

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

<i>Факультет</i>	<i>Кафедра</i>
вычислительной техники	МОиПЭВМ
Направление подготовки	09.03.04 «Программная инженерия»
Направленность	Программная инженерия

БАКАЛАВРСКАЯ РАБОТА

на тему

**Серверное приложение для системы
менеджмента хранилища нефтепродуктов**

Студент	_____	<u>Лункова София Антоновна</u>
	(подпись, дата)	(ФИО полностью)
Руководитель	_____	<u>Шибанов С.В.</u>
	(подпись, дата)	(фамилия, инициалы)
Нормоконтролёр	_____	<u>Такташкин Д.В.</u>
	(подпись, дата)	(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой	_____	<u>Козлов А.Ю.</u>
	(подпись)	(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК	_____	<u>Попова Н.А.</u>
	(подпись)	(фамилия, инициалы)

Пенза, 2024

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

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ А.Ю. Козлов
«25» декабря 2023 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Лункова София Антонвна гр. 20ВП1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Серверное приложение для менеджмента хранилища нефтепродуктов

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20 _____

2. Руководитель работы к.т.н., доцент кафедры «МО и ПЭВМ» Шибанов С.В.

3. Задание на работу (назначение разработки, исходные данные и т.п.)

1) Назначение: Серверное приложение для менеджмента хранилища нефтепродуктов

2) Среда выполнения: Windows 10.

Языковые средства разработки: C#, Transact-SQL.

Инструментальные средства: Visual Studio 2019, SQL Server Management Studio 19, draw.io, Visual Paradigm.

Технологии разработки: ООП, сервисно-ориентированный подход.

3) Основные функции разрабатываемого программного обеспечения:

а) распределение нефтепродуктов по резервуарам;

б) добавление/удаление резервуаров;

в) отображение состояний емкостей и насосов;

г) закачка/откачка нефтепродуктов;

д) редактирование параметров резервуаров и насосов;

- е) ведение журнала действий;
- ж) добавление/удаление документов;
- з) форматирование документов;
- и) анализ деятельности компании и сотрудников;
4. Перечень подлежащих разработке вопросов
- 1) Анализ предметной области и постановка задачи.
- 2) Разработка требований и проектирование.
- 3) Реализация.
- 4) Контроль качества программного обеспечения.
- 5) Оформление пояснительной записки.

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	08.04.2024 – 13.04.2024	
2	Проектирование программных средств	20%	15.04.2024 – 25.04.2024	
3	Реализация программных средств	30%	26.04.2024 – 12.05.2024	
4	Тестирование программных средств	20%	13.05.2024 – 22.05.2024	
5	Оформление пояснительной записки	20%	23.05.2024 – 03.06.2024	

Дата выдачи задания «21» декабря 2023г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 125 страниц, 40 рисунков, 46 таблиц и 22 использованных источника.

МИКРОСЕРВИСНАЯ АРХИТЕКТУРА, БАЗА ДАННЫХ, ХРАНИЛИЩЕ НЕФТЕПРОДУКТОВ, ИНФОРМАЦИОННАЯ СИСТЕМА, GRPC, SQL

Цель выпускной квалификационной работы – разработать программные средства для серверного приложения для менеджмента хранилища нефтепродуктов.

Средства разработки – СУБД MS SQL SERVER 2019, язык программирования C#, case-средство Visual Paradigm, среда разработки Visual Studio 19 Community Edition.

Среда исполнения – Windows 10.

Результаты работы – в ходе выполнения выпускной квалификационной работы были созданы программные средства серверного приложения для менеджмента хранилища нефтепродуктов. Все требования к разработке приложения соблюдены.

					ПГУ 09.03.04 – 8ВР201.16 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Серверное приложение для системы менеджмента хранилища нефтепродуктов	Лит.	Лист	Листов
Разраб.		Лункова С.А.					4	125
Пров.		Шибанов С.В.				Группа 20ВП1		
Реценз.								
Н. Контр.		Такташкин Д.В.						
Утв.		Козлов А.Ю						

Содержание

Введение.....	8
1 Анализ предметной области и разработка концептуальной модели базы данных информационной системы.....	10
1.1 Анализ предметной области приложения	10
1.1.1 Анализ структурной организации предметной области	10
1.1.2 Анализ свойств объектов управления и информационных объектов предметной области	12
1.1.3 Анализ процессов управления и ведения данных в предметной области	14
1.2 Сравнительный анализ аналогичных разработок	17
1.2.1 Цели и задачи сравнительного анализа	17
1.2.2 Выбор приложений-аналогов для анализа	17
1.2.3 Обзор приложений-аналогов	18
1.2.4 Результаты сравнительного анализа	19
1.3 Постановка задачи на разработку серверного приложения	21
1.3.1 Цели разработки и назначение приложения	21
1.3.2 Перечень автоматизируемых функций	22
1.3.3 Требования к организации серверного приложения	23
1.4 Планирование и оценка процесса разработки серверного приложения	25
1.4.1 Выбор нотации и программных средств планирования и оценки процесса разработки	25
1.4.2 Планирование процесса разработки приложения	26
1.4.3 Оценка временных и финансовых затрат на разработку	29
2 Проектирование серверного приложения	33
2.1 Моделирование функциональных возможностей приложения	33
2.2 Разработка единой концептуальной модели данных	38
2.3 Обобщенная архитектура системы менеджмента хранилища нефтепродуктов	41

2.4 Требования к аппаратно-программному обеспечению серверного приложения	44
2.5 Обоснование выбора технологий и средств разработки серверного приложения	45
3 Разработка программного обеспечения серверного приложения.....	48
3.1 Разработка архитектуры серверного приложения	48
3.2 Разработка физической модели распределенной базы данных.....	51
3.3 Разработка программного обеспечения микросервиса «Авторизация».....	53
3.4 Разработка программного обеспечения микросервиса «Диспетчер»	55
3.5 Разработка программного обеспечения микросервиса «Менеджер»	58
4 Контроль качества программного обеспечения.....	63
4.1 Планирование процесса контроля качества программного обеспечения ..	63
4.1.1 Мероприятия по контролю качества программного обеспечения.....	63
4.1.2 Обоснование выбора модели тестирования	64
4.1.3 Обоснование выбора метрик программного кода	65
4.2 Тестирование программных средств.....	66
4.2.1 Функциональное тестирование.....	66
4.2.2 Нагрузочное тестирование	68
4.3 Оценка качества программного кода на основе метрик	71
4.3.1 Метрика Холстеда.....	71
4.3.2 Метрика Чепина	75
Заключение	77
Список использованных источников	79
Приложение А (справочное) Глоссарий	82
Приложение Б (рекомендуемое) Сценарии вариантов использования	84
Приложение В (рекомендуемое) Спецификации атрибутов физической модели данных и диаграмма физической модели данных	87
Приложение Г (рекомендуемое) Результат функционального тестирования	93
Приложение Д (рекомендуемое) Результат нагрузочного тестирования.....	96
Приложение Е (рекомендуемое) Листинг серверной части приложения	101

Приложение Ж (рекомендуемое) Исходные коды запросов	112
Приложение И (рекомендуемое) Исходные коды скриптов хранимых процедур и функций.....	115
Приложение К (рекомендуемое) Исходные коды скриптов триггеров.....	122

Введение

В настоящее время люди стараются автоматизировать все процессы своей жизни, производственная сфера стала одной из первых, где возникла такая необходимость. Так и индустрия по хранению, обработке и транспортировке нефтепродуктов не стала исключением – с ростом добычи ресурсов перед людьми появились новые вызовы: как эффективнее распределять добытые материалы, отслеживать исполнение производственных задач, оптимизировать документирование процессов купли-продажи нефтепродуктов.

Разработка приложения для системы управления нефтехранилищем – актуальная тема на сегодняшний день, такое приложение поможет повысить эффективность работы предприятия и ускорить рабочие процессы. Соответствующее приложение поможет решить ряд производственных проблем, начиная от формирования шаблонов документов и заканчивая проведением анализа работы предприятия. Не менее важным станет тот факт, что создание такого приложения не только упростит взаимодействие между администрацией и сотрудниками предприятия, сократит его расходы на оплату труда, но и позволит улучшить управление системами предприятия.

Целью данной работы является разработка серверного приложения для системы менеджмента хранилища нефтепродуктов. Это приложение должно позволять менеджерам предприятия просматривать и добавлять документы различных типов, редактировать их содержимое, генерировать на основе введенных данных полноценные договоры, проводить анализ сделок, а диспетчерам – манипулировать множеством резервуаров и насосов одновременно, отслеживать поломки на станции, анализировать переданный и полученный объем нефтепродуктов.

Исходя из цели данной работы, были поставлены следующие задачи:

- проанализировать предметную область и поставить задачу на разработку серверного приложения;
- спроектировать серверное приложение;

- разработать программное обеспечение;
- проконтролировать качество программного обеспечения.

В ходе выполнения выпускной квалификационной работы был проведен сравнительный анализ приложений-аналогов, реализованы поставленные бизнес-требования, проведена оценка времени и затрат на разработку приложения, спроектированы и разработаны 3 микросервиса для авторизации пользователей, аккаунтов диспетчера и менеджера, а также соответствующие им базы данных.

В результате получено полноценное клиент-серверное приложение современным дизайном, понятное даже начинающему пользователю. В дальнейшем можно будет оснащать приложение новым функционалом, согласно требованиям индустрии и предлагать производителям на рынке нефтедобывающих или обрабатывающих производств.

Так же стоит отметить, что приложение было представлено на международном конкурсе научно-технических проектов в области IT для студентов и учащихся ТехноФьюжн и заняло призовые места в номинациях «Лучшая техническая разработка» и «Лучшее прототипное решение».

1 Анализ предметной области и разработка концептуальной модели базы данных информационной системы

1.1 Анализ предметной области приложения

1.1.1 Анализ структурной организации предметной области

Хранилище нефтепродуктов (нефтехранилище, нефтебаза) – предприятие, состоящее из комплекса сооружений и установок, предназначенных для приема, хранения и отпуска нефтепродуктов потребителям [1]. Нефтепродукты – это продукты, получаемые в результате переработки нефти. К ним относятся бензин, дизельное топливо, керосин, мазут и прочие.

По существующим юридическим нормам все хранилища нефти и нефтепродуктов подразделяются на две группы:

- самостоятельные предприятия, предназначенные для снабжения потребителей нефтепродуктами;
- структурные подразделения, входящие в состав промышленных, транспортных, энергетических и других предприятий (речные и морские порты, аэропорты, теплоэнергоцентралы и т. п.).

По назначению нефтехранилища подразделяются на перевалочные, при заводские, завозные и распределительные.

Перевалочные нефтехранилища осуществляют перегрузку нефти или нефтепродуктов с одного вида транспорта на другой.

Распределительные нефтехранилища, непосредственно снабжающие нефтепродуктами мелких потребителей, рассчитаны на ограниченный район обслуживания и предназначены для непродолжительного хранения нефтепродуктов и снабжения ими потребителей обслуживаемого района.

Перевалочно-распределительные нефтехранилища совмещают функции перевалочных и распределительных нефтебаз [2].

Общая организационно-производственная структура нефтебазы приведена на рисунке 1.

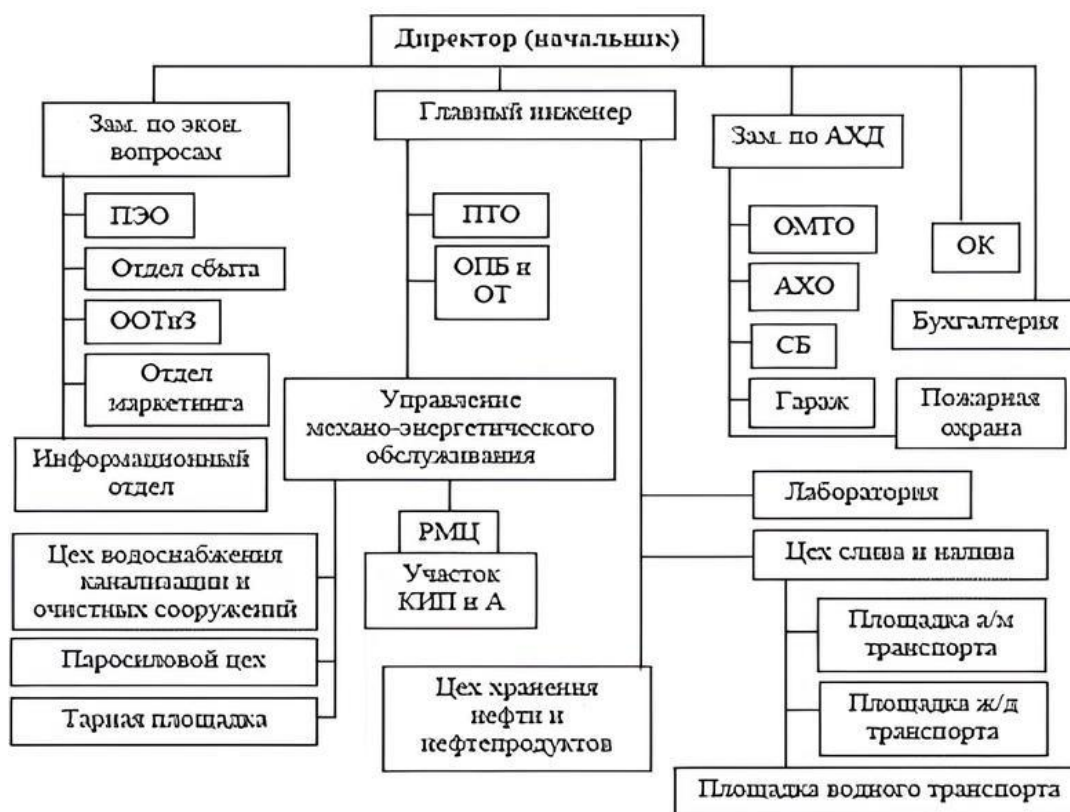


Рисунок 1 – Схема организационно-производственной структуры нефтебазы

Корпоративная сеть хранилищ нефтепродуктов состоит из нескольких распределительных и перевалочно-распределительных хранилищ, принадлежащих одной компании. Каждое из хранилищ нефти относится к определенному классу по грузообороту и пожароопасности.

Исходя из конкретных задач и сооружений нефтебазы, формируют ее цехи и службы: цех хранения нефти и нефтепродуктов, цех слива и налива, ремонтно-механический цех, паросиловой цех, лаборатория и т. д.

Цех хранения нефти и нефтепродуктов предназначен для обслуживания резервуарного парка. Этот цех обеспечивает количественный контроль за приемом и выдачей нефтепродуктов. Цех слива и налива осуществляет прием и слив нефтепродуктов с различных видов транспорта (железнодорожного, водного, трубопроводного и др.), а также передачу и налив нефтепродуктов для отправки их получателям.

Ремонтно-механический цех выполняет планово-предупредительный осмотр и ремонт оборудования, сооружений и других видов производственных объектов нефтебазы.

Отдел менеджмента ведет соглашения с контрагентами о поставках нефтепродуктов, о сдаче резервуаров в аренду и создает заказы на отпуск нефтепродуктов.

1.1.2 Анализ свойств объектов управления и информационных объектов предметной области

Для проектируемой информационной системы существуют такие объекты управления, как резервуары и насосы. В качестве информационных объектов выступают документы.

Существует несколько различных классификаций резервуаров, каждая из которых имеет свои особенности. Например, в зависимости от своего расположения, ёмкости могут быть:

- наземными, когда весь резервуар находится на поверхности земли;
- полуподземными, когда основная часть цистерны находится в земле, а наружу выходит только горловина;
- полностью подземными, предназначенными для более долгого хранения и экономии свободного места;
- подводными, требующими повышенного внимания.

Важным критерием классификации резервуаров является тип материала, из которого выполнены резервуары. В зависимости от него, их можно разделить на каркасные и мягкие. Ещё одним различием можно считать саму форму резервуара, которая бывает следующих видов:

- цилиндрическая, обеспечивающая возможность создания резервуара любой вместительности и не требующая особых усилий при обслуживании;
- сферическая отличается круглой формой и требует специальную конструкцию для установки;

– каплевидная форма характерна для резервуаров из мягких материалов, для их создания не используется каркас, поэтому его можно оборудовать практически в любом месте [3].

Так же существует три класса резервуаров, учитывающих объем емкости и ее расположение:

– первый класс включает в себя наиболее опасные резервуары, объем которых превышает 10000 кубических метров, а также ёмкости, чей объем превышает 5000 кубических метров и если они расположены в черте города;

– второй класс включает в себя резервуары повышенной опасности объемом от 5 до 10 тысяч кубических метров;

– к третьему классу относятся ёмкости объемом от 100 до 5000 кубических метров [2].

Нефтяные насосы представляют собой агрегаты, предназначенные для работы с нефтепродуктами. Именно они выполняют функцию перекачки, при этом обеспечивая высокие показатели надежности. Нефтяные насосы могут иметь разные показатели эффективности в зависимости от модели, однако немало влияет на этот показатель и привод. Его тип значительно воздействует на эксплуатационные характеристики разрабатываемых скважин.

Нефтяные насосы делятся на два вида: винтовые и центробежные.

Первый тип чаще всего используется в случае, если условия эксплуатации будут суровыми. Даная установка применяется для перекачки пульпы, шлама, рассола и сырой нефти. Центробежные насосные установки используются для перекачки нефти и веществ, чья температура не выше 200 градусов.

Нефтяные насосы проектируются и создаются исходя из условий эксплуатации. Это означает, что существуют модели, оптимизированные под тот или иной климат или условия. К примеру, есть насосные установки, способные функционировать при экстремально низких температурах или даже под толщей воды на морских нефтедобывающих объектах [4].

Документы можно разделить по их видам: договор на потребление и сбыт нефтепродуктов, договор на аренду резервуаров.

1.1.3 Анализ процессов управления и ведения данных в предметной области

Хранилище нефти состоит из нескольких наземных и подземных цилиндрических резервуаров и предназначено для накопления, хранения и отгрузки нефтепродуктов. Каждый из резервуаров предназначен для работы с определенным видом нефтепродуктов, может находиться в рабочем или нерабочем состоянии, оборудован двумя центробежными насосами, один из которых осуществляют закачку нефти в резервуар, а второй откачку нефти из резервуара. Оба насоса каждого из резервуаров могут работать одновременно, откачка нефти из резервуара возможна только тогда, когда уровень нефтепродуктов в резервуаре не менее определенного нижнего допустимого уровня, так же закачка нефти в резервуар возможна только тогда, когда уровень нефтепродуктов в резервуаре не превышает определенного верхнего допустимого уровня.

Насосы каждого резервуара управляются диспетчером в ручном режиме. При этом автоматически контролируется уровень нефтепродуктов в резервуаре, и, в случае необходимости, любой из насосов может быть автоматически отключен. Кроме того, диспетчер может устанавливать параметры системы: нижний допустимый уровень нефтепродуктов, верхний допустимый уровень нефтепродуктов для каждого резервуара, и скорость перекачки нефтепродуктов каждым из насосов. Так же присутствует управление и контроль доступа пользователей к резервуарам.

Для аналитики работы диспетчера и менеджера за смену генерируется сводный отчет с представлением графических схем по различным параметрам: рейтинг контрагентов по объему продаж и поставок, рейтинг контрагентов по выгоде сделок, рейтинг используемых нефтепродуктов и др.

Сценарий выполнения работы с резервуарами диспетчера в виде диаграммы деятельности представлен на рисунке 2.

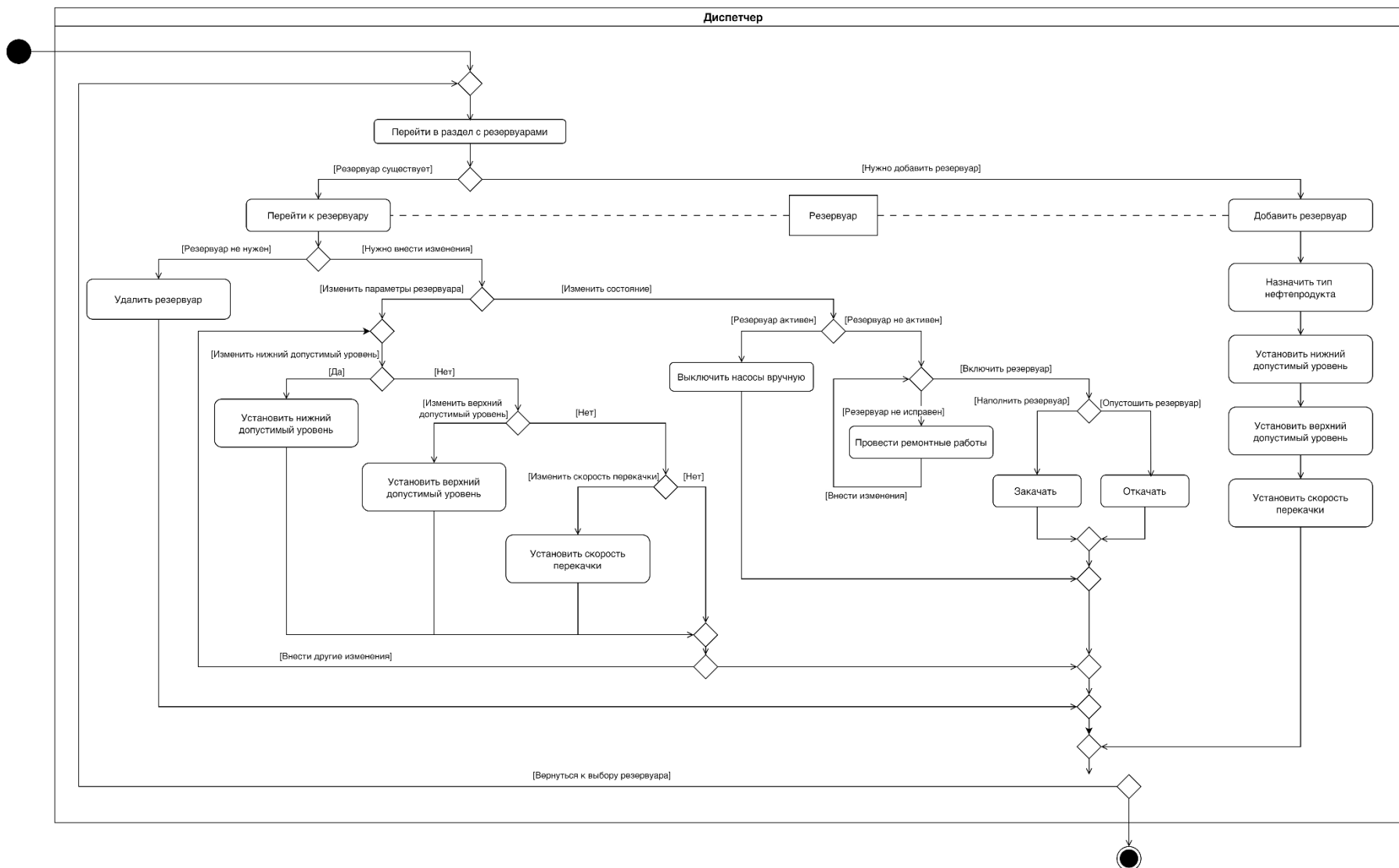


Рисунок 2 – Диаграмма деятельности для работы с резервуарами

Кроме того, в системе предусмотрена генерация юридических документов на аренду резервуаров, потребление и сбыт нефтепродуктов, посредством занесения менеджером необходимых данных в систему.

Сценарий выполнения работы с документами менеджера в виде диаграммы деятельности представлен на рисунке 3.

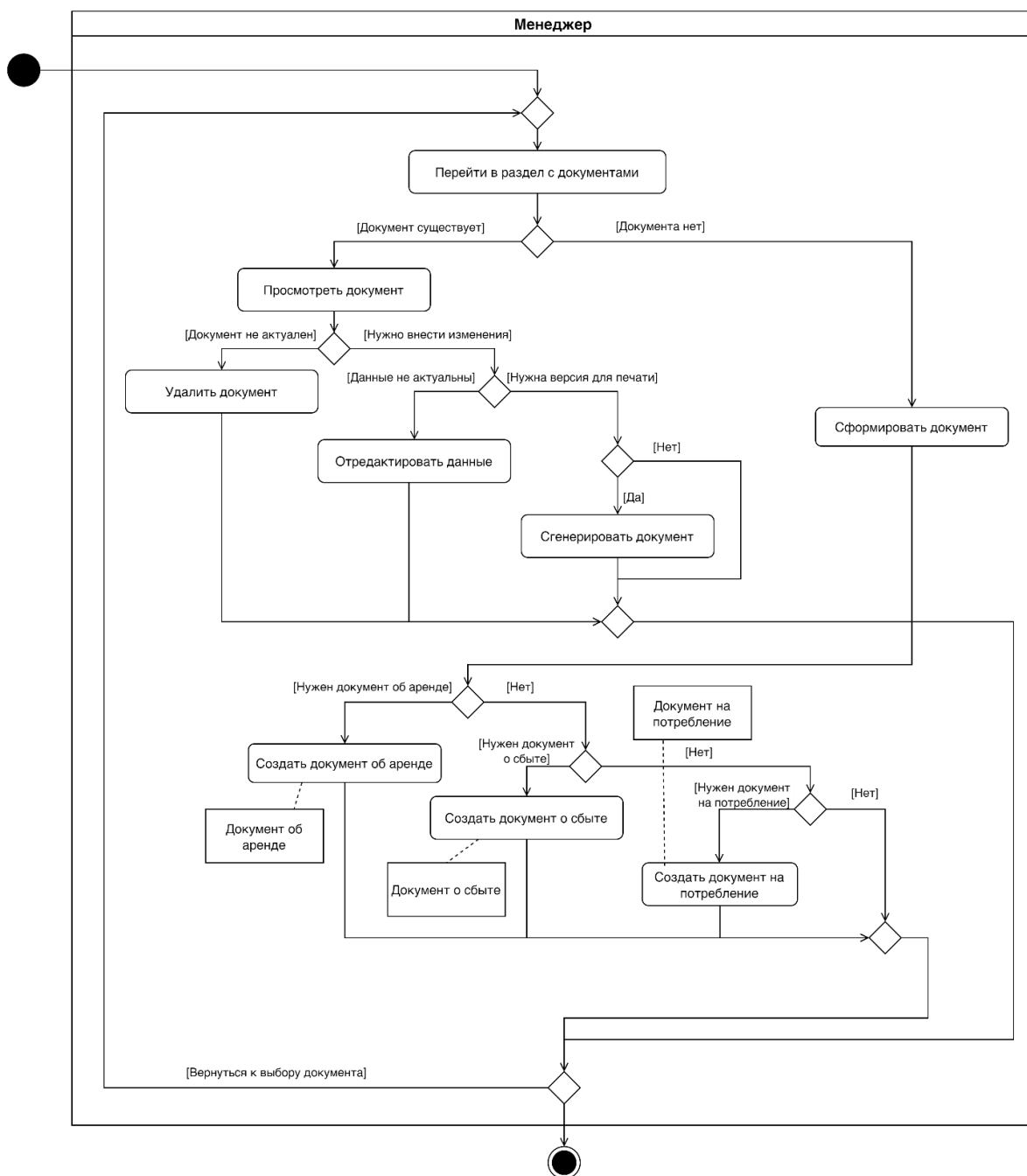


Рисунок 3 – Диаграмма деятельности для работы с документами

На данных диаграммах представлено подробное описание процессов, описывающих работу диспетчера и менеджера.

1.2 Сравнительный анализ аналогичных разработок

1.2.1 Цели и задачи сравнительного анализа

Для решения задачи управления нефтехранилищами в компаниях уже существуют различные мобильные и настольные приложения, которые могут предоставлять функционал, схожий с проектируемым.

Сравнительный анализ приложений-аналогов проводится в целях выявления преимуществ и недостатков каждого из существующих на рынке решений, определения базового функционала, который необходим для реализации приложения.

Задачи сравнительного анализа включают в себя:

- определение критериев оценки;
- выбор приложений к сравнению из потенциальных аналогов для анализа;
- исследование каждого из приложений на основе выбранных критериев;
- обработку полученных результатов.

Определим основные критерии анализа приложений-аналогов, по которым будет проводиться сравнение:

- полнота и качество исполнения основных функций;
- удобство интерфейса;
- скорость работы приложения и его надежность;
- масштабируемость;
- безопасность;

СТОИМОСТЬ.

Рассмотрим несколько аналогов и проанализируем их основные возможности.

1.2.2 Выбор приложений-аналогов для анализа

Для сравнительного анализа было выбрано 5 приложений. Краткие сведения о приложениях представлены в таблице 1.

Таблица 1 – Сведения о приложениях-аналогах

Приложение	Производитель	Описание
Программа «Альфа–нефтебаза 2.0»	ПК-ЭЛЕКТРОНИКС	Предназначена для автоматизированного управления технологическими процессами приемки, налива, хранения и коммерческого учета нефтепродуктов посредством персонального компьютера.
Программно-аппаратный комплекс для нефтебаз	Сибнефтекарт	Предназначен для автоматизации технологических процессов приема-отпуска топлива на нефтебазах, упрощения технологии учета движения топлива и ведения единой информационной базы данных.
Система управления «ДОЗА-Программа управления нефтебазой»	АЗСДозаСервис	Программа позволяет проводить слив нефтепродукта с различного транспорта с оформлением актов приема и занесением в базу данных.
Программа управления резервуарами	Аврога	Приложение, которое позволяет производителям регистрировать добычу и транспортировку жидкости, используя точные измерения уровня и расхода.
АСУ «Топаз-нефтебаза»	ТОПАЗ-электро	Комплексная система управления автоматизированными системами налива и приемными устройствами нефтебазы обеспечивает учет движения горюче-смазочных материалов, получение отчетности, автоматизирует работу оператора, облегчает администрирование нефтебазой, работает с основным и широким спектром дополнительного оборудования

Каждое из программных продуктов предназначено для управления нефтестанциями и активно используется на рынке.

1.2.3 Обзор приложений-аналогов

Проведем подробный обзор приложений-аналогов, выбранных к сравнению, согласно выделенным критериям оценки.

Программа «Альфа–нефтебаза 2.0» обладает широким спектром функциональных возможностей, включая базовые: формирование заявок на отпуск и приемку нефтепродуктов, управление нефтестанцией, контроль резервуарного парка, учет всех действий, проводимых в системе, обработка нештатных ситуаций и многое другое. Однако данная программа имеет сложный интерфейс, и неопытный пользователь не сможет интуитивно понять устройство системы. В любой момент времени оператору АЗС и удаленному офису доступна

информация о всех характеристиках станции, так что программа является высокоскоростной. Программа Альфа Нефтебаза имеет модульный характер и легко масштабируется [5].

Программно-аппаратный комплекс для нефтебаз предназначен в первую очередь для налива топлива на АЗС, поэтому он не обладает базовым функционалом для управления самим нефтехранилищем, так же система является плохо масштабируемой. Значительными преимуществами данной системы являются простота и удобство интерфейса, быстрота отклика, а также общедоступность [6].

Система управления «ДОЗА» имеет наиболее обширные функциональные возможности для гибкого управления нефтестанцией посредством различных прав доступа к системе. Так же она обладает интуитивно понятным интерфейсом, высокой производительностью и безопасностью [7].

Программа управления резервуарами предназначена для фиксации количества производимых и доставляемых нефтепродуктов, ведения журнала транспортировки и создания графических схем с статистическими данными о резервуарах. Она отвечает всем требованиям к производительности и удобству интерфейса, однако безопасность программы и ее масштабируемость находятся на ненадлежащем уровне, и сейчас ведутся работы по их улучшению [8].

Топаз-нефтебаза предоставляет отличный функционал, агрегирует в себе управление потоком нефтепродуктов, контроль состояний резервуаров, ведение справочников контрагентов, формирование любого вида отчетности и др. Интерфейс не только удобный, но и имеет возможность настройки отображения, чтобы во время длительной работы оператора снижалась психологическая и физическая нагрузка. Так же хорошо реализована защита данных и разделение прав доступа к системе. Значительным недостатком является отсутствие масштабируемости системы для версии, находящейся в свободном доступе [9].

1.2.4 Результаты сравнительного анализа

По результатам сравнительного анализа приложение «Топаз-нефтебаза»

фирмы «ТОПАЗ-электро» стало лидирующим для некоммерческого применения, так как оно отвечает большинству критериев и простое в использовании. Наглядная сводка результатов сравнения приложений представлена в таблице 2.

Таблица 2 – Сравнение приложений-аналогов

Название	Критерии сравнения					
	Функциональные возможности	Удобство интерфейса	Скорость работы	Масштабируемость	Безопасность	Стоимость
Альфа–нефтебаза 2.0	+	-	+	+	+	-
Программно-аппаратный комплекс для нефтебаз	-	+	+	-	-	+
ДОЗА	+	+	+	+	+	-
Программа управления резервуарами	-	+	+	-	-	-
Топаз-нефтебаза	+	+	+	-	+	+

В ходе сравнения приложений-аналогов были выявлены базовые функции, которые присутствуют во всех приложениях:

- печать товарно-транспортных накладных;
- прием и отпуск нефтепродуктов;
- формирование отчетности по наличию нефтепродуктов;
- отображение состояний резервуаров и физических характеристик нефтепродуктов в них.

В приложении «Альфа–нефтебаза 2.0» присутствует ряд интересных функций таких, как видеонаблюдение в удаленном офисе за резервуарным парком и операторами, звуковое и визуальное оповещение об аварийных ситуациях, автоматическое формирование сменных отчетов.

Так же в приложении «ДОЗА» реализованы технологические операции для контроля качества оборудования, а также компаундирование.

Исходя из полученных данных был отобран основной функционал для проектируемой системы. В качестве дополнения к возможностям системы будут

реализованы следующие функции:

- автогенерация договоров на сбыт, потребление нефтепродуктов и аренду резервуаров;
- группировка резервуаров в соответствии с договорами;
- ведение журнала действий в системе;
- фильтрация резервуаров и договоров по заданным критериям.

1.3 Постановка задачи на разработку серверного приложения

1.3.1 Цели разработки и назначение приложения

Проанализировав предметную область и изучив приложения-аналоги на рынке, было сформулировано основное назначение приложения для системы менеджмента для управления нефтехранилищем.

Информационная система послужит компании для ускорения работы предприятия по хранению нефтепродуктов, а также для упрощения процесса ведения сделок с контрагентами и анализа деятельности компании. Данное приложение усилит контроль за всеми процессами на нефтебазе, предотвращая возможные аварийные ситуации, что немаловажно для обеспечения безопасности сотрудников и сохранения дорогостоящего оборудования. Кроме того, информационная система поможет компании сократить затраты на оплату труда и уменьшить количество ошибок, возникающих при ручном вводе данных, благодаря автоматизации процессов составления договоров, которые ускоряют связь между контрагентами и организацией.

В рамках данной выпускной квалификационной работы разрабатывается серверная часть приложения. Разрабатываемые программные средства системы менеджмента для управления хранилищем нефтепродуктов должны обеспечивать пользователя набором удобных инструментов для поддержания основных функций приложения таких, как обработка запросов от клиентов, взаимодействие с базой данных и управление доступом к различным ресурсам приложения. Серверное приложение должно обеспечивать стабильную и эффективную работу системы для выполнения ее бизнес-процессов.

1.3.2 Перечень автоматизируемых функций

Программные средства информационной системы должны обладать следующими автоматизируемыми функциями: упрощение ручной обработки данных при составлении договоров, распределение нефтепродуктов по резервуарам и их транспортировку в конечные точки, а также создание отчетов о проделанной работе диспетчеров и менеджеров станции.

Перечень автоматизируемых функций для диспетчера представлен в таблице 3.

Таблица 3 – Перечень автоматизируемых функций для диспетчера

Функция	Описание
Распределение нефтепродуктов по резервуарам	Диспетчер назначает каждому резервуару определенный вид нефтепродукта в соответствии с типом резервуара и договором с контрагентом.
Отображение состояний емкостей	Для каждого резервуара отображается полная информация о его номере, типе, хранящемся нефтепродукте, уровне жидкости в резервуаре, верхнем и нижнем допустимом уровне, направлении транспортировки и его текущем состоянии, так же часть этих данных представлена в миниатюре для выбора резервуара.
Добавление/удаление резервуаров	Диспетчер может вводить в систему новый резервуар или сразу список резервуаров, задавая их типы и номера, которые не подлежат редактированию. Так же есть возможность удаления резервуаров по заданному номеру.
Закачка/откачка нефтепродуктов	За каждым резервуаром закреплены насос закачки и выпускной насос, для которых можно задать скорость перекачки, контролировать процесс включения и выключения вручную или автоматически, по достижению верхнего/нижнего допустимых уровней.
Редактирование параметров резервуара	В системе доступны для редактирования следующие параметры: установка нижнего и верхнего допустимых уровней резервуара, назначение определённого нефтепродукта, изменение скорости перекачки насосов.
Ведения журнала действий	Система фиксирует все действия, производимые в системе, и записывает в журнал над чем произведена операция, номер ресурса, что именно было сделано и фиксирует дату и время исполнения
Анализ работы диспетчера	По завершению смены создается отчет о проделанной работе, включающий в себя резервуары, с которыми было взаимодействие, объем перекаченных нефтепродуктов, их стоимость и время работы. А также выводятся диаграммы для сравнения объема сбыта и поставок нефтепродуктов и типов нефтепродуктов, с которыми чаще осуществляется работа.

В таблице 4 приведено описание автоматизируемых функций менеджера.

Таблица 4 – Перечень автоматизируемых функций для менеджера

Функция	Описание
Добавление/удаление документов	Менеджер заводит в систему необходимые данные (номер, сумма, период действия, объем поставки/сбыта, контрагент, нефтепродукт) в соответствии с типом добавляемого документа. Так же есть возможность удаления документов по заданному номеру.
Формирование документов	Система генерирует полноценный юридический документ в необходимом формате, в соответствии с хранящимися данными.
Редактирование данных документов, отслеживание их актуальности	В системе доступны для редактирования следующие параметры: дата начала и окончания действия договора, сумма сделки, объем поставки/сбыта, название компании контрагента и нефтепродукта. Так же в миниатюре указывается текущее состояние
Анализ деятельности компании	По завершению смены создается отчет о проделанной работе, анализ деятельности компании по различным критериям: рейтинг контрагентов по выгоде сделок, по объему сбыта и поставок продукции, а также рейтинг нефтепродуктов по поставке и продаже.
Ведения журнала действий	Система фиксирует все действия, производимые в системе, и записывает в журнал над чем произведена операция, номер документа, что именно было сделано и фиксирует дату и время исполнения

Таким образом, на предприятии повысится эффективность работы и контроль за качеством исполнения работы сотрудниками, что позволит своевременно выявить и устранить недочеты.

1.3.3 Требования к организации серверного приложения

Система для управления нефтехранилищем будет организована согласно микросервисной архитектуре, состоять из нескольких независимых модулей, каждый из которых будет создан для конкретных задач. Серверное приложение должно иметь возможность масштабироваться для обслуживания большого числа запросов, исполнять весь функционал, обеспечивающий логику работы приложения, возвращать результаты клиенту, корректно обработав входящие запросы, обрабатывать исключения, и выводить соответствующие сообщения об ошибках.

Микросервисная архитектура приложения представляет собой подход, при котором приложение строится как набор небольших сервисов, каждый из которых работает в собственном процессе и взаимодействует с остальными,

используя API. Микросервисная архитектура должна строиться согласно ключевым концепциям функционирования сервисов:

- возможность независимого развертывания;
- для определения границ сервисов, их стоит моделировать вокруг предметных областей бизнеса;
- каждый микросервис должен иметь собственную базу данных для контроля ситуации и сокрытия информации;
- микросервисы должны быть гибкими в отношении масштабирования, надежности и технического направления;
- микросервисы должны обеспечивать надлежащий уровень безопасности и быстро восстанавливаться после сбоев [10].

Так же для организации баз данных информационной системы существуют свои требования для обеспечения сохранности и упорядоченности данных и быстрой работы с ними. К ним относятся:

- выстраиваются многосторонние связи, чтобы обеспечивалась возможность установления надежных связей элементов данных и быстрого согласования вносимых изменений;
- должна обеспечиваться хорошая пропускная способность с минимальными задержками ответа на запрос;
- нужно избегать избыточности данных во избежание конфликтов и загрузки памяти;
- возможность построения и обработки запросов и ответов, которые заранее не планировались;
- должна обеспечиваться целостность данных, чтобы в случае сбоев система могла восстановиться без потерь;
- для обеспечения целостности данных, программные компоненты должны поддерживать концепцию транзакций.

Серверная часть сервиса будет представлена gRPC-сервером, который служит промежуточным слоем между запрашиваемыми ресурсами и базой данных. Сервер принимает запросы от любого модуля клиентской части

приложения, обрабатывает его, затем формирует и отправляет SQL-запрос согласно пришедшим от клиента данным. После того как от базы данных получен ответ, сервер обрабатывает его и направляет модулю, который запрашивал данные.

При разработке архитектуры важно опираться на основные требования к ее организации, чтобы избежать основных трудностей, возникающих в процессе разработки.

1.4 Планирование и оценка процесса разработки серверного приложения

1.4.1 Выбор нотации и программных средств планирования и оценки процесса разработки

В современном мире существует множество инструментов для планирования проектов, позволяющих организовывать и контролировать выполнение задач. Они упрощают процесс коммуникации в команде, облегчают работу руководителей и позволяют им следить за прогрессом выполнения задач, управлять ресурсами и бюджетом проекта.

Основная цель планирования – это составление графика выполнения задач и приблизительный расчет затрат на проект. При разработке программного обеспечения одной из главных задач является грамотное планирование проекта, так как от него зависит качество конечного продукта и его своевременное завершение.

Для определения длительности работ, составления графика задач и оценки процесса разработки будет использоваться приложение для управления программными продуктами Microsoft Office Project.

Microsoft Office Project – это программа для управления проектами, разработанная корпорацией Microsoft. Данное приложение активно используется среди разработчиков, благодаря своей функциональной простоте, широкому набору инструментов, а также возможности совместной работы пользователей. Оно поддерживает построение различных графиков и диаграмм для проекта,

управление его ресурсами, анализ рабочих нагрузок и генерацию отчетов [11].

1.4.2 Планирование процесса разработки приложения

Для приложения для системы менеджмента хранилища нефтепродуктов были поставлены задачи на проектирование и разработку. За дату начала проекта выбрано 8 апреля 2024 года, работа осуществлялась по пятидневной рабочей недели. Перечень задач проекта, их связи, длительности приведены в таблице 5.

Таблица 5 – Список задач проекта

№	Название задачи	Длительность (дней)	Предшественники
1	Начало проекта	0	-
2	Анализ предметной области и постановка задачи на разработку серверного приложения	11,5	-
3	Анализ предметной области приложения	1	-
4	Анализ структурной организации предметной области	0,33	1
5	Анализ свойств объектов управления и информационных объектов предметной области	0,34	4
6	Анализ процессов управления и ведения данных в предметной области	0,33	5
7	Сравнительный анализ аналогичных разработок	1	-
8	Цели и задачи сравнительного анализа	0,25	6
9	Выбор приложений-аналогов для анализа	0,25	8
10	Обзор приложений-аналогов	0,25	9
11	Результаты сравнительного анализа	0,25	10
12	Постановка задачи на разработку серверного приложения	7	-
13	Цели разработки и назначение приложения	1	11
14	Перечень автоматизируемых функций	2	13
15	Требования к организации серверного приложения	2	14
16	Постановка требований к интерфейсу программных средств	2	15
17	Планирование и оценка процесса разработки серверного приложения	2,5	-
18	Выбор нотации и программных средств планирования и оценки процесса разработки	0,25	16
19	Планирование процесса разработки приложения	2	18
20	Оценка временных и финансовых затрат на разработку	0,25	19
21	Проектирование серверного приложения	7,25	-
22	Моделирование функциональных возможностей приложения	1	20
23	Разработка единой концептуальной модели данных	2	22

Продолжение таблицы 5

№	Название задачи	Длительность (дней)	Предшественники
24	Обобщённая архитектура системы менеджмента хранилища нефтепродуктов	3	23
25	Требования к аппаратно-программному обеспечению серверного приложения	1	24
26	Обоснование выбора технологий и средств разработки серверного приложения	0,25	25
27	Разработка программного обеспечения серверного приложения	25	-
28	Разработка архитектуры серверного приложения	5	26
29	Разработка физической модели распределенной базы данных	5	28
30	Разработка программного обеспечения микросервиса «Авторизация»	3	29
31	Разработка программного обеспечения микросервиса «Диспетчер»	7	30
32	Разработка программного обеспечения микросервиса «Менеджер»	5	31
33	Разработка пользовательского интерфейса	5	29
34	Контроль качества программного обеспечения	8	-
35	Планирование процесса контроля качества программного обеспечения	3	33
36	Тестирование программных средств	3	35
37	Оценка качества программного кода	2	36
38	Сдача проекта	1	34
39	Оформление и сбор документации	5	38
40	Ввод в эксплуатацию	1	39
41	Конец проекта	0	40

Далее необходимо определить используемые ресурсы и распределить их по каждой из задач. В таблице 6 представлен перечень ресурсов проекта.

Таблица 6 – Перечень ресурсов проекта

Название ресурса	Тип	Затраты	
		Станд. ставка	Ставка сверхур.
Аналитик	Т	200р./ч	300р./ч
Программист 1	Т	310р./ч	450р./ч
Программист 2	Т	270р./ч	400р./ч
Разработчик базы данных	Т	310р./ч	450р./ч
Дизайнер	Т	200р./ч	300р./ч
Тестировщик	Т	200р./ч	300р./ч
Интернет	Т	-	-
Компьютер	Т	70000р.	-

После составления списка ресурсов для проекта, необходимо распределить

их по задачам. Данная информация представлена в таблице 7.

Таблица 7 – Распределение ресурсов проекта по задачам

№	Задача	Ресурс	Единицы (затраты)
1	Начало проекта	-	-
2	Анализ предметной области и постановка задачи на разработку серверного приложения	Интернет	300
3	Анализ предметной области приложения	-	-
4	Анализ структурной организации предметной области	Аналитик	100
5	Анализ свойств объектов управления и информационных объектов предметной области	Аналитик	100
6	Анализ процессов управления и ведения данных в предметной области	Аналитик	100
7	Сравнительный анализ аналогичных разработок	-	-
8	Цели и задачи сравнительного анализа	Аналитик	100
9	Выбор приложений-аналогов для анализа	Аналитик	100
10	Обзор приложений-аналогов	Аналитик	100
11	Результаты сравнительного анализа	Аналитик	100
12	Постановка задачи на разработку серверного приложения	-	-
13	Цели разработки и назначение приложения	Программист 1 Программист 2	50 50
14	Перечень автоматизируемых функций	Программист 1	100
15	Требования к организации серверного и клиентского приложения	Программист 1 Программист 2	50 50
16	Постановка требований к интерфейсу программных средств	Программист 2	100
17	Планирование и оценка процесса разработки серверного приложения	-	-
18	Выбор нотации и программных средств планирования и оценки процесса разработки	Программист 1 Программист 2	50 50
19	Планирование процесса разработки приложения	Аналитик	100
20	Оценка временных и финансовых затрат на разработку	Аналитик	100
21	Проектирование серверного приложения	Интернет	200
22	Моделирование функциональных возможностей приложения	Программист 1	100
23	Разработка единой концептуальной модели данных	Разработчик базы данных	100
24	Обобщенная архитектура системы менеджмента хранилища нефтепродуктов	Программист 1	100
25	Требования к аппаратно-программному обеспечению серверного приложения	Программист 1 Программист 2	50 50

Продолжение таблицы 7

№	Задача	Ресурс	Единицы (затраты)
26	Обоснование выбора технологий и средств разработки серверного приложения	Программист 1 Программист 2	50 50
27	Разработка программного обеспечения серверного приложения	Интернет	350
28	Разработка архитектуры серверного приложения	Программист 1	10
29	Разработка физической модели распределенной базы данных	Разработчик базы данных	100
30	Разработка программного обеспечения микросервиса «Авторизация»	Программист 1	100
31	Разработка программного обеспечения микросервиса «Диспетчер»	Программист 1	100
32	Разработка программного обеспечения микросервиса «Менеджер»	Программист 1	100
33	Разработка пользовательского интерфейса	Дизайнер Программист 2	100 50
34	Контроль качества программного обеспечения	Интернет	200
35	Планирование процесса контроля качества программного обеспечения	Тестировщик	100
36	Тестирование программных средств	Тестировщик	100
37	Оценка качества программного кода	Тестировщик	100
38	Сдача проекта	-	-
39	Оформление и сбор документации	Программист 1 Программист 2	50 50
40	Ввод в эксплуатацию	Программист 1	100
41	Конец проекта	-	-

В систему Microsoft Project были внесены перечни задач и ресурсов с их распределением.

1.4.3 Оценка временных и финансовых затрат на разработку

После занесения данных в Microsoft Project программой были сгенерированы диаграмма Ганта, отчеты по общим затратам по проекту и использованию ресурсов.

Диаграмма Ганта представляет собой график проекта, который отображает его реализацию с течением времени. По вертикали отображаются задачи проекта, а по горизонтали – сроки их исполнения.

Диаграмма Ганта представлена на рисунке 4, она отображает работы по проекту с использованием соответствующим им ресурсам. За дату начала работ было взято 8 апреля 2024.

Согласно полученной диаграмме Ганта окончание работ по проекту запланировано на 13 июня.

Далее, программой был сгенерирован отчет по общим затратам по проекту, представленный на рисунке 5.

Отчет о бюджете представляет собой общую смету проекта, с учетом планируемых затрат, связанных с выполнением конкретных задач по проекту, а также расходов, понесенных в связи с выполнением работы.

Отчет о бюджете от Вс 05.05.24
Проект1

Ид.	Название задачи	Фиксированные затраты	Начисление фикс. затрат	Общие затраты
31	Разработка программного обеспечения	0,00 Р	Пропорциональное	17 360,00 Р
33	Разработка пользовательского интерфейса	0,00 Р	Пропорциональное	13 400,00 Р
28	Разработка архитектуры серверного приложения	0,00 Р	Пропорциональное	12 400,00 Р
29	Разработка физической модели рас	0,00 Р	Пропорциональное	12 400,00 Р
32	Разработка программного обеспечения	0,00 Р	Пропорциональное	12 400,00 Р
39	Оформление и сбор документации	0,00 Р	Пропорциональное	11 600,00 Р
24	Обобщенная архитектура системы м	0,00 Р	Пропорциональное	7 440,00 Р
30	Разработка программного обеспечения	0,00 Р	Пропорциональное	7 440,00 Р
14	Перечень автоматизируемых функц	0,00 Р	Пропорциональное	4 960,00 Р
23	Разработка единой концептуальной	0,00 Р	Пропорциональное	4 960,00 Р
35	Планирование процесса контроля ка	0,00 Р	Пропорциональное	4 800,00 Р
36	Тестирование программных средств	0,00 Р	Пропорциональное	4 800,00 Р
15	Требования к организации серверно	0,00 Р	Пропорциональное	4 640,00 Р
16	Постановка требований к интерфей	0,00 Р	Пропорциональное	4 320,00 Р
19	Планирование процесса разработки	0,00 Р	Пропорциональное	3 200,00 Р
37	Оценка качества программного кода	0,00 Р	Пропорциональное	3 200,00 Р
22	Моделирование функциональных вс	0,00 Р	Пропорциональное	2 480,00 Р
40	Ввод в эксплуатацию	0,00 Р	Пропорциональное	2 480,00 Р
13	Цели разработки и назначение прил	0,00 Р	Пропорциональное	2 320,00 Р
25	Требования к аппаратно-программн	0,00 Р	Пропорциональное	2 320,00 Р
18	Выбор нотации и программных сред	0,00 Р	Пропорциональное	580,00 Р
26	Обоснование выбора технологий и с	0,00 Р	Пропорциональное	580,00 Р
5	Анализ свойств объектов управлен	0,00 Р	Пропорциональное	544,00 Р
4	Анализ структурной организации пр	0,00 Р	Пропорциональное	528,00 Р
6	Анализ процессов управления и вед	0,00 Р	Пропорциональное	528,00 Р
8	Цели и задачи сравнительного анал	0,00 Р	Пропорциональное	400,00 Р
9	Выбор приложений-аналогов для а	0,00 Р	Пропорциональное	400,00 Р
10	Обзор приложений-аналогов	0,00 Р	Пропорциональное	400,00 Р
11	Результаты сравнительного анализ	0,00 Р	Пропорциональное	400,00 Р
20	Оценка временных и финансовых з	0,00 Р	Пропорциональное	400,00 Р
1	Начало проекта	0,00 Р	Пропорциональное	0,00 Р
38	Сдача проекта	0,00 Р	Пропорциональное	0,00 Р
41	Конец проекта	0,00 Р	Пропорциональное	0,00 Р
		0,00 Р		143 680,00 Р

Рисунок 5 – Отчет об общих затратах проекта

Так же Microsoft Project предоставила отчет по использованию ресурсов проекта. Данный отчет показывает распределение времени на материально-технические и трудовые ресурсы. Ресурсами считается все, чем располагает проект: сотрудники, оборудование, материалы и сырье, технологии, время, информация и т. д.

Отчет об использовании ресурсов представлен на рисунке 6.

	Итого
Аналитик	34 ч
Анализ структурной организации предметной области	2,63 ч
Анализ свойств объектов управления и информационных объектов предметной области	2,72 ч
Анализ процессов управления и ведения данных в предметной области	2,63 ч
Цели и задачи сравнительного анализа	2 ч
Выбор приложений-аналогов для анализа	2 ч
Обзор приложений-аналогов	2 ч
Результаты сравнительного анализа	2 ч
Планирование процесса разработки приложения	16 ч
Оценка временных и финансовых затрат на разработку	2 ч
Программист 1	254 ч
Цели разработки и назначение приложения	4 ч
Перечень автоматизируемых функций	16 ч
Требования к организации серверного и клиентского приложения	8 ч
Выбор нотации и программных средств планирования и оценки процесса разработки	1 ч
Моделирование функциональных возможностей приложения	8 ч
Обобщенная архитектура системы менеджмента хранилища нефтепродуктов	24 ч
Требования к аппаратно-программному обеспечению серверного приложения	4 ч
Обоснование выбора технологий и средств разработки серверного приложения	1 ч
Разработка архитектуры серверного приложения	40 ч
Разработка программного обеспечения микросервиса "Авторизация"	24 ч
Разработка программного обеспечения микросервиса "Диспетчер"	56 ч
Разработка программного обеспечения микросервиса "Менеджер"	40 ч
Оформление и сбор документации	20 ч
Ввод в эксплуатацию	8 ч
Программист 2	74 ч
Цели разработки и назначение приложения	4 ч
Требования к организации серверного и клиентского приложения	8 ч
Постановка требований к интерфейсу программных средств	16 ч
Выбор нотации и программных средств планирования и оценки процесса разработки	1 ч
Требования к аппаратно-программному обеспечению серверного приложения	4 ч
Обоснование выбора технологий и средств разработки серверного приложения	1 ч
Разработка пользовательского интерфейса	20 ч
Оформление и сбор документации	20 ч
Дизайнер	40 ч
Разработка пользовательского интерфейса	40 ч
Разработчик базы данных	56 ч
Разработка единой концептуальной модели данных	16 ч
Разработка физической модели распределенной базы данных	40 ч
Тестировщик	64 ч
Планирование процесса контроля качества программного обеспечения	24 ч
Тестирование программных средств	24 ч
Оценка качества программного кода	16 ч
Интернет	
Анализ предметной области и постановка задачи на разработку серверного приложения	
Проектирование серверного приложения	
Разработка программного обеспечения серверного приложения	
Контроль качества программного обеспечения	
Компьютеры	
Итого	522 ч

Рисунок 6 – Отчет об использовании ресурсов

По данным отчета о ресурсах можно определить, что общая занятость трудовых ресурсов составляет 522 часа с 08.04.2024 по 13.06.2024 с учетом пятидневной рабочей недели. На основе отчета о бюджете проекта можно сделать вывод, что для разработки программных средств потоковой обработки событий необходимо 143680 рублей.

2 Проектирование серверного приложения

2.1 Моделирование функциональных возможностей приложения

На основе сформулированных задач в разделе 1.3.2 были выделены бизнес-процессы приложения для системы менеджмента хранилища нефтепродуктов. На рисунке 7 изображена диаграмма вариантов использования для данного приложения.

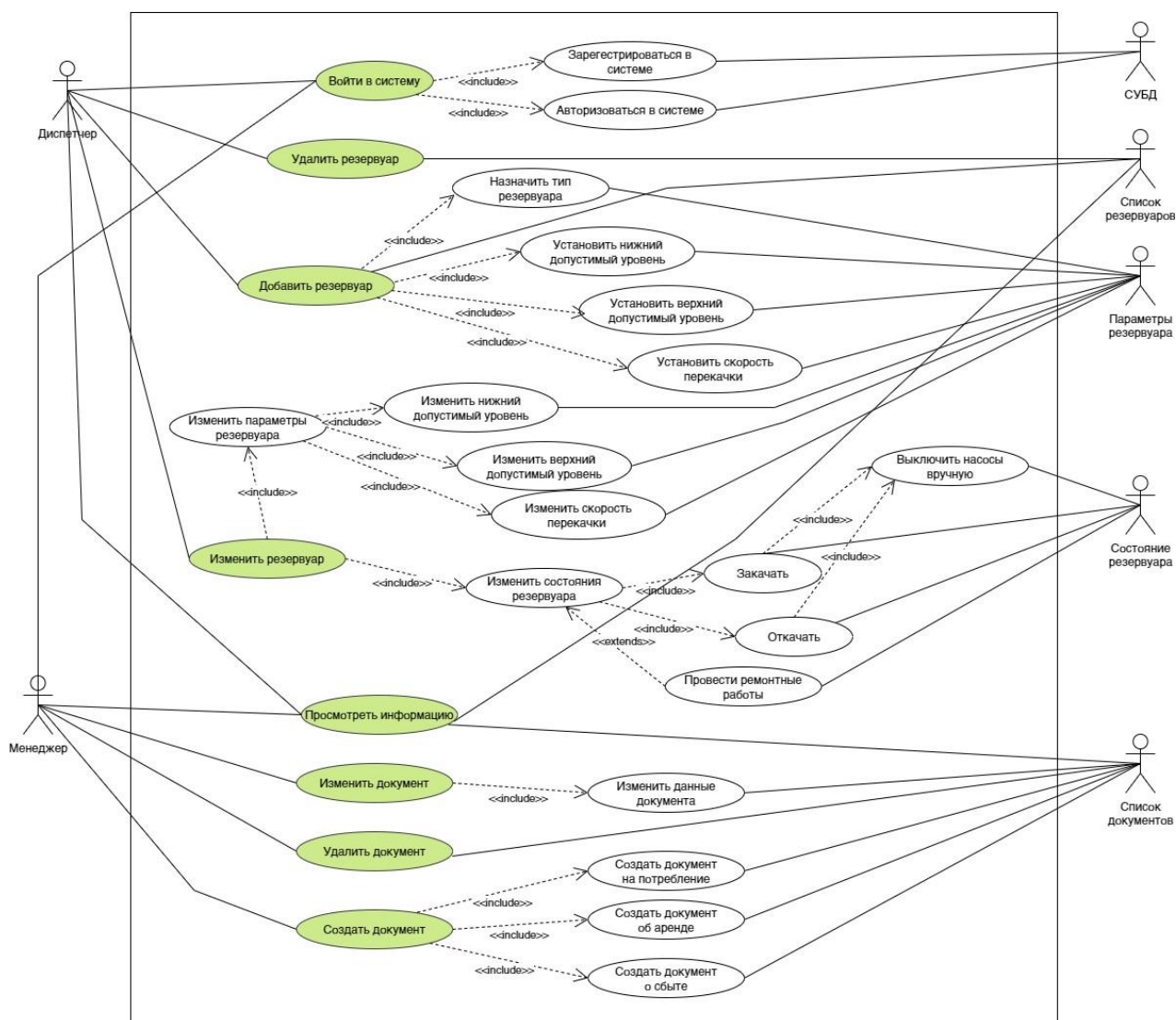


Рисунок 7 – Диаграмма вариантов использования

Подробное описание бизнес-процессов для информационной системы менеджмента хранилища нефтепродуктов, представленных на диаграмме вариантов использования, приведено в таблице 8.

Таблица 8 – Бизнес-процессы приложения для системы менеджмента
хранилища нефтепродуктов

№ п/ п	Бизнес- процесс	Испол нитель	Входные данные		Выходные данные		Ограничен ия
			Пост авщи к	Содержание	Потр ебите ль	Содержание	
1	2	3	4	5	6	7	8
1	Войти в систему	Диспетчер, менеджер	Диспетчер, менеджер	Логин и пароль	Диспетчер, менеджер	Доступ к функционалу системы и данным аккаунта	Пользователь должен быть зарегистрирован
2	Перейти в раздел с документами	Менеджер	Менеджер	Переход во вкладку с документами	Менеджер	Список добавленных документов (при наличии)	Пользователь должен быть авторизован
3	Просмотреть документ	Менеджер	Менеджер	Переход на страницу документа	Менеджер	Страница документа	Документ должен быть добавлен
4	Отредактировать данные	Менеджер	Менеджер	Исходные данные	Менеджер	Страница документа с измененными параметрами	Документ должен быть добавлен
5	Сформировать документ	Менеджер	Менеджер	Переход на страницу добавления документа	Менеджер	Страница добавляемого документа	
6	Удалить документ	Менеджер	Менеджер	Исходный документ	Менеджер	Документ удален	Документ должен быть добавлен до удаления
7	Создать документ об аренде	Менеджер	Менеджер	Переход на страницу добавления документа	Менеджер	Страница добавленного документа	Страница добавляемого документа открыта
8	Создать документ о сбыте	Менеджер	Менеджер	Переход на страницу добавления документа	Менеджер	Страница добавленного документа	Страница добавляемого документа открыта
9	Создать документ на потребление	Менеджер	Менеджер	Переход на страницу добавления документа	Менеджер	Страница добавленного документа	Страница добавляемого документа открыта

Продолжение таблицы 8

1	2	3	4	5	6	7	8
10	Получить список контрагентов в по выгоде сделок	Менеджер	Менеджер	Выбор критерия фильтрации	Менеджер	Список контрагентов и общая сумма сделок	Открыта страница анализа
11	Произвести аналитику	Менеджер	Менеджер	Выбор критерия фильтрации	Менеджер	Список контрагентов и общая сумма сделок	Открыта страница анализа
12	Получить список контрагентов в по объему продаж	Менеджер	Менеджер	Выбор критерия фильтрации	Менеджер	Список контрагентов и общая сумма сделок	Открыта страница анализа
13	Получить список контрагентов в по объему поставок	Менеджер	Менеджер	Выбор критерия фильтрации	Менеджер	Список контрагентов и общая сумма сделок	Открыта страница анализа
14	Получить список нефтепродуктов по объему продаж	Менеджер	Менеджер	Выбор критерия фильтрации	Менеджер	Список контрагентов и общая сумма сделок	Открыта страница анализа
15	Получить список нефтепродуктов по объему поставок	Менеджер	Менеджер	Выбор критерия фильтрации	Менеджер	Список контрагентов и общая сумма сделок	Открыта страница анализа
16	Перейти в раздел с резервуарами	Диспетчер	Диспетчер	Переход во вкладку с резервуарами	Диспетчер	Список добавленных резервуаров (при наличии)	Пользователь должен быть авторизован как диспетчер
17	Добавить резервуар	Диспетчер	Диспетчер	Переход на страницу добавления резервуара	Диспетчер	Страница добавленного резервуара	Пользователь должен быть авторизован как диспетчер

Продолжение таблицы 8

1	2	3	4	5	6	7	8
18	Перейти к резервуару	Диспетчер	Диспетчер	Переход на страницу резервуара	Диспетчер	Страница резервуара	Резервуар должен быть добавлен
19	Изменить параметры резервуара	Диспетчер	Диспетчер	Исходные параметры	Диспетчер	Страница резервуара с измененными параметрами	Резервуар должен быть добавлен
20	Удалить резервуар	Диспетчер	Диспетчер	Исходный резервуар	Диспетчер	Резервуар удален	Резервуар должен быть добавлен до удаления
21	Назначить тип резервуара	Диспетчер	Диспетчер	Страница добавления резервуара	Диспетчер	Тип резервуара назначен	
22	Установить нижний допустимый уровень	Диспетчер	Диспетчер	Страница добавления резервуара	Диспетчер	Нижний допустимый уровень установлен	
23	Установить верхний допустимый уровень	Диспетчер	Диспетчер	Страница добавления резервуара	Диспетчер	Верхний допустимый уровень установлен	
24	Установить скорость перекачки	Диспетчер	Диспетчер	Страница добавления резервуара	Диспетчер	Скорость перекачки установлена	
25	Выключить насосы вручную	Диспетчер	Диспетчер	Переход на страницу резервуара	Диспетчер	Насосы выключены	Резервуар должен быть активен
26	Провести ремонтные работы	Диспетчер	Диспетчер	Переход на страницу резервуара	Диспетчер	Резервуар исправен	Резервуар должен быть не активен
27	Закачать	Диспетчер	Диспетчер	Переход на страницу резервуара	Диспетчер	Резервуар активен	Резервуар должен быть не активен

Окончание таблицы 8

1	2	3	4	5	6	7	8
28	Откачать	Диспетчер	Диспетчер	Переход на страницу резервуара	Диспетчер	Резервуар активен	Резервуар должен быть не активен
29	Просмотреть журнал	Менеджер, диспетчер	Менеджер, диспетчер	Переход на страницу журнала	Менеджер, диспетчер	Список действий пользователя	Открыта страница журнала

Сценарий выполнения вариантов использования «Удалить резервуар» и «Изменить резервуар» представлены в таблице 9 – 10.

Таблица 9 – Сценарий выполнения варианта использования «Удалить резервуар»

Наименование:	Удалить резервуар
ID:	6
Краткое описание:	Пользователь удаляет резервуар
Действующие лица:	диспетчер
Предусловие:	Резервуар существует и выбран
Основной поток:	Пользователь удаляет резервуар
Постусловие:	Резервуар удален

Таблица 10 – Сценарий выполнения варианта использования «Изменить резервуар»

Наименование:	Изменить резервуар
ID:	5
Краткое описание:	Пользователь изменяет параметры резервуара или его состояние
Действующие лица:	диспетчер
Предусловие:	Пользователь выбрал резервуар для редактирования
Основной поток:	<ol style="list-style-type: none"> 1. Пользователь устанавливает нижний допустимый уровень 2. Пользователь устанавливает верхний допустимый уровень 3. Пользователь устанавливает скорость перекачки 4. Пользователь выключает насосы вручную Альтернативные потоки: 1.a. Пользователь начинает закачку 2.a. Пользователь начинает откачку
Постусловие:	Резервуар изменен

Полный перечень сценариев вариантов использования представлен в приложении Б. Полученные результаты являются исходными данными для реализации запросов на выборку, представлений, функций, хранимых процедур и триггеров, как компонентов серверного приложения.

2.2 Разработка единой концептуальной модели данных

Концептуальная модель – это модель, отображающая данные и связи предметной области с учетом их семантического содержания, независимо от представления в компьютере [12].

При разработке концептуальной модели данных важно правильно определить сущности, их атрибуты и взаимосвязи между ними. Концептуальная модель должна максимально точно отражать реальную предметную область и требования к данным, но при этом оставаться независимой от физической реализации базы данных. Одним из ключевых этапов построения концептуальной модели является нормализация данных. Это процесс организации данных в базе таким образом, чтобы свести к минимуму избыточность, устранить аномалии вставки, удаления и обновления, и в целом оптимизировать структуру данных.

В таблице 11 представлены сущности, сформированные на основании входных и выходных данных бизнес-процессов.

Таблица 11 – Описание сущностей предметной области

№ п/п	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
1	Пользователь	Список пользователей	Стержневая	2	Редко
2	Роль	Роли пользователей	Стержневая	2	Нет
3	Журнал состояний	Записи изменений документов	Кодовая	200	Очень часто
4	Нефтепродукт	Перечень нефтепродуктов	Ассоциативная	7	Редко
5	Документ	Список документов и их параметров	Стержневая	10	Часто
6	Вид документа	Виды документов	Кодовая	3	Очень редко

Продолжение таблицы 11

№ п/п	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
7	Договор о сбыте	Параметры договора о сбыте	Характеристическая, Стержневая	5	Часто
8	Договор на потребление	Параметры договора на потребление	Характеристическая, Стержневая	3	Часто
9	Договор об аренде	Параметры договора об аренде	Характеристическая, Стержневая	2	Часто
10	Контрагенты	Список сотрудничающих контрагентов	Кодовая	4	Часто
11	Причина	Список возможных состояний документа	Кодовая	20	Очень редко
12	Пользователь	Список пользователей	Стержневая	2	Редко
13	Роль	Роли пользователей	Стержневая	2	Нет
14	Журнал состояний	Записи изменений резервуаров	Кодовая	200	Очень часто
15	Нефтепродукт	Перечень нефтепродуктов	Ассоциативная	7	Редко
16	Резервуар	Параметры резервуара	Кодовая	10	20% в полгода
17	Насос	Параметры насоса	Стержневая	20	20% в полгода
18	Тип насоса	Тип насоса	Кодовая	2	Нет
19	Параметры резервуара	Список параметров резервуара	Кодовая	10	Часто
20	Контрагенты	Список сотрудничающих контрагентов	Кодовая	4	Часто
21	Причина	Список возможных состояний резервуара	Кодовая	20	Очень редко

Графическое представление концептуальной модели базы данных в виде диаграммы «сущность-связь» для предметной области приложения представлено на рисунке 8.

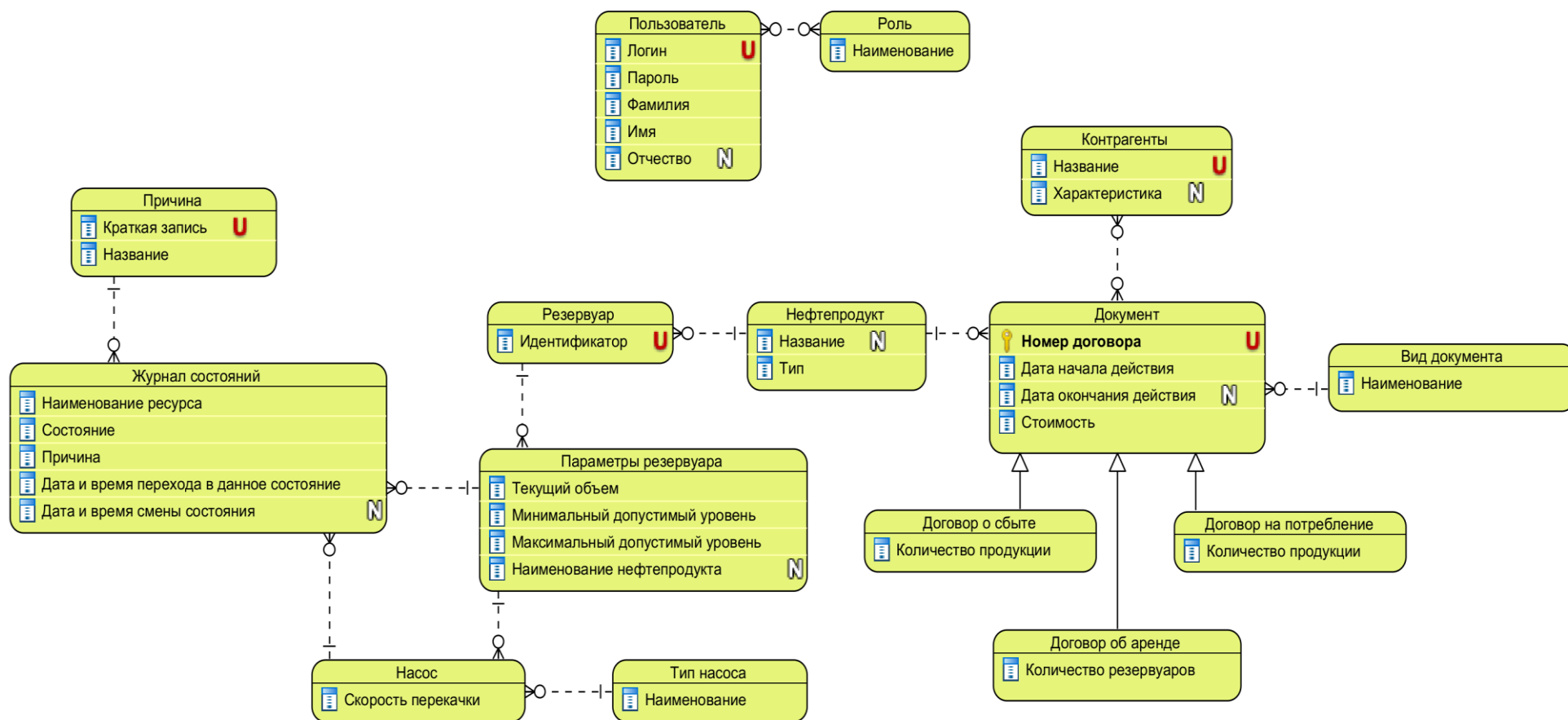


Рисунок 8 – Графическое представление концептуальной модели данных

На данной диаграмме представлена общая концептуальная модель совмещенных баз данных, чтобы их структура была простой и понятной для каждого пользователя.

2.3 Обобщенная архитектура системы менеджмента хранилища нефтепродуктов

Архитектура серверного приложения для системы менеджмента хранилища нефтепродуктов является микросервисной и содержит в себе три независимых модуля. Серверная часть разработана на языке C# при помощи фреймворка ASP.NET Core, для обеспечения обмена данными между клиентом и сервером используется gRPC. При этом, для хранения и управления данными используется SQL Server Management Studio.

Эта архитектура позволяет создавать гибкие и масштабируемые приложения благодаря следующим преимуществам:

- приложение разделено на несколько независимых служб, в результате отказ одной службы не приведет к отказу всего приложения;
- разработка, тестирование и развертывание каждого модуля происходит независимо друг от друга, что ускоряет процесс разработки и уменьшает риск ошибок;
- каждый микросервис может быть написан на любом языке программирования, иметь свою базу и инструментарий;
- gRPC как механизм обмена данными гарантирует высокую производительность и безопасную передачу данных между клиентом и сервером;
- эффективное управление данными;
- легкость обслуживания системы, путем обновления отдельных модулей, не прерывая работы всей системы.

Для визуализации полной архитектуры была разработана диаграмма развертывания, представленная на рисунке 9.

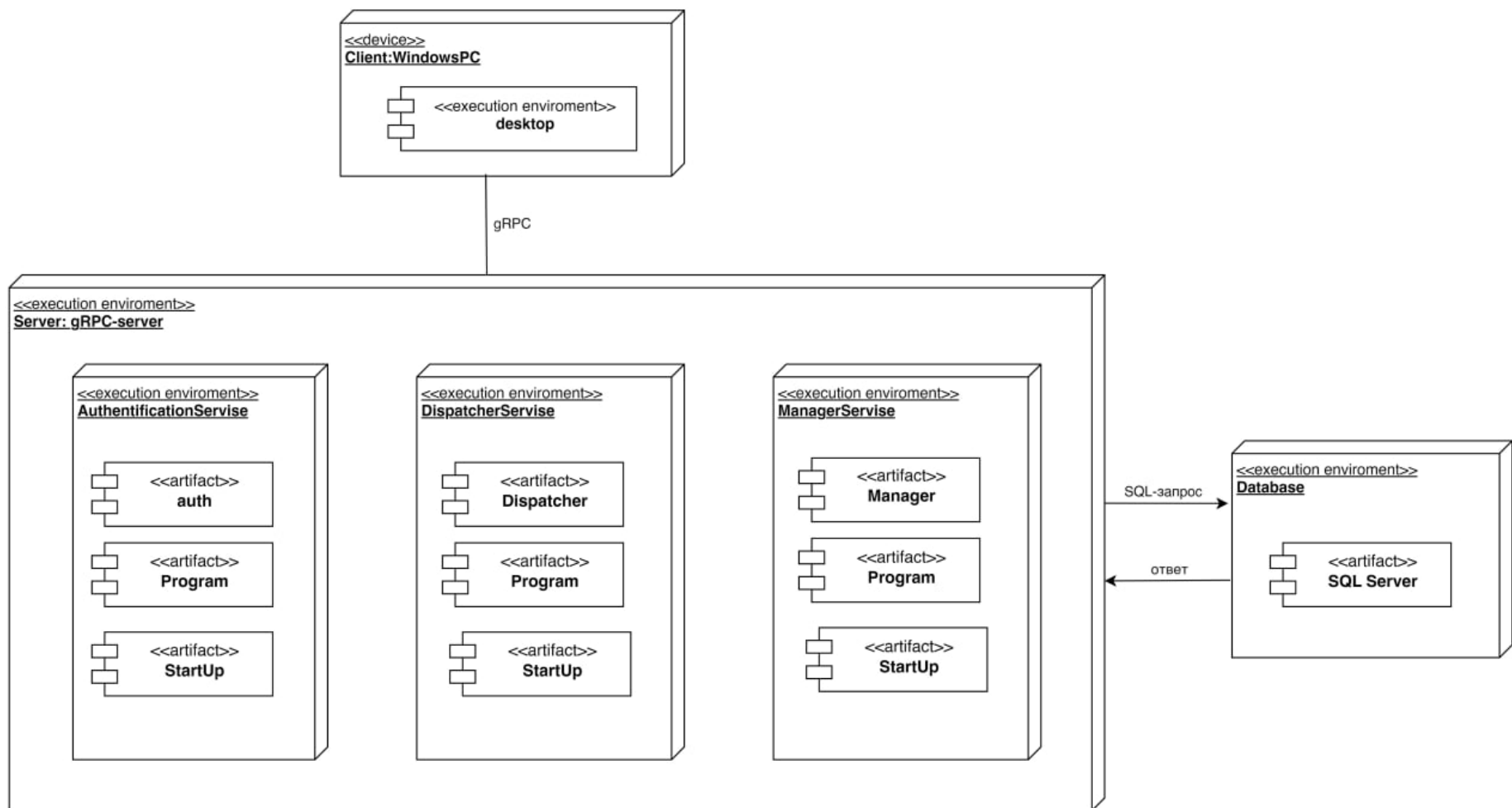


Рисунок 9 – Диаграмма развертывания серверного приложения

Диаграмма развертывания наглядно демонстрирует распределение основных компонентов приложения по физическим и виртуальным узлам. Пользователь взаимодействует с клиентским приложением и отправляет запросы на сервер через gRPC. Серверное приложение обрабатывает эти запросы, обращаясь к нужным модулям, при необходимости делая запросы к базе данных на SQL Server. После обработки запросов серверное приложение формирует ответ и отправляет его пользователю через gRPC.

Для наглядной демонстрации сценария взаимодействия клиентского и серверного приложения была спроектирована обобщенная диаграмма последовательности, представленная на рисунке 10.

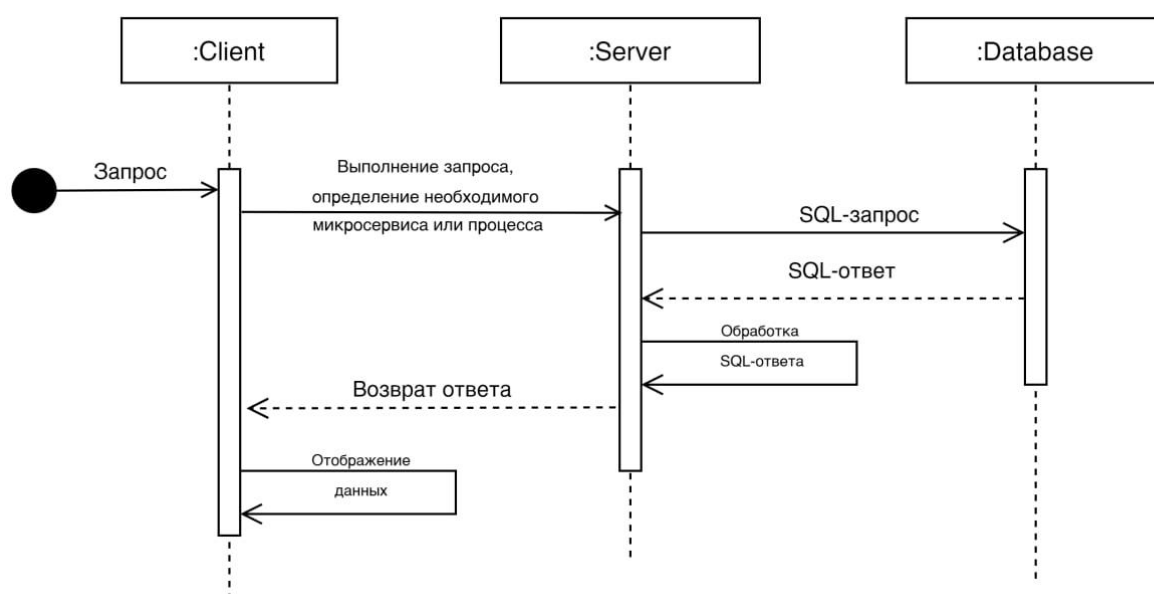


Рисунок 10 – Взаимодействие компонентов приложения

Данная диаграмма последовательности отражает поэтапную обработку полученной информации в запросе к серверу. Пользователь взаимодействует с пользовательским интерфейсом, в результате чего генерируется соответствующий запрос к серверу через установленное gRPC соединение. Сервер обрабатывает запрос, выполняет соответствующий к нему SQL запрос к базе данных и отправляет ответ обратно клиенту через gRPC. Затем клиент обрабатывает полученные данные и обновляет пользовательский интерфейс. Это визуализация помогает понять процесс работы программы на каждом этапе.

2.4 Требования к аппаратно-программному обеспечению серверного приложения

Исходя из выбранных технологий и средств разработки были выявлены основные требования к аппаратно-программному обеспечению. Наличие хороших параметров компьютера обеспечивает стабильную и быструю работу системы, которая важна для правильного и своевременного управления ресурсами предприятия.

Минимальные требования предназначены для обеспечения базовой функциональности и производительности, они подходят для малых и средних систем. Однако, если используется большое количество обрабатываемых данных, рекомендуется использовать оборудование и программное обеспечение, соответствующие рекомендуемым техническим требованиям. Минимальные и рекомендуемые параметры аппаратного обеспечения представлены в таблице 12.

Таблица 12 – Параметры аппаратного обеспечения

	Минимальные требования	Рекомендуемые требования
Среда выполнения	Windows 7	Windows 10
Тактовая частота процессора	Двухъядерный процессор с частотой 2.0 ГГц	Четырехъядерный процессор с частотой 2.5 ГГц
Сетевой адаптер	1 КБ/с	1 МБ/с
Оперативная память	1 МБ	2 МБ
Жесткий диск	200 КБ	1 МБ

Минимальные и рекомендуемые требования к программному обеспечению представлены в таблице 13.

Таблица 13 – Требования к программному обеспечению

	Минимальные требования	Рекомендуемые требования
Операционная система	Windows XP	Windows 10
СУБД	SQL Server Management Studio 2019	SQL Server Management Studio 2019 последней версии
Фреймворк для клиент-серверного взаимодействия	gRPC	gRPC последней версии
Поддержка языка программирования	C#	C#

Учет потребностей в аппаратно-программном обеспечении и четкое описание требований к серверному оборудованию является неотъемлемой частью процесса разработки программного обеспечения и важным условием его успешного внедрения и функционирования. При выборе аппаратуры и программного обеспечения необходимо учитывать текущие и прогнозируемые потребности системы, чтобы обеспечить ее надежную работу и возможность масштабирования в будущем.

2.5 Обоснование выбора технологий и средств разработки серверного приложения

Согласно функциональным и архитектурным требованиям, были определены необходимые технологии и средства разработки серверного приложения системы менеджмента для управления нефтехранилищем.

Выбор стека технологий определяется несколькими ключевыми критериями: специфичные потребности проекта, уровень знаний разработчиков, а также надежность и перспективы технологии. В нашем случае, выбор пал на SQL Server Management Studio 19 и Visual Studio 2019, как на среды, отвечающие этим требованиям.

Существует большое разнообразие программного обеспечения, предназначенного для разработки и тестирования приложений.

Notepad++ – это платформа для разработки веб-приложений. Приложение больше напоминает текстовый редактор с подсветкой синтаксиса, нежели полноценную среду разработки. Но именно поэтому к нему прибегают множество разработчиков. В основном Notepad++ используется для написания HTML-страниц на языках HTML, XML PHP и CSS, но также платформа имеет расширения для программирования на других языках.

IntelliJ IDEA – это интегрированная среда разработки, созданная для программирования на Java, Kotlin, JavaScript и других языках. Оно упрощает процесс кодирования, благодаря автодополнению кода, рефакторингу и многому другому, и снижает количество ошибок, поскольку предлагает мощные

инструменты отладки и проверки кода. Это платное средство разработки, а его бесплатная версия имеет ограниченный функционал.

Visual Studio 2019 – одна из самых мощных и гибких интегрированных сред разработки от корпорации Microsoft. Она поддерживает много языков программирования, позволяет создавать кроссплатформенные приложения, включает в себя подсветку синтаксиса, умную проверку кода и функции отладки, что облегчает процесс разработки. Visual Studio имеет мощную систему плагинов и расширений, что позволяет разработчикам значительно расширять функциональность, предлагает встроенные инструменты для работы в команде, включая систему контроля версий и инструменты для планирования и отслеживания проектов [13].

Для организации микросервисного взаимодействия была выбрана технология gRPC от компании Google. gRPC – это современный фреймворк для удаленного вызова процедуры, которое использует протокол HTTP/2 для передачи данных. Фреймворк предназначен для разработки масштабируемых и быстрых API, обеспечивает высокую производительность и позволяет разработчикам писать на любом языке программирования [14].

GraphQL так же служит механизмом обработки запросов клиентов. Он был разработан для оптимизации работы приложений-клиентов с базами данных. Однако, GraphQL требует большого количества кода для обработки запросов на сервере, и его поддержка сторонними библиотеками может быть ограничена по сравнению с gRPC [15].

REST (Representational State Transfer) – это способ создания API с помощью протокола HTTP/HTTPS. Он широко применяется в веб-разработке из-за своей простоты и надежности. Однако, в отличие от gRPC, REST не поддерживает передачу бинарных данных, и поэтому может быть неэффективен для передачи больших объемов данных [16].

На сегодняшний день существует множество систем для создания баз данных. Несмотря на то, что все они выполняют одну и ту же задачу, сам процесс выполнения этой задачи варьируется в широких пределах.

PostgreSQL – это инструмент для создания объектно-реляционных СУБД для хранения и управления базами данных на SQL и других языках программирования. Поддерживает инструменты всех операционных систем, обладает системой обобщенного поиска и повышенным уровнем безопасности.

MySQL Workbench – программа для работы с MySQL. Это решение предлагает довольно широкий функционал, включая шаблоны для обработки запросов, модуль для визуального проектирования и редактор SQL-запросов [17].

SQL Server Management Studio 19 (SSMS) – это интегрированная среда для управления любой инфраструктурой SQL. Он предлагает богатый функционал, особенно если речь идет о визуализации данных, многофункциональные редакторы скриптов для доступа к SQL Server для разработчиков и администраторов баз данных. Несмотря на свой обширный инструментарий, SSMS полностью бесплатен для использования. Кроме того, поддерживает интеграцию с другими продуктами Microsoft, такими как Azure и Visual Studio, что упрощает работу для команд разработчиков, уже использующих эти технологии [18].

Построение диаграмм осуществляется с помощью языка UML 2.0 в среде draw.io. Это бесплатный онлайн-сервис, позволяющий создавать диаграммы, схемы и прочие представления данных: диаграммы классов, последовательности, вариантов использования, развертывания и т. д. [19]

Так же для создания концептуальных, логических и физических моделей баз данных использовалась Visual Paradigm. Это пакет инструментов моделирования бизнес-процессов, ПО и баз данных. Visual Paradigm позволяет автоматически генерировать коды Java, C++, .NET, скрипты баз данных, формировать и настраивать отчеты в различных форматах.

Таким образом, выбор этих технологий и средств разработки обеспечивает надежность, производительность и масштабируемость серверного приложения, а также удобство процесса разработки.

3 Разработка программного обеспечения серверного приложения

3.1 Разработка архитектуры серверного приложения

Для разработки серверного приложения была выбрана микросервисная архитектура. Приложение разделено на отдельные сервисы, каждый из которых отвечает за конкретную часть функциональности.

Одним из главных этапов в проектировании микросервисного приложения является выбор типа построения API. Есть несколько основных типов построения API: SOA, RPC, gRPC, WebSocketAPI, REST. Для коммуникации между микросервисами выбран протокол gRPC, обеспечивающий высокую производительность и возможность строгой типизации.

Для реализации обработки gRPC запросов в программе существуют специальные интерфейсы с описанной спецификой запроса. Пример обработки запросов для авторизации пользователя показан в листинге 1.

Листинг 1 – Обработка запросов для авторизации пользователей

```
public override async Task<AuthorizationResult>
UserAuthorization(UserLogPass userLogPass, ServerCallContext context)
{
    using var sqlConnection = await GetConnection();
    if (await UserExist(sqlConnection, userLogPass.Login) is not
true) {
        return new()
        { Result = AuthorizationResultType.AuthUserNotExists }; }

    if (await PasswordIsCorrect(sqlConnection, userLogPass) is not
true)
    {
        return new() { Result =
AuthorizationResultType.AuthIncorrectPass }; }

    return new()
    {
        Result = AuthorizationResultType.AuthSuccess
    };
};
```

Проанализировав требования к разрабатываемому приложению, были определены основные компоненты, из которых состоят программные средства. Исходя из этих требований были спроектированы три основных микросервиса и три базы данных.

Структура программных средств представлены на рисунке 11.

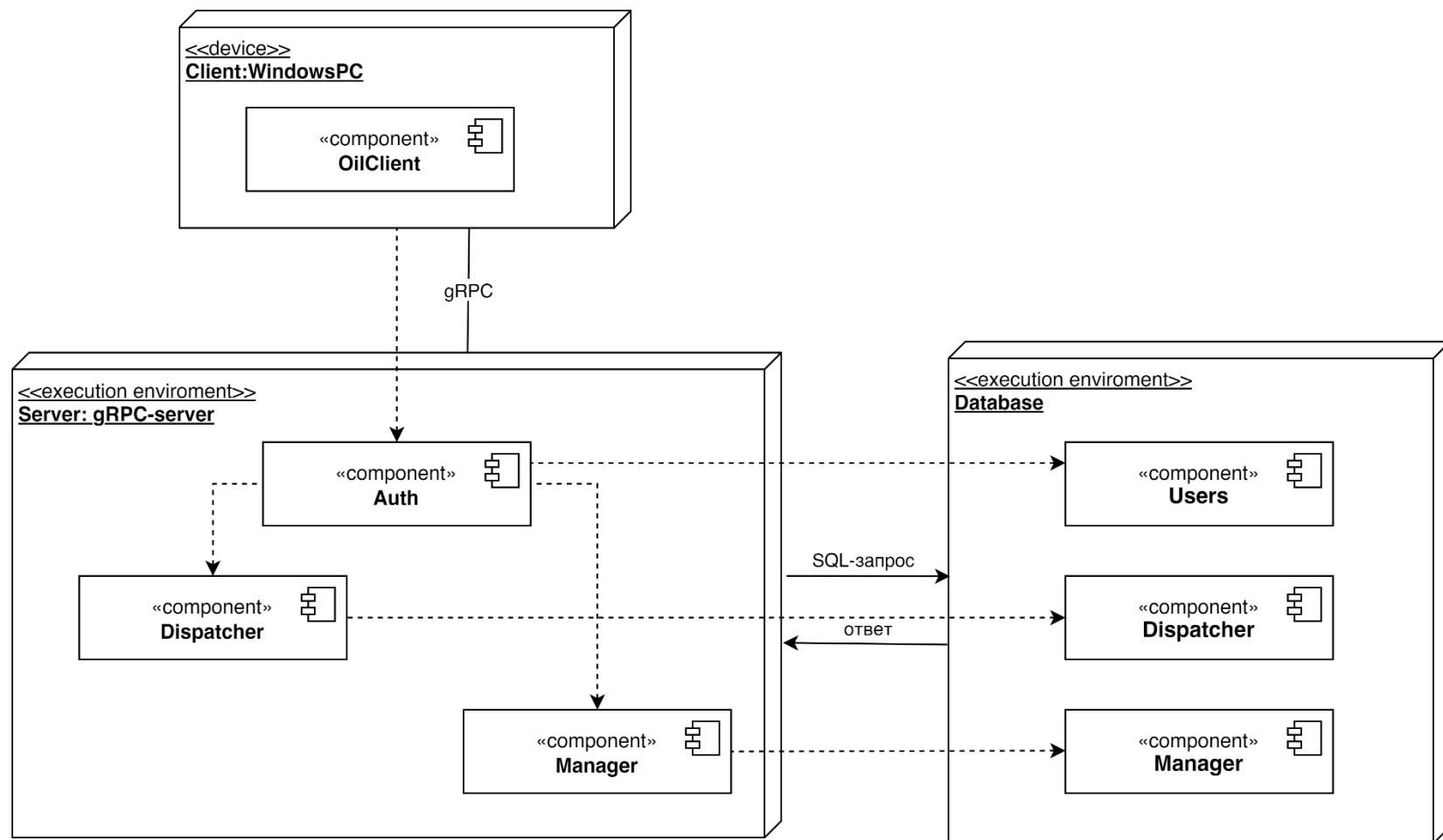


Рисунок 11 – Диаграмма основных компонентов приложения

Описание основных действий каждого компонента диаграммы представлено в таблице 14.

Таблица 14 – Описание компонентов приложения

№	Название	Описание
1	OilClient	Пользовательский интерфейс, через который осуществляется взаимодействие с системой.
2	Auth	Микросервис, отвечающий за управление пользователями и их правами доступа.
3	Dispatcher	Микросервис, отвечающий за управление хранилищем нефтепродуктов.
4	Manager	Микросервис, отвечающий за управление документооборотом.
5	Users	База данных, используемая микросервисом Auth, содержит данные о пользователях.
6	Data base Dispatcher	База данных, используемая микросервисом Dispatcher, содержит данные о нефтепродуктах.
7	Data base Manager	База данных, используемая микросервисом Manager, содержит данные о документах.

OilClient посылает запросы на API gRPC, который проверяет аутентификацию и авторизацию, а затем маршрутизирует запросы к соответствующим микросервисам (Dispatcher, Manager). Микросервисы взаимодействуют с соответствующими базами данных для получения или обновления данных. Пример структуры проекта одного из микросервисов представлен на рисунке 12.

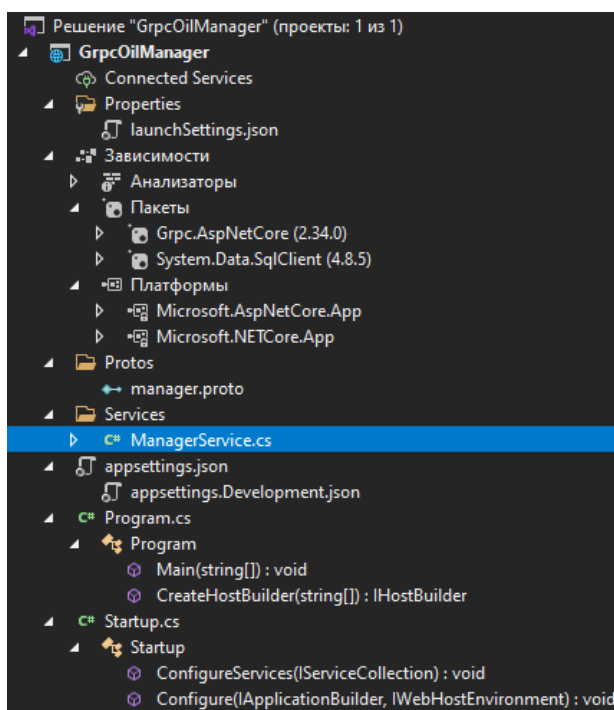


Рисунок 12 – Структура проекта для микросервиса Диспетчера

В проекте представлены следующие микросервисы:

- GrpcUtils – класс вспомогательных методов для работы с gRPC;
- GrpcDispatcher – класс со вспомогательными методами для работы с gRPC API диспетчера;
- GrpcManager – класс со вспомогательными методами для работы с gRPC API менеджера;
- GrpcAuthentication – класс со вспомогательными методами для работы с gRPC API аутентификацией;
- GrpcLogs – класс со вспомогательными методами для работы с gRPC API журнала.

В ходе разработки информационной системы было реализовано серверное приложение, код которого приведен в приложении Е.

3.2 Разработка физической модели распределенной базы данных

На основании концептуальной модели была разработана физическая модель базы данных. Она учитывает детали хранения данных, такие как типы данных, индексы, ключи, физические структуры таблиц и т. д. Физическая модель может быть представлена в виде SQL-скриптов, диаграмм хранения данных или других технических спецификаций.

Проектирование физической модели базы данных позволяет выделить основные методы доступа микросервисов к базе данных, а также особенности использования базы данных в рамках информационной системы.

Физические модели для баз данных Auth, Dispathter, Manager показаны на рисунках 13 – 14. Для базы данных Manager приведена в приложении В.

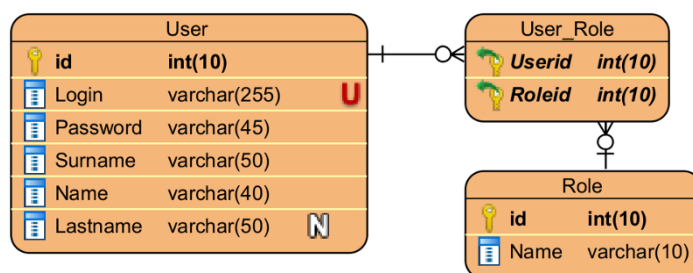


Рисунок 13 – Физическая модель данных Auth

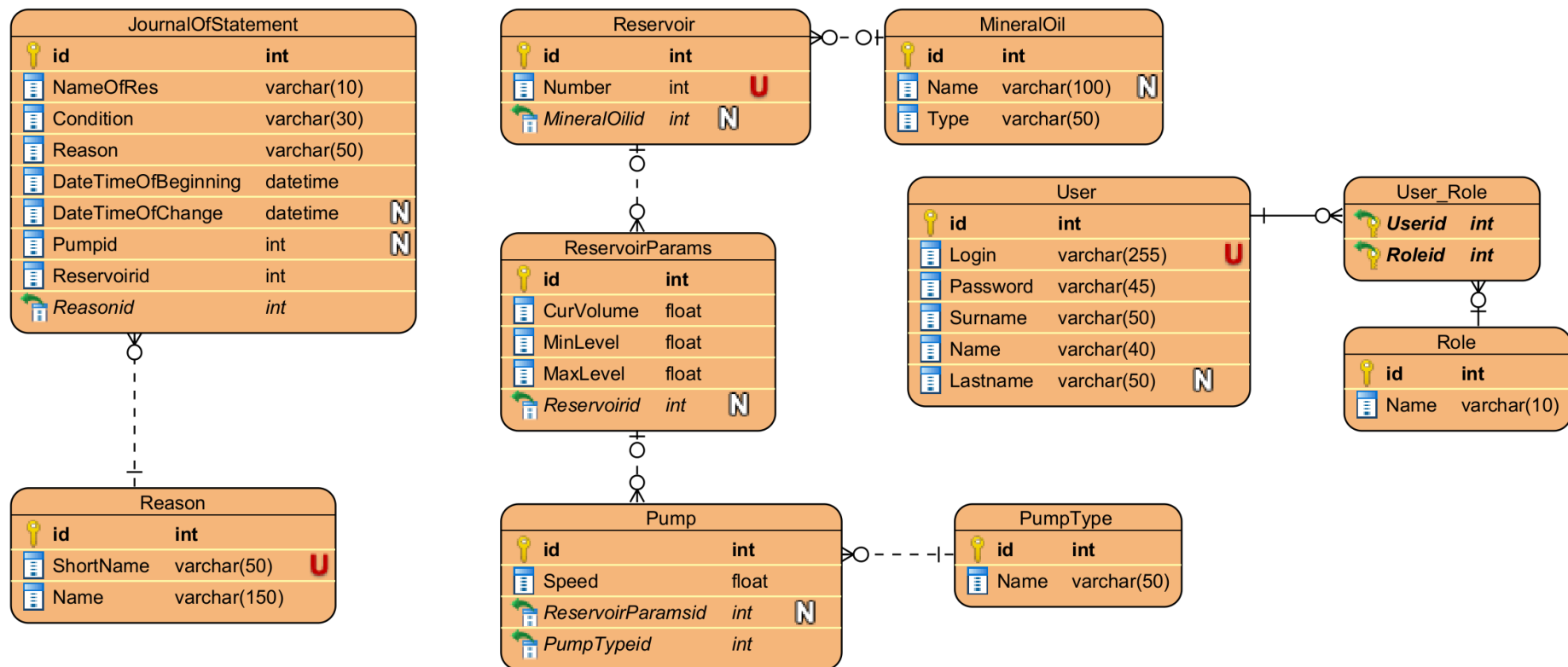


Рисунок 14 – Физическая модель данных Dispathер

Согласно физической модели, была создана база данных в MS SQL Server. Спецификации атрибутов для таблиц User, Role, User_Role представлены в таблицах 15 – 17. Полная спецификация атрибутов приведена в приложении В.

Таблица 15 – Спецификации строк таблицы User

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	id	Идентификатор	int		PK	Да	0 - 9
2	Login	Логин	varchar	255	AK	Да	A - я0 - 9
3	Password	Пароль	varchar	45		Да	A - яA - z0 - 9
4	Surname	Фамилия	varchar	50		Да	A - я
5	Name	Имя	varchar	40		Да	A - я
6	Lastname	Отчество	varchar	50		Нет	A - я

Таблица 16 – Спецификации строк таблицы Role

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	id	Идентификатор	int		PK	Да	0 - 9
2	Name	Вид роли	varchar	10		Да	A - я

Таблица 17 – Спецификации строк таблицы User_Role

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	Userid	Идентификатор пользователя	int		FK	Да	0 - 9
2	Roleid	Идентификатор роли	int		FK	Да	0 - 9

Данная физическая модель базы данных позволяет наглядно отразить основные сущности для работы микросервисов, а также определить специфику хранения и управления данными информационной системы.

3.3 Разработка программного обеспечения микросервиса

«Авторизация»

Микросервис «Авторизация» предназначен для регистрации пользователь в системе и их авторизации в соответствии с определенной ролью. Микросервис реализует следующий функционал:

- регистрация новых пользователей в системе;
- валидация данных пользователя;

- проверка учетных данных пользователя для предоставления доступа к системе;
- назначение ролей пользователям;
- проверка прав доступа на основе ролей при выполнении действий в системе.

Для микросервиса «Авторизация» была составлена диаграмма классов, представленная на рисунке 15.

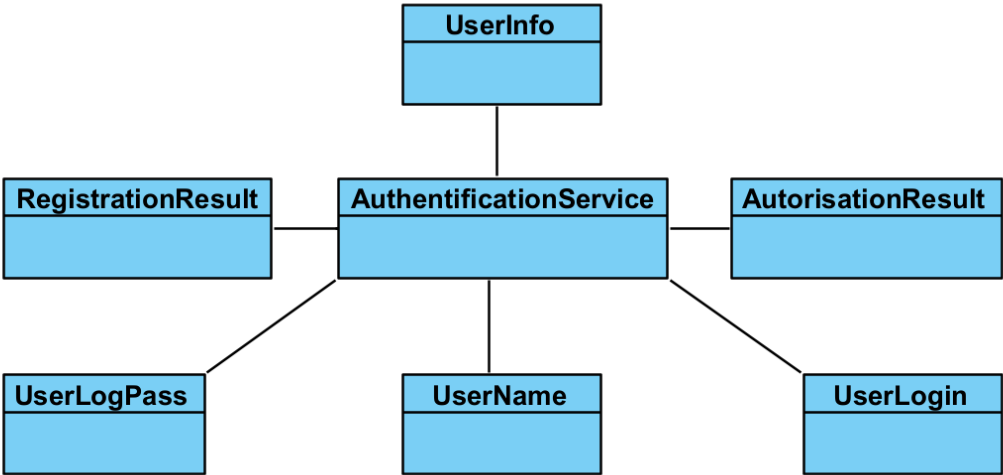


Рисунок 15 – Диаграмма классов микросервиса «Авторизация»

В таблице 18 представлено описание основных API методов микросервиса.

Таблица 18 – Основные методы API управления пользователем

URL запроса	Http метод	Описание
/auth.Authentication/UserRegistration	POST	Регистрация пользователя
/auth.Authentication/UserAuthorization	POST	Авторизация пользователя
/auth.Authentication/GetUserRole	POST	Получение роли пользователя
/auth.Authentication/GetUserName	POST	Получение имени и фамилии пользователя

Этот микросервис обеспечивает регистрацию и аутентификацию пользователей, управление ролями и правами доступа, а также содержит механизмы для обеспечения безопасности, масштабируемости и отказоустойчивости.

3.4 Разработка программного обеспечения микросервиса «Диспетчер»

Микросервис «Диспетчер» предназначен для управления и координации работы хранилища нефтепродуктов. Микросервис реализует следующий функционал:

- распределение нефтепродуктов по резервуарам;
- отображение состояний емкостей;
- добавление/удаление резервуаров;
- закачка/откачка нефтепродуктов;
- редактирование параметров резервуара;
- обеспечение логирования операций;
- анализ работы.

Для микросервиса «Авторизация» была составлена диаграмма классов, представленная на рисунке 16.

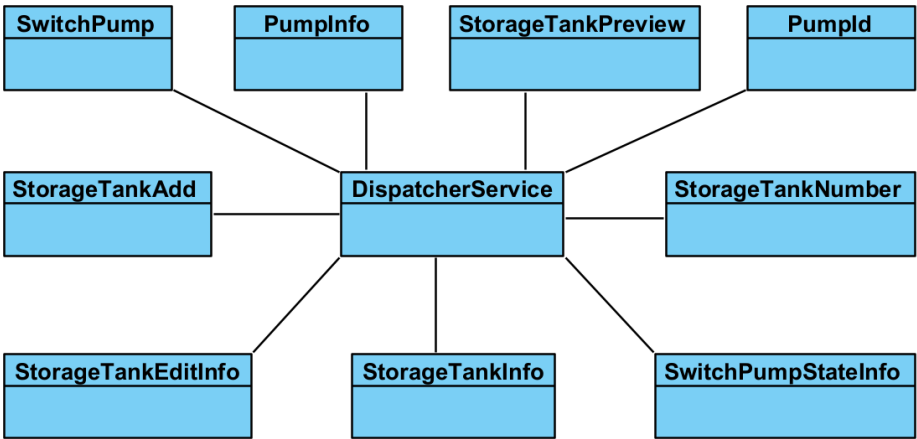


Рисунок 16 – Диаграмма классов микросервиса «Диспетчер»

В таблице 19 описаны основные API методы микросервиса.

Таблица 19 – Основные методы API Диспетчера

URL запроса	Http метод	Описание
/ro_dispatcher.RODispatcher/GetROListOfTanksStream	POST	Получение списка резервуаров
/ro_dispatcher.RODispatcher/GetROStorageTankInfo	POST	Получение информации о резервуаре

Продолжение таблицы 19

URL запроса	Http метод	Описание
/ro_dispatcher.RODispatcher/GetROInputPumpInfo	POST	Получение информации о насосе заправки
/ro_dispatcher.RODispatcher/GetROOutputPumpInfo	POST	Получение информации о выпускном насосе

Для микросервиса были разработаны запросы к базе данных для реализации аналитических бизнес-процессов. Спецификации запросов приведены в таблице 20. Скрипты запросов приведены в приложении Ж.

Таблица 20 – Спецификации запросов к базе данных

№ п/п	Содержание	Используемые таблицы	Параметры группировки	Параметры выбора данных
1	Рейтинг резервуаров по типу нефтепродуктов	MineralOil, Reservuar	MineralOil.Type	Count
2	Сортировка резервуаров по их объему	Reservuar, ReservoirParams	Reservoir.Number	MaxLevel

В таблице 21 представлены спецификации функций и хранимых процедур.

Таблица 21 – Спецификации функций и хранимых процедур учетных бизнес-процессов

№ п/п	Имя функции (процедуры)	Назначение	Описание логики работы	Параметры и возвращаемые данные
1	GetAllRes	Получение списка резервуаров	Процедура собирает данные по нескольким таблицам и возвращает перечень документов с их статусом и процентом заполнения	Процедура возвращает результат выборки (SELECT) по номеру резервуара, состоянию и проценту заполнения.
2	InsertRes	Добавление одного резервуара	Процедура получает номер резервуара, название и тип нефтепродукта и добавляет в таблицу Reservoir запись по номеру и типу	Процедура принимает 3 параметра: номер резервуара, название и тип нефтепродукта
3	DeleteRes	Удаление резервуара	Процедура удаляет запись из таблицы Reservoir по полученному номеру	Процедура принимает номер резервуара
4	GetInfoAbout Res	Получение списка параметров резервуара	Процедура собирает данные по нескольким таблицам и возвращает параметры резервуара	Процедура принимает номер документа и возвращает результат выборки (SELECT) по номеру резервуара

Продолжение таблицы 21

№ п/п	Имя функции (процедуры)	Назначение	Описание логики работы	Параметры и возвращаемые данные
5	GetInfoAbout PumpIn	Получение списка параметров насоса закачки	Процедура собирает данные по нескольким таблицам и возвращает параметры насоса закачки	Процедура принимает номер документа и возвращает результат выборки (SELECT) по номеру резервуара
6	GetInfoAbout PumpOut	Получение списка параметров выпускного насоса	Процедура собирает данные по нескольким таблицам и возвращает параметры выпускного насоса	Процедура принимает номер документа и возвращает результат выборки (SELECT) по номеру резервуара
5	UpdateParams	Обновление параметров резервуара	Сначала процедура добавляет запись в журнал об изменении резервуара и насосов, затем заносит изменения в 3 таблицы по полученным параметрам	Процедура принимает 6 параметров: номер резервуара, название нефтепродукта, минимальный и максимальный уровень резервуара, скорости насосов
6	AddCollRes	Добавление коллекции резервуаров	Процедура при помощи цикла получает идентификатор нефтепродукта и добавляет с номером в таблицу	Процедура принимает в качестве параметра коллекцию данных, в которые входят номер резервуара, название и тип нефтепродукта

Для хранимой процедуры, обрабатывающей коллекции информационных объектов на рисунке 17 приведена диаграмма деятельности.

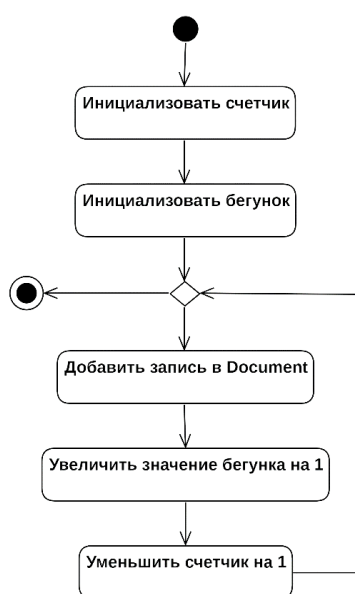


Рисунок 17 – Диаграмма деятельности

Так же разработаны спецификации триггера AFTER для реализации автоматически исполняемого учетного бизнес-процесса, триггера INSTEAD OF для модификации данных при выполнении операций UPDATE для необновляемого представления. Спецификации приведены в таблице 22.

Таблица 22 – Спецификация триггеров

№ п/п	Назначение триггера	Вид триггера	Событие триггера	Таблица триггера	Описание работы триггера
1	Добавление и обновление данных в связанных таблицах при вставке нового резервуара и добавление записи о добавлении в журнал	AFTER	insert	Reservoir	Триггер создает запись в журнале о добавлении нового резервуара, затем устанавливает значения по умолчанию в связанные таблицы
2	Изменение резервуара на удаленный и добавление записи об удалении в журнал	INSTEAD OF	delete	Reservoir	Триггер создает запись в журнале об удалении резервуара, а в списке документов устанавливает статус удален
3	Изменение состояния насоса на включенный или выключенный, добавление записи об изменении в журнал	INSTEAD OF	update	ReservoirParams	Триггер проверяет, какой насос включают или выключают, просматривает по журналу последнее состояние насоса и меняет на противоположное, согласно ограничениям, добавляет записи в журнал и обновляет записи по состоянию в 2 таблицах

Полный текст скриптов триггеров представляются в приложении К.

3.5 Разработка программного обеспечения микросервиса

«Менеджер»

Микросервис «Менеджер» предназначен для управления документами нефтехранилища. Микросервис реализует следующий функционал:

- добавление/удаление документов;
- формирование документов;

- редактирование данных документов, отслеживание их актуальности;
- анализ деятельность компании;
- обеспечение логирования операций.

Для микросервиса «Менеджер» была составлена диаграмма классов, представленная на рисунке 18.

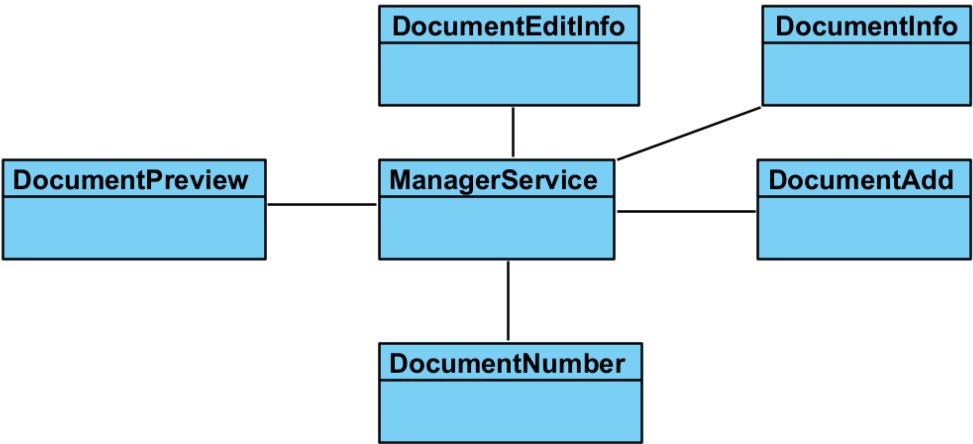


Рисунок 18 – Диаграмма классов микросервиса Менеджера

В таблице 23 представлено описание основных API методов микросервиса.

Таблица 23 – Основные методы API Менеджера

URL запроса	Http метод	Описание
/manager.Manager/GetListOfDocsStream	POST	Получение списка документов
/manager.Manager/AddDocument	POST	Добавление документа
/manager.Manager/DellDocument	POST	Удаление документа
/manager.Manager/GetDocumentInfo	POST	Получение информации о документе
/manager.Manager/EditDocument	POST	Изменение документа

Для микросервиса были разработаны запросы к базе данных для реализации аналитических бизнес-процессов. Спецификации запросов приведены в таблице 24.

Таблица 24 – Спецификации запросов к базе данных

№ п/п	Содержание	Используемые таблицы	Параметры группировки	Параметры выбора данных
1	Рейтинг контрагентов по выгоде сделок за период времени	Document, Coontractor_Document, Coontractor	Coontractor.Name	Cost
2	Рейтинг контрагентов по объему продаж	Document, Coontractor_Document, Coontractor, SalesAgreement	Coontractor.Name	Amount
3	Рейтинг контрагентов по объему поставок	Document, Coontractor_Document, Coontractor, ConsumptionAgreement	Coontractor.Name	Amount
4	Рейтинг продаж нефтепродуктов	Document, MineralOil, SalesAgreement	MineralOil.Name	Cost
5	Рейтинг поставок нефтепродуктов	Document, MineralOil, LeaseAgreement	MineralOil.Name	Cost

Скрипты запросов и представлений представляются в приложении Ж.

Далее представлены спецификации функций и хранимых процедур в таблице 25.

Таблица 25 – Спецификации функций и хранимых процедур учетных бизнес-процессов

№ п/п	Имя функции (процедуры)	Назначение	Описание логики работы	Параметры и возвращаемые данные
1	GetAllDocs	Получение списка документов	Процедура собирает данные по нескольким таблицам и возвращает перечень документов с их статусом	Процедура возвращает результат выборки (SELECT) по номеру документа, его типу и состоянию.
2	InsertDoc	Добавление одного документа	Процедура получает идентификатор по типу документа и вставляет полученному значения в таблицу Document	Процедура принимает 2 параметра: номер документа и его тип
3	DeleteDoc	Удаление документа	Процедура удаляет запись из таблицы Document по полученному номеру	Процедура принимает номер документа
4	GetInfoAboutDocs	Получение списка параметров документа	В процедуре проверяется, какой тип документа обрабатывается, и возвращаются значения, в соответствии поступившему номеру	Процедура принимает номер документа и возвращает результат выборки по номеру документа, типу, сроку действия, стоимости, объему сделки

Продолжение таблицы 25

№ п/п	Имя функции (процедуры)	Назначение	Описание логики работы	Параметры и возвращаемые данные
5	UpdateParams	Обновление данных документа	Сначала процедура добавляет запись в журнал об изменении документа, затем заносит изменения в 3 таблицы по полученным параметрам	Процедура принимает 7 параметров: номер документа, срок действия, стоимость и объем сделки, имя контрагента и название нефтепродукта
6	AddCollDocs	Добавление коллекции документов	Процедура при помощи цикла получает идентификатор типа документа и добавляет с номером в таблицу	Процедура принимает в качестве параметра коллекцию данных, в которые входят номер документа и его тип
1	GetAllDocs	Получение списка документов	Процедура собирает данные по нескольким таблицам и возвращает перечень документов с их статусом	Процедура возвращает результат выборки (SELECT) по номеру документа, его типу и состоянию.
2	InsertDoc	Добавление одного документа	Процедура получает идентификатор по типу документа и вставляет полученному значения и значения по умолчанию в таблицу Document	Процедура принимает 2 параметра: номер документа и его тип

Диаграмма деятельности для хранимой процедуры, обрабатывающей коллекции информационных объектов приведена на рисунке 19.



Рисунок 19 – Диаграмма деятельности

Полный текст скриптов функций и хранимых процедур представляются в приложении И.

Так же разработаны спецификации триггера AFTER для реализации автоматически исполняемого учетного бизнес-процесса, триггера INSTEAD OF для модификации данных при выполнении операций UPDATE для не обновляемого представления. Спецификации приведены в таблице 26.

Таблица 26 – Спецификация триггеров

№ п/п	Назначение триггера	Вид триггера	Событие Триггера	Таблица триггера	Описание работы триггера
1	Добавление и обновление данных в связанных таблицах при вставке нового документа и добавление записи о дообавлении в журнал	AFTER	insert	Document	Триггер создает запись в журнале о добавлении нового документа, затем устанавливает значения по умолчанию в связанные таблицы
2	Изменение статуса документа на удаленный и добавление записи об удалении в журнал	INSTEAD OF	delete	Document	Триггер создает запись в журнале об удалении документа, а в списке документов устанавливает статус удален

Полный текст скриптов триггеров представляются в **Приложении К**.

4 Контроль качества программного обеспечения

4.1 Планирование процесса контроля качества программного обеспечения

4.1.1 Мероприятия по контролю качества программного обеспечения

Контроль качества ПО – это процесс, определяющий соответствие показателей качества установленным требованиям. Как правило, производится на финальном этапе разработки с целью выявления ошибок, установления их причин и дальнейшего их устранения и предотвращения новых дефектов. Этот процесс предназначен для обеспечения надлежащего уровня качества конечного продукта, а также для контроля работы команды разработчиков [20].

Контроль качества программного обеспечения достигается, в первую очередь, путем тестирования продукции. Для этого проводится комплексная проверка программного обеспечения, обеспечивается достаточное тестовое покрытие, чтобы каждая часть кода была проверена. Иначе есть риск упущения критических ошибок, которые могут подорвать стабильность функционирования продукта в дальнейшем.

Чтобы предотвратить разрастание кода, которое замедляет и даже останавливает разработку, существует анализ метрик кода. Ранняя диагностика проблем позволит оптимизировать код, избежать ненужных сложностей и сохранить плавность процесса разработки.

Ещё один важный момент – производительность. Одной из важных показателей качества программного обеспечения является производительность. Чтобы отследить связанные с ней ошибки, помогут различные методики тестирования, которые могут предотвращать потенциальные «узкие места» в работе программного обеспечения.

Для контроля качества разработанного программного продукта необходимо провести несколько видов тестирования и оценить качество

продукта с помощью специальных метрик.

4.1.2 Обоснование выбора модели тестирования

Методы тестирования можно разделить на ручное и автоматическое тестирование. В режиме ручного тестирования человек осуществляет проверку функционала приложения самостоятельно либо путем взаимодействия с программным обеспечением [21].

Различают следующие методы тестирования:

- модульное тестирование;
- интеграционное тестирование;
- функциональное тестирование;
- тестирование производительности;
- нагрузочное тестирование.

Модульное тестирование – это методика тестирования программного обеспечения, направленная на изолированную проверку отдельных компонентов и модулей. Автоматизация модульных тестов является эффективной практикой. Она минимизирует затраты и позволяет проводить тестирование с высокой скоростью, используя серверы непрерывной интеграции.

Интеграционное тестирование представляет собой методику тестирования программного обеспечения, направленную на проверку корректности взаимодействия между различными модулями и сервисами приложения. Основное применение данного метода заключается в тестировании взаимодействия между клиентской и серверной частями приложения посредством API.

Функциональное тестирование – это методика тестирования программного обеспечения, основанная на проверке соответствия реального поведения продукта заданным функциональным требованиям. Тестирование проводится путем сравнения ожидаемых и фактических результатов выполнения тестовых кейсов.

Тестирование производительности представляет собой методику, направленную на оценку надежности, скорости и отзывчивости программного обеспечения. Данный метод позволяет определить соответствие приложения требованиям к производительности, выявить «узкие места», оценить стабильность в периоды пиковой нагрузки и т. д.

Нагрузочное тестирование – это тип тестирования программного обеспечения, который имитирует нагрузку на систему, создаваемую реальными пользователями, чтобы проверить её производительность и стабильность под высоким давлением.

В качестве метода ручного тестирования программного обеспечения было выбрано функциональное тестирование. Данный выбор обусловлен следующими факторами:

- тестирование основано на анализе спецификаций функциональности компонента или системы в целом, рассматривая заранее указанное поведение;
- позволяет сфокусироваться на ключевых функциях и не упустить наиболее важные;
- имитирует реальное использование системы.

Кроме того, был выбран вид автоматического тестирования – нагрузочное тестирование, поскольку оно позволяет выявить узкие места в базе данных, которые могут привести к проблемам производительности при реальной нагрузке. Также, данный метод тестирования помогает проверить, как база данных справляется с увеличением объема данных и количества пользователей.

Тестирование разработанного приложения осуществляется на примере системы, описанной в приложении Г.

4.1.3 Обоснование выбора метрик программного кода

Метрики программного кода представляют собой инструмент для оценки качественных характеристик создаваемого или существующего

программного обеспечения. Они позволяют охарактеризовать программное обеспечение, предсказать объем работ, провести количественную оценку качества проектных решений, разрабатываемых систем и их компонентов, описать сложность и надежность. Результаты измерения программного кода могут использоваться для корректировки рабочего процесса программистов, так как метрики применяются для оценки продуктивности разработки программного обеспечения.

Различают несколько метрик, которые можно использовать для оценки программного кода с разных точек зрения, с разным уровнем усилий и времени, и используя различные методы для расчета результатов. Примеры метрик программного кода:

- количество строк кода;
- комплексный показатель качества кода;
- метрика Холстеда;
- метрика Чепина.

Двумя основными метриками являются метрика Холстеда и метрика Чепина. Они и были взяты для оценки качества программного кода.

Метрика Холстеда основана на анализе операндов, операторов и количества строк программного кода. С ее помощью можно рассчитать показатели программы, определить объем работ и возможность расширения функционала.

Метрика Чепина, или метрика сложности Чепина, является инструментом для оценки сложности программного обеспечения.

4.2 Тестирование программных средств

4.2.1 Функциональное тестирование

После завершения разработки программного обеспечения было проведено функциональное тестирование. Рассмотрим тестирование базы данных «Диспетчер». План тестирования представлен в таблице 27.

Таблица 27 – Результаты функционального тестирования

№ п/п	Назначение теста	Описание теста	Результаты теста
1	Проверка работы хранимой процедуры GetAllRes	Осуществляется выборка резервуаров по их текущему состоянию	Сервер вернул искомые данные (рисунок 20)
2	Проверка работы хранимой процедуры InsertRes и триггера AddReservoir	Процедура осуществляет вставку входных параметров в таблицу Reservoir, проверить, что триггер добавляет дефолтные значения в связанные таблицы и создает запись в журнале о добавлении нового документа	Сервер корректно отработал все действия (рисунок 21)
3	Проверка работы процедуры DeleteRes и триггера DeleteReservoir	Процедура осуществляет удаление строки из таблицы Reservoir, передавая номер удаляемого документа. Триггер должен добавить соответствующую запись в журнал, удалить записи из связанных таблиц	Сервер корректно отработал все действия (рисунок Г.1)
4	Проверка работы процедур GetInfoAboutPumpIn, GetInfoAboutPumpOut	Процедуры должны по номеру резервуара и вернуть все связанные с ним значения	Сервер вернул искомые данные (рисунок Г.2)
5	Проверка работы процедуры UpdateParams	Процедура должна создать запись в журнале о модификации данных, и обновить вносимые параметры в таблицы	Сервер корректно отработал все действия (рисунок Г.3)
6	Проверка работы процедуры AddCollRes	Процедура должна добавить записи сразу о нескольких резервуарах в основную таблицу и в связанные при помощи триггера AddReservoir, а также создать запись в журнал	Сервер корректно отработал все действия (рисунок Г.4)
7	Проверка запроса на получение рейтинга резервуаров по нефтепродукту	Запрос должен вернуть таблицу со списком нефтепродуктов и количеству занимаемых резервуаров	Сервер корректно отработал запрос (рисунок Г.5)
8	Проверка запроса сортировки резервуаров по их объему	Запрос должен вернуть таблицу со списком резервуаров и их объему	Сервер корректно отработал запрос (рисунок Г.6)

	Number	ShortName	Procent
1	1000100	Добавлен	0
2	1000101	Добавлен	0
3	1000102	Добавлен	0
4	1000103	Добавлен	0
5	1000104	Добавлен	0
6	1000105	Добавлен	0
7	1000106	Добавлен	0
8	1000107	Добавлен	0

Рисунок 20 – Результат теста 1

`exec InsertRes 1000108, 'ПТ', 'Сжиженные углеводородные газы'`

105 %

Results Messages

	id	Number	MineralOilid	ReservoirParamsid	Statementid
8	9	1000107	7	9	24
9	10	1000108	5	10	26

	id	CurVolume	MinLevel	MaxLevel	PumpInid	PumpOutid	Procent
8	9	0	0	100	17	18	0
9	10	0	0	100	19	20	0

	id	Speed	PumpTypeid	Statelid
17	19	1	1	25
18	20	1	2	26

	id	DateTimeOfBeginning	Reasonid	Reservoirid	Pumpid
25	25	2023-12-28 09:33:39.110	7	10	19
26	26	2023-12-28 09:33:39.110	7	10	20

Рисунок 21 – Результат теста 2

Результаты тестирования приведены в приложении Г. Программа успешно прошла все функциональные тесты.

4.2.2 Нагрузочное тестирование

Было произведено нагрузочное тестирование базы данных с помощью приложения SQLQueryStress для функций добавления документов, получения списка всех документов и включения или выключения насосов. Результаты тестирования приведены в таблице 28.

Таблица 28 – Результаты нагрузочного тестирования

№ п/п	Наименование операции	Количество итераций	Количество потоков	Затраченное время	Ср. время ожидания клиента	Фактическое затраченное время	Количество логических операций
1	Добавление документа (рисунки 22 – 25)	100	1	0,3686	0,0018	0,0029	31
		10	20	0,4592	0,0111	0,0037	
		1000	1	3,1313	0,0014	0,0011	
		1000	2	5,8685	0,0032	0,0024	
2	Получение списка всех документов (рисунки Д.1 – Д.4)	1	100	0,5006	0,0508	0,0475	69
		100	1	2,1845	0,019	0,0181	
		10	200	27,9352	0,647	0,5327	
		1000	1	23,0341	0,0199	0,0186	
3	Включение или выключение насоса (рисунки Д.5 – Д.8)	10	200	6,289	0,0551	0,0037	8
		200	1	0,6613	0,0016	0,0016	
		1000	2	4,9578	0,0027	0,0011	
		2000	1	6,4939	0,0016	0,0011	

Результаты тестирования приведены в приложении Д.

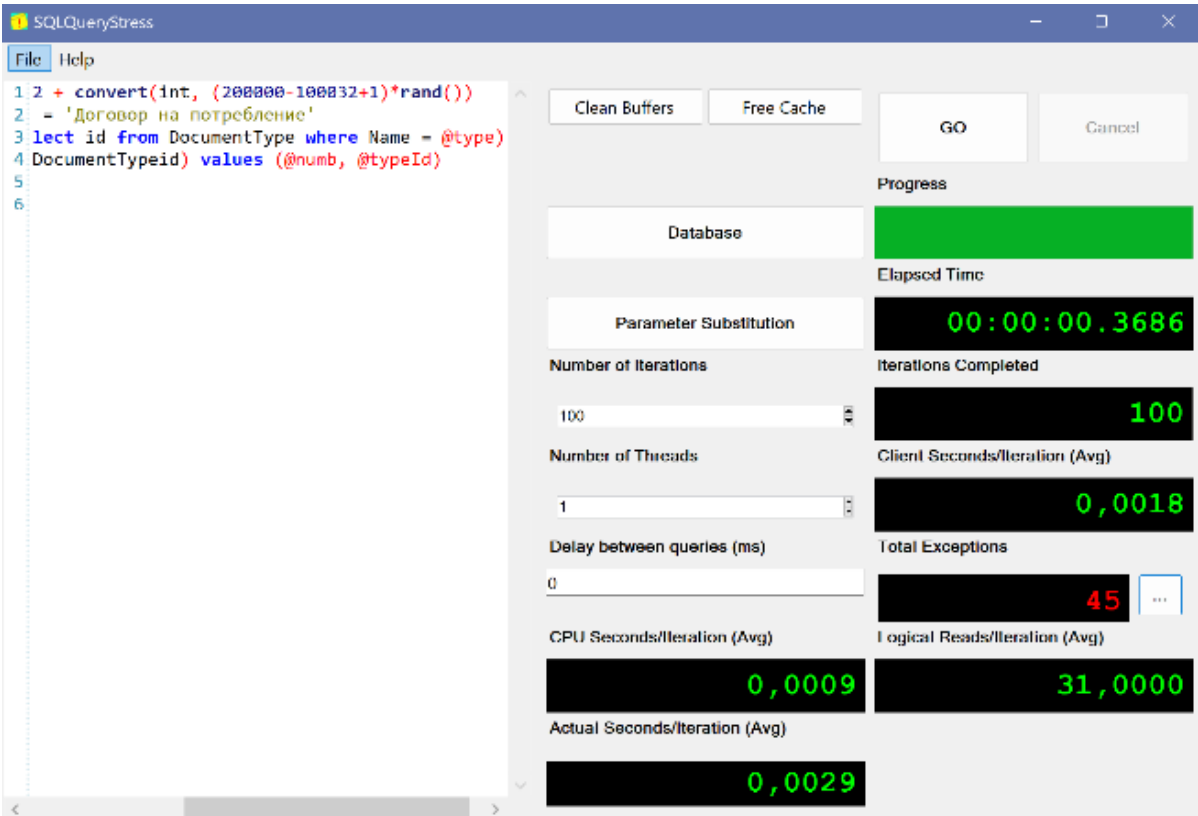


Рисунок 22 – Результат теста 1.1

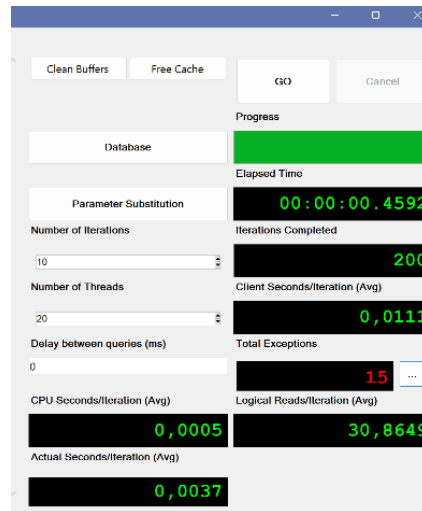


Рисунок 23 – Результат теста 1.2

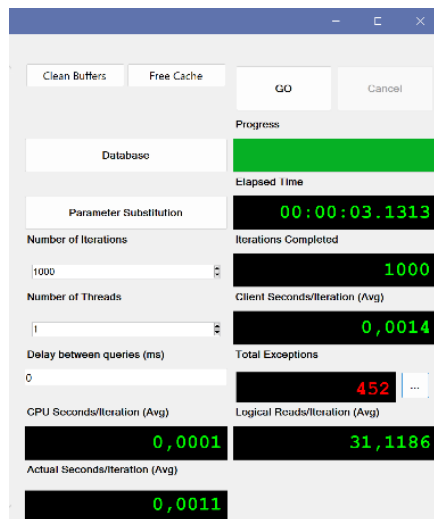


Рисунок 24 – Результат теста 1.3

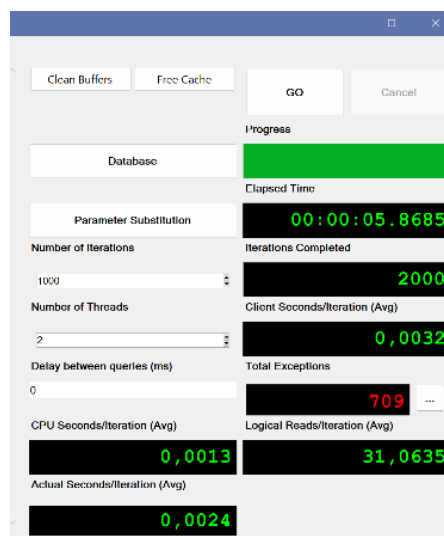


Рисунок 25 – Результат теста 1.4

Таким образом, при увеличении количества потоков нагрузка на систему возрастает незначительно, система работает стабильно. Среднее время отклика системы составляет 0,0679 с. при различном количестве логических операций, что не доставит пользователям дискомфорта и позволит работать в нормальном режиме.

4.3 Оценка качества программного кода на основе метрик

4.3.1 Метрика Холстеда

Метрика Холстеда относится к метрикам, вычисляемым на основании анализа числа строк и синтаксических элементов исходного кода программы [22].

Основу метрики Холстеда составляют четыре измеряемые характеристики программы:

- NUOprrtr – число уникальных операторов программы, включая символы-разделители, имена процедур и знаки операций (словарь операторов);
- NUOprrnd – число уникальных операндов программы (словарь операндов);
- Noprtr – общее число операторов в программе;
- Noprnd – общее число операндов в программе.

Проведем подсчет данной метрики в процедуре AddCollDocs, триггере AddDocument, и пользовательском типе данных NumbType, служащие для добавления нескольких документов в базу данных. Они приведены в листинге 2.

Листинг 2 – AddCollDocs, AddDocument, NumbType

```
CREATE TYPE NumbType AS TABLE
(
    Id int,
    Numberr int,
    Typee varchar(50)
)

CREATE PROCEDURE AddCollDocs
```

```

@mas NumbType READONLY
AS BEGIN
    SET NOCOUNT ON;
    declare @cnt int = (select count(*) from @mas)
    declare @cur int = 1
    while (@cnt>0)
    begin
        insert into Document (Number, DocumentTypeid) values
            ((select Numbeerr from @mas where id = @cur), (select id from
            DocumentType where Name = (select Typee from @mas where id =
            @cur)))
        set @cur = @cur +1
        set @cnt = @cnt-1
    end
END

CREATE TRIGGER AddDocument
ON Document after insert
AS BEGIN
    declare @doc int = (select inserted.id from inserted)
    declare @typeid int = (select inserted.DocumentTypeid from
inserted)
    declare @typeName varchar(50) = (select Name from DocumentType
where id = @typeid)
    declare @numb int = (select inserted.Number from inserted)
    INSERT JournalOfStatement VALUES (GETDATE(), @numb, @typeName, 3)
    update Document
    set DateOfBeginning = GETDATE(),
    DateOfEnd = GETDATE(),
    Cost = 0,
    MineralOilid = 1,
    Contractorid = 1,
    StateId = iif(DateOfEnd >= getdate(), 1, 2) where id = @doc
    if (@typeid = 1)
        INSERT ConsumptionAgreement VALUES (0, @doc)
    else if (@typeid = 2)
        INSERT SalesAgreement VALUES (0, @doc)
    else INSERT LeaseAgreement VALUES (0, @doc)
    INSERT Cocontractor_Document VALUES ((select Contractorid from
Document where id = @doc), @doc)
END

```

Операторы приведенного класса представлены в таблице 29, операнды представлены в таблице 30.

Таблица 29 – Операторы

№	Операторы	Число вхождений
1	=	22
2	.	3
3	as	3
4	If	2
5	CREATE	3
6	TYPE NumbType	2
7	TABLE	1
8	PROCEDURE AddCollDocs	1

Продолжение таблицы 29

№	Операторы	Число вхождений
9	TRIGGER AddDocument	1
10	declare	6
11	SET	2
12	while	1
13	INSERT	6
14	select	9
15	update	1
16	inserted	3
17	;	1
18	,	18
19	()	27
20	-	1
21	+	1
22	GETDATE()	4
23	iif	1
24	n1 = 23	N1 = 119

Таблица 30 – Операнды

№	Операнды	Число вхождений
1	Id	1
2	Numberr	2
3	Typee	2
4	@mas	3
5	@cnt	4
6	@cur	5
7	Document	4
8	Number	3
9	DocumentTypeid	2
10	id	6
11	DocumentType	2
12	Name	2
13	@doc	7
14	inserted.id	1
15	@typeid	4
16	inserted.DocumentTypeid	1
17	@typeName	2
18	@numb	2
19	inserted.Number	1
20	JournalOfStatement	1
21	DateOfBeginning	1
22	DateOfEnd	2
23	Cost	1
24	MineralOilid	1
25	Contractorid	2
26	StateId	1
27	ConsumptionAgreement	1
28	SalesAgreement	1

Продолжение таблицы 30

№	Операнды	Число вхождений
29	LeaseAgreement	1
30	Coontractor_Document	1
31	n2 = 30	N2 = 67

Длина реализации программы рассчитывается по формуле:

$$N = N_1 + N_2 = 119 + 67 = 186.$$

Словарь программы рассчитывается по формуле:

$$n = n_1 + n_2 = 23 + 30 = 53.$$

Длина программы рассчитывается по формуле:

$$N' = n_1 \times K \times \log_2 n_1 + n_2 \times \log_2 n_2 = 110,3 + 164,3 = 274,6.$$

Объем программы рассчитывается по формуле:

$$NV = N \times \log_2 n = 186 \times 5,7 = 1060,2 \text{ бит.}$$

Потенциальный объем программы

$$V' = (2 + n') \times \log_2(2 + n') = 14 \times \log_2(4) = 8.$$

Теоретический уровень программы рассчитывается по формуле:

$$L' = \frac{V'}{NV} = \frac{8}{1060,2} = 0,0075.$$

Сложность программы рассчитывается по формуле:

$$HD = \frac{n_1}{2} \times \frac{N_2}{n_2} = \frac{23}{2} \times \frac{67}{30} = 25,7.$$

Работа программирования рассчитывается по формуле:

$$E = \frac{NV}{L} = \frac{1060,2}{0,0075} = 141\,333,3.$$

Все полученные показатели программы находятся в пределах нормы и удовлетворяют требованиям по метрикам программного кода: длина реализации меньше длины программы, следовательно, несовершенства в программе отсутствуют; теоретический уровень программы весьма низкий, так как потенциальный объем программы в значительной степени меньше ее реального объема; возможности языка программирования использованы на достаточном уровне.

4.3.2 Метрика Чепина

Суть метода Чепина состоит в оценке информационной прочности отдельно взятого программного модуля с помощью анализа характера использования переменных из списка ввода-вывода [22].

Все множество переменных, составляющих список ввода-вывода, разбивается на 4 функциональные группы:

- Р – вводимые переменные для расчетов и для обеспечения вывода;
- М – модифицируемые внутри программы переменные;
- С – переменные, участвующие в управлении работой программного модуля (управляющие переменные);
- Т – не используемые в программе («паразитные») переменные.

Поскольку каждая переменная может выполнять одновременно несколько функций, необходимо учитывать ее в каждой соответствующей функциональной группе. Метрика Чепина:

$$Q = a_1 \times P + a_2 \times M + a_3 \times C + a_4 \times T,$$

где a_1, a_2, a_3, a_4 – весовые коэффициенты.

Весовые коэффициенты использованы для отражения различного влияния на сложность программы каждой функциональной группы. По мнению автора метрики, наибольший вес, равный трем, имеет функциональная группа С, так как она влияет на поток управления программы. Весовые коэффициенты остальных групп распределяются следующим образом: a_1 равно одному, a_2 равно двум, a_4 равно одной второй. Весовой коэффициент группы Т не равен нулю, поскольку «паразитные» переменные не увеличивают сложность потока данных программы, но иногда затрудняют ее понимание. С учетом весовых коэффициентов:

$$Q = P + 2 \times M + 3 \times C + 0.5 \times T.$$

В функциях, представленных в листинге 2:

$$P = 1,$$

$$M = 13,$$

$$C = 2,$$

$$T = 0,$$

$$Q = 1 + 2 \times 13 + 3 \times 2 + 0.5 \times 0 = 33.$$

Полученный показатель метрики Чепина указывает на то, что функции имеют среднюю сложность, и ее разработка не потребует больших усилий.

Заключение

В рамках выпускной квалификационной работы была спроектирована и разработана система менеджмента хранилища нефтепродуктов на основе микросервисной архитектуры.

На первом этапе был проведен анализ предметной области: изучены структура организации нефтехранилищ, их типы, особенности управления резервуарами и выделены бизнес-процессы, осуществлен сравнительный анализ существующих аналогов, определены требования к разрабатываемому приложению. Кроме того, были определены цели разработки и назначение программных средств, был описан перечень основных автоматизируемых функций. После было произведено планирование разработки, а также функционально-стоимостной анализ разработки программных средств. Затраты на проект составили 143680 рублей. Начало работ было определено как 8 апреля, а окончание работ по проекту запланировано на 10 июня текущего года. Затраты трудовых ресурсов составили 522 часа.

На следующем этапе были спроектированы функциональные возможности приложения в виде диаграммы вариантов использования и описаны сценарии вариантов использования. Разработана единая концептуальная модель данных с учетом всех сущностей предметной области. Определена обобщенная архитектура системы с визуализацией на диаграммах развертывания и последовательности. Произведен выбор аппаратного и программного обеспечения для реализации приложения.

Затем было разработано серверное приложение в соответствии со спроектированной архитектурой. Разработка программных средств проводилась в среде разработки Visual Studio 2019 и SQL Server Management Studio под управлением операционной систем Windows 10. В качестве технологической базы использованы ASP.NET Core, gRPC, язык программирования C#, СУБД SQL Server.

На заключительном этапе проведен контроль качества программного обеспечения. Разработанное программное средство было протестировано при

помощи функционального и нагрузочного тестирования. Все тесты приложение прошло успешно. Кроме того, выполнена оценка качества кода посредством метрик Холстеда и Чепина. Данные метрики показали достаточный уровень разработанных модулей системы.

Таким образом, в результате выполнения выпускной квалификационной работы разработан прототип приложения для управления нефтехранилищем, позволяющее автоматизировать основные бизнес-процессы по контролю и распределению нефтепродуктов, а также оптимизировать документооборот предприятия. Программное обеспечение отвечает требованиям качества, надежности и производительности, соответствует современным тенденциям в области программной инженерии.

Система может быть использована на действующих нефтехранилищах и распределительных базах для автоматизации процессов управления и сокращения расходов на ручную обработку данных. Внедрение позволит оптимизировать логистику, сократить время на формирование документов, повысить контроль резервуарного парка.

Благодаря модульности и масштабируемости архитектуры, возможна гибкая интеграция с корпоративными информационными системами предприятий для комплексной автоматизации бизнес-процессов.

Направления дальнейшего развития:

- интеграция приложения в реальное производство;
- добавление функционала для поддержки интеграции с системами учета, бухгалтерии, управления транспортной логистикой;
- разработка мобильного клиента для удаленного управления и мониторинга;
- внедрение элементов машинного обучения для прогнозной аналитики на основе накопленных данных;
- реализация системы удаленного видеонаблюдения за резервуарами.

Список использованных источников

1. Сачивко, А. В. Технология транспортировки и хранения газа, нефти и нефтепродуктов / А. В. Сачивко // Технология хранения нефти и нефтепродуктов: учебное пособие СибГУ им. М. Ф. Решетнева: Изд-во СибГУ им. М. Ф. Решетнева, 2018. – с. 7.
2. Классификация складов нефти и нефтепродуктов. Хранение нефтепродуктов. // StudFiles [Электронный ресурс]. URL: <https://studfile.net/preview/1359495/page:5/> (дата обращения: 12.06.2024).
3. Виды резервуаров для хранения нефтепродуктов. // Производственное объединение РЕМСТРОЙМАШ [Электронный ресурс]. URL: <https://www.nprommz.ru/blog/vidy-rezervuarov-dlya-khraneniya-nefti> (дата обращения: 12.06.2024).
4. Нефтяные насосы. // Химия-2024 [Электронный ресурс]. URL: <https://www.chemistry-expo.ru/ru/ui/17018/> (дата обращения: 12.06.2024).
5. Программа для нефтебазы («Альфа Нефтебаза 2.0»). // АО «ПК - Электроникс» [Электронный ресурс]. URL: http://www.pk-electronics.ru/productions/02/alfa_Neftebaza_2.0 (дата обращения: 12.06.2024).
6. Программно-аппаратный комплекс для нефтебаз. // Группа компаний Сибнефтекарт [Электронный ресурс]. URL: <https://www.sncard.ru/company?view=article&id=20:snk-asn-sistema-upravleniya-dlya-neftebaz&catid=2&ysclid=lvetkjlw0ab75460053> (дата обращения: 12.06.2024).
7. Система управления «ДОЗА-Программа управления нефтебазой» / АЗСДозаСервис. – М.: Руководство пользователя, 2016. – с. 4 – 5.
8. Программа управления резервуарами. // Emerson Electric Co. [Электронный ресурс]. URL: <https://www.emerson.com/en-us/catalog/emerson-tank-manager> (дата обращения: 12.06.2024).
9. АСУ «Топаз-нефтебаза». // Топаз-сервис. [Электронный ресурс]. URL: https://www.topazelectro.ru/product/azs/program/topaz_baza (дата обращения: 12.06.2024).

10. Ньюмен, С. Создание микросервисов / Сэм Ньюмен. – М.: O'Reilly, 2023. – с. 28 – 39.
11. Каштанов, Д.А. Система управления проектами «Microsoft Office Project. // Справочник от автор 24. [Электронный ресурс]. URL: https://spravochnik.ru/menedzhment/sistema_upravleniya_proektami_microsoft_office_project/ (дата обращения: 12.06.2024).
12. Рудь, В. Концептуальное логическое и физическое моделирование данных. // Марк Аврелий. [Электронный ресурс]. URL: https://marcus-aurelius.ru/articles/Data_modeling.html (дата обращения: 12.06.2024).
13. ТОП-10 платформ для разработки программного обеспечения // otzyvmarketing. [Электронный ресурс]. URL: <https://otzyvmarketing.ru/articles/platformy-dlya-razrabotki-programmnogo-obespecheniya/> (дата обращения: 12.06.2024).
14. Ахунов, А. gRPC и все-все-все // proglib. [Электронный ресурс]. URL: <https://proglib.io/p/grpc-i-vse-vse-vse-chast-i-vvedenie-2021-03-26> (дата обращения: 12.06.2024).
15. GraphQL // Skillfactory Media. [Электронный ресурс]. URL: <https://blog.skillfactory.ru/glossary/graphql/> (дата обращения: 12.06.2024).
16. Топ-15 программ для создания базы данных. // Skillfactory Media. [Электронный ресурс]. URL: <https://blog.skillfactory.ru/programmy-dlya-sozdaniya-baz-dannykh/> (дата обращения: 12.06.2024).
17. Что такое SQL Server Management Studio (SSMS)? // Microsoft Learn. [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver16> (дата обращения: 12.06.2024).
18. REST API // Skillfactory Media. [Электронный ресурс]. URL: <https://blog.skillfactory.ru/glossary/rest-api/> (дата обращения: 12.06.2024).
19. Draw.io // Skillfactory Media. [Электронный ресурс]. URL: <https://blog.skillfactory.ru/draw-io/> (дата обращения: 12.06.2024).
20. Различия между контролем качества и обеспечением качества в IT

// Wezom [Электронный ресурс]. URL: <https://wezom.com.ua/blog/razlichija-mezhdu-kontrolem-kachestva-i-obespecheniem-kachestva-v-it> (дата обращения: 12.06.2024).

21. Различные виды тестирования ПО // Atlassian [Электронный ресурс]. URL: <https://www.atlassian.com/ru/continuous-delivery/software-testing/types-of-software-testing> (дата обращения: 12.06.2024).

22. Новичков, А. Метрики Кода И Их Практическая Реализация В Subversion И Clearcase // RusArticles [Электронный ресурс]. URL: <https://web.archive.org/web/20110913071425/http://www.rusarticles.com/programmy-statya/metriki-koda-i-ix-prakticheskaya-realizaciya-v-subversion-i-clearcase-chast-1-metriki-603328.html> (дата обращения: 12.06.2024).

Приложение А
(справочное)
Глоссарий

gRPC (Remote Procedure Calls) – это высокопроизводительный фреймворк для создания распределенных систем, который позволяет разрабатывать клиент-серверные приложения с использованием сильно типизированного интерфейса.

Фреймворк – программная платформа, которая задает структуру и определяет правила написания кода.

API – это набор правил и команд, который позволяет разным программам взаимодействовать друг с другом.

Микросервисная архитектура – это подход к разработке программного обеспечения, при котором систему разделяют на небольшие независимые сервисы. Каждый выполняет определённую функцию и взаимодействует с другими сервисами через API.

.NET – это платформа для разработчиков, состоящая из инструментов, языков программирования и библиотек для создания множества различных типов приложений. ASP.NET Core расширяет платформу разработчика .NET инструментами и библиотеками, специально предназначенными для создания веб-приложений.

Система управления базами данных (СУБД) — это набор инструментов, которые позволяют удобно управлять базами данных: удалять, добавлять, находить элементы, менять их структуру и создавать резервные копии.

SOA (service-oriented architecture) – это набор архитектурных принципов, не зависящих от технологий и продуктов, совсем как полиморфизм или инкапсуляция.

WebSocketAPI – это продвинутая технология, позволяющая страницам использовать протокол WebSocket для открытия сеанса двусторонней интерактивной связи между браузером пользователя и сервером.

Компаундирование – это технология производства бензина, путем добавления присадок из различных резервуаров в отдельный резервуар.

Физическая модель – логическая модель базы данных, выраженная в терминах языка описания конкретной СУБД.

Приложение Б
(рекомендуемое)
Сценарии вариантов использования

Таблица Б.1 – Сценарий выполнения варианта использования «Создать документ»

Наименование	Создать документ
ID	1
Краткое описание	Пользователь добавляет данные документа
Действующие лица	Менеджер
Предусловие	Пользователь прошел авторизацию
Основной поток	1. Добавить данные для документа об аренде Альтернативные потоки: 1.а. Добавить данные для документа о сбыте 1.б. Добавить данные для документа на потребление
Постусловие	Документ создан

Таблица Б.2 – Сценарий выполнения варианта использования «Изменить документ»

Наименование	Изменить документ
ID	2
Краткое описание	Пользователь редактирует данные документа
Действующие лица	Менеджер
Предусловие	Пользователь выбрал документ для редактирования
Основной поток	1. Изменить данные документа
Постусловие	документ отредактирован

Таблица Б.3 – Сценарий выполнения варианта использования «Удалить документ»

Наименование	Удалить документ
ID	3
Краткое описание	Пользователь удаляет документ
Действующие лица	Менеджер
Предусловие	Пользователь выбрал документ для удаления
Основной поток	1. Удаление данных документа
Постусловие	Документ удален

Таблица Б.4 – Сценарий выполнения варианта использования «Добавить резервуар»

Наименование:	Добавить резервуар
ID:	4
Краткое описание:	Пользователь добавляет резервуар
Действующие лица:	диспетчер
Предусловие:	пользователь прошел авторизацию
Основной поток:	<ol style="list-style-type: none"> 1. Назначить тип резервуара 2. Установить нижний допустимый уровень 3. Установить верхний допустимый уровень 4. Установить скорость перекачки
Постусловие:	Резервуар добавлен

Приложение В

(рекомендуемое)

Спецификации атрибутов физической модели данных
и диаграмма физической модели данных

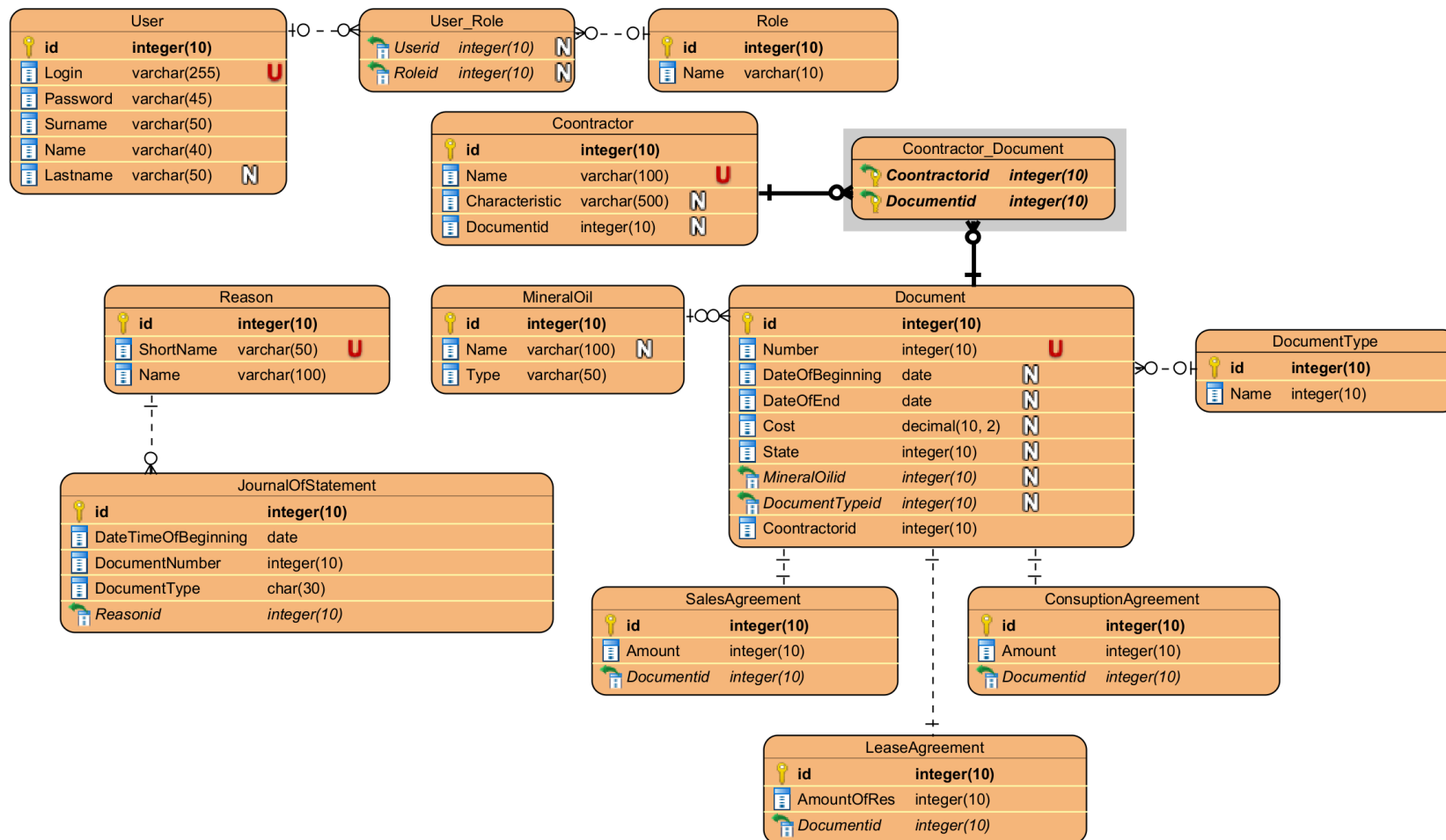


Рисунок В.1 – Физическая модель данных Manager

Таблица В.1 – Спецификации строк таблицы Cocontractor

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	id	Идентификатор	int		PK	Да	0 – 9
2	Name	Название компании или данные физ лица	varchar	100	AK	Да	A – я0 – 9
3	Characteristic	Краткая характеристика о контрагенте	varchar	500		Нет	A – я0 – 9

Таблица В.2 – Спецификации строк таблицы Document

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	id	Идентификатор	int		PK	Да	0 – 9
2	Number	Номер договора	int		AK	Да	0 – 9
3	DateOfBeginning	Дата, с которой договор вступает в силу	date	6		Да	0 – 9, -
4	DateOfEnd	Дата, с которой договор прекращает действовать	date	6		Нет?	0 – 9, -
5	Cost	Сумма сделки	decimal	10		Да	0 – 9
6	DocumentTypeId	Идентификатор типа документа	int		FK	Да	0 – 9
7	MineralOilId	Идентификатор нефтепродукта	int		FK	Да	0 – 9
9	ContractorId	Идентификатор контрагента	int		FK	Да	0 – 9
8	StateId	Идентификатор состояния	int		FK	Да	0 – 9

Таблица В.3 – Спецификации строк таблицы Cocontractor_Document

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	CocontractorId	Идентификатор контрагента	int		FK	Да	0 – 9
2	DocumentId	Идентификатор документа	int		FK	Да	0 – 9

Таблица В.4 – Спецификации строк таблицы DocumentType

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	id	Идентификатор	int		PK	Да	0 – 9
2	Наименование	Вид договора	varchar	30		Да	A – я

Таблица В.5 – Спецификации строк таблицы SalesAgreement

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	id	Идентификатор	int		PK	Да	0 – 9
2	Amout	Объем поставляемой продукции	int			Да	0 – 9
3	Documentid	Идентификатор документа	int		FK	Да	0 – 9

Таблица В.6 – Спецификации строк таблицы ConsumptionAgreement

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	id	Идентификатор	int		PK	Да	0 – 9
2	Amout	Объем поставляемой продукции	int			Да	0 – 9
3	Documentid	Идентификатор документа	int		FK	Да	0 – 9

Таблица В.7 – Спецификации строк таблицы LeaseAgreement

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	id	Идентификатор	int		PK	Да	0 – 9
2	AmountOfRes	Количество занимаемых ресурсов	int			Да	0 – 9
3	Documentid	Идентификатор документа	int		FK	Да	0 – 9

Таблица В.8 – Спецификации строк таблицы MineralOil

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	id	Идентификатор	int		PK	Да	0 – 9
2	Name	Название нефтепродукта	varchar	100		Нет	A – я
3	Type	Класс нефтепродукта	varchar	50		Да	A – я

Таблица В.9 – Спецификации строк таблицы JournalOfStatement

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	id	Идентификатор	int		PK	Да	0 – 9
2	DateTimeOfBeginning	Дата и время перехода в данное состояние	datetime			Да	0 – 9, -

Продолжение таблицы В.9

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
3	DocumentNumber	Номер документа	int		FK	Да	0 – 9
4	DocumentType	Тип документа	int		FK	Да	0 – 9
5	ReasonId	Идентификатор причины	int		FK	Да	0 – 9

Таблица В.10 – Спецификации строк таблицы Reason

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	id	Идентификатор	int		PK	Да	0 – 9
2	ShortName	Краткая запись ошибки или состояния	Целый	2	AK	Да	0 – 9
3	Name	Название причины	Символьный	50		Да	A – я

Таблица В.11 – Спецификации строк таблицы Coontractor

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	id	Идентификатор	int		PK	Да	0 – 9
2	Name	Название компании или данные физ лица	varchar	100	AK	Да	A – я0 – 9
3	Characteristic	Краткая характеристика о контрагенте	varchar	500		Нет	A – я0 – 9

Таблица В.12 – Спецификации строк таблицы Reservoir

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	id	Идентификатор	int	10	PK	Да	0 – 9
2	Number	Номер резервуара	Целый	10	AK	Да	0 – 9
3	MineralOilid	Идентификатор нефтепродукта	int	10	FK	Да	0 – 9

Таблица В.13 – Спецификации строк таблицы ReservoirParams

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
1	id	Идентификатор	int		PK	Да	0 – 9
2	CurVolume	Текущий уровень	double	10		Нет	0 – 9
3	MinLevel	Минимальный допустимый уровень	double	3		Нет	0 – 9

Продолжение таблицы В.13

№ п/п	Наименование	Описание	Тип данных	Размер	Роль	Обязательный	Ограничения значений
4	MaxLevel	Максимальный допустимый уровень	double	10		Нет	0 – 9
5	Procent	Процент заполнения	double	3		Нет	0 – 9
6	PumpInId	Номер насоса закачки	Int		FK	Да	0 – 9
7	PumpOutId	Номер выпускного насоса	int		FK	Да	0 – 9
8	ReservoirId	Идентификатор резервуара	int		FK	Да	0 – 9

Приложение Г
(рекомендуемое)

Результат функционального тестирования

`exec DeleteRes 1000107`

.05 %

Results Messages

	id	Number	MineralOilid	ReservoirParamsid	Statementid
7	8	1000106	6	8	22
8	10	1000108	5	10	28

	id	CurVolume	MinLevel	MaxLevel	Pumplnid	PumpOutid	Procent
7	8	0	0	100	15	16	0
8	10	0	100	2000	19	20	0

	id	Speed	PumpTypeid	Statelid
13	15	1	1	21
14	16	1	2	22
15	19	4	1	27
16	20	5	2	28

	id	DateTimeOfBeginning	Reasonid	Reservoirid	Pumpid
29	29	2023-12-28 09:44:29.773	8	9	17
30	30	2023-12-28 09:44:29.777	8	9	18

Рисунок Г.1 – Результат теста 3

`exec GetInfoAboutRes 1000108`
`exec GetInfoAboutPumpIn 1000108`
`exec GetInfoAboutPumpOut 1000108`

.05 %

Results Messages

	Number	Statement	MineralOilType	MineralOilName	CurrentLevel	MinLevel	MaxLevel	Procent
1	1000108	Изменен	Сжиженные углеводородные газы	ПТ	0	100	2000	0

	Number	Statement	Speed
1	19	Изменен	4

	Number	Statement	Speed
1	20	Изменен	5

Рисунок Г.2 – Результат теста 4

`exec UpdateParams 1000108, 'ПТ', 100, 2000, 4, 5`

.05 %

Results Messages

	id	Number	MineralOilid	ReservoirParamsid	Statementid
8	9	1000107	7	9	24
9	10	1000108	5	10	28

	id	CurVolume	MinLevel	MaxLevel	Pumplnid	PumpOutid	Procent
8	9	0	0	100	17	18	0
9	10	0	100	2000	19	20	0

	id	Speed	PumpTypeid	Statelid
17	19	4	1	27
18	20	5	2	28

	id	DateTimeOfBeginning	Reasonid	Reservoirid	Pumpid
27	27	2023-12-28 09:39:52.797	9	10	19
28	28	2023-12-28 09:39:52.800	9	10	20

Рисунок Г.3 – Результат теста 5

<pre> DECLARE @values NumbNameType INSERT INTO @values (Numberr, Namee, Typee) VALUES (1000109, 'АИ-92', 'Бензин'), (1000110, 'АИ-95', 'Бензин'), (1000111, 'АИ-98', 'Бензин') exec AddCollRes @values </pre>							
105 %							
Results Messages							
id	Number	MineralOilId	ReservoirParamsid	Statementid			
9	11	1000109	1	11	32		
10	12	1000110	2	12	34		
11	13	1000111	3	13	36		
id	CurVolume	MinLevel	MaxLevel	Pumplnid	PumpOutid	Procent	
9	11	0	0	100	21	22	0
10	12	0	0	100	23	24	0
11	13	0	0	100	25	26	0
id	Speed	PumpTypeid	Statelid				
17	21	1	1	31			
18	22	1	2	32			
19	23	1	1	33			
20	24	1	2	34			
21	25	1	1	35			
22	26	1	2	36			
id	DateTimeOfBeginning	Reasonid	Reservoirid	Pumpid			
31	31	2023-12-28 09:50:15.897	7	11	21		
32	32	2023-12-28 09:50:15.897	7	11	22		
33	33	2023-12-28 09:50:15.900	7	12	23		
34	34	2023-12-28 09:50:15.900	7	12	24		
35	35	2023-12-28 09:50:15.900	7	13	25		
36	36	2023-12-28 09:50:15.900	7	13	26		

Рисунок Г.4 – Результат теста 6

	Тип нефтепродукта	Количество
1	Дизельное топливо	1
2	Керосин	2
3	Сжиженные углеводородные газы	2
4	Бензин	3

Рисунок Г.5 – Результат теста 7

	Номер резервуара	Объем
1	1000106	75
2	1000100	540
3	1000104	550
4	1000105	600
5	1000102	700
6	1000101	950
7	1000103	1000

Рисунок Г.6 – Результат теста 8

Приложение Д
(рекомендуемое)

Результат нагрузочного тестирования

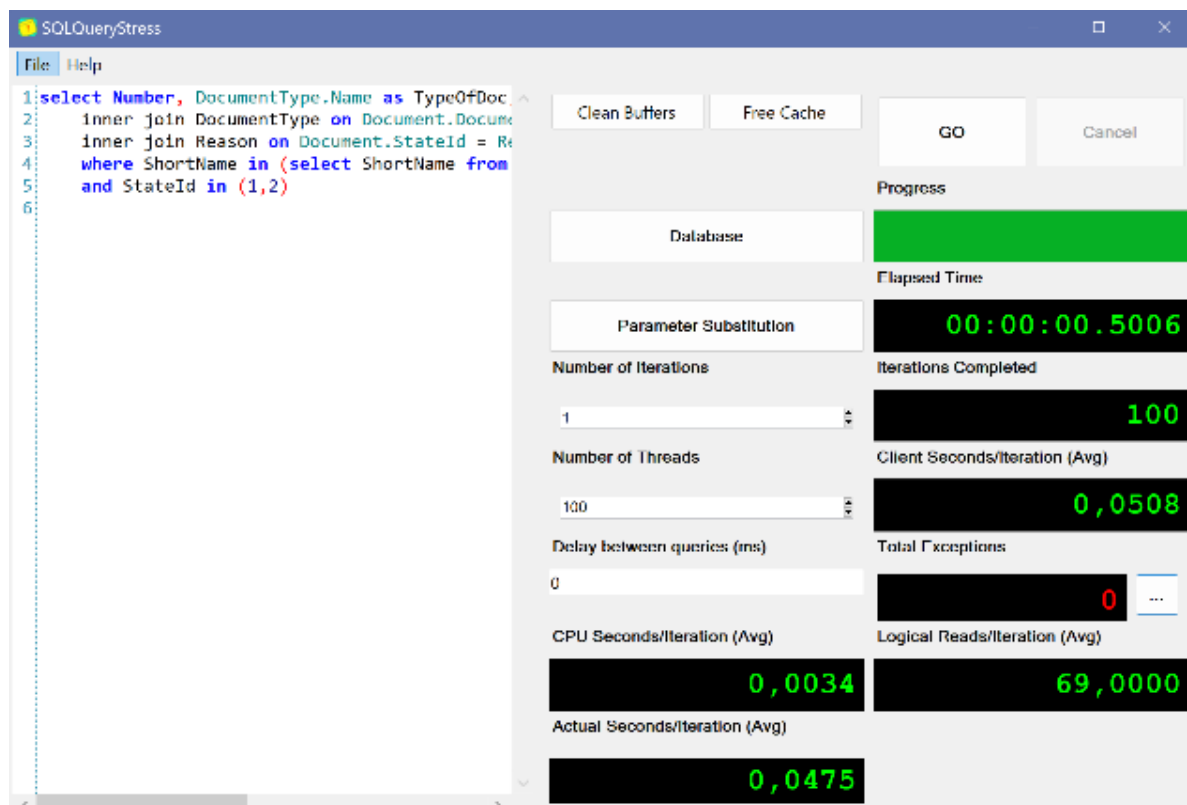


Рисунок Д.1 – Результат теста 2.1

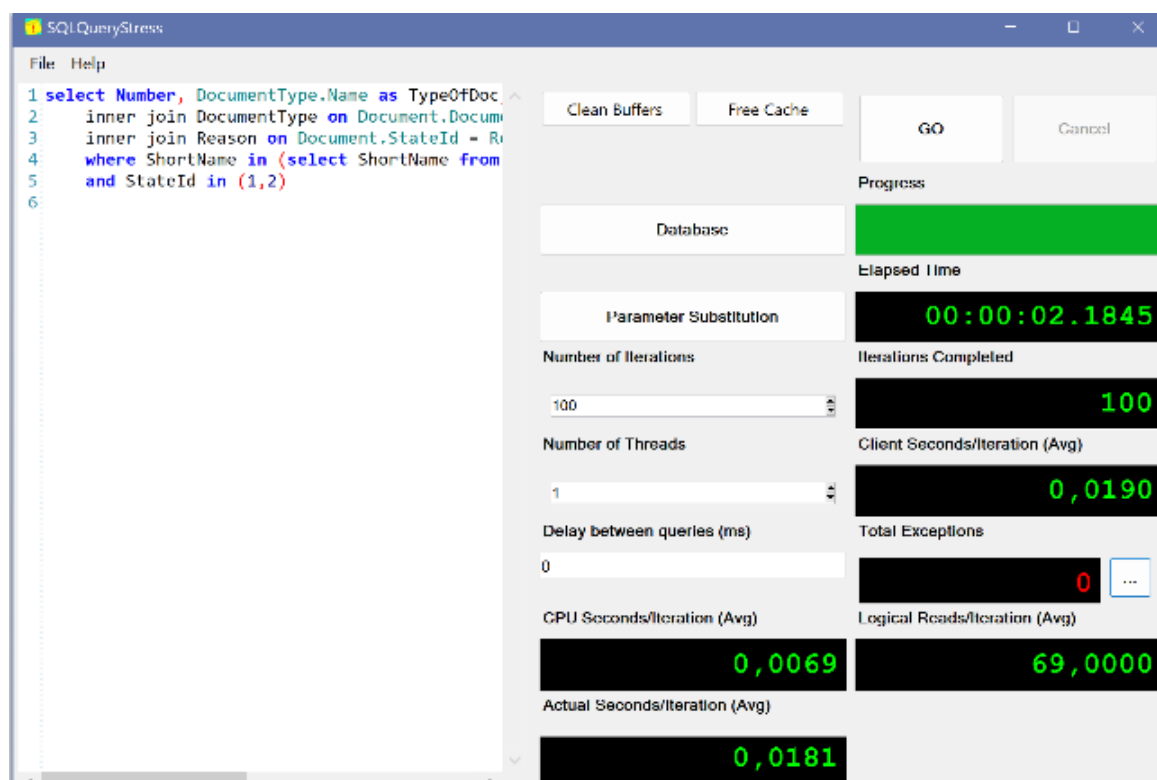


Рисунок Д.2 – Результат теста 2.2

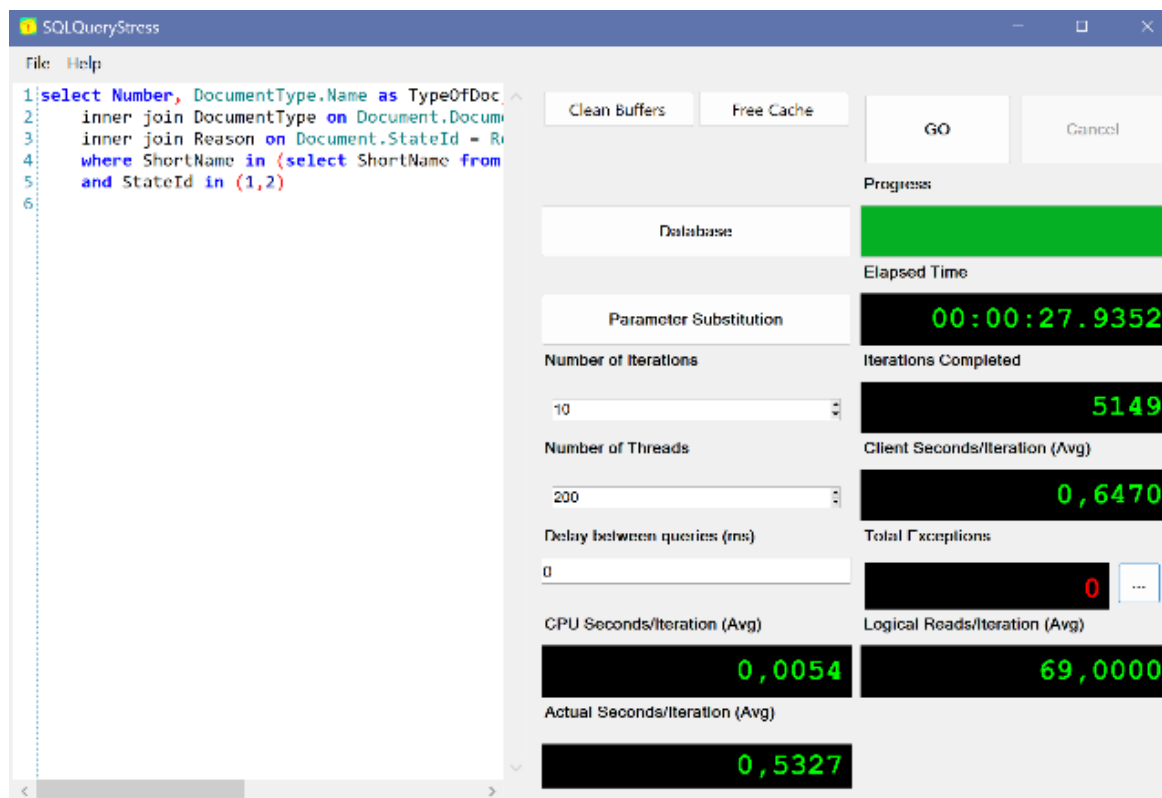


Рисунок Д.3 – Результат теста 2.3

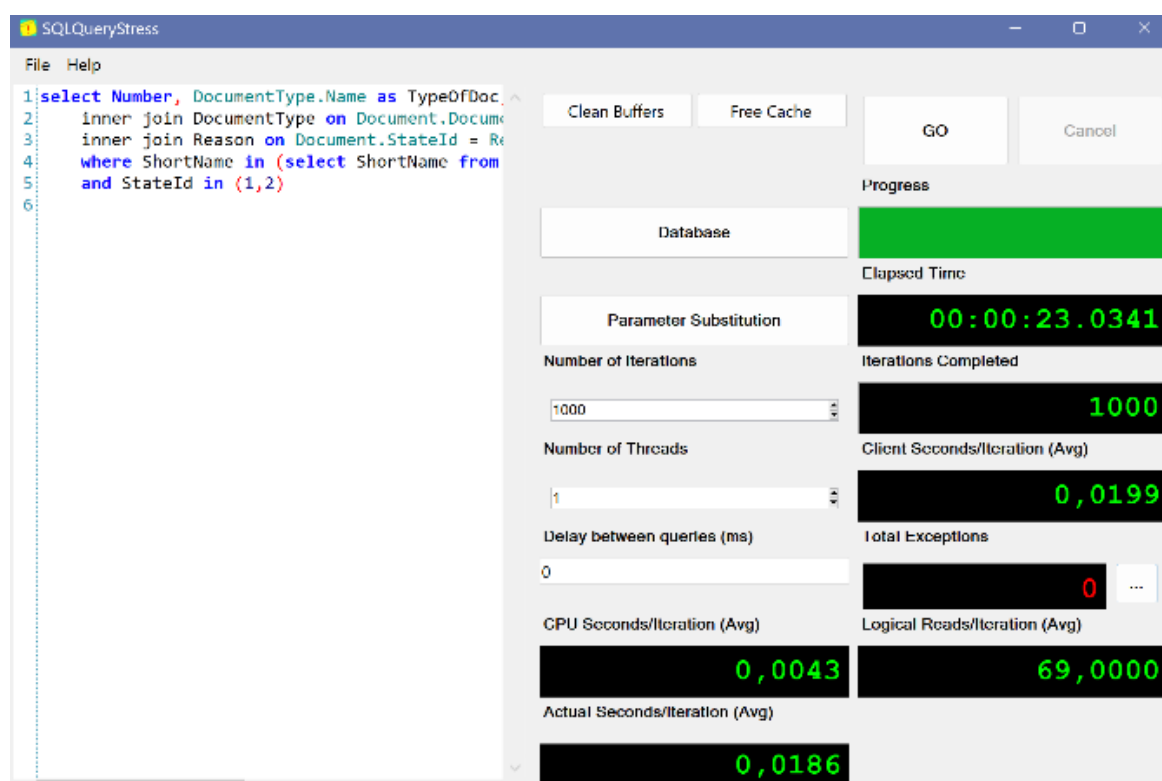


Рисунок Д.4– Результат теста 2.4

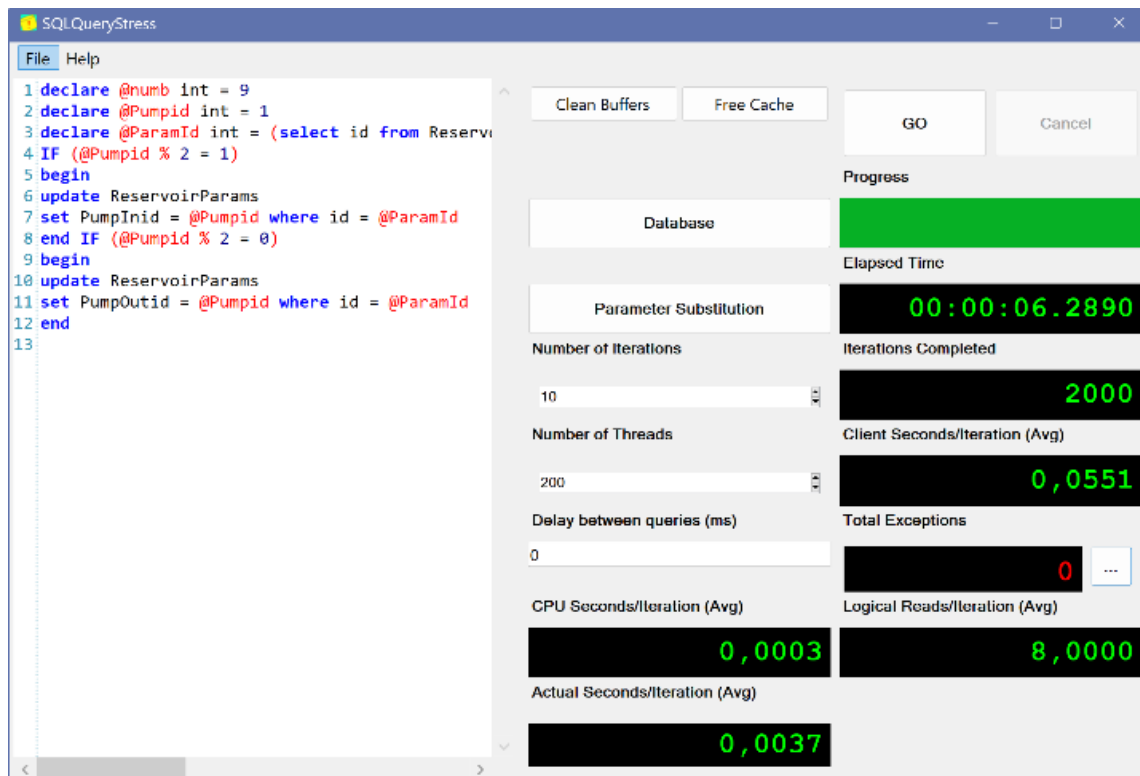


Рисунок Д.5– Результат теста 3.1

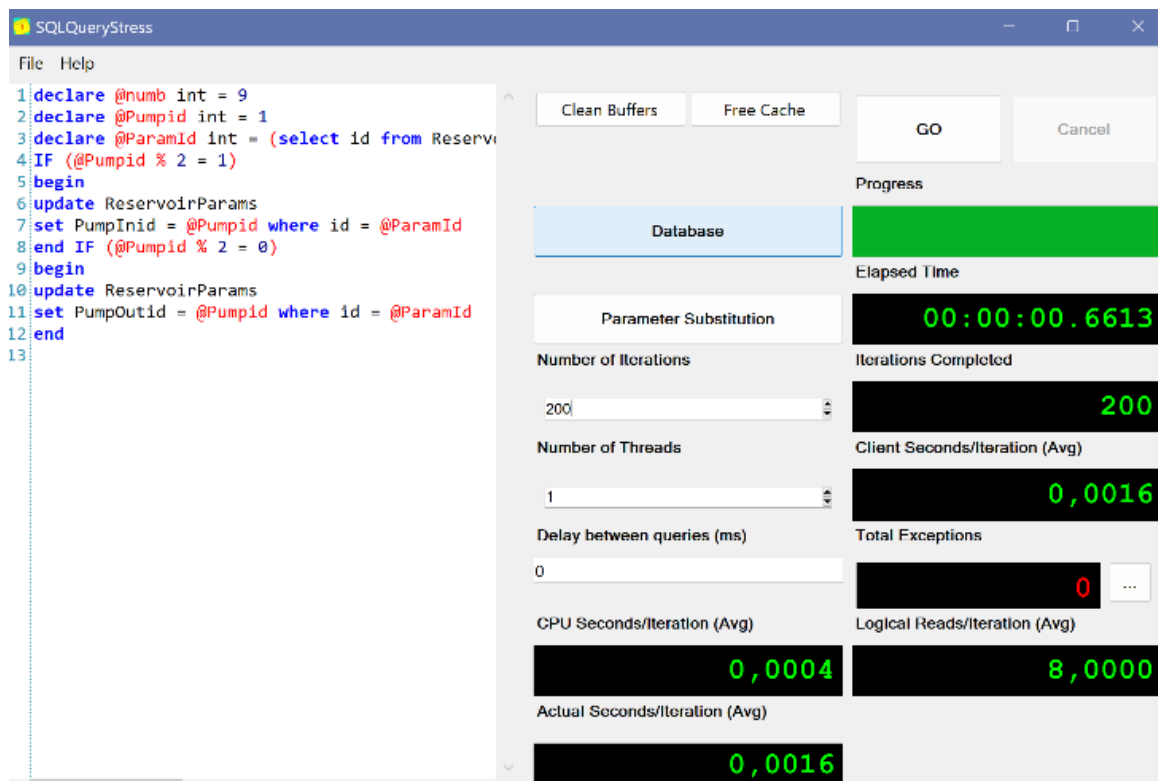


Рисунок Д.6– Результат теста 3.2

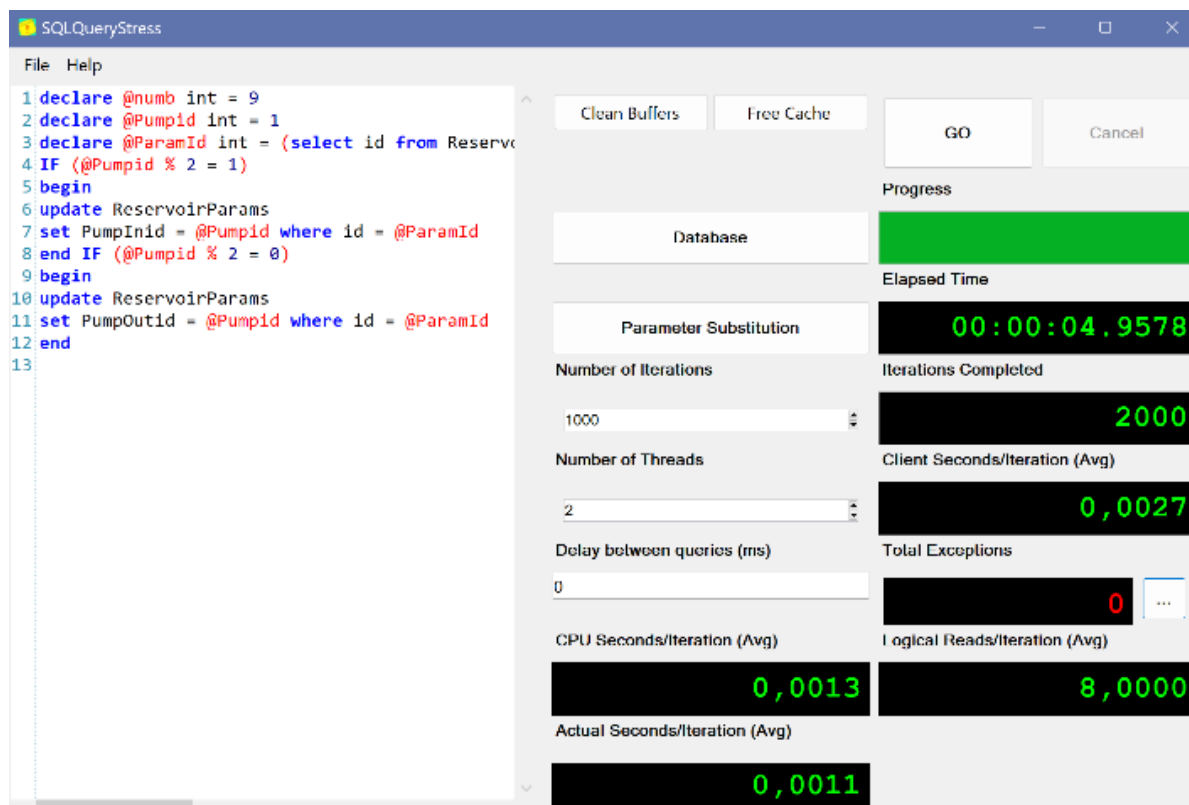


Рисунок Д.7– Результат теста 3.3

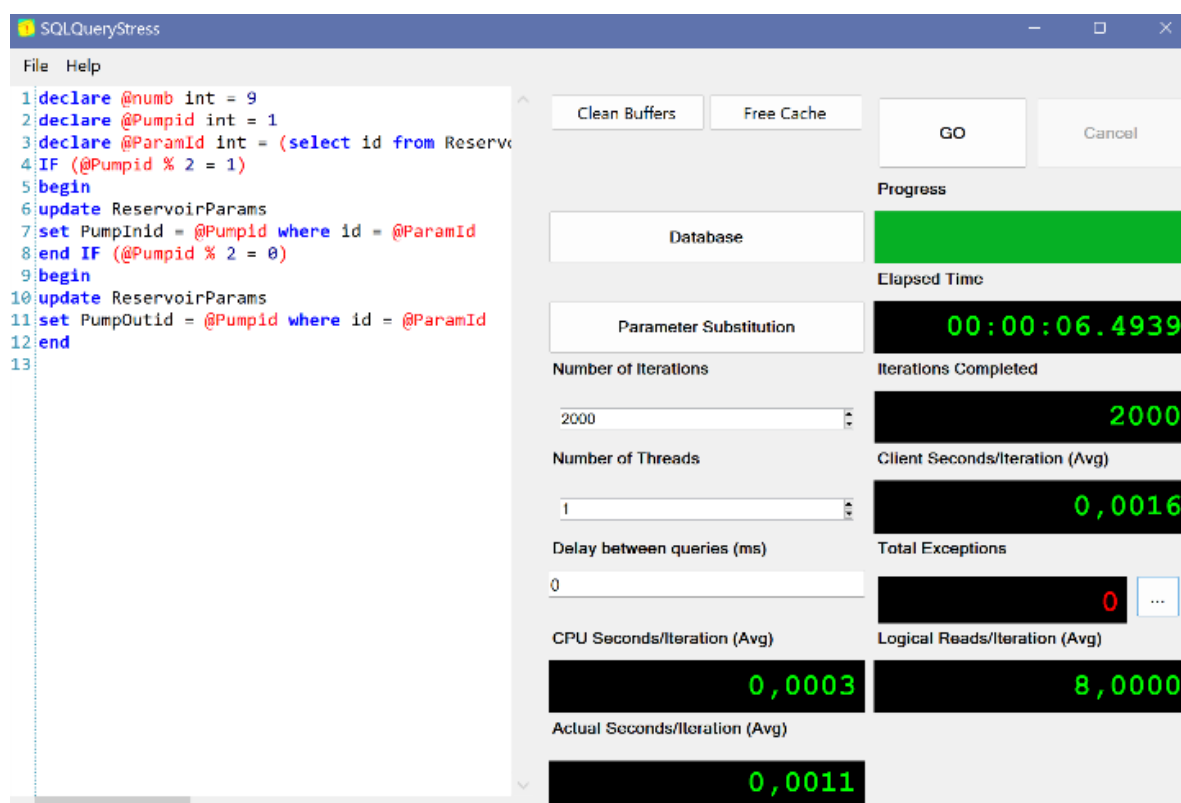


Рисунок Д.8– Результат теста 3.4

Приложение Е
(рекомендуемое)

Листинг серверной части приложения

Листинг Е.1 – Микросервис «Авторизация»

```
using Grpc.Core;
using Microsoft.Extensions.Logging;
using System.Data;
using System.Data.SqlClient;
using System.Threading.Tasks;

namespace GrpcOilAuth
{
    public class AuthenticationService :
    Authentication.AuthenticationBase
    {
        private readonly ILogger<AuthenticationService> _logger;
        public AuthenticationService(ILogger<AuthenticationService>
logger)
        {
            _logger = logger;
        }
        private static async Task<SqlConnection> GetConnection()
        {
            var serverName = "DESKTOP-0AUAV6E";
            var databaseName = "Users";
            var connectionString = @"Data
Source={serverName};Initial Catalog={databaseName};Integrated
Security=True";
            var sqlConnection = new
SqlConnection(connectionString);
            await sqlConnection.OpenAsync();
            return sqlConnection;
        }
        private static async Task UserInsert(SqlConnection
sqlConnection, UserInfo userInfo)
        {
            var sqlExpression = $"exec AddUser '{userInfo.Login}',
'{userInfo.Password}', '{userInfo.Surname}', '{userInfo.Name}',
'{userInfo.Lastname}', '{userInfo.Role}'";
            var sqlCommand = new SqlCommand(sqlExpression,
sqlConnection);
            await sqlCommand.ExecuteNonQuery();
        }
        private static async Task<bool> UserExist(SqlConnection
sqlConnection, string login)
        {
            var sqlExpression = $"select id from Users where
Login='{login}'";
            var sqlCommand = new SqlCommand(sqlExpression,
sqlConnection);
            using var sqlDataReader = await
sqlCommand.ExecuteReaderAsync();
            return sqlDataReader.HasRows;
        }
        private static async Task<bool>
PasswordIsCorrect(SqlConnection sqlConnection, UserLogPass
userLogPass)
        {
            var sqlExpression = $"select id from Users where
Login='{userLogPass.Login}' and Password='{userLogPass.Password}'";
```

```

        var sqlCommand = new SqlCommand(sqlExpression,
sqlConnection);
        using var sqlDataReader = await
sqlCommand.ExecuteReaderAsync();
        return sqlDataReader.HasRows;
    }
    public static async Task<string>
GetRoleByLogin(SqlConnection sqlConnection, string login)
    {
        var sqlExpression = $"select Name from Role where id =
(select Roleid from User_Role where Userid = (select id from Users
where login = '{login}'))";
        var sqlCommand = new SqlCommand(sqlExpression,
sqlConnection);
        using var sqlDataReader = await
sqlCommand.ExecuteReaderAsync();
        await sqlDataReader.ReadAsync();
        var roleName =
sqlDataReader.GetString(sqlDataReader.GetOrdinal("Name"));
        return roleName;
    }
    public override async Task<UserRole> GetUserRole(UserLogin
userLogin, ServerCallContext context)
    {
        using var sqlConnection = await GetConnection();
        var roleName = await GetRoleByLogin(sqlConnection,
userLogin.Login);
        return new() { Role = roleName };
    }
    public override async Task<RegistrationResult>
UserRegistration(UserInfo userInfo, ServerCallContext context)
    {
        using var sqlConnection = await GetConnection();
        if (await UserExist(sqlConnection, userInfo.Login))
        {
            return new() { Result =
RegistrationResultType.RegUserExists };
        }
        await UserInsert(sqlConnection, userInfo);
        return new() { Result =
RegistrationResultType.RegSuccess };
    }
    public override async Task<AuthorizationResult>
UserAuthorization(UserLogPass userLogPass, ServerCallContext context)
    {
        using var sqlConnection = await GetConnection();
        if (await UserExist(sqlConnection, userLogPass.Login)
is not true)
        {
            return new() { Result =
AuthorizationResultType.AuthUserNotExists };
        }
        if (await PasswordIsCorrect(sqlConnection,
userLogPass) is not true)
        {
            return new() { Result =
AuthorizationResultType.AuthIncorrectPass };
        }
    }

```

```

        return new() { Result =
AuthorizationResultType.AuthSuccess };
    }
    public static async Task<string>
GetUserNameByLogin(SqlConnection sqlConnection, string login)
    {
        var sqlExpression = $"exec GetFullName '{login}'";
        var sqlCommand = new SqlCommand(sqlExpression,
sqlConnection);
        using var sqlDataReader = await
sqlCommand.ExecuteReaderAsync();
        await sqlDataReader.ReadAsync();
        var userName =
sqlDataReader.GetString(sqlDataReader.GetOrdinal("Name"));
        return userName;
    }
    public override async Task<UserName> GetUserName(UserLogin
userLogin, ServerCallContext context)
    {
        using var sqlConnection = await GetConnection();
        var userName = await GetUserNameByLogin(sqlConnection,
userLogin.Login);
        return new() { Name = userName };
    }
}
}

```

Листинг Е.2 – Микросервис «Менеджер»

```

using Google.Protobuf.WellKnownTypes;
using Grpc.Core;
using Microsoft.Extensions.Logging;
using System.Data.SqlClient;
using System.Threading.Tasks;

namespace GrpcOilManager
{
    public class ManagerService : Manager.ManagerBase
    {
        private readonly ILogger<ManagerService> _logger;
        public ManagerService(ILogger<ManagerService> logger)
        {
            _logger = logger;
        }
        private static async Task<SqlConnection> GetConnection()
        {
            var serverName = "DESKTOP-0AUAV6E";
            var databaseName = "Manager";
            var connectionString = @"Data
Source={serverName};Initial Catalog={databaseName};Integrated
Security=True";
            var sqlConnection = new
SqlConnection(connectionString);
            await sqlConnection.OpenAsync();
            return sqlConnection;
        }
        public override async Task GetListOfDocsStream(Empty _,
IServerStreamWriter<DocumentPreview> documents, ServerCallContext

```

```

context)
    {
        using var sqlConnection = await GetConnection();
        var sqlExpression = $"exec GetAllDocs";
        var sqlCommand = new SqlCommand(sqlExpression,
sqlConnection);
        using var sqlDataReader = await
sqlCommand.ExecuteReaderAsync();
        while (await sqlDataReader.ReadAsync())
        {
            var documentPreview = new DocumentPreview
            {
                Number =
sqlDataReader.GetInt32(sqlDataReader.GetOrdinal("Number")).ToString(),
                State =
sqlDataReader.GetString(sqlDataReader.GetOrdinal("State")),
                Type =
sqlDataReader.GetString(sqlDataReader.GetOrdinal("TypeOfDoc"))
            };
            await documents.WriteAsync(documentPreview);
        }
    }
    public override async Task<Empty> AddDocument(DocumentAdd
documentAdd, ServerCallContext context)
    {
        using var sqlConnection = await GetConnection();
        var sqlExpression = $"exec InsertDoc
{documentAdd.Number}, '{documentAdd.Type}'";
        var sqlCommand = new SqlCommand(sqlExpression,
sqlConnection);
        await sqlCommand.ExecuteNonQueryAsync();
        return new();
    }
    public override async Task<Empty> DelDocument(DocumentNumber
documentNumber, ServerCallContext context)
    {
        using var sqlConnection = await GetConnection();
        var sqlExpression = $"exec DeleteDoc
{documentNumber.Number}";
        var sqlCommand = new SqlCommand(sqlExpression,
sqlConnection);
        await sqlCommand.ExecuteNonQueryAsync();
        return new();
    }
    public override async Task<DocumentInfo>
GetDocumentInfo(DocumentNumber documentNumber, ServerCallContext
context)
    {
        using var sqlConnection = await GetConnection();
        var sqlExpression = $"exec GetInfoAboutDocs
{documentNumber.Number}";
        var sqlCommand = new SqlCommand(sqlExpression,
sqlConnection);
        using var sqlDataReader = await
sqlCommand.ExecuteReaderAsync();
        await sqlDataReader.ReadAsync();
        var documentInfo = new DocumentInfo()
        {

```

```

        Type =
sqlDataReader.GetString(sqlDataReader.GetOrdinal("Type")),
        DateBegin =
sqlDataReader.GetDateTime(sqlDataReader.GetOrdinal("DateOfBeginning"))
.ToString(),
        DateEnd =
sqlDataReader.GetDateTime(sqlDataReader.GetOrdinal("DateOfEnd")).ToString(),
        Cost =
sqlDataReader.GetSqlMoney(sqlDataReader.GetOrdinal("Cost")).ToString()
,
        Amount =
sqlDataReader.GetInt32(sqlDataReader.GetOrdinal("Amount")).ToString(),
        Customer =
sqlDataReader.GetString(sqlDataReader.GetOrdinal("Customer")),
        OilType =
sqlDataReader.GetString(sqlDataReader.GetOrdinal("OilType")),
    };
    return documentInfo;
}
public override async Task<Empty>
EditDocument(DocumentEditInfo documentEditInfo, ServerCallContext
context)
{
    using var sqlConnection = await GetConnection();
    var sqlExpression = $"exec UpdateParams
{documentEditInfo.Number}, " +
        $"{documentEditInfo.DateBegin}',
'{documentEditInfo.DateEnd}', {documentEditInfo.Cost}, " +
        $"{documentEditInfo.Amount},
'{documentEditInfo.Customer}', '{documentEditInfo.OilType}'";
    var sqlCommand = new SqlCommand(sqlExpression,
sqlConnection);
    await sqlCommand.ExecuteNonQueryAsync();
    return new();
}
}
}

```

Листинг Е.3 – Микросервис «Диспетчер»

```

using Google.Protobuf.WellKnownTypes;
using Grpc.Core;
using Microsoft.Extensions.Logging;
using System.Data.SqlClient;
using System.Threading.Tasks;

namespace GrpcOilDispatcher
{
    public class DispatcherService : Dispatcher.DispatcherBase
    {
        private readonly ILogger<DispatcherService> _logger;
        public DispatcherService(ILogger<DispatcherService> logger)
        {
            _logger = logger;
        }
        private static async Task<SqlConnection> GetConnection()
        {

```

```

        var serverName = "DESKTOP-0AUAV6E";
        var databaseName = "Dispatcher";
        var connectionString = @"Data
Source={serverName};Initial Catalog={databaseName};Integrated
Security=True";
        var sqlConnection = new
SqlConnection(connectionString);
        await sqlConnection.OpenAsync();
        return sqlConnection;
    }
    public override async Task GetListOfTanksStream(Empty _,
IServerStreamWriter<StorageTankPreview> storageTanks,
ServerCallContext context)
    {
        using var sqlConnection = await GetConnection();
        var sqlExpression = $"exec GetAllRes";
        var sqlCommand = new SqlCommand(sqlExpression,
sqlConnection);
        using var sqlDataReader = await
sqlCommand.ExecuteReaderAsync();
        while (await sqlDataReader.ReadAsync())
        {
            var storageTankPreview = new StorageTankPreview
            {
                Number =
sqlDataReader.GetInt32(sqlDataReader.GetOrdinal("Number")).ToString(),
                State =
sqlDataReader.GetString(sqlDataReader.GetOrdinal("ShortName")),
                FillPercentage =
sqlDataReader.GetSqlSingle(sqlDataReader.GetOrdinal("Procent")).ToStri
ng()
            };
            await
storageTanks.WriteAsync(storageTankPreview);
        }
    }
    public override async Task<Empty>
AddStorageTank(StorageTankAdd storageTankAdd, ServerCallContext
context)
    {
        using var sqlConnection = await GetConnection();
        var sqlExpression = $"exec InsertRes
{storageTankAdd.Number}, '{storageTankAdd.OilName}',
'{storageTankAdd.OilType}'";
        var sqlCommand = new SqlCommand(sqlExpression,
sqlConnection);
        await sqlCommand.ExecuteNonQueryAsync();
        return new();
    }
    public override async Task<Empty>
DelStorageTank(StorageTankNumber storageTankNumber, ServerCallContext
context)
    {
        using var sqlConnection = await GetConnection();
        var sqlExpression = $"exec DeleteRes
{storageTankNumber.Number}";
        var sqlCommand = new SqlCommand(sqlExpression,
sqlConnection);

```

```

        await sqlCommand.ExecuteNonQueryAsync();
        return new();
    }

    public override async Task<StorageTankInfo>
    GetStorageTankInfo(StorageTankNumber storageTankNumber,
    ServerCallContext context)
    {
        using var sqlConnection = await GetConnection();
        var sqlExpression = $"exec GetInfoAboutRes
{storageTankNumber.Number}";
        var sqlCommand = new SqlCommand(sqlExpression,
sqlConnection);
        using var sqlDataReader = await
sqlCommand.ExecuteReaderAsync();
        await sqlDataReader.ReadAsync();
        var storageTankInfo = new StorageTankInfo()
        {
            State =
sqlDataReader.GetString(sqlDataReader.GetOrdinal("Statement")),
            OilType =
sqlDataReader.GetString(sqlDataReader.GetOrdinal("MineralOilType")),
            OilName =
sqlDataReader.GetString(sqlDataReader.GetOrdinal("MineralOilName")),
            CurrLevel =
sqlDataReader.GetSqlSingle(sqlDataReader.GetOrdinal("CurrentLevel")).To
oString(),
            MinLevel =
sqlDataReader.GetSqlSingle(sqlDataReader.GetOrdinal("MinLevel")).ToStr
ing(),
            MaxLevel =
sqlDataReader.GetSqlSingle(sqlDataReader.GetOrdinal("MaxLevel")).ToStr
ing(),
            FillPercentage =
sqlDataReader.GetSqlSingle(sqlDataReader.GetOrdinal("Procent")).ToStri
ng(),
        };
        return storageTankInfo;
    }

    private static async Task<PumpInfo> GetPumpInfo(string
procName, string number)
    {
        using var sqlConnection = await GetConnection();
        var sqlExpression = $"exec {procName} {number}";
        var sqlCommand = new SqlCommand(sqlExpression,
sqlConnection);
        using var sqlDataReader = await
sqlCommand.ExecuteReaderAsync();
        await sqlDataReader.ReadAsync();
        var pumpInfo = new PumpInfo
        {
            Number =
sqlDataReader.GetInt32(sqlDataReader.GetOrdinal("Number")).ToString(),
            State =
sqlDataReader.GetString(sqlDataReader.GetOrdinal("Statement")),
            Speed =
sqlDataReader.GetSqlSingle(sqlDataReader.GetOrdinal("Speed")).ToString
()
        };
    }

```

```

        return pumpInfo;
    }

    public override async Task<PumpInfo>
    GetInputPumpInfo(StorageTankNumber storageTankNumber,
    ServerCallContext context)
    {
        return await GetPumpInfo("GetInfoAboutPumpIn",
    storageTankNumber.Number);
    }

    public override async Task<PumpInfo>
    GetOutputPumpInfo(StorageTankNumber storageTankNumber,
    ServerCallContext context)
    {
        return await GetPumpInfo("GetInfoAboutPumpOut",
    storageTankNumber.Number);
    }

    private static async Task<PumpId>
    GetPumpIdByTankNumber(string procName, string field, string number)
    {
        using var sqlConnection = await GetConnection();
        var sqlExpression = $"exec {procName} {number}";
        var sqlCommand = new SqlCommand(sqlExpression,
    sqlConnection);
        using var sqlDataReader = await
    sqlCommand.ExecuteReaderAsync();
        await sqlDataReader.ReadAsync();
        var pumpId =
    sqlDataReader.GetInt32(sqlDataReader.GetOrdinal(field)).ToString();
        return new() { Id = pumpId };
    }

    public override async Task<PumpId>
    GetInputPumpIdByTankNumber(StorageTankNumber storageTankNumber,
    ServerCallContext context)
    {
        return await
    GetPumpIdByTankNumber("GetInPumpIdByTankNumber", "PumpInid",
    storageTankNumber.Number);
    }

    public override async Task<PumpId>
    GetOutputPumpIdByTankNumber(StorageTankNumber storageTankNumber,
    ServerCallContext context)
    {
        return await
    GetPumpIdByTankNumber("GetOutPumpIdByTankNumber", "PumpOutid",
    storageTankNumber.Number);
    }

    public override async Task<Empty>
    SwitchPumpState(SwitchPumpStateInfo switchPumpStateInfo,
    ServerCallContext context)
    {
        using var sqlConnection = await GetConnection();
        var sqlExpression = $"exec TurnOnOffPumps
    {switchPumpStateInfo.TankNumber}, {switchPumpStateInfo.PumpId}";
        var sqlCommand = new SqlCommand(sqlExpression,
    sqlConnection);

        await sqlCommand.ExecuteNonQuery();
        return new();
    }

```

```

        }
        public override async Task<Empty>
EditStorageTank(StorageTankEditInfo storageTankEditInfo,
ServerCallContext context)
        {
            using var sqlConnection = await GetConnection();
            var sqlExpression = $"exec UpdateParams " +
                $"{storageTankEditInfo.TankNumber},
'{{storageTankEditInfo.OilName}}'," +
                $"{storageTankEditInfo.MinLevel},
{{storageTankEditInfo.MaxLevel}}," +
                $"{storageTankEditInfo.IpSpeed},
{{storageTankEditInfo.OpSpeed}}";
            var sqlCommand = new SqlCommand(sqlExpression,
sqlConnection);
            await sqlCommand.ExecuteNonQuery();
            return new();
        }
    }
}

```

Листинг Е.4 – Микросервис «Журнал»

```

using Grpc.Core;
using Microsoft.Extensions.Logging;
using System.Data.SqlClient;
using System.Threading.Tasks;

namespace GrpcOilLogs
{
    public class LogsService : Logs.LogsBase
    {
        private readonly ILogger<LogsService> _logger;
        public LogsService(ILogger<LogsService> logger)
        {
            _logger = logger;
        }
        private static async Task<SqlConnection>
GetConnection(string databaseName)
        {
            var serverName = "DESKTOP-0AUAV6E";
            var connectionString = @"Data
Source={{serverName}};Initial Catalog={{databaseName}};Integrated
Security=True";
            var sqlConnection = new
SqlConnection(connectionString);
            await sqlConnection.OpenAsync();
            return sqlConnection;
        }
        public override async Task GetEntryStream(DatabaseName
databaseName, IServerStreamWriter<LogEntry> entries, ServerCallContext
context)
        {
            using var sqlConnection = await
GetConnection(databaseName.Name);
            var sqlExpression = $"exec GetInfoJournal";
            var sqlCommand = new SqlCommand(sqlExpression,
sqlConnection);

```

```

        var sqlDataReader = await
sqlCommand.ExecuteReaderAsync();
        while (await sqlDataReader.ReadAsync())
        {
            var logEntry = new LogEntry
            {
                Resource =
sqlDataReader.GetString(sqlDataReader.GetOrdinal("Resours")),
                Identifier =
sqlDataReader.GetInt32(sqlDataReader.GetOrdinal("Number")).ToString(),
                Datetime =
sqlDataReader.GetDateTime(sqlDataReader.GetOrdinal("DateTimeOfBeginnin
g")).ToString(),
                State =
sqlDataReader.GetString(sqlDataReader.GetOrdinal("Statement"))
            };
            await entries.WriteAsync(logEntry);
        }
    }
}

```

Приложение Ж
(рекомендуемое)
Исходные коды запросов

Листинг Ж.1 – Рейтинг контрагентов по выгоде сделок за период времени

```
select Cocontractor.Name, SUM(Cost) as 'Стоимость'
from Document inner join Cocontractor_Document on Document.id =
Cocontractor_Document.Documentid
inner join Cocontractor on Cocontractor_Document.Cocontractorid =
Cocontractor.id
group by Cocontractor.Name
order by 'Стоимость'
```

Листинг Ж.2 – Рейтинг контрагентов по объему поставок

```
select Cocontractor.Name, SUM(Amount) as 'Объем'
from Document inner join Cocontractor_Document on Document.id =
Cocontractor_Document.Documentid
inner join Cocontractor on Cocontractor_Document.Cocontractorid =
Cocontractor.id
inner join ConsumptionAgreement on Document.id =
ConsumptionAgreement.Documentid
group by Cocontractor.Name
order by 'Объем'
```

Листинг Ж.3 – Рейтинг продаж нефтепродуктов

```
select MineralOil.Name, SUM(Cost) as 'Стоимость'
from Document inner join MineralOil on Document.MineralOilid =
MineralOil.id
inner join SalesAgreement on Document.id = SalesAgreement.Documentid
group by MineralOil.Name
order by 'Стоимость'
```

Листинг Ж.4 – Рейтинг резервуаров по типу нефтепродуктов

```
select MineralOil.Type as 'Тип нефтепродукта', Count(Reservoir.id) as
'Количество'
from MineralOil inner join Reservoir on Reservoir.MineralOilid =
MineralOil.id
group by MineralOil.Type
order by 'Количество'
```

Листинг Ж.5 – Сортировка резервуаров по их объему

```
select Reservoir.Number as 'Номер резервуара',
Max(ReservoirParams.MaxLevel) as 'Объем'
from Reservoir inner join ReservoirParams on
Reservoir.ReservoirParamsid = ReservoirParams.id
group by Reservoir.Number
order by 'Объем'
```

Листинг Ж.6 – Рейтинг контрагентов по объему продаж

```
select Cocontractor.Name, SUM(Amount) as 'Объем'
from Document inner join Cocontractor_Document on Document.id =
Cocontractor_Document.Documentid
inner join Cocontractor on Cocontractor_Document.Cocontractorid =
Cocontractor.id
inner join SalesAgreement on Document.id = SalesAgreement.Documentid
group by Cocontractor.Name
order by 'Объем'
```

Листинг Ж.7 – Рейтинг поставок нефтепродуктов

```
select MineralOil.Name, SUM(Cost) as 'Стоимость'
from Document inner join MineralOil on Document.MineralOilid =
MineralOil.id
inner join LeaseAgreement on Document.id = LeaseAgreement.Documentid
group by MineralOil.Name
order by 'Стоимость'
```

Приложение И
(рекомендуемое)

Исходные коды скриптов хранимых процедур и функций

Листинг И.1 – Получение списка документов

```
create procedure GetAllDocs
as begin
    select Number, DocumentType.Name as TypeOfDoc, Reason.ShortName
as State from Document
    inner join DocumentType on Document.DocumentTypeid =
DocumentType.id
    inner join Reason on Document.StateId = Reason.id where ShortName
in (select ShortName from Reason where id in (select StateId from
Document))and StateId in (1,2)
end
```

Листинг И.2 – Добавление документа

```
create procedure InsertDoc
@numb int,
@type varchar(50)
as Begin
    declare @typeId int = (select id from DocumentType where Name =
@type)
    insert Document (Number, DocumentTypeid) values (@numb, @typeId)
end
```

Листинг И.3 – Псевдоудаление документа

```
alter procedure DeleteDoc
@numb int
as Begin
    delete Document where id = (select id from Document where Number
= @numb)
End
```

Листинг И.4 – Получение списка параметров

```
alter procedure GetInfoAboutDocs
@numb int
as begin
    declare @doctypeid int = (select DocumentTypeid from Document
where Number = @numb)
    if (@doctypeid = 1)
        select DocumentType.Name, Number, DateOfBeginning,
DateOfEnd, Cost, ConsumptionAgreement.Amount as Amount,
Cocontractor.Name, MineralOil.Name from Document
        inner join Cocontractor_Document on Document.id =
Cocontractor_Document.Documentid
        inner join Cocontractor on Cocontractor_Document.Cocontractorid
= Cocontractor.id
        inner join MineralOil on Document.MineralOilid =
MineralOil.id
        inner join DocumentType on Document.DocumentTypeid =
DocumentType.id
        inner join ConsumptionAgreement on Document.id =
ConsumptionAgreement.Documentid
        where Number = @numb
    else if (@doctypeid = 2)
        select DocumentType.Name, Number, DateOfBeginning,
DateOfEnd, Cost,
SalesAgreement.Amount as Amount, Cocontractor.Name,
```

```

MineralOil.Name from Document
        inner join Coontractor_Document on Document.id =
Coontractor_Document.Documentid
        inner join Coontractor on Coontractor_Document.Coontractorid
= Coontractor.id
        inner join MineralOil on Document.MineralOilid =
MineralOil.id
        inner join DocumentType on Document.DocumentTypeid =
DocumentType.id
        inner join SalesAgreement on Document.id =
SalesAgreement.Documentid
        where Number = @numb
    else
        select DocumentType.Name, Number, DateOfBeginning,
DateOfEnd, Cost,
        LeaseAgreement.AmountOfRes as Amount, Coontractor.Name,
MineralOil.Name from Document
        inner join Coontractor_Document on Document.id =
Coontractor_Document.Documentid
        inner join Coontractor on Coontractor_Document.Coontractorid
= Coontractor.id
        inner join MineralOil on Document.MineralOilid =
MineralOil.id
        inner join DocumentType on Document.DocumentTypeid =
DocumentType.id
        inner join LeaseAgreement on Document.id =
LeaseAgreement.Documentid
        where Number = @numb
End

```

Листинг И.5 – Обновление данных документа

```

alter procedure UpdateParams
@numb int,
@dateBegin date,
@dateEnd date,
@cost money,
@amoun int,
@contrName varchar(50),
@oilName varchar(50)
as begin
    declare @docName varchar(50) = (select Name from DocumentType
where id = (select DocumentTypeid from Document where Number = @numb))
    declare @docid int = (select id from Document where Number =
@numb)
    declare @contrid int = (select id from Coontractor where Name =
@contrName)
    declare @oilod int = (select id from MineralOil where Name =
@oilName)
    declare @typeid int = (select id from DocumentType where id =
(select DocumentTypeid from Document where Number = @numb))
    INSERT JournalOfStatement VALUES (GETDATE(), @numb, @docName, 5)
    update Coontractor_Document set Coontractorid = @contrid where
Documentid = @docid
    update Document
    set DateOfBeginning = @dateBegin,
    DateOfEnd = @dateEnd,
    Cost = @cost,

```

```

        Contractorid = @contrid,
        MineralOilid = @oilod where Number = @numb
        if (@typeid = 1)
            Update ConsumptionAgreement set Amount = @amoun where
Documentid = @docid
        else if (@typeid = 2)
            Update SalesAgreement set Amount = @amoun where Documentid =
@docid
        else Update LeaseAgreement set AmountOfRes = @amoun where
Documentid = @docid

        update Document set StateId = iif(DateOfEnd >= getdate(), 1, 2)
end

```

Листинг И.6 – Добавление нескольких документов

```

CREATE TYPE NumbType AS TABLE
(
    Id int,
    Numherr int,
    Typee varchar(50)
)

alter PROCEDURE AddCollDocs
@mas NumbType READONLY
AS BEGIN
SET NOCOUNT ON;
    declare @cnt int = (select count(*) from @mas)
    declare @cur int = 1
    while (@cnt>0)
    begin
        insert into Document (Number, DocumentTypeId) values
            ((select Numherr from @mas where id = @cur), (select id from
DocumentType where Name = (select Typee from @mas where id = @cur)))
        set @cur = @cur +1
        set @cnt = @cnt-1
    end
END

```

Листинг И.7 – Получение записей журнала

```

create procedure GetInfoJournalDocs
as begin
    select DateTimeOfBeginning, DocumentNumber, DocumentType,
Reason.ShortName from JournalOfStatement
    inner join Reason on JournalOfStatement.ReasonId = Reason.id
end

```

Листинг И.8 – Добавление 1 резервуара

```

create procedure InsertRes
@numb int,
@name varchar(100),
@type varchar(50)
as Begin
    declare @typeId int = (select id from MineralOil where Type =
@type and Name = @name)
    insert Reservoir values (@numb, @typeId, null, null)
end

```

Листинг И.9 – Множественное добавление

```
create PROCEDURE AddCollRes
@mas NumbNameType READONLY
AS BEGIN
SET NOCOUNT ON;
    declare @cnt int = (select count(*) from @mas)
    declare @cur int = 1
    while (@cnt>0)
    begin
        insert into Reservoir (Number, MineralOilid) values
        ((select Numbeerr from @mas where id = @cur),
        (select id from MineralOil where Name = (select Namee from
@mas where id = @cur) and Type = (select Typee from @mas where id =
@cur)))
        set @cur = @cur +1
        set @cnt = @cnt-1
    end
END
```

Листинг И.10 – Удаление резервуара

```
create procedure DeleteRes
@numb int
as Begin
    delete Reservoir where Number = @numb
End
```

Листинг И.11 – Получение списка параметров

```
create function [dbo].[GetParamidPumpid] (@numb int)
returns table
as return(
    select id, PumpInid, PumpOutid from ReservoirParams where id =
(select ReservoirParamsid from Reservoir where Number = @numb))

create procedure [dbo].[GetInfoAboutPumpIn]
@numb int
as begin
    select ReservoirParams.PumpInid as Number, Reason.ShortName as
Statement, Pump.Speed as Speed from ReservoirParams
inner join Reservoir on ReservoirParams.id =
Reservoir.ReservoirParamsid
inner join Pump on ReservoirParams.PumpInid = Pump.id
inner join JournalOfStatement on Pump.id =
JournalOfStatement.Pumpid
inner join Reason on JournalOfStatement.Reasonid = Reason.id
where JournalOfStatement.id = (select StateId from Pump
where id = (select PumpInid from ReservoirParams where id = (select
ReservoirParamsid from Reservoir where Number = @numb)))
End

create procedure [dbo].[GetInfoAboutPumpOut]
@numb int
as begin
    select ReservoirParams.PumpOutid as Number, Reason.ShortName as
Statement, Pump.Speed as Speed from ReservoirParams
inner join Reservoir on ReservoirParams.id =
Reservoir.ReservoirParamsid
```

```

        inner join Pump on ReservoirParams.PumpOutid = Pump.id
        inner join JournalOfStatement on Pump.id =
JournalOfStatement.Pumpid
        inner join Reason on JournalOfStatement.Reasonid = Reason.id
        where JournalOfStatement.id = (select StateId from Pump
where id = (select PumpOutid from ReservoirParams where id = (select
ReservoirParamsid from Reservoir where Number = @numb)))
end

create procedure [dbo].[GetInfoAboutRes]
@numb int
As begin
    select Reservoir.Number as Number, Reason.ShortName as Statement,
MineralOil.Type as MineralOilType, MineralOil.Name as MineralOilName,
    CurVolume as CurrentLevel, MinLevel as MinLevel, MaxLevel as
MaxLevel, Procent from ReservoirParams
    inner join Reservoir on ReservoirParams.id =
Reservoir.ReservoirParamsid
    inner join MineralOil on Reservoir.MineralOilid = MineralOil.id
    inner join JournalOfStatement on Reservoir.id =
JournalOfStatement.Reservoirid
    inner join Reason on JournalOfStatement.Reasonid = Reason.id
    where ShortName = (select ShortName from Reason where id =
(select Reasonid from JournalOfStatement where id = (select max(id)
from JournalOfStatement where Reservoirid = (select id from Reservoir
where Number = @numb))))
    and JournalOfStatement.id = (select max(id) from
JournalOfStatement where Reservoirid = (select id from Reservoir where
Number = @numb))
End

```

Листинг И.12 – Получение списка резервуаров

```

alter procedure UpdateParams
create procedure GetAllRes as
begin
    select Number, Reason.ShortName, ReservoirParams.Procent from
Reservoir
    inner join ReservoirParams on Reservoir.ReservoirParamsid
=ReservoirParams.id
    inner join JournalOfStatement on Reservoir.id =
JournalOfStatement.Reservoirid
    inner join Reason on JournalOfStatement.Reasonid = Reason.id
    where ShortName in (select ShortName from Reason where id in
(select Reasonid from JournalOfStatement where id in (select
Statementid from Reservoir)))
    and JournalOfStatement.id in (select id from JournalOfStatement
where id in (select Statementid from Reservoir))
end

```

Листинг И.13 – Включить или выключить насос

```

create procedure TurnOnOffPumps
@numb int,
@Pumpid int
as begin
    declare @ParamId int = (select id from ReservoirParams where id =
(select ReservoirParamsid from Reservoir where Number = @numb))

```

```
IF (@Pumpid % 2 = 1)
begin
    update ReservoirParams
    set PumpInid = @Pumpid where id = @ParamId
end IF (@Pumpid % 2 = 0)
begin
    update ReservoirParams
    set PumpOutid = @Pumpid where id = @ParamId
end
end
```

Приложение К
(рекомендуемое)
Исходные коды скриптов триггеров

Листинг К.1 – Добавление документа

```
ALTER TRIGGER [dbo].[AddReservoir]
ON [dbo].[Reservoir] after insert
AS BEGIN
    declare @res int = (select MAX(id) from Reservoir)
    INSERT Pump VALUES (1, 1,null)
    INSERT Pump VALUES (1, 2,null)
    declare @pumpInid int = (select max(id)-1 from Pump)
    declare @pumpOutid int = (select max(id) from Pump)
    INSERT ReservoirParams VALUES (0, 0, 100, @pumpInid, @pumpOutid,
0)
    update Reservoir
    set ReservoirParamsid = (select max(id) from ReservoirParams)
where id = @res
    INSERT JournalOfStatement VALUES (GETDATE(), 7, @res, @pumpInid)
    INSERT JournalOfStatement VALUES (GETDATE(), 7, @res, @pumpOutid)
    Update Reservoir set Statementid = (select MAX(id) from
    JournalOfStatement where Reservoirid = @res and Reasonid = 7)
    where id = @res
    Update Pump set StateId = (select MAX(id) from JournalOfStatement
    where Pumpid = @pumpInid and Reasonid = 7) where id = @pumpInid
    Update Pump set StateId = (select MAX(id) from JournalOfStatement
    where Pumpid = @pumpOutid and Reasonid = 7) where id = @pumpOutid
END
```

Листинг К.2 – Псевдоудаление документа

```
create TRIGGER DeleteDocs
ON Document instead of delete
AS BEGIN
    declare @id int = (select deleted.id from deleted)
    declare @numb int = (select deleted.Number from deleted)
    declare @typeid int = (select deleted.DocumentTypeId from
deleted)
    declare @typeName varchar(50) = (select Name from DocumentType
where id = @typeid)
    INSERT JournalOfStatement VALUES (GETDATE(), @numb, @typeName, 4)
    update Document set StateId = 4 where id = @id
end
```

Листинг К.2 – Добавление 1 резервуара

```
CREATE TRIGGER AddReservoir
ON Reservoir after insert
AS BEGIN
    declare @res int = (select MAX(id) from Reservoir)
    INSERT Pump VALUES (1, 1,null)
    INSERT Pump VALUES (1, 2,null)
    declare @pumpInid int = (select max(id)-1 from Pump)
    declare @pumpOutid int = (select max(id) from Pump)
    INSERT ReservoirParams VALUES (0, 0, 100, @pumpInid, @pumpOutid,
0)
    update Reservoir
    set ReservoirParamsid = (select max(id) from ReservoirParams)
where id = @res
    INSERT JournalOfStatement VALUES (GETDATE(), 7, @res, @pumpInid)
    INSERT JournalOfStatement VALUES (GETDATE(), 7, @res, @pumpOutid)
    Update Reservoir set Statementid = (select MAX(id) from
```

```

JournalOfStatement where Reservoirid = @res and Reasonid = 7) where id
= @res
    Update Pump set StateId = (select MAX(id) from JournalOfStatement
where Pumpid = @pumpInid and Reasonid = 7) where id = @pumpInid
    Update Pump set StateId = (select MAX(id) from JournalOfStatement
where Pumpid = @pumpOutid and Reasonid = 7) where id = @pumpOutid
END

```

Листинг К.3 – Удаление резервуара

```

CREATE TRIGGER DeleteReservoir
    ON Reservoir after delete
    AS BEGIN
        INSERT JournalOfStatement VALUES (GETDATE(), 8, (select id from
deleted), (select PumpInid from ReservoirParams where id = (select
ReservoirParamsid from deleted)))
        INSERT JournalOfStatement VALUES (GETDATE(), 8, (select id from
deleted), (select PumpOutid from ReservoirParams where id = (select
ReservoirParamsid from deleted)))
        declare @pumpIn int, @pumpOut int
        set @pumpIn = (select PumpInid from ReservoirParams where id =
(select ReservoirParamsid from deleted))
        set @pumpOut = (select PumpOutid from ReservoirParams where id =
(select ReservoirParamsid from deleted))
        delete ReservoirParams where id = (select ReservoirParamsid from
deleted)
        delete Pump where id = @pumpIn
        delete Pump where id = @pumpOut
    END

```

Листинг К.4 – Включить или выключить насос

```

ALTER TRIGGER [dbo].[TurnOnOffPump]
ON ReservoirParams instead of update
AS BEGIN
    IF (UPDATE (PumpOutid))
    begin
        declare @lastOut int
        declare @resOut int
        declare @pumpOut int
        set @pumpOut = (select PumpOutid from ReservoirParams where
id = (select id from inserted))
        set @resOut = (select id from Reservoir where
ReservoirParamsid = (select id from inserted))
        set @lastOut = (select max(id) from JournalOfStatement where
Reasonid in (1, 2) and (Pumpid = @pumpOut) and (Reservoirid =
@resOut))
        if ((select Reasonid from JournalOfStatement where id =
@lastOut) = 2)
        begin
            INSERT JournalOfStatement VALUES (GETDATE(), 1,
@resOut, @pumpOut)
            INSERT JournalOfStatement VALUES (GETDATE(), 3,
@resOut, @pumpOut)
        end
        else if ((select Reasonid from JournalOfStatement where id =
@lastOut) = 1)
        begin

```

```

        INSERT JournalOfStatement VALUES (GETDATE(), 2, @resOut,
@pumpOut)
        INSERT JournalOfStatement VALUES (GETDATE(), 4, @resOut,
@pumpOut)
    end
    else begin
        INSERT JournalOfStatement VALUES (GETDATE(), 1, @resOut,
@pumpOut)
        INSERT JournalOfStatement VALUES (GETDATE(), 3, @resOut,
@pumpOut)
    end
    Update Pump set StateId = (select MAX(id) from JournalOfStatement
where Pumpid = @pumpOut and Reasonid in (1,2)) where id = @pumpOut
    Update Reservoir set Statementid = (select MAX(id) from
JournalOfStatement where Reservoirid = @resOut and Reasonid in (3,4))
where id = @resOut
end
    else IF (UPDATE (PumpInid))
    begin
        declare @lastIn int
        declare @resIn int
        declare @pumpIn int
        set @pumpIn = (select PumpInid from ReservoirParams where id
= (select id from inserted))
        set @resIn = (select id from Reservoir where
ReservoirParamsid = (select id from inserted))
        set @lastIn = (select max(id) from JournalOfStatement where
Reasonid in (1, 2) and (Pumpid = @pumpIn) and (Reservoirid =
@resIn))
        if ((select Reasonid from JournalOfStatement where id =
@lastIn) = 2)
        begin
            INSERT JournalOfStatement VALUES (GETDATE(), 1,
@resIn, @pumpIn)
            INSERT JournalOfStatement VALUES (GETDATE(), 3,
@resIn, @pumpIn)
        end
    else if ((select Reasonid from JournalOfStatement where id =
@lastIn) = 1)
    begin
        INSERT JournalOfStatement VALUES (GETDATE(), 2, @resIn,
@pumpIn)
        INSERT JournalOfStatement VALUES (GETDATE(), 4, @resIn,
@pumpIn)
    end
    else begin
        INSERT JournalOfStatement VALUES (GETDATE(), 1, @resIn,
@pumpIn)
        INSERT JournalOfStatement VALUES (GETDATE(), 3, @resIn,
@pumpIn)
    end
    Update Pump set StateId = (select MAX(id) from JournalOfStatement
where Pumpid = @pumpIn and Reasonid in (1,2)) where id = @pumpIn
    Update Reservoir set Statementid = (select MAX(id) from
JournalOfStatement where Reservoirid = @resIn and Reasonid in (3,4))
where id = @resIn end
END

```