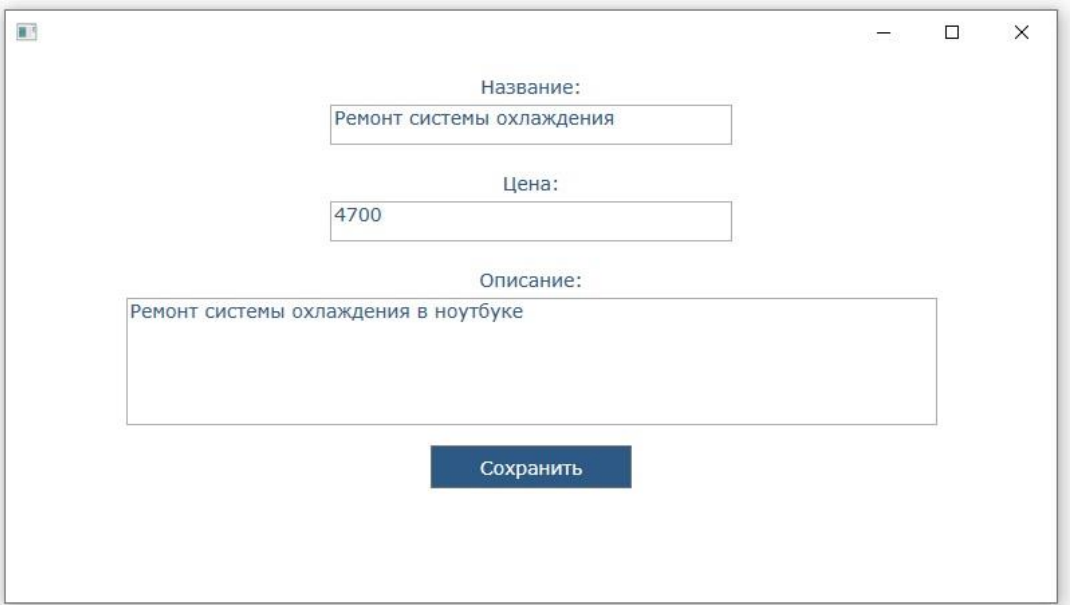


Рисунок 31 – Редактирование записи об услуге

Сообщение об изменении записи показано на рисунке 32.

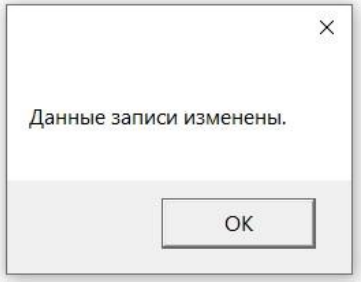


Рисунок 32– Сообщение об успешном изменении записи об услуге

Сообщение об удалении услуги показано на рисунке 33. Для удаления записи об услуге необходимо выбрать строку в таблице и нажать на пункт меню «Удалить услугу».

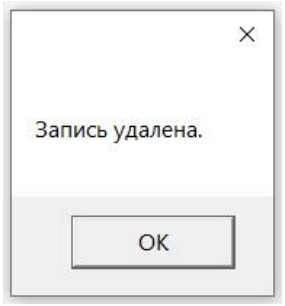


Рисунок 33 – Удаление услуги

Вкладка «Устройства» показана на рисунке 34.

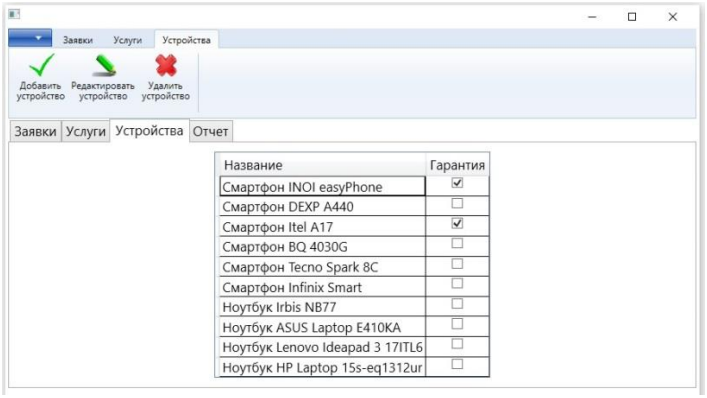


Рисунок 34 – Вкладка «Устройства»

Пункт меню «Добавить устройство» открывает форму для добавления новой записи об устройстве (рисунок 35).

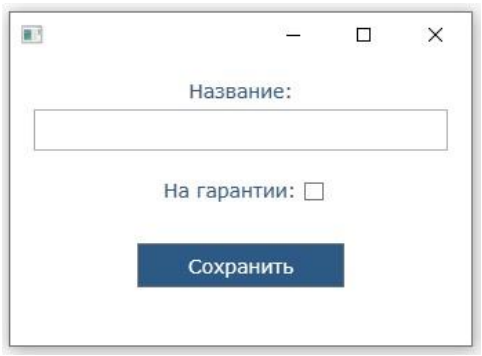


Рисунок 35 – Форма добавления нового устройства

Для заполнения формы необходимо ввести название устройства и указать, находится ли оно на гарантии. Если название не заполнено, то система выдает следующее предупреждение (рисунок 36):

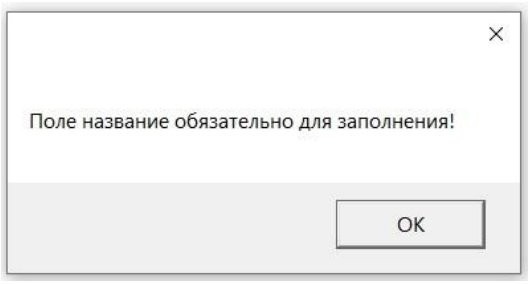


Рисунок 36 – Ошибка при добавлении записи

Сообщение об успешном добавлении записи показано на рисунке 37.

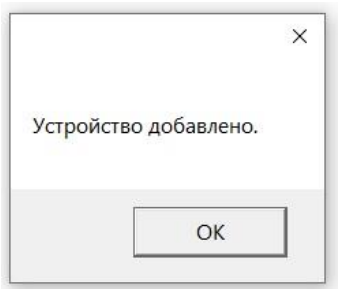


Рисунок 37 – Сообщение о добавлении нового устройства

Форма редактирования записи об устройстве показана на рисунке 38.

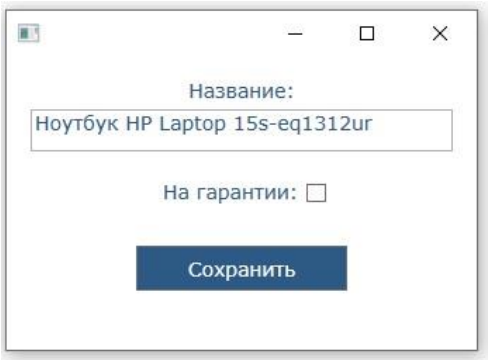


Рисунок 38 – Форма редактирования записи об устройстве

Сообщения об изменении и удалении записи об устройстве показаны на рисунках 39 и 40 соответственно.

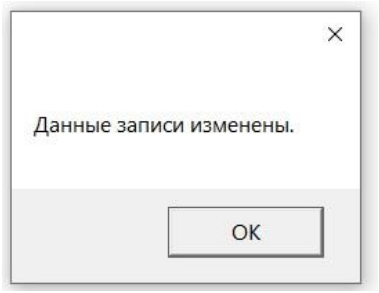


Рисунок 39 – Редактирование устройства

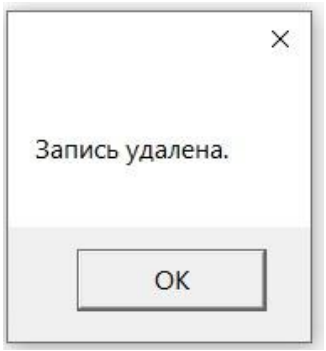


Рисунок 40 – Удаление устройства

На рисунке 41 показана вкладка «Отчет».

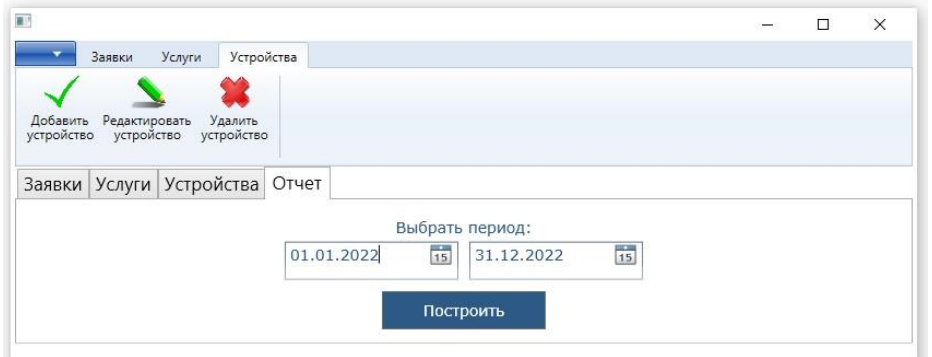


Рисунок 41 – Вкладка «Отчет»

Для построения отчета необходимо выбрать временной период и нажать кнопку «Построить». Сообщение об успешном построении отчета показано на рисунке 42.

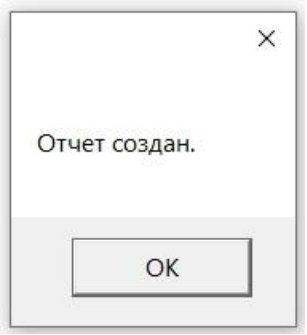


Рисунок 42 – Отчет построен

Для просмотра отчета используется MS Excel. Пример отчета показан на рисунке 43.

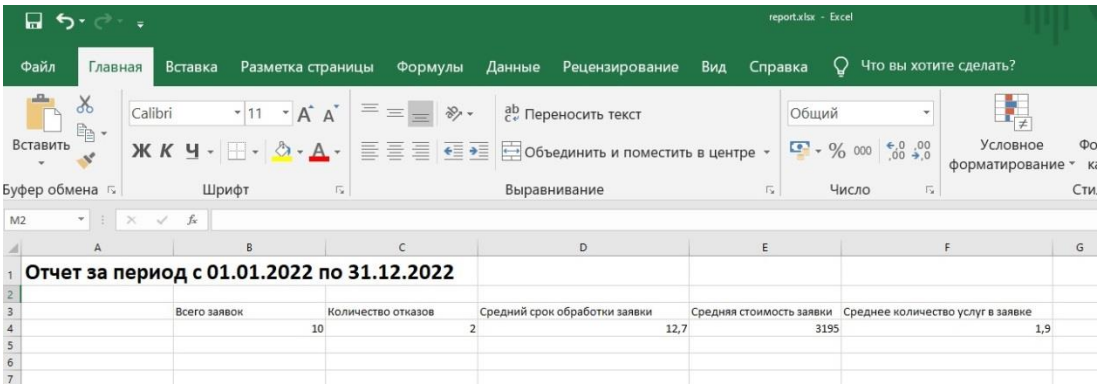


Рисунок 43 – Пример отчета

Если за выбранный период данные отсутствуют, то система отобразит следующее предупреждение (рисунок 44):

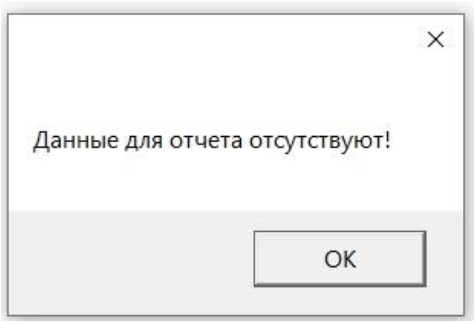


Рисунок 44 – Сообщение об отсутствии данных для отчета

Данное сообщение также отображается при выборе некорректного периода.

3.7 Оценка и обоснование экономической эффективности разработки информационной системы сопровождения заказов

3.7.1 Выбор методики расчета экономической эффективности

Оценка экономической эффективности позволяет оценить, насколько эффективна реализация автоматизируемого бизнес-процесса с использованием внедренной технологии. Как правило, использование программных продуктов, при решении пользовательских задач, позволяет сократить время, затрачиваемое на рутинные операции.

Данная оценка выполнялась на основе расчета прямой эффективности от внедрения программного продукта. В рамках указанной методики выполняется сравнение базового и предлагаемого способа обработки информации [24].

Базовый вариант – обработка данных с применением инструментов пакета MS Office. Предлагаемым вариантом является применение разработанного приложения для сервисного центра.

За основу для проводимых расчетов взяты временные затраты рабочего времени и его стоимость. Стоимость одного часа рабочего времени равна 291 р. из расчета заработной платы с учетом отчислений в 46550 р.

Расчеты проводятся для следующих процессов и задач:

– формирование заявки;

– редактирование заявки;

– построение отчета;

– формирование результата по заявке.

Для каждого из указанных выше объектов проведены расчеты затрат по времени. Затраты вычислены как для выполнения задач в речном режиме, без применения информационной системы, так и для решения задач с применением разработанной информационной системы.

Формулы и полученные результаты приведены в таблицах ниже.

3.7.2 Расчет показателей экономической эффективности проекта

В таблице 11 показаны расчеты показателей эффективности для процесса «Формирование заявки».

Таблица 11 – Процесс «Формирование заявки»

Трудоемкость	Затраты		Абсолютное изменение затрат	Коэффициент изменения затрат	Индекс изменения затрат
	Базовый вариант	Проектный вариант			
	T_0 (мин)	T_1 (мин)	$\Delta T = T_0 - T_1$	$К_T = \frac{\Delta T}{T_0} \cdot 100\%$	$Y_T = \frac{T_0}{T_1}$
	7	5	2	28,6	1,4
Стоимость	C_0 (руб)	C_1 (руб)	$\Delta C = C_0 - C_1$ (руб)	$К_C = \frac{\Delta C}{C_0} \cdot 100\%$	$Y_C = \frac{C_0}{C_1}$
	4,2	3,0	1,2	28,6	1,4

В таблице 12 показаны расчеты для процесса редактирования заявки.

Таблица 12 – Процесс «Редактирование заявки»

	Затраты		Абсолютное изменение затрат	Коэффициент изменения затрат	Индекс изменения затрат
	Базовый вариант	Проектный вариант			
	T_0 (мин)	T_1 (мин)	$\Delta T = T_0 - T_1$	$К_T = \frac{\Delta T}{T_0} \cdot 100\%$	$Y_T = \frac{T_0}{T_1}$
Трудоемкость	15	5	10	66,7	3
Стоимость	C_0 (руб)	C_1 (руб)	$\Delta C = C_0 - C_1$ (руб)	$К_C = \frac{\Delta C}{C_0} \cdot 100\%$	$Y_C = \frac{C_0}{C_1}$
	9,1	3	6,1	66,7	3

В таблице 13 показаны расчеты для построения отчета по заявкам.

Таблица 13 – Процесс «Построение отчета»

Трудоемкость	Затраты		Абсолютное изменение затрат	Коэффициент изменения затрат	Индекс изменения затрат
	Базовый вариант	Проектный вариант			
	T_0 (мин)	T_1 (мин)	$\Delta T = T_0 - T_1$	$К_T = \frac{\Delta T}{T_0} \cdot 100\%$	$Y_T = \frac{T_0}{T_1}$
	120	4	116	96,7	30
Стоимость	C_0 (руб)	C_1 (руб)	$\Delta C = C_0 - C_1$ (руб)	$К_C = \frac{\Delta C}{C_0} \cdot 100\%$	$Y_C = \frac{C_0}{C_1}$
	72,8	2,4	70,3	96,7	30,0

В таблице 14 приведены расчеты для формирования результата по заявке.

Таблица 14 –Формирование результата по заявке

Трудоемкость	Затраты		Абсолютное изменение затрат	Коэффициент изменения затрат	Индекс изменения затрат
	Базовый вариант	Проектный вариант			
	T_0 (мин)	T_1 (мин)	$\Delta T = T_0 - T_1$	$К_T = \frac{\Delta T}{T_0} \cdot 100\%$	$Y_T = \frac{T_0}{T_1}$
	20	3	17	85	6,7
Стоимость	C_0 (руб)	C_1 (руб)	$\Delta C = C_0 - C_1$ (руб)	$К_C = \frac{\Delta C}{C_0} \cdot 100\%$	$Y_C = \frac{C_0}{C_1}$
	12,1	1,8	10,3	85,0	6,7

Внедрение информационной системы для сервисного центра позволяет сократить время, затрачиваемое на обработку заявок.

По результатам проекта произошла автоматизация учета заявок, благодаря которой сотрудники экономят свое время на составление, редактирование и закрытие сделок. Чтобы понять экономическую эффективность (Э) по экономии время, необходимо использовать следующую формулу:

$$Э = \left(\frac{T1 \times N - T2 \times N}{T1 \times N}\right) \times 100\,\%, \quad (1)$$

где Э – экономическая эффективность;

N - деловые операции работы с заявками;

T – единица времени.

Например, 50 операций по учету заявок выполнялось до внедрения информационной системы за 120 минут, а после внедрения информационной системы за 25 минут. Экономическая эффективность по времени учета заявок составляет 79%. То есть время стало использоваться эффективнее.

Что касается сроков окупаемости проекта, они могут быть рассчитаны по следующей формуле:

$$PP = \frac{I}{PR}\ , \qquad (2)$$

где PR – чистый инвестиционный доход;

PP – период, в течение которого достигается нулевая точка;

I – совокупная сумма вложенного капитала.

Благодаря данной формуле и имеющимся данным (затратам на проект, полученной прибыли после внедрения проекта) можно отметить, что проект новой информационной системы окупится через 1 год и 2-3 месяца.

Выводы по главе 3

Приведены результаты проектирования и разработки информационной системы для ведения учета заявок сервисного центра Ref:Port. Представлено описание функциональных возможностей программы, приведены схемы, отображающие структуру приложения.

Проведен расчет экономической эффективности от внедрения разработанного приложения.

Заключение

Процесс учета сопровождения заказов в автоматизированном виде – это важная и актуальная задача. Ведь в современных условиях развития информационных технологий важно минимизировать затраты времени на учет сопровождения заказов. В процессе выполнения ВКР рассматривалось выполнение данной задачи.

Благодаря информации, которая была получена в процессе реализации задач, была спроектирована информационная система, которая позволит быстро и качественно вести учет сопровождения заказов. Также стоит указать, что были выполнены такие задачи, как анализ предметной области, возможных методов проектирования решения поставленных задач, существующих программных средств. Исходя из расчета эффективности, стоит учесть, что данный проект оказался рентабельным и полностью оправдал ожидания сотрудников Ref:Port.

При разработке использовались:

- объектно-ориентированный язык C#;
- среда разработки Visual Studio 2022 Community;
- фреимворк для работы с базами данных Entityframework;
- СУБД MS Sql Server.

Итак, результат выпускной квалификационной работы – спроектированная, а далее разработанная и внедренная информационная система, позволяющая вести учет сопровождения заказов в сервисном центре Ref:Port. Данная информационная система на сегодняшний день уже успешно используется в сервисном центре Ref:Port. В перспективе, можно предложить дальнейшую доработку информационной системы и разработку мобильного приложения, позволяющего вести учет сопровождения заказов в сервисном центре Ref:Port.

Список используемой литературы и используемых источников

1. Атабаева Э. Р., Моисеенко Н. А. Важные функции программного обеспечения CRM. Грозный : Universum, 2021. 41 с.
2. Верёвкин Д. М. Применение CASE-технологий для анализа бизнес-процессов при проектировании информационных систем. Павлодар : Наука, техника и образование, 2022. 51 с.
3. Грекул В. И., Денищенко Г. Н., Коровкина Н. Л. Проектирование информационных систем // Интернет-университет информационных технологий. 2018. 420 с.
4. Губко Ю. А. Применение CRM-системы управления взаимоотношения с клиентом в розничной торговле. М. : Скиф. Вопросы студенческой науки, 2020. 293 с.
5. Гумеров Р. Х. Цели и функции управления заказами. Самара : Инновационная наука, 2020. 50 с.
6. Гусятников В.Н., Безруков А. И. Стандартизация и разработка программных систем. М. : Финансы и статистика, 2020. 288 с.
7. Дубейковский В. И. Практика функционального моделирования с AllFusion Process Modeler. М. : ДИАЛОГ-МИФИ, 2021. 464 с.
8. Карзаева Н.Н. Учет товарных операций. М.: Финансы и статистика, 2018. 415 с.
9. Константинов П. К., Павлов И. С. Программы лояльности как неотъемлемая часть современного маркетинга. М. : Научно-образовательный журнал для студентов и преподавателей «StudNet», 2021. 25 с.
10. Лосева А. В. Автоматизированные системы обработки информации. Пенза : Вестник Пензенского государственного университета, 2021. 95 с.
11. Максумов Р. Э., Караева М. Р. Автоматизация складского пространства. Ростов-на-Дону : Инновационная наука, 2022. 26 с.
12. Мартыщенко Д. О. Современные технологии в маркетинге. Ростов-На-Дону : Молодой исследователь Дона, 2020. 113 с.

13. Медведев М.Ю. Учетная политика: бухгалтерская и налоговая. М. : ИД ФБК-Пресс, 2018. 46 с.

14. Радченко М. Г. 1С: Предприятие 8.0. Практическое пособие разработчика. Примеры и типовые приемы. М. : ООО «1С-Паблишинг», 2018. 656 с.

15. Рубичев Н. А. Измерительные информационные системы: Учебное пособие. М. : Дрофа, 2019. 334 с.

16. Уткин В. Б. Информационные системы в экономике: Учебник для студентов высших учебных заведений. М. : ИЦ Академия, 2018. 288 с.

17. Фаулер М. Архитектура корпоративных программных приложений. М. : Издательский дом «Вильямс», 2021. 544 с.

18. Федоренко И. В. Содержание бухгалтерского учета в современных условиях его нормативного регулирования // Аудит и финансовый анализ. 2018. No 6. С. 83–88.

19. Федорова Г. Н. Информационные системы: Учебник для студ. учреждений сред. проф. образования. М. : ИЦ Академия, 2021. 208 с.

20. Ходарев А. С. Автоматизация управленческого учета. М. : ЗАО "Финстатинформ", 2020. 533 с.

21. Шаханова Л. Ш. Учет и анализ товаров в розничной торговле. Самара : Архивариус, 2021. 43 с.

22. Якобсон А. Унифицированный процесс разработки программного обеспечения СПб. : Питер, 2019. 496 с.

23. Ясенев В. Н. Информационные системы и технологии в экономике.: Учебное пособие для студентов вузов. М. : ЮНИТИ-ДАНА, 2018. 560 с.

24. Antonets V. A., Bedny B. I., Abubakirova K. N., Nechaeva N. V., Runova E. V., Surkova A. S. System of distance courses on technological entrepreneurship // Bulletin of the UNN State University. 2019. No 5-2.

25. Likhobanov M. Yu. Technologies of manipulation in advertising (methods of zombification). M. : RGB, 2019. 144 с.

26. Rozhkov K. L. The process of globalization and the National Economy. M. : Abstract, 2020. 17 c.

27. Wil van der Aalst, Process Mining Data Science in Action Second Edition. M. : Abstract, 2016. 165 c.

28. Wil van der Aalst, Process Mining Discovery, Conformance and Enhancement of Business Processes. M. : Abstract, 2014. 25 c.

Приложение А

Модели

```
using System.ComponentModel.DataAnnotations;

namespace ServiceCenter.Models
{
   public class Employee // сотрудник
   {
      public int Id { get; set; }

      [Required]
      [MaxLength(80)]
      public string LastName { get; set; } //фамилия

      [Required]
      [MaxLength(80)]
      public string Name { get; set; } // имя

      [MaxLength(80)]
      public string  Patronymic { get; set; } // отчество

      [Required]
      [MaxLength(100)]
      public string Place { get; set; } // должность

      public override string ToString()
      {
         return LastName+" "+Name+" "+Patronymic;
      }
   }
}
using System.ComponentModel.DataAnnotations;

namespace ServiceCenter.Models
{
  public class Item // устройство
   {
      public int Id { get; set; }

      [Required]
      [MaxLength(80)]
      public string Title { get; set; }   // название
      public bool Guarantee { get; set; } // находится ли на гарантии
      public override string ToString()
      {
         return Title;
      }
```

```
}
}
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace ServiceCenter.Models
{
   public class Order // заявка
   {
      public int Id { get; set; }
      public DateTime OpenDate { get; set; }  // дата создания заявки
      public DateTime CloseDate { get; set; } // дата закрытия заявки

      [MaxLength(300)]
      public string ClientName { get; set; } // клиент

      [MaxLength(20)]
      public string ClientPhone { get; set; } // клиент

      public Employee Manager { get; set; } // менеджер, принявший заявку

      [MaxLength(3000)]
      public string Comment { get; set; } // проблема

      [MaxLength(80)]
      public string Status { get; set; } // статус

      [MaxLength(3000)]
      public string Result { get; set; } // результат ремонта
      public virtual ICollection<Service> Services { get; set; } // услуги заявки
      public Item Item { get; set; }
   }
}
using System.Data.Entity;
namespace ServiceCenter.Models
{
  public class SContext : DbContext // представление для БД
   {
      private static SContext sContext;
      /// <summary>
      /// таблицы
      /// </summary>
      public DbSet<Item> Items { get; set; }
      public DbSet<Order> Orders { get; set; }
      public DbSet<Service> Services { get; set; }
      public DbSet<User> Users { get; set; }
      public DbSet<Employee> Employees { get; set; }
      public SContext()
         :base("name=SConnection")
```

```
{}
        public static SContext GetSContext()
        {
            if(sContext == null)
            {
                sContext = new SContext();
            }
            return sContext;
        }
    }
}
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace ServiceCenter.Models
{
  public class Service // услуга
    {
        public int Id { get; set; }

        [Required]
        [MaxLength(255)]
        public string Title { get; set; } // название

        [MaxLength(3000)]
        public string Description { get; set; } // описание

        [Required]
        public double Price { get; set; } // цена

        public virtual ICollection<Order> Orders { get; set; } // заявки услуги
        public override string ToString()
        {
            return Title;
        }
    }
}
namespace ServiceCenter.Models
{
  public class Session // класс, хранящий данные авторизованного пользователя
    {
        public static Employee autorizedEmployee; // авторизованный пользователь
    }
}
using System.ComponentModel.DataAnnotations;

namespace ServiceCenter.Models
{
  public class User // пользователь
    {
```

```
public int Id { get; set; }

        [Required]
        [MaxLength(80)]
        public string Login { get; set; } // логин

        [Required]
        [MaxLength(80)]
        public string Pass { get; set; } // пароль

        [Required]
        [MaxLength(300)]
        public string Role { get; set; } // роль
        public Employee Employee { get; set; } // сотрудник - владелец учетной записи
    }
}
```

Приложение Б

Бизнес-логика

```
using ServiceCenter.Models;
using System.ComponentModel;

namespace ServiceCenter
{
   public abstract class BaseViewModel: INotifyPropertyChanged // родительская модель
представления
   {
      public event PropertyChangedEventHandler PropertyChanged;
      public SContext Context { get; set; } // БД
      public SController Controller { get; set; } // контроллер (методы для навигации)

      protected void NotifyPropertyChanged(string propertyName) // отслеживание изменения
свойств
      {
         PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
      }
   }
}
using ServiceCenter.Models;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;

namespace ServiceCenter
{
   public class StartViewModel: BaseViewModel // авторизация
   {
      private static StartViewModel startViewModel;

      private string login;
      public string Login // логин
      {
         get { return login; }
         set { login = value; NotifyPropertyChanged("Login"); }
      }

      public ICommand AutorizeCommand { get; set; } // команда обработки авторизации


      private StartViewModel() // конструктор
      {
         AutorizeCommand = new SCommand(Autorization);
         Context = SContext.GetSContext();
```

```
}

        public static StartViewModel GetStartViewModel() // метод для контроля наличия только 1 экземпляра класса
        {
            if (startViewModel == null)
            {
                startViewModel = new StartViewModel();
            }
            return startViewModel;
        }

        private void Autorization(object parameter)  // метод для авторизации
        {
            var passwordBox = parameter as PasswordBox;
            var password = passwordBox.Password;
            var user = Context.Users.Include("Employee").Where(u => u.Login.Equals(Login) & u.Pass.Equals(password));

            if (user.Count() > 0)
            {
                Session.autorizedEmployee = user.FirstOrDefault().Employee;

                Controller = SController.GetSController();
                Controller.OpenForm("managerw");
}
            else
            {
                MessageBox.Show("Учетная запись не существует");
            }
        }
    }
}
using ServiceCenter.Models;
using System;
using System.Collections.ObjectModel;
using System.Linq;
using System.Windows;
using System.Windows.Input;
using Excel = Microsoft.Office.Interop.Excel;

namespace ServiceCenter.ViewModels
{
    public class ManagerViewModel : BaseViewModel
    {
        private static ManagerViewModel managerViewModel;

        /// <summary>
        /// команды для связи интерфейса с методами
```

```
/// </summary>
    public ICommand ShowOrderInfoCommand { get; set; }
    public ICommand OpenFormCommand { get; set; }
    public ICommand SaveOrderCommand { get; set; }
    public ICommand DelOrderCommand { get; set; }
    public ICommand AddServiceToOrderCommand { get; set; }
    public ICommand DelServiceFromOrderCommand { get; set; }
    public ICommand SaveServiceCommand { get; set; }
    public ICommand DelServiceCommand { get; set; }
    public ICommand SaveItemCommand { get; set; }
    public ICommand DelItemCommand { get; set; }
    public ICommand SaveReportCommand { get; set; }

    //  свойства привязки данных к формам

    private bool iGuarantee;
    public bool IGuarantee
    {
       get { return iGuarantee; }
       set { iGuarantee = value; NotifyPropertyChanged("IGuarantee"); }
    }

    private string iTitle;
    public string ITitle
    {
       get { return iTitle; }
       set { iTitle = value; NotifyPropertyChanged("ITitle"); }
    }

    private string sTitle;
    public string STitle
    {
       get { return sTitle; }
       set { sTitle = value; NotifyPropertyChanged("STitle"); }
    }

    private string sPrice;
    public string SPrice
    {
       get { return sPrice; }
       set { sPrice = value; NotifyPropertyChanged("SPrice"); }

    }

    private string sDesc;
    public string SDesc
    {
       get { return sDesc; }
       set { sDesc = value; NotifyPropertyChanged("SDesc"); }
```

```
}

    private string result;
    public string Result
    {
       get { return result; }
       set { result = value; NotifyPropertyChanged("Result"); }
    }
    private string status;
    public string Status
    {
       get { return status; }
       set { status = value; NotifyPropertyChanged("Status"); }
    }
    private string comment;
    public string Comment
    {
       get { return comment; }
       set { comment = value; NotifyPropertyChanged("Comment"); }
    }
    private string clientName;
    public string ClientName
    {
       get { return clientName; }
       set { clientName = value; NotifyPropertyChanged("ClientName"); }
    }
    private string clientPhone;
    public string ClientPhone

    {
       get { return clientPhone; }
       set { clientPhone = value; NotifyPropertyChanged("ClientPhone"); }
    }
    private DateTime startDate;
    public DateTime StartDate
    {
       get { return startDate; }
       set { startDate = value; NotifyPropertyChanged("StartDate"); }
    }
    private DateTime endDate;
    public DateTime EndDate
    {
       get { return endDate; }
       set { endDate = value; NotifyPropertyChanged("EndDate"); }
    }
    private DateTime orderDT;
    public DateTime OrderDT
    {
       get { return orderDT; }
       set { orderDT = value; NotifyPropertyChanged("OrderDT"); }
```

```
}
    private DateTime closeDT;
    public DateTime CloseDT
    {
       get { return closeDT; }
       set { closeDT = value; NotifyPropertyChanged("CloseDT"); }
    }
    private string totalOrder;
    public string TotalOrder
    {

       get { return totalOrder; }
       set { totalOrder = value; NotifyPropertyChanged("TotalOrder"); }
    }
    private Service selectedOService;
    public Service SelectedOService
    {
       get { return selectedOService; }
       set { selectedOService = value; NotifyPropertyChanged("SelectedOService"); }
    }
    private Service selectedService;
    public Service SelectedService
    {
       get { return selectedService; }
       set { selectedService = value; NotifyPropertyChanged("SelectedService"); }
    }
    private Order selectedOrder;
    public Order SelectedOrder
    {
       get { return selectedOrder; }
       set { selectedOrder = value; NotifyPropertyChanged("SelectedOrder"); }
    }
    private Item selectedItem;
    public Item SelectedItem
    {
       get { return selectedItem; }
       set { selectedItem = value; NotifyPropertyChanged("SelectedItem"); }
    }
    private string selectedOrderData;
    public string SelectedOrderData
    {
       get { return selectedOrderData; }
       set { selectedOrderData = value; NotifyPropertyChanged("SelectedOrderData"); }
    }
    /// <summary>
    /// коллекции данных
    /// </summary>

    private ObservableCollection<Order> ordersList;
```

```
public ObservableCollection<Order> OrdersList
    {
       get { return ordersList; }
       set { ordersList = value; NotifyPropertyChanged("OrdersList"); }
    }
    private ObservableCollection<Item> itemsList;
    public ObservableCollection<Item> ItemsList
    {
       get { return itemsList; }
       set { itemsList = value; NotifyPropertyChanged("ItemsList"); }
    }
    private ObservableCollection<Service> servicesList;
    public ObservableCollection<Service> ServicesList
    {
       get { return servicesList; }
       set { servicesList = value; NotifyPropertyChanged("ServicesList"); }
    }
    private ObservableCollection<Service> oServicesList;
    public ObservableCollection<Service> OServicesList
    {
       get { return oServicesList; }
       set { oServicesList = value; NotifyPropertyChanged("OServicesList"); }
    }
    private ManagerViewModel() // конструктор класса
    {
       Context = SContext.GetSContext();
       Controller = SController.GetSController();
       OrdersList = new ObservableCollection<Order>();
       OServicesList = new ObservableCollection<Service>();

       ItemsList = new ObservableCollection<Item>();
       ServicesList = new ObservableCollection<Service>();
       ShowOrderInfoCommand = new SCommand(ShowOrderInfo);
       OpenFormCommand = new SCommand(OpenForm);
       SaveOrderCommand = new SCommand(SaveOrder);
       DelOrderCommand = new SCommand(DelOrder);
       AddServiceToOrderCommand = new SCommand(AddServiceToOrder);
       DelServiceFromOrderCommand = new SCommand(DelServiceFromOrder);
       SaveServiceCommand = new SCommand(SaveService);
       DelServiceCommand = new SCommand(DelService);
       SaveItemCommand = new SCommand(SaveItem);
       DelItemCommand = new SCommand(DelItem);
       SaveReportCommand = new SCommand(SaveReport);
       SetOrdersList();
       SetItemsList();
       SetServicesList();
       CloseDT = DateTime.Now.Date;
       OrderDT = DateTime.Now.Date;
       StartDate = new DateTime(DateTime.Now.Year, 1, 1);
```

```
 EndDate = new DateTime(DateTime.Now.Year, 12, 31);
        }
        private void SaveReport(object obj) // построение отчета в экселе
        {
            var excelApp = new Excel.Application();
            excelApp.Workbooks.Add();
            Excel._Worksheet workSheet = (Excel.Worksheet)excelApp.ActiveSheet;
            int row = 1;
            workSheet.Cells[row, "A"] = "Отчет за период с "+StartDate.ToShortDateString()+" по
"+EndDate.ToShortDateString();
            workSheet.get_Range("A1", "A1").Cells.Font.Size = 20;
            workSheet.get_Range("A1", "A1").Cells.Font.Bold = true;
            row++;
            row++;
            workSheet.Cells[row, "B"] = "Всего заявок";
            workSheet.Cells[row, "C"] = "Количество отказов";
            workSheet.Cells[row, "D"] = "Средний срок обработки заявки";
            workSheet.Cells[row, "E"] = "Средняя стоимость заявки";
            workSheet.Cells[row, "F"] = "Среднее количество услуг в заявке";

            row++;

            var rows = Context.Orders.Include("Item").Where(r => r.OpenDate >= StartDate &
r.CloseDate <= EndDate).ToList();
            if(rows.Count == 0)
            {
                MessageBox.Show("Данные для отчета отсутствуют!");
                return;
            }
            int cancels = 0;
            double workTime = 0, cost = 0, s_amount = 0;
            foreach (var item in rows)
            {
                if(item.Status.ToLower() == "отменена")
                {
                    cancels++;
                }
                workTime += (item.CloseDate.Date - item.OpenDate.Date).TotalDays;
                if (item.Services.Count > 0)
                {
                    cost += item.Services.Sum(i => i.Price);
                    s_amount += item.Services.Count;
                }
            }
            workSheet.Cells[row, "B"] = rows.Count;
            workSheet.Cells[row, "C"] = cancels;
            workSheet.Cells[row, "D"] = Math.Round(workTime/ rows.Count, 2);
            workSheet.Cells[row, "E"] = Math.Round(cost /rows.Count, 2);
            workSheet.Cells[row, "F"] = Math.Round(s_amount / rows.Count, 2);
```

```
 workSheet.get_Range("B1", "F1").Cells.Font.Bold = true;

          workSheet.get_Range("B1", "F1").Cells.Font.Size = 16;
          workSheet.get_Range("B1", "G1").Cells.HorizontalAlignment =
Microsoft.Office.Interop.Excel.XlHAlign.xlHAlignCenter;
          workSheet.Columns[1].ColumnWidth = 25;
          workSheet.Columns[2].ColumnWidth = 25;
          workSheet.Columns[3].ColumnWidth = 25;
          workSheet.Columns[4].ColumnWidth = 35;
          workSheet.Columns[5].ColumnWidth = 25;
          workSheet.Columns[6].ColumnWidth = 35;
          string path = System.AppDomain.CurrentDomain.BaseDirectory + "report.xlsx";
          //MessageBox.Show(path);
          workSheet.SaveAs(path);
          excelApp.Quit();
          MessageBox.Show("Отчет создан.");
       }
       private void DelService(object obj) // удалить услугу
       {
          if (SelectedService == null)
          {
             MessageBox.Show("Выберите запись!");
             return;
           }
          var delServ = Context.Services.Find(SelectedService.Id);
          if (delServ != null)
          {
             Context.Services.Remove(delServ);
             Context.SaveChanges();
             SetServicesList();
             MessageBox.Show("Запись удалена.");
          }
       }
       private void DelItem(object obj) // удалить устройство
       {
          if (SelectedItem == null)
          {
             MessageBox.Show("Выберите запись!");
             return;
          }
          var delI = Context.Items.Find(SelectedItem.Id);
          if (delI != null)
          {
             Context.Items.Remove(delI);
             Context.SaveChanges();
             SetItemsList();
             MessageBox.Show("Запись удалена.");
          }
       }
```

```
 private void SaveService(object obj) // добавление услуги
      {
         if (string.IsNullOrEmpty(STitle) || string.IsNullOrEmpty(SPrice) ||
string.IsNullOrEmpty(SDesc))
         {
            MessageBox.Show("Поля название, цена и описание обязательны для заполнения!");
            return;
         }
         double tempPrice;
         if (!double.TryParse(SPrice, out tempPrice) || tempPrice < 0)
         {
            MessageBox.Show("Цена - положительное дробное число!");
            return;
         }
         if (SelectedService == null)
         {
            Service newService = new Service
            {
               Title = STitle,
               Price = tempPrice,
               Description = SDesc
            };

            Context.Services.Add(newService);
            Context.SaveChanges();
            MessageBox.Show("Услуга добавлена.");
         }
         if (SelectedService != null)
         {
            var serv = Context.Services.Find(SelectedService.Id);
            if (serv != null)
            {
               serv.Title = STitle;
               serv.Price = tempPrice;
               serv.Description = SDesc;
               Context.SaveChanges();
               MessageBox.Show("Данные записи изменены.");
            }
         }
         ClearFields();
         SetServicesList();
         Controller.CloseWin("adds");
      }
      private void SaveItem(object obj) // добавление устройства
      {
         if (string.IsNullOrEmpty(ITitle) || string.IsNullOrWhiteSpace(ITitle))
         {
            MessageBox.Show("Поле название обязательно для заполнения!");
            return;
```

```
}
        if (SelectedItem == null)
        {
            Item newItem = new Item
            {
                Title = ITitle,
                Guarantee = IGuarantee
            };
            Context.Items.Add(newItem);
            Context.SaveChanges();
            MessageBox.Show("Устройство добавлено.");
        }
        if (SelectedItem != null)
        {
            var item = Context.Items.Find(SelectedItem.Id);
            if (item != null)
            {
                item.Title = ITitle;
                item.Guarantee = IGuarantee;
                Context.SaveChanges();
                MessageBox.Show("Данные записи изменены.");
            }
        }
        ClearFields();
        SetItemsList();
        Controller.CloseWin("itemw");
    }
    private void DelServiceFromOrder(object obj) // удалить услугу из заявки
    {
        if (SelectedOrder == null)
        {
            MessageBox.Show("Выберите заявку!");
            return;
        }
        if (SelectedOService == null)
        {
            MessageBox.Show("Выберите услугу!");
            return;
        }
        var delItem = Context.Orders.Find(SelectedOrder.Id);
        var delServ = Context.Services.Find(SelectedOService.Id);

        if (delItem != null)
        {
            delItem.Services.Remove(delServ);
            Context.SaveChanges();
            ShowOrderInfo(null);
            MessageBox.Show("Запись об услуге удалена.");
        }
```

```
}
    private void AddServiceToOrder(object obj) // добавить услугу к заявке
    {
        if (SelectedService == null)
        {
            MessageBox.Show("Выберите услугу!");
            return;
        }
        var order = Context.Orders.Find(SelectedOrder.Id);
        var service = Context.Services.Find(SelectedService.Id);
            order.Services.Add(service);
            Context.SaveChanges();
            MessageBox.Show("Услуга добавлена к заявке.");
            Controller.CloseWin("ordserv");
    }
    private void DelOrder(object obj) // удалить заявку
    {
        if (SelectedOrder == null)
        {
            MessageBox.Show("Выберите заявку!");
            return;
        }
        var delOrder = Context.Orders.Find(selectedOrder.Id);
        if (delOrder != null)
        {
            Context.Orders.Remove(delOrder);
            Context.SaveChanges();
            SetOrdersList();
            MessageBox.Show("Заявка удалена.");
        }
    }
    public static ManagerViewModel GetManagerViewModel() // метод для контроля наличия
только 1 экземпляра класса
    {
        if (managerViewModel == null)
        {
            managerViewModel = new ManagerViewModel();
        }
        return managerViewModel;
    }
    private void SaveOrder(object obj) // добавить заявку
    {
        if (string.IsNullOrEmpty(ClientName) || string.IsNullOrEmpty(ClientName) ||
string.IsNullOrEmpty(ClientPhone) || string.IsNullOrEmpty(ClientPhone) ||
string.IsNullOrEmpty(Comment) || string.IsNullOrEmpty(Comment) || SelectedItem == null ||
Status == null || SelectedItem == null)
        {
            MessageBox.Show("Все поля, кроме результата, обязательны для заполнения!");
            return;
```

```
}
        if (CloseDT < OrderDT) CloseDT = OrderDT;
        if (SelectedOrder == null)
        {
            Order newOrder = new Order
            {
                ClientName = ClientName,
                ClientPhone = ClientPhone,
                Comment = Comment,
                CloseDate = CloseDT,
                OpenDate = OrderDT,
                Item = SelectedItem,
                Status = Status,

                Manager = Session.autorizedEmployee
            };
            Context.Orders.Add(newOrder);
            Context.SaveChanges();
            MessageBox.Show("Заявка добавлена.");
        }
        if (SelectedOrder != null)
        {
            var order = Context.Orders.Find(SelectedOrder.Id);
            if (order != null)
            {
                order.ClientName = ClientName;
                order.ClientPhone = ClientPhone;
                order.Comment = Comment;
                order.CloseDate = CloseDT;
                order.OpenDate = OrderDT;
                order.Item = SelectedItem;
                order.Status = Status;
                order.Result = Result;
            }
            Context.SaveChanges();
            MessageBox.Show("Данные заявки изменены.");
        }
        ClearFields();
        SetOrdersList();
        Controller.CloseWin("orderw");
    }
    private void SetOrdersList() // получить список заявок
    {
        OrdersList.Clear();
        var orders = Context.Orders.Include("Item").ToList();
        if (orders.Count() == 0)
        {
            MessageBox.Show("Записи о заявках отсутствуют!");
            return;
```

```
}
        foreach (var item in orders)
        {
            OrdersList.Add(item);
        }
    }
    private void SetItemsList() // получить список устройств
    {
        ItemsList.Clear();
        var items = Context.Items;
        if (items.Count() == 0)
        {
            MessageBox.Show("Записи об устройствах отсутствуют!");
            return;
        }
        foreach (var item in items)
        {
            ItemsList.Add(item);
        }
    }
    private void SetServicesList() // получить список услуг
    {
        ServicesList.Clear();
        var services = Context.Services;
        if (services.Count() == 0)
        {
            MessageBox.Show("Записи об услугах отсутствуют!");
            return;
        }
        foreach (var item in services)
        {
            ServicesList.Add(item);

        }
    }
    private void ShowOrderInfo(object obj) // показать информацию о заявке
    {
        TotalOrder = null;
        OServicesList.Clear();
        var order =
Context.Orders.Include("Item").Include("Services").Include("Manager").Where(o => o.Id ==
SelectedOrder.Id).FirstOrDefault();
        if (order != null)
        {
            SelectedOrderData = "Клиент: " + order.ClientName + ", " + order.ClientPhone +
"\nМенеджер: " + order.Manager.ToString() + "\nПроблема: " + order.Comment + "\nРезультат:
" + order.Result;
        }
        double tempTotal = 0;
```

```
  if (order.Services.Count > 0)
        {
           foreach (var o in order.Services)
           {
              OServicesList.Add(o);
              tempTotal += o.Price;
           }
           TotalOrder = "Общая сумма: " + tempTotal.ToString();
        }
     }
     private void OpenForm(object obj) // открыть форму
     {
        if (obj.ToString() == "adds")
        {
           SelectedService = null;
           Controller.OpenForm("adds");
        }
        if (obj.ToString() == "edits")
        {
           if (SelectedService == null)
           {
              MessageBox.Show("Выберите запись!");
              return;
           }
           var serv = Context.Services.Find(SelectedService.Id);
           if (serv != null)
           {
              STitle = serv.Title;
              SPrice = serv.Price.ToString();
              SDesc = serv.Description;
           }
           Controller.OpenForm("adds");
        }
        if (obj.ToString() == "addo")
        {
           SelectedOrder = null;
           Controller.OpenForm("orderw");
        }
        if (obj.ToString() == "edito")
        {
           if (SelectedOrder == null)
           {
              MessageBox.Show("Выберите заявку!");
              return;
           }
           var order = Context.Orders.Include("Item").Include("Manager").Where(o => o.Id ==
SelectedOrder.Id).FirstOrDefault();
           if (order != null)
           {
```

```
OrderDT = order.OpenDate.Date;
            CloseDT = order.CloseDate.Date;
            ClientName = order.ClientName;
            ClientPhone = order.ClientPhone;
            Comment = order.Comment;

            Status = order.Status;
            SelectedItem = order.Item;
            Result = order.Result;
        }
        Controller.OpenForm("orderw");
    }
    if (obj.ToString() == "adddtoo")
    {
        if (SelectedOrder == null)
        {
            MessageBox.Show("Выберите заявку!");
            return;
        }
        Controller.OpenForm("ordserv");
    }
    if (obj.ToString() == "addi")
    {
        SelectedItem = null;
        Controller.OpenForm("itemw");
    }
    if (obj.ToString() == "editi")
    {
        if (SelectedItem == null)
        {
            MessageBox.Show("Выберите запись!");
            return;
        }
        var item = Context.Items.Find(SelectedItem.Id);
        if (item != null)
        {
            ITitle = item.Title;
            IGuarantee = item.Guarantee;
        }
        Controller.OpenForm("itemw");
    }
}
private void ClearFields() //  очистка полей
{
    OrderDT = DateTime.Now.Date;
    CloseDT = DateTime.Now.Date;
    ClientName = null;
    ClientPhone = null;
    Comment = null;
```

```
 SelectedItem = null;
            Status = null;
            SelectedService = null;
            SelectedOrderData = null;
            SelectedOrder = null;
            SelectedOService = null;
            TotalOrder = null;
            StartDate = new DateTime(DateTime.Now.Year, 1, 1);
            EndDate = new DateTime(DateTime.Now.Year, 12, 31);
            Result = null;
            SDesc = null;
            SPrice = null;
            STitle = null;
            ITitle = null;
            IGuarantee = false;
        }
    }
}
```