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

Кафедра «Прикладная математика и информатика»
(наименование)

09.04.03 Прикладная информатика
(код и наименование направления подготовки)

Управление корпоративными информационными процессами
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему «Методы и модели обмена данными в многоплатформенной корпоративной информационной системе»

Обучающийся К.П. Ефременко
(Инициалы Фамилия) (личная подпись)

Научный руководитель канд.пед.наук, доцент О.М. Гущина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Оглавление

Введение

Информационная корпоративная система представляет собой всю инфраструктуру компании, задействованную в процессе управления всеми бизнес-процессами компании.

Первые корпоративные информационные системы появились в 60-х годах прошлого века. Одной из самых ранних таких систем была Master Planning Scheduling (MPS), разработанная для организации производственных планов на предприятиях. MPS основывалась на актуальных данных о спросе для составления графиков выпуска продукции. Основными факторами, вызвавшими создание MPS, стали задержки в производственных процессах и поставках материалов, перенесённых на более ранний срок.MPS, работала по принципу: формирование план продаж с распределением по временным периодам; составление план-графика закупки и собственного производства.

Одной из центральных задач MPS было точное предсказание объёмов и сроков поставок, что требовало более детального долгосрочного планирования потребительского спроса, периода производства и оптимального использования складских ресурсов. С течением времени появились и другие корпоративные информационные системы. В начале 90-х годов XX века была разработана ERP-система, представляющая собой комплекс интегрированных приложений. Эта система позволяет автоматизировать планирование, учёт, контроль и анализ ключевых бизнес-процессов компании. Впоследствии на базе ERP разработали CRM-систему — инструмент для управления взаимоотношениями с клиентами. Многие эксперты рассматривают CRM не только как набор технологий и информационную систему, но и как стратегию ведения бизнеса. В стандартный набор функций CRM входят модули для управления контактами и автоматизации продаж. Данные исторические факты свидетельствуют об актуальности корпоративной системы и обмена данными в ней в прошлом времени. Также на базе этих моделей строят их современные системы.

В наше время актуальность темы исследования обосновывается тем, что использование корпоративных информационных систем характеризует уровень развития общества, его динамику, возможность интеграции в мировое сообщество, способность специалистов выдерживать темпы научно-технического прогресса. Поэтому очень важно изучать методы создания корпоративных информационных систем, их специфику, разнообразие, платформы для выбора программирования, методов и моделей обменов данными, а также создание приложений корпоративных программ.

Объектом исследования являются процесс обмена данными в многоплатформенной корпоративной информационной системе.

Предметом исследования является методы и модели создания многоплатформенной корпоративной информационной системы.

Целью данной научной диссертации является разработка и обоснование методов и моделей обмена данными в многоплатформенной корпоративной информационной системе, базирующейся на платформе 1С:Предприятие, с учетом анализа существующих подходов и практик в данной сфере. В рамках исследования предполагается провести детальный анализ текущего состояния и тенденций развития многоплатформенных систем, а также разработать прототип корпоративной информационной системы, которая обеспечит эффективный и безопасный обмен данными между различными платформами.

Для успешного достижения заявленной цели требуется выполнить ряд задач:

- провести детальное изучение и анализ предметной области; исследовать различные подходы к созданию архитектуры корпоративных информационных систем;
- выполнить анализ методов и моделей для обмена данными внутри корпоративной системы; описать выбранный метод, обосновав его предпочтительность и эффективность, разработать процесс обмена данными в корпоративной среде;

– рассмотреть концептуальное и логическое моделирование; определить структурно-функциональную схему ИТ-решения; разработать модель на базе «1С:Предприятие 8. Язык программирования»;
– провести тестирование и испытание информационной системы; определить область внедрения разработанной модели.

Гипотеза исследования: использование методов и моделей обмена данными в многоплатформенной корпоративной информационной системе позволит повысить эффективность разработки и внедрения корпоративной информационной системы на базе платформы 1С:Предприятие.

Методологическая основа исследования опирается на ключевые труды в области обмена данными, созданные как российскими, так и международными исследователями, которые изучают аспекты интеграции, консолидации данных и технологии их обмена. Для выполнения задач данной диссертации применялись методы системного анализа, моделирования, а также методики объектно-ориентированного проектирования и программирования.

Научная новизна работы заключается в разработке и обосновании нового метода обмена данными для многоплатформенной корпоративной информационной системы на основе платформы «1С:Предприятие 8», что обеспечило повышение производительности, надежности и совместимости с современными архитектурными подходами.

Практическая значимость данного исследования определяется внедрением корпоративной системы, которая способствует ускорению обмена данными между различными подразделениями организации и автоматизации всех ключевых бизнес-процессов.

Выпускная квалификационная работа охватывает 99 страниц, включая текстовый материал, модели, графики, таблицы, иллюстрации и код программ. Структура работы состоит из введения, четырёх глав, выводов по каждой из глав, заключительной части, списка использованных ресурсов и приложения.

Глава 1 Анализ методов и моделей обменов данными в многоплатформенной корпоративной информационной системе

В главе проводится анализ методов и моделей обменов данными в многоплатформенной корпоративной информационной системе, с целью анализа теоретических знаний об особенностях строения корпоративной информационной системы.

Для решения поставленной задачи необходимо проанализировать цели создания, виды и особенности корпоративных информационных систем, проанализировать подходы построения архитектуры корпоративных информационных систем, методы обмена данными в корпоративно системе и модели обмена данными в корпоративно системе.

1.1 Описание и анализ предметной области

«Корпоративная информационная система - это открытая интегрированная автоматизированная система реального времени по автоматизации бизнес-процессов компании всех уровней, в том числе, и бизнес-процессов принятия управленческих решений. При этом степень автоматизации бизнес-процессов определяется исходя из обеспечения максимальной прибыли компании» [17].

Корпоративные информационные системы, далее сокращенно КИС предназначены для [8]:

- оснащения бизнес-процессов всех или почти всех всего предприятия и его филиалов;
- структурирования и анализа данных о компании и внешней среде для решения задач от первичной информации до поддержки принятия решений высшим руководством, а также задач всех направлений деятельности и технологических операций.

– Корпоративные информационные системы (КИС) объединяют инструменты для управления документами, поддерживают разные информационные аспекты, обеспечивают коммуникации, совместную работу сотрудников и содержат различные технологические продукты. Они представляют собой совокупность программно-аппаратных платформ и приложений, обеспечивающих выполнение задач компании. КИС «помогает контролировать процессы и создает базу данных для принятия эффективных управленческих решений» [22].
– Также КИС позволяет эффективно управлять материально-техническими, финансовыми, технологическими и интеллектуальными ресурсами компании. Это позволяет «получить максимальную прибыль в компании, вследствие чего появляется возможность повысить уровень заработной платы сотрудников» [25].

КИС модно классифицировать по различным признакам (таблица 1)

Таблица 1 - Классификация КИС

Признак	Описание
Функциональные возможности	КИС могут выполнять функции бухгалтерского учета, управления складом, управления ассортиментом, управления закупками, управления финансовыми потоками, управления производственной деятельности, контроль документооборота, общения с клиентами.
Масштаб предприятия	Стоимость и срок внедрения и разработки КИС зависит от масштаба предприятия. Чем больше, тем дольше и дороже происходит разработка и внедрение информационной системы.
Тиражируемые, полузаказные и заказные.	Тиражируемые КИС создают малое количество бизнес-процессов компании. Часто ими пользуются в малом бизнесе, т.к. их не сложно освоить и имеют низкую стоимость. Полузаказные КИС характеризуются настройкой под конкретною компанию и гибкостью. Они дороже тиражируемых КИС, но дешевле заказных КИС. Заказные КИС создаются для конкретного предприятия, которое является монополистом в своей отрасли. Данная разработка требует больших финансовых вложений [18].

Продолжение таблицы 1

Признак	Описание
Архитектура КИС	«Локальные КИС, в которых базы данных, система управления базами данных, клиентские приложения работают только на одном автоматизированном рабочем месте. Распределённые КИС, в которых компоненты распределены по нескольким автоматизированным рабочим местам в сети. Они бывают нескольких видов. Файл-серверными КИС и клиент-серверными КИС» [29].

КИС должна соответствовать определенным требованиям [28]:

- соответствие потребностям заказчика и его финансовым возможностям. Например, для малого бизнеса не стоит создавать дорогую заказную КИС, а тиражируемая КИС подходит;
- каждый модуль должен иметь интеграцию с другими модулями;
- КИС автоматизирует управление на предприятии на всех участках работы, на каждом рабочем месте и во всех филиалах и дочерних фирмах компании;
- поддержка принятых стандартов управления;
- Корпоративная информационная система (КИС) должна быть структурирована таким образом, чтобы программирование было разделено на отдельные функциональные блоки. Этот подход позволяет снизить издержки при создании КИС, так как становится возможным избежать приобретения системных компонентов, не соответствующих требованиям системы или компании, или которые могут быть исключены на начальном этапе разработки;
- открытость для доработки, которая почти всегда требуется. Поэтому «КИС должна быть открытой для включения дополнительных модулей, которые придется разрабатывать после анализа эксплуатации данной системы. Также могут внедрять уже используемые модули, которые зарекомендовали себя в компании» [29];

- адаптивность к росту, образованию дочерних фирм и филиалов. В таких ситуациях КИС должна дорабатываться под измененные условия компании. Изменения могут быть связаны с изменением видов выпускаемой продукции, услуг или изменением структуры компании, тут КИС также перестраивается без препятствий, временных и крупных денежных затрат;
- устойчивость функционирования системы даже в случае выхода из строя отдельных ее компонентов;
- безопасность включает защиту данных от утраты, предотвращение несанкционированного доступа к системе, а также контроль и разграничение прав на доступ.;
- использование системы управления базами данных и технологий клиент-серверной обработки данных;
- обеспечение интернетом;
- простой пользовательский интерфейс для освоения, т.е. на русском языке и наличие специализированных курсов для обучения сотрудников;
- поддержка разработчика, которая состоит из бесплатного или по акционной программе получения новых версий программного обеспечения и консультации по горячей линии;
- сопутствие специалистом при необходимости внесения изменений в систему, неполадок и усовершенствовании ее [20].

«Процесс управления - это целенаправленное воздействие управляющей системы на управляемую, ориентированное на достижение определенной цели и использующее информационный поток. Задача управления - это наилучшая организация преобразования поступающих на вход ресурсов в конечный продукт» [23].

Эффективность управления предприятием зависит от следующих основных факторов:

- точность и полнота исходной информации;

- актуальность исходной информации;
- качество принятого решения;
- скорость доведения до исполнителей принятого решения;
- эффективность контроля руководителя над исполнением при принятии решения [25].

Всего существует четыре основных уровня управления и соответствующие им системы:

- технологический уровень, к которому относятся автоматизированная система управления, технология программирования, системы обработки данных. Автоматизированная система управления производственными процессами занимается задачами по их организации, охватывая ключевые производственные операции и управление логистикой на входе и выходе. Для выполнения этих функций используются «MES-системы, которые помогают разработать процедуры для планирования на среднесрочный период, а также анализа и координации деятельности» [1];
- оперативный уровень включает в себя такие программы, как MES и системы обработки данных (СОД). «СОД представляют собой технологии, разработанные для оперативной обработки информации и предназначенные для управления бизнес-процессами предприятия на оперативном уровне. Главная функция СОД заключается в создании элементарных бизнес-процессов, таких как учет прихода и расхода материалов на складах и в производственных цехах, осуществление платежей за материальные ресурсы и предоставленные услуги через банковские учреждения, а также ведение учета рабочего времени сотрудников» [14];
- тактический уровень, к которому относятся автоматизированная система управления и информационная система управления;
- стратегический уровень охватывает автоматизированные системы управления предприятием и системы поддержки принятия решений.

Эти системы помогают в решении вопросов, связанных с объёмом и качеством производства, а также выведением нового продукта на рынок. Кроме того, они «способствуют поиску новых рынков сбыта и определению оптимальных источников финансирования» [2];

«Корпоративная информационная система охватывает компьютерную инфраструктуру предприятия и взаимосвязанные на ее основе подсистемы, которые способствуют решению задач компании.

В качестве таких подсистем могут быть:

- Информационно-справочные системы, включая гипертекстовые и геоинформационные платформы;
- система управления документооборотом;
- система изменения информации в базах данных;
- система поддержки принятия решений» [11].

Информационные системы можно классифицировать по способу их организации следующим образом: системы с архитектурой файл-сервер, клиент-серверные системы, трехуровневые системы, а также системы, использующие интернет или интранет технологии.

Сервер можно определить как любую систему, которая может быть либо отдельным компьютером с установленным на нем специализированным программным обеспечением, либо отдельным программным компонентом в составе общего программного обеспечения. Основная функция сервера заключается в предоставлении вычислительных ресурсов другим системам, которые называются клиентами[12].

Существующие решения для обмена данными в многоплатформенных корпоративных информационных системах с оценкой их эффективность представлены в таблице 2.

Таблица 2 – Эффективность методов обмена

Метод	Описание	Преимущества	Недостатки
RESTful API	является одним из самых популярных методов для обмена данными между веб-сервисами. Использует HTTP(S) протоколы и обычно форматирует данные в JSON или XML.	Широко используется и хорошо поддерживается. Простота и легкость в реализации. Поддерживает стандартный HTTP, что облегчает интеграцию с веб-сервисами.	Ограниченная поддержка состояния (статлес). Производительность может быть низкой из-за текстового формата JSON или XML.
GraphQL	Запросный язык и среда выполнения для API, который позволяет клиенту запрашивать только те данные, которые ему необходимы.	Гибкость в запросах: клиенты могут запрашивать только те данные, которые им нужны. Сокращает количество запросов к серверу.	Более сложная настройка и обучение по сравнению с REST. Может потребовать вложенных запросов, что увеличивает нагрузку на сервер.
WebSocket	Протокол для двусторонней связи в режиме реального времени по одному TCP-соединению.	Отлично подходит для обрабатывания данных реального времени.	Избыточен для простых задач запроса-ответа.
1С:Предприятие и РИБ	Ориентированное на российские компании решение для автоматизации бизнес-процессов, включая метод распределенных информационных баз.	Глубокая интеграция с бухгалтерскими и управленческими системами.	Требуется высокая степень профессиональной квалификации для настройки и сопровождения.

При оценке эффективности каждого решения следует учитывать такие факторы, как сложность системы, требования к безопасности, необходимая скорость выполнения, и легкость интеграции с другими системами. Для многоплатформенных корпоративных систем обычно требуется баланс между гибкостью, производительностью и простотой поддержки, что может повлиять на выбор метода обмена данными в зависимости от специфики проекта.

Анализ современных тенденций в данной области позволяет определить ключевые направления, которые влияют на обмен данными и эффективность функционирования таких систем.

- Облачные технологии – представляют собой один из ключевых драйверов цифровой трансформации в корпоративной сфере. Они позволяют компаниям арендовать ИТ-ресурсы на базе платных подписок, обеспечивая при этом доступ к вычислительным мощностям, хранению данных и другим сервисам через интернет. Это значительно снижает издержки на поддержание собственной ИТ-инфраструктуры и упрощает масштабирование бизнес-процессов. Облака также обеспечивают высокий уровень отказоустойчивости и доступны из любой точки мира, что особенно важно в условиях удаленной работы.
- «Микросервисная архитектура – создание приложения в виде набора независимых сервисов, каждый из которых реализует определенные бизнес-функции. Это подход к разработке позволяет гибко настраивать систему, масштабировать ее и упрощать процесс развертывания обновлений. Каждый микросервис может быть написан на разных языках программирования и использовать различные технологии хранения данных, что дает командам разработчиков высокий уровень свободы и адаптивности» [11].
- Интероперабельность и стандарты - корпоративные информационные системы сталкиваются с проблемой взаимодействия между различными платформами и технологиями. Интероперабельность и стандартизация становятся ключевыми аспектами для обеспечения эффективного обмена данными. Использование общепринятых стандартов, таких как REST, SOAP, JSON и XML, способствует упрощению интеграции и совместимости между различными составляющими КИС.

- Большие данные и аналитика - Тенденция к использованию больших данных и аналитики в бизнес-процессах требует обновления архитектуры корпоративных систем для обработки огромных объемов информации. Интеграция аналитических инструментов, основанных на искусственном интеллекте и машинном обучении, позволяет улучшить принятие решений и оптимизировать процессы обмена данными.
- Безопасность и защита данных - В условиях увеличения числа угроз информационной безопасности защита данных становится неотъемлемой частью многоплатформенных КИС. Современные системы должны обеспечивать надежную аутентификацию, шифрование и контроль доступа к информации. Компании инвестируют в решения по управлению идентификацией и доступом (IAM), а также в технологии блокчейн для обеспечения безопасности данных.
- Интернет вещей (IoT) - Интеграция Интернета вещей в корпоративные информационные системы расширяет возможности сбора и обмена данными. IoT устройства генерируют огромное количество данных в режиме реального времени, что требует изменений в архитектуре КИС для эффективной обработки и анализа информации. Это создает новые вызовы и возможности для бизнеса, повышая скорость и точность бизнес-процессов.

1.2 Анализ подходов построения архитектуры корпоративных информационных систем

Архитектура: это структурный дизайн системы, включающий компоненты, их взаимосвязи, и принципы, которыми руководствуются для достижения определенных целей. Архитектура может включать в себя

различные уровни, такие как сетевой уровень, уровень приложений, уровня данных и т.д.

Исследование различных методов и моделей обмена данными позволяет выявить наиболее подходящие решения для каждой конкретной организации. Однако, чтобы эти решения были действительно эффективными, они должны базироваться на тщательно продуманной архитектуре КИС, следование основным принципам, поможет добиться эффективности гибкости и устойчивости.

- Масштабируемость: Архитектура должна обеспечивать возможность роста вместе с бизнесом, позволяя легко добавлять новых пользователей, процессы и данные без необходимости в значительных структурных изменениях. Даже если на начальном этапе объемы компании небольшие, система КИС должна быть рассчитана на высокие нагрузки в будущем
- Интероперабельность: для эффективного взаимодействия различных компонентов и технологий внутри и вне организации система должна поддерживать стандартизацию протоколов и форматов данных. Ключевую роль играет соблюдение стандартов связи: использование протоколов, таких как HTTP, REST и SOAP, помогает интегрировать разнообразные приложения и сервисы. Важная составляющая – унификация форматов данных. Стандарты вроде XML, JSON или EDIFACT облегчают обмен информацией между системами, минимизируя сложные преобразования, что снижает вероятность ошибок и увеличивает скорость взаимодействия.
- Безопасность: Архитектура должна предусматривать надежные механизмы защиты данных и обеспечения безопасности, способные адаптироваться к новым угрозам. Это включает в себя управление доступом, шифрование информации и мониторинг безопасности.

- Гибкость и адаптивность: Архитектура должна быть способна быстро адаптироваться к изменениям бизнеса или технической среды без значительных затрат времени и ресурсов.
- Управляемость и мониторинг: Одним из ключевых аспектов является наличие средств для эффективного управления и мониторинга системы, что позволяет своевременно выявлять и устранять возникающие проблемы.
- Производительность и надежность: Система должна демонстрировать высокую производительность и стабильность, сводя к минимуму время простоя и обеспечивая непрерывную работу критически важных процессов.
- Целостность данных: важно, чтобы архитектура обеспечивала точность и целостность данных, предупреждая их искажение или потерю в процессе хранения и передачи.
- Экономическая эффективность: Архитектурные решения должны достигать оптимального соотношения между стоимостью и функциональностью системы, учитывая также будущие расходы на её поддержание и развитие.
- Соответствие нормативным требованиям: Архитектура должна соответствовать всем применимым нормативным и отраслевым стандартам и требованиям, включая те, которые действуют в различных юрисдикциях и отраслевых сегментах.
- Поддержка мобильности и удаленного доступа: Необходима возможность работы с мобильными устройствами и удаленный доступ к информации, что особенно важно в условиях глобализации и распределенной рабочей силы. Например, платформа 1С:Предприятие предоставляет функции для работы в удаленном режиме и на мобильных устройствах, предлагая веб-клиент для работы через браузер и приложения для iOS и Android.

– Управление изменениями: Система должна предусматривать методологию управления изменениями, включая интеграцию новых технологий и процессов без нарушения текущих операций [32].

Следуя этим принципам при проектировании, КИС будет отвечать всем требованиям бизнеса, для наиболее эффективной работы предприятия. Но в современном мире ИТ-технологий, часто приходится пользоваться уже готовыми, и проверенными разработками, несмотря на то что каждая компания требует индивидуального подхода, для проектирования КИС, некоторые архитектурные подходы себя очень хорошо показали.

Разберем существующие архитектурные подходы.

Клиент-серверная архитектура — это один из самых распространенных подходов, который подразумевает разделение системы на клиентскую и серверную части. Клиенты запрашивают данные и сервисы у серверов, которые обрабатывают эти запросы и возвращают результаты. Такая архитектура позволяет централизовать управление данными и обеспечить их безопасность, а также упрощает обслуживание и обновление системы.

В клиент-серверной архитектуре (рисунок 1) сервер выполняет роль центрального узла, который обрабатывает запросы от клиентов, управляет данными и предоставляет необходимые сервисы. Клиенты, в свою очередь, взаимодействуют с сервером через сеть, отправляя запросы для получения данных или выполнения операций.

Технические особенности клиент-серверной архитектуры включают:

– Сервера: мощные компьютеры или виртуальные машины, которые обеспечивают централизованную обработку данных, управление ресурсами и обеспечение безопасности. Серверы могут использовать разные операционные системы, такие как Windows Server, Linux и прочие.
– Клиенты: устройства конечных пользователей, которые могут быть настольными компьютерами, лэптопами, планшетами или смартфонами. Клиенты обычно имеют меньшую вычислительную

мощность по сравнению с серверами и полагаются на сервер для обработки тяжелых операций.

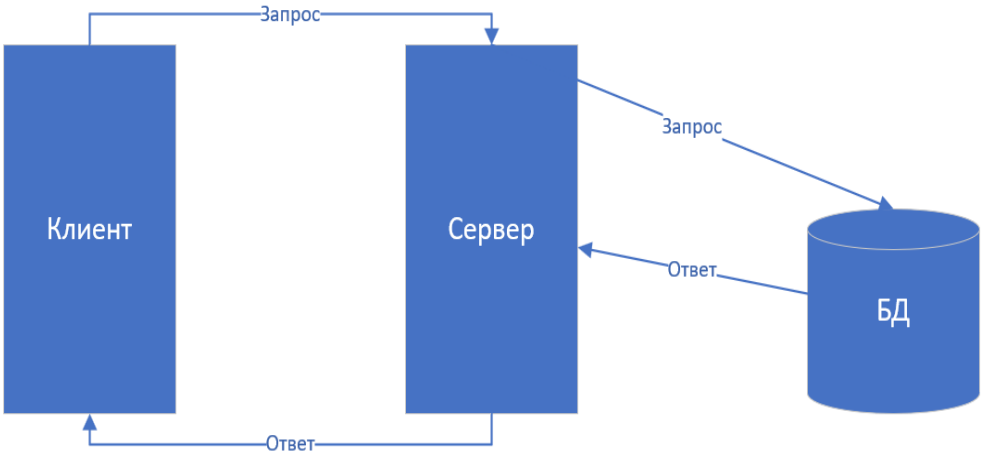

Рисунок 1 – Клиент-серверная архитектура

Аппаратная часть системы базируется на сети, которая соединяет клиента и сервер. Это может быть локальная сеть (LAN) в пределах одного офиса или глобальная сеть (WAN) для распределенных территориально систем.

На диаграмме последовательности (рисунок 2) мы видим, как клиент взаимодействует с сервером в рамках бесконечного цикла запросов и ответов. Клиент отправляет запрос серверу (сообщение "request"), а сервер, находясь в состоянии постоянного ожидания (операция "listen"), получает и обрабатывает этот запрос.

Процесс на стороне сервера выполняется параллельно: сервер одновременно слушает новые запросы и обрабатывает поступившие, что показано блоком "par". После завершения обработки сервер отправляет результат обратно клиенту (сообщение "result"). Цикл с условием [true] указывает на то, что процесс повторяется бесконечно, предполагая постоянный обмен данными между клиентом и сервером.

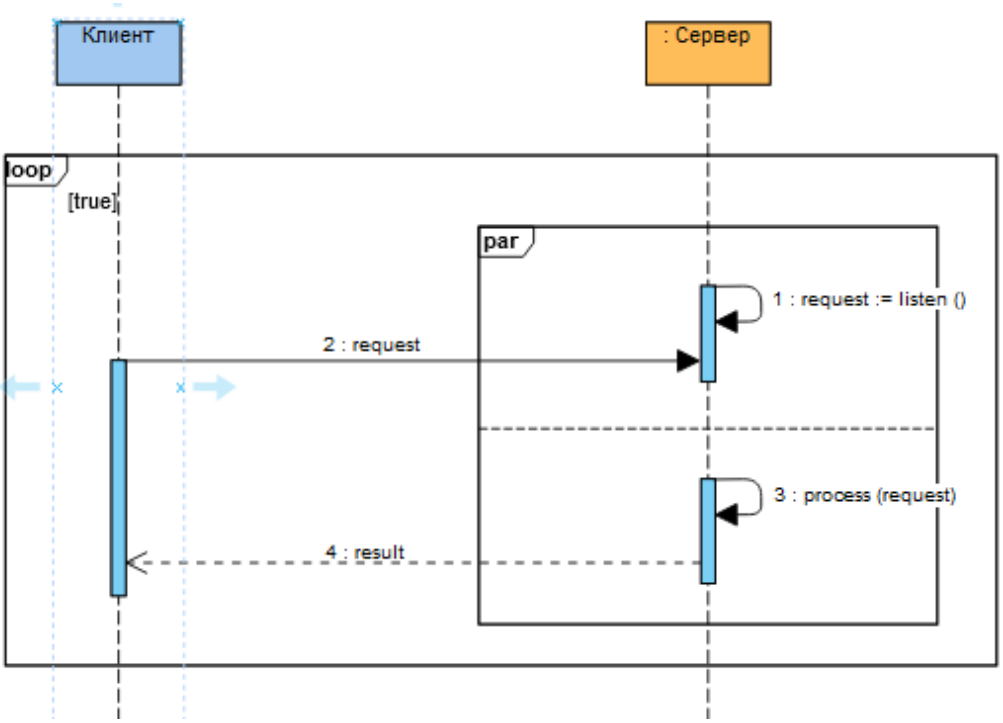

Рисунок 2 – Диаграммапоследовательности клиент-серверной архитектуры

Преимущества клиент-серверной архитектуры:

- Централизованное управление данными: Серверы централизуют хранение и управление данными, что обеспечивает их целостность и упрощает администрирование.
- Упрощенное обслуживание: Все обновления программного обеспечения, исправления и модернизации проводятся на сервере, что устраняет необходимость внесения изменений на каждом клиенте по отдельности.
- Безопасность: Центральное управление доступом на сервере упрощает реализацию политики безопасности и мониторинга, облегчает защиту данных от несанкционированного доступа.
- Распределение нагрузки: Серверы могут разделять ресурсы между многими клиентами, оптимизируя использование аппаратных средств.

– Масштабируемость: Легкость добавления новых клиентов без значительных изменений в существующей инфраструктуре.

Недостатки клиент-серверной архитектуры:

– Узкое место в системе: Серверы могут стать узким местом, если они не справятся с возросшим числом запросов. Это может повлиять на производительность системы.
– Зависимость от сети: Клиент должен иметь стабильное подключение к серверу для функционирования приложения, и проблемы с сетью могут привести к снижению доступности системы.
– Сложность серверной стороны: Установка, настройка и поддержка серверной части могут быть сложными и потребовать значительных ресурсов.
– Высокие затраты на инфраструктуру: Необходимость в мощных серверах и средствах защиты может привести к увеличенным расходам на аппаратное и программное обеспечение.
– Риски единой точки отказа: В случае выхода из строя серверов или их компонентов может возникнуть полное прерывание предоставления услуг клиентам.

Микросервисная архитектура — это стиль разработки программного обеспечения, в котором приложение разделяется на небольшие, независимые модули или "микросервисы". Каждый из этих «микросервисов отвечает за выполнение определенной бизнес-логики и может разрабатываться, масштабироваться независимо от остальных. Это контрастирует с традиционной монолитной архитектурой, где все компоненты приложения тесно связаны и управляются как единое целое» [33].

Взаимодействие между микросервисами (рисунок 3) организуется через легковесные сетевые протоколы, такие как HTTP/REST или gRPC. Это позволяет сервисам обмениваться данными, сохраняя при этом свою независимость. API играет ключевую роль, обеспечивая стандартизацию и упрощая интеграцию различных компонентов. Второй важный аспект — это

возможность использования разных технологий для разных микросервисов. Разработчики могут выбирать наиболее подходящий язык программирования, фреймворк или базу данных для каждого сервиса, что увеличивает гибкость и позволяет оптимизировать производительность конкретных частей приложения.

Рисунок 3 – Микросервисная архитектура

Микросервисы часто разворачиваются с использованием контейнеризации, например, с применением Docker. Это упрощает управление зависимостями и окружениями и обеспечивает консистентность развертывания. Для оркестрации контейнеров часто применяется Kubernetes или аналогичные инструменты, которые автоматизируют задачи, связанные с масштабированием и отказоустойчивостью.

На уровне данных микросервисная архитектура предполагает возможность использования отдельных баз данных для каждого сервиса, что уменьшает зависимость от централизованной базы данных и позволяет гибко

выбирать тип хранилища данных в зависимости от требований (например, реляционные SQL базы, NoSQL решения и т.д.).

Аппаратный уровень для микросервисных приложений часто реализуется в облачной среде. Облачные платформы, такие как AWS, Google Cloud и Microsoft Azure, предлагают сервисы для динамического масштабирования ресурсов в ответ на изменения нагрузки, а также обеспечивают высокий уровень доступности и отказоустойчивости. Хотя такая инфраструктура может быть более сложной, она обеспечивает значительную гибкость, масштабируемость и устойчивость к сбоям. Это делает «микросервисную архитектуру особенно привлекательной для современных приложений, которые требуют высокой нагрузочной способности и распределённости» [31].

На диаграмме последовательности (рисунок 4) продемонстрировано взаимодействие между микросервисами двух различных систем во время обработки пользовательского запроса, поступающего через браузер.

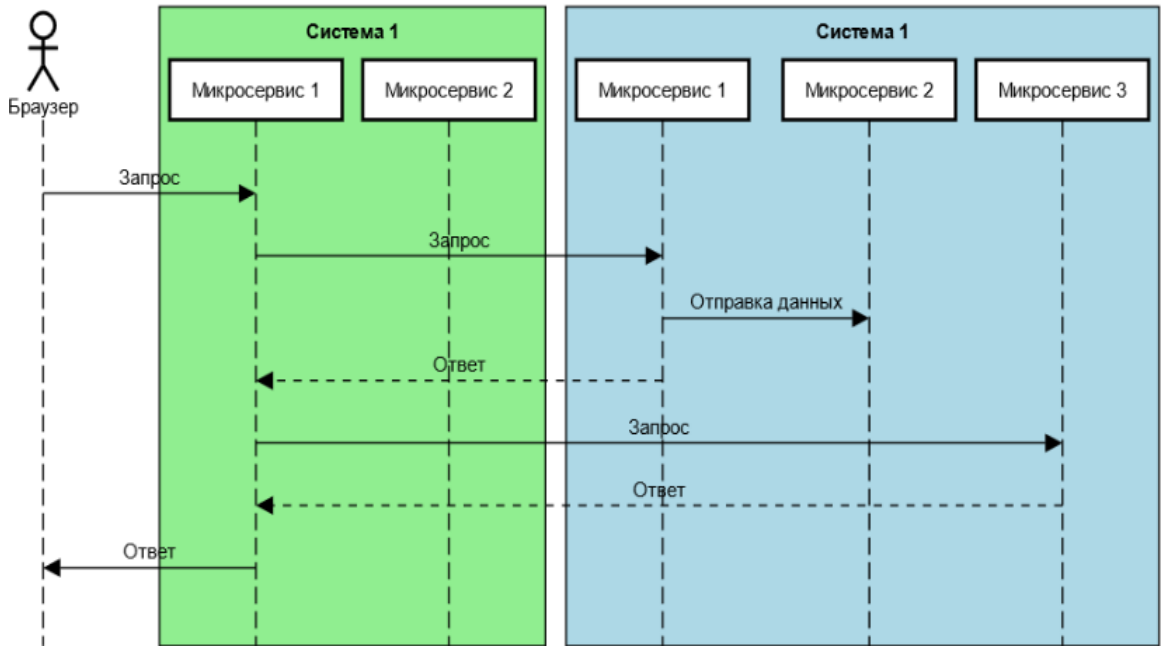

Рисунок 4 – Диаграмма последовательности микросервисы

Ключевые аспекты микросервисной архитектуры, отраженные на диаграмме:

- Изоляция микросервисов: Каждый микросервис выполняет определённую функцию и отвечает за конкретный этап обработки запроса, что позволяет каждому модулю развиваться и масштабироваться независимо от других.
- Обмен данными между системами: Диаграмма иллюстрирует взаимодействие микросервисов из разных систем, где они обмениваются данными для выполнения задач, требующих объединённых ресурсов нескольких систем.
- Гибкость и масштабируемость: Архитектурный подход, показанный на диаграмме, обеспечивает лёгкость в добавлении новых микросервисов и расширении системы, не нарушая функционирование других компонентов, что упрощает внесение изменений в отдельные сервисы.

Эта диаграмма (рисунок 4) наглядно иллюстрирует процесс обработки запроса в микросервисной архитектуре, когда компоненты разных систем обмениваются данными для выполнения комплексных операций.

Преимущества микросервисной архитектуры:

- Масштабируемость: Каждая отдельная часть приложения может быть масштабирована индивидуально, в зависимости от текущей нагрузки. Это позволяет сократить издержки и повысить эффективность использования ресурсов.
- Гибкость разработки: Различные команды могут параллельно разрабатывать разные микросервисы, выбирая технологии, которые наилучшим образом соответствуют их задачам, не мешая работе других команд.
- Устойчивость к сбоям: Ошибки и сбои в одном микросервисе обычно не оказывают значительного влияния на работу остальных частей

системы, что способствует улучшению общей устойчивости системы.

- Быстрота развертывания: Возможность часто выпускать обновления и исправления для отдельных микросервисов способствует более быстрому выведению продукта на рынок.
- Упрощение управления: Работа с небольшими кодовыми базами облегчает поддержку и тестирование отдельных функций приложения.

Недостатки микросервисной архитектуры:

- Сложность взаимодействия: Управление связями между многочисленными микросервисами требует тщательной настройки API и постоянного мониторинга взаимодействий.
- Дополнительные затраты на инфраструктуру: поскольку микросервисы функционируют в распределенной среде, это может потребовать более сложной инфраструктуры, а также управления логами и мониторингом.
- Сложность тестирования: Интеграционное тестирование микросервисов требует специализированных подходов и может оказаться более сложным по сравнению с тестированием монолитных систем.
- Затраты на разработку: Первичная настройка инфраструктуры и процессов для микросервисного подхода может потребовать значительного объема времени и ресурсов.
- Сетевая задержка: Взаимодействие между микросервисами через сеть может вызвать задержку и дополнительные накладные расходы по сравнению с прямыми вызовами в монолитной системе.

Микросервисная архитектура стала предпочтительным выбором для крупных компаний, таких как Сбербанк и Яндекс, благодаря своей способности улучшать гибкость и скорость разработки. Разделение сложных приложений на независимые компоненты позволяет отдельным командам

работать над собственными микросервисами, ускоряя процесс внедрения инноваций и повышая эффективность.

Эти компании обладают необходимыми ресурсами для инвестиций в сложную аппаратную инфраструктуру и облачные решения, что обеспечивает эффективное масштабирование и отказоустойчивость. Значительные финансовые возможности позволяют покрывать высокие начальные издержки, связанных с внедрением и поддержкой микросервисной архитектуры.

Благодаря обширному штату DevOps-инженеров, такие организации могут автоматизировать процессы развертывания и мониторинга, что гарантирует стабильную и надежную работу системы. Микросервисная архитектура предоставляет масштабируемость и устойчивость, что особенно важно для компаний с большими объемами пользователей, стремящихся минимизировать риски простоев и поддерживать высокое качество услуг.

Двухуровневая клиент-серверная архитектура - изначально такие системы основывались на традиционной двухуровневой клиент-серверной архитектуре (рисунок 5).

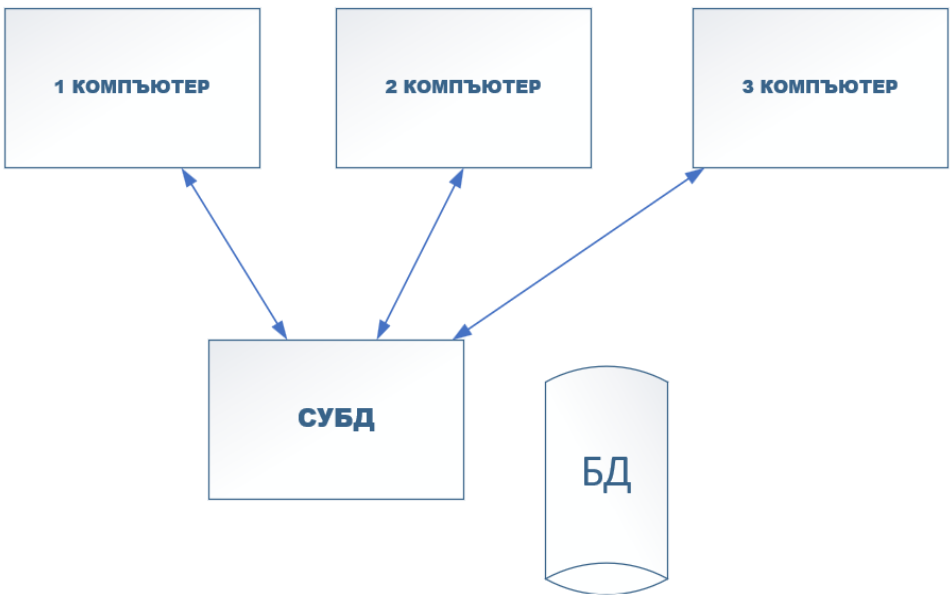

Рисунок 5 - Двухуровневая клиент-серверная архитектура

«Эта клиент-серверная архитектура (рисунок 5) включает в себя два независимых модуля, которые взаимодействуют между собой: автоматизированное рабочее место (компьютер) и сервер базы данных. В качестве сервера базы данных можно использовать такие системы, как Microsoft SQL Server, Oracle, Sybase и другие. Основные функции сервера баз данных включают хранение, управление и обеспечение целостности данных, а также предоставление возможности одновременного доступа для нескольких пользователей. Клиентская часть представлена «толстым клиентом», то есть приложением, в котором сосредоточены ключевые правила функционирования системы и находится пользовательский интерфейс программы» [18].

«Несмотря на простоту данной архитектуры, она имеет множество недостатков. К ключевым минусам можно отнести высокие требования к сетевым ресурсам и пропускной способности сети компании, а также сложность в обновлении программного обеспечения из-за распределения бизнес-логики между автоматизированным рабочим местом и сервером баз данных. Более того, с ростом количества рабочих мест возрастают требования к аппаратному обеспечению сервера баз данных, который, как известно, является самым дорогостоящим компонентом в любой информационной системе» [22]. Поскольку у этой архитектуры много недостатков, а решение выделить бизнес-логику в отдельный слой не всегда является приемлемым, была разработана новая трехуровневая клиент-серверная архитектура.

«В этой архитектуре (рисунок 6) концентрация бизнес-логики на сервере приложений позволяет использовать различные базы данных. Это освобождает сервер базы данных от необходимости распараллеливания работы для разных пользователей, что значительно уменьшает его аппаратные требования. Кроме того, клиентские устройства теперь требуют меньше ресурсов, потому что все ресурсоемкие операции выполняются на сервере приложений, а клиенты занимаются только визуализацией данных. Из-за этих

особенностей такую архитектуру часто называют архитектурой тонкого клиента» [6].

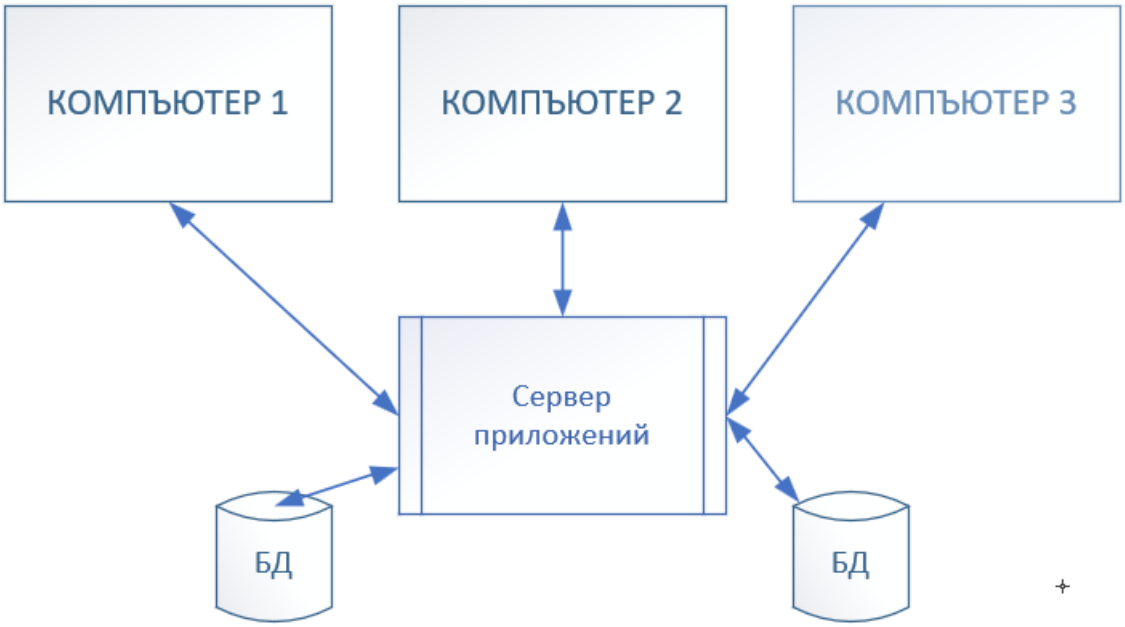

Рисунок 6 - Трехуровневая клиент-серверная архитектура

Однако «у этой архитектуры есть и недостатки. Одним из них являются повышенные требования к пропускной способности сети. Это ограничение усложняет использование таких систем в сетях с неустойчивым соединением и низкой пропускной способностью, таких как Интернет, GPRS и мобильная связь» [6].

«Есть также еще один важный аспект использования систем с такой архитектурой. Самый верхний уровень, обладая значительной вычислительной мощностью, зачастую простаивает, выполняя лишь функции отображения информации на экране пользователя [9]. Учитывая эти недостатки, была разработана еще одна распределенная архитектура системы (рисунок 7).

В настоящее время благодаря доступным и недорогим аппаратным средствам, реализация подобной архитектуры систем для малого и среднего

бизнеса стала возможной. Современные ноутбуки обладают мощностью, которая несколько лет назад была доступна только серверам крупных корпораций и позволяла обрабатывать значительные и критически важные отчеты для всех их сотрудников» [6].

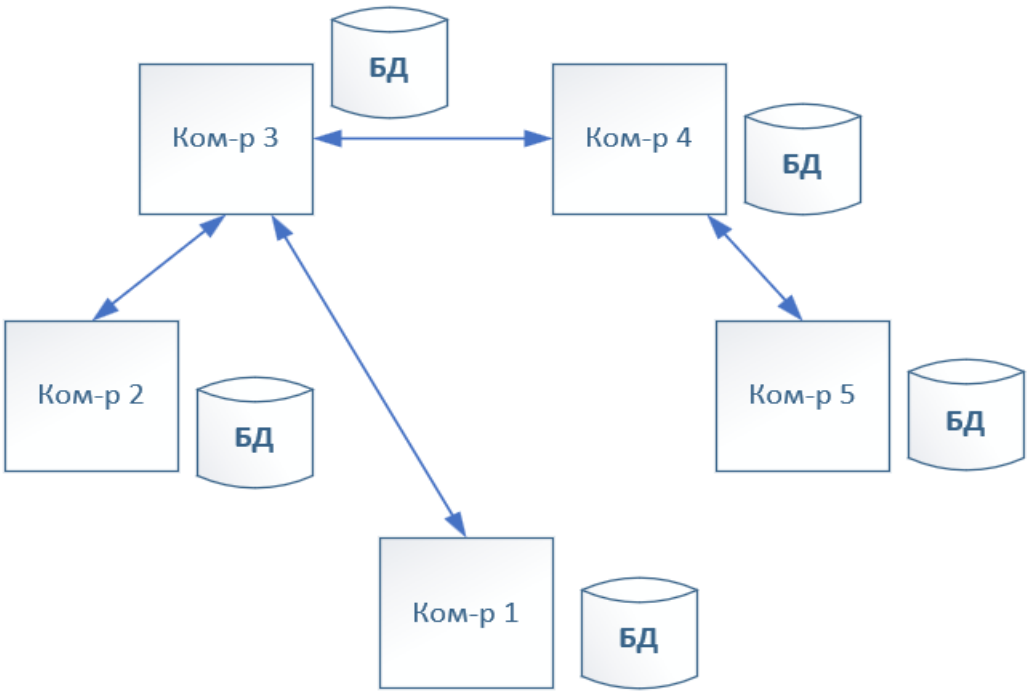

Рисунок 7 - Распределенная архитектура системы

«Подавляющее большинство данных — более 95%, — используемых для управления предприятием, могут храниться на одном персональном компьютере, позволяя ему работать автономно. Объем корректировок и дополнений, генерируемых на этом компьютере, относительно невелик по сравнению с общим объемом используемых данных. Поэтому, если хранить часто используемые данные локально на компьютерах и организовать обмен исправлениями и дополнениями, суммарное количество передаваемых данных значительно уменьшается. Это позволяет снизить требования к связующим каналам между компьютерами и активнее использовать асинхронные методы связи. В результате можно создавать устойчиво функционирующие

распределенные информационные системы, способные действовать даже при нестабильной связи через Интернет, мобильные сети или коммерческие спутниковые каналы. Минимизация трафика между элементами делает такие системы более экономичными в эксплуатации. Однако следует учитывать, что реализация подобной системы не проста и требует решения ряда задач, одной из которых является своевременная синхронизация данных» [25].

Каждое «автоматизированное рабочее место (АРМ) функционирует самостоятельно, располагая только необходимыми для работы данными. Актуальность информации в системе поддерживается через постоянный обмен сообщениями с другими АРМ. Этот обмен может осуществляться различными методами, от электронной почты до передачи данных по сетям» [8].

«Одним из преимуществ такой схемы и архитектуры системы является возможность персональной ответственности за защиту данных. Поскольку данные, доступные на конкретном рабочем месте, хранятся исключительно на этом компьютере, использование шифрования и личных аппаратных ключей исключает доступ посторонних, включая IT-администраторов.

Эта архитектура также позволяет организовать распределенные вычисления между клиентскими устройствами. Например, сложные вычислительные задачи можно разделить между соседними АРМ, поскольку у них, как правило, имеются сходные данные в их базах данных, что позволяет добиться максимальной производительности системы» [14].

Таким образом, предложенная модель распределенных систем способна эффективно выполнять функции современного программного обеспечения для малого и среднего бизнеса. Системы, построенные на этой архитектуре, будут обладать надежностью, безопасностью данных и высокой вычислительной скоростью, что является их основным требованием.

В условиях многоплатформенной корпоративной информационной системы (КИС) выбранная архитектура играет решающую роль в обеспечении совместимости и гибкости.

Для того чтобы понимать, какая архитектура может подойти для каждого конкретного случая, необходимо выявить факторы, влияющие на выбор архитектуры. Ключевыми критериями для оценки и выбора архитектурного подхода являются:

- Степень сложности – насколько сложна реализация, настройка и поддержка архитектуры. Сложные архитектуры могут потребовать дополнительных ресурсов и опыта в управлении распределенными компонентами.
- Масштабируемость – способность системы увеличивать производительность при росте числа пользователей или объема данных. Важно учитывать, планируется ли система для небольшого числа пользователей или должна поддерживать масштабирование на уровень крупных предприятий.
- Производительность – способность системы быстро обрабатывать запросы при увеличении нагрузки. Некоторые архитектуры более эффективно справляются с высокой нагрузкой за счет распределения вычислений или асинхронного взаимодействия компонентов.
- Гибкость – насколько легко архитектуру адаптировать к изменениям. Гибкие архитектуры позволяют быстро вносить изменения в компоненты или добавлять новые функции, что может быть полезно для систем с постоянно изменяющимися требованиями.
- Безопасность – требования к защите данных и компонентов. Для некоторых систем критически важно наличие надежных механизмов аутентификации и авторизации, а также возможность защищать каналы передачи данных.
- Интеграция – насколько легко архитектура позволяет интегрировать внешние сервисы и компоненты. Этот критерий важен, если система должна взаимодействовать с другими приложениями или системами, обмениваться данными или предоставлять доступ через API.

- Обслуживаемость – способность системы к легкому обслуживанию, обновлению и устранению неисправностей. Архитектуры с возможностью независимого обновления компонентов часто удобнее и дешевле в поддержке.
- Подход к разработке – архитектура влияет на стиль и процессы разработки. Например, микросервисная архитектура требует модульного подхода, тогда как традиционные клиент-серверные архитектуры могут поддерживать монолитную разработку.
- Используемые технологии – для каждой архитектуры применимы свои технологии и инструменты. При выборе подхода важно учитывать доступные ресурсы и знания команды в использовании определенных технологий (контейнеризация, веб-сервисы, очереди сообщений и др.).
- Области применения – каждая архитектура имеет свои предпочтительные области применения. Например, «микросервисы и SOA лучше подходят для корпоративных приложений, требующих высокой гибкости и масштабируемости, тогда как двухуровневая архитектура подойдет для простых приложений с ограниченным числом пользователей» [19].

Современные тенденции в построении корпоративных информационных систем (КИС) демонстрируют значительное развитие архитектурных подходов, которые позволяют компаниям эффективно адаптироваться к меняющимся условиям рынка и технологическим инновациям. Однако как видно из таблицы 3 каждая архитектура может обладать сильными и слабыми сторонами.

Одной из самых обсуждаемых и внедряемых архитектурных моделей сегодня является микросервисная архитектура.

Таблица 3 – Оценки КИС

Критерии	Клиент-серверная	Двухуровневая клиент-серверная	Трехуровневая клиент-серверная	Распределенная архитектура системы	Микросервисная
Степень сложности	Низкая	Низкая	Средняя	Высокая	Высокая
Масштабируемость	Ограниченная	Ограниченная	Высокая	Очень высокая	Очень высокая
Производительность	Зависит от нагрузки на сервер	Зависит от нагрузки на сервер	Высокая	Высокая	Высокая
Гибкость	Низкая	Низкая	Средняя	Высокая	Очень высокая
Безопасность	Зависит от настройки сервера	Зависит от настройки сервера	Зависит от настройки серверов и сети	Зависит от сетевой безопасности	Зависит от управления сервисами
Интеграция	Ограниченная	Ограниченная	Высокая	Очень высокая	Высокая (API для взаимодействия)
Обслуживаемость	Умеренная	Умеренная	Высокая	Высокая	Высокая (независимое обновление сервисов)
Подход к разработке	Монолитный	Монолитный	Модульный	Распределенный	Микросервисный
Используемые технологии	Клиентские приложения, серверы	Клиентские приложения, серверы баз данных	Клиентские приложения сервер приложений	Узлы, облачные технологии	Контейнеры, API, инструменты оркестрации
Области применения	Простые приложения	Приложения для небольших и средних бизнесов	Корпоративные приложения со сложной логикой	Распределенные системы, облачные решения	Мобильные и облачные приложения

Микросервисный подход предполагает модульную разработку, где приложение разделяется на независимые компоненты (микросервисы), которые взаимодействуют друг с другом через четко определенные интерфейсы (обычно API). Такой подход обеспечивает высокую гибкость, позволяет легче масштабировать системы и упрощает развертывание обновлений. Крупные компании, такие как Яндекс и Сбербанк, активно

используют микросервисную архитектуру. Наличие обширных ресурсов в виде команды специалистов и бюджета позволяет им не только внедрять этот подход, но и поддерживать его на должном уровне.

Тем не менее, микросервисы требуют серьезных инвестиций как в разработку, так и в инфраструктуру, что делает этот подход не всегда оптимальным для более мелких организаций. Временные затраты на налаживание системы, интеграцию новых сервисов и решение вопросов совместимости могут быть значительными, особенно если компания не обладает необходимыми ресурсами.

Нельзя забывать и о том, что традиционные архитектуры, такие как клиент-серверные и распределенные системы, по-прежнему находят свое применение. Клиент-серверная архитектура легко управляется и подходит компаниям с менее сложными приложениями и ограниченными ресурсами. С другой стороны, «распределенные системы, которые предполагают взаимодействие между удаленными ресурсами, могут быть идеальным решением для организаций, работающих с географически рассредоточенными данными и нуждающихся в высокой доступности и отказоустойчивости» [34].

Каждая архитектура имеет свои сильные и слабые стороны, и выбор подхода должен базироваться на специфических потребностях бизнеса, его масштабе и доступных ресурсах. В конечном счете успешность любой архитектуры определяется не столько ее современностью или популярностью, сколько тем, насколько хорошо она отвечает текущим и будущим требованиям компании.

1.3 Анализ методов обмена данными в корпоративно системе

Обмен данными является процессом, передачи информации между различными системами, компьютерными приложениями, устройствами, компонентами программного обеспечения или пользователями. Это ключевой

аспект любой информационной системы, обеспечивающий совместимость, взаимодействие и целостность данных.

Сущность обмена данными можно рассматривать через несколько аспектов:

- Передача информации – Обмен данными включает передачу информации от одного участника к другому. При этом передача может происходить как в реальном времени, так и в отложенном режиме, быть синхронной или асинхронной. Использует такие протоколы связи как TCP/IP, HTTP, FTP и другие.
- Форматы данных – Для правильной интерпретации передаваемой информации обеими сторонами данные должны быть упорядочены в общем формате. Существуют стандарты форматов данных, такие как XML, JSON, CSV и другие.
- Протоколы обмена - определяют правила и порядок, по которым данные передаются и принимаются. Могут включать механизмы установления связи, передачи сообщений, проверки целостности данных и подтверждения их получения. Примеры таких протоколов включают HTTP/HTTPS для веб-трафика, SMTP для электронной почты и другие.
- Безопасность и авторизация - Обеспечение конфиденциальности и целостности передаваемой информации выполняется с помощью шифрования данных, аутентификации пользователей и авторизации доступов. Используются протоколы, такие как SSL/TLS, чтобы предотвратить перехват и модификацию данных.
- Целостность и согласованность данных - подразумевает, что информация передается и хранится без искажений и потерь. Это особенно важно в многозвенных архитектурах, где данные могут переходить через множество промежуточных узлов.
- Скорость передачи - Скорость обмена данными является важными характеристикой, в системах, требующих высокого уровня

реального времени, таких как торговые платформы или системы управления промышленным оборудованием.

Существует несколько методов обмена данными между пользователем и программой, самым простым является использование командной строки. В данном «методе пользователь вводит команды в программу, а она выполняет соответствующие операции и выводит результаты на экран. Данный метод помогает обработать большое количество данных или выполнить сложные вычисления» [26].

Существуют сложные методы обмена данными, которые включают в себя использование графического интерфейса пользователя и веб-технологий. Графический интерфейс – это интерфейс программы, который позволяет пользователям взаимодействовать с программой с помощью элементов управления, таких как кнопки, текстовые поля и список выбора. «Веб-технологии – это технологии обмена данными между пользователем и программой с использованием интернета. Они позволяют пользователю вводить данные, а сервер обрабатывает эти данные и возвращает результаты пользователю» [26].

Важным аспектом обмена данными является проверка введенных пользователем данных на корректность. Некорректные данные могут привести к неправильным результатам или даже к ошибкам в программе.

Обмен данных может осуществляться по разным каналам, таким как: веб-страницы, электронная почта, социальные сети, мессенджеры, сетевые протоколы.

Основными методами обмена данными являются:

- GET-запросы - передача данных от клиента к серверу путем добавления их в URL-адрес запрашиваемой веб-страницы. Этот метод обмена данных широко используется для получения информации от сервера;
- POST-запросы - передача данных от клиента к серверу путем отправки их в теле HTTP-запроса. Этот метод обмена данных часто

используется для передачи конфиденциальной информации и отправки данных на сервер;

- AJAX - технология, позволяющая отправлять асинхронные запросы на сервер без перезагрузки страницы и обмениваться данными в фоновом режиме [26].

Кроме базовых методов обмена данными, таких как использование форм с отправкой GET или POST запросов, существуют более продвинутые методы, которые расширяют возможности взаимодействия между пользователем и программой. К таким методам относятся:

- Метод XMLHttpRequest используется для отправки асинхронных запросов к серверу, позволяя загружать данные без перезагрузки страницы. Применяется в основном для работы с AJAX (Asynchronous JavaScript and XML), который является важным инструментом веб-разработки;
- Протокол WebSocket обеспечивает постоянное двустороннее соединение между клиентом и сервером. Он позволяет обмениваться данными в режиме реального времени, что часто используется для создания чатов, онлайн-игр и других приложений, где требуется мгновенное обновление информации;
- Технология WebRTC позволяет установить прямое соединение между браузерами, минуя сервер. Она обеспечивает передачу аудио и видео в реальном времени и широко применяется в видеочатах, голосовых вызовах и других коммуникационных приложениях;
- Технология Server-Sent Events (SSE) позволяет серверу отправлять данные клиенту через однонаправленное соединение. SSE поддерживает передачу потоковых данных, таких как обновления в реальном времени, новости или уведомления, без необходимости постоянного опроса сервера;
- GraphQL - это язык запросов, разработанный для взаимодействия с API, который позволяет клиентам запрашивать исключительно те

данные, которые им необходимы, вместо получения всей информации, доступной на сервере. С помощью GraphQL можно указать точную структуру необходимых данных, что позволяет серверу вернуть только актуальную информацию. Такой подход «уменьшает объем передаваемых данных и снижает время выполнения запроса» [15].

Проанализируем некоторые методы, которые могут использоваться для систем на базе «1С:Предприятие 8».

«WebSocket — это простой метод обмена данными, который обеспечивает значительные преимущества, включая двустороннюю коммуникацию и возможность реального времени обмена информацией между сервером и клиентом. Эта технология позволяет создавать приложения, которые обрабатывают данные мгновенно и остаются актуальными. Однако имеет недостатки, такие как сложность реализации как на серверной, так и на клиентской стороне, а также потребность в дополнительных мерах обеспечения безопасности» [15]. Тем не менее, после первоначальной настройки WebSocket демонстрирует высокую удобность и эффективность в работе. WebSocket (рисунок 8) идеально подходит для задач, требующих непрерывного обновления данных, таких как реалтайм-уведомления, обновление состояния интерфейса в режиме реального времени и синхронизация данных между несколькими клиентами.

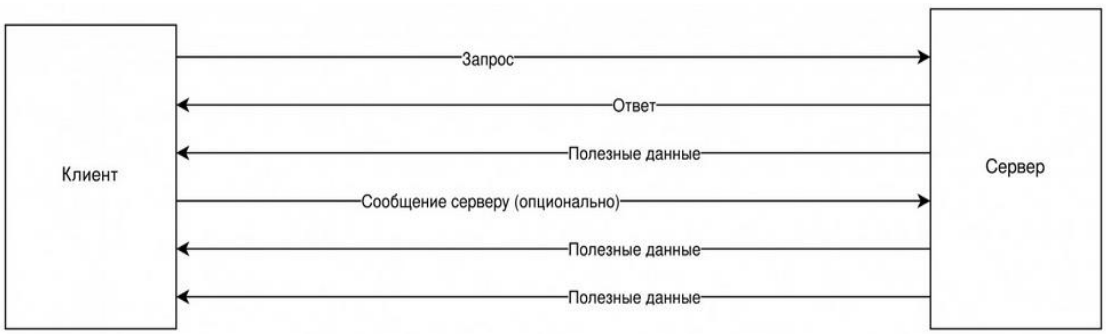

Рисунок 8 – WebSocket

AJAX - в отличие от WebSocket, метод AJAX используется для задач, которые не требуют постоянного соединения с интернетом и допускают выполнение с задержкой, например, загрузка данных по запросу пользователя, отправка форм, выполнение операций CRUD (создание, чтение, обновление, удаление) и т.д. AJAX легко внедряется и не требует поддержания постоянного соединения, однако он категорически не подходит для приложений, требующих моментального двустороннего взаимодействия.

AJAX и WebSocket могут эффективно дополнять друг друга, предоставляя разработчикам возможность выбирать наиболее подходящие инструменты для решения различных задач в зависимости от требований их приложений. Использование этих технологий вместе позволяет создавать более гибкие, производительные и эффективные веб-приложения, способные удовлетворить широкий спектр пользовательских потребностей.

Метод Server-Sent Events (SSE) - в рамках «1С:Предприятие 8» предоставляет возможность отправки данных с сервера на клиент в режиме реального времени. SSE является частью HTML5 и предоставляет простой способ установления однонаправленного соединения, при котором сервер может отправлять обновления клиенту.

Метод является однонаправленным и обеспечивает соединение от сервера к клиенту, то есть сервер может отправлять данные, а клиент их получать, но обратного сценария быть не может.

К преимуществам метода относится то что в случае разрыва соединения браузер автоматически переподключает клиента к серверу. Так же данный метод легче реализовать, в сравнении с WebSocket, так как он использует стандартизированные HTTP/2 механизмы.

Поддерживает (JSON, XML, обычный текст и т. д.), не предназначен для передачи бинарных данных.

Применение этих методов в корпоративной системе на базе «1С:Предприятие 8» позволяет обеспечить оперативное и эффективное обновление информации, а также надежный обмен данными между

различными отделами компании. Эти методы позволяют значительно улучшить взаимодействие внутри системы, обеспечивая плавную и бесперебойную передачу данных.

Важно отметить, что остальные методы обмена данными, такие как RESTful API, gRPC или различных видов RPC (Remote Procedure Call), скорее подходят для других платформ и решений. Они могут оказаться менее оптимальными для корпоративной системы на базе «1С:Предприятие 8» из-за специфики интеграции и работы этой платформы. В связи с этим, WebSocket, AJAX и Server-Sent Events являются наиболее предпочтительными для данной системы.

В таблице 4 показаны главные особенности каждого выбранного обмена данных.

Таблица 4 – Сравнение выбранных методов

WebSocket	Server-Sent Events	AJAX
Двунаправленность: и сервер, и клиент могут обмениваться сообщениями	Однонаправленность: данные посылает только сервер	Однонаправленность
Бинарные и текстовые данные	Только текст	Бинарные и текстовые данные
Протокол WebSocket	Обычный HTTP	использует HTTP(S)

Подходящими для корпоративной системы на базе «1С:Предприятие 8» методами, будут простые методы обмена данными: WebSocket и AJAX. И сложный метод обмена данными - Server-Sent Events (SSE). Так как они позволяют быстро обновлять изменения в системе и обмениваться данными между разными структурными подразделениями. Остальные методы подходят для других платформ в большей степени, чем для «1С:Предприятие 8».

1.4 Анализ моделей обмена данными в корпоративно системе

«Open Systems Interconnection, далее OSI является базовой модель взаимодействия систем. Она представляет стандартный способ организации различных аспектов коммуникаций в рамках обмена данными» [10].

Модель OSI разделяет схему сетевого взаимодействия на семь уровней, где каждый уровень выполняет определенные функции и имеет свою роль в обеспечении сетевых коммуникаций. Таблица 5 содержит примеры реальных протоколов и технологий, которые соответствуют каждому уровню.

Таблица 5 – Уровни модели OSI

Уровень	Основные функции	Примеры технологий и протоколов
Физический	Передача битов через физическую среду, модуляция сигналов, синхронизация, управление доступом к физической среде	Ethernet, USB, Bluetooth
Канальный	Формирование кадров, контроль ошибок, управление доступом к среде, управление потоком информации	MAC, LLC, ARP, CSMA/CD, Wi-Fi
Сетевой	Маршрутизация данных, разбиение на пакеты, назначение логических адресов	IP, ICMP, OSPF, BGP
Транспортный	Разбиение на сегменты, управление потоком, обнаружение и исправление ошибок, мультиплексирование	TCP, UDP
Сеансовый	Управление сеансами, синхронизация, управление диалогом, обеспечение безопасности, управление восстановлением	NetBIOS, RPC, PPTP
Представительский	Шифрование, сжатие, преобразование данных, форматирование, обеспечение совместимости данных	SSL/TLS, JPEG, GIF, ASCII, Unicode
Прикладной	Предоставление интерфейса для взаимодействия с сетью, администрирование сеансов, поддержка сетевых приложений	HTTP, SMTP, FTP, DNS, SNMP

Главные функции:

- преобразование цифровой информации в «аналоговые сигналы для их передачи через различные среды. В зависимости от параметров передачи могут применяться амплитудная, частотная или фазовая модуляция;
- установление типа физического подключения между программно-аппаратными компонентами сети, как, например, использование коаксиальных и оптоволоконных кабелей или витой пары. Также сюда относится определение максимальной длины кабеля и других характеристик» [9];
- организация начальных и завершающих сигналов для передачи данных, а также синхронизация устройств для корректного получения информации;
- выявление и исправление ошибок, влияющих на целостность и точность данных во время их отправки и получения. Это обеспечивает надежность и точность передаваемой информации в сети;
- @управление доступом к физической среде передачи и получения данных, что помогает избежать конфликтов и коллизий, которые могут возникнуть при одновременной передаче информации несколькими устройствами;
- передача битов. Физический уровень фактически передает биты данных между устройствами, используя физические каналы и кодировку сигналов» [24].

Физический уровень предоставляет основную инфраструктуру для передачи информации, и его характеристики определяются технологиями, используемыми в данной сети. Канальный уровень управляет процессами передачи и приема данных между различными программно-аппаратными компонентами в пределах локальной сети. Он выполняет следующие функции:

- формирование кадров — пакетов данных, предназначенных для передачи через физическую среду. Эти кадры содержат

синхронизирующие биты, адреса сетевых устройств, контрольные суммы, а также основную информацию. При приеме данных происходит обработка кадров, после чего они передаются на сетевой уровень;

- «контроль доступа к среде передачи данных по заданным параметрам, что помогает избежать столкновений. Это особенно актуально в сетях Ethernet, где устройства должны следовать определенным протоколам доступа, таким как CSMA/CD;
- выявление и устранение ошибок. Это включает контроль ошибок и проверку целостности данных во время их передачи. Уровень проверяет кадры на наличие ошибок, и в случае их обнаружения кадр отбрасывается или пересылается повторно;
- управление потоком информации. Некоторые протоколы канального уровня способны управлять потоком данных, чтобы гарантировать, что отправитель не перегружает буферы получателя» [12].

Сетевой уровень играет важную роль в маршрутизации, а также в передаче и приёме данных между сетями и их сегментами.

Основные функции:

- Определение маршрута для передачи данных внутри сети включает выбор оптимального пути с учетом таких факторов, как скорость и надежность;
- Каждое устройство в сети обладает уникальным логическим адресом, используемым для его идентификации. Примером такого адреса является IP-адрес;
- На сетевом уровне данные могут быть разделены на более мелкие части, которые называются пакетами, для передачи. Эти пакеты затем собираются на стороне получателя, что способствует эффективной передаче через сети с различными ограничениями на длину кадра;

- Применение методов обнаружения ошибок в передаваемых данных. В отличие от канального уровня, исправление ошибок не производится; сетевой уровень может запросить повторную передачу данных;

Контроль трафика и управление перегрузками включают обеспечение оптимальной передачи данных через эффективное управление трафиком. Это подразумевает решение проблем, связанных с перегрузкой сети, чтобы поддерживать стабильность и надежность связи.

Транспортный уровень играет ключевую роль в обеспечении надежной, точной и эффективной передачи данных в сети. «Вот его основные функции:

- разбиение на сегменты и сборка данных: потоки информации от верхних уровней разбиваются на более мелкие сегменты или пакеты для передачи по сети. Это помогает эффективно использовать сетевые ресурсы и позволяет передавать большие объемы данных;
- управление потоком: контроль скорости передачи данных и регулировка потока помогают избежать перегрузки и потерь информации, что особенно важно при взаимодействии с сетями различной пропускной способности;
- мультиплексирование и демультиплексирование: транспортный уровень позволяет одновременно передавать данные для различных приложений (мультиплексирование) и обеспечивает их корректное распределение и прием у получателя (демультиплексирование) с использованием портов и сессий;
- установка, управление и завершение соединений: этот уровень отвечает за установку и завершение соединений между устройствами в сети;
- обнаружение и исправление ошибок: транспортный уровень способен выявлять и восстанавливать утерянные или поврежденные пакеты данных, чтобы обеспечить их надежную доставку;

Уровень сеанса занимается установкой, поддержанием и завершением сеансов взаимодействия между сетевыми устройствами. Он выполняет следующие функции» [10]:

- Управление сеансами: этот уровень отвечает за создание, поддержание и завершение временных взаимодействий между устройствами, включая передачу данных. При установке соединения для выполнения задачи происходит управление сеансом;
- Синхронизация данных: обеспечивает скоординированную передачу данных между отправителем и получателем, включая установку контрольных точек, которые помогают восстановить передачу после сбоев или повторно отправить утерянные данные;
- «Управление диалогом: определяет порядок передачи данных между устройствами и управляет ходом взаимодействия. Это позволяет организовать обмен данными в определенной последовательности и решать вопросы, связанные с доступом к ресурсам;
- Управление блокировками и параллельными доступами: в некоторых случаях сеансовый уровень занимается управлением блокировками и параллельным доступом к общим ресурсам для предотвращения конфликтов и обеспечения целостности данных;
- Обеспечение безопасности: поддерживает механизмы безопасности, такие как шифрование и аутентификация, для защиты сеансов связи;
- Управление восстановлением: в случае разрывов соединения или сбоев этот уровень обеспечивает механизмы восстановления сеанса и продолжения обмена данными» [4].

Уровень представления данных отвечает за преобразование, кодирование и шифрование данных, что важно для корректного отображения и передачи информации между разными системами и устройствами в сети. Основные задачи этого уровня включают [10]:

- Сжатие данных: уменьшение объема данных перед их отправкой помогает ускорить процесс передачи и сделать его более эффективным;
- «Шифрование данных: обеспечение конфиденциальности и защиты целостности данных во время передачи. Это особенно важно для сохранения безопасности информации в сети;
- Кодирование данных: преобразование данных таким образом, чтобы они могли быть правильно интерпретированы и обработаны на приемной стороне. Этот процесс включает кодирование символов, обработку различных форматов данных и преобразование между различными кодировками;
- Форматирование и структурирование данных: приведение данных в соответствие с требованиями получателя, что позволяет устройствам с различными техническими характеристиками правильно интерпретировать передаваемую информацию;
- Конвертация между форматами данных: обеспечение совместимости и поддержки обмена данными между разнообразными приложениями и устройствами, независимо от того, используются ли текстовые, графические, видео- или аудиофайлы» [17].

Управление сессиями соединений. Представительский уровень данных занимается администрированием сессий соединений, включая их установление и завершение. Среди известных протоколов и стандартов этого уровня можно упомянуть SSL/TLS, которые используются для шифрования информации на веб-страницах. Также на этом уровне происходит кодирование данных с помощью таких стандартов, как ASCII, Unicode и JPEG, применяемых для обработки текстовых и графических данных. Представительский уровень данных отвечает за обеспечение совместимости, безопасности и правильную интерпретацию информации в сетевых системах и приложениях.

Прикладной уровень наиболее приближен к пользователям. Он отвечает за предоставление функциональных возможностей и взаимодействие между приложениями и сервисами в сетевой среде.

Вот это главные функции [10]:

- поддержка прикладных сервисов. Уровень приложений предоставляет пользователям и программам доступ к разным сетевым ресурсам и услугам, таким как веб-сайты, электронная почта, файловые серверы, базы данных и другие приложения;
- «передача данных между приложениями. Этот уровень позволяет программам обмениваться данными с помощью стандартных протоколов и форматов;
- пользовательские интерфейсы. Уровень обеспечивает интерфейсы для работы с программами и сервисами, включая графические интерфейсы, командные строки и веб-интерфейсы;
- администрирование сеансов. Он управляет созданием, установкой и завершением сеансов связи между программами, включая управление браузерными сессиями и аутентификацию;
- совместимость данных и протоколов. Этот уровень выполняет преобразования данных и протоколов для обеспечения совместимости между приложениями с различными требованиями;
- безопасность. Здесь реализуются механизмы защиты, такие как аутентификация и шифрование, для защиты данных и коммуникаций между приложениями [10].

Механизм обмена данными в «1С:Предприятия 8» представляет собой два механизма обмена данными:

- «механизм распределённых информационных баз, обеспечивающий обмен данными исключительно с идентичными конфигурациями «1С:Предприятия 8» и строго регламентирующий структуру создаваемой системы. Он схож с компонентом «Управление распределенными информационными базами» в платформе

«1С:Предприятие 7.7», но в большей степени позволяет гибко настраивать и поддерживает разнообразные схемы обмена;

- универсальный механизм для обмена данными, позволяющий создавать любые распределенные системы и практически не имеющий ограничений по структуре разрабатываемой системы» [25].

Оба механизма задействуют различные средства технологической платформы, которые разработчик может использовать отдельно или комбинировать, в зависимости от специфики задачи. Этот подход обеспечивает гибкость и адаптивность механизмов обмена, позволяя эффективно решать широкий спектр задач.

В состав средств платформы, используемых для построения схем обмена данными, входят:

- планы обмена: Эти объекты конфигурации позволяют описывать список узлов распределенной информационной системы, с которыми будет происходить обмен данными, а также определять состав данных, участвующих в обмене;
- средства XML-сериализации: используются для преобразования данных различных типов из «1С:Предприятие 8» в формат XML и наоборот;
- инструменты для чтения и записи XML-документов: позволяют работать с данными в формате XML на базовом уровне, не завязанном на объекты «1С:Предприятие 8»;

«Одним из ключевых преимуществ данного набора инструментов является его высокая готовность для работы в распределенных системах - организация обмена не требует значительных усилий по разработке. Достаточно в интерактивном режиме определить, какие данные будут участвовать в обмене, и механизм самостоятельно создаст и загрузит соответствующие сообщения. Также платформа автоматически управляет обменом данными таким образом, чтобы передавалась только измененная

информация, контролирует получение сообщений, определяет надобность повторной отправки, решает конфликты и проверяет целостность полученных данных» [10].

Широкие возможности настройки позволяют создавать различные конфигурации топологии узлов обмена, включая структуры типа звезда, снежинка или схемы без центрального узла. Состав данных и правила разрешения конфликтов можно определить по желанию. Механизмы обмена, с одной стороны, уменьшают объем пересылаемых данных, а с другой - обеспечивают надежность при потере сообщений. Таким образом, система может эффективно работать как при наличии гарантированной доставки, так и в ее отсутствие.

Вывод по главе

В главе проведен анализ существующих методов и моделей обмена данными в корпоративных информационных системах. Рассмотрены архитектурные подходы, такие как клиент-серверная, микросервисная и распределенная архитектуры, а также методы обмена данными, включая WebSocket, AJAX и Server-Sent Events. На основе анализа можно сделать вывод, что выбор архитектуры и методов обмена данными зависит от специфики корпоративной системы и ее требований к гибкости, масштабируемости и безопасности. Оптимальным для многоплатформенной корпоративной системы является комбинированное использование распределенной архитектуры и современных методов обмена, таких как Server-Sent Events для актуализации данных в реальном времени.

Глава 2 Методы разработки обмена данными в многоплатформенной корпоративной системе

В данной главе рассматриваются и детально анализируются различные методы обмена данными в программной среде «1С:Предприятие». Основной целью этого анализа является определение такого метода обмена, который будет наиболее подходящим для использования в дальнейшей разработке корпоративных систем. Для успешного решения поставленной задачи необходимо последовательно выполнить несколько шагов. В первую очередь, требуется провести тщательное описание всех доступных методов обмена данными, которые поддерживаются в рамках программы «1С:Предприятие». После проведения этого анализа следующее, что нужно сделать — это выбрать наиболее оптимальный метод, основываясь на таких критериях, как эффективность, надежность и соответствие поставленным требованиям. Важно не только остановиться на конкретном методе, но и подробно обосновать сделанный выбор, опираясь на анализ плюсов и минусов каждого рассмотренного варианта. Завершающим этапом в решении задачи станет разработка и внедрение выбранной стратегии обмена данными в корпоративной системе, что позволит обеспечить эффективное взаимодействие между различными модулями и компонентами системы.

2.1 Описание использованного метода

«Механизмы обмена данными в 1С:Предприятие 8.3 позволяют создавать распределенные информационные системы, которые могут обмениваться данными в оффлайн-режиме без постоянного подключения. С помощью этих механизмов можно интегрировать не только различные информационные базы 1С:Предприятие 8, но и создавать сложные гетерогенные системы, которые наряду с решениями на платформе 1С:Предприятие 8 включают внешние приложения» [26].

Платформа поддерживает функционирование механизмов обмена данными.

Механизм распределенных информационных баз предназначен для обмена данными только между идентичными конфигурациями 1С:Предприятие 8 и строго определяет структуру создаваемой системы. В распределенных информационных базах (РИБ) всегда присутствует четкая иерархия управления с «главным» и «подчиненными» узлами (рисунок 9).

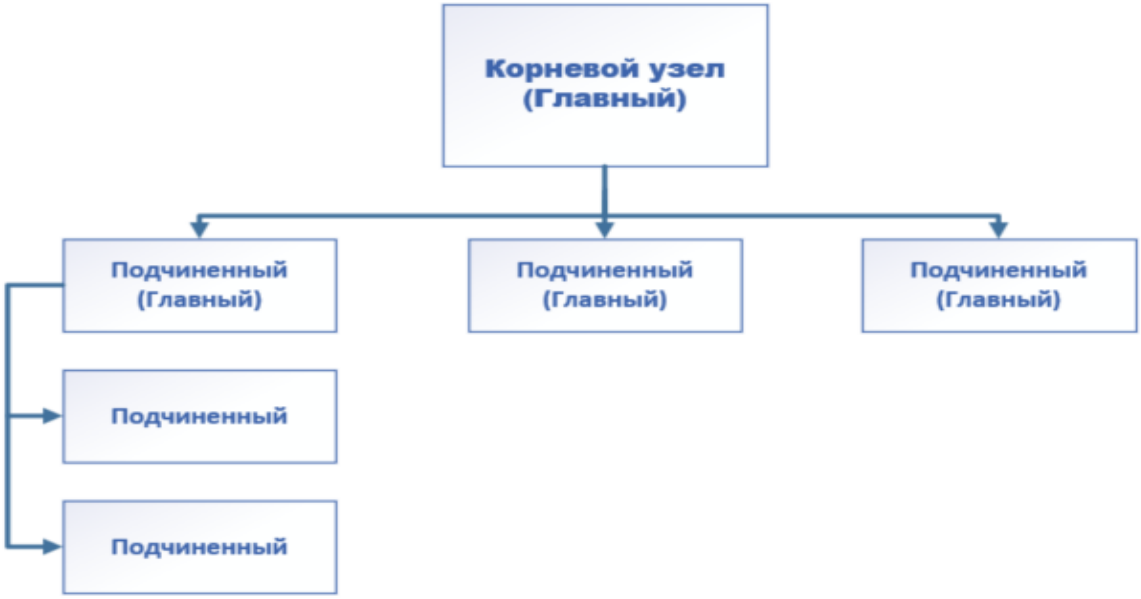

Рисунок 9- Структура управления

Для создания распределенной информационной базы (РИБ) требуется специальный шлюз. Этот шлюз управляет обменом данными между информационными базами, обеспечивая их соединение, контроль передачи данных и совместимость.

«При настройке РИБ необходимо задать структуру и параметры обмена данными. Обычно структура формируется из групп данных и правил их передачи. Группы данных объединяют схожие объекты, такие как

справочники или документы, в то время как правила передачи определяют порядок и условия обмена данными» [26].

РИБ поддерживает различные методы обмена информацией, включая механизмы событий и системы управления версиями. Механизмы событий позволяют автоматически инициировать процессы обмена данными при наступлении определённых событий, к примеру, при изменении данных в конкретном документе. Системы управления версиями обеспечивают контроль за целостностью и актуальностью данных при обмене.

Универсальный механизм обмена данными служит инструментом для создания распределенных систем любой сложности и структуры. Его часто применяют, когда необходимо разделить различные виды учета, к примеру, вести оперативный учет в торговой системе, а регламентный — в бухгалтерской. Обмен осуществляется посредством конвертации данных в универсальный формат XML. Данный механизм предоставляет возможность создавать произвольные распределенные системы без существенных ограничений на их структуру.

Формат обмена данными EnterpriseData предназначен для передачи информации в формате XML как между различными базами 1С, так и с внешними приложениями. Одним из его преимуществ является отсутствие необходимости в дополнительной доработке для импорта и экспорта данных, поскольку поддерживающие EnterpriseData системы имеют единую точку «входа-выхода».

Среди современных решений от компании 1С особо выделяется 1С:Шина, предназначенная для эффективного обмена данными как между системами на платформе «1С:Предприятие», так и с программами на иных платформах. В масштабных организациях обычно осуществляется несколько различных видов деятельности, требующих взаимодействия с различными электронными сервисами, что приводит к формированию сложной инфраструктуры обмена данными, использующей множество протоколов и электронных потоков. Именно здесь на помощь приходит 1С:Шина, главная

задача которой — обеспечить стабильный и точный обмен данными между системами. Это позволяет автоматизировать бизнес-процессы, улучшать синхронизацию данных как внутри различных подразделений компании, так и между разными организациями, сокращая затраты времени и ресурсов на обработку информации.

Эти «механизмы используют различные средства технологической платформы, которые разработчики могут комбинировать или применять отдельно, в зависимости от конкретных задач. Такой подход обеспечивает гибкость и настройку механизмов обмена для решения максимально широкого спектра задач.

В состав средств платформы, используемых для построения схем обмена данными, входят:

- планы обмена: Эти объекты конфигурации используются для описания списка узлов в распределённой информационной системе, с которыми будут обмениваться данные, а также для определения состава участвующих в обмене данных;
- средства XML-сериализации: Эти инструменты предназначены для преобразования различных типов данных из системы «1С:Предприятие 8» в формат XML и обратно;
- инструменты для работы с XML-документами: Эти средства позволяют на базовом уровне манипулировать данными в формате XML, без прямой привязки к объектам системы «1С:Предприятие 8»;

Для организации обмена данными доступно множество технологий: XML, локальные или сетевые директории, текстовые файлы, Excel, ADO- и COM-соединения, FTP-ресурсы, веб-сервисы и электронная почта. Выбор подходящей технологии зависит от возможностей базы данных. Если имеется доступ к базе-приемнику, часто используется ADO-соединение, обеспечивающее прямое подключение источника данных к его получателю. Это решение удобно для пользователя, так как после единовременной настройки параметров обмена и авторизации обмен данными может

осуществляться одним нажатием кнопки или автоматически по расписанию. В случае отсутствия прямого доступа данные выгружаются в промежуточный файл, который затем передается приемнику; другой вариант — использование общего FTP-ресурса» [26].

Одним из значительных преимуществ данного набора технологий является его готовность к работе в распределенной среде, что практически не требует дополнительных усилий на разработку. Нужно лишь интерактивно определить состав данных для обмена, и механизм самостоятельно займется формированием и загрузкой сообщений. Платформа автоматически обменяется только измененными данными, контролирует получение сообщений, определяет необходимость повторной отправки, разрешает конфликты и проверяет целостность загружаемой информации.

«Гибкие настройки позволяют создавать практически любую топологию узлов обмена, будь то звезда, снежинка или схемы без центрального узла. Пользователь может произвольно задавать состав данных для обмена и правила разрешения конфликтов. Механизм обмена уменьшает объем передаваемых данных, отправляя только те, что изменились, и обеспечивает устойчивость к потере сообщений. Таким образом, система функционирует как в условиях с гарантированной доставкой сообщений, так и без этого» [26].

Для настройки обмена данными между базами 1С нужно:

- обозначить между какими базами будет организован обмен;
- определить обмен между базами будет двусторонним или односторонним. Если односторонний, то нужно определить базу для приемки информации и базу-источник информации;
- разработать схему построения баз;
- выбрать оптимальный механизм обмена;
- выбрать подходящий транспорт для выполнения обмена;
- определить массив данных - документов, справочников и других объектов;
- выполнить настройку правил синхронизации;

– составить расписание периодичности обмена [27].

2.2 Обоснование выбора метода

Для разработки многоплатформенной корпоративной системы на базе «1С:Предприятия 8» выбран метод обмена данными - механизм распределенных информационных баз далее (РИБ). В данном механизме главная база может быть только одна, подчиненных может быть много. Каждая подчиненная база может иметь свои подчиненные базы, для которых она будет главной.

РИБ обеспечивает интеграцию нескольких информационных баз в единую структуру, что позволяет централизованно управлять данными и ускоряет обмен информацией между базами.

Главное преимущество РИБ заключается в возможности обработки данных на разных серверах. Это снижает нагрузку на один сервер, распределяет задачи между базами данных и повышает общую эффективность системы. Также благодаря сетевому взаимодействию данные в РИБ могут оперативно обновляться, обеспечивая актуальность и точность информации в режиме реального времени.

Основные принципы работы механизма распределенных информационных баз 1С:

– работа через серверы баз данных. Для функционирования РИБ необходимо наличие серверов баз данных на каждом участке развития информационной системы;
– синхронизация данных. При использовании РИБ данные синхронизируются между информационными базами в реальном времени или по расписанию, в зависимости от настроек системы;
– обеспечение безопасности данных. Механизм РИБ обеспечивает защиту данных от несанкционированного доступа или несанкционированных изменений;

– удобство использования. РИБ позволяет создать единую точку доступа к данным из различных информационных баз, что существенно упрощает и ускоряет процесс работы с информацией [28].

РИБ имеет свои преимущества и недостатки (таблица 6).

Таблица 6 - Преимущества и недостатки РИБ

Преимущества	Недостатки
Повышение уровня автоматизации бизнес-процессов и обеспечение прозрачности и надежности обмена данными внутри компании	На каждом компьютере необходима лицензия платформы и основная лицензия системы лицензирования и защиты конфигураций
Каждое подразделение может иметь свою локальную базу данных, все данные могут быть синхронизированы с центральной базой данных, что обеспечивает единую точку управления и контроля за информацией.	
Интерактивное создание распределенной системы и выполнение обмена данными без дополнительного программирования	Каждое рабочее место должно соответствовать минимальным системным требованиям, для работы в «1С:Предприятие 8»
Идентичность конфигураций информационных баз, входящих в состав распределенной системы	
В рамках одной распределенной информационной базы может быть создано несколько схем обмена	Изменения конфигурации передаются только от главного узла к подчиненному.
Восстановление обмена данными в таких случаях, как восстановление информационных баз из резервных копии и т.д.;	При выгрузке данных справочники, документы и прочие объекты блокируются для новых изменений.
Сжатие сообщений обмена в формате ZIP и автоматическая распаковка сообщений обмена при приеме	Обмен данными осуществляется между абсолютно идентичными конфигурациями 1С

Анализируя таблицу 6, можно сделать очевидный вывод, что метод РИБ имеет больше плюсов, чем минусов для компании. Также данный метод не требует больших финансовых затрат. В таблице 7 приведен сравнительный анализ РИБ с другими методами обмена данными в «1С:Предприятия 8»

Таблица 7 - Сравнительный анализ РИБ

Критерий	РИБ	Универсальный механизм обмена данными	Формат обмена данными EnterpriseData	1С:Шина
Обмен данными между абсолютно идентичными конфигурациями баз данных 1С	+	+	+	+
Обмен данными между различными конфигурациями баз данных 1С	-	+	+	+
Обмен данными между программой 1С и внешней программой.	+	+	+	+
Трудоемкость	-	+	-	-
Цена	-	+	+	+

В данной таблице «+» были отмечены те критерии у методов обменов данными, которые свойственны определенному методу, а - были отмечены не свойственные критерии или свойственные в минимальном значении по сравнению с другими.

Анализируя таблицу 7, можно сделать следующие выводы:

- самая низка цена свойственна РИБ, что является большим плюсом для любой компании;
- самым трудоёмким методом является универсальный механизм обмена данными, что объясняется множеством возможностей данного метода;
- обмен данными между программой 1С и внешней программой свойственен для каждого метода обмена данными;
- обмен данными между различными конфигурациями баз данных 1С свойственен всем методам, кроме РИБ. Для РИБ свойственно только обмен данными между абсолютно идентичными конфигурациями баз данных 1С;

– обмен данными между абсолютно идентичными конфигурациями баз данных 1С свойственен всем методам.

Выбор метода обмена данными в «1С:Предприятие» - РИБ обосновывается низкой ценой, нужным функционалом для удовлетворения потребностей компании и быстрой скоростью создания данного обмена. У данного метода есть недостатки, но они не являются недостатками для компании, в которых требуется обмен между идентичными конфигурациями баз данных 1С.

2.3 Настройка распределенной информационной базы в 1С:Предприятие

Распределенная база состоит из одного центрального узла и одного (или нескольких) периферийных узлов. Чаще всего задача обменов между узлами РИБ сводится к выгрузке данных из периферийных узлов в центральную базу.

Разработка распределенной базы в 1С:Управление торговлей 11 происходит быстро и в несколько этапов (Приложение А):

– в главной базу нужно зайти в раздел НСИ и администрирование, далее в –синхронизация данных, потом в настройка синхронизации данных, потом нажать на кнопку - новая синхронизация данных и выбрать - распределенная информационная база;
– в открывшемся окне нужно создать архивную копию базы, нажатием на - создать резервную копию. Далее выбираем необходимый путь для создания копии и нажимаем на кнопку сохранить резервную копию;
– нужно перейти по ссылке - настроить параметры подключения;
– нажимаем на - вариант использовать локальный или сетевой каталог для синхронизации данных и нажимаем -далее;
– укажем наименование программы – корреспондента и префикс и нажимаем-далее;

- видим на экране, что настройки подключения для этой программы завершены;
- совершаем настройку правил отправки и получения данных;
- нажимаем - создание начального образа с файлами и указываем каталог, в котором создастся периферийная база. В качестве расширения должно быть указано 1Cv8 и далее нажимаем- создать начальный образ;
- открывается окно создания начального образа и через какое-то время ожидания оно будет завершено;
- нужно добавить распределенную информационную базу в список программ, зайти в неё и продолжить настройку;
- нажимаем настроить параметры подключения;
- выбираем каталог и нажимаем - далее;
- снова нажимаем- далее и видим, что настройки подключения второй базы сохранены;
- нажимаем - настроить правила отправки и получения данных;
- настройка завершена.

Видно, что настройка происходит достаточно быстро и не требует много затрат, все это может совершить один человек по инструкции в своей компании.

Выводы по главе

Во второй главе работы были рассмотрены методы обмена данными в программе «1С: Предприятие» и выявлен самый подходящий метод -РИБ. Для данного метода было приведено подробное описание и составлено сравнение с другими методами обмена данными.

Также на основе выбранного метода был разработан обмен данными в «1С:Управление торговлей». Данный метод имеет весомые преимущества, которые видны в ходе разработки- быстро и не требует финансовых вложений.

Глава 3 Модель обмена данными в многоплатформенной корпоративной информационной системе

В корпоративной информационной системе на базе 1С:Предприятие одним из наиболее эффективных методов обмена данными является использование распределённых информационных баз (РИБ). Проектирование, разработка и последующая эксплуатация такой системы требует всестороннего и детализированного подхода.

Первоначальный этап проектирования включает анализ требований и планирование. Необходимо определить ключевые бизнес-процессы, которые нуждаются в обмене данными, классифицировать типы данных и оценить ожидаемый объём информации и частоту её передачи. На основе этого анализа формируется архитектурное решение, включающее в себя разработку структуры распределённой информационной системы с выделением центральной базы данных (ЦБД) и соответствующих региональных информационных бассейнов (РИБ).

«Проектирование базы данных является одной из самых сложных и важных задач при разработке информационной системы. Этот процесс должен определить состав базы данных, оптимальный метод структурирования данных для всех её пользователей, а также подходящие инструменты для управления этими данными» [21].

«Проектирование баз данных осуществляется в три этапа:

- на этапе концептуального проектирования происходит сбор, анализ и корректировка требований к данным. Итогом этого этапа становится концептуальная модель, независимая от структуры базы данных, которая часто представляется в формате модели "сущность-связь";
- логическое проектирование включает преобразование требований в конкретные структуры данных. Результатом этого этапа является

структура базы данных, ориентированная на СУБД, а также разработка спецификаций для прикладных программ;

– структурно-функциональная схема ИТ-решения включает определение характерных особенностей хранения данных, методов их доступа и других аспектов» [21].

«Концептуальная модель представляет собой изображение информационных объектов, их атрибутов и взаимосвязей, при этом не описывая способы физического хранения данных. Она также известна как модель предметной области, и иногда её называют информационно-логической или инфологической моделью.

Информационными объектами обычно являются сущности - обособленные объекты или события, информацию о которых необходимо сохранять, имеющие определенные наборы свойств – атрибутов» [20].

1С:Предприятие поддерживает несколько баз данных, включая Oracle Database, PostgreSQL, Microsoft SQL Server. Однако на практике чаще всего используется именно Microsoft SQL Server. Это связано с его высокой производительностью, надежностью и широким распространением на рынке.

Этап разработки начинается с создания и настройки региональных баз, которые должны быть копиями основной конфигурации 1С центральной базы данных, с учётом уникальных параметров для каждого филиала. Проектирование данных для обмена предполагает определение и структурирование наборов данных, которые будут передаваться, а также разработку механизмов идентификации и разрешения возможных конфликтов.

Реализация механизмов обмена данных включает настройку регламентированных заданий для создания пакетов обмена на центральной базе и обеспечение возможности приёма и обработки этих пакетов региональными бассейнами. Важным аспектом является обработка ошибок и ведение логирования для мониторинга процесса обмена данных и своевременного реагирования на возникающие проблемы.

После разработки системы необходимо провести интеграционное тестирование, целью которого является проверка корректности обмена данными. «Тестирование позволяет обнаружить и устранить возможные ошибки и несоответствия в передаче информации, что обеспечивает надёжную и безошибочную работу всей системы» [5].

Следующий важный этап, это эксплуатация и поддержка. Эксплуатация системы обмена данными требует постоянного мониторинга и управления обменными процессами. Важно регулярно обновлять конфигурации центральной и региональных баз данных, обеспечивая их актуальность и безопасность. Мониторинг состояния обменов и обработка сбоев обеспечивают надёжность и непрерывность работы системы.

Для защиты данных применяются механизмы шифрования при передаче информации, а также аутентификация пользователей и настройка прав доступа. Это помогает предотвратить несанкционированный доступ и утечку конфиденциальной информации.

Регулярное обновление системы, контроль за совместимостью версий программного обеспечения и обучение пользователей основам работы с распределённой системой обмена данными являются важными задачами в процессе эксплуатации. Оперативная техническая поддержка и решение возникающих у пользователей вопросов способствуют эффективной работе системы и повышают её устойчивость к возможным сбоям.

3.1. Концептуальное моделирование

Для разработки концептуальной модели следует определиться с ее составом (рисунок 10):

- регистрация нужна для регистрации изменений в объекте, которые передают между базами;
- сообщение - файл обмена, который идет, когда есть интернет;

– номер сообщения, который показывает принятые сообщения и помогает не принять повторно;
– узел- каждая база данных, которая обменивается данными;
– код узла - номер узла.

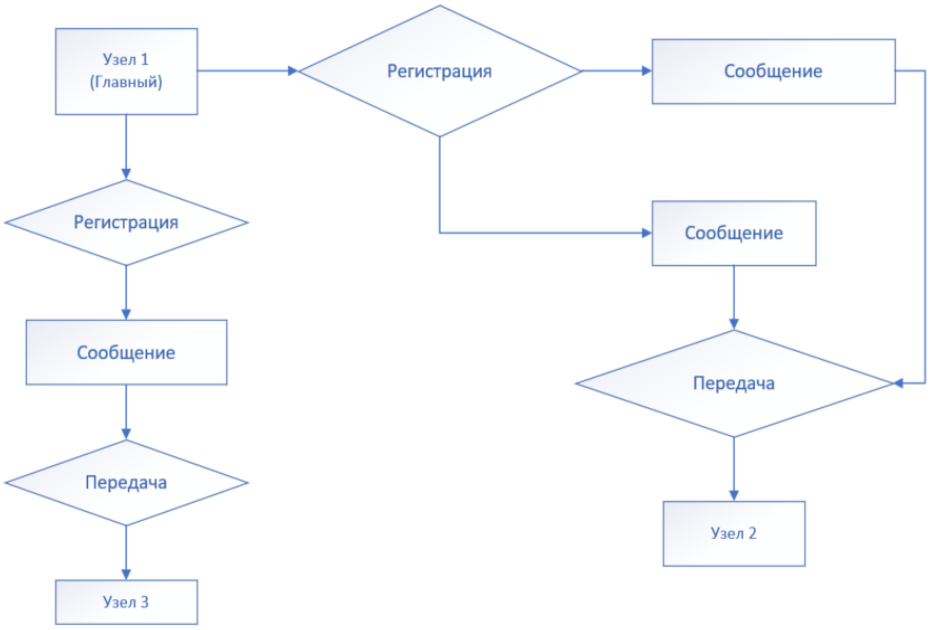

Рисунок 10 - Концептуальная модель распределенной информационной базы

Из рисунка 10 видно, что в методе обмена данными - распределенные информационные базы передаются сообщения только из главной информационной базы (узла) пройдя предварительную регистрацию, а не главных баз может быть несколько, но лучше не больше 10. Также от главной базы может передаваться не одно сообщение, а сразу несколько.

3.2. Логическое моделирование

«Логическое проектирование включает разработку схемы базы данных, основанную на выбранной модели данных. Логическая модель представляет собой набор схем отношений, обычно содержащих первичные ключи и "связи"

между отношениями, выраженные через внешние ключи. Преобразование концептуальной модели в логическую происходит по установленным правилам. Этот процесс может быть автоматизирован с использованием специализированного программного обеспечения» [16].

«Логический уровень представляет собой абстрактное представление данных, отражающее их реальное состояние и наименование в реальном мире. Логическая модель данных является универсальной и не привязана к конкретной реализации баз данных. Она служит первоначальным прототипом будущей базы данных. Качество разработки логической структуры напрямую влияет на конечные физические модели и, в итоге, на эффективность функционирования базы данных, что важно для работы всей автоматизированной информационной системы. Поэтому в процессе разработки логической модели особое внимание следует уделять тщательности проектирования» [16] .

«Для моделирования данных будет использоваться метод IDEF1X в сочетании с CASE-средством ERWin. Методология IDEF1X, классифицируемая как методология "сущность - связь", специально предназначена для разработки баз данных» [16].

Процесс создания логической модели данных с использованием метода IDEF1X можно разделить на следующие этапы:

- выделение сущностей с помощью ERD-метода путем сопоставления с диаграммой классов, которая позволяет импортировать объектную модель;
- определение зависимостей между сущностями (ассоциации и агрегации преобразуем в неидентифицирующие отношения);
- задание первичных и альтернативных ключей;
- определение неключевых атрибутов сущностей;
- построение IDEF1X -диаграммы с помощью доступных CASE-средств;
- генерация SQL- скриптов реализуемой реляционной базы данных.

На рисунке 11 представлена логическая модель распределенной информационной базы.

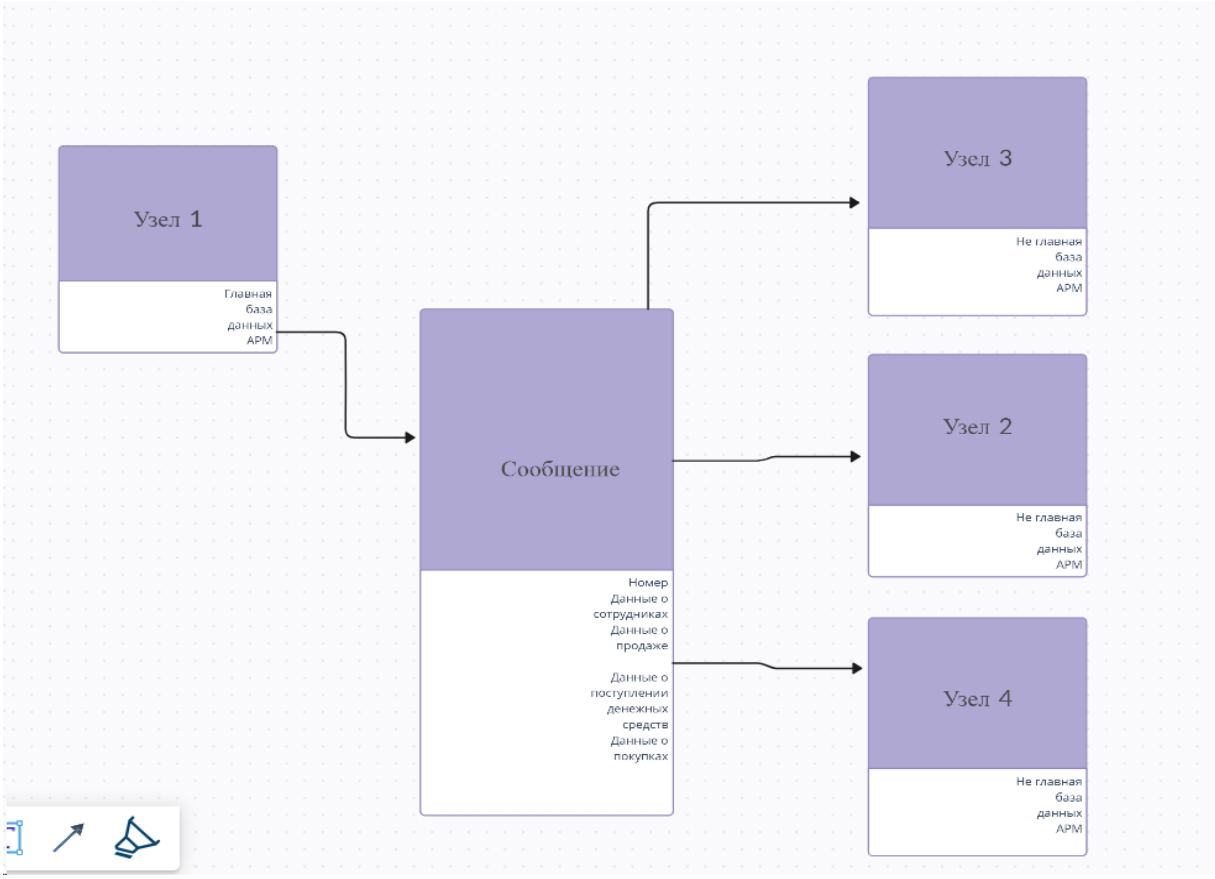

Рисунок 11 - Логическая модель распределенной информационной базы

На рисунке 11 отображено:

- есть только одна главная база данных, остальные получают от нее информацию;
- сообщение может содержать в себе данные о сотрудниках, данные о продаже, данные о поступлении денежных средств, данные о покупках;
- от главного узла может передаваться одно сообщение или сразу несколько в другой узел;
- между не главными узлами нет обмена сообщениями.

Главным достоинством распределенной информационной базы является ее возможность передавать информацию при условии отсутствия интернета

3.3 Структурно-функциональная схема ИТ-решения

На рисунке 12 изображена структурно-функциональная схема ИТ-решения.

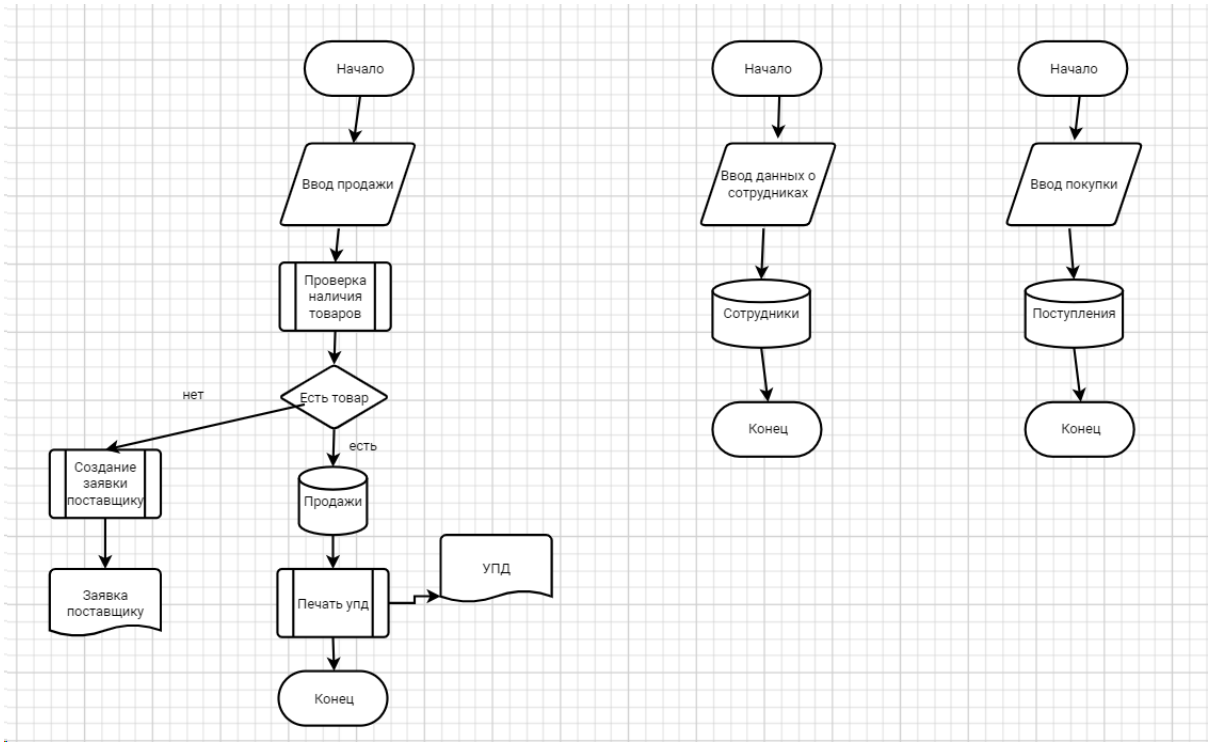

Рисунок 12 - Структурно-функциональная схема ИТ-решения

Структурно-функциональная схема ИТ-решения - это схема, которая отражает состав и взаимодействие по управлению частей разрабатываемого программного обеспечения. Она определяется архитектурой разрабатываемого программного обеспечения и схемой взаимодействия компонентов программного обеспечения с описанием информационных потоков, состава данных в потоках и указанием используемых файлов и

устройств. Для изображения функциональных схем используют специальные обозначения, установленные стандартом.

На схеме изображено (рисунок 12):

- сотрудник вводит продажу, покупку или измененные данные по сотруднику;
- при продаже он печатает универсальный придаточный документ, и продажа попадает в базу данных по продажам;
- при покупке сотрудник вводит информацию о покупке на основании документа, и покупка попадает в базу данных по покупкам;
- сотрудник вводит в базе изменения по сотрудникам (больничные, отпуска, увольнения) и эта информация отражается в базе данных о сотрудниках;
- схема отражает действия сотрудника на автоматизированном рабочем месте с главной базой данных (главным узлом), от которой на другие автоматизированные рабочие места поступает информация (сообщение), которое они видят и информируют покупателей о наличии товара.

Вывод по главе

В главе было разработано концептуальное моделирование, логическое моделирование и структурно-функциональная схема ИТ-решения для выбранной модели обмена данными - распределенной информационной базы.

Концептуальная модель показала информационные объекты, их свойства и связи между ними без указания способов физического хранения информации. Логическое моделирование преобразовала концептуальную модель в более подробную с составом объектов. Структурно-функциональная схема ИТ-решения показала схему взаимодействия пользователей программы и объектов программы.

В состав распределенной информационной базы входят: внешние объекты (пользователи); внутренние объекты (узлы, сообщения).

Глава 4 Апробация модели обмена данными в многоплатформенной корпоративной информационной системе

Цель данной главы — продемонстрировать процесс разработки модели на основе «1С:Предприятие 8», а также проанализировать различные подсистемы, созданные в рамках данной модели. Важной частью исследования является проверка работоспособности и надёжности модели через тщательное тестирование и испытания. Кроме того, будет рассмотрена область её потенциального внедрения, что позволит оценить практическую значимость и эффективность разработанного решения в реальных бизнес-условиях.

В ходе исследования будет показано, как интеграция различных подсистем в единое информационное пространство может повысить производительность, обеспечить бесперебойный обмен данными и улучшить управляемость бизнес-процессами. Главный акцент сделан на использовании распределенной информационной базы, что позволяет ей адаптироваться к требованиям различных предприятий.

Таким образом, данная глава представляет собой комплексное исследование разработки, тестирования и внедрения модели обмена данными в многоплатформенной корпоративной информационной системе. Полученные результаты подтвердят эффективность разработанного подхода и его применимость в условиях современного бизнеса.

4.1. Разработка модели на базе «1С:Предприятие 8. Язык программирования»

Проходя практику на безе школы дополнительного образования, были определены цели и задачи системы (таблица 8).

Таблица 8 - Цели и задачи системы обмена данных [7]

Параметры	Описание
Цель системы	- Формирование эффективной и надежной модели обмена данными среди участников образовательного процесса. - Обеспечение доступа к учебным материалам, расписанию, заданиям и другим ресурсам в многоплатформенной среде (мобильные устройства, стационарные компьютеры и т.д.).
Задачи системы	- Автоматизация административных процессов (зачисление учащихся, ведение успеваемости, расписание занятий и т.д.). - Обеспечение непрерывного и синхронизированного обмена данными между всеми платформами. - Обеспечение безопасности данных и конфиденциальности личной информации студентов и преподавателей.
Академические процессы	- Регистрация и зачисление студентов. - Формирование и управление расписанием занятий. - Ведение журналов успеваемости и посещаемости.
Типы данных	- Личные данные: имя, возраст, адрес, контактная информация студентов и преподавателей. - Академические данные: расписание занятий, учебные материалы, задания, оценки, посещаемость. - Административные данные: отчетность, документы, регламентные данные. - Финансовые данные: информация о платежах и финансовых операциях. - Коммуникационные данные: сообщения, уведомления, обмен информацией.
Функциональные требования	- Интеграция с различными платформами и устройствами. - Поддержка многопользовательского режима. - Возможность генерировать отчеты и документы.

При разработке модели обмена данными на базе «1С:Предприятие 8» необходимо определить узлы информационной базы и их роли. Узлы могут быть центральными или периферийными, и правильное определение их ролей играет ключевую роль в обеспечении эффективного и надежного обмена данными.

– Центральный узел – это главная информационная база, которая обычно находится в головном офисе компании. Он выполняет следующие функции:

– Периферийные узлы – это базы данных, находящиеся в филиалах или отдельных департаментах компании.

Перед тем ка создавать базу данных, определим центральную и периферийною базы, для нашей организации (таблица 9).

Таблица 9 - Узлы организации

Тип узла	Описание
Центральный Узел	- Локация: Головной офис - Функции: Централизованное управление, консолидация данных всех филиалов, репликация и восстановление данных.
Периферийные Узлы	- Локация: Корпус 1 - Функции: Локальная работа с данными Корпуса 1, синхронизация с главным узлом.

Создание распределенной базы данных (Приложение А)

В программе «1С предприятие» подключимся к существующей базе. Переходим в раздел «НСИ и администрирование», затем в «синхронизация», ставим галочку в пункте синхронизация данных. Теперь система будет автоматически обрабатывать и обмениваться данными между несколькими информационными базами. Теперь можно перейти к синхронизации данных.

Настройка синхронизации данных предлагает нам несколько вариантов: Настройка «распределенной информационной базы» и настройка «РИБ с фильтрами», в последнем можно обмениваться документы с отбором по организации или по подразделениям. Мы же будем рассматривать полный вариант «распределенной информационной база».

Окно настройки предлагает обмен через каталог. Обмен через каталог означает, что синхронизация данных между различными узлами распределенной информационной базы будет происходить с использованием файлов, которые сохраняются в указанных каталогах (обычно на диске). Выбираем путь, по которому будет храниться конфигурация РИБ.

В следующем окне мы видим наименование нашей центральной базы и наименование периферийной базы. Лучше будет сделать префикс ПР для того,

чтобы отличать данные центральной базы от периферийной базы. После чего все будет готово для начала синхронизации данных.

После окончания синхронизации мы переходим к этапу создания «начального образа информационной базы», программа предложит два варианта: на данном компьютере, то есть создает образ локально и на сервере 1с: предприятия, образ будет храниться образ на сервере. Я выбираю первый вариант, так как хочу что бы образ хранился на компьютере. Теперь необходимо выбрать путь для образа, будет удобно назвать директорию так же, как будет называться область применения периферийной базы. Например, если база будет использоваться в филиале, или для какого-то конкретного офиса в моем случае база будет называться «ИМЯ БАЗЫ». Следующем шагом произойдет полная выгрузка базы, с инициализацией периферийной базы.

Перейдем к настройке плана обмена, необходимо определить какие данные будут участвовать в обмене.

В первую очередь в компании необходимо организовать обмен данными между головным офисом «Корпус 1» и филиалами «Корпус 2». Для этого в конфигураторе 1С создается новый объект "План обмена".

Открываем конфигуратор 1С и загружает конфигурацию "Управление предприятием". В дереве конфигурации выбирается раздел "Общие" и создается новый объект "План обмена", который получает наименование "ОбменДаннымиФилиалы". Необходимо установить, какие данные будут участвовать в обмене между головным и подчиненными узлами.

На вкладке "Наборы данных" плана обмена "ОбменДаннымиФилиалы" добавляются справочники "Контрагенты" и "Товары", а также документы "Заказы клиентов". Для каждого из этих наборов данных определяются ключевые поля (подписи данных), которые будут использоваться для идентификации. Например, для "Контрагентов" это поле "ИНН", а для "Товаров" — "Артикул".

Настроим правила синхронизации. Они определяют логику и частоту обновления данных, минимизируют конфликты и позволяют выбирать, какие

данные и когда должны синхронизироваться. Это «помогает поддерживать консистентность информации, оптимизировать использование системных ресурсов и обеспечить бесперебойное функционирование бизнес-процессов, позволяя каждое подразделение работать с актуальными и корректными данными» [13].

На вкладке "Правила синхронизации" администратор настраивает так, чтобы изменения в справочниках моментально отражались во всех узлах, а обмен документами происходил раз в час.

Важным шагом является создание связей между головным офисом и филиалами. Вкладка "Узлы" настроена так, что головной офис является мастер-узлом, а все филиалы (например, филиалы в Екатеринбурге и Новосибирске) — подчинены.

На вкладке "Связи" создаются устойчивые каналы обмена данными между головным офисом и каждым филиалом. Для всех данных определяется способ идентификации — основной ключ, состоящий из комбинации полей, таких как "ИНН+КПП" для контрагентов и "Артикул+Склад" для товаров. Устанавливаются обязательные поля, которые должны быть заполнены для успешного обмена, "Наименование" для всех объектов.

Демонстрация существующей системы начинается с определения правил обмена данными между различными узлами, что является ключевым для эффективного функционирования системы.

Идентификация данных для обмена: на первом этапе демонстрации анализируются основные сценарии использования системы в школьной среде. Задача состоит в том, чтобы определить, какие именно данные будут передаваться от одного узла к другому. В контексте школы это может включать расписание уроков, оценки учеников, данные о посещаемости, информацию о преподавателях и администраторах, обновления об образовательных материалах и ресурсах, а также уведомления о школьных событиях и мероприятиях. Например, система может автоматически передавать данные о расписании уроков от администратора к классным

руководителям и ученикам, обновлять информацию об успеваемости, которую учителя вносят в электронный журнал, или отправлять родителям уведомления о предстоящих собрании или мероприятиях.

Был разработан алгоритм обмена мы можем рассмотреть следующие аспекты, такие как подробная настройка параметров обмена, обработка ошибок и регистрация событий (Приложение Б).

Алгоритм, реализованный в процедуре «ВыполнитьОбменДаннымиМеждуОфисами», предназначен для автоматизации обмена данными между офисами, что важно для поддержания согласованности информации в различных организациях. Процедура начинается с логгирования события, фиксируя начало процесса обмена и идентифицируя участвующие стороны — источник и приемник. Это способствует не только мониторингу, но и истории изменений, что важно с точки зрения аудита и управления данными.

Ключевым аспектом является настройка правил обмена данными. Создается структура, в которую включаются элементы данных, подлежащие обмену, такие как документы и справочники. Это позволяет гибко управлять процессом, адаптируя его под текущие бизнес-требования. Важным компонентом алгоритма является блок Попытка, который обеспечивает выполнение основных действий с обработкой ошибок. Этот подход усиливает надежность системы, позволяя ей справляться с возможными сбоями в процессе передачи данных.

Первым шагом в блоке Попытка является загрузка данных из источника на основе заданных правил. Любая неудача на этом этапе фиксируется в журнале регистрации как ошибка, и дальнейшее выполнение прекращается. При успешной загрузке данных процедура переходит к этапу выгрузки в указанный приемник. Этот процесс также сопровождается проверкой на успешность, что критично для гарантии целостности и актуальности данных в конечной точке их использования.

После успешного завершения обоих этапов данных выполняется очистка временных данных. Это действие способствует более эффективному использованию ресурсов системы и предотвращает возможное накопление избыточной информации. Завершающий этап алгоритма включает логгирование успешного завершения обмена. Логгируются также ошибки, возникшие в процессе выполнения процедуры, что критично для последующего анализа и диагностики причин неполадок.

После завершения настройки распределенной информационной базы (РИБ) между филиалами и центральной базой данных часто возникает необходимость в обработке новых событий. Такие события могут обрабатываться с помощью встроенных механизмов разработки на платформе 1С:Предприятие. Одним из таких событий является уведомление пользователей о появлении новых данных в реальном времени. Для этого в системе можно задействовать механизм серверных событий (SSE) с использованием HTTP-запросов к соответствующему серверу.

После успешной синхронизации данных между филиалами и центральным офисом с использованием РИБ, в обработчике необходимо инициировать HTTP-запрос к SSE-серверу. Эта операция позволяет уведомить всех подключённых клиентов о поступлении свежих данных. SSE-сервер, который можно реализовать на базе Node.js, поддерживает постоянные подключения с клиентами, позволяя передавать данные без необходимости их повторных запросов. Работа сервера заключается в регулярном прослушивании поступающих HTTP-запросов от системы 1С и отправке соответствующих уведомлений всем подключенным клиентам, например, через веб-браузеры. В Приложении В приведет код программы.

Типичный пример подобной реализации включает сервер, который обрабатывает два основных URL. Во-первых, это URL /events, через который устанавливается постоянное подключение с клиентами. Во-вторых, это URL /new-order, через который сервер принимает уведомления от 1С о поступлении новых заказов.

Для настройки вызова SSE из 1С после завершения процесса синхронизации, в системе «1С:Предприятие 8» создается специальная обработка (Приложение Г). Эта обработка срабатывает после успешного завершения обмена данными и отправляет на сервер SSE уведомления о появлении новых заказов. Для ускорения обработки данных и повышения их актуальности пользователи в центральном офисе получают обновления сразу после поступления новой информации, что исключает необходимость ожидания следующей полной синхронизации в рамках РИБ.

Применение подобной схемы связи предлагает значительный потенциал. SSE позволяет не только оповещать о новых заказах, но и использовать коммуникацию для связи с клиентом, например, чтобы уведомить о сроках внесения следующего платежа за услуги или предоставлении актуального расписания занятий. В подобном случае требуются дополнительные разработки интерфейса на HTML. В текущем примере демонстрируется простой обмен данными по заданному URL с использованием клиентского интерфейса, который показывает, что связь с базой данных успешно поддерживается пример программного года HTML предложен в Приложении Д [30].

Так, подключившись через URL /events, клиенты могут в реальном времени получать новые сообщения от сервера. После поступления нового заказа информация передается всем подключенным клиентам, и пользователи могут видеть изменения на своих экранах. В результате внедрение РИБ обеспечивает регулярную и своевременную синхронизацию данных между централизованной базой и филиалами, а интеграция с SSE позволяет оперативно уведомлять пользователей о появлении новых данных. Это повышает актуальность информации и позволяет быстро реагировать на изменения, первым узнавая о новых событиях.

Таким образом, данная процедура представляет собой комплексное решение, ориентированное на безопасную и эффективную передачу данных между различными офисами, а также клиентами. Это достигается за счет

продуманного механизма обработки ошибок, гибких настроек правил обмена и подробного ведения записей в журнале событий. Эти аспекты делают алгоритм особенно важным в контексте современных требований к интеграции и управлению данными в распределенных информационных системах.

4.2 Тестирование и испытания информационной системы

В данном пункте рассматриваются основные подходы и методы, использованные для тестирования обмена данными в многоплатформенной корпоративной информационной системе, созданной на базе платформы «1С:Предприятие», а также приводятся результаты проверки её корректной работы.

Для тестирования модели использовались несколько подходов:

- Функциональное тестирование: проверка корректности выполнения обмена данными между компонентами системы, включая обмен между узлами РИБ и отправку уведомлений на клиентскую часть с использованием SSE. Особое внимание уделялось проверке сценариев обновления данных, создания новых записей и синхронизации данных между базами.
- Тестирование производительности: измерение времени, необходимого для синхронизации данных между распределёнными базами, а также время реакции на серверные изменения для отправки уведомлений через SSE. Тесты проводились с увеличением нагрузки, чтобы оценить, как система справляется с большим объёмом данных и количеством пользователей.
- Тестирование на отказоустойчивость: проверка способности системы корректно восстанавливать обмен данными при внезапных сбоях связи между узлами РИБ или потерях подключения клиентов, использующих SSE. Тесты включали отключение узлов системы,

симуляцию сбоев сети и проверку восстановления работы после восстановления связи.

- Тестирование безопасности: проверка правильности разграничения прав доступа к данным при обмене информацией. Это включало контроль на предмет предотвращения несанкционированного доступа к данным, которые передаются между узлами, и проверку защищенности SSE от потенциальных угроз, таких как атаки типа "человек посередине".

В процессе исследования методов обмена данными в корпоративной информационной системе на базе "1С:Предприятие" было реализовано решение, использующее технологию Server-Sent Events (SSE) для уведомлений об изменениях в базах данных. Использование SSE позволило обеспечить стабильную передачу обновлений с сервера на клиентскую сторону, что особенно важно для поддержания актуальности данных в режиме реального времени. В приложении Е мы видим, как данные в обновленной базе, отобразились в браузере.

Реализация механизма уведомлений на базе SSE показала свою эффективность благодаря минимальным задержкам при доставке информации о произошедших изменениях в данных. Данные уведомления об изменениях в информационной базе приходят практически мгновенно, что позволяет пользователям оперативно реагировать на актуальные изменения, происходящие в системе. Важно отметить, что SSE поддерживает постоянное соединение с сервером

Тесты распределенной информационной базы продемонстрировали, скорость передачи, безопасность, и масштабируемость будет завесить от построения архитектуры корпоративно информационной системы и продуманности её структуры. Не маловажным для нас будет протестировать обмен данными между различными платформами (рисунок 13, рисунок 14).

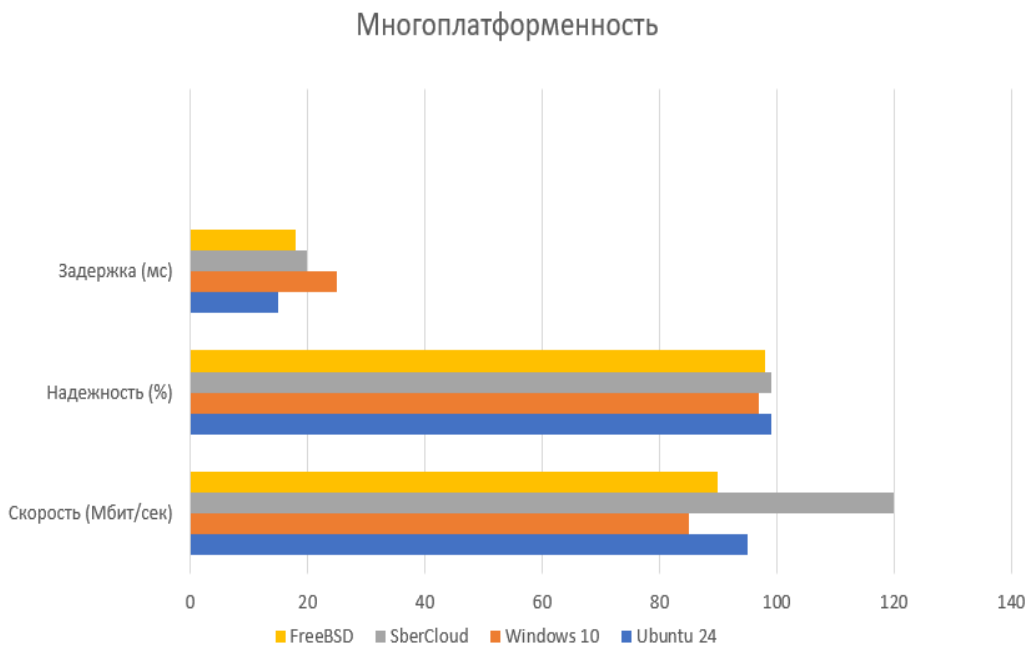

Рисунок 13 – Тесты обмена различных платформ

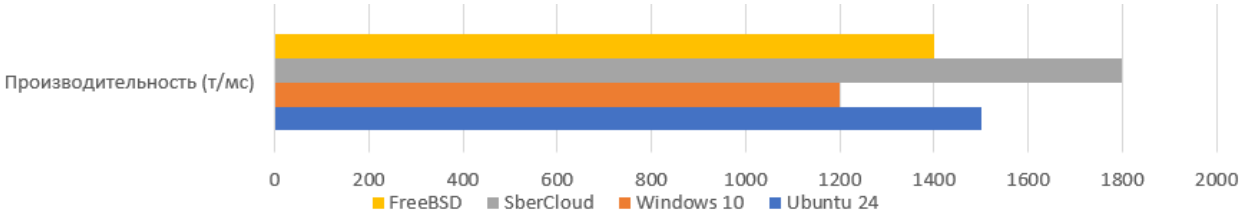

Рисунок 14 – Производительность платформ

Из таблицы 10 видно то, что обмен данными через распределенные информационные базы (РИБ) и с использованием технологии Server-Sent Events (SSE) не имеет значительных отличий в производительности на различных платформах. Это свидетельствует о том, что оба подхода обеспечивают высокую универсальность и могут быть эффективно использованы как в локальных, так и в облачных инфраструктурах. При этом облачные серверы выигрывают по ряду параметров, таких как масштабируемость, доступность, снижение затрат на обслуживание, удобство интеграции.

Таблица 10 – Результаты тестирования

Платформа	**Скорость передачи (Мбит/с)**	**Надежность (%)**	**Задержка (мс)**	**Производительность (транзакций/сек)**
Ubuntu 24	95	99	15	1500
Windows 10	85	97	25	1200
SberCloud	120	99	20	1800
FreeBSD	90	98	18	1400

Анализируя результаты надёжности обмена данными (рисунок 15), видно, что в большинстве случаев обмен данными происходил без существенных потерь и с высокой надежностью. Однако в цикле 3 и цикле 5 наблюдалась небольшая неточность в передаче данных. Это может быть связано с несколькими факторами. В частности, во время данных циклов возросла фоновая нагрузка на сеть из-за других действий, таких как обновление программного обеспечения или резервное копирование. Несмотря на небольшие отклонения, общая надежность системы остается в допустимых пределах.

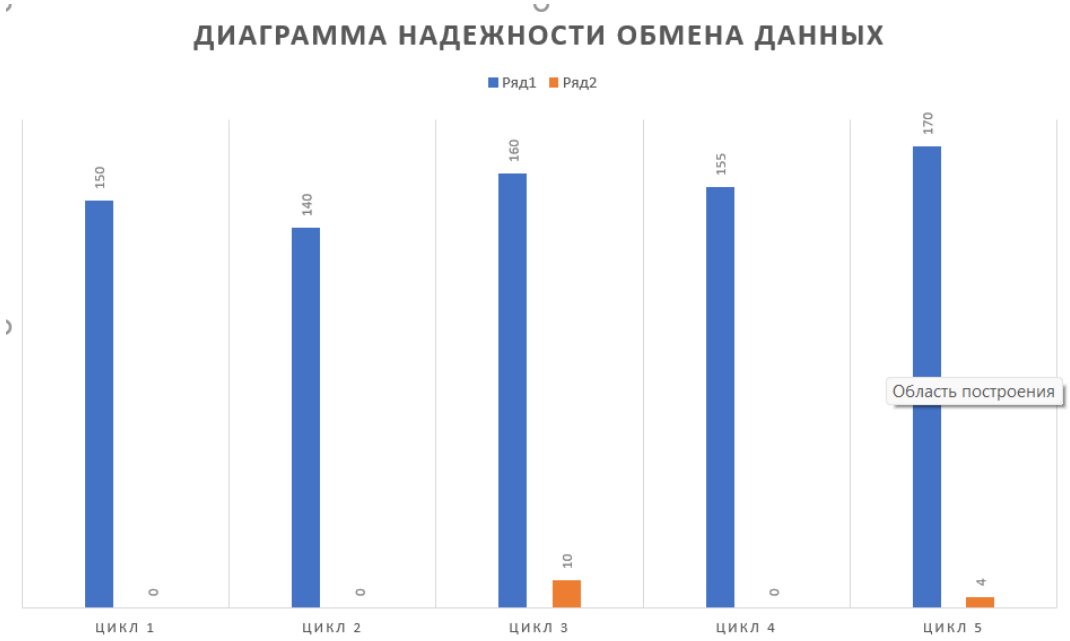

Рисунок 15 – Надежность обмена

Диаграмма масштабируемости (рисунок 16) показывает, что время обмена данными возрастает при росте количества филиалов. Показывает зависимость времени выполнения обмена данными от количества узлов системы.

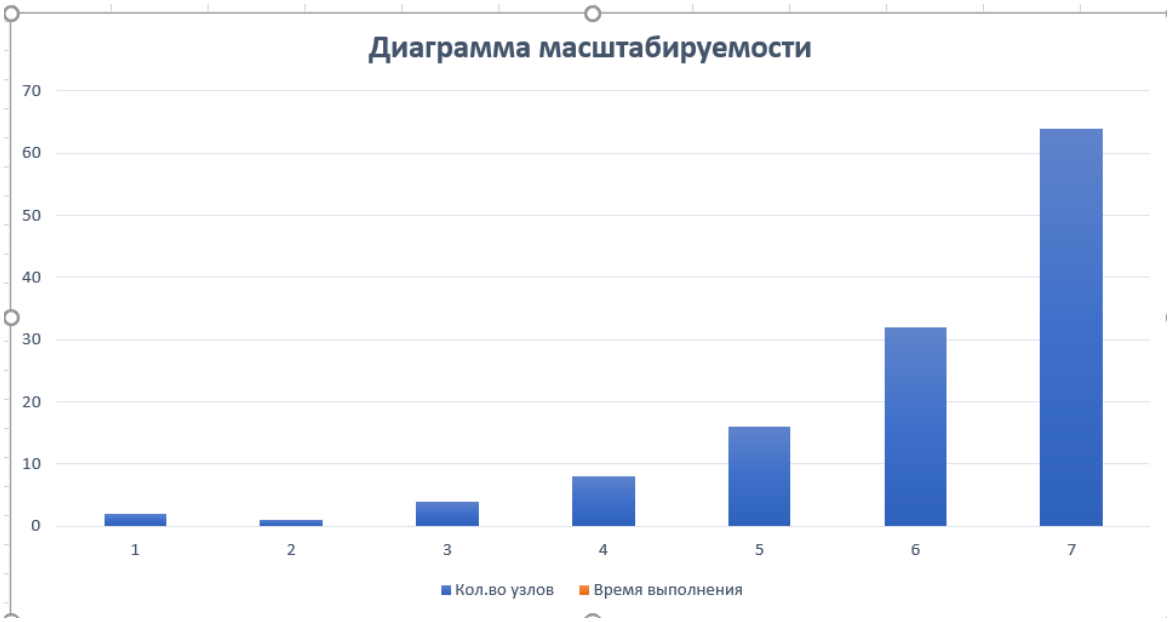

Рисунок 16 – Масштабируемость

Проведённое тестирование подтвердило, что разработанная модель обмена данными с использованием распределённых информационных баз и технологии SSE успешно справляется с задачами синхронизации данных и уведомления пользователей. Система продемонстрировала высокую устойчивость к сбоям, надёжность при больших объёмах данных и безопасность при передаче информации.

Таким образом, предложенная модель может быть рекомендована для внедрения в корпоративные информационные системы, требующие эффективного и безопасного обмена данными между различными узлами и платформами.

Тестирование позволило собрать данные о производительности, надежности и масштабируемости разработанной модели обмена данными.

Полученные результаты дали возможность оценить соответствие предложенной модели критериям корпоративной информационной системы и подтвердили ее применимость в условиях многоплатформенной среды.

Проведённое тестирование подтвердило, что разработанная модель обмена данными с использованием распределённых информационных баз и технологии SSE успешно справляется с задачами синхронизации данных и уведомления пользователей. Система продемонстрировала высокую устойчивость к сбоям, надёжность при больших объёмах данных и безопасность при передаче информации.

Таким образом, предложенная модель может быть рекомендована для внедрения в корпоративные информационные системы, требующие эффективного и безопасного обмена данными между различными узлами и платформами.

4.3 Область внедрения разработанной модели

Разработанная модель обмена данными на базе распределённых информационных баз (РИБ) и технологии Server-Sent Events (SSE) обладает широкой областью применения в корпоративных информационных системах. Эта модель ориентирована на многоплатформенные корпоративные среды, где требуется высокая надёжность, гибкость и оперативность при обмене данными между различными узлами и компонентами системы. В данной главе рассматриваются основные направления внедрения предложенной модели в различных отраслях и типах организаций.

Одной из ключевых областей применения разработанной модели является автоматизация бизнес-процессов в корпоративных информационных системах. Модель подходит для компаний, работающих с большими объёмами данных, которые распределены между различными филиалами или подразделениями. Использование распределённых информационных баз (РИБ) позволяет синхронизировать данные между удалёнными офисами в

режиме реального времени, обеспечивая консистентность информации и минимизируя риск несоответствий данных.

«Технология SSE, используемая в модели, позволяет оперативно уведомлять сотрудников о событиях и изменениях в корпоративной базе данных. Это может быть особенно полезно для систем управления взаимоотношениями с клиентами (CRM), систем управления ресурсами предприятия (ERP), а также для финансовых и бухгалтерских систем, где своевременное обновление данных имеет решающее значение» [3].

Разработанная модель также находит применение в системах управления и мониторинга, где необходимо обеспечить постоянное наблюдение за состоянием оборудования, ресурсов или процессов. Технология SSE позволяет оперативно передавать данные о состоянии системы в реальном времени, что позволяет быстро реагировать на возможные отклонения или неисправности.

Использование РИБ в таких системах позволяет обеспечить распределённое хранение данных и возможность синхронизации между различными узлами, что особенно важно для крупных предприятий с географически распределённой инфраструктурой. Это может быть актуально для промышленных предприятий, транспортных компаний и организаций, занимающихся логистикой.

Ещё одной областью внедрения разработанной модели является розничная торговля и складская логистика. В условиях больших объёмов данных, таких как информация о товарах, запасах, поставках и продажах, использование РИБ позволяет поддерживать актуальность данных в различных торговых точках и складах. Это снижает вероятность ошибок и позволяет оптимизировать управление запасами.

Технология SSE позволяет быстро уведомлять персонал о поступлении новых товаров, изменении цен или изменении статуса заказов, что повышает эффективность работы сотрудников и способствует улучшению клиентского сервиса. В условиях конкурентного рынка своевременное обновление данных

и уведомление персонала играет важную роль в обеспечении высокого уровня обслуживания клиентов.

Разработанная модель может быть использована в образовательных учреждениях для управления учебным процессом и обменом данными между различными подразделениями. РИБ позволяет синхронизировать данные между разными факультетами, учебными корпусами или филиалами, обеспечивая актуальность информации о расписании, учебных материалах и успеваемости студентов.

Технология SSE может использоваться для уведомления преподавателей и студентов о важных изменениях, таких как изменение расписания, новые задания или результаты экзаменов. Это способствует улучшению организации учебного процесса и повышению удобства для всех участников образовательного процесса.

Вывод по главе

В четвертой главе описана апробация разработанной модели на базе 1С:Предприятие 8. Проведены тестирование и испытания системы, а также определена область ее применения. Результаты апробации показали, что предложенная модель обеспечивает высокую производительность и надежность обмена данными, удовлетворяя требованиям к безопасности и масштабируемости. Разработанная модель может быть успешно внедрена в компаниях, использующих 1С:Предприятие для автоматизации бизнес-процессов, особенно в условиях необходимости интеграции разнородных платформ и модулей.

Заключение

В данном исследовании предпринята попытка комплексного анализа существующих методов и моделей обмена данными в многоплатформенных корпоративных информационных системах, с целью выявления наиболее эффективных подходов для их дальнейшего применения и улучшения. В процессе работы был осуществлён глубокий обзор текущих тенденций в области обмена данными, проанализированы преимущества и недостатки различных подходов, включая как традиционные, так и инновационные решения.

Особое внимание было уделено распределённой информационной базе (РИБ), которая в последнее время завоевала популярность благодаря своей гибкости и возможности эффективного распределения данных между различными платформами. Благодаря способности РИБ обеспечивать актуальность и согласованность данных в реальном времени, была выбрана именно эта модель для разработки рабочей модели на базе 1С:Предприятие.

Платформа 1С:Предприятие, обладая широкими функциональными возможностями и высокой адаптируемостью, стала идеальной основой для реализации предложенной модели. В процессе её внедрения удалось продемонстрировать значительные улучшения в обмене данными между различными модулями системы, благодаря чему повысилась оперативность и точность бизнес-процессов, а также уменьшились риски возникновения ошибок при передаче информации.

Проведен сравнительный анализ существующих методов обмена данными, предложена новая модель, сочетающая использование распределенных информационных баз (РИБ) и метода Server-Sent Events (SSE), а также разработана методика сравнительного тестирования методов обмена данными.

Экспериментальная проверка предложенной модели показала её высокую устойчивость к изменяющимся условиям эксплуатационной среды и

способность эффективно функционировать в условиях разнотипных платформ и нагрузок. Модель успешно справляется с такими задачами, как синхронизация данных, уменьшение задержек в их передаче и снижение избыточности информации. В результате компании получают возможность более гибко реагировать на изменения рынка и внутренние запросы, быстро внедряя необходимые изменения и улучшая свои бизнес-процессы.

Перспективы дальнейшего развития данной работы могут заключаться в адаптации разработанной модели под более широкий спектр бизнес-сценариев, а также в исследовании способов её интеграции с новыми технологическими решениями, включая облачные технологии и искусственный интеллект.

Список используемой литературы

1. Баринов В.А. Теория систем и системный анализ в управлении организациями / В.А. Баринов, Л.С. Болотова, В.Н..

2. Варфоломеева Е.В. Информационные системы в экономике: Учебное пособие / Е.В. Варфоломеева, Т.В. Воропаева и др.; Под ред. Д.В. Чистова - М.: НИЦ ИНФРА-М, 2015.

3. Виктор Бирюков, Владимир Дрожжинов Введение в CRM / «PC Week» 2001.

4. Гвоздева Т.В., Баллод Б.А. Проектирование информационных систем: учебное пособие. - Ростов н/Д.: Феникс, 2009.

5. Евдокимов В.В. и др. Экономическая информатика. /Учебник для вузов./ Под ред.д.э.н., профессора В.В. Евдокимова.-СПб.: Санкт-Петербург, 2007.

6. Егоров А. Основы теории управления. - М.: Физматлит. 2007.

7. Заботина Н.Н. Проектирование информационных систем: Учебное пособие / Н.Н. Заботина. - М.: ИНФРА-М, 2011.

8. Закорюкин В.Б. Надежность, эргономика и качество АСОИУ. - М.: МИРЭА. 2006.

9. Затонский А.В. Информационные технологии: разработка информационных моделей и систем: Учеб. пос. / А.В.Затонский - М.: ИЦ РИОР: НИЦ ИНФРА-М, 2014.

10. Иванова, Е.А. Особенности функционирования корпораций и корпоративное управление в современной экономике. Монография / Е.А. Иванова. - М. : МЭСИ, 2012.

11. Кнут Д. Искусство программирования, том 1. Основные алгоритмы. - М.: Вильямс, 2006.

12. Ковалев С., Ковалев В. Секреты успешных предприятий: бизнес-процессы и организационная структура. - М.: БИТЕК, 2012.

13. Кусов А.А. Проблемы интеграции корпоративных информационных систем. / Управление экономическими системами: электронный научный журнал. – 2011.

14. Лодон Дж., Лодон К. Управление информационными системами. / Пер. с англ. под ред. Трутнева Д.Р. - СПб.: Питер. 2005.

15. Мкртычев С.В. Объектно-структурное моделирование страховых информационных систем // Вектор науки Тольяттинского государственного университета.

16. Мухтарова Г. Внедрение ERP-систем. Основные ошибки/ Директор-инфо, 2003.

17. Пагонин В.А. Корпоративные информационные системы / В.А. Погонин, А.Г. Схиртладзе, С.И. Татаренко, С.Б. Путин. - Тамбов: Изд-во ФГБОУ ВПО «ТГТУ». 2012.

18. Пирогов, В.Ю. Информационные системы и базы данных: организация и проектирование: Учебное пособие / В.Ю. Пирогов. - СПб.: БХВ-Петербург, 2014.

19. Степанов Д.Ю. Перспективные направления развития корпоративных информационных систем на примере программных решений компании SAP. // Аспирант и соискатель. – 2013.

20. Степанов Д.Ю. Способы интеграции данных корпоративных информационных систем // Естественные и технические науки. – 2014.

21. Степанов Д.Ю. Формирование универсальных требований к пользовательским программам при подготовке спецификации на ABAP-разработку // Актуальные проблемы современной науки. – 2014.

22. Столингс В. Современные компьютерные сети / В. Столингс. - 2-е изд. - СПб. : Питер, 2003.

23. Титоренко Г.А. Автоматизированные информационные технологии в экономике: Учебник /Под ред. проф.-М.: Компьютер, ЮНИТП, 2007; университета. – 2013. - №1(23).

24. Уткин В., Балдин К. Информационные системы в экономике. - М.: Academia, 2012.

25. Фаулер, М. Шаблоны корпоративных приложений: пер. с англ. / М. Фаулер. - М.: Вильямс, 2016.

26. Филипенко И. Выбор ПО для автоматизации управления - Корпоративные системы, 2001.

27. Хаббард Дж. Автоматизированное проектирование баз данных - М.: Мир, 2014.

28. Хрусталева Е. Ю. Технологии интеграции 1С:Предприятия 8.3», 2023 г..

29. Шеннон К. Работы по теории информации и кибернетики. - М.: Информационная литература. 1963.

30. Artur E Web Scalability for Startup Engineers. - 1-е изд. - Boston: McGraw Hill; 1st edition, 2015.

31. Gregor H., Bobby W. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. - Addison-Wesley Professional; 1st edition, 2003.

32. Mark R., Neal F. Fundamentals of Software Architecture: An Engineering Approach. - 1-е изд. - Boston: O'Reilly Media; 1st edition, 2020.

33. Robert C., Martin S. Clean Architecture: A Craftsman's Guide to Software Structure and Design. - Pearson, 2017.

34. Tyler A., Slava C., Reuven L. Streaming Systems: The What, Where, When, and How of Large-Scale Data Processing. - O'Reilly Media, 2018.

Приложение А

Создание распределенной базы данных

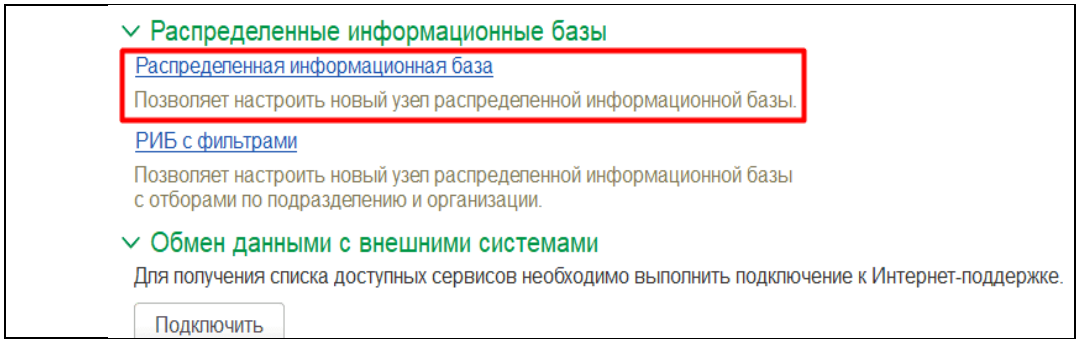

Рисунок А.1 – Создание

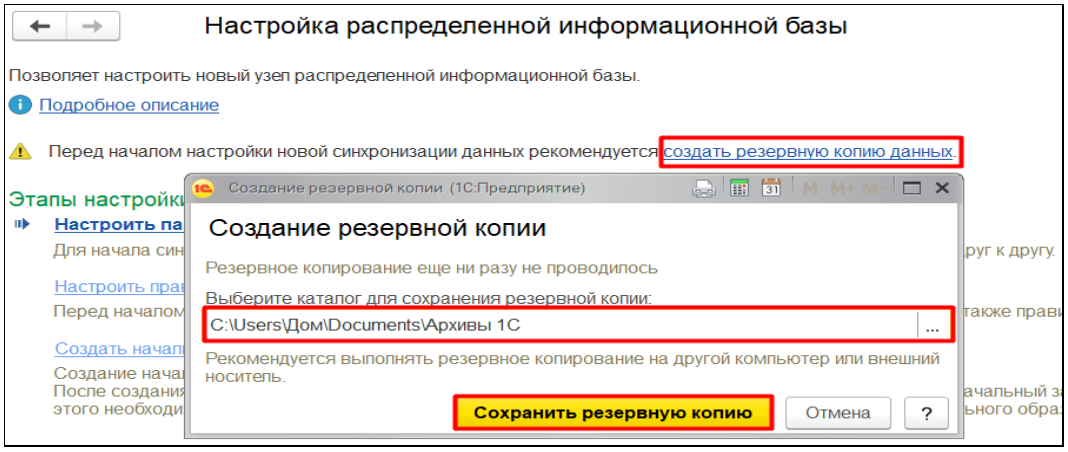

Рисунок А.2 – Настройка. Шаг 1

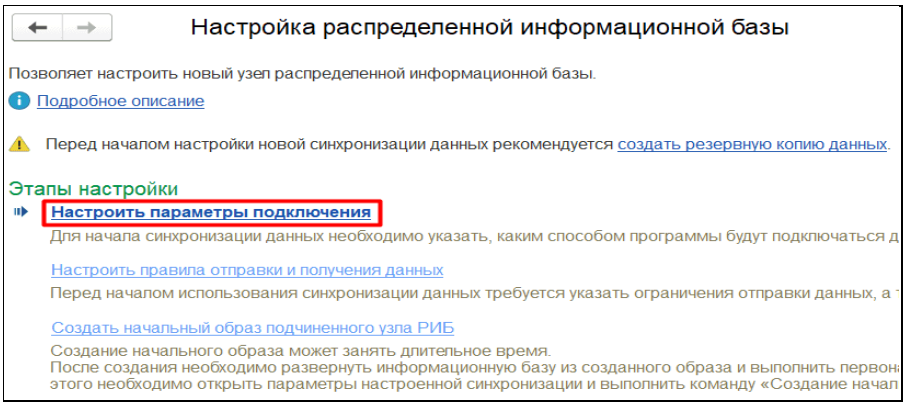

Рисунок А.3 – Настройка. Шаг 2

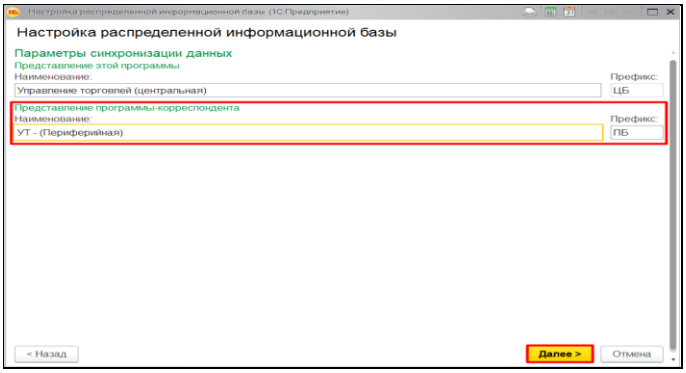

Рисунок А.4 – Настройка. Шаг 3

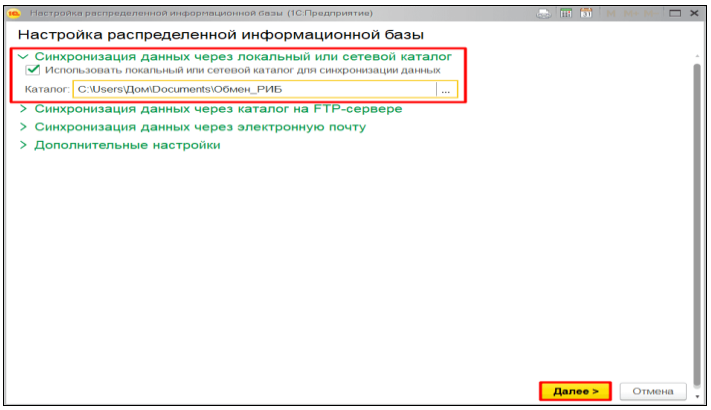

Рисунок А.5 – Настройка. Шаг 4

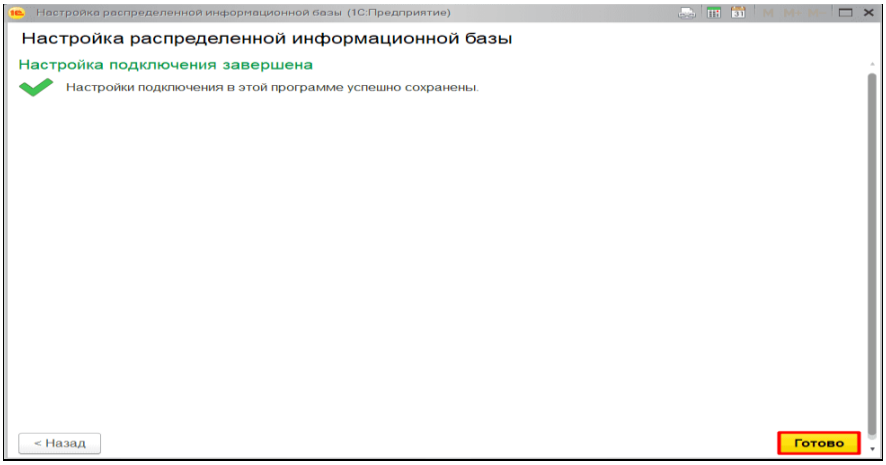

Рисунок А.6 – Настройка. Шаг 5

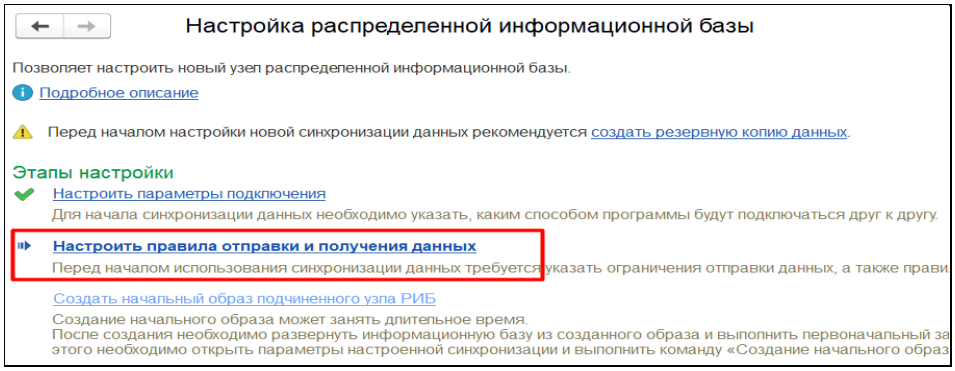

Рисунок А.7 – Настройка. Шаг 6

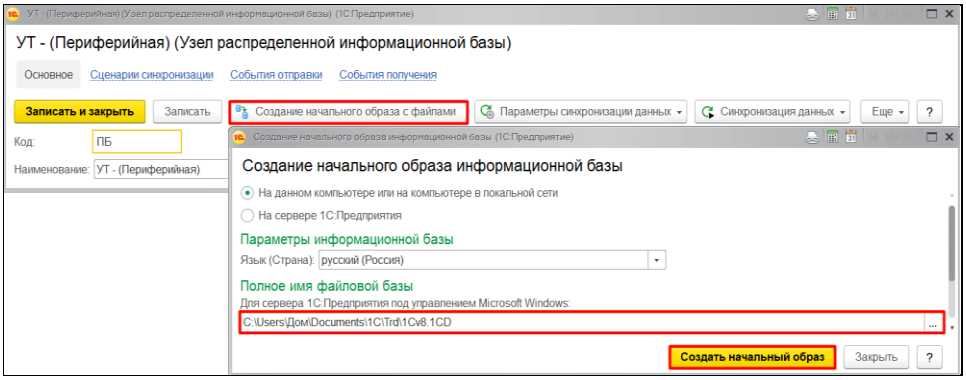

Рисунок А.8 – Настройка. Шаг 7

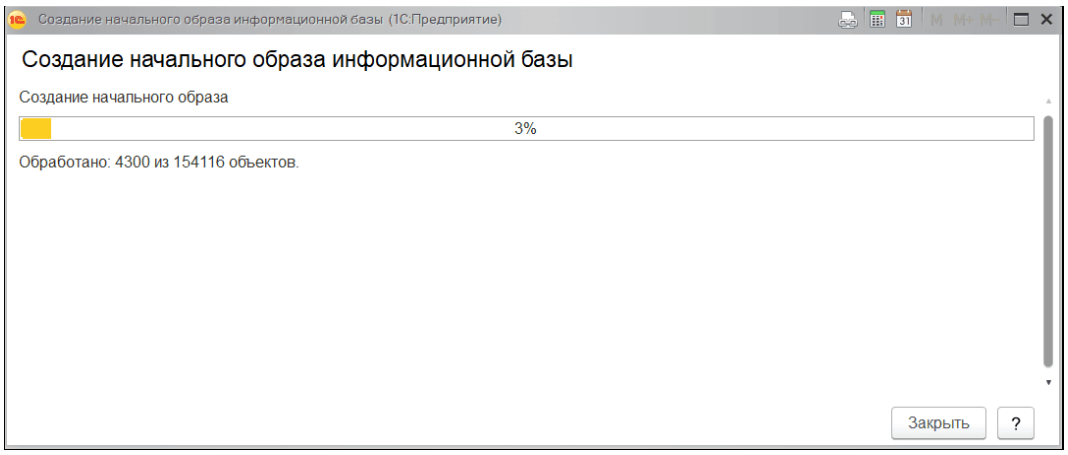

Рисунок А.9 – Настройка. Шаг 8

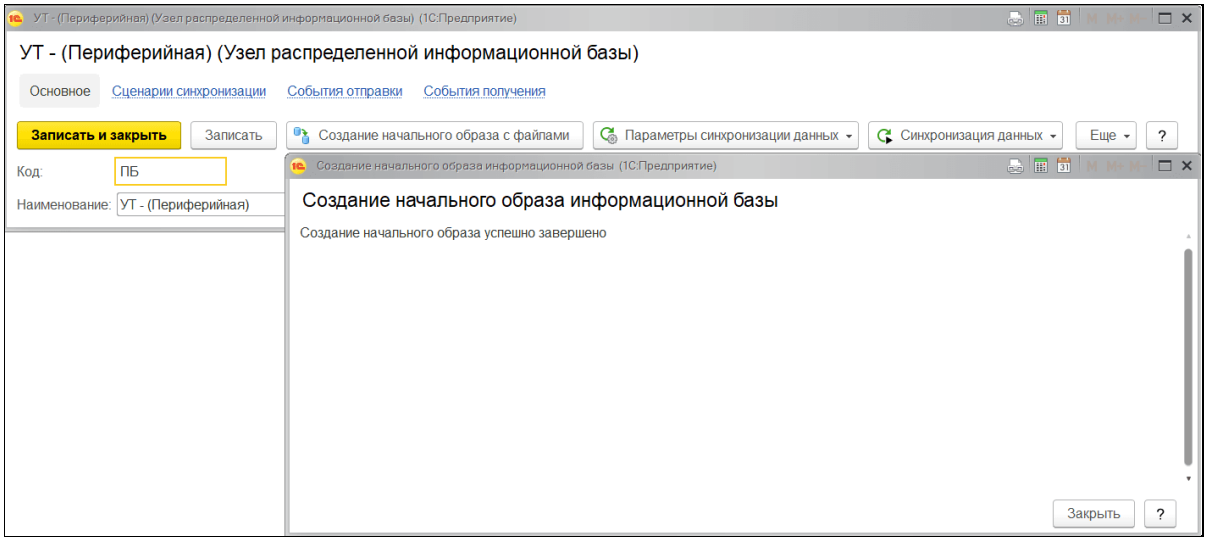

Рисунок А.10 – Настройка. Шаг 9

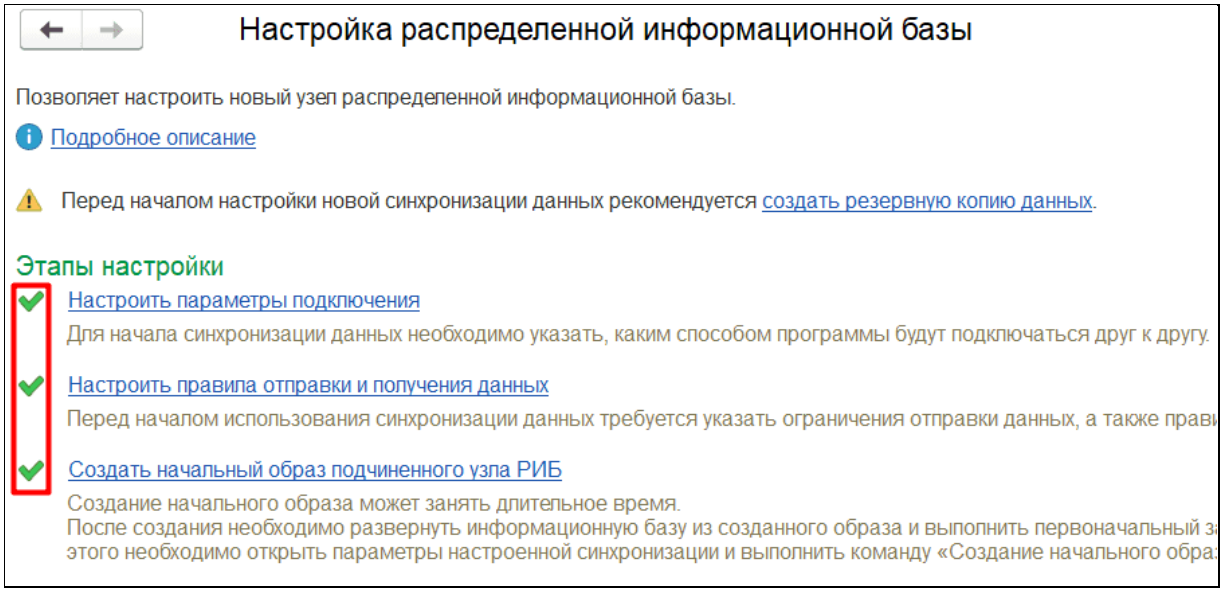

Рисунок А.11 – Настройка. Шаг 10

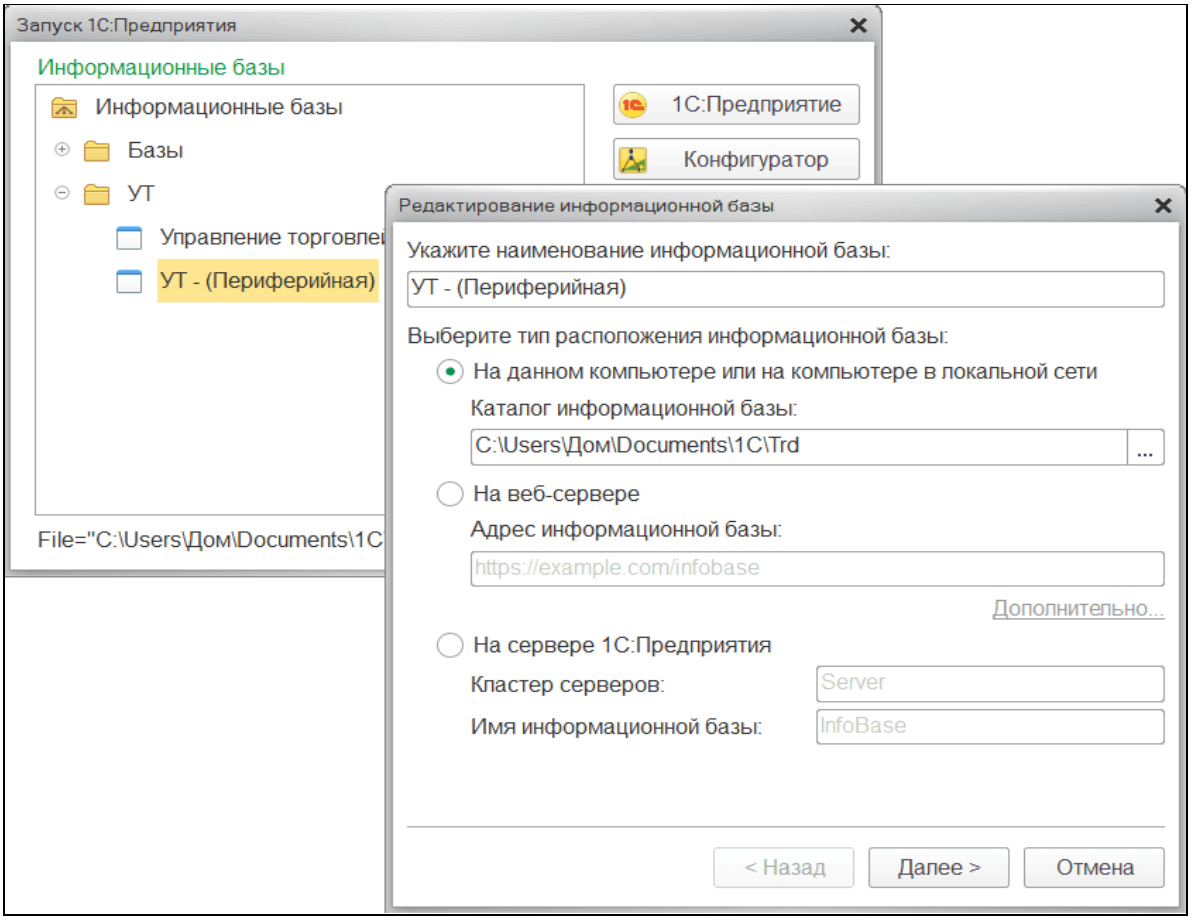

Рисунок А.12 – Запуск

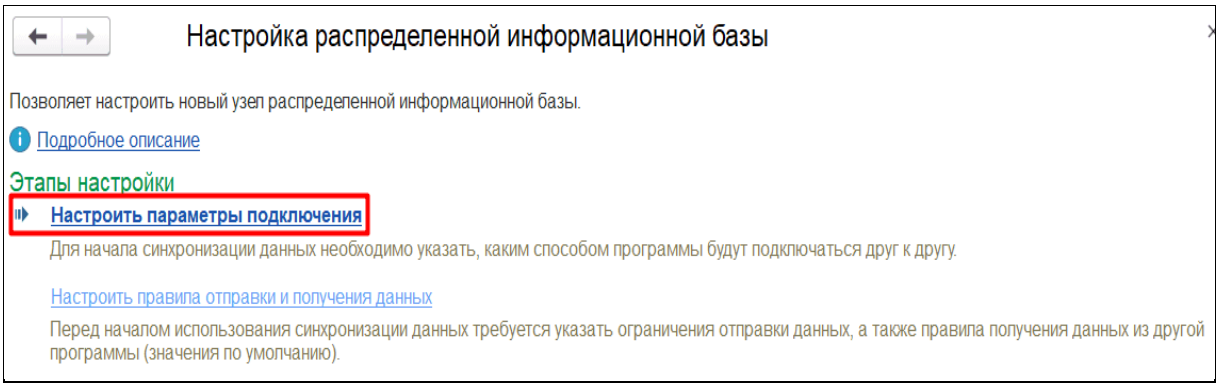

Рисунок А.13 – Настройка. Шаг 11

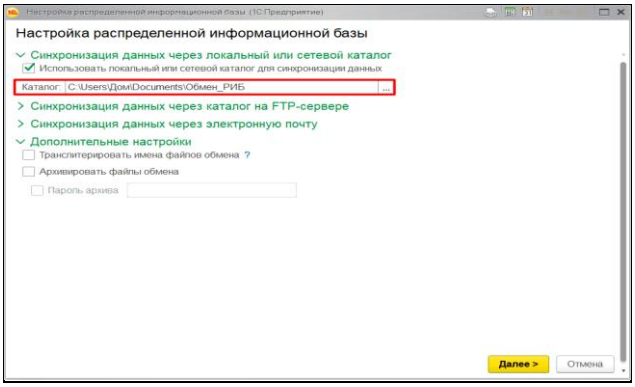

Рисунок А.14 – Настройка. Шаг 12

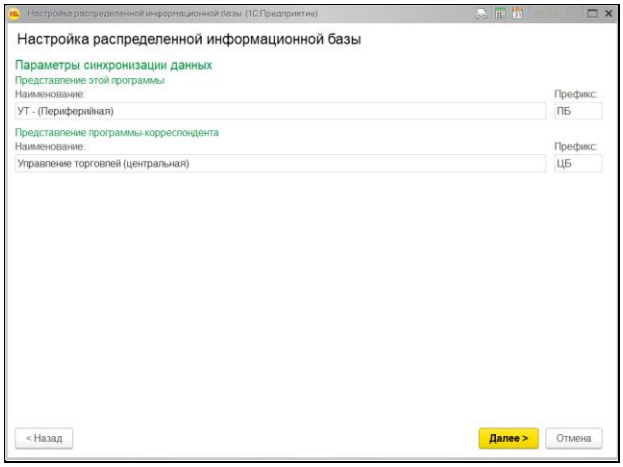

Рисунок А.15 – Настройка. Шаг 13

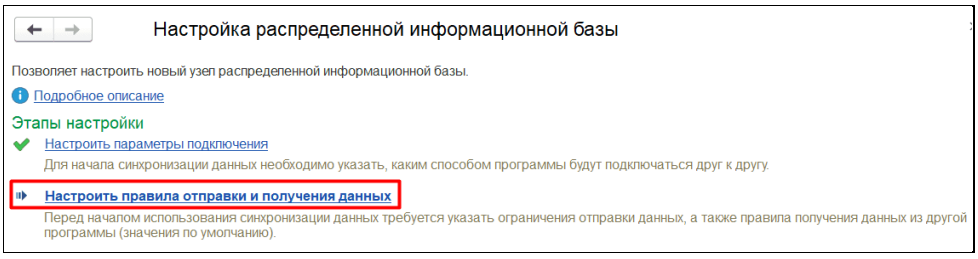

Рисунок А.16 – Настройка. Шаг 14

Приложение Б

Алгоритм обмена

```
Процедура ВыполнитьОбменДаннымиМеждуОфисами(Источник, Приемник)
    // Логгирование начала обмена
    ЗаписьЖурналаРегистрации("Начало обмена данными. Источник: " + Источник + ", Приемник: " + Приемник, УровеньЖурналаРе

    // Определение и настройка правил обмена данными
    ПравилаОбмена = Новый Структура;
    ПравилаОбмена.Вставить("Документы", Истина);
    ПравилаОбмена.Вставить("Справочники", Истина);

    Попытка
        // Загрузка данных из источника
        Если Не ОбменДанными.ЗагрузитьИз(Источник, ПравилаОбмена) Тогда
            ЗаписьЖурналаРегистрации("Ошибка загрузки данных из " + Источник, УровеньЖурналаРегистрации.Ошибка);
            Возврат;
        КонецЕсли;

        // Выгрузка данных в приемник
        Если Не ОбменДанными.ВыгрузитьВ(Приемник, ПравилаОбмена) Тогда
            ЗаписьЖурналаРегистрации("Ошибка выгрузки данных в " + Приемник, УровеньЖурналаРегистрации.Ошибка);
            Возврат;
        КонецЕсли;

        // Обновление результатов обмена и очистка временных данных
        ОбменДанными.ОчиститьДанные();

        // Логгирование успешного завершения обмена
        ЗаписьЖурналаРегистрации("Обмен данными между офисами завершен успешно. Источник: " + Источник + ", Приемник: " +

    Исключение
        // Обработка ошибок и логгирование
        СообщениеОбОшибке(ОписаниеОшибки());
        ЗаписьЖурналаРегистрации("Обмен данными между офисами завершился с ошибкой: " + ОписаниеОшибки(), УровеньЖурналаР
    КонецПопытки;
КонецПроцедуры
```

```
Функция ПолучитьОтфильтрованныеДанные(КритерииОтбора)
        // Пример фильтрации данных на стороне источника
        Запрос = Новый Запрос;
        Запрос.Текст = "ВЫБРАТЬ ... ИЗ Справочник.Данные КАК Данные ГДЕ Данные.Критерий = &Критерий";
        Запрос.УстановитьПараметр("Критерий", КритерииОтбора);
        Результат = Запрос.Выполнить().Выгрузить();
        Возврат Результат;
    КонецФункции
```

Дай определение этому коду, опиши что происходит

Приложение В

Код программы

```
const http = require('http');

let clients = [];

http.createServer((req, res) => {
    if (req.url === '/events') {
        res.writeHead(200, {
            'Content-Type': 'text/event-stream',
            'Cache-Control': 'no-cache',
            'Connection': 'keep-alive'
        });
        clients.push(res);

        req.on('close', () => {
            clients = clients.filter(client => client !== res);
        });
    }

    if (req.url === '/new-order') {
        let body = '';
        req.on('data', chunk => {
            body += chunk;
        });
        req.on('end', () => {
            clients.forEach(client => client.write(`data: ${body}\n\n`));
            res.writeHead(200);
            res.end();
        });
    }
}).listen(8080);
```

Приложение Г

Алгоритм специальной обработки

```
  Процедура ПослеСинхронизацииРИБ()
    // Получаем список новых заказов, которые были синхронизированы с филиалов
    НовыеЗаказы = ЗапросНовыхЗаказов();

    // Создаем JSON объект с информацией о новых заказах
    ДанныеДляОтправки = Новый Структура("Тип", "НовыйЗаказ", "Данные", НовыеЗаказы);
    ТелоСообщения = ТекстJSON(ДанныеДляОтправки);

    // Отправляем HTTP-запрос на сервер SSE
    HTTPСоединение = Новый HTTPСоединение("http://localhost", 8080);
    HTTPСообщение = HTTPСоединение.ПолучитьHTTPСообщение("POST", "/new-order",
    Новый Структура("Content-Type", "application/json"));
    HTTPСообщение.УстановитьТело(ТелоСообщения);
    HTTPСоединение.Отправить(HTTPСообщение);
КонецПроцедуры
```

Приложение Д

Пример программного года HTML

```
<script>
    const eventSource = new EventSource('http://localhost:8080/events');
    eventSource.onmessage = function(event) {
        const данные = JSON.parse(event.data);
        if (данные.Тип === 'НовыйЗаказ') {
            alert('Поступил новый заказ от филиала: ' + JSON.stringify(данные.Данные))
        }
    };
</script>
```

Приложение Е

Отображение в браузере данных в обновленной базе

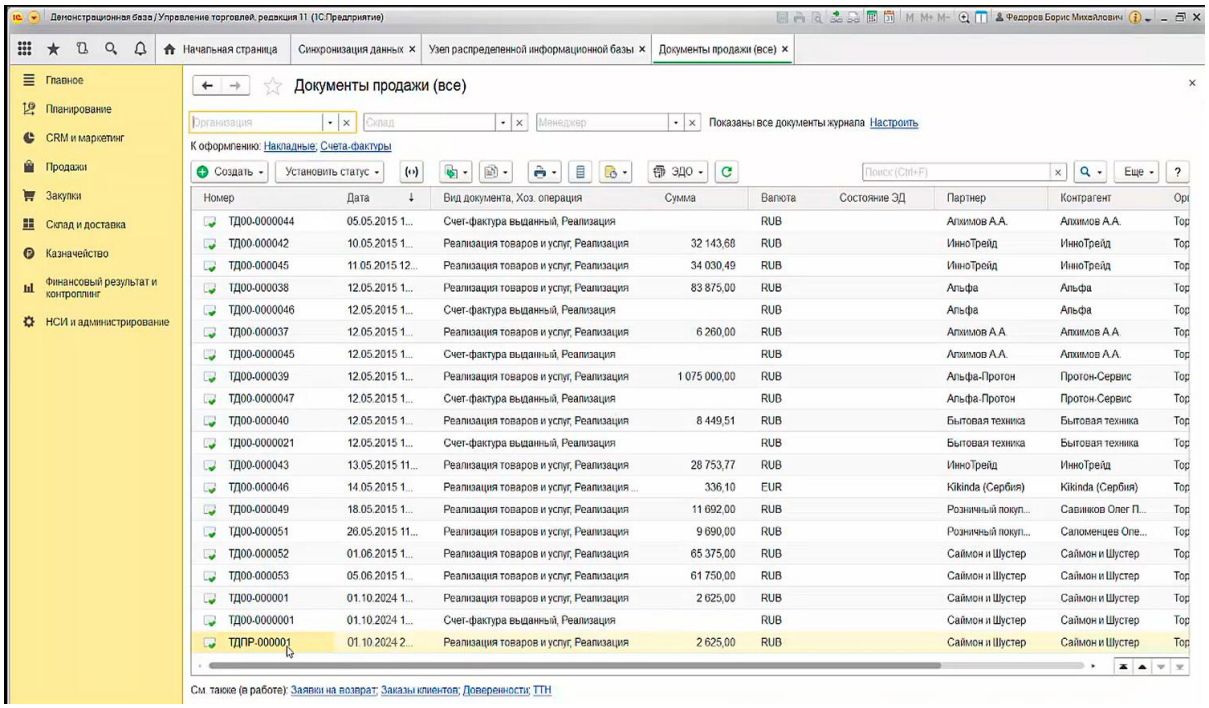

Рисунок Е.1 – Основной вид

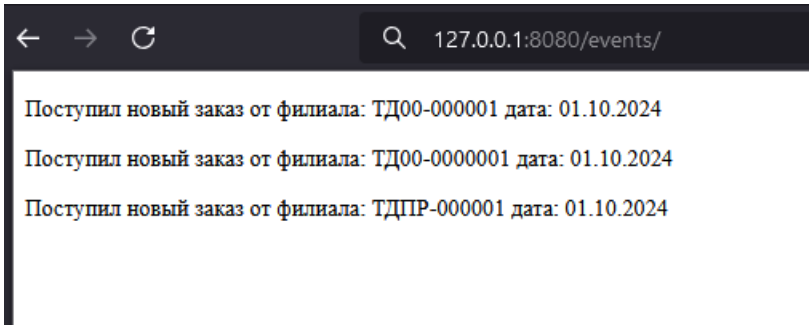

Рисунок Е.2 – Результат