ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
МОРДОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМ. Н. П. ОГАРЁВА»**

Институт электроники и светотехники

Кафедра автоматизированных систем обработки информации и управления

УТВЕРЖДАЮ

Зав. кафедрой

к. т. н., проф.

_____С. А. Федосин

(подпись)

«___»_____2020 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

РАЗРАБОТКА АЛГОРИТМА ОБЪЕДИНЕНИЯ РАСПРЕДЕЛЕННЫХ БАЗ ДАННЫХ

Автор магистерской диссертации (подпись) (дата) А. В. Бальзамов

Обозначение магистерской диссертации МД–02069964–09.04.01–02–20

Направление 09.04.01 Информатика и вычислительная техника

Руководитель работы

канд. техн. наук, проф. (подпись) (дата) С. А. Федосин

Нормоконтролер

канд. техн. наук (подпись) (дата) А. А. Аббакумов

Рецензент

канд. техн. наук, доц. (подпись) (дата) В. С. Дубровин

Саранск

2020

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
МОРДОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМ. Н. П. ОГАРЁВА»**

Институт электроники и светотехники

Кафедра автоматизированных систем обработки информации и управления

УТВЕРЖДАЮ
Зав. кафедрой
к. т. н., проф.
_____С. А. Федосин
(подпись)
«___»_____2018 г.

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

(в форме магистерской диссертации)

Студент Бальзамов Александр Витальевич

1 Тема «Разработка алгоритма объединения распределенных баз данных»

Утверждена по МордГу № 41815 от 07.11.2018

2 Срок представления работы к защите 26.06.2020

3 Исходные данные для научного исследования (проектирования)

база методических материалов и корпуса документов по различным дисциплинам

4 Содержание выпускной квалификационной работы

4.1 Основные теоретические положения 14

4.2 Техническое задание 31

4.3 Разработка системы автоматизированного объединения распределенных баз данных 47

Руководитель работы _____ С. А. Федосин

подпись, дата

Задание принял к исполнению _____ А. В. Бальзамов

подпись, дата

№ строки	Формат	Обозначение	Наименование	Кол. листов	Примечание
1					
2			Документация текстовая		
3					
4	А4	МД-02069964-09.04.01-02-20	Пояснительная записка	81	
5					
6			Документация графическая		
7					
8	А1	МД-02069964-09.04.01-02-20	Диаграмма вариантов	1	
9			использования		
10	А1	МД-02069964-09.04.01-02-20	Диаграмма последовательности	1	
11			для работы пользователя с		
12			системой		
13	А1	МД-02069964-09.04.01-02-20	Диаграмма развертывания	1	
14			системы		
15	А1	МД-02069964-09.04.01-02-20	Диаграмма деятельности	1	
16					
17			Документация прочая		
18					
19	А4	МД-02069964-09.04.01-02-20	Частичный программный код	41	Прил. А
20			разработанных модулей		
21	А4	МД-02069964-09.04.01-02-20	Графические материалы	3	Прил. Б
22	А4	МД-02069964-09.04.01-02-20	Акт о внедрении	1	Прил. В
23			научно-технических		
24			результатов работы		

Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв.№ подл.

Изм.	Лист	№ докум.	Подп.	Дата
Разраб.		Бальзамов		
Пров.		Федосин		
Н.контр.		Аббакумов		
Утв.		Федосин		

МД-02069964-09.04.01-02-20

Разработка алгоритма объединения распределенных баз данных

Лит.	Лист	Листов
	4	137

МГУ им. Н.П. Огарева, ИЭС, ИВТ, 641

РЕФЕРАТ

Пояснительная записка содержит 81 лист, 30 рисунков, 3 таблицы, 31 использованный источник, 4 приложения.

СИСТЕМА, БАЗЫ ДАННЫХ, РАСПРЕДЕЛЕННЫЕ БАЗЫ ДАННЫХ, АЛГОРИТМ, ДЕРЕВО ВЗАИМОСВЯЗЕЙ В РАПСРЕДЕЛОННОЙ БАЗЕ ДАННЫХ, СИСТЕМА УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ.

Объектом разработки является система объединения распределенных баз данных.

Цель работы – создание системы объединения распределённых баз данных для решения задач по автоматизированному объединению распределённых баз данных различных информационных систем.

В результате проведенной работы был проведен анализ предметной области, разработаны алгоритмы по определению и объединению бизнес-сущностей и справочных сущностей, построена диаграмма вариантов использования, диаграммы последовательности и диаграмма развертывания. Также при реализации графического интерфейса пользователя, были разработаны вертикальные и горизонтальные прототипы приложения.

Эффективность разработанной системы заключается в автоматизации задачи объединения распределенных баз данных различных информационных систем, в частности в автоматизированной информационной системе «Электронный Социальный Регистр Населения Республики Мордовия».

Разработанная система может быть применена образовательными, государственными и иными организациями, где есть системы, основанные на распределённых базах данных для их автоматизированного объединения.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность. В современном мире почти все информационные системы и различное программное обеспечение обрабатывают и хранят большие объемы различных данных. Для осуществления эффективного хранения и обработки больших объемов данных используются базы данных. Поэтому в любых информационных системах, как правило, одной из важнейших подсистем является подсистема работы с базой данных.

База данных – это организованный набор структурированной информации или данных, обычно хранящихся в электронной форме в компьютерной системе. База данных обычно управляется системой управления базами данных (СУБД). Вместе данные и СУБД, а также связанные с ними подсистемы называются системой баз данных, часто просто сокращенно до «база данных».

Данные в наиболее распространенных типах баз данных, работающих сегодня, как правило, моделируются в строках и столбцах, которые в свою очередь принадлежат определенной таблице, чтобы сделать обработку и запрос данных эффективными. После этого данные могут быть легко доступны, управляться, модифицироваться, обновляться, контролироваться и организовываться.

Большинство реляционных баз данных используют язык структурированных запросов (SQL) для создания, модификации и управления данными. Данный язык структурированных запросов позволяет не просто просматривать и модифицировать данные, а также строить сложные SQL запросы, позволяющие каскадно модифицировать и управлять данными, связанными между собой различными внешними ключами по заданными условиям и критерием.

В крупных информационных системах, которые хранят и обрабатывают большое количество данных, разделяют базы данных в соответствии с каким-либо признаком из предметной области, например, таким признаком чаще всего

выступает: департамент, салон, муниципальное образование и так далее. Вследствие такого деления, данные базы данных становятся децентрализованными или распределенными. К плюсам такого подхода можно отнести:

– каждая база данных хранит только ту информацию, которая относится к признаку деления;

– уменьшается объем данных в каждой базе, что ведет к уменьшению стоимости дискового пространства, нужного для каждой базы данных;

– решается вопрос с правами доступа к информации, не относящийся к данному признаку деления;

– уменьшается риск разовой потери всех данных в случае аварии и других программных сбоев.

К минусам такого подхода можно отнести:

– каждая база данных требует обслуживания;

– возрастает нагрузка на вычислительные мощности сервера или кластера серверов;

– встает вопрос с хранением общих данных, которые не поддаются делению по выбранному признаку;

– синхронизация данных между различными базами данных;

– объединение баз данных в случае их унификации по признаку деления.

Не исключением является и Министерство социальной защиты, труда и занятости населения Республики Мордовия, использующее автоматизированную информационную систему «Электронный социальный регистр населения Республики Мордовия» (АИС ЭСРН РМ). Данная система основана на платформе Sitex, которая представляет набор механизмов для построения и функционирования основных подсистем, включая ORM для работы с БД. Далее в эту платформу интегрируется нужная бизнес логика для построения полноценной автоматизированной информационной системы.

Данная АИС ЭСРН РМ расположена во всех муниципальных районах Республики Мордовия, где создается отдельная БД для хранения и обработки

информации только своего муниципального образования. Это показывает, что данная система является децентрализованной и, как следствие, появляются задачи по объединению баз данных различных муниципальных районов (например, в случае объединения районных учреждений социальной защиты).

Цель магистерской диссертации **–** разработка методов и алгоритмов объединения распределенных баз данных на примере АИС ЭСРН РМ и программного комплекса, позволяющего решать задачи автоматизированного объединения распределенных баз данных различных муниципальных образований.

Для достижения поставленной цели необходимо решить ***следующие задачи***:

– проанализировать литературу и современные методы в области объединения распределенных баз данных;

– провести анализ предметной области и программно-аппаратного комплекса, на которой функционирует АИС ЭСРН РМ;

– разработать техническое задание на реализацию программного комплекса по автоматизированному объединению распределенных баз данных;

– реализовать систему, решающую задачу автоматизированного объединения распределенных баз данных муниципальных районов АИС ЭСРН РМ;

– спроектировать и реализовать клиентский интерфейс пользователя на основе веб-приложения для более удобной и эффективной работы с системой.

Результатом научно-конструкторских работ стал программный комплекс, содержащий систему объединения распределенных баз данных, решающую задачу автоматизированного объединения баз данных муниципальных районов в АИС ЭСРН РМ, и графический интерфейс пользователя на основе веб-приложения.

Решение поставленных задач определило структуру выпускной квалификационной работы. В первой части проведено исследование инструментальной платформы для построения бизнес-приложений Sitex,

рассмотрены теоретические положения об распределенных базах данных и метаинформации.

Во второй – разработано техническое задание по задаче объединения распределенных баз данных муниципальных районов в АИС ЭСРН РМ.

В третьей главе описана реализация основных алгоритмов для системы объединения распределенных баз данных: алгоритм определения дерева взаимосвязей бизнес-сущностей, алгоритм определения и объединения справочных сущностей, методы объединения бизнес-сущностей. Также были реализованы дополнительные подсистемы логирования и резервного копирования баз данных.

В четвертой части описана реализация пользовательского интерфейса на основе веб-приложения, а именно проектировка горизонтального и вертикального прототипа, выбор средств реализации и демонстрация получившегося интерфейса.

Объект исследования – распределённые базы данных.

Предмет исследования – методы и алгоритмы объединения распределенных баз данных.

Методология и методы исследования. При выполнении магистерской диссертационной работы использованы методы функционального анализа, системного анализа, построения графических интерфейсов.

Научная новизна и теоретическая значимость работы. Новизна данной магистерской диссертационной работы заключается в том, что в ней:

– разработаны алгоритмы объединения распределенных баз данных на основе АИС ЭСРН РМ и инструментальной платформы Sitex, *отличающиеся* своей уникальностью и не имеющих аналогичных решений.

– разработан программный комплекс, включающий в себя систему объединения распределенных баз данных в АИС ЭСРН РМ, и пользовательский графический интерфейс на основе веб-приложения;

– были проведены реальные испытание на объединение распределенных баз данных АИС ЭСРН РМ Кадошкинского и Инсарского района, и дальнейшее внедрение данной системы Министерством социальной защиты населения.

Результаты исследований докладывались на конференции «Огаревские чтения».

Практическая значимость. В ходе выполнения работы был создан программный комплекс, включающий в себя систему объединения распределенных баз данных, который решает задачу автоматизированного объединения распределенных баз данных АИС ЭСРН РМ различных муниципальных районов, а так же графический пользовательский интерфейс на основе веб-приложения, который позволил более эффективно и удобно работать с системой. Данный программный комплекс был внедрен в Министерстве социальной защиты населения, что подтверждается настоящим актом о внедрении научно-технических результатов работы.

1 Основные теоретические положения

Так как реализация задачи автоматизированного объединения распределенных баз данных будет на основе автоматизированной информационной системы «Электронный социальный регистр населения Республики Мордовия» (АИС ЭСРН РМ), использующейся в Министерстве социальной защиты населения Республики Мордовия, то необходимо провести функциональный и структурный анализ данной АИС

Данная система основана на инструментальной платформе разработки бизнес-приложений Sitex, которая представляет набор механизмов и готовых базовых подсистем для построения различных АИС и функционирования основных бизнес-процессов, включая ORM для работы с различными провайдерами баз данных, подсистему хранения файлов, подсистему создания новых различных объектов системы (бизнес, справочных и др.), подсистему планирования задач, различные подсистемы интеграции со сторонними государственными сервисами (например СМЭВ – система межведомственного электронного взаимодействия), а так же множество других вспомогательных подсистем. Далее на основе это платформы реализуется нужная бизнес-логика, физическая модель базы данных и ее объекты в системе, для построения полноценной автоматизированной информационной системы.

Одно из основных преимуществ данной платформы для построения бизнес-приложений заключается в том, что она позволяет в дальнейшем не только специализированным программистам, но и обычным администраторам системы создавать, редактировать, удалять или редактировать политику безопасности нужного объекта в системе в реальном времени без написания какого-либо программного кода, описывая их в терминах системы с использованием интерактивной графической оболочки.

АИС ЭСРН РМ расположена во всех муниципальных районах Республики Мордовия, где создается отдельная БД для хранения и обработки информации

только своих муниципальных образований. Это показывает, что данная система является распределенной и, как следствие, появляются задачи по объединению баз данных различных муниципальных районов.

1.1 Инструментальная система разработки бизнес-приложений SiTex

1.1.1 Назначение системы

Инструментальная система разработки бизнес-приложений SiTex (Система) является платформой для создания корпоративных распределенных систем и поддерживает:

– реализацию хранилища данных и бизнес-логики работы приложения, созданного на основе Системы;

– решение широкого круга задач, описание их в терминах Системы;

– расширение системы, как с точки зрения задач, решаемых системой, так и используемых технологий;

– встроенную подсистему безопасности;

– встроенную систему репликации между несколькими системами;

– механизмы управления потоками работ.

Система реализует трехзвенную архитектуру (слой хранения данных, слой бизнес-логики и презентационный слой), позволяющую оптимально распределить нагрузку на составляющие распределенной системы.

Использование объектной модели в качестве основы Системы дает возможность описывать любые объекты в терминах предметной области.

Реализация набора шаблонов проектирования в процессе разработки системы является ключом для повторного использования кода. Расширение функций Системы возможно посредством инкрементального построения и динамической загрузки классов.

Код ядра системы написан на Java. В качестве базового хранилища данных может использоваться одна из перечисленных БД – MS SQL Server, Oracle, MySQL.

Система предоставляет как интерфейс прикладного программирования ядра (API ядра), так и его использующий прикладной API. Функции прикладного API непосредственно используются в коде прикладного приложения.

При этом определены интерфейсы, инкапсулирующие методы работ с различными типами хранилищ данных и различными провайдерами безопасности, таким образом, определены пути по организации работы системы с хранилищами различных типов: база данных, файловая система.

Система предназначена как для конечных пользователей, так и прикладных программистов. Возможности Системы как средства разработки иллюстрирует рисунок 1.

Главной особенностью Системы является реализация методики быстрой разработки бизнес-приложений, включающей следующие этапы:

– описание сущностей (классов) предметной области и связей между ними в терминах метамодели Системы, которая предоставляет набор базовых операций над объектами этих классов (создание, редактирование свойств, удаление, копирование, перемещение, определение набора прав доступа и т.п.);

– написание кода бизнес операций на Java с использованием API системы;

– регистрация в Системе операций, как общих для разных предметных областей, так и специфичных. Зарегистрированные операции можно связать с любым классом объектов.

Так же, одной из особенностью системы является реализация методики расширения функциональности:

– описание новых классов (например, классов новых документов) объектов бизнес – процессов и связей между ними в терминах метамодели системы, которая предоставляет набор базовых операций над объектами этих классов (создание, редактирование свойств, удаление, копирование, перемещение, определение набора прав доступа и т.п.), без программирования;

– определение операций по работе с созданными классами объектов. В системе реализован набор базовых операций (шаблонов) над объектами Системы. Разработчику приложения на базе системы достаточно их специфицировать, задав параметры;

– в случае необходимости – написание кода бизнес операций на Java с использованием API системы. В Системе реализованы базовые механизмы управления потоками работ (бизнес-процессами), позволяющие свести прикладное программирование к минимуму, требующее от прикладного программиста написание только кода атомарной операции работы.

Консоль управления ресурсами Системы предоставляет удобный интерфейс (похожий на интерфейс программы Проводник Windows) для описания сущностей любой предметной области.

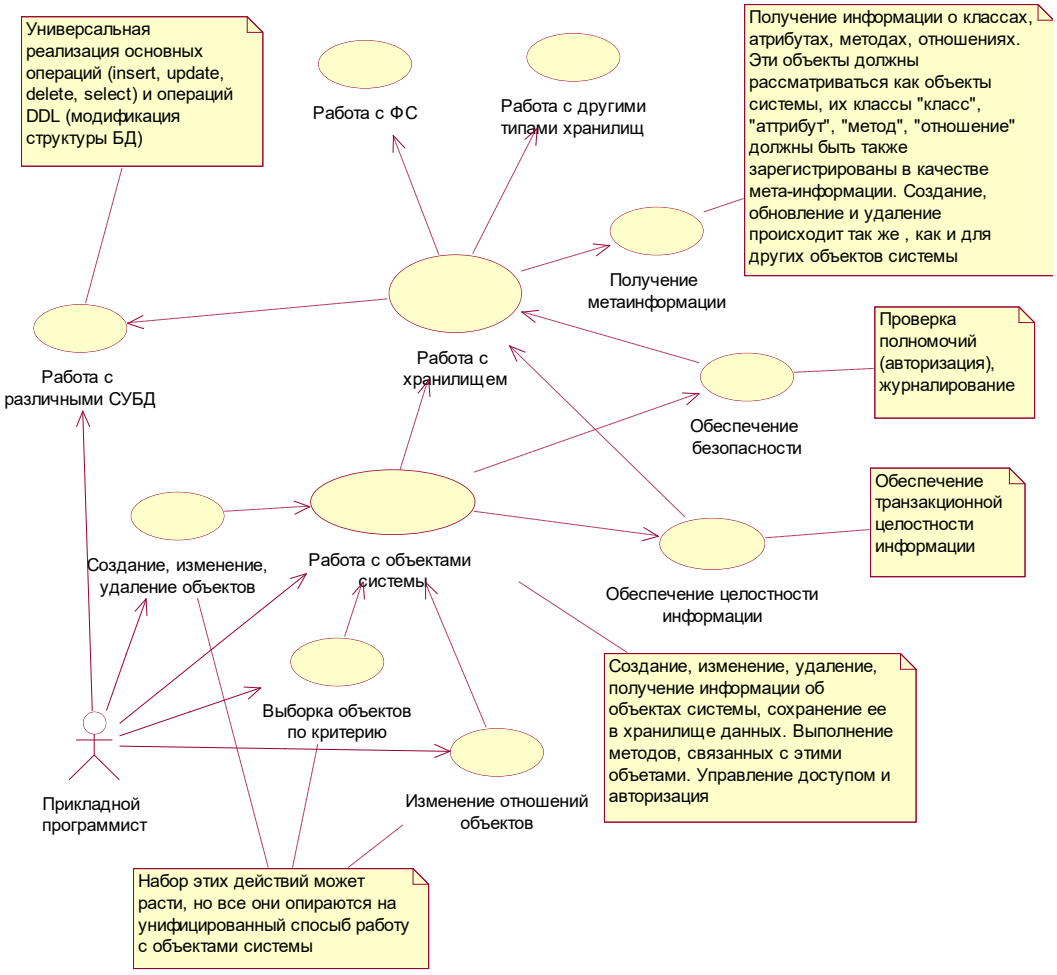


Рисунок 1 – Функции, предоставляемые Системой

1.1.2 Архитектура системы

Любое приложение, построенное на базе Системы, состоит из трех частей: клиента, сервера (ядра) и систем постоянного хранения. В качестве системы постоянного хранения могут выступать СУБД, LDAP, файловая система, XML хранилище и т.д.

Идеологически Система реализует стандартную трехзвенную модель, где обмен данными между клиентом и сервером происходит в виде структурированных сообщений. Клиентом системы может являться как стандартное GUI или WEB приложение (стандартный клиент) так и другая система построенная по принципу обмена структурированными сообщениями.

Ядро Системы обрабатывает запросы клиента и возвращает результат в виде сообщения определенного формата.

Системы постоянного хранения используются в качестве долговременного хранилища объектов.

Общая архитектура системы изображена на рисунке 2.

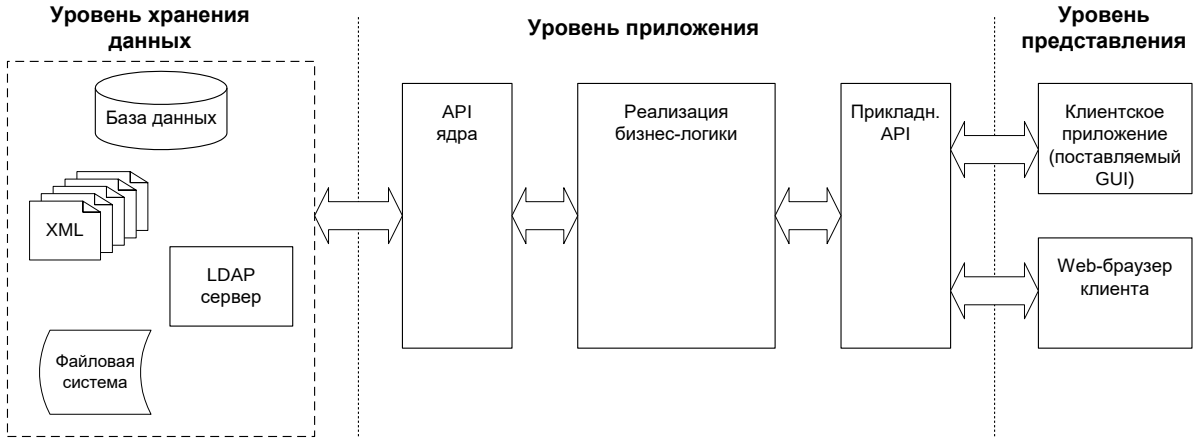


Рисунок 2 – Общая архитектура Системы

На уровне ядра реализованы:

– объектный слой – реализация основных операций с объектами Системы;

– слой метаинформации – реализация работы с метаинформацией о структуре хранилища данных и предоставляющая доступ к физическим данным через унифицированную схему метаданных;

– подсистема безопасности, реализующая собственный механизм безопасности Системы, а также предоставляющий интерфейс для работы с внешними провайдерами безопасности;

– подсистема хранилища данных, реализующая команды языков определения данных (DDL) и манипуляции данными (DML);

– подсистема управления потоками работ.

Система SiTex является объектно-ориентированной средой:

– основной сущностью в системе является объект;

– все сущности Системы, в том числе класс, атрибут и метод являются зарегистрированными объектами Системы;

– папка (также объект Системы) реализует объект-контейнер.

1.1.3 Принципы реализации основных подсистем

Концептуальная модель сервера Системы состоит из:

– ядра, реализующего основную функциональность, связанную с загрузкой и управлением объектов на основе метаинформации о предметной области;

– сервисов и подсистем, обеспечивающих дополнительную функциональность.

Иерархия классов, описывающие основные сервисы и подсистемы платформы, изображена на рисунке 3.

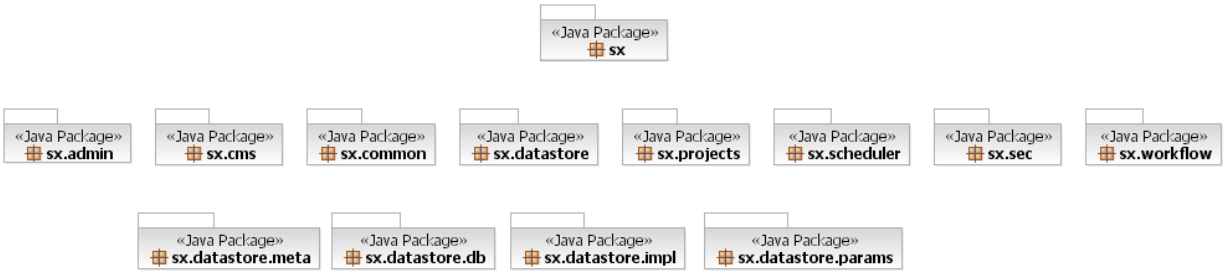


Рисунок 3 – Иерархия классов Системы

Основными пакетами ядра системы являются:

– meta – предназначен для хранения и управления метаинформации;

– Object – хранение и управление системными объектами;

– mapper – управление разнотипными обработчиками при отображении объектных моделей в хранилище;

– datastore – организует работу с разнотипными хранилищами;

– pool – подсистема кэширования серверных объектов;

– security – реализует сервис безопасности;

– transaction – обеспечивает выполнение объектных транзакций;

– replication – подсистема репликации;

– report – подсистема создания отчетов;

– scheduler – планировщик задач;

– externaldata – организация работы с внешними хранилищами;

– tools – дополнительные сервисы;

– workflow – подсистема управления потоками работ;

– search – индексирование и полнотекстовый поиск;

– messaging – сервис передачи системных сообщений, очереди;

– iterators – итераторы и нумераторы;

– cryptography – криптография;

– logging – подсистема логгирования системных событий;

1.1.4 Метамодель системы

Метамодель Системы позволяет описывать объекты пользователя и их поведение в рамках унифицированной модели, не заботясь о способе отображения данных в хранилище.

Метамодель представляет собой набор классов. Класс описывает множество объектов со сходной структурой, поведением и связями с другими объектами. У класса есть атрибуты и методы.

Класс описывает однотипные объекты предметной области. Классы Системы можно условно разделить на системные и прикладные. Первые описывают системные объекты и сущности, вторые предназначены для реализации дополнительной функциональности, специфичной для некоторой прикладной области. Все классы Системы, описывающие объект, наследуются от класса SXObj (исключением являются виртуальные классы).

Любой класс может иметь свою программную реализацию в java и содержать помимо данных и методов базового класса дополнительные атрибуты и методы. Кроме того, методы базового класса могут быть перегружены по правилам языка java.

Для создания нового класса нет необходимости программировать его на языке программирования, достаточно зарегистрировать его в системе. Класс предназначен для описания объектов, хранящих информацию о классах системы.

Каждому из классов, зарегистрированных в системе должна соответствовать таблица, в которой будут храниться значения атрибутов объектов данного класса. Отображение атрибутов в базу данных происходит в соответствии с их типами. Метаописание пользовательского типа данных (строка, логический тип, дата и т.д.) указывает соответствие типам данных в языке программирования и БД.

Атрибут класса описывает то или иное свойство объекта предметной области. К любому атрибуту объекта Системы доступ может быть осуществлен только через методы «getAttr» и «setAttr».

Метамодель реализуется пакетом «sx.datastore.meta». На работу этого пакета опираются классы пакета «sx.datastore», определяющего методы работы с хранилищами данных. К метаинформации относятся классы, атрибуты и отношения между классами (ассоциации и агрегации). Вся работа с объектами метамодели этих классов ведется так же, как и с любыми другими объектами Системы.

Фактически иерархия классов Системы является набором связанных объектов класса «класс». При этом дерево наследования классов в хранилище данных Системы и дерево наследования классов в программном коде могут иметь множество отличий.

Иерархия наследования классов в Системе организуется с помощью связи классов по атрибуту «Parent», являющемуся ссылкой на родительский класс.

1.1.5 Подсистема хранения данных

Подсистема хранения данных определяет основные способы постоянного хранения объектов в системе SiTex. Данный подход позволяет серверу хранить свои объекты в разных источниках данных, не заботясь о том, как они хранятся и как они расположены физически, что облегчает программирование прикладной части системы. Так же, в системе SiTex реализован пул хранилищ.

1.2 Распределенные базы данных

Распределенная система управления базами данных (РСУБД) это программное обеспечение, которое управляет распределенными базами данных, и обеспечивает механизм доступа, который делает это распределение прозрачным для пользователя. Система распределенных баз данных является интеграцией распределенных баз данных и системы управления распределенными базами данных. Эта интеграция достигается путем слияния базы данных и сетевых технологий вместе. Или это может быть описано как система, которая работает на кластере машин (серверов), которые находятся в одной сети, и выглядит для пользователя как одна единая база данных.

Распределенная база данных представляет собой набор нескольких логически взаимосвязанных баз данных, распространяемых по компьютерной сети.

Распределенные системы управления базами данных похожи на распределенные файловые системы в том, что оба облегчают доступ к распределенным данным. Тем не менее, существуют важные различия в структуре и функциональности, которые характеризуют систему распределенных баз данных:

– распределенные файловые системы просто позволяют пользователям получать доступ к файлам, которые находятся на компьютерах, отличных от их собственных. Эти файлы не имеют явной структуры, а отношения между данными в разных файлах не управляются системой и являются ответственностью пользователей. Распределенные базы данных, с другой стороны, организованы в соответствии со схемой, которая определяет как структуру распределенных данных, так и отношения между данными. Схема определяется в соответствии с некоторой моделью данных, которая обычно является реляционной или объектно-ориентированной;

– распределенная файловая система предоставляет пользователям простой интерфейс, который позволяет им открывать, читать, записывать, перемещать и удалять файлы. Распределенная система управления базами данных обладает всеми функциональными возможностями обычной системы управления базами данных. Она обеспечивает высокоуровневые декларативные возможности запросов, управление транзакциями, управление параллельным доступом, восстановление и обеспечение целостности. В этом отношении распределенные системы управления базами данных также отличаются от систем обработки транзакций, поскольку последние предоставляют только некоторые из этих функций [1,2];

– распределенная система управления базами данных обеспечивает прозрачный доступ к данным, в то время как в распределенной файловой системе пользователь должен знать (до некоторой степени) местоположение данных. Также распределённая база данных может быть разделена (фрагментирована) и реплицирована в дополнение к распределению по нескольким узлам (серверам).

В этом смысле технология распределенных баз данных расширяет концепцию независимости данных, которая является центральным понятием в управлении базами данных, на среды, где данные распространяются и реплицируются на нескольких машинах, соединенных сетью. Таким образом, с точки зрения пользователя, распределенная база данных – это логически единая база данных, даже если она физически распределена, то есть выполняется условие прозрачности, что в некоторых случаях считается фундаментальным принципом для построения распределённых баз данных [3].

1.2.1 Типы распределенных баз данных

Системы распределенных баз данных делят на два типа – это гомогенные и гетерогенные распределенные базы данных [5].

Гомогенная распределенная система баз данных – это сеть из двух или более баз данных с одной и той же системой управления распределенными базами данных, которые могут храниться на одной или нескольких машинах.

Таким образом, в этой системе данные могут быть доступны и изменены одновременно на нескольких базах данных в сети без какой-либо обработки и трансляции запросов. Также гомогенные распределенные системы баз данных просты в обращении, обслуживании и проектировании. Пример схемы гомогенной системы распределенных баз данных представлен на рисунке 4.

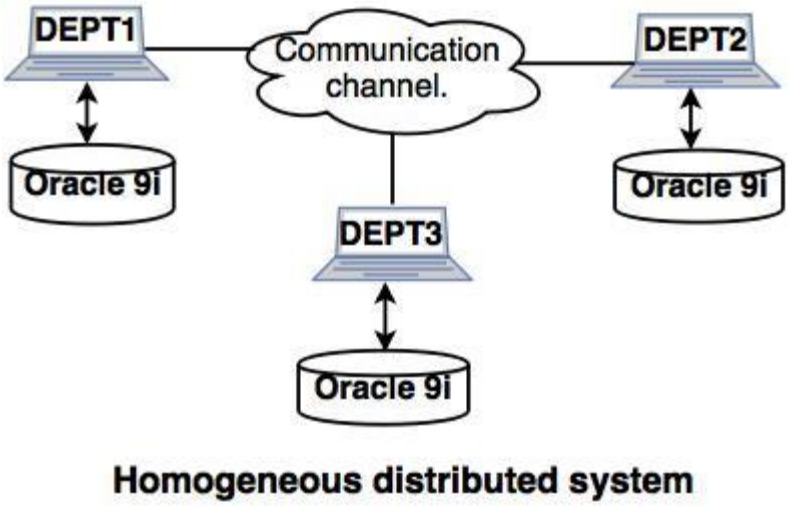


Рисунок 4 – Гомогенная распределенная база данных

Гетерогенная распределенная система баз данных представляет собой сеть из двух или более баз данных с различными типами систем управления базами данных, которые могут храниться на одной или нескольких машинах [4].

В этой системе данные могут быть доступны нескольким базам данных в сети с помощью универсального подключения (ODBC и JDBC), либо в данных системах при обращении к базам данных, имеющих другой тип системы управления базой данных, используется программное обсечение для трансляции запросов, а также для обработки и корректировки данных ответа, так как структура и схема баз данных может значительно различаться. Пример схемы гетерогенной системы распределенных баз данных представлена на рисунке 5.

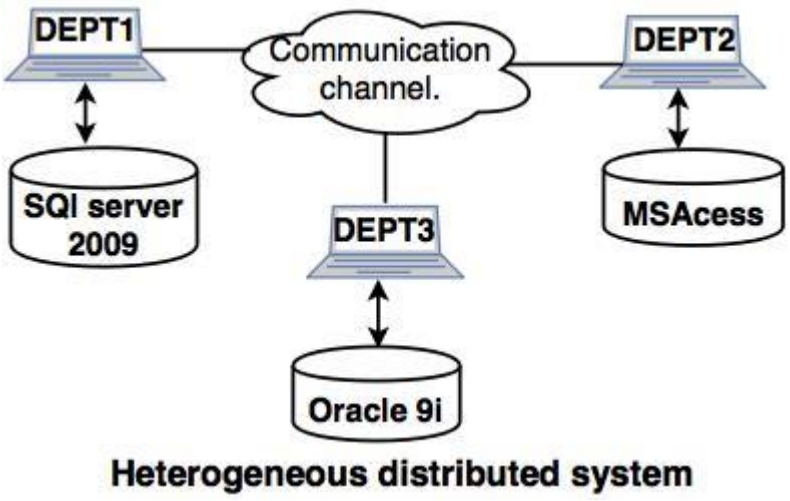


Рисунок 5 – Гетерогенная распределенная база данных

1.2.2 Основные архитектурные типы систем с распределенными базами данных

Основной, самой простой и популярной архитектурой для систем с распределенными базами данных это клиент – серверная архитектура. Данная архитектура имеет несколько клиентов и несколько серверов, соединенных в единую сеть. Клиент отправляет запрос на один из серверов, который выбирается и критериев доступности и загруженности [6]. Данный сервер обрабатывает запрос клиента и отдает результат. Данная архитектура проста в проектировании, реализации и масштабировании благодаря централизованной серверной системе. Топология клиент-серверной архитектуры представлена на рисунке 6.

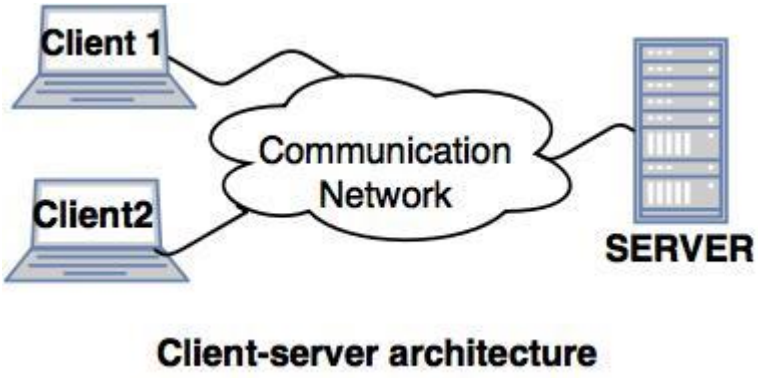


Рисунок 6 – Клиент-серверная архитектура систем с распределенной базой данных

Другой вариант архитектуры систем с распределенными базами данных менее популярен и прост, так как используется в системах, где имеется очень большое количество данных (big data), а также выполняются достаточное долгие и ресурсоемкие запросы, которые манипулируют значительной частью этих данных [7]. Примерами данных систем могут выступать различные информационные системы для сбора и анализа статистических данных за долгий промежуток времени, с последующим анализом и составлением отчетности. Данная архитектура называется «Коллаборацией серверов». Эта архитектура предназначена для систем с распределенными базами данных, чтобы выполнять запрос одновременно на нескольких узлах [8]. Центральный сервер, который обычно отвечает за обработку запросов клиента и отправку результата выполнения запроса, разбивает запрос на отдельные подзапросы и выполняет каждый подзапрос на отдельном узле системы [10]. После выполнения всех подзапросов, сервер возвращает полученный результат клиенту.

Данная архитектура достаточно сложна в реализации, так как требует построения сложной системы корректного разделения запроса на подзапросы.

Топология данной архитектуры представлена на рисунке 7.

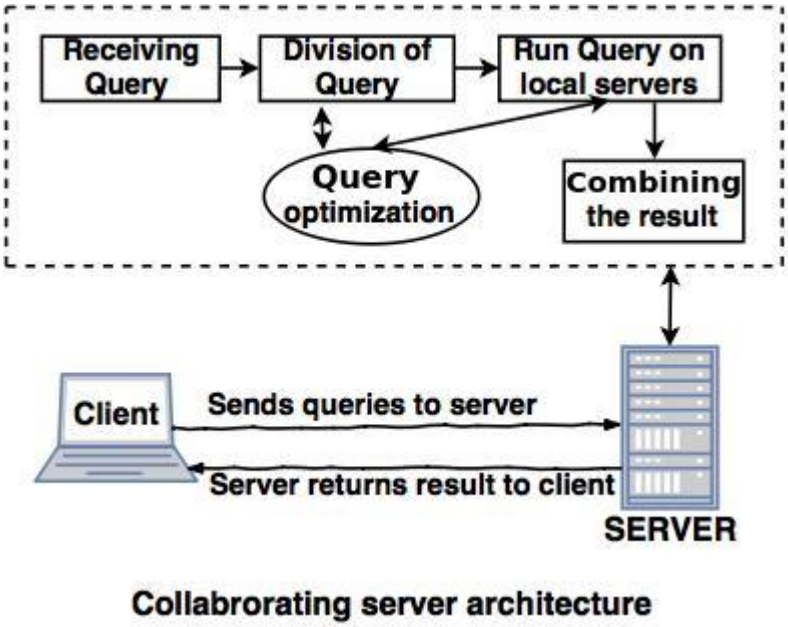


Рисунок 7 – Архитектура «Коллаборация серверов» систем с распределенной базой данных

1.3 Метаданные

Метаданные – это данные о данных. Другими словами, это информация, которая используется для описания данных, содержащихся в различных документах или объектах: веб-страницы, документы или файлы. Другой определение метаданных – это краткое объяснение или резюме того, что представляют собой данные.

Простой пример, метаданные документа могут включать набор таких сведений, как автор, размер файла, дата создания документа и ключевые слова для описания документа. Метаданные музыкального файла могут включать имя исполнителя, альбом и год его выпуска [11, 12].

Для компьютерных файлов метаданные могут храниться в самом файле или в другом месте, как в случае с некоторыми файлами книг «EPUB», которые хранят метаданные в связанном файле «ANNOT».

Метаданные представляют собой закулисную информацию, которая используется в каждой отрасли, по-разному. Она широко распространена в информационных системах, социальных сетях, веб-сайтах, программном обеспечении, музыкальных сервисах и онлайн-торговле. Метаданные могут быть созданы вручную, чтобы выбрать то, что включено, но они также могут быть созданы автоматически на основе данных.

1.3.1 Типы метаданных

Метаданные бывают нескольких типов и используются для различных целей, которые можно приблизительно классифицировать как деловые, технические или операционные.

Например, описательные свойства метаданных включают название, тему, жанр, автора и дату создания.

Метаданные прав могут включать статус авторских прав, правообладателя или условия лицензии.

Свойства технических метаданных включают типы файлов, размер, дату и время создания, а также тип сжатия. Технические метаданные часто используются для управления цифровыми объектами и обеспечения их совместимости.

Сохранение метаданных используется в навигации. Примеры свойств метаданных сохранения включают место элемента в иерархии или последовательности.

Языки разметки включают метаданные, используемые для навигации и взаимодействия. Свойства могут включать заголовок, имя, дату, список или абзац [13].

1.3.2 Метаданные и базы данных

Метаданные в мире управления базами данных могут быть связаны с размером и форматированием или другими характеристиками элемента данных. Это очень важно для интерпретации содержания данных базы данных. Расширяемый язык разметки (XML) — это один из языков разметки, который определяет объекты данных с помощью формата метаданных.

Например, если есть набор данных с датами и именами, разбросанными повсюду, нельзя определенно знать, что представляют собой данные или что описывают столбцы и строки. С помощью базовых метаданных, таких как имена столбцов, можно быстро просмотреть базу данных и понять, что описывает конкретный набор данных [14].

2 Техническое задание

2.1 Список сокращений и определений

Список всех сокращений и определений приведен в таблице1.

Таблица 1 – Список всех сокращений и определений

МСЗН РМ	Министерство социальной защиты, труда и занятости населения Республики Мордовия и подведомственные учреждения
ГКУ СЗН	Государственное казенное учреждение социальной защиты населения районного уровня
АИС ЭСРН РМ	Автоматизированная информационная система «Электронный социальный регистр населения Республики Мордовия» Министерства социальной защиты, труда и занятости населения Республики Мордовия
ЛД	Личное дело получателя государственных услуг и социальных услуг АИС ЭСРН РМ
МСП	Мера социальной поддержки
ЛК	Льготная категория
GUID	Статический уникальный 128-битный идентификатор. Его главная особенность — уникальность, которая позволяет создавать расширяемые сервисы и приложения без опасения конфликтов, вызванных совпадением идентификаторов.

2.2 Общие сведения

2.2.1 Наименование услуг

Объединение сегментов автоматизированной информационной системы «Электронный социальный регистр населения Республики Мордовия» Инсарского и Кадошкинского районов Республики Мордовия.

2.2.2 Краткое наименование услуг

Объединение сегментов АИС ЭСРН РМ Инсарского и Кадошкинского районов Республики Мордовия.

Краткое наименование – «Объединение».

2.2.3 Плановые работы в рамках проекта

Оказание услуг по объединению сегментов автоматизированной информационной системы «Электронный социальный регистр населения Республики Мордовия» Инсарского и Кадошкинского районов Республики Мордовия.

2.2.4 Основания для проведения услуг

Выполнение работ по объединению сегментов автоматизированной информационной системы «Электронный социальный регистр населения Республики Мордовия» Инсарского и Кадошкинского районов Республики Мордовия проводится в рамках распоряжения Правительства Республики Мордовия от 21.12.2015 г. № 984-Р.

2.2.5 Заказчик

Государственное казенное учреждение «Социальная защита населения по Инсарскому району Республики Мордовия» (межрайонная).

2.2.6 Исполнитель

Общество с ограниченной ответственностью «Электронные и программные системы».

2.2.7 Плановые сроки начала и окончания услуг

Срок оказания услуг: 3 месяца с момента подписания контракта за исключением Гарантийного сопровождения.

2.2.8 Источники и порядок финансирования

Источник и порядок финансирования определяется условиями контракта, заключённого между Заказчиком и Исполнителем.

2.2.9 Перечень нормативных документов, на основании которых ведутся работы

Распоряжение Правительства Республики Мордовия от 21.12.2015 г. № 984-Р.

2.3 Цели и задачи оказываемых услуг

Целями данных услуг является:

– повышение эффективности социальной поддержки населения Республики Мордовия, осуществляемой в рамках социальных программ, оказываемой посредством АИС ЭСРН РМ;

– улучшение качества оказания государственных и социальных услуг жителям Республики Мордовия, повышение эффективности и качества социального обслуживания населения;

– уменьшение расходов на обслуживание материально-технической базы Государственного казенного учреждения «Социальная защита населения по Инсарскому району Республики Мордовия» (межрайонная) в части серверного оборудования АИС ЭСРН РМ.

Реализация поставленных целей достигается одновременно с решением следующих задач:

– синхронизация информационных сред сегментов АИС ЭСРН РМ Инсарского и Кадошкинского районов (справочники и классификаторы);

– объединение баз данных АИС ЭСРН РМ Инсарского и Кадошкинского сегментов;

– объединение серверов приложений, реализованных на инструментальной среде разработки Sitex-ЭСРН;

– настройка центрального сегмента АИС ЭСРН РМ для взаимодействия с объединенными сегментами.

2.4 Характеристика объектов автоматизации

2.4.1 АИС ЭСРН РМ

АИС ЭСРН РМ – территориально-распределенная информационная система, обеспечивающая функционирование системы социальной защиты населения Республики Мордовия, обеспечивает доступ к документам, управление бизнес – процессами, получение адекватной информации для поддержки принятия решений. Организовывает пользователям таких систем «прозрачный» доступ к информационным ресурсам вне зависимости от их физического расположения.

АИС ЭСРН РМ построена на платформе SiTex. SiTex – это объектно-ориентированное средство, предназначенное для пользователей, поддерживающих и разрабатывающих трехуровневые приложения. Средства платформы объединяют объектно-ориентированную и реляционную концепции, позволяя описывать бизнес-объекты (понятия предметной области, например, «личное дело клиента», «справка») и правила их взаимодействия на языке мета описания платформы.

Размещении информации АИС ЭСРН РМ осуществляется в распределенном хранилище данных, в качестве которого выступают реляционные базы данных Microsoft Sql Server.

Поддержка распределенных транзакций предоставляет возможность функционирования единой системы МСЗН РМ и территориально-удаленных подведомственных учреждений.

Встроенные средства импорта/экспорта данных из внешних источников, а также инструменты работы с данными из внешних источников, позволяют интегрировать внешние информационные системы и осуществлять информационный обмен в различных форматах.

2.4.2 Инструментальная платформа Sitex

SiTex является платформой для создания корпоративных распределенных систем и поддерживает:

– реализацию хранилища данных и бизнес – логики работы приложения, созданного на основе Системы;

– решение широкого круга задач, описание их в терминах SiTex;

– расширение системы, как с точки зрения выполняемых задач, так и используемых технологий;

– встроенную подсистему безопасности;

– механизмы управления потоками работ.

Система реализует трехзвенную архитектуру (слой хранения данных, слой бизнес – логики и презентационный слой), позволяющую оптимально распределить нагрузку на составляющие распределенной системы и обеспечить:

– прозрачный доступ ко всем ресурсам приложения, построенного на базе SiTex;

– масштабируемость приложения, построенного на базе SiTex;

– поддержку распределенной структуры приложения, построенного на базе SiTex.

Особенностью SiTex является реализация методики расширения функциональности:

– описание новых классов (например, классов новых документов) объектов бизнес – процессов и связей между ними в терминах метамодели SiTex, которая

предоставляет набор базовых операций над объектами этих классов (создание, редактирование свойств, удаление, копирование, перемещение, определение набора прав доступа и т.п.), без программирования;

– определение операций по работе с созданными классами объектов. В системе реализован набор базовых операций (шаблонов) над объектами Системы. Разработчику приложения на базе SiTex достаточно их специфицировать, задав параметры.

В случае необходимости – написание кода бизнес операций на Java с использованием API SiTex. В Системе реализованы базовые механизмы управления потоками работ (бизнес-процессами), позволяющие свести прикладное программирование к минимуму, требующее от прикладного программиста написание только кода атомарной операции работы.

Любое приложение, построенное на базе SiTex, состоит из трех частей: клиента, сервера (ядра) и систем постоянного хранения. В качестве системы постоянного хранения могут выступать СУБД, LDAP, файловая система, XML хранилище и т.д. Клиентом системы может являться как стандартное GUI или WEB приложение (стандартный клиент) так и другая система построенная по принципу обмена документами.

Общая архитектура системы на платформе SiTex представлена на рисунке 8.

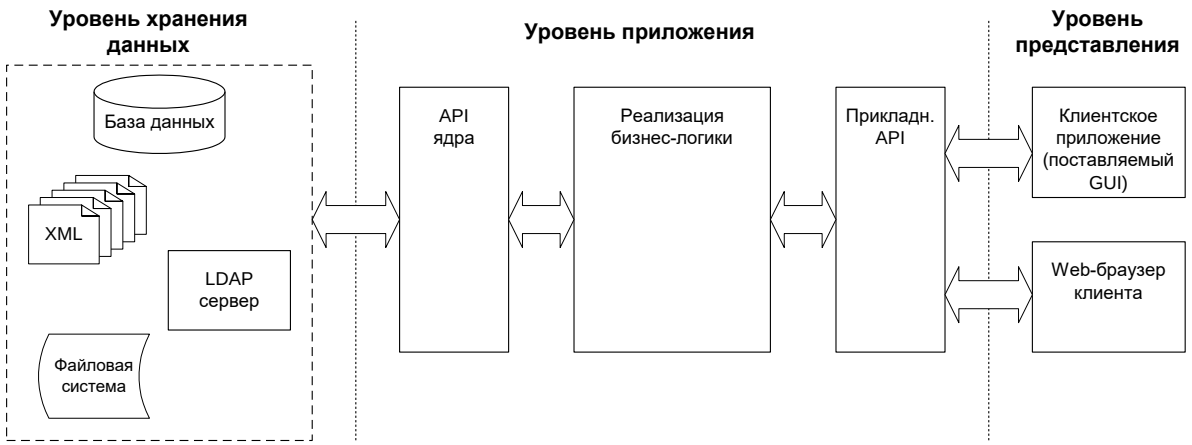


Рисунок 8 – Общая архитектура системы

Ядро SiTex обрабатывает запросы клиента и возвращает результат в виде структурированного сообщения. Системы постоянного хранения используются в качестве долговременного хранилища объектов.

На уровне ядра реализованы:

– объектный слой, реализация основных операций с объектами Системы;

– слой метаинформации, реализация работы с метаинформацией о структуре хранилища данных и предоставляющая доступ к физическим данным через унифицированную схему метаданных;

– подсистема безопасности, реализующая собственный механизм безопасности Системы, а также предоставляющая интерфейс для работы с внешними провайдерами безопасности;

– подсистема репликаций, реализующая обмен данными по принципу «Публикатор - Подписчик»;

– подсистема хранилища данных, реализующая команды языков определения данных (DDL) и манипуляции данными (DML);

– подсистема управления потоками работ**,** предоставляющая средства автоматизации бизнес-процессов организации;

– подсистема отчетов, реализующая настройку и получение отчетов.

SiTex является объектно-ориентированной средой:

– основной сущностью в системе является объект;

– все сущности Системы, в том числе класс, атрибут и метод являются зарегистрированными объектами Системы;

– папка (также объект Системы) реализует объект-контейнер.

Метамодель SiTex позволяет описывать объекты пользователя и их поведение в рамках унифицированной модели, не заботясь о способе отображения данных в хранилище. Метамодель представляет собой набор классов. Класс описывает множество объектов со сходной структурой, поведением и связями с другими объектами. У класса есть атрибуты и методы.

Для создания нового класса нет необходимости программировать его на языке программирования, достаточно зарегистрировать его в Системе. Класс предназначен для описания объектов, хранящих информацию о классах SiTex.

Каждому из классов, зарегистрированных в Системе должна соответствовать таблица, в которой будут храниться значения атрибутов объектов данного класса. Отображение атрибутов в базу данных происходит в соответствии с их типами. Метаописание пользовательского типа данных (строка, логический тип, дата и т.д.) указывает соответствие типам данных в языке программирования и БД.

2.5 Требования к выполняемым работам

2.5.1 Требования к синхронизации информационных сред

Заказчик должен провести выверку и при необходимости синхронизацию справочников и классификаторов, используемых в сегментах АИС ЭСРН РМ Инсарского и Кадошкинского районов.

Подсистема нормативной справочной информации в АИС ЭСРН РМ 2-х районов должна быть проанализирована. Справочники должны быть сопоставлены между собой. Должны быть сформированы протоколы разногласий. Разногласия должны быть устранены.

2.5.2 Требования к объединению баз данных

Структура БД 2-х районов должна быть проанализирована. Должна быть разработана технология объединения 2-х БД.

Состав данных личного дела должен быть проанализирован. Должны быть сформулированы предложения по идентификации личных дел, составу данных личных дел к объединению и архивации.

Должна быть определена главная и побочная БД. Должна быть разработана задача слияния (конвертации) сведений побочной БД и главной БД. По итогам слияния (конвертации) должны быть сформированы протоколы разногласий, включающие в себя детализацию состава данных личных дел. Разногласия должны быть устранены.

Подсистема внешнего и внутреннего информационного обмена должна быть проанализирована и доработана с учетом объединения 2-х БД. Перечень информационных обменов приведен в приложении 2.

Задача межведомственных электронных запросов (ответов) посредством системы межведомственного электронного взаимодействия должна быть проанализирована. При необходимости данная задача должна быть доработана с учетом объединения 2-х БД.

Задача формирования регистра получателей социальных услуг должна быть проанализирована. При необходимости данная задача должна быть доработана с учетом объединения 2-х БД.

Функциональность центральной площадки АИС ЭСРН РМ (http://10.0.0.1:8080) должна быть проанализирована. При необходимости данная функциональность должна быть доработана с учетом объединения 2-х БД.

При объединении 2-х БД АИС ЭСРН РМ не должна быть нарушена ее работоспособность.

2.5.3 Требования к объединению таблиц

Перед объединением основных таблиц должны быть синхронизированы справочники и классификаторы.

При объединении основных таблиц необходимо произвести переиндексацию первичных ключей побочной БД.

Далее необходимо проанализировать метаописание Sitex для выявления внешних ключей для данного первичного.

Далее необходимо переиндексировать таблицы, содержащие внешние ключи.

После переиндексации всех ключевых полей должно производиться физическое слияние таблиц.

После слияния таблиц необходимо провести анализ на наличие повторяющихся глобальных идентификаторов GUID. При наличии таких объектов необходимо провести замену GUID.

2.5.4 Требования к объединению отчетной подсистемы

Подсистема отчетности должна быть проанализирована. Необходимые отчеты должны быть доработаны в части их формирования в целом по Государственному учреждению (итоговые данные) и включению в себя детализации сведений по муниципальным образованиям.

Формирование и сверка соответствующих отчетов проводятся Заказчиком с формированием протокола разногласий. Исполнитель должен предпринять меры (изменение кода и/или формы отчета) для устранения соответствующих разногласий.

Заказчик перед началом процедур по объединению сегментов формирует всю необходимую статистическую отчетность по сегментам. После объединения Заказчик формирует статистическую отчетность с объединенного сегмента и составляет протокол разногласий. Исполнитель предпринимает меры по устранению данных разногласий.

2.5.5 Требования к объединению подсистемы безопасности

Исполнитель должен обеспечить перенос пользователей побочной БД в основную с сохранением матрицы прав доступа к объектам АИС ЭСРН РМ. Слияние таблиц с пользователями должно производиться согласно требованиям к объединению таблиц. Далее дублирующийся объекты должны быть удалены на логическом уровне (чтобы не потерять информацию аудита).

2.5.6 Требования к функционированию БД «register»

БД «register» является реплицируемой БД с центральным сегментом АИС ЭСРН РМ. Она используется для инфообмена МТСЗН РМ с внешними контрагентами (ПФР). Необходимо модернизировать задачи импорта сведений в БД «register» и задачи заполнения сведений на объединяемых районах.

2.6 Требования к видам обеспечения

2.6.1 Требования к информационному обеспечению

При синхронизации нормативно-справочной информации сегментов АИС ЭСРН необходимо предусмотреть использование справочников, классификаторов:

1) Принятых к использованию на территории Республики Мордовия;

2) Ведомственных справочников и классификаторов АИС ЭСРН.

Создаваемое программное обеспечение должно обеспечивать объединение следующих видов данных:

1) Прикладные данные (данные, вводимые пользователями);

2) Системные данные, в частности настройки, используемые системой.

Должны быть предусмотрены средства резервного копирования и восстановления данных после сбоя. Все базы должны быть скопированы на резервные носители перед объединением.

Доступ к данным должен быть предоставлен только авторизованным пользователям с учетом их полномочий.

2.6.2 Требования к программному обеспечению

Для работы сегментов АИС ЭСРН РМ, обеспечивающей реализацию функциональных возможностей, приведенных в данном документе, необходимо использование программных средств, удовлетворяющих требованиям, представленным в таблице 2.

Таблица 2 – Программные средства

№ п/п	Категория ПО	Требование
1	Сервер АИС ЭСРН (центральный сегмент)	Операционная система Microsoft Windows Server 2003 R2 Standard. СУБД Microsoft SQL Server 2005 версии Standard или выше. Установка программного обеспечения Sitex «ЭСРН» версии 1.4 CriptoPro JSP v1.0.53(и выше) jdk7_21, Tomcat7

Окончание таблицы 2

2	Сервер АИС ЭСРН (районный сегмент)	Операционная система Microsoft Windows Server 2003 R2 Standard, Microsoft Windows 7 Proffesional. СУБД Microsoft SQL Server 2005 версии Standard или выше. Установка программного обеспечения Sitex «ЭСРН» версии 1.4 jdk7_21, Tomcat7

2.6.3 Требования к техническому обеспечению

Приемка услуг осуществляется приемочной комиссией. Состав приемочной комиссии, место и время проведения испытаний определяется Заказчиком. В состав комиссии по проведению испытаний должны входить представители Заказчика и Исполнителя. Заказчик может привлечь для приемки оказанных услуг сторонние организации.

Комиссии по приемке предъявляются результаты всех оказанных работ. В рамках проводимой экспертизы документов комиссией оцениваются:

– полнота;

– качество;

– непротиворечивость;

– стилистическая однородность.

В случае выявления недостатков и замечаний к документации, Заказчик вправе потребовать от Исполнителя их полного устранения в согласованные сторонами сроки. Сторонами оформляется протокол замечаний, где описываются все обнаруженные недостатки, замечания к документации, устанавливаются сроки их устранения.

Услуги считаются оказанными, в случае устранения всех замечаний и подписания сторонами акта оказанных услуг по контракту.

2.6.4 Проведение предварительных испытаний

На этапе проведения предварительных испытаний объединенной базы данных Кадошкинского и Инсарского района создается отдельная база данных и контекст веб-сервера.

Проводятся предварительные испытания в соответствии с перечнем статистических отчетов и задач АИС ЭСРН РМ.

2.6.5 Опытная эксплуатация

В ходе проведения опытной эксплуатации специалисты ГКУ проводят тестирование отдельного контекста объединенной БД.

В процессе, а также по итогам опытной эксплуатации осуществляется исправление технических ошибок и доработка интеграционных решений.

В рамках данного этапа необходимо добиться устойчивого функционирования интеграции АИС ЭСРН РМ удовлетворяющего всем требованиям Технического задания.

2.6.6 Ввод в промышленную эксплуатацию

После исправления всех замечаний Заказчика проводится объединение баз данных с последующим вводом в промышленную эксплуатацию.

Результатом успешного выполнения всех предшествующих этапов работ является подписание Акта сдачи-приемки всех выполненных работ по контракту.

3 Разработка системы автоматизированного объединения распределённых баз данных

Так, как данная система в дальнейшем будет использоваться для объединения распределенных баз данных АИС ЭСРН РМ, то можно выделить некоторые основные положения:

– все базы данных данной автоматизированной информационной системы являются одинаковыми по схеме, что упрощает и унифицирует метод определения дерева взаимосвязей;

– исходя из предметной области, можно сделать вывод что основной и центральной сущностью является человек, которому назначаются меры социальной поддержки, или проводиться другие действия в отношении его. Следовательно, что все остальные сущности (объекты) должны быть зависимы от данной сущности, и иметь с ней взаимосвязь. Это дает нам то, что мы можем отталкиваться от данной сущности, как от некоторой стартовой точки дерева, рекурсивно находя дочерние узлы;

– в данной автоматизированной информационной системе, в системных таблицах храниться вся метаинформация о всех функциональных сущностях, которые используются в системе. Это упрощает методы определения функциональных сущностей, так как можно опираться на данную метаинформацию, а также определять тип взаимосвязи и количество дочерних узлов;

– также для дальнейшего эффективного и удобного использования системы объединения распределенных баз данных в АИС ЭСРН РМ на других районах, необходимо реализовать графический интерфейс пользователя на основе веб-приложения, которое будет находиться в защищенной сети АИС ЭСРН РМ для доступа ко всем базам данных.

Также, проанализировав предметную область, можно выявить следующие функциональные группы сущностей, содержащиеся в автоматизированной информационной системе:

– ключевые сущности отражающие объекты предметной области автоматизированной информационной системы (транзакционные таблицы);

– сущности, содержащие набор структурированной часто используемой информации в определенной узкой сфере предметной области (таблицы справочники);

– сущности, содержащие данные по различным настройкам и параметрам функциональных модулей приложения (таблицы настроек);

– сущности, содержащие данные по различным настройка, параметрам, а также ведущие процесс логирования действий пользователей и системы (системные таблицы).

Исходя из анализа предметной области, справочники так же можно разделить на две группы:

– централизованные;

– децентрализованные.

Такое разделение обусловлено тем, что в данной предметной области есть справочники, которые должны быть одинаковыми во всех районах (централизованные), например, справочник организаций РМ. Такие справочники содержат поле «GUID» (Globally Unique Identifier) которое имеет одинаковое значение определенного объекта во всех других БД АИС ЭСРН РМ. И так же есть справочники, которые ведутся в каждом районе свои (децентрализованные), например, справочник Фамилий. Такие справочники не содержат поля «GUID», а только поле «OUID» (Organization Unique Identifier).

Из этого следует, что необходимо реализовать алгоритм определения и объединения справочных сущностей.

Так как целостность баз данных АИС ЭСРН РМ поддерживается на уровне приложения, то, следовательно, выявить взаимосвязи программным методом

между сущностями на уровне работы с БД не получится. Также не получится однозначно определить к какой функциональной группе относится сущность.

Исходя из этого, необходимо реализовать алгоритм построения дерева взаимосвязей всех ключевых бизнес-сущностей методом рекурсивного поиска по прямым, множественным или обратным ссылкам, а также алгоритм объединения найденных бизнес-сущностей по различным типам связи.

Также, в данных алгоритмах необходимо учитывать то, что сущности могут иметь не одну, а множество взаимосвязей друг с другом. Тем самым, мы должны предусмотреть рекурсивное зацикливание, а также уникальность узлов дерева взаимосвязей, то есть чтобы в дереве взаимосвязей сущность была отображена только один раз.

Также, необходимо обеспечивать безопасность и возможность отката действий системы, в случае непредвиденного результата или неправильной работы системы объединения распределенных баз данных. К этому же, необходимо записывать все действия системы объединения распределенных баз данных для дальнейшего анализа и изучения неправильной работы системы [15, 16].

Данный функционал будут обеспечивать подсистемы логирования действий системы и резервного копирования объединяемых баз данных.

Из выше сказанного следует, что выявленные критерии усложняют рассматриваемую задачу и предполагают реализацию следующих алгоритмов и подсистем:

– алгоритм построения дерева взаимосвязей всех ключевых бизнес-сущностей методом рекурсивного поиска по прямым, множественным или обратным ссылкам и их объединения;

– алгоритм определения и объединения справочных сущностей;

– подсистема логирования операции;

– подсистема резервного копирования.

Также в процесс объединения не будут включаться системные таблицы так как это приведет к неправильному функционированию АИС ЭСРН РМ.

3.1 Реализация алгоритма определения и объединения справочных сущностей

Так как заранее не известно какие таблицы относятся к справочным сущностям, то необходимо сформировать набор критериев для определения типа. Для справочных сущностей можно выделить следующий набор критериев:

– так как рассматриваемая информационная система является централизованно-распределенной, то справочные сущности должны быть идентичны на всех базах данных и без потери ссылочной связанности. То есть, при переходе объекта из одной базы данных в другую, ссылочная связь должна также указывать на тот же элемент справочной сущности, что и в исходной базе данных. Это достигается путем введения глобального уникального идентификатора, который присваивается каждому элементу справочной сущности и идентичен на всех базах данных. Следовательно, один из критериев определения справочного типа – это наличие глобального уникального идентификатора (GUID);

– также в рассматриваемой информационной системе в именование таблицы (сущности) указывается префикс, определяющий принадлежность данной таблицы к определенному типу (например «wm» - обозначает бизнес-сущности, «sx» - означает системные сущности и т.д.). Следовательно, заранее можно исключить таблицы, не принадлежащие к справочному типу, по вхождению группы префиксов в имя таблицы.

Таким образом, теперь мы можем выполнить SQL запрос к базе данных и получить список таблиц, удовлетворяющих нашим критериям. В SQL запросе для получения наименования всех таблиц, нам необходимо обратиться к системной таблиц «sys.objects», которая содержит перечисление и описание всех таблиц в той базе данных, к которой осуществляется запрос [31, 30]. Далее мы вводим условие выборки только тех таблиц, которые содержат поле с наименованием «GUID».

```
declare @tmp table(Table_Name nvarchar(max))
insert into @tmp SELECT tbl.name FROM sys.objects tbl
inner join INFORMATION_SCHEMA.COLUMNS sch
on tbl.name = sch.TABLE_NAME
WHERE tbl.type in (N'U') and sch.COLUMN_NAME like 'GUID' and tbl.Name in (SELECT
tbl.name FROM [result].sys.objects tbl
inner join [result].INFORMATION_SCHEMA.COLUMNS sch
on tbl.name = sch.TABLE_NAME
WHERE tbl.type in (N'U') and sch.COLUMN_NAME like 'GUID')
Declare @tbl_name nvarchar(max)
DECLARE @sqlCommand nvarchar(1000)
Declare @count float
Declare @sourceCount int
Declare @result table (Table_Name nvarchar(max), Percents float)
Declare cur cursor for select * from @tmp
open cur
FETCH NEXT FROM cur
INTO @tbl_name
WHILE @@FETCH_STATUS = 0
BEGIN
SET @sqlCommand = 'select @sourceCount = Count(*) from ' + @tbl_name +
'; if @sourceCount > 0 begin select @cnt = ((100 * (select Count(*) from ' +
@tbl_name+' t1 inner join [result].dbo.'+@tbl_name+' t2 on t1.guid = t2.guid)) /
(@sourceCount)) end Else begin select @cnt = 0 end'
EXECUTE sp_executesql @sqlCommand, N'@sourceCount int,@cnt float
OUTPUT',@sourceCount = @sourceCount, @cnt = @count OUTPUT
INSERT INTO @result select @tbl_name, @count
FETCH NEXT FROM cur
INTO @tbl_name
END
select * from @result where Percents = 100 and (Table_Name not like '%cms%' and
Table_Name not like '%sx%' and Table_Name not like '%temp%
```

Однако, в данный список могут попасть по заданным нами критериям не только справочные сущности, но и другие. Так как поле с глобальным уникальными идентификатором могут содержать и другие таблицы, не соответствующие справочному типу, и также существуют справочные сущности, являющиеся полностью несинхронными (данные заполняется на каждом сегменте по-своему), то следовательно мы должны исключить их списка таблиц для реализации дальнейшего алгоритма объединения. Поэтому для корректировки списка и удаления из него ненужных нам сущностей мы SQL запросом будем сравнивать идентичность данных и глобальных уникальных идентификаторов на базе данных другого сегмента [18].

```
declare @tmp table(Table_Name nvarchar(max))
insert into @tmp SELECT tbl.name FROM sys.objects tbl
inner join INFORMATION_SCHEMA.COLUMNS sch
```

```
	on tbl.name = sch.TABLE_NAME
	WHERE tbl.type in (N'U') and sch.COLUMN_NAME like 'GUID' and tbl.Name in (SELECT
tbl.name FROM [result].sys.objects tbl
	inner join [result].INFORMATION_SCHEMA.COLUMNS sch
	on tbl.name = sch.TABLE_NAME
	WHERE tbl.type in (N'U') and sch.COLUMN_NAME like 'GUID')
	Declare @tbl_name nvarchar(max)
	DECLARE @sqlCommand nvarchar(1000)
	Declare @count float
	Declare @sourceCount int
	Declare @result table (Table_Name nvarchar(max), Percents float)
	Declare cur cursor for select * from @tmp
	open cur
	FETCH NEXT FROM cur
	INTO @tbl_name
	WHILE @@FETCH_STATUS = 0
	BEGIN
	SET @sqlCommand = 'select @sourceCount = Count(*) from ' + @tbl_name
	+ '; if @sourceCount > 0 begin select @cnt = ((100 * (select Count(*) from '
	+ @tbl_name+' t1 inner join [result].dbo.'+@tbl_name+' t2 on t1.guid = t2.guid))
/ (@sourceCount)) end Else begin select @cnt = 0 end'
	EXECUTE    sp_executesql    @sqlCommand,    N'@sourceCount    int,@cnt    float
OUTPUT',@sourceCount = @sourceCount, @cnt = @count OUTPUT
	INSERT INTO @result select @tbl_name, @count
	FETCH NEXT FROM cur
	INTO @tbl_name
	END
	select * from @result where Percents > 90 and (Table_Name not like '%cms%' and
Table_Name not like '%sx%' and Table_Name not like '%temp%')
```

В данном SQL запросе мы также находим список всех таблиц, удовлетворяющих нашим критериям, на базе данных другого сегмента. Далее мы делаем выборку таблиц, удовлетворяющих условиям идентичности данных более чем на 90%. Идентичность данных проверяется по совпадению значения ключевого поля «GUID». Такое процентное соотношение взято из того, что некоторые элементы справочных сущностей могут быть изменены, или добавлены новые уникальные для каждого сегмента. Также, в соответствии с выведенными критериями, и для увеличения производительности и уменьшения времени выполнения запроса мы дополнительно исключаем условием выборки таблицы, название которых содержит префиксы cms, sx и temp, так как данный таблицы используются для хранения совершенно других временных и системных данных, не относящиеся к справочным сущностям. Диаграмма деятельности данного процесса определения справочных сущностей изображен на рисунке 9.

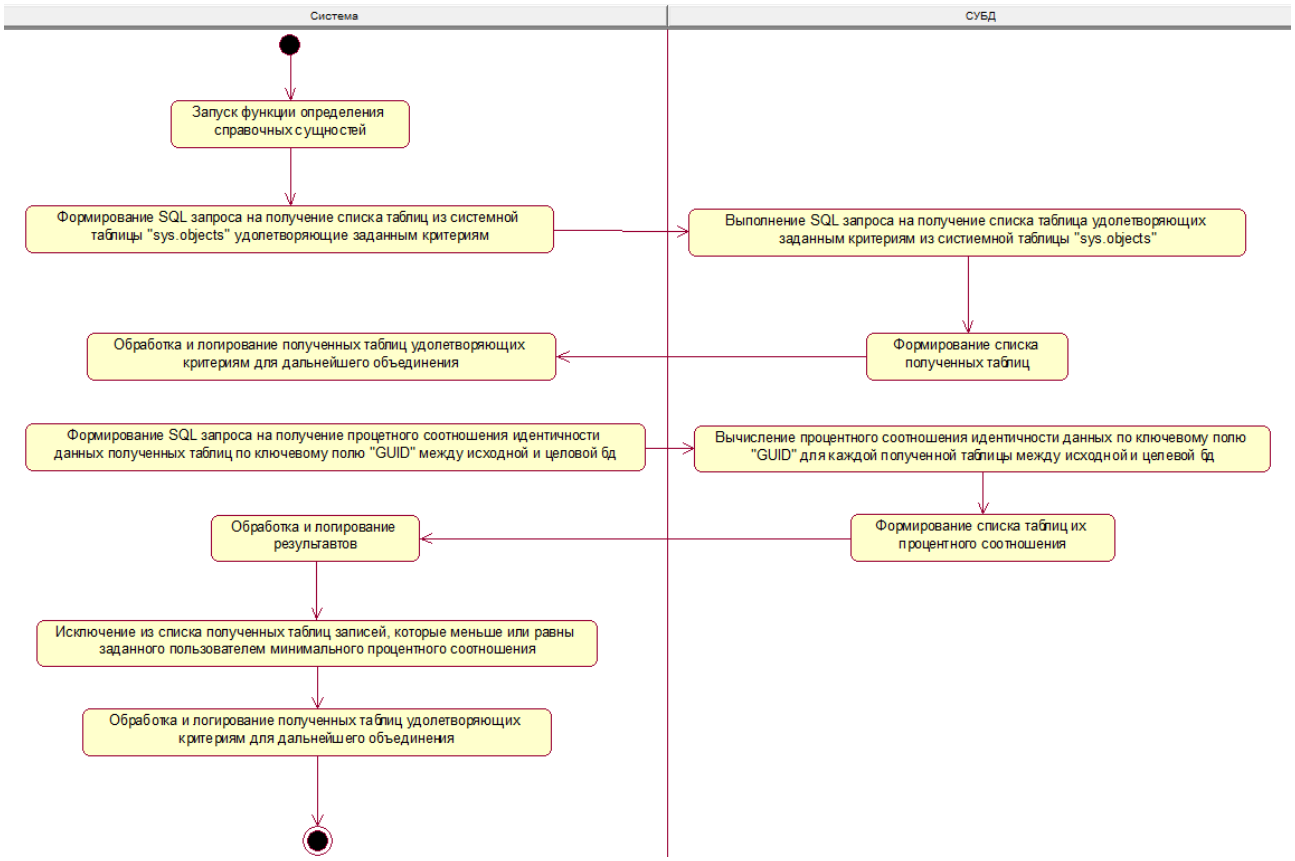


Рисунок 9 – Диаграмма деятельности процесса определения справочных сущностей

После нахождения всех справочных сущностей, мы в цикле проходим по каждой сущности и выполняем следующие действия:

– сначала мы полностью копируем все элементы данной сущности из целевой в результирующую базу данных;

– далее мы находим по ключевому полю «GUID» и копируем те элементы сущности в исходной базе данных, которые не входят в результирующую базу данных.

Таким образом, результирующая база данных имеет объединенные справочные сущности из исходной и целевой баз данных. Для реализации данного алгоритма, в программе был объявлен класс «Spr_Merge», который реализует весь функционал по поиску, определению, сравнению и объединению справочных сущностей [19, 20].

3.2 Реализация алгоритма определения дерева взаимосвязей бизнес-сущностей и их объединения

Исходя из выведенных общих положений, при определении дерева взаимосвязей мы будем отталкиваться от сущности человека, которая представлена таблицей «WM_PERSONAL_CARD». Данная таблица хранит основную информацию на человека, а так множество связей на другие таблицы, которые представляют другие объекты (адреса, документы, справочники фамилий, имен, отчеств и т.д.). Для реализации был объявлен класс «WM_MERGE», который реализует методы по определению дерева взаимосвязей и объединения сущностей по данному дереву. В данном классе определен один публичный метод «Execute», который запускает весь процесс определения и объединения бизнес-сущностей.

В начале данного метода производится проверка на объединенность таблицы «WM_PERSONAL_CARD». Если проверка не проходит успешно, то объединяется таблица «WM_PERSONAL_CARD». Также, у всех объединенных записей первичный ключ увеличивается на порядок больше для того, чтобы в дальнейшем можно было наглядно понять по записям, какие принадлежали исходной базе данных, а какие целевой. За это отвечает функция «GetDegree», объявленная в классе «Esrn_utils». Данная функция принимает два аргумента – число, которое представляет первичный ключ, и порядок увеличения. Результатом данной функции является число, которое увеличено на переданный порядок [29, 21].

Также, данный класс предоставляет набор утилитарных функции, которые могут понадобиться множество раз в процессе разработки.

Далее начинается поиск всех сущностей (таблиц), которые ссылаются на главную таблицу «WM_PERSONAL_CARD», по прямым, множественным и обратным ссылкам. Данный поиск ведется на основе метаинформации, которая так же хранится в базе данных. Найденные взаимосвязанные сущности

добавляются в массив. После, для первично найденных сущностей запускается функция объединения «Merge». Данная функция перебирает массив найденных сущностей, и для каждой найденной сущности, в зависимости от типа связи, применяет определенный алгоритм объединения. Этот алгоритм проверяет признак объединенности исходной сущности или признак объединенности ссылающейся сущности, и в зависимости от типа связи объединяет исходную и/или ссылающеюся сущность. Если признак объединенности одной из сущностей ложный, а для данного типа связи он должен быть истинным, то алгоритм сначала запускает процесс объединения данной сущности. Также, данные сущности проверяются на наличие записей, так как нет смысла объединять пустые таблицы. Все объединенные сущности добавляются в массив «Table_used», для исключения их в последующих процедурах объединения и поисках дочерних узлов дерева взаимосвязей у данной сущности.

После объединения первоначально найденных сущностей, массив найденных сущностей перебирается в цикле, где для каждой сущности применяется функция RecursiveExecute. Данная функция также рекурсивно определяет все взаимосвязи переданной ей сущности по прямым, множественным или обратным ссылкам, и после чего также перебирает их в цикле и рекурсивно вызывает себя для найденных сущностей. Данный метод работает до тех пор, пока не будут найдены все взаимосвязи от каждой переданной сущности. Так же на каждом вызове функции, после нахождения взаимосвязей переданной ей сущности, вызывается функция объединения найденных сущностей «Merge», в которую передается массив найденных сущностей.

Таким образом, мы находим все взаимосвязанные сущности, начиная от основной «WM_PERSONAL_CARD», и рекурсивно строим дерево взаимосвязей в глубину от каждой связанной сущности. На каждой итерации нахождения взаимосвязей мы применяем функцию объединения, для найденных сущностей, и записываем их в переменную «Table_used» для дальнейшего игнорирования. Тем самым мы получаем базу данных с объединёнными сущностями, и дерево

взаимосвязей, по которому можно понять, какие сущности были объединены и по какому алгоритму [22].

3.3 Реализация подсистемы логирования действий системы

Логирование действий системы – это процесс, который играет одну из важнейших ролей в любой системе. Так как невозможно сразу реализовать систему, где учтены все возможные нештатные и критические ситуации, то процесс логирования действий позволяет зафиксировать все данные (порядок действий, данные, место в программном коде, временные и другие параметры системы), которые помогут в дальнейшем понять, при каких обстоятельствах произошла данная критическая ситуация.

Подсистемы логирования действий системы будет реализована на основе одной из популярных библиотек для логирования на .Net Framework – NLog.

NLog – это гибкая и бесплатная библиотека для ведения журнала операций системы для различных платформ .NET, включая .NET Standard. NLog позволяет легко вести процесс логирования в несколько целевых объектов (база данных, файл, консоль) и изменить конфигурацию ведения журнала на лету. NLog имеет поддержку структурированного и традиционного ведения журнала.

В данной системе объединения распределенных баз данных, подсистема логирования настроена на ведение процесса в csv файл, с указание времени, функции, вызвавшей процесс записи в лог файл, уровень логирования и сообщения. Все алгоритмы и их функции, обрабатывающие записи в базах данных всегда вызывают функцию логирования с указанием всех переданных параметров и совершаемых действий. Например, при объединении бизнес-сущностей функцией «Merge» или «RecursiveMerge», в лог файл записывается наименование сущностей, с которыми в данный момент идет процесс

объединения, тип связи между ними, количество записей у каждой сущности и результат процесса.

В итоге, мы можем после каждой работы программы посмотреть лог файл, выявить ошибки и неточности работы системы по объединению распределенных баз данных, и в последующем доработать систему для дальнейшего избегания данных ситуаций.

3.4 Реализация подсистемы резервного копирования баз данных

Так как деятельность органов социальной защиты населения РМ ведется каждый день, и, следовательно, много сотен тысяч данных меняется в базах данных ежедневно, то при очередном процессе объединения распределённых баз данных районов может возникнуть ошибка, ведущая за собой неработоспособность системы АИС ЭСРН РМ и потерю данных.

Для предотвращения данной ситуации, была реализована подсистема резервного копирования данных, которая перед каждым запуском системы по объединению распределенных баз данных, делает полные резервные копии исходной и целевой баз данных. В случае возникновении описанной или критической ситуации, подсистема восстанавливает исходную и целевую базу данных в их первоначальное состояния из сделанных резервных копий.

3.5 Инструменты реализации системы объединения распределенных баз данных

Объединение распределенных баз данных — всегда непростая задача. Она зависит от многих параметров построения системы и предметной области. К

каждой задаче необходим свой подход и свои алгоритмы, так как нет возможности унифицировать задачу в силу множества вариаций проектирования систем.

Прежде чем объединять продуктивные базы данных, необходимо несколько раз протестировать алгоритм на резервных базах, и в тестевом режиме запустить автоматизированную информационную систему на полученной объединенной БД, проверить работоспособность системы, выявить ошибки и внести изменения.

Программная реализация будет на платформе .NET Core, в частности на языке программирования C#. Данная платформа выбрана по причине кроссплатформенности, большей производительности по сравнению с .Net Framework, и отрытого исходного кода, что подразумевает большую поддержку сообщества и отсутствие лицензионных ограничений. Так же эта платформа выбрана, потому что имеется большой опыт разработок на ней, а также работы с БД как посредством исполнения SQL запросов, так и через ORM библиотеки, и для полной интеграции с графическим интерфейсом пользователя на основе веб-приложения.

Так же будут использоваться части SQL хранимых процедур и функций, выполняемых СУБД, так как извлечение обработка большого количества данных на стороне прикладного приложения может сильно увеличить время выполнения задачи, и также потребность в большом количестве оперативной памяти.

В итоге, разработанное приложение будет состоять из двух компонентов. Прикладное приложение на платформе .NET Core будет реализовывать различные бизнес-процессы, связанные с критериями и условиями задачи объединения. Все «CRUD» операции по работе с данными будут выполнять на стороне серверов, под управлением СУБД.

4 Проектирование пользовательского графического интерфейса на основе веб-приложения

В настоящий момент, при решении вопроса разработки графического интерфейса пользователя большое количество разработчиков в качестве основополагающей архитектуры выбирают веб-приложение.

Классическое веб приложение — клиент-серверное приложение, в котором клиент взаимодействует с веб-сервером при помощи браузера. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения являются кроссплатформенными приложениями.

Веб-приложение состоит из клиентской и серверной частей, тем самым реализуя технологию «клиент-сервер». Клиентская часть реализует пользовательский интерфейс, как с точки зрения дизайна, так и бизнес логики поведения и управления самого интерфейса, формирует запросы к серверу и обрабатывает ответы от него. Серверная часть получает запрос от клиента, выполняет вычисления, после этого формирует веб-страницу либо данные ответа и отправляет её клиенту по сети с использованием протокола HTTP. Само веб-приложение может выступать в качестве клиента других служб, например, базы данных или другого веб-приложения, расположенного на другом сервере. Ярким примером веб-приложения является система управления содержимым статей Википедии: множество её участников могут принимать участие в создании сетевой энциклопедии, используя для этого браузеры своих операционных систем (будь то Microsoft Windows, GNU/Linux или любая другая операционная система) и не загружая дополнительных исполняемых модулей для работы с базой данных статей.

В настоящее время набирает популярность новый подход к разработке веб-приложений, называемый Ajax. При использовании Ajax страницы веб-приложения не перезагружаются целиком, а лишь догружают необходимые данные с сервера, что делает их более интерактивными и производительными.

Также в последнее время набирает большую популярность технология WebSocket, которая не требует постоянных запросов от клиента к серверу, а создает двунаправленное соединение, при котором сервер может отправлять данные клиенту без запроса от последнего. Таким образом появляется возможность динамически управлять контентом в режиме реального времени.

Для создания веб-приложений на стороне сервера используются разнообразные технологии и любые языки программирования, способные осуществлять вывод в стандартную консоль.

Также в последнее время клиент-серверные веб приложения принято делить сообществом на frontend (клиент) и backend (сервер) части.

4.1 Инструменты реализации графического интерфейса пользователя на основе веб-приложения

Так как классическое веб-приложение состоит из двух частей, то, следовательно, реализации веб-приложения будет состоять из двух этапов:

– реализация frontend части;

– реализация backend части.

Для реализации frontend части будет использоваться следующий стек языков программирования, технологий и фреймворков:

HTML – стандартизированный язык разметки документов во Всемирной паутине. Большинство веб-страниц содержат описание разметки на языке HTML (или XHTML). Язык HTML интерпретируется браузерами; полученный в

результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

CSS – формальный язык описания внешнего вида документа, написанного с использованием языка разметки. Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML [22].

Less – это динамический метаязык на основе каскадных таблиц стилей (Cascading Style Sheets - CSS), предназначенный для упрощения, масштабирования и поддержки большого объёма CSS кода [23].

JavaScript – мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией языка ECMAScript (стандарт ECMA-262). JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам. JavaScript встраивается непосредственно в исходный текст HTML-документа и интерпретируется браузером по мере загрузки этого документа. С помощью JavaScript можно динамически изменять текст загружаемого HTML-документа и реагировать на события, связанные с действиями посетителя или изменениями состояния документа или окна. Важная особенность JavaScript - объектная ориентированность. Программисту доступны многочисленные объекты, такие, как документы, гиперссылки, формы, фреймы и т.д. Объекты характеризуются описательной информацией (свойствами) и возможными действиями (методами) [24].

Vue.js – JavaScript-фреймворк с открытым исходным кодом для создания пользовательских интерфейсов. Легко интегрируется в проекты с использованием других JavaScript-библиотек. Может функционировать как веб-фреймворк для разработки одностраничных приложений в реактивном стиле. Vue.js является JavaScript библиотекой для создания веб-интерфейсов с использованием шаблона архитектуры MVVM (Model-View-ViewModel).

Поскольку Vue.js работает только на уровне представления и не используется для промежуточного программного обеспечения и бэкэнда, он может легко интегрироваться с другими проектами и библиотеками. Vue.js содержит широкую функциональность для уровня представлений и может использоваться для создания мощных одностраничных веб-приложений.

Функции Vue.js:

– реактивные интерфейсы;

– декларативный рендеринг;

– связывание данных;

– директивы (все директивы имеют префикс «V-». В директиву передается значение состояния, а в качестве аргументов используются html атрибуты или Vue JS события);

– логика шаблонов;

– компоненты;

– обработка событий;

– свойства;

– переходы и анимация CSS;

– фильтры.

Основная библиотека Vue.js имеет достаточно маленький размер (всего 17 кБ). Это показывает, что проект, реализованный с помощью Vue.js, будет достаточно быстро загружаться в браузере пользователя при этом затрачивая меньше сетевых ресурсов [25,26].

Backend часть приложения будет написана на языке программирования C#, с использование ASP.NET Core фреймворка. ASP.NET Core — свободно-распространяемый кроссплатформенный фреймворк для создания веб-приложений с открытым исходным кодом. Данная платформа разрабатывается компанией Майкрософт совместно с сообществом и имеет большую производительность по сравнению с ASP.NET и, в частности, всего .Net Framework. Имеет модульную структуру и совместима с такими операционными системами как Windows, Linux и macOS. Также данная платформа была выбрана

по причине дальнейшей интеграции в нее системы объединения распределенных баз данных.

Backend часть приложения может быть развернута на любом компьютере или сервере, под управлением операционной системы семейства GNU\Linux или Microsoft Windows.

В случае операционной системы Microsoft Windows за работу приложение может отвечать веб-сервер Microsoft IIS Web-Server. Microsoft IIS Web-Server – проприетарный набор серверов для нескольких служб Интернета от компании Microsoft. IIS распространяется вместе с операционной системой Microsoft Windows в качестве службы приложения [27].

При управлении операционной системы семейства GNU\Linux в качестве веб-сервера может выступать Nginx. Nginx является мощным кроссплатформенным open-source (бесплатным) продуктом, предназначенным для выполнения различных веб-приложении, балансирование нагрузки и других немаловажных вещей, связанных с работой веб сервисов.

4.2 Функциональные требования к веб-приложению

Функциональные требования к приложению выполнены в виде диаграммы вариантов использования.

Диаграмма вариантов использования – диаграмма, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне

Данное приложение использует один актор – пользователь системы. Его возможности представлены на Use-case диаграмме приложения (рисунок 10).

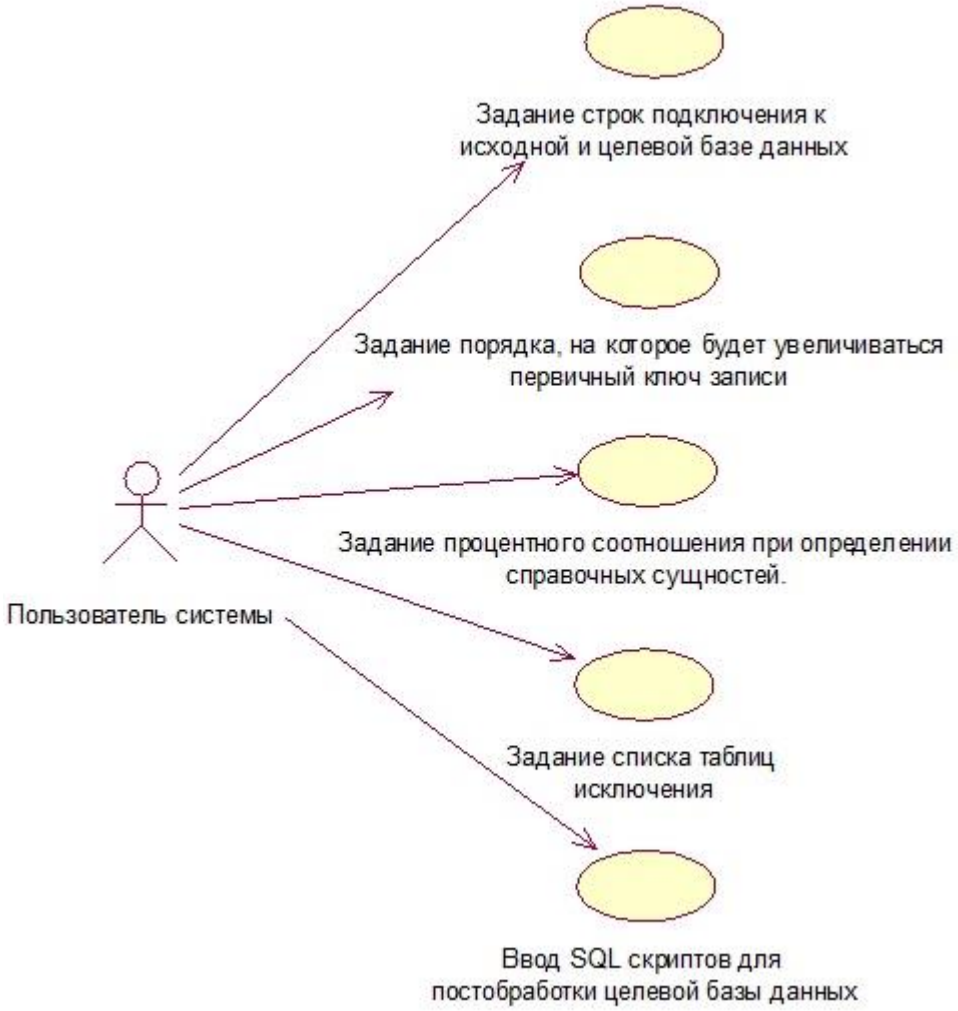


Рисунок 10 – Диаграмма вариантов использования WEB-приложения

Пользователь системы имеет возможность задавать параметры объединения бизнес-сущностей. В эти параметры входят:

– задание строк подключения к исходной и целевой базе данных

– задание порядка, на которое будет увеличиваться первичный ключ записи

– задание процентного соотношения, для исключения несинхронных или динамических сущностей, при определении справочных сущностей.

– задание списка таблиц исключения

– ввод SQL скриптов для постобработки целевой базы данных

Подробное описание возможностей пользователя системы представлены в таблице реестра вариантов использования (таблица 3)

Таблица 3 – Реестр вариантов использования

Код	Основной актор	Наименование	Формулировка	
A1	Пользователь	Редактирование списка исключений	Этот вариант использования позволяет пользователю системы редактировать список исключенных таблиц. Таблицы в данном списке будут игнорироваться при выполнении процесса объединения.	
A2	Пользователь	Ввод SQL скриптов для постобработки целевой базы данных	Этот вариант использования позволяет пользователю системы задать SQL скрипты, которые будут выполнены после процесса объединения на целевой базе данных.	
A3	Пользователь	Задание порядка увеличения первичного ключа	Этот вариант использования позволяет пользователю системы задать порядок увеличения первичного ключа у бизнес-сущностей.	
A4	Пользователь	Задание процентного соотношения, для исключения несинхронных или динамических сущностей, при сравнении справочных сущностей	Этот вариант использования позволяет пользователю системы задать процентное соотношение, при котором после сравнения справочных-сущностей, сущности ниже данного процентного соотношения будут исключены из алгоритма объединения справочных-сущностей.	

Данная таблица описывает возможности действий, которые предоставляет система для пользователя.

4.3 Модель данных

Данная система не подразумевает хранение данных пользователя, так как является автоматизированным инструментом для одноразового решения задачи объединения для конкретных распределенных баз данных и так же может часто использоваться локально, так как современные информационные системы и другие автоматизированные платформенные решения не предоставляют внешний доступ к своим базам данных. Следовательно, хранение данных в базе данных является не целесообразным. Более целесообразным в будущем будет ввести функционал экспорта настроек в конфигурационный файл в формате XML или JSON, тем самым повысив кроссплатформенность и переносимость данной системы.

4.4 Диаграмма развертывания

Диаграмма развертывания предназначена для визуализации элементов и компонентов системы, существующих лишь на этапе ее исполнения (runtime). При этом представляются только компоненты-экземпляры программы, являющиеся исполняемыми файлами или динамическими библиотеками. Очевидно, что компоненты, которые не используются на этапе исполнения, на диаграмме развертывания не показываются. Так, компоненты с исходными текстами программ могут присутствовать только на диаграмме компонентов. На

диаграмме развертывания они не указываются. Диаграмма развертывания приложения представлена на рисунке 11.

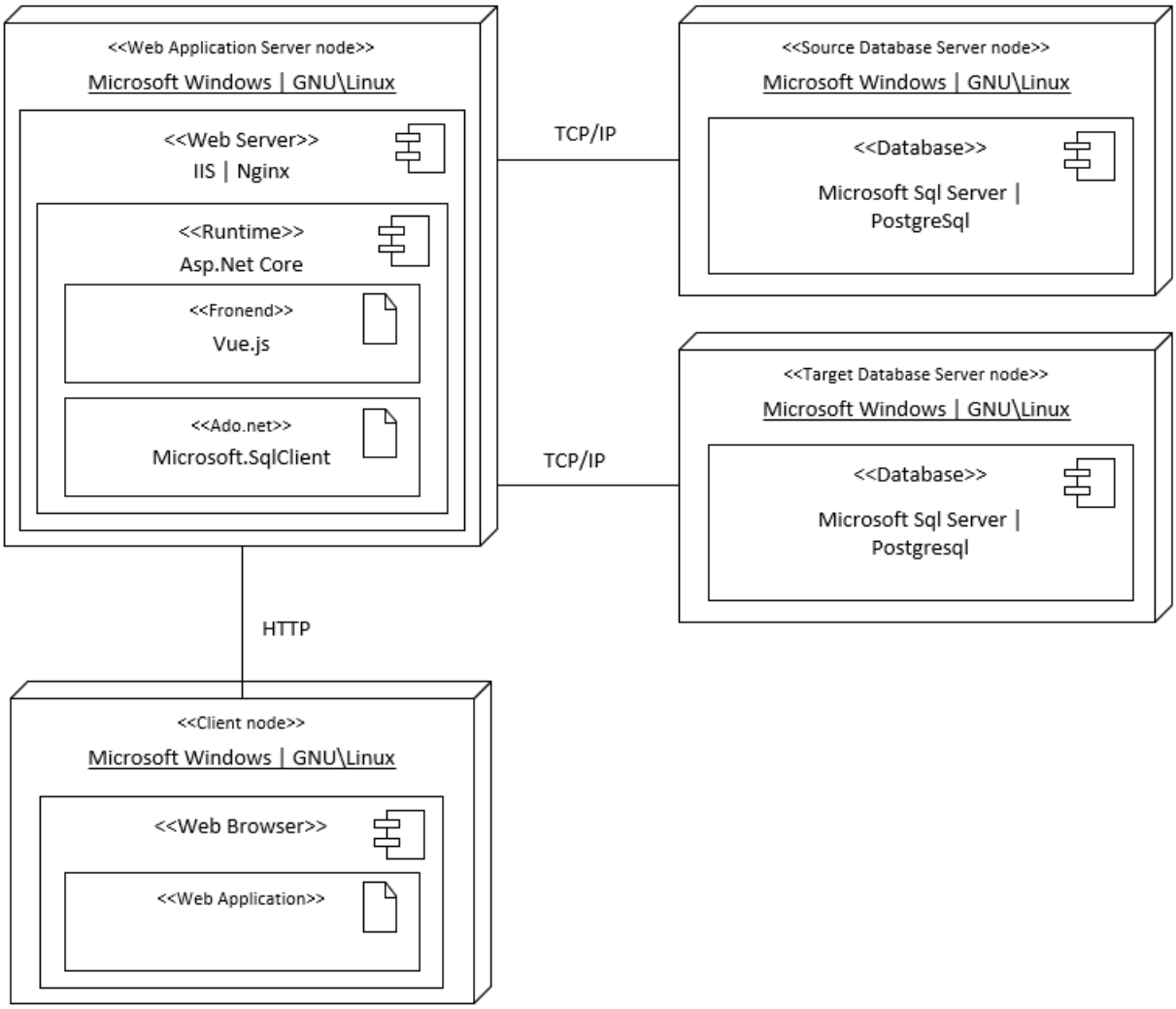


Рисунок 11 – Диаграмма развертывания

Диаграмма развертывания состоит из четырех основных компонентов: клиентское устройство, сервер приложения, исходный сервер базы данных, целевой сервер базы данных.

Клиент взаимодействует с сервером приложения через веб браузер по средствам протокола HTTP. Клиент отсылает HTTP запросы на получение контента страницы (Vue.js), запрашивает и отсылает бизнес данные, отсылает запрос на выполнение бизнес-функций.

Сервер приложения может работать под управлением операционной системы Microsoft Windows или GNU\Linux. В качестве веб сервера может выступать программное обеспечение Microsoft IIS Web Server или Nginx. В качетсве среды выполнения выступает Asp.Net Core Framework.

Сервера баз данных могут работать под управление операционной системы Microsoft Windows или GNU\Linux. В качестве системы управление базами данных может выступать программное обеспечение Microsoft Sql Server или PostgreSql. Сервера баз данных, так же, как и СУБД, должны быть доступны серверу приложения по протоколу TCP\IP.

4.5 Аппаратная инфраструктура

Технические характеристики сервера приложения:

– процессор с тактовой частотой не менее 2 Ггц;

– оперативная память не менее 8 GB;

– свободное дисковое пространство не менее 50 Гб;

– пропускная способность сетевого интерфейса не менее 50 Мбит/с.

Технические характеристики серверов баз данных:

– процессор с тактовой частотой не менее 2 Ггц;

– оперативная память не менее 12 GB;

– свободное дисковое пространство не менее 200 Гб;

– пропускная способность сетевого интерфейса не менее 1 Гбит/с.

К клиентской части приложения особых требований не выдвигается. Для использования полного функционала приложения клиенту необходимо постоянное подключение к сети Интернет и Web-браузер, способный отображать Web-страницы.

4.6 Программная архитектура

Программные требования к серверу приложения:

– операционная система семейства Microsoft Windows или GNU\Linux;

– веб сервер Microsoft IIS или Nginx;

– набор библиотек Microsoft .Net Core версии 2.2 и выше.

Программные требования к серверам баз данных:

– операционная система семейства Microsoft Windows или GNU\Linux;

– система управления базами данных Microsoft Sql Server или PostreSql.

Программные требования к клиенту: Веб браузер Google Chrome, Mozilla Firefox, Opera, Internet Explorer (версии 11) или Edge.

4.7 Разработка прототипа веб-приложения для объединения распределенных баз данных

Как правило, работа над созданием клиентской части веб-приложения начинается с оформления прототипа, его подробной структуры. Самый простой способ это сделать — нарисовать ручкой на бумаге. Но чем сложнее проект и требовательнее заказчик, тем выше требования к макету, и тут эскизом на бумаге уже не обойтись.

Чтобы создать прототип сайта, используют специальные программы и онлайн-сервисы, причем популярность последних постоянно растет из-за простоты и удобства.

Прототип — это базовый макет сайта, который визуализирует расположение всех элементов и функций. Он позволяет наглядно проиллюстрировать все задумки, а также внести правки ценой минимальных усилий и расходов.

Прототипы отличаются по виду, уровню визуализации, интерактивности. От рисунка на листке бумаги до кликабельной многостраничной структуры, все прототипы выполняют одну задачу — синхронизировать представления заказчика и исполнителя о том, как должен выглядеть результат.

Обычно пользователи под прототипом понимают поведенческую модель, в которой не реализуются все слои архитектуры системы, но которая обладает предполагаемым интерфейсом пользователя. Такой прототип называется горизонтальным, или поведенческим.

Горизонтальный прототип не реализует функциональности системы в действительности, он создает только ее видимость, поэтому трудоемкость разработки горизонтального прототипа невелика. Экраны интерфейса пользователя, навигация между ними показывают функциональные возможности и структуру доступа к информации, что позволяет пользователю исследовать поведение системы в различных ситуациях, выяснить возможность выполнения необходимой работы и уточнить требования.

Вертикальный, или структурный прототип служит для проверки концепции, реализует часть функциональности системы на всех уровнях от интерфейса пользователя до сервисных функций. Такой прототип действует как настоящая система и позволяет оптимизировать алгоритмы, оценить базу данных, определить критические временные требования. Вертикальный прототип полезен для исследования критически важных требований к интерфейсу и времени исполнения, для сокращения рисков на этапе проектирования системы.

4.8 Горизонтальный прототип приложения

Для разработки горизонтального прототипа приложения было использовано программное обеспечение Adobe Experience Design (Adobe XD).

Adobe XD — это программный инструмент, созданный разработчиками Adobe и предназначенный для разработки интерфейсов и прототипов мобильных, веб и десктопных приложений. Интерфейс программы Adobe XD представлен на рисунке 12.

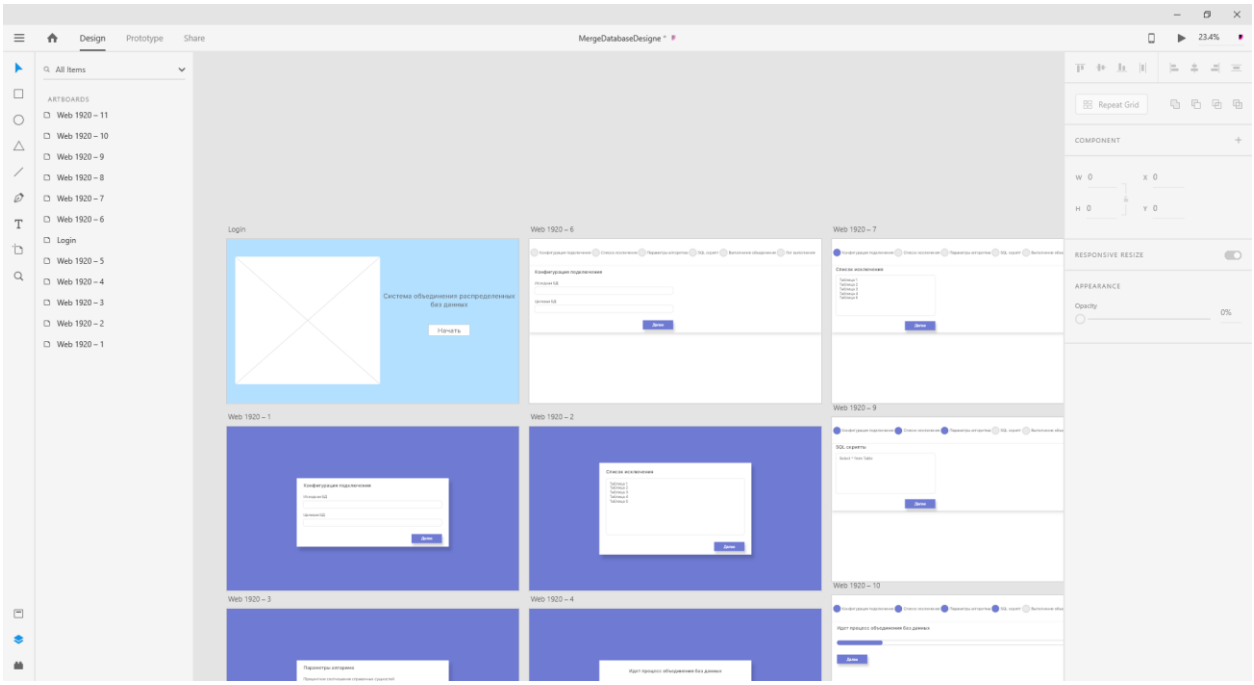


Рисунок 12 – Интерфейс программы Adobe XD

Точкой входа в приложение является стартовый экран, представленный на рисунке 13. На стартовом экран отображено название приложения и кнопка действия начала работы с приложением.

Так как данное приложение создано для решения задачи объединения распределенных баз данных, то основной интерфейс взаимодействия веб-приложения было сделано в пошаговом виде, подобно установке программного обеспечения на персональный компьютер. Пользователь на каждом шаге заполняет необходимые данные и переходит на следующий шаг. В конце происходит выполнение алгоритма объединения, с заданными, пользователем, параметрами. Данное решение также позволило сделать более интуитивный и

понятный интерфейс для конечного пользователя и уменьшить вероятность неправильной работы программы или действий пользователя.

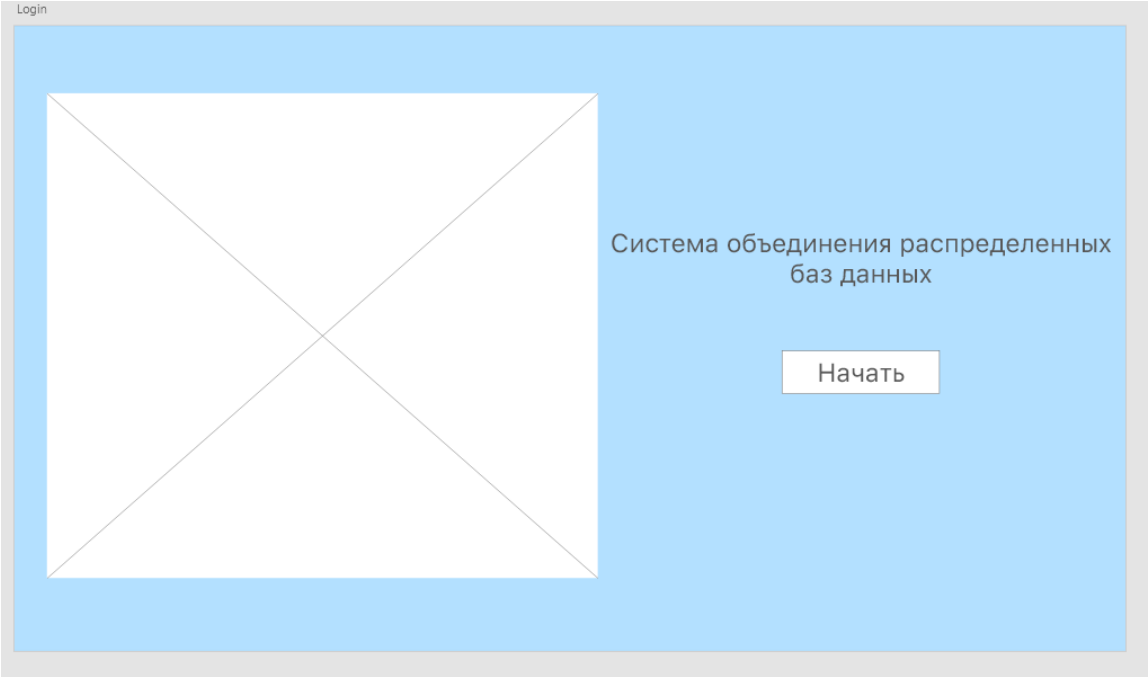


Рисунок 13 – Прототип стартового экрана приложения

На первом шаге «Конфигурация подключения» пользователю необходимо ввести строки подключения к исходной и целевой базе данных. После ввода, пользователь нажимает кнопку далее, приложение отправляет HTTP запрос на сервер, который в свою очередь проверяет наличие подключения к базам данных. Если проверка прошла успешно, пользователь переходит на следующий шаг. Если же проверка проходит с ошибкой, приложение уведомляет пользователя об отсутствии наличия подключения к данным базам данных и ожидает на текущем шаге. Прототип первого шага изображен на рисунке 14.

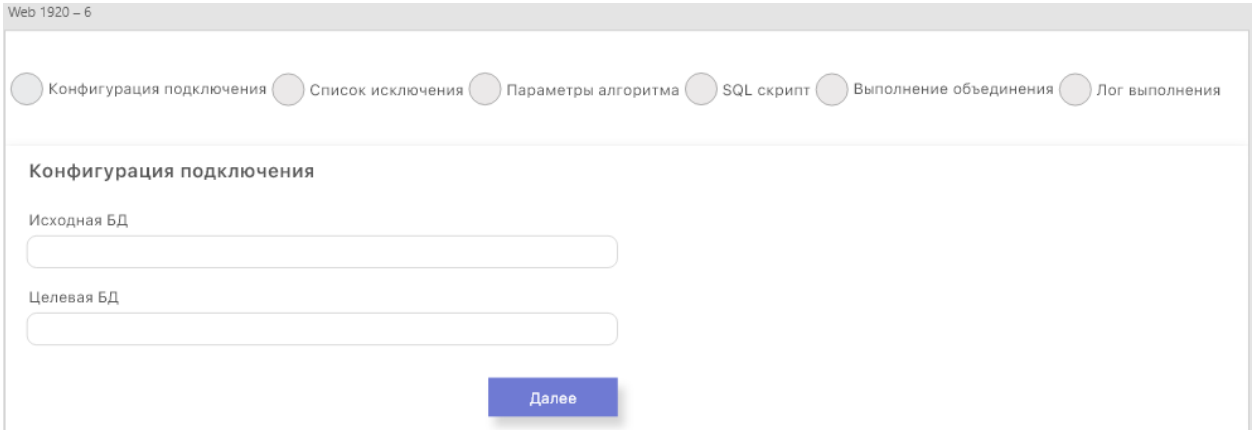


Рисунок 14 – Прототип первого шага «Конфигурация подключения»

Далее пользователь переходит на второй шаг «Список исключения», на котором ему предлагается внести имена таблиц, которые будут исключены из работы алгоритма объединения. На данном шаге заполнения поля «Список исключения» является не обязательным. По нажатию на кнопку «Далее», пользователь переходит к следующему шагу. Прототип второго шага изображен на рисунке 15.

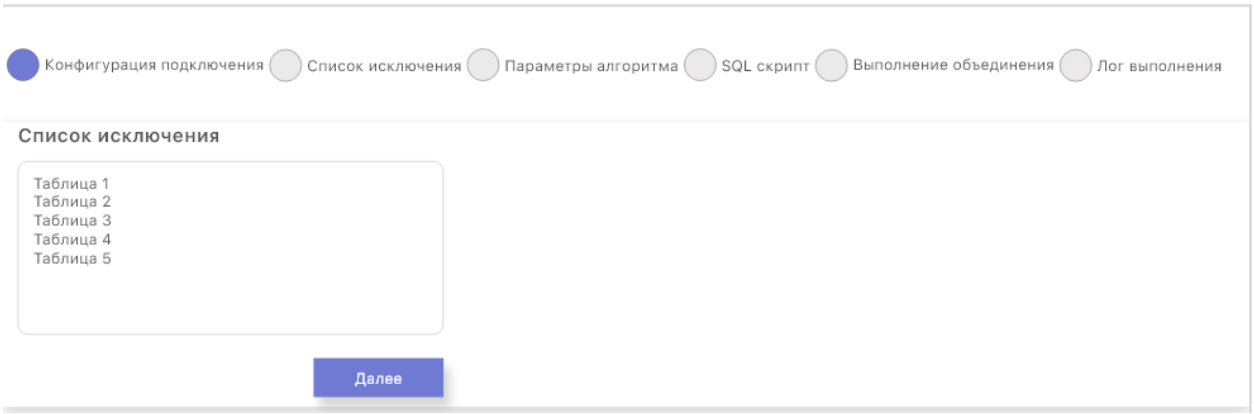


Рисунок 15 – Прототип второго шага «Список исключения»

Далее пользователь переходит на третий шаг «Параметры алгоритма», на котором ему необходимо заполнить следующие параметры:

– процентное соотношение справочных сущностей;

– степень увеличение первичного ключа.

Изначально, данные поля заполнены значениями по умолчанию. По желанию пользователя, он может изменить данные параметры или отставить по умолчанию. Так же, на данных полях присутствует валидация, что позволяет не допустить ввод некорректных данных. По нажатию на кнопку «Далее», происходит валидация введенных данных, и в случае успеха, пользователь переходит к следующему шагу. Прототип третьего шага изображен на рисунке 16.

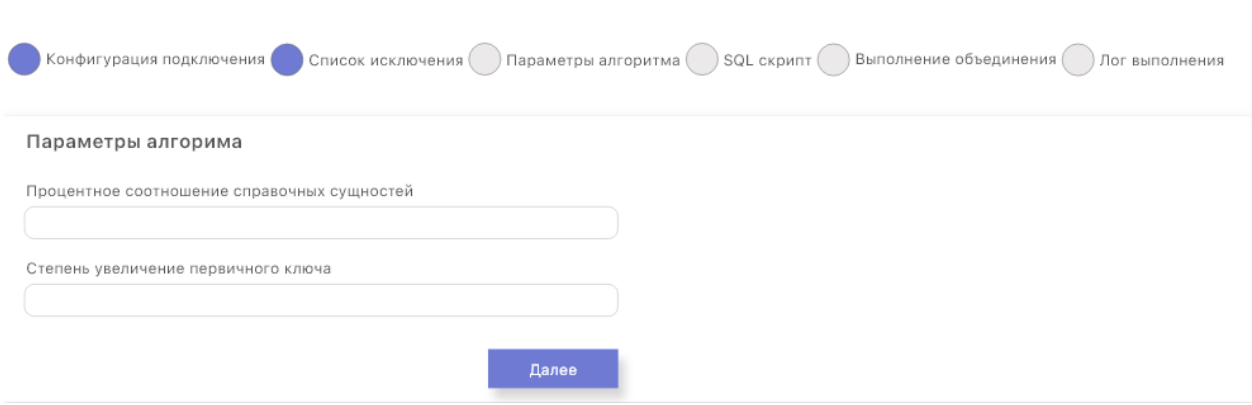


Рисунок 16 – Прототип третьего шага «Параметры алгоритма»

Далее пользователь переходит на четвертый шаг «SQL скрипты», на котором ему предлагается заполнить SQL скрипты, которые будут выполнены после выполнения алгоритма объединения баз данных. На данном шаге, заполнения поля «SQL скрипты» является не обязательным. По нажатию на кнопку «Далее» пользователь переходит к следующему шагу. Прототип четвертого шага изображен на рисунке 17.

Рисунок 17 – Прототип четвертого шага «SQL скрипты»

Далее пользователь переходит на пятый шаг «Выполнение объединения». На данном шаге, пользователю отображается шкала процесса выполнения алгоритма объединения баз данных. В период выполнения процесса объединения кнопка далее «Далее» находится в состоянии disabled. По завершению процесса выполнения, пользователю становятся доступной кнопка «Далее», по нажатию на которую, он переходит к следующему шагу. Прототип пятого шага изображен на рисунке 18.

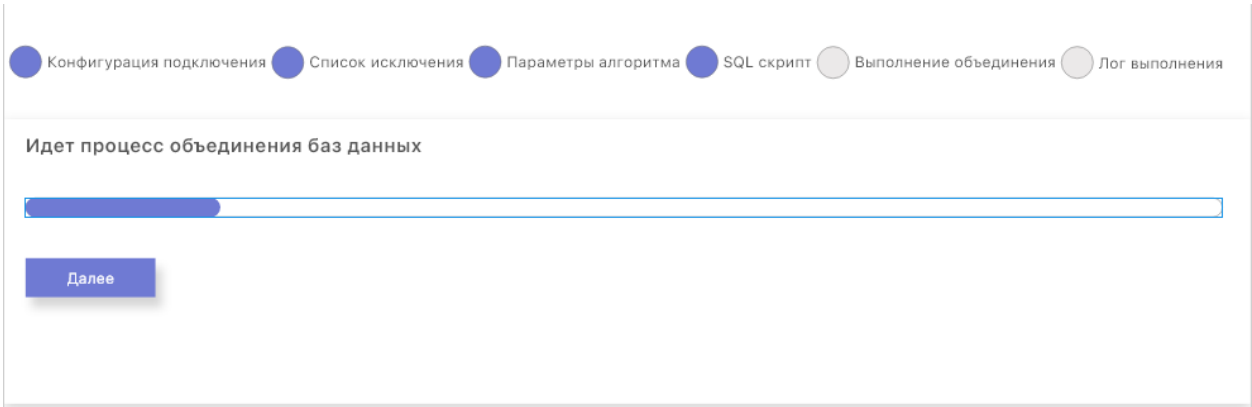


Рисунок 18 – Прототип пятого шага «Выполнение объединения»

Далее пользователь переходит на заключительный шестой шаг «Лог выполнения». На данном шаге, пользователю предлагается скачать лог выполнения алгоритма объединения в текстовом формате. Скачивание

осуществляется по нажатию кнопки «Скачать». Прототип шестого шага изображен на рисунке 19.

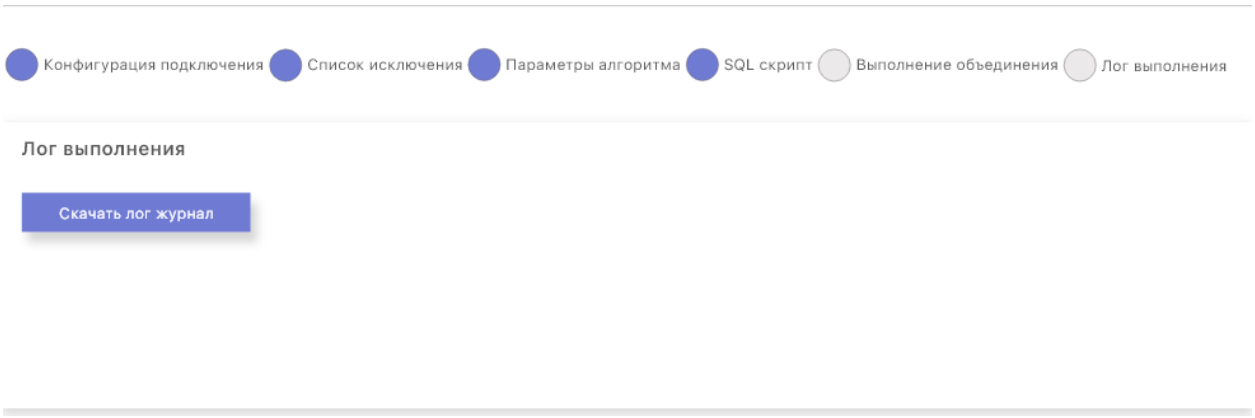


Рисунок 19 – Прототип шестого шага «Лог выполнения»

После этого пользователь может закончить работу в приложении или вернуться на стартовый экран для того, чтобы начать новый процесс объединения распределенных баз данных АИС ЭСРН РМ.

4.9 Вертикальный прототип приложения

Разработка вертикального прототипа приложения велась в редакторе кода Visual Studio Code, с использование языка разметки HTML, формального языка описания внешнего вида документа CSS, языка программирования JavaScript и JavaScript-фреймворка с открытым исходным кодом для создания пользовательских интерфейсов Vue.js. Редактор кода Visual Stuido Code был выбран по причинам высокой поддержки различных плагинов, помогающих в написании кода, подсветки синтаксиса языка, технологий авто дополнения кода и других функций. В соответствии с горизонтальным прототипом сначала был реализован стартовый экран приложения, изображенный на рисунке 20.

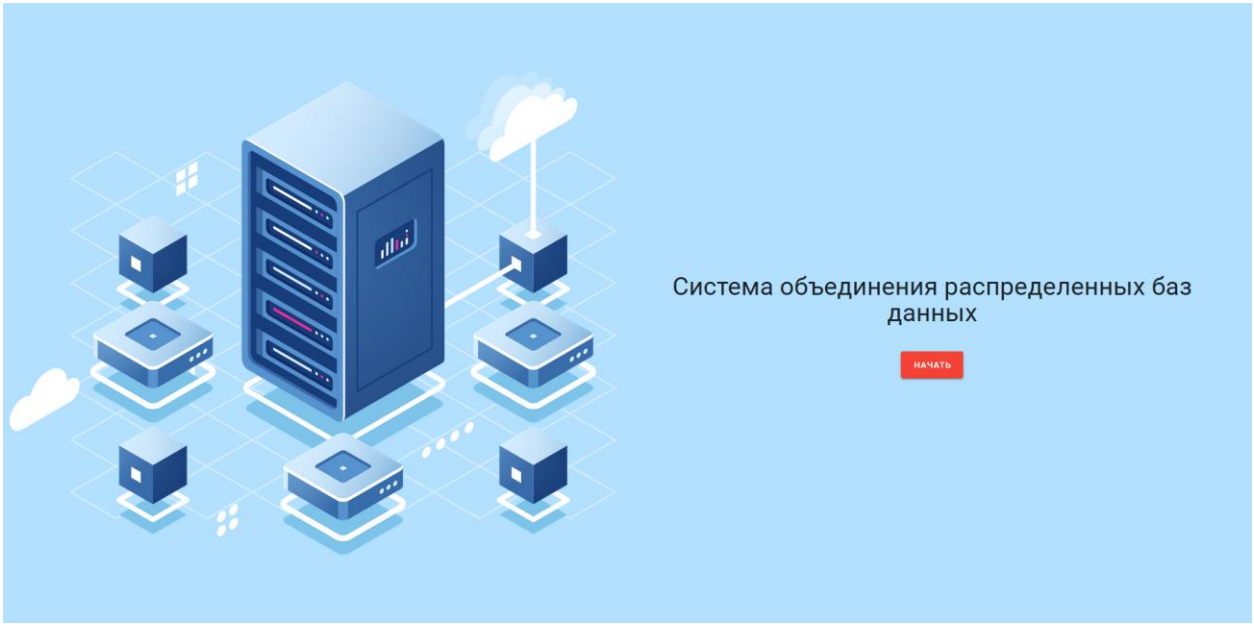


Рисунок 20 – Стартовый экран приложения

Далее в соответствии с горизонтальным прототипом был реализован интерфейс первого шага «Конфигурация подключения», изображенный на рисунке 21. Данный интерфейс содержит два поля ввода, в которых задаются строки подключения к исходной и целевой базе данных ключа, а также кнопку назад для возвращения на предыдущий шаг и далее, для перехода на следующий шаг.

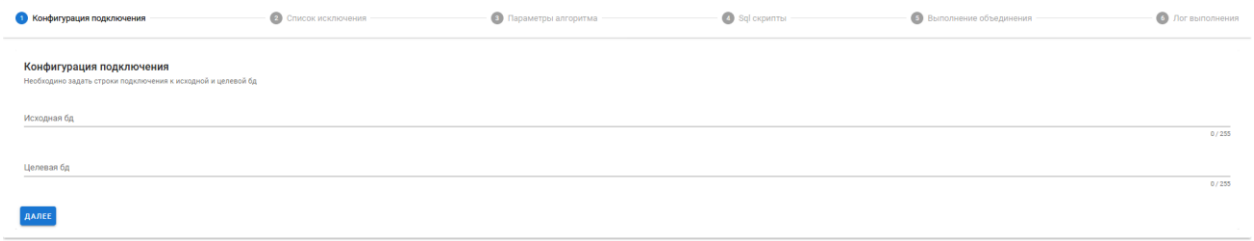


Рисунок 21 – Интерфейс первого шага «Конфигурация подключения»

Также на данном шаге было реализованы следующий функциональные возможности:

– валидация полей формы;

– отправка HTTP запроса на сервер о проверке наличия подключения к базам данных.

Пример реализации валидации полей формы можно видеть на рисунке 22.

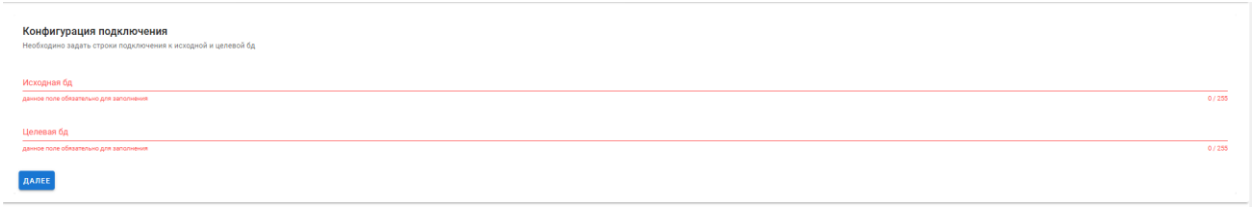


Рисунок 22 – Валидация полей формы на шаге «Конфигурация подключения»

Далее в соответствии с горизонтальным прототипом был реализован интерфейс второго шага «Список исключения», изображенный на рисунке 23. Данный интерфейс содержит строку ввода многострочного текста, куда вводиться через запятую наименование таблиц для исключения их обработки при объединении распределенных баз данных, а также кнопку назад для возвращения на предыдущий шаг и далее, для перехода на следующий шаг.

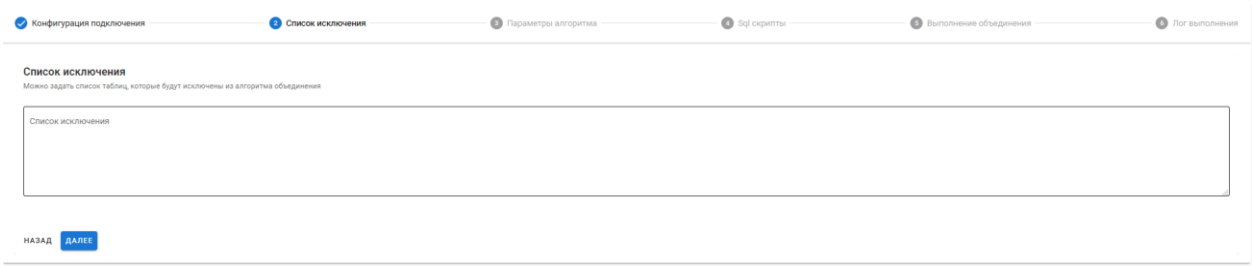


Рисунок 23 – Интерфейс второго шага «Список исключения»

Далее в соответствии с горизонтальным прототипом был реализован интерфейс третьего шага «Параметры алгоритма», изображенный на рисунке 24. Данный интерфейс содержит два поля ввода, в которых задаются параметры сравнения справочных сущностей и степень увеличения первичного ключа, а также кнопку назад для возвращения на предыдущий шаг и далее, для перехода на следующий шаг. Так же, был реализован функционал по валидации полей формы.

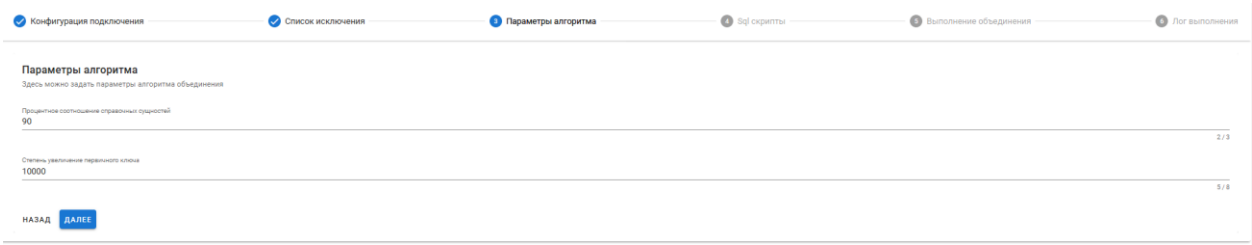


Рисунок 24 – Интерфейс третьего шага «Параметры алгоритма»

Далее в соответствии с горизонтальным прототипом был реализован интерфейс четвертого шага «SQL скрипты», изображенный на рисунке 25. Данный интерфейс содержит строку ввода многострочного текста, куда вводиться SQL скрипты, которые будут выполнены после процедуры объединения распределенных баз данных, а также кнопку назад для возвращения на предыдущий шаг и далее, для перехода на следующий шаг.

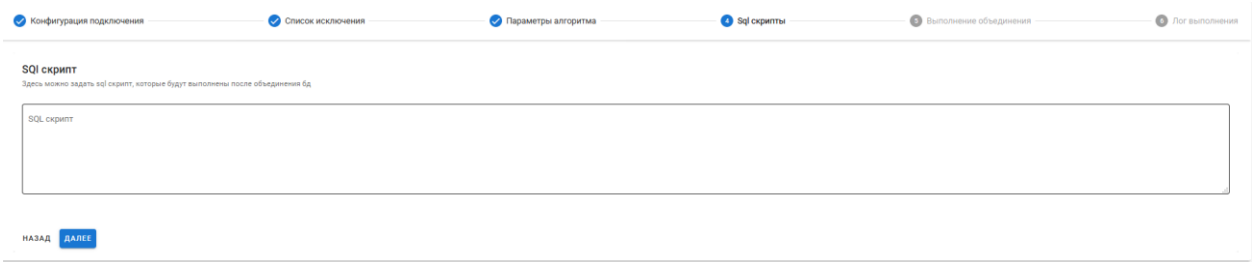


Рисунок 25 – Интерфейс четвертого шага «SQL скрипты»

Далее в соответствии с горизонтальным прототипом был реализован интерфейс пятого шага «Выполнение объединения», изображенный на рисунке 26. Данный интерфейс содержит интерактивную шкалу процесса выполнения, а также кнопку назад для возвращения на предыдущий шаг и далее, для перехода на следующий шаг.

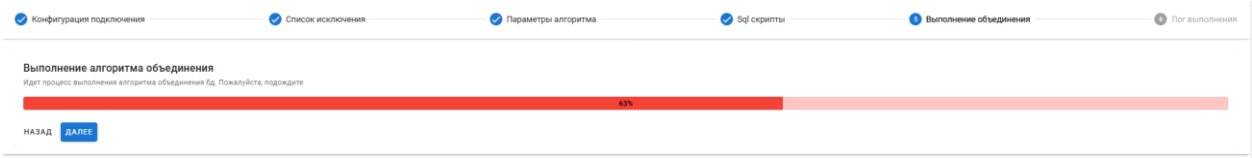


Рисунок 26 – Интерфейс пятого шага «Выполнение объединения»

И в конце был реализован интерфейс заключительного шестого шага «Лог выполнения», изображенный на рисунке 27.

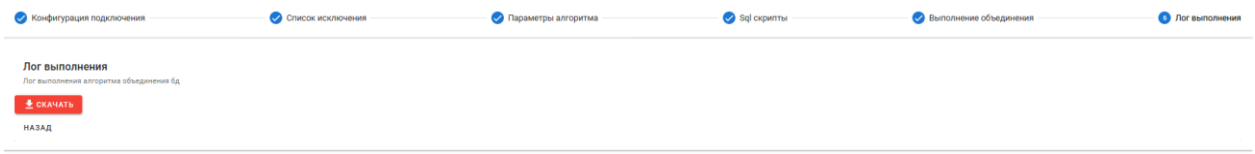


Рисунок 27 – Интерфейс шестого шага «Лог выполнения»

Так же, на данном шаге был реализован функционал по скачиванию лог-файла выполнения алгоритма объединения в текстовом виде.

В заключение была реализована адаптивная поддержка мобильных устройств. Данная поддержка была реализована по следующим причинам:

на сегодняшний день, доля мобильных устройств в сети интернет намного больше персональных компьютеров и других устройств;

стандартная версия интерфейса сайта для персональных компьютеров не может обеспечить пользователю с мобильным устройством должного уровня удобства и комфорта, а в ряде случае происходит искривление интерфейса и выход за границы видимости пользователя.

Для решения вопроса адаптации интерфейса веб-приложения для мобильных устройств существует несколько подходов:

разработка адаптивного интерфейса;

разработка отдельной версии интерфейса веб-приложения для мобильных устройств.

В данном случае был применен адаптивный интерфейс, который динамически, в реальном времени, сам подстраивается под параметры экрана пользователя.

Примеры реализации интерфейсов приложения для мобильных устройств изображены на рисунках 28-29.

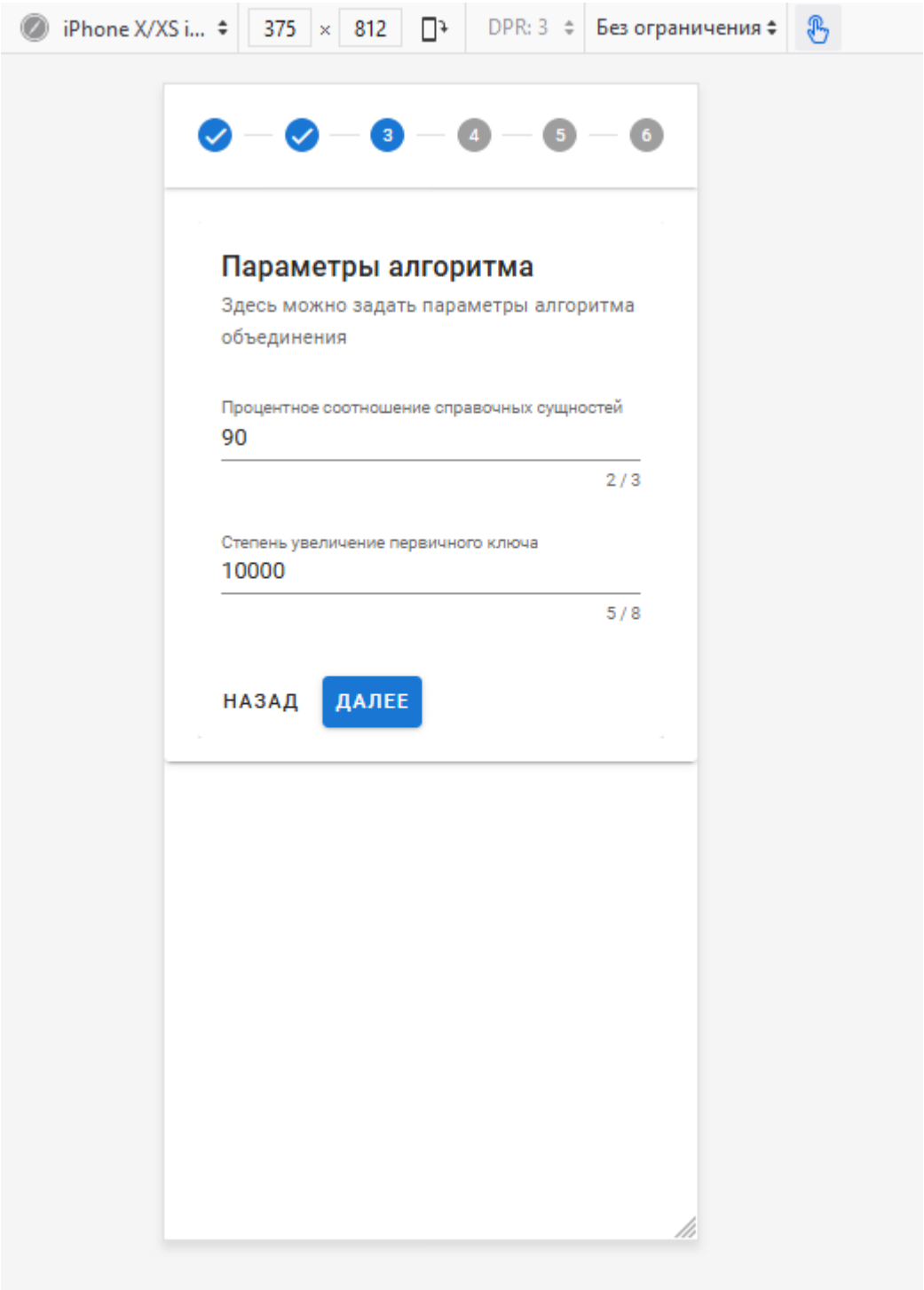


Рисунок 28 – Интерфейс третьего шага «Параметры алгоритма» на мобильных устройствах

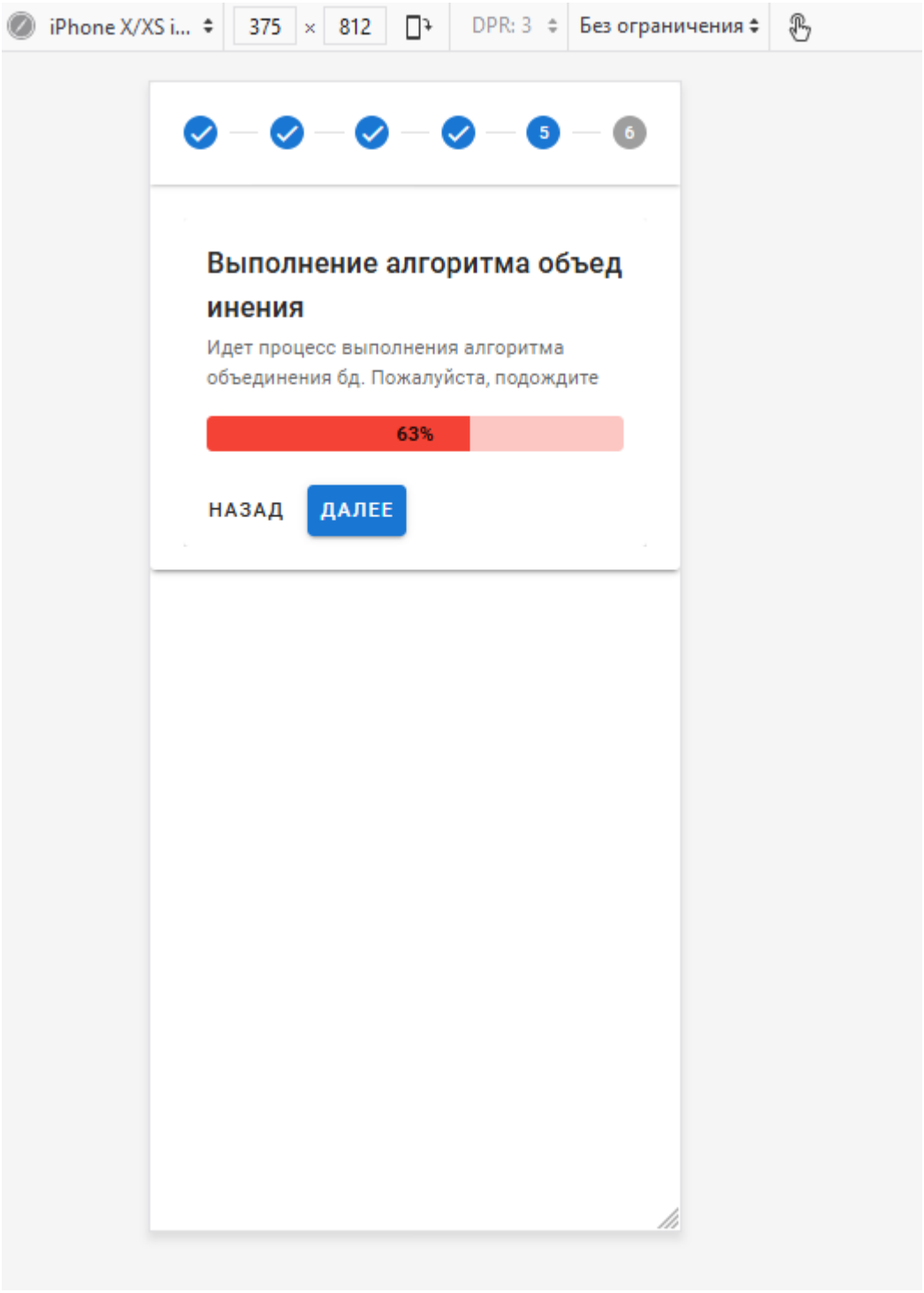


Рисунок 29 – Интерфейс пятого шага «Выполнение объединения» на мобильных устройствах

4.9 Взаимодействие пользователя с системой

Диаграммы последовательностей представляют взаимодействия между линиями жизни как упорядоченную последовательность событий. Это самая богатая и гибкая форма диаграммы взаимодействий. В некоторых случаях моделирование начинают с создания эскиза реализации прецедента с помощью коммуникационной диаграммы, потому что на диаграмме легко размещать и соединять линии жизни.

Для описания взаимодействия пользователя с системой была использована диаграмма последовательности. Описание взаимодействия пользователя с приложением и работа системы происходит следующим образом:

– пользователь вводит строки подключения к исходной и целевой базе данных;

– система проверяет подключение и существование баз данных;

– пользователь заполняет все предложенные параметры системы: вводит список исключения, задает параметр соотношения справочных сущностей, параметр увеличения первичного ключа для бизнес-сущностей, вводит SQL скрипты;

– системы делает резервные копии целевой и исходной базы данных;

– система определяет и объединяет справочные сущности по заданным параметрам;

– система определяет и объединяет бизнес-сущности;

– системы выполняет восстановление ссылочной целостности;

– система выполняет SQL скрипты пользователя;

– система возвращает пользователю лог файл со всеми выполненными операциями.

На каждом шаге выполнения объединения распределенных баз данных система ведет логирование всех своих действий и их результатов, после чего, на конечном шаге, возвращает их пользователю.

Диаграмма последовательности взаимодействия пользователя с приложением и работы системы приведена на рисунке 30.

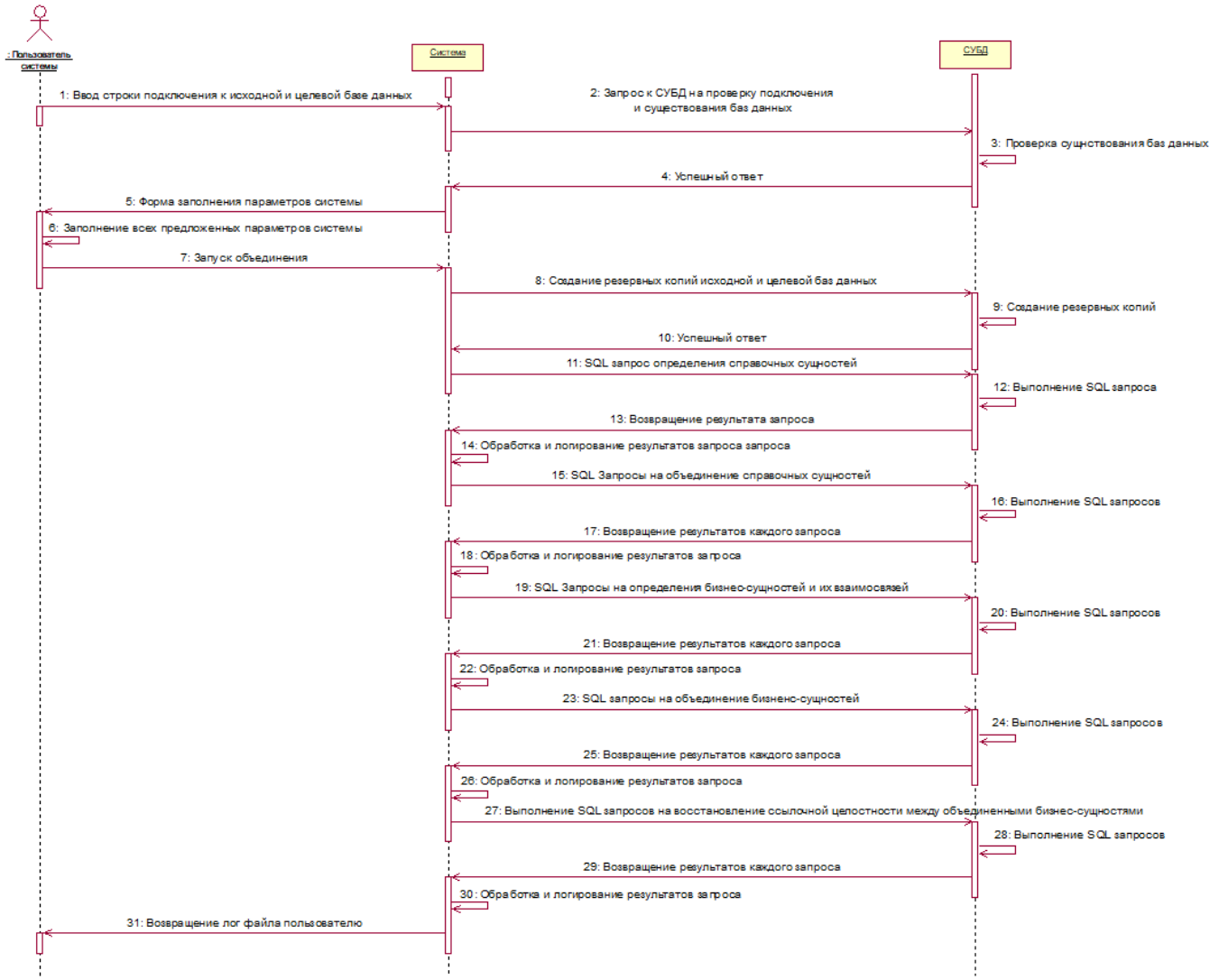


Рисунок 30 – Диаграмма последовательности работы пользователя с системой

ЗАКЛЮЧЕНИЕ

В настоящее время, реализованный программный комплекс внедрен в Министерстве социальной защиты населения и продолжает свою работу по решению задачи автоматизированного объединения распределённых баз данных АИС ЭСРН РМ различных муниципальных районов.

В первой главе были рассмотрены основные теоретические положения о распределенных базах данных, инструментальной платформе разработки бизнес-приложений SiTex, а также метаданных.

Так как реализация данной системы проходила в рамках настоящего задания от Министерства социальной защиты населения Республики Мордовия на объединение распределённых баз данных Инсарского и Кадашкинского муниципального района, то в второй главе было описано техническое задание.

В третьей главе, на основе полученных теоретических знаний и разработанного технического задания была реализована система объединения распределённых баз данных. Реализация данной системы началась с анализа предметной области, в частности в выявлении основных критериев и декомпозиции задачи, которые кардинальным образом влияют на процесс разработки. В ходе декомпозиции задачи были выявлены и реализованы основные алгоритмы по объединению распределенных баз данных: алгоритм определения дерева взаимосвязей бизнес-сущностей, алгоритм определения и объединения справочных-сущностей и алгоритм объединения бизнес-сущностей по выявленному дереву взаимосвязей.

Также, в этой главе были рассмотрены возможные проблемы с безопасностью, в частности возможной потери данных и дальнейшей неработоспособности системы АИС ЭСРН РМ. По итогу были внедрены и реализованы дополнительные подсистемы, предотвращающие возможную потерю данных и неработоспособность системы АИС ЭСРН РМ, а также позволяющие в случае наступления данных ситуаций изучить, разобрать и

исправить данные проблемы.

В четвертой был реализован графический интерфейс пользователя на основе веб-приложения, для комфортного и эффективного использования системы любым пользователем. Далее, в это веб-приложение была внедрена система объединения распределённых баз данных. Данный программный комплекс был развернут в защищенной сети Министерства социальной зашиты населения Республики Мордовия.

В ходе разработки веб-приложения было уделено большое внимание клиентскому интерфейсу, так как основная задача веб-приложения – скрыть от конечного пользователя все сложности и всю специфику работы системы объединения распределенных баз данных, и предоставить пользователю удобный и интуитивно понятный интерфейс.

Реализация веб-приложения началась с разработки горизонтального прототипа клиентской части веб-приложения в среде Adobe XD, который схематично отобразил будущий интерфейс. Далее был спроектирован и реализован вертикальный прототип, который позволил в действительности оценить и проверить работу клиентской части, устранить недостатки и оптимизировать алгоритмы управления клиентской логикой.

В итоге, данное приложение было внедрено и протестировано в реальных условиях, на основе АИС ЭСРН РМ на базах данных Инсарского и Кадошкинского района. Внедрение и тестирование выявило недочеты и ошибки, которые были устранены и данный программный комплекс позволил успешно решить поставленную задачу автоматизированного объединения распределенных баз данных АИС ЭСРН РМ.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Маклаков С. В. BPwin и ERwin. Case-средства разработки информационных систем / С. В. Маклаков. — М. : Диалог-Мифи, 1999. — 256 с.

2. Язык UML Руководство пользовател [Электронный ресурс]. — Режим доступа: http://dit.isuct.ru/ivt/books/CASE/case11/ch30.htm.— Загл. с экрана.

3. Червенчук И. В. Информационные системы и процессы, моделирование и управление. Моделирование информационных систем с помощью UML: учебное пособие/ И. В. Червенчук. — Омск: Омский государственный институт сервиса, 2006. — 48с.

4. Диаграммы классов UML. Логическое моделирование [Электронный ресурс]. — Режим доступа: http://www.informicus.ru/default.aspx?SECTION=6&id=73&subdivisionid=3.— Загл. с экрана.

5. Технология разработки программного обеспечения [Электронный ресурс]. — Режим доступа: http://unesco.kemsu.ru/study_work/method/po/UMK/lab_pract/lab04.html.— Загл. с экрана.

6. Баженова И.Ю. Разработка распределенных приложений баз данных: Курс лекций. - М.: МГУ им. М.В. Ломоносова, 2006. - 203 с.

7. Баканов В.М. Введение в язык SQL запросов к базам данных: Учебное пособие. - М.: МГАПИ, 2002. - 61 с.: ил.

8. Бураков П.В., Петров В.Ю. Введение в системы баз данных: Учебное пособие. - СПб: СПбГУ ИТМО, 2010. - 128 с.

9. Гудов А.М., Шмакова Л.Е. Введение в язык структурированных запросов SQL: Учебное пособие. - Кемерово, Кемеровский госуниверситет, 2001. - 118 с.

10. Дьяков И.А. Базы данных. Язык SQL: Учебное пособие. - Тамбов: ТГТУ, 2004. - 80 с.

11. Зиборов В.В. MS Visual C++ 2010 в среде .NET. Библиотека программиста. - Спб.: Питер, 2012. - 320 с.: ил.

12. Кетков Ю.Л., Кетков А.Ю. Свободное программное обеспечение. FREE PASCAL для студентов и школьников. - СПб.: БХВ-Петербург, 2011. - 384 с.: ил.

13. Копейкин М.В., Спиридонов В.В., Шумова Е.О. Базы данных. Основы SQL реляционных баз данных: Учебное пособие. - СПб.: СЗТУ, 2005. - 160 с.

14. Э.В. Сысоев, Е.В. Бурцева. Базы данных: лекции к курсу. - Тамбов : Изд-во Тамб. гос. техн. ун-та, 2007. - 48 с.

15. Токмаков Г. П. Базы данных. Концепция баз данных, реляционная модель данных, языки SQL и XML: учебное пособие / Г. П. Токмаков. - Ульяновск: УлГТУ, 2010. ? 192 с.

16. Фаро С., Паскаль Л. Рефакторинг SQL-приложений. - Пер. с англ. - СПб: Символ-Плюс, 2009. - 336 с.: ил.

17. Шварц Б., Зайцев П., Ткаченко В., Заводны Дж., Ленц А., Бэллинг Д. MySQL. Оптимизация производительности, 2-е издание. - Пер. с англ. - Спб.: Символ-Плюс, 2010. - 832 с.: ил.

18. Michael Alexander, Richard Kusleika. Access 2013 Bible. - New Jersey: Wiley, 2013 - 1296 p.

19. Grant Allen, Mike Owens. The Definitive Guide to SQLite, Second Edition. - New York: Apress, 2010. - 368 p.

20. Donald Bales. Beginning Oracle PL/SQL. Second edition. - New York: Apress, 2015. - 492 p.

21. Alan Beaulieu. Learning SQL, Second Edition. - California: O'Reilly Media, Inc., 2009. - 338 p.

22. Paul DuBois. MySQL (5th Edition) (Developer's Library). - Massachusetts: Addison-Wesley Professional, 2013. - 1176 p.

23. Ken Greenwood. Sams Teach Yourself ABAP/4 in 21 Days. - Indiana: Sams, 1998. - 752 p.

24. Ken Jones. Microsoft® SQL Server® 2008 T-SQL Fundamentals. - Washington: Microsoft Press, 2008. - 688 p.

25. James Wood. Object-Oriented Programming with ABAP Objects. 2nd, updated and revised edition 2016. - Massachusetts: SAP Press, 2015 - 470 p.

26. Базы данных URL: http://www.site-do.ru/db/db1.php, свободный доступ.

27. Википедия - свободная энциклопедия [Электронный ресурс]. - URL: https://ru.wikipedia.org/wiki/SQL, свободный доступ.

28. Bibeault B., Kats Y. jQuery in Action. – Dreamtech Press, 2008

29. Costa P. J., Duarte P., Costa C. J. WebStorm: mixing brainstorming with art in the web //Proceedings of the 25th annual ACM international conference on Design of communication. – ACM, 2007. – С. 170-175.

30. Duckett J. HTML & CSS: design and build websites. – Indianapolis, IN : Wiley, 2011. – Т. 15.

31. Бальзамов А.В – РАЗРАБОТКА АЛГОРИТМА ОПРЕДЕЛЕНИЯ СПРАВОЧНЫХ СУЩНОСТЕЙ В РАСПРЕДЕЛЕННЫХ РЕЛЯЦИОННЫХ БАЗАХ ДАННЫХ – [Электронный ресурс] — Режим доступа: https://elibrary.ru/item.asp?id=42720183.

ПРИЛОЖЕНИЕ А

(обязательное)

Частичный программный код разработанных модулей

Листинг А.1 — Класс, отвечающий за реализацию определение и объединение бизнес-сущностей

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;

namespace ConsoleApp1
{
    public class WM_Merge
    {
        public List<string> Tables_used { get; set; } = new List<string>();

        public void Execute()
        {
            MergeOrgBase();
            if (Sql.ExecuteSqlDataSet(string.Format("select Count(*) from WM_PERSONAL_CARD where OUID > {0}", (object)Esrn_utils.getDegree("WM_PERSONAL_CARD"))).Tables[0].AsEnumerable().Select<DataRow, int>((Func<DataRow, int>)(t => (int)t.ItemArray[0])).FirstOrDefault<int>() == 0)
                Sql.ExecuteSql(string.Format("SET IDENTITY_INSERT dbo.WM_PERSONAL_CARD ON INSERT INTO WM_PERSONAL_CARD (OUID, A_PY, A_NKAR, A_DATZK, BIRTHDATE, SURNAME, A_NAME, A_SECONDNAME, A_SEX, A_IDCODE, A_SOCIALID, A_SPUNID, A_CITIZENSHIP, A_PHOTO, A_CARDSTATUS, A_DATPSY, A_DEATHDATE, A_PLACEOFBIRTH, A_ARMYDATEFROM, A_ARMYDATETO, SocialCategory, A_CODE_FML, A_CODE_SZ, A_CODE_CP, GUID, A_CREATEDATE, A_CROWNER, TS, A_EDITOWNER, A_STATUS, A_REGFLAT, A_LIVEFLAT, A_REGFLATDATE, A_REMREGFLATDATE, A_DATIVENAME, A_ISTEMPREGFLAT, A_ISNOTCOMPRALADR, A_PENSIONNUM, A_TEMPREGFLAT, A_TEMPREGFLATDATE, A_REMTREGFLATDATE, A_DEAL, A_INN, A_PERSONAGE, A_SPUNID_DATE, A_ACTIVE_SOC_CARD, A_VED_SYS_JKH_INFOOBMEN, A_VED_SYS_SZN_INFOOBMEN, A_PERSON_JKHID, A_PERSON_SZNID, A_MUNICIPAL_OBRAZ, A_SNILS, A_ISTEMPORARY, A_DESCRIPTION, A_NUM_ENS_PENSOF, A_END_DATE, A_BEGIN_DATE, A_TRYDST, A_PRAB, A_DEBT_AGREE, A_FERBOROUGH, A_GOS, A_REGFLATCOMPPL, A_TEMPREGFLATCOMPPL, A_TEL_NEED, A_FUEL_DATE, A_FUEL_TYPE, A_TRANSPORT_NEED, A_HOME_DATE, A_HOME_SERV_TYPE, A_STAC_DATE,
```

```
A_STAC_ORG_TYPE, A_STAGMONTH, A_STDAY, A_SELFSERVICE, A_PCSTATUS, A_STATUSCHANGEDATE,
A_IPR, A_ISREPLOBJ, A_REG_ORGNAME, A_MUNICSTAGE, A_STATESTAGE, A_BIRTH_PL_IMP,
A_FACT_PL_IMP, A_CONTROL, A_hobbies, A_MANY_FLAT, A_LIVEFLATCOMPPL, A_PFRPENSIONNUM,
A_REGIONCOEFF, A_SYSTEMCLASS, A_LIVEFLATDATE, A_REMLIVEFLATDATE, A_FAMILY_USON, A_EMAIL,
A_SOC_CAT) Select OUID = OUID + {0}, A_PY, A_NKAR, A_DATZK, BIRTHDATE, SURNAME, A_NAME,
A_SECONDNAME, A_SEX, A_IDCODE, A_SOCIALID, A_SPUNID, A_CITIZENSHIP, A_PHOTO,
A_CARDSTATUS, A_DATPSY, A_DEATHDATE, A_PLACEOFBIRTH, A_ARMYDATEFROM, A_ARMYDATETO,
SocialCategory, A_CODE_FML, A_CODE_SZ, A_CODE_CP, GUID, A_CREATEDATE, A_CROWNER, TS,
A_EDITOWNER, A_STATUS, A_REGFLAT, A_LIVEFLAT, A_REGFLATDATE, A_REMREGFLATDATE,
A_DATIVENAME, A_ISTEMPREGFLAT, A_ISNOTCOMPRALADR, A_PENSIONNUM, A_TEMPREGFLAT,
A_TEMPREGFLATDATE, A_REMTREGFLATDATE, A_DEAL, A_INN, A_PERSONAGE, A_SPUNID_DATE,
A_ACTIVE_SOC_CARD, A_VED_SYS_JKH_INFOOBMEN, A_VED_SYS_SZN_INFOOBMEN, A_PERSON_JKHID,
A_PERSON_SZNID, A_MUNICIPAL_OBRAZ, A_SNILS, A_ISTEMPORARY, A_DESCRIPTION,
A_NUM_ENS_PENSOF, A_END_DATE, A_BEGIN_DATE, A_TRYDST, A_PRAB, A_DEBT_AGREE, A_FERBOROUGH,
A_GOS, A_REGFLATCOMPPL, A_TEMPREGFLATCOMPPL, A_TEL_NEED, A_FUEL_DATE, A_FUEL_TYPE,
A_TRANSPORT_NEED, A_HOME_DATE, A_HOME_SERV_TYPE, A_STAC_DATE, A_STAC_ORG_TYPE,
A_STAGMONTH, A_STDAY, A_SELFSERVICE, A_PCSTATUS, A_STATUSCHANGEDATE, A_IPR, A_ISREPLOBJ,
A_REG_ORGNAME, A_MUNICSTAGE, A_STATESTAGE, A_BIRTH_PL_IMP, A_FACT_PL_IMP, A_CONTROL,
A_hobbies, A_MANY_FLAT, A_LIVEFLATCOMPPL, A_PFRPENSIONNUM, A_REGIONCOEFF, A_SYSTEMCLASS,
A_LIVEFLATDATE, A_REMLIVEFLATDATE, A_FAMILY_USON, A_EMAIL, A_SOC_CAT FROM
[kadosh].dbo.WM_PERSONAL_CARD SET IDENTITY_INSERT dbo.WM_PERSONAL_CARD OFF",
(object)Esrn_utils.getDegree("WM_PERSONAL_CARD")));
                List<object[]> list = Sql.ExecuteSqlDataSet("select class.OUID,
class.MAP, attr.OUID, attr.OUIDDATATYPE, attr.MAP, class2.OUID, class2.MAP from SXCLASS
class left join SXATTR attr on attr.OUIDSXCLASS = class.OUID left join SXDATATYPE type
on attr.OUIDDATATYPE = type.OUID left join SXATTR attr2 on attr.REF_ATTR = attr2.OUID
left join SXCLASS class2 on attr.REF_CLASS = class2.OUID where (class.MAP LIKE 'wm%' OR
class.MAP LIKE 'ESRN_SERV%' OR class.MAP LIKE 'SPR_ORG_BASE') and type.A_CODE in (7,9,10)
and class2.MAP like 'WM_PERSONAL_CARD'").Tables[0].AsEnumerable().Select<DataRow,
object[]>((Func<DataRow, object[]>)(t => t.ItemArray)).ToList<object[]>();
                Merge(list);
                foreach (object[] objArray in list)
                {
                    if (!Tables_used.Contains(objArray[1].ToString()))
                        RecursiveExecute(objArray[1].ToString());
                }
                Merge(Sql.ExecuteSqlDataSet("select class.OUID, class.MAP, attr.OUID,
attr.OUIDDATATYPE, attr.MAP, class2.OUID, class2.MAP from SXCLASS class  left join SXATTR
attr on attr.OUIDSXCLASS = class.OUID left join SXDATATYPE type on attr.OUIDDATATYPE =
type.OUID left join SXATTR attr2 on attr.REF_ATTR = attr2.OUID left join SXCLASS class2
on attr.REF_CLASS = class2.OUID where (class.MAP LIKE 'wm%' OR class.MAP LIKE
```

```
'ESRN_SERV%')     and     type.A_CODE     in     (7,9,10)     and     class2.MAP     like
'SPR_ORG_BASE'").Tables[0].AsEnumerable().Select<DataRow,       object[]>((Func<DataRow,
object[]>)(t => t.ItemArray)).ToList<object[]>());
            }

            public void RecursiveExecute(string table)
            {
                Tables_used.Add(table);
                List<object[]>  list  =  Sql.ExecuteSqlDataSet(string.Format("select
class.OUID, class.MAP, attr.OUID, attr.OUIDDATATYPE, attr.MAP, class2.OUID, class2.MAP
from SXCLASS class  left join SXATTR attr on attr.OUIDSXCLASS = class.OUID left join
SXDATATYPE type on attr.OUIDDATATYPE = type.OUID left join SXATTR attr2 on attr.REF_ATTR
= attr2.OUID left join SXCLASS class2 on attr.REF_CLASS = class2.OUID where (class.MAP
LIKE  'wm%'  OR  class.MAP  LIKE  'ESRN_SERV%'  OR  class.MAP  LIKE  'SPR_ORG_BASE')  and
type.A_CODE       in       (7,9,10)       and       class2.MAP       like       '{0}'",
(object)table)).Tables[0].AsEnumerable().Select<DataRow,       object[]>((Func<DataRow,
object[]>)(t => t.ItemArray)).ToList<object[]>();
                Merge(list);
                foreach (object[] objArray in list)
                {
                    if (!Tables_used.Contains(objArray[1].ToString()))
                        RecursiveExecute(objArray[1].ToString());
                }
            }

            private void MergeOrgBase()
            {
                Sql.ExecuteSql(string.Format("SET   IDENTITY_INSERT   dbo.SPR_ORG_BASE
ON\t\t\t\t\t\t\tINSERT          INTO          SPR_ORG_BASE({0})\t\t\t\t\t\t\tSelect
{1}\t\t\t\t\t\t\tFROM [kadosh].dbo.SPR_ORG_BASE Where GUID not in (select GUID from
SPR_ORG_BASE)\t\t\t\t\t\t\tSET      IDENTITY_INSERT      dbo.SPR_ORG_BASE      OFF",
(object)Esrn_utils.getColumnList("SPR_ORG_BASE"),
(object)Esrn_utils.getColumnList("SPR_ORG_BASE").Replace(Esrn_utils.getPKColumn("SPR_OR
G_BASE"), string.Format("{0} = ((select Max({1}) from SPR_ORG_BASE) + ROW_NUMBER() over
(order        by        {2}))",        (object)Esrn_utils.getPKColumn("SPR_ORG_BASE"),
(object)Esrn_utils.getPKColumn("SPR_ORG_BASE"),
(object)Esrn_utils.getPKColumn("SPR_ORG_BASE"))));
            }

            public void Merge(List<object[]> dataset)
            {
```

```
                    if (dataset.Count == 0)
                        return;
                    Console.WriteLine("[Update Domain] - " + dataset[0][6].ToString());
                    foreach (object[] objArray in dataset)
                    {
                        if        (Esrn_utils.isObject(objArray[1].ToString())        &&
Esrn_utils.isObjectContains(objArray[1].ToString()))
                        {
                            Esrn_utils.checkCount(objArray[1].ToString());
                            switch ((int)objArray[3])
                            {
                                case 10121247:
                                    string sql_command1 = "";
                                    try
                                    {
                                        if (objArray[6].ToString() == "SPR_ORG_BASE")
                                            sql_command1  =  string.Format("UPDATE  resSET
res.{0} = sob.OUIDFROM {1} AS resINNER JOIN [kadosh].dbo.{2} AS kadON res.{3} = kad.{4}
+ {5}INNER JOIN [kadosh].dbo.SPR_ORG_BASE kad_sobON kad.{6} = kad_sob.OUIDINNER JOIN
SPR_ORG_BASE    sobON    kad_sob.GUID    =    sob.GUIDWHERE    res.{7}    >    {8}",
(object)objArray[4].ToString(),                          (object)objArray[1].ToString(),
(object)objArray[1].ToString(), (object)Esrn_utils.getPKColumn(objArray[1].ToString()),
(object)Esrn_utils.getPKColumn(objArray[1].ToString()),
(object)Esrn_utils.getDegree(objArray[1].ToString()),   (object)objArray[4].ToString(),
(object)Esrn_utils.getPKColumn(objArray[1].ToString()),
(object)Esrn_utils.getDegree(objArray[1].ToString()));
                                        else
                                            sql_command1  =  string.Format("UPDATE  resSET
res.{0} = kad.{1} + {2}FROM {3} AS resINNER JOIN [kadosh].dbo.{4} AS kadON res.{5} =
kad.{6}    +    {7}WHERE    res.{8}    >    {9}",    (object)objArray[4].ToString(),
(object)objArray[4].ToString(),   (object)Esrn_utils.getDegree(objArray[6].ToString()),
(object)objArray[1].ToString(),                          (object)objArray[1].ToString(),
(object)Esrn_utils.getPKColumn(objArray[1].ToString()),
(object)Esrn_utils.getPKColumn(objArray[1].ToString()),
(object)Esrn_utils.getDegree(objArray[1].ToString()),
(object)Esrn_utils.getPKColumn(objArray[1].ToString()),
(object)Esrn_utils.getDegree(objArray[1].ToString()));
                                        Sql.ExecuteSql(sql_command1);
                                        continue;
                                    }
                                    catch (Exception ex)
```

```
                                {
                                    Console.WriteLine("[ERROR UPDATE DOMAINS 1x1] - "
+ ex.Message + Environment.NewLine + "SQL : " + sql_command1 + Environment.NewLine +
"LinkOUID: " + objArray[2].ToString());
                                    Log.Logging("[ERROR UPDATE DOMAINS 1x1] - " +
ex.Message + Environment.NewLine + "SQL : " + sql_command1 + Environment.NewLine +
"LinkOUID: " + objArray[2].ToString());
                                    continue;
                                }
                            case 10121248:
                                string sql_command2 = "";
                                try
                                {
                                    sql_command2       =       string.Format("select
A_CLASS_DESCR from SXATTR where ouid = {0}", (object)objArray[2].ToString());
                                    string                    str                    =
Sql.ExecuteSqlDataSet(sql_command2).Tables[0].AsEnumerable().Select<DataRow,
string>((Func<DataRow,                              string>)(t                              =>
t.ItemArray[0].ToString())).FirstOrDefault<string>();
                                    if (string.IsNullOrEmpty(str))
                                    {
                                        Console.WriteLine("[ERROR UPDATE DOMAINS MxN]
- NO CLASS DESCR" + Environment.NewLine + "LinkOUID: " + objArray[2].ToString());
                                        Log.Logging("[ERROR UPDATE DOMAINS MxN] - NO
CLASS DESCR" + Environment.NewLine + "LinkOUID: " + objArray[2].ToString());
                                        continue;
                                    }
                                    sql_command2  =  string.Format("select  A_SQL_VIEW
from SXCLASS where ouid = {0}", (object)str);
                                    if
(string.IsNullOrEmpty(Sql.ExecuteSqlDataSet(sql_command2).Tables[0].AsEnumerable().Sele
ct<DataRow,              string>((Func<DataRow,              string>)(t              =>
t.ItemArray[0].ToString())).FirstOrDefault<string>()))
                                    {
                                        sql_command2 = string.Format("select Map from
SXCLASS where OUID = {0}", (object)str);
                                        string                 table                 =
Sql.ExecuteSqlDataSet(sql_command2).Tables[0].AsEnumerable().Select<DataRow,
string>((Func<DataRow,                              string>)(t                              =>
t.ItemArray[0].ToString())).FirstOrDefault<string>();
                                        if (!string.IsNullOrEmpty(table))
```

```
                                        {
                                            if (!Tables_used.Contains(table))
                                            {
                                                Esrn_utils.checkCount(table);
                                                Tables_used.Add(table);
                                                sql_command2 = string.Format("select
attr.MAP, class2.MAP from SXCLASS class  left join SXATTR attr on attr.OUIDSXCLASS =
class.OUID left join SXDATATYPE type on attr.OUIDDATATYPE = type.OUID left join SXATTR
attr2 on attr.REF_ATTR = attr2.OUID left join SXCLASS class2 on attr.REF_CLASS =
class2.OUID where class.MAP LIKE '{0}'  and type.A_CODE in (7,9,10) and class2.OUID is
not Null and class2.MAP is NOT NULL", (object)table);
                                                List<object[]>         list         =
Sql.ExecuteSqlDataSet(sql_command2).Tables[0].AsEnumerable().Select<DataRow,
object[]>((Func<DataRow, object[]>)(t => t.ItemArray)).ToList<object[]>();
                                                sql_command2 = string.Format("update
{0} set {1} = {2} + {3}, {4} = {5} + {6} where {7} > {8}", (object)table, list[0][0],
list[0][0], (object)Esrn_utils.getDegree(list[0][1].ToString()), list[1][0], list[1][0],
(object)Esrn_utils.getDegree(list[1][1].ToString()),
(object)Esrn_utils.getPKColumn(table), (object)Esrn_utils.getDegree(table));
                                                Sql.ExecuteSql(sql_command2);
                                                continue;
                                            }
                                            continue;
                                        }
                                        continue;
                                    }
                                    continue;
                                }
                                catch (Exception ex)
                                {
                                    Console.WriteLine("[ERROR UPDATE DOMAINS NxM] - "
+ ex.Message + Environment.NewLine + "SQL : " + sql_command2 + Environment.NewLine +
"LinkOUID: " + objArray[2].ToString());
                                    Log.Logging(string.Format("[ERROR  UPDATE  DOMAIN
NxM]   -   {0}   SQL_Command:   {1}Link_OUID:   {2}\\n",   (object)(ex.Message   +
Environment.NewLine),       (object)(sql_command2       +       Environment.NewLine),
(object)objArray[2].ToString()));
                                    continue;
                                }
                            case 10121250:
                                string sql_command3 = "";
```

```
                                try
                                {
                                    if (objArray[6].ToString() == "SPR_ORG_BASE")
                                        sql_command3 = string.Format(" UPDATE res SET
res.{0} = sob.OUID FROM {1} AS res INNER JOIN [kadosh].dbo.{2} AS kad ON res.{3} = kad.{4}
+ {5} INNER JOIN [kadosh].dbo.SPR_ORG_BASE kad_sob ON kad.{6} = kad_sob.OUID INNER JOIN
SPR_ORG_BASE sob ON kad_sob.GUID = sob.GUID WHERE res.{7} > {8}",
(object)objArray[4].ToString(),                          (object)objArray[1].ToString(),
(object)objArray[1].ToString(), (object)Esrn_utils.getPKColumn(objArray[1].ToString()),
(object)Esrn_utils.getPKColumn(objArray[1].ToString()),
(object)Esrn_utils.getDegree(objArray[1].ToString()),   (object)objArray[4].ToString(),
(object)Esrn_utils.getPKColumn(objArray[1].ToString()),
(object)Esrn_utils.getDegree(objArray[1].ToString()));
                                    else
                                        sql_command3 = string.Format(" UPDATE res SET
res.{0} = kad.{1} + {2} FROM {3} AS res INNER JOIN [kadosh].dbo.{4} AS kad ON res.{5} =
kad.{6} + {7} WHERE res.{8} > {9}", (object)objArray[4].ToString(),
(object)objArray[4].ToString(),   (object)Esrn_utils.getDegree(objArray[6].ToString()),
(object)objArray[1].ToString(),                          (object)objArray[1].ToString(),
(object)Esrn_utils.getPKColumn(objArray[1].ToString()),
(object)Esrn_utils.getPKColumn(objArray[1].ToString()),
(object)Esrn_utils.getDegree(objArray[1].ToString()),
(object)Esrn_utils.getPKColumn(objArray[1].ToString()),
(object)Esrn_utils.getDegree(objArray[1].ToString()));
                                    Sql.ExecuteSql(sql_command3);
                                    continue;
                                }
                                catch (Exception ex)
                                {
                                    Console.WriteLine("[ERROR UPDATE DOMAINS Nx1] - "
+ ex.Message + Environment.NewLine + "SQL : " + sql_command3 + Environment.NewLine +
"LinkOUID: " + objArray[2].ToString());
                                    Log.Logging("[ERROR UPDATE DOMAINS Nx1] - " +
ex.Message + Environment.NewLine + "SQL : " + sql_command3 + Environment.NewLine +
"LinkOUID: " + objArray[2].ToString());
                                    continue;
                                }
                            default:
                                Console.WriteLine("[ERROR UPDATE DOMAINS] - undefined
link type \nLinkOUID: " + objArray[2].ToString());
```

```
                                    Log.Logging(string.Format("ERROR   UPDATE   DOMAIN   -
StackTrace:undefined link type \\nLinkOUID: {0}\\n", (object)objArray[2].ToString()));
                                    continue;
                                }
                            }
                        }
                    }
                }
            }
```

Листинг А.2 — Класс, отвечающий за реализацию определения и объединения справочных-сущностей

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;

namespace ConsoleApp1
{
  public class SPR_Merge
  {
    public void Execute()
    {
      MergeNameSpr();
      List<string> list = Sql.ExecuteSqlDataSet("declare @tmp table(Table_Name
nvarchar(max))\r\n                                    insert into @tmp SELECT tbl.name FROM
sys.objects tbl\r\n                                     inner join
INFORMATION_SCHEMA.COLUMNS sch \r\n                                                on tbl.name =
sch.TABLE_NAME\r\n                                    WHERE tbl.type in (N'U') and
sch.COLUMN_NAME like 'GUID' and tbl.Name in (SELECT tbl.name FROM [result].sys.objects
tbl\r\n                                 inner join [result].INFORMATION_SCHEMA.COLUMNS
sch \r\n                                  on tbl.name = sch.TABLE_NAME\r\n
WHERE tbl.type in (N'U') and sch.COLUMN_NAME like 'GUID')\r\n
Declare @tbl_name nvarchar(max)\r\n                                                DECLARE @sqlCommand
nvarchar(1000)                                    \r\n
Declare @count float\r\n                                          Declare @sourceCount int\r\n
Declare @result table (Table_Name nvarchar(max), Percents float)\r\n
Declare cur cursor for select * from @tmp\r\n                                                  open
cur\r\n                                 FETCH NEXT FROM cur   \r\n
INTO @tbl_name\r\n                                    WHILE @@FETCH_STATUS = 0  \r\n
BEGIN\r\n                                   SET @sqlCommand = 'select @sourceCount =
Count(*) from ' + @tbl_name + '; if @sourceCount > 0 begin select @cnt = ((100 *
(select Count(*) from '+@tbl_name+' t1 inner join [result].dbo.'+@tbl_name+' t2 on
t1.guid = t2.guid)) / (@sourceCount)) end Else begin select @cnt = 0 end'\r\n
EXECUTE sp_executesql @sqlCommand, N'@sourceCount int,@cnt float OUTPUT',@sourceCount =
@sourceCount, @cnt = @count OUTPUT\r\n                                     INSERT INTO
@result select @tbl_name, @count\r\n                                   FETCH NEXT FROM cur
\r\n                                INTO @tbl_name\r\n
END\r\n                                   select * from @result where Percents > 90 and
Percents < 100 and (Table_Name not like '%cms%' and Table_Name not like '%sx%' and
```

```
Table_Name not like '%temp%')").Tables[0].AsEnumerable().Select<DataRow,
string>((Func<DataRow, string>) (t => t.ItemArray[0].ToString())).ToList<string>();
      Console.WriteLine("---Start Merge Spr\n");
      for (int index = 0; index < list.Count; ++index)
      {
        Console.WriteLine("---Merge Spr: " + (object) index + " \\ " + (object)
(list.Count - 1) + " - " + list[index]);
        MergeSpr(list[index]);
      }
    }

    public void MergeSpr(string table)
    {
      string pkColumn = Esrn_utils.getPKColumn(table);
      if (string.IsNullOrEmpty(pkColumn))
        return;
      string columnList = Esrn_utils.getColumnList(table);
      string sql_command = string.Format("SET IDENTITY_INSERT dbo.{0}
ON\r\n\t\t\t\t\t\t\t\tINSERT INTO {1}({2})\r\n\t\t\t\t\t\t\t\tSelect
{3}\r\n\t\t\t\t\t\t\t\tFROM [kadosh].dbo.{4}\r\n                                    Where
GUID not in (select GUID from {5})\r\n\t\t\t\t\t\t\t\tSET IDENTITY_INSERT dbo.{6} OFF",
(object) table, (object) table, (object) columnList, (object)
columnList.Replace(pkColumn, string.Format("{0} = ((select Max({1}) from {2}) +
ROW_NUMBER() over (order by {3}))", (object) pkColumn, (object) pkColumn, (object)
table, (object) pkColumn)), (object) table, (object) table, (object) table);
      try
      {
        Sql.ExecuteSql(sql_command);
      }
      catch (Exception ex1)
      {
        if (ex1.Message.Contains("identity"))
        {
          try
          {
            sql_command = string.Format("INSERT INTO {0}({1})\r\n\t\t\t\t\t\t\tSelect
{2}\r\n\t\t\t\t\t\t\tFROM [kadosh].dbo.{3}\r\n                                    Where
GUID not in (select GUID from {4})", (object) table, (object) columnList, (object)
columnList.Replace(pkColumn, string.Format("{0} = ((select Max({1}) from {2}) +
ROW_NUMBER() over (order by {3}))", (object) pkColumn, (object) pkColumn, (object)
table, (object) pkColumn)), (object) table, (object) table);
            Sql.ExecuteSql(sql_command);
          }
          catch (Exception ex2)
          {
            Console.WriteLine("[ERROR MERGE SPR WITH FIX IDENTITY] - " +
string.Format("StackTrace: {0}\\nSQL_Command: {1}\\nTable: {2}", (object) ex2.Message,
(object) sql_command, (object) table));
            Log.Logging(string.Format("ERROR MERGE SPR WITH FIX IDENTITY - StackTrace:
{0}{1}SQL_Command: {2}{3}\\nTable: {4}{5}", (object) ex2.Message, (object)
Environment.NewLine, (object) sql_command, (object) Environment.NewLine, (object)
table, (object) Environment.NewLine));
          }
        }
        else
        {
          Console.WriteLine("[ERROR MERGE SPR] - " + string.Format("StackTrace:
{0}\\nSQL_Command: {1}\\nTable: {2}\\n", (object) ex1.Message, (object) sql_command,
(object) table));
          Log.Logging(string.Format("ERROR MERGE SPR WITH FIX IDENTITY - StackTrace:
{0}{1}SQL_Command: {2}{3}\\nTable: {4}{5}", (object) ex1.Message, (object)
```

```
Environment.NewLine, (object) sql_command, (object) Environment.NewLine, (object)
table, (object) Environment.NewLine));
        }
      }
    }

    private void MergeNameSpr()
    {
      Console.WriteLine("[Merge Spr] - Merge FIO SPR");
      Sql.ExecuteSql("SET IDENTITY_INSERT dbo.SPR_FIO_SURNAME ON\r\n            INSERT
INTO SPR_FIO_SURNAME(OUID, A_NAME, COMMON_ACCESS, TS, A_DATIVESURNAME,
A_GENITIVESURNAME, A_ACCUSATSURNAME, A_ABLATIVESURNAME, A_PREPOSITSURNAME,
A_SYSTEMCLASS, A_CROWNER, A_EDITOR, A_CREATEDATE)\r\n            Select OUID = ((select
Max(OUID) from SPR_FIO_SURNAME) + ROW_NUMBER() over (order by OUID)), A_NAME,
COMMON_ACCESS, TS, A_DATIVESURNAME, A_GENITIVESURNAME, A_ACCUSATSURNAME,
A_ABLATIVESURNAME, A_PREPOSITSURNAME, A_SYSTEMCLASS, A_CROWNER, A_EDITOR,
A_CREATEDATE\r\n            FROM [temp].dbo.SPR_FIO_SURNAME\r\n            Where A_NAME
not in (select A_NAME from SPR_FIO_SURNAME)\r\n            SET IDENTITY_INSERT
dbo.SPR_FIO_SURNAME OFF\r\n\r\n            SET IDENTITY_INSERT dbo.SPR_FIO_SECONDNAME
ON\r\n            INSERT INTO SPR_FIO_SECONDNAME(OUID, A_NAME, COMMON_ACCESS, TS,
A_DATIVESECONDNAME, A_GENITIVESECNAME, A_ACCUSSECNAME, A_ABLATSECNAME, A_PREPSECNAME,
A_SYSTEMCLASS, A_CROWNER, A_EDITOR, A_CREATEDATE)\r\n            Select OUID = ((select
Max(OUID) from SPR_FIO_SECONDNAME) + ROW_NUMBER() over (order by OUID)), A_NAME,
COMMON_ACCESS, TS, A_DATIVESECONDNAME, A_GENITIVESECNAME, A_ACCUSSECNAME,
A_ABLATSECNAME, A_PREPSECNAME, A_SYSTEMCLASS, A_CROWNER, A_EDITOR, A_CREATEDATE\r\n
FROM [temp].dbo.SPR_FIO_SECONDNAME\r\n            Where A_NAME not in (select A_NAME
from SPR_FIO_SECONDNAME)\r\n            SET IDENTITY_INSERT dbo.SPR_FIO_SECONDNAME
OFF\r\n\r\n            SET IDENTITY_INSERT dbo.SPR_FIO_NAME ON\r\n            INSERT
INTO SPR_FIO_NAME(OUID, A_NAME, COMMON_ACCESS, TS, A_DATIVENAME, A_GENITIVENAME,
A_ACCUSATIVENAME, A_ABLATIVENAME, A_PREPOSITNAME, A_SYSTEMCLASS, A_CROWNER, A_EDITOR,
A_CREATEDATE, A_TYPE_DECLINATION)\r\n            Select OUID = ((select Max(OUID) from
SPR_FIO_NAME) + ROW_NUMBER() over (order by OUID)), A_NAME, COMMON_ACCESS, TS,
A_DATIVENAME, A_GENITIVENAME, A_ACCUSATIVENAME, A_ABLATIVENAME, A_PREPOSITNAME,
A_SYSTEMCLASS, A_CROWNER, A_EDITOR, A_CREATEDATE, A_TYPE_DECLINATION\r\n
FROM [temp].dbo.SPR_FIO_NAME\r\n            Where A_NAME not in (select A_NAME from
SPR_FIO_NAME)\r\n            SET IDENTITY_INSERT dbo.SPR_FIO_NAME OFF");
      string sql_command = string.Format("UPDATE wpc\r\n                            SET
wpc.A_NAME = sfn.OUID, wpc.A_SECONDNAME = sfs.OUID, wpc.SURNAME = sfs1.OUID\r\n
FROM WM_PERSONAL_CARD wpc\r\n                            INNER JOIN
[kadosh].dbo.WM_PERSONAL_CARD kad_wpc\r\n                            ON kad_wpc.OUID =
wpc.OUID - {0}\r\n                            INNER JOIN [kadosh].dbo.SPR_FIO_NAME
kad_sfn\r\n                            ON kad_wpc.A_NAME = kad_sfn.OUID\r\n
INNER JOIN [kadosh].dbo.SPR_FIO_SECONDNAME kad_sfs\r\n                            ON
kad_wpc.A_SECONDNAME = kad_sfs.OUID\r\n                            INNER JOIN
[kadosh].dbo.SPR_FIO_SURNAME kad_sfs1\r\n                            ON
kad_wpc.SURNAME = kad_sfs1.OUID\r\n                            INNER JOIN
SPR_FIO_NAME sfn\r\n                            ON kad_sfn.A_NAME = sfn.A_NAME\r\n
INNER JOIN SPR_FIO_SECONDNAME sfs\r\n                            ON kad_sfs.A_NAME =
sfs.A_NAME\r\n                            INNER JOIN SPR_FIO_SURNAME sfs1\r\n
ON kad_sfs1.A_NAME = sfs1.A_NAME\r\n                            WHERE wpc.OUID > {1}",
(object) Esrn_utils.getDegree("WM_PERSONAL_CARD"), (object)
Esrn_utils.getDegree("WM_PERSONAL_CARD"));
      Console.WriteLine("[Merge Spr] - Fix WM_PERSONAL_CARD Link on FIO");
      Sql.ExecuteSql(sql_command);
    }
  }
}
```

Листинг А.3 — Класс, отвечающий за реализацию основных функции по работе с базой данных

```
using System;
using System.Data;
using System.Data.SqlClient;

namespace ConsoleApp1
{
  public static class Sql
  {
    public static string ConnectionString = "Data Source=esrn-insar;Initial
Catalog=result;Integrated Security=True";

    public static DataSet ExecuteSqlDataSet(string sql_command)
    {
      DataSet dataSet = new DataSet();
      using (SqlConnection selectConnection = new SqlConnection(Sql.ConnectionString))
      {
        SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(sql_command,
selectConnection);
        sqlDataAdapter.SelectCommand.CommandTimeout = 0;
        sqlDataAdapter.Fill(dataSet);
      }
      return dataSet;
    }

    public static void ExecuteSql(string sql_command)
    {
      DataSet dataSet = new DataSet();
      using (SqlConnection sqlConnection = new SqlConnection(Sql.ConnectionString))
      {
        sqlConnection.Open();
        SqlCommand command = sqlConnection.CreateCommand();
        command.CommandText = sql_command;
        command.CommandTimeout = 0;
        command.Transaction = sqlConnection.BeginTransaction();
        try
        {
          command.ExecuteNonQuery();
          command.Transaction.Commit();
          sqlConnection.Close();
        }
        catch (Exception ex)
        {
          if (command.Transaction != null)
            command.Transaction.Rollback();
          sqlConnection.Close();
          throw ex;
        }
      }
    }
  }
}
```

Листинг А.4 — Класс, отвечающий за реализацию утилитарных функции для алгоритмов объединения

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;

namespace ConsoleApp1
{
    public static class Esrn_utils
    {
        public static string getColumnList(string table)
        {
            List<string> list = Sql.ExecuteSqlDataSet(string.Format("select COLUMN_NAME
from information_schema.columns where table_name = '{0}'",
(object)table)).Tables[0].AsEnumerable().Select<DataRow, string>((Func<DataRow,
string>)(t => t.ItemArray[0].ToString())).ToList<string>();
            string str1 = "";
            foreach (string str2 in list)
                str1 += str2 == list[list.Count - 1] ? str2 : str2 + ", ";
            return str1;
        }

        public static string getPKColumn(string table)
        {
            return Sql.ExecuteSqlDataSet(string.Format("select COLUMN_NAME from
INFORMATION_SCHEMA.KEY_COLUMN_USAGE where TABLE_NAME = '{0}' and CONSTRAINT_NAME like
'%PK%'", (object)table)).Tables[0].AsEnumerable().Select<DataRow,
string>((Func<DataRow, string>)(t =>
t.ItemArray[0].ToString())).FirstOrDefault<string>();
        }

        public static int getDegree(string table)
        {
            string sql_command = string.Format("select POWER(10, (select
LEN(cast(Max({0}) AS NVARCHAR(MAX))) from [insar_mg].dbo.{1}))",
(object)Esrn_utils.getPKColumn(table), (object)table);
            try
            {
                return
Sql.ExecuteSqlDataSet(sql_command).Tables[0].AsEnumerable().Select<DataRow,
int>((Func<DataRow, int>)(t => (int)t.ItemArray[0])).FirstOrDefault<int>();
            }
            catch (Exception ex)
            {
                throw new Exception(ex.Message + "\nSqlCommand: " + sql_command);
            }
        }

        public static bool isObject(string table)
        {
            return !string.IsNullOrEmpty(Sql.ExecuteSqlDataSet(string.Format("SELECT *
FROM sys.objects WHERE name LIKE '{0}'",
(object)table)).Tables[0].AsEnumerable().Select<DataRow, string>((Func<DataRow,
string>)(t => t.ItemArray[0].ToString())).FirstOrDefault<string>());
        }

        public static bool isObjectContains(string table)
        {
```

```
            return Sql.ExecuteSqlDataSet(string.Format("SELECT Count(*) FROM {0}",
(object)table)).Tables[0].AsEnumerable().Select<DataRow, int>((Func<DataRow, int>)(t =>
(int)t.ItemArray[0])).FirstOrDefault<int>() != 0;
        }

        public static void checkCount(string table)
        {
            Console.WriteLine("[Check Count] - " + table);
            Console.WriteLine();
            int num = Sql.ExecuteSqlDataSet(string.Format("Select Count(*) from {0}",
(object)table)).Tables[0].AsEnumerable().Select<DataRow, int>((Func<DataRow, int>)(t =>
(int)t.ItemArray[0])).ToList<int>()[0];
            if (Sql.ExecuteSqlDataSet(string.Format("Select Count(*) from
[insar_mg].dbo.{0}", (object)table)).Tables[0].AsEnumerable().Select<DataRow,
int>((Func<DataRow, int>)(t => (int)t.ItemArray[0])).ToList<int>()[0] +
Sql.ExecuteSqlDataSet(string.Format("Select Count(*) from [kadosh].dbo.{0}",
(object)table)).Tables[0].AsEnumerable().Select<DataRow, int>((Func<DataRow, int>)(t =>
(int)t.ItemArray[0])).ToList<int>()[0] == num)
                return;
            Console.WriteLine("[Check Count] - MERGE");
            Console.WriteLine();
            string pkColumn = Esrn_utils.getPKColumn(table);
            string columnList = Esrn_utils.getColumnList(table);
            string str = string.Format("SET IDENTITY_INSERT dbo.{0} ON INSERT INTO
{1}({2}) Select {3} FROM [kadosh].dbo.{4} SET IDENTITY_INSERT dbo.{5} OFF",
(object)table, (object)table, (object)columnList, (object)columnList.Replace(pkColumn,
pkColumn + " = " + pkColumn + string.Format(" + POWER(10,(select LEN(cast(Max({0}) AS
NVARCHAR(MAX))) from [insar_mg].dbo.{1}))", (object)pkColumn, (object)table)),
(object)table, (object)table);
            try
            {
                Sql.ExecuteSql(str);
                Log.Logging(str);
            }
            catch (Exception ex1)
            {
                if (ex1.Message.Contains("identity"))
                {
                    try
                    {
                        str = string.Format(" INSERT INTO {0}({1}) Select {2} FROM
[temp].dbo.{3}", (object)table, (object)columnList,
(object)columnList.Replace(pkColumn, pkColumn + " = " + pkColumn + string.Format(" +
POWER(10,(select LEN(cast(Max({0}) AS NVARCHAR(MAX))) from [insar_mg].dbo.{1}))",
(object)pkColumn, (object)table)), (object)table);
                        Sql.ExecuteSql(str);
                        Log.Logging("[Check Count] - " + str);
                    }
                    catch (Exception ex2)
                    {
                        Log.Logging("[Check Count ERROR] - " + str +
Environment.NewLine + "Stack Trace - " + ex1.Message);
                        Console.WriteLine(ex2.Message);
                        Console.WriteLine();
                    }
                }
                else
                {
                    Log.Logging("[Check Count ERROR] - " + str + Environment.NewLine +
"Stack Trace - " + ex1.Message);
                    Console.WriteLine(ex1.Message);
```

```
                    Console.WriteLine();
                }
            }
        }
    }
}
```

Листинг А.5 — Vue.js компонент клиентского интерфейса

```
<template>
  <v-stepper v-model="step">
    <v-stepper-header>
      <v-stepper-step :complete="step > 1" step="1">Конфигурация подключения</v-
stepper-step>
      <v-divider></v-divider>
      <v-stepper-step :complete="step > 2" step="2">Список исключения</v-stepper-
step>
      <v-divider></v-divider>
      <v-stepper-step :complete="step > 3" step="3">Параметры алгоритма</v-
stepper-step>
      <v-divider></v-divider>
      <v-stepper-step :complete="step > 4" step="4">Sql скрипты</v-stepper-step>
      <v-divider></v-divider>
      <v-stepper-step :complete="step > 5" step="5">Выполнение объединения</v-
stepper-step>
      <v-divider></v-divider>
      <v-stepper-step step="6">Лог выполнения</v-stepper-step>
    </v-stepper-header>

    <v-stepper-items>
      <v-stepper-content step="1">
        <v-card>
          <v-card-title>Конфигурация подключения</v-card-title>
          <v-card-subtitle
            >Необходино задать строки подключения к исходной и целевой бд</v-card-
subtitle
          >
          <v-card-text>
            <v-form v-model="connectionModelValid" ref="connection-form">
              <v-row>
                <v-col cols="12">
                  <v-text-field
```

```
                label="Исходная бд"
                required
                :rules="[$rules.NotNull]"
                :counter="255"
                v-model="connectionSource"
              ></v-text-field>
            </v-col>
            <v-col cols="12">
              <v-text-field
                label="Целевая бд"
                required
                :rules="[$rules.NotNull]"
                :counter="255"
                v-model="connectionTarget"
              ></v-text-field>
            </v-col>
          </v-row>
        </v-form>
      </v-card-text>
      <v-card-actions>
        <v-btn color="primary" @click="step += 1">
          Далее
        </v-btn>
      </v-card-actions>
    </v-card>
  </v-stepper-content>

  <v-stepper-content step="2">
    <v-card>
      <v-card-title>Список исключения</v-card-title>
      <v-card-subtitle
        >Можно задать список таблиц, которые будут исключены из алгоритма
        объединения</v-card-subtitle
      >
      <v-card-text>
        <v-row>
          <v-col cols="12">
            <v-textarea
              outlined
              label="Список исключения"
              v-model="exludedTable"
```

```
                        hint="Введите названия таблиц через запятую"
                      ></v-textarea>
                    </v-col>
                  </v-row>
                </v-card-text>
                <v-card-actions>
                  <v-btn text @click="step -= 1">Назад</v-btn>
                  <v-btn color="primary" @click="step += 1">
                    Далее
                  </v-btn>
                </v-card-actions>
              </v-card>
            </v-stepper-content>

            <v-stepper-content step="3">
              <v-card>
                <v-card-title>Параметры алгоритма</v-card-title>
                <v-card-subtitle>Здесь можно задать параметры алгоритма объединения</v-
card-subtitle>
                <v-card-text>
                  <v-form v-model="parametrsModelValid" ref="parametrs-form">
                    <v-row>
                      <v-col cols="12">
                        <v-text-field
                          label="Процентное соотношение справочных сущностей"
                          required
                          :rules="[$rules.NotNull, $rules.Number]"
                          :counter="3"
                          v-model="sprPercent"
                        ></v-text-field>
                      </v-col>
                      <v-col cols="12">
                        <v-text-field
                          label="Степень увеличение первичного ключа "
                          required
                          :rules="[$rules.NotNull, $rules.Number]"
                          :counter="8"
                          v-model="pkDegree"
                        ></v-text-field>
                      </v-col>
                    </v-row>
```

```
            </v-form>
          </v-card-text>
          <v-card-actions>
            <v-btn text @click="step -= 1">Назад</v-btn>
            <v-btn color="primary" @click="step += 1">
              Далее
            </v-btn>
          </v-card-actions>
        </v-card>
      </v-stepper-content>

      <v-stepper-content step="4">
        <v-card>
          <v-card-title>SQl скрипт</v-card-title>
          <v-card-subtitle
            >Здесь можно задать sql скрипт, которые будут выполнены после
объединения
            бд</v-card-subtitle
          >
          <v-card-text>
            <v-row>
              <v-col cols="12">
                <v-textarea
                  outlined
                  label="SQL скрипт"
                  v-model="sqlScript"
                  hint="Введите sql скрипт"
                ></v-textarea>
              </v-col>
            </v-row>
          </v-card-text>
          <v-card-actions>
            <v-btn text @click="step -= 1">Назад</v-btn>
            <v-btn color="primary" @click="step += 1">
              Далее
            </v-btn>
          </v-card-actions>
        </v-card>
      </v-stepper-content>

      <v-stepper-content step="5">
```

```
                <v-card>
                  <v-card-title>Выполнение алгоритма объединения</v-card-title>
                  <v-card-subtitle
                    >Идет процесс выполнения алгоритма объединения бд. Пожалуйста,
                    подождите</v-card-subtitle
                  >
                  <v-card-text>
                    <v-progress-linear value="63" height="25" rounded color="red">
                      <template v-slot="{ value }">
                        <strong>{{ Math.ceil(value) }}%</strong>
                      </template>
                    </v-progress-linear>
                  </v-card-text>
                  <v-card-actions>
                    <v-btn text @click="step -= 1">Назад</v-btn>
                    <v-btn color="primary" @click="step += 1">
                      Далее
                    </v-btn>
                  </v-card-actions>
                </v-card>
              </v-stepper-content>
              <v-stepper-content step="6">
                <v-card>
                  <v-card-title>Лог выполнения</v-card-title>
                  <v-card-subtitle>Лог выполнения алгоритма объединения бд</v-card-
subtitle>
                  <v-btn color="red" dark
                    ><v-icon>mdi-download</v-icon>
                    Скачать
                  </v-btn>
                  <v-card-actions>
                    <v-btn text @click="step -= 1">Назад</v-btn>
                  </v-card-actions>
                </v-card>
              </v-stepper-content>
            </v-stepper-items>
          </v-stepper>
        </template>

        <script>
        export default {
```

```
  data() {
    return {
      step: 1,
      connectionModelValid: false,
      connectionSource: null,
      connectionTarget: null,
      exludedTable: null,
      parametrsModelValid: false,
      sprPercent: 90,
      pkDegree: 10000,
      sqlScript: null
    };
  },
  methods: {}
};
</script>

<style>
</style>
```

Листинг А.6 — JavaScript модуль обработки http запросов

```
import axios from 'axios'
export default {
    install(Vue) {
        var result = {
            post: (url, data) => {
                return new Promise((resolve, reject) => {
                    axios.post(url, data)
                        .then((response) => { resolve(response.data) })
                        .catch((error) => { reject(error) });
                })
            },
            get: (url, data) => {
                return new Promise((resolve, reject) => {
                    axios.get(url, {
                        params: data
                    })
                        .then((response) => { resolve(response.data) })
                        .catch((error) => { reject(error) });
```

```
                    })
                },
                delete: (url, data) => {
                    return new Promise((resolve, reject) => {
                        axios.delete(url, {
                            params: data
                        })
                            .then((response) => { resolve(response.data) })
                            .catch((error) => { reject(error) });
                    })
                }
            };
            Vue.prototype.$http = result;
        }
    }
```

Листинг А.7 — JavaScript модуль по обработке валидации полей формы

```
export default {
    install(Vue) {
        var result = {
            NotNull: (value) => {
                if (!value && value !== 0) {
                    return "данное поле обязательно для заполнения"
                }
                return true;
            },
            Number: (value) => {
                if (!!value) {
                    if (/^\d+$/.test(value)) {
                        return true
                    }
                    return "неверно заполнено поле"
                }
                return true;
            },
            Phone: (value) => {
                if (!!value) {
                    if        (!(/\(?([0-9]{3})\)?([        .-]?)([0-9]{3})\2([0-
9]{4})/.test(value))) {
```

```
                                return "неверно заполнено поле"
                            }
                            return true;
                        } else {
                            return true;
                        }
                    },
                    Summ: (value) => {
                        if (!!value) {
                            if                          (!(/^([0-9]{1,8}[\.][0-9]{2})$|^([0-
9]{1,8})$/g.test(value))) {
                                return "неверно заполнено поле"
                            }
                            return true;
                        }
                        return true
                    }
                };
                Vue.prototype.$rules = result;
            }
        }
```

Листинг А.8 — ASP.Net Core Controller основных представлений

```
using System;
using System.Collections.Generic;
using System.Security.Claims;
using System.Threading.Tasks;
using Data;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Logging;


namespace WebApplication.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;
        private readonly IWebHostEnvironment environment;
        private readonly DataBaseContext context;

        public class LoginModel
```

```
        {
            public string Login { get; set; }
            public string Password { get; set; }
        }
        public HomeController(ILogger<HomeController> logger, IWebHostEnvironment
environment, DataBaseContext context)
        {
            _logger = logger;
            this.environment = environment;
            this.context = context;
        }
        [Authorize]
        public IActionResult Index()
        {
            var encoding = new System.Text.UTF8Encoding();
            var htm = System.IO.File.ReadAllText(environment.ContentRootPath +
"/wwwroot/index.html", encoding);
            byte[] data = encoding.GetBytes(htm);
            return File(data, "text/html");

        }
        [AllowAnonymous]
        public IActionResult Login()
        {
            var encoding = new System.Text.UTF8Encoding();
            var htm = System.IO.File.ReadAllText(environment.ContentRootPath +
"/wwwroot/login.html", encoding);
            byte[] data = encoding.GetBytes(htm);
            return File(data, "text/html");
        }
        [HttpPost]
        public async Task<IActionResult> Login([FromBody]LoginModel model)
        {

            try
            {
                if (User.Identity.IsAuthenticated)
                {
                    return Redirect("/Home/Index");
                }
                var user = await context.Users.AsNoTracking().FirstOrDefaultAsync(x =>
x.Login == model.Login && x.Password == model.Password);
                if (user == null)
                {
                    return BadRequest($"Ошибка авторизации. Неверный логин или
пароль.");
                }
                var claims = new List<Claim>
                {
                    new Claim(ClaimsIdentity.DefaultNameClaimType, user.Id.ToString())
                };
                ClaimsIdentity claimsIdentity = new ClaimsIdentity(claims,
"ApplicationCookie", ClaimsIdentity.DefaultNameClaimType,
ClaimsIdentity.DefaultRoleClaimType);
                await
HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new
ClaimsPrincipal(claimsIdentity));
                return Ok();
            }
            catch (Exception ex)
            {
```

```
                return BadRequest($"Ошибка авторизации. {ex.Message}");
            }
        }
    }
}
```

Листинг А.9 — .Net Core модуль обработки данных передаваемых с сервера на клиент с помощью рефлексии

```
using System;
using System.Linq;

namespace WebApplication
{
    public static class Mapper
    {

        public static DataTransferObject<T> Map<T>(this T root) where T : class
        {
            try
            {
                var result = new DataTransferObject<T>(root);
                if(root != null)
                {
                    var fields = root.GetType().GetProperties().Where(x =>
!x.GetMethod.IsVirtual).ToList();
                    foreach (var field in fields)
                    {
                        result.AddField(field.Name, field.GetValue(root));
                    }
                }
                return result;
            }
            catch (Exception ex)
            {
                throw new Exception($"[DataTransferMapper] - Error in Dynamic map
object. Message: {ex.Message}", ex);
            }
        }
        public static DataTransferObject<T> Include<T>(this DataTransferObject<T> obj,
Func<T, object> f, string name) where T : class
        {
            try
            {
                var new_obj = f(obj.BaseValue);
                var dto = new_obj.Map();
                obj.AddField(name, dto.Value == null ? null : dto.Value);
                return obj;
            }
            catch (Exception ex)
            {
                throw new Exception($"[DataTransferMapper] - Error in Dynamic include
object. Message: {ex.Message}", ex);
            }
        }
        public static DataTransferObject<T> Add<T>(this DataTransferObject<T> obj,
Func<T, object> f, string name) where T : class
```

```
        {
            try
            {
                var value = f(obj.BaseValue);
                obj.AddField(name, value);
                return obj;
            }
            catch (Exception ex)
            {
                throw new Exception($"[DataTransferMapper] - Error in Dynamic include
object. Message: {ex.Message}", ex);
            }
        }
        public static object Get<T>(this DataTransferObject<T> obj) where T : class
        {
            try
            {
                return obj.Value;
            }
            catch (Exception ex)
            {
                throw new Exception($"[DataTransferMapper] - Error in Dynamic get
object. Message: {ex.Message}", ex);
            }
        }

    }
}
```

Листинг А.10 — JavaScript модуль работы с датами

```
import { format } from 'date-fns';
export default {
    install(Vue) {
        var utils = {
            convert: function (date, withTime) {
                if (!!date) {
                    if (date instanceof Date) {
                        return format(date, !!withTime ? "dd.MM.yyyy HH:mm" :
"dd.MM.yyyy");
                    } else {
                        return format(this.parse(date), !!withTime ? "dd.MM.yyyy
HH:mm" : "dd.MM.yyyy");
                    }
                }
                return "";
            },
            send: function (date) {
                if (!!date) {
```

```
                        if (date instanceof Date) {
                            return format(date, "yyyy-MM-dd'T'HH:mm:ss+03:00");
                        } else {
                            return format(this.parse(date), "yyyy-MM-
dd'T'HH:mm:ss+03:00");
                        }
                    }
                    return date;
                },
                parse: function (date) {
                    if (date instanceof Date) {
                        return date;
                    }
                    if (!!date) {
                        var parts = date.split('T');
                        var result_date = new Date(parts[0]);
                        var time = parts[1];
                        var _hours = time.split(':')[0];
                        var _minutes = time.split(':')[1];
                        result_date.setHours(_hours);
                        result_date.setMinutes(_minutes);
                        return result_date;
                    }
                    return "";
                }
            };
            Vue.prototype.$date = utils;
        }
    }
```

Листинг А.11 — Vue.js компонент для загрузки и скачивания файлов

```
    <template>
      <div class="upload">
        <div class="upload_interface" v-show="file === null">
          <input ref="upload_input" type="file" @change="inputChange"
:accept="exstension" />
          <Button type="text" icon="md-attach" @click="open"
:disabled="disabled">Выбрать файл</Button>
        </div>
```

```
    <div class="upload_list" v-if="!!file">
      <div>
        <div>
          <a @click.prevent="download">{{ file.fileName }}</a>
        </div>
        <a @click.prevent="removeFile" style="margin-left: .5em">Удалить</a>
      </div>
    </div>
  </div>
</template>
<script>
import b64toBlob from "b64-to-blob";
import { saveAs } from "file-saver";
import axios from "axios";
export default {
  props: {
    disabled: {
      type: Boolean,
      default: false
    },
    exstension: {
      type: String,
      default: ""
    },
    value: {
      type: Object
    },
    downloadProps: {
      type: Object
    },
    maxSize: {
      type: Number,
      default: 104857600
    }
  },
  data() {
    return {
      file: null
    };
  },
  methods: {
```

```
    open() {
      this.$refs["upload_input"].click();
    },
    inputChange() {
      if (this.$refs["upload_input"].files[0].size > this.maxSize) {
        this.$Notice.error({
          title: "Ошибка прикрепления файла",
          desc:
            "Максимальный размер прикрепленного файла не должен превышать " +
            this.maxSize / 1024 / 1024 +
            " МБ."
        });
      } else {
        this.getBase64(this.$refs["upload_input"].files[0]).then(res => {
          this.file = {
            fileName: this.$refs["upload_input"].files[0].name,
            source: res
          };
          this.$emit("input", this.file);
        });
      }
    },
    removeFile() {
      this.file = null;
      this.$refs["upload_input"].value = "";
      this.$emit("input", this.file);
    },
    getBase64(file) {
      return new Promise((resolve, reject) => {
        const reader = new FileReader();
        reader.readAsDataURL(file);
        reader.onload = () =>
          resolve(reader.result.substring(reader.result.indexOf(";base64") + 8));
        reader.onerror = error => reject(error);
      });
    },
    download() {
      if (!!this.downloadProps) {
        window.location.href                                                    =
`/api/File/GetFile?source=${this.downloadProps.source}&id=${this.downloadProps.id}&file
Name=${this.file.fileName}`;
```

```
      } else {
        saveAs(b64toBlob(this.file.source), this.file.fileName);
      }
    }
  },
  mounted() {
    this.$nextTick(() => {
      this.file = this.value;
    });
  },
  watch: {
    value: function() {
      this.$nextTick(() => {
        this.file = this.value;
      });
    }
  }
};
</script>
<style scoped>
.upload {
  display: flex;
  width: 100%;
  height: 100%;
  flex-direction: column;
  justify-content: flex-start;
  align-items: flex-start;
  border: 1px dashed #c5c8ce;
  padding: 1em;
  background: #e8eaec;
}
.upload_interface {
  display: flex;
  justify-content: center;
  align-items: center;
  width: 100%;
}
.upload_interface input {
  display: none;
}
.upload_list {
```

```
  display: flex;
  width: 100%;
  flex-direction: column;
  justify-content: flex-start;
  align-items: flex-start;
}
.upload_list > div {
  display: flex;
  width: 100%;
  justify-content: space-between;
  align-items: center;
}
.upload_list > div > div {
  display: flex;
  justify-content: flex-start;
  align-items: center;
  width: 80%;
}
.upload_list > div > div > a {
  overflow: hidden;
  text-overflow: ellipsis;
}
</style>
```

Листинг А.12 — .Net Core Controller обработки загрузки и скачивания файлов

```
using System;
using System.IO;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Reflection;
using System.Text;
using System.Threading.Tasks;
using System.Web;
using System.Web.Http;
using Business;
```

```
using Client.Controllers.Utils;

namespace Client.Controllers
{
    [Authorize]
    public class FileController : ApiController
    {
        [HttpGet]
        public HttpResponseMessage GetFile(string source, int id, string fileName)
        {
            try
            {
                switch (source)
                {
                    case "jm":
                        {
                            byte[] file;
                            using (var authCore = new Business.Authorization())
                            {
                                var user = authCore.GetAuthorizedUser(int.Parse(User.Identity.Name));
                                using (var core = new Core(user))
                                {
                                    file = core.GetAudioProtocol(id);
                                }
                            }
                            HttpResponseMessage result = Request.CreateResponse(HttpStatusCode.OK);
                            result.Content = new ByteArrayContent(file);
                            result.Content.Headers.ContentLength = file.LongLength;
                            result.Content.Headers.ContentDisposition = new ContentDispositionHeaderValue("attachment")
                            {
                                FileName = fileName
                            };
                            result.Content.Headers.ContentType = new MediaTypeHeaderValue(MimeMapping.GetMimeMapping(fileName));
                            return result;
                        }
                    case "doc":
```

```
                {
                    byte[] file;
                    using (var authCore = new Business.Authorization())
                    {
                        var user = authCore.GetAuthorizedUser(int.Parse(User.Identity.Name));
                        using (var core = new Core(user))
                        {
                            file = core.GetDocument(id).text;
                        }
                    }
                    HttpResponseMessage result = Request.CreateResponse(HttpStatusCode.OK);
                    result.Content = new ByteArrayContent(file);
                    result.Content.Headers.ContentLength = file.LongLength;
                    result.Content.Headers.ContentDisposition = new ContentDispositionHeaderValue("attachment")
                    {
                        FileName = fileName
                    };
                    result.Content.Headers.ContentType = new MediaTypeHeaderValue(MimeMapping.GetMimeMapping(fileName));
                    return result;
                }
            case "doc-uf":
                {
                    byte[] file;
                    using (var authCore = new Business.Authorization())
                    {
                        var user = authCore.GetAuthorizedUser(int.Parse(User.Identity.Name));
                        using (var core = new Core(user))
                        {
                            file = core.GetDocument(id).uftext;
                        }
                    }
                    HttpResponseMessage result = Request.CreateResponse(HttpStatusCode.OK);
                    result.Content = new ByteArrayContent(file);
```

```
                                result.Content.Headers.ContentLength                =
file.LongLength;
                                result.Content.Headers.ContentDisposition    =    new
ContentDispositionHeaderValue("attachment")
                                {
                                    FileName = fileName
                                };
                                result.Content.Headers.ContentType       =       new
MediaTypeHeaderValue(MimeMapping.GetMimeMapping(fileName));
                                return result;
                            }
                        case "execution-list":
                            {
                                byte[] file;
                                using (var authCore = new Business.Authorization())
                                {
                                    var                    user                    =
authCore.GetAuthorizedUser(int.Parse(User.Identity.Name));
                                    using (var core = new Core(user))
                                    {
                                        var                  data                  =
core.GetReportFile(core.GetReportExecutionList(id).GetType().GetProperties(BindingFlags
.Instance | BindingFlags.Public)
                                           .ToDictionary(prop   =>   prop.Name,   prop   =>
prop.GetValue(core.GetReportExecutionList(id), null)?.ToString()), false);
                                        file = data.File;
                                        fileName = data.Name;
                                    }
                                }
                                HttpResponseMessage             result              =
Request.CreateResponse(HttpStatusCode.OK);
                                result.Content = new ByteArrayContent(file);
                                result.Content.Headers.ContentLength                =
file.LongLength;
                                result.Content.Headers.ContentDisposition    =    new
ContentDispositionHeaderValue("attachment")
                                {
                                    FileName = fileName
                                };
                                result.Content.Headers.ContentType       =       new
MediaTypeHeaderValue(MimeMapping.GetMimeMapping(fileName));
```

```
                                return result;
                            }
                        case "material-portal-file":
                            {
                                byte[] file;
                                using (var authCore = new Business.Authorization())
                                {
                                    var user = authCore.GetAuthorizedUser(int.Parse(User.Identity.Name));
                                    using (var core = new Core(user))
                                    {
                                        var data = core.GetApplicationFile(id);
                                        file = data.content;
                                        fileName = data.filename;
                                    }
                                }
                                HttpResponseMessage result = Request.CreateResponse(HttpStatusCode.OK);
                                result.Content = new ByteArrayContent(file);
                                result.Content.Headers.ContentLength = file.LongLength;
                                result.Content.Headers.ContentDisposition = new ContentDispositionHeaderValue("attachment")
                                {
                                    FileName = fileName
                                };
                                result.Content.Headers.ContentType = new MediaTypeHeaderValue(MimeMapping.GetMimeMapping(fileName));
                                return result;
                            }
                        default:
                            {
                                throw new Exception("Error in get file: invalid parameters. Source: " + source + ". Id: " + id + ". File Name: " + fileName);
                            }
                    }
                }
                catch (Exception e)
                {
                    HttpResponseMessage result = new HttpResponseMessage(HttpStatusCode.BadRequest);
```

```
                result.Content = new StringContent(e.Message);
                return result;
            }
        }
        [HttpGet]
        public IHttpActionResult GetDocumentHtml(int docId, string source)
        {
            try
            {
                using (var auth_core = new Business.Authorization())
                {
                    var                          user                          =
auth_core.GetAuthorizedUser(int.Parse(User.Identity.Name));
                    using (var core = new Core(user))
                    {
                        var doc = core.GetDocument(docId);
                        switch (source)
                        {
                            case "origin":
                                {
                                    if
(doc.path.Split('.')[doc.path.Split('.').Length - 1] != "html")
                                    {
                                        return
Ok(ConvertDocument.ConvertToHtml(doc.text,
doc.path.Split('.')[doc.path.Split('.').Length - 1]));
                                    }
                                    return Ok(Encoding.UTF8.GetString(doc.text));
                                }
                            case "uf":
                                {
                                    return
Ok(Encoding.UTF8.GetString(doc.uftext));
                                }
                            default:
                                {
                                    throw new Exception("Непривильный входные
параметры в контроллер. DocId: " + docId + ". Source: " + source);
                                }
                        }
                    }
```

```
                    }
                }
                catch (Exception ex)
                {
                    return BadRequest("Ошибка получения html документа. Id: " + docId
+ ". " + ex.Message);
                }
            }
            [HttpPost]
            public async Task<IHttpActionResult> ConvertDocumentToHtml()
            {
                try
                {
                    var provider = new MultipartMemoryStreamProvider();
                    await Request.Content.ReadAsMultipartAsync(provider);
                    string file = await provider.Contents[0].ReadAsStringAsync();
                    string              contentType              =              await
provider.Contents[1].ReadAsStringAsync();
                    return
Ok(ConvertDocument.ConvertToHtml(Convert.FromBase64String(file), contentType));
                }
                catch (Exception ex)
                {
                    return  BadRequest("Ошибка  преобразования  файла  в  html.  "  +
ex.Message);
                }
            }
            [HttpPost]
            public async Task<IHttpActionResult> AttachOriginDocToDocument()
            {
                try
                {
                    var provider = new MultipartMemoryStreamProvider();
                    await Request.Content.ReadAsMultipartAsync(provider);
                    string file = await provider.Contents[0].ReadAsStringAsync();
                    string fileName = await provider.Contents[1].ReadAsStringAsync();
                    string doc_id = await provider.Contents[2].ReadAsStringAsync();
                    string                          contentType                          =
fileName.Split('.')[fileName.Split('.').Length - 1];
                    byte[] resfile;
                    if (contentType != "html")
```

```
                {
                    file                                                    =
ConvertDocument.ConvertToHtml(Convert.FromBase64String(file), contentType);
                    using (MemoryStream stream = new MemoryStream())
                    {
                        using (StreamWriter writer = new StreamWriter(stream))
                        {
                            writer.Write(file);
                            writer.Flush();
                            stream.Position = 0;
                        }
                        resfile = stream.ToArray();
                        stream.Close();
                        var exstension = fileName.Split('.');
                        exstension[exstension.Length - 1] = "html";
                        fileName = string.Join(".", exstension);
                    }
                }
                else
                {
                    resfile = Convert.FromBase64String(file);
                }
                using (var auth_core = new Business.Authorization())
                {
                    var                             user                    =
auth_core.GetAuthorizedUser(int.Parse(User.Identity.Name));
                    using (var core = new Core(user))
                    {
                        var doc = core.GetDocument(int.Parse(doc_id));
                        doc.path = fileName;
                        doc.text = resfile;
                        core.EditDocument(doc);
                    }
                }
                return Ok();
            }
            catch (Exception ex)
            {
                return BadRequest("Ошибка прикрепления файла к документу. " +
ex.Message);
            }
```

```
        }
        [HttpPost]
        public async Task<IHttpActionResult> AttachUfDocToDocument()
        {
            try
            {
                var provider = new MultipartMemoryStreamProvider();
                await Request.Content.ReadAsMultipartAsync(provider);
                string file = await provider.Contents[0].ReadAsStringAsync();
                string fileName = await provider.Contents[1].ReadAsStringAsync();
                string doc_id = await provider.Contents[2].ReadAsStringAsync();
                using (var auth_core = new Business.Authorization())
                {
                    var                          user                          =
auth_core.GetAuthorizedUser(int.Parse(User.Identity.Name));
                    using (var core = new Core(user))
                    {
                        var doc = core.GetDocument(int.Parse(doc_id));
                        doc.pathuf = fileName;
                        doc.uftext = Convert.FromBase64String(file);
                        core.EditDocument(doc);
                    }
                }
                return Ok();
            }
            catch (Exception ex)
            {
                return  BadRequest("Ошибка  прикрепления  обезличенного  файла  к
документу. " + ex.Message);
            }
        }
        [HttpPost]
        public async Task<IHttpActionResult> AttachToJm()
        {
            try
            {
                var provider = new MultipartMemoryStreamProvider();
                await Request.Content.ReadAsMultipartAsync(provider);
                string file = await provider.Contents[0].ReadAsStringAsync();
                string fileName = await provider.Contents[1].ReadAsStringAsync();
```

```
                        int             jm_id             =             int.Parse(await
provider.Contents[2].ReadAsStringAsync());
                        using (var auth_core = new Business.Authorization())
                        {
                            var                           user                          =
auth_core.GetAuthorizedUser(int.Parse(User.Identity.Name));
                            using (var core = new Core(user))
                            {
                                var old_jm = core.GetJudicialMeeting(jm_id);
                                if (!String.IsNullOrEmpty(old_jm.audiofile))
                                {
                                    core.DeleteAudioProtocol(jm_id);
                                }
                                core.AddAudioProtocol(jm_id,
Convert.FromBase64String(file), fileName);
                            }
                        }
                        return Ok();
                    }
                    catch (Exception ex)
                    {
                        return BadRequest("Ошибка загрузки файла");
                    }

                }
                [HttpGet]
                public  HttpResponseMessage  GetUskToBsr(DateTime  dateStart,  DateTime
dateEnd)
                {
                    try
                    {
                        using (var auth_core = new Business.Authorization())
                        {
                            var                           user                          =
auth_core.GetAuthorizedUser(int.Parse(User.Identity.Name));
                            using (var core = new Core(user))
                            {
                                var export = core.ExportUskToBsr(dateStart, dateEnd);
                                using (MemoryStream stream = new MemoryStream())
                                {
                                    export.Save(stream);
```

```
                                    stream.Flush();
                                    stream.Position = 0;
                                    string    fileName    =    $"{user.JudgeArea.Vnkod}-
{dateEnd.Day.ToString().PadLeft(2,            '0')}{dateEnd.Month.ToString().PadLeft(2,
'0')}{dateEnd.Year}.xml";
                                    HttpResponseMessage             result             =
Request.CreateResponse(HttpStatusCode.OK);
                                    result.Content                  =                  new
ByteArrayContent(stream.ToArray());
                                    result.Content.Headers.ContentLength               =
stream.ToArray().LongLength;
                                    result.Content.Headers.ContentDisposition    =    new
ContentDispositionHeaderValue("attachment")
                                    {
                                        FileName = fileName
                                    };
                                    result.Content.Headers.ContentType       =       new
MediaTypeHeaderValue(MimeMapping.GetMimeMapping(fileName));
                                    return result;
                                }
                            }
                        }
                    }
                    catch (Exception ex)
                    {
                        HttpResponseMessage            result            =            new
HttpResponseMessage(HttpStatusCode.BadRequest);
                        result.Content  =  new  StringContent("Ошибка  получения  отчета
\"Экспорт УСК в БСР\". DateStart: " + dateStart.ToString("dd.MM.yyyy") + ". DateEnd: " +
dateEnd.ToString("dd.MM.yyyy") + ". Error: " + ex.Message);
                        return result;
                    }
                }
            }
        }
```

Листинг А.13 — JavaScript модуль конфигурации сборки Webpack

```
const $mode = process.env.NODE_ENV === 'production';
const $utils = require('./webpack.utils.js')($mode);
```

```
module.exports = {
    context: $utils.srcPath,
    entry: {
        polyfill: '@babel/polyfill',
        index: './index.js',
        login: './login.js'
    },
    output: {
        path: $utils.distPath,
        filename: $utils.filename('js'),
        publicPath: '/'
    },
    devtool: $utils.devTool(),
    plugins: $utils.plugins(),
    optimization: $utils.optimization(),
    resolve: {
        extensions: ['.js', '.vue'],
        alias: {
            '@c': $utils.srcPath + '/Components',
            '@': $utils.srcPath,
            'vue$': 'vue/dist/vue.esm.js'
        }
    },
    module: {
        rules: [{
            test: /\.css$/,
            use: $utils.cssLoaders()
        },
        {
            test: /\.less$/,
            use: $utils.cssLoaders('less')
        },
        {
            test: /\.(png|jpg|svg|gif|ttf|woff|woff2|eot)$/,
            use: ['file-loader']
        },
        {
            test: /\.js$/,
            exclude: /node_modules/,
            use: $utils.jsLoaders()
        },
        {
            test: /\.vue$/,
            use: ['vue-loader'],
        }]
    }
  }
```

Листинг А.14 — JavaScript модуль утилитарных функции для конфигурации сборки Webpack

```
const path = require('path');
const HtmlWebpackPlugin = require('html-webpack-plugin');
const CleanWebpackPlugin = require('clean-webpack-plugin').CleanWebpackPlugin;
const MiniCssExtractPlugin = require('mini-css-extract-plugin');
```

```
const OptimizeCssAssetWebpackPlugin = require('optimize-css-assets-webpack-plugin');
const TerserWebpackPlugin = require('terser-webpack-plugin');
const CopyWebpackPlugin = require('copy-webpack-plugin');
const VueLoaderPlugin = require('vue-loader/lib/plugin');

module.exports = (mode) => {
    var srcPath = path.resolve(__dirname, 'source');
    console.log(srcPath);
    return {
        srcPath: path.resolve(__dirname, 'source'),
        distPath: path.resolve(__dirname, 'wwwroot'),
        filename: (ext) => {
            return `[name].[contenthash].${ext}`;
        },
        plugins: () => [
            new HtmlWebpackPlugin({
                template: './index.html',
                chunks: ['index'],
                filename: './index.html',
                minify: {
                    collapseWhitespace: mode
                }
            }),
            new HtmlWebpackPlugin({
                template: './login.html',
                chunks: ['login'],
                filename: './login.html',
                minify: {
                    collapseWhitespace: mode
                }
            }),
            new CleanWebpackPlugin(),
            new MiniCssExtractPlugin({
                filename: '[name].[contenthash].css'
            }),
            new CopyWebpackPlugin([{
                from: './favicon.ico',
                to: this.distPath
            }]),
            new VueLoaderPlugin()
        ],
        optimization: () => {
            var base = {
                splitChunks: {
                    chunks: 'all'
                }
            };
            if (mode) {
                base.minimizer = [
                    new OptimizeCssAssetWebpackPlugin(),
                    new TerserWebpackPlugin()
                ]
            }
            return base;
        },
        cssLoaders: (preProcessor) => {
            var base = [{
                loader: MiniCssExtractPlugin.loader,
                options: {
                    hmr: !mode,
                    reloadAll: true
```

```
                },
            },
                'css-loader'
            ];
            if (preProcessor === 'less') {
                base.push('less-loader');
            }
            return base;
        },
        jsLoaders: () => {
            var base = [{
                loader: 'babel-loader',
                options: {
                    presets: ['@babel/preset-env']
                }
            }];
            return base;
        },
        devTool: () => {
            return mode ? '' : 'eval-source-map';
        }
    }
  }
```

ПРИЛОЖЕНИЕ Б

(обязательное)

Графический материал

1) Диаграмма вариантов использования системы

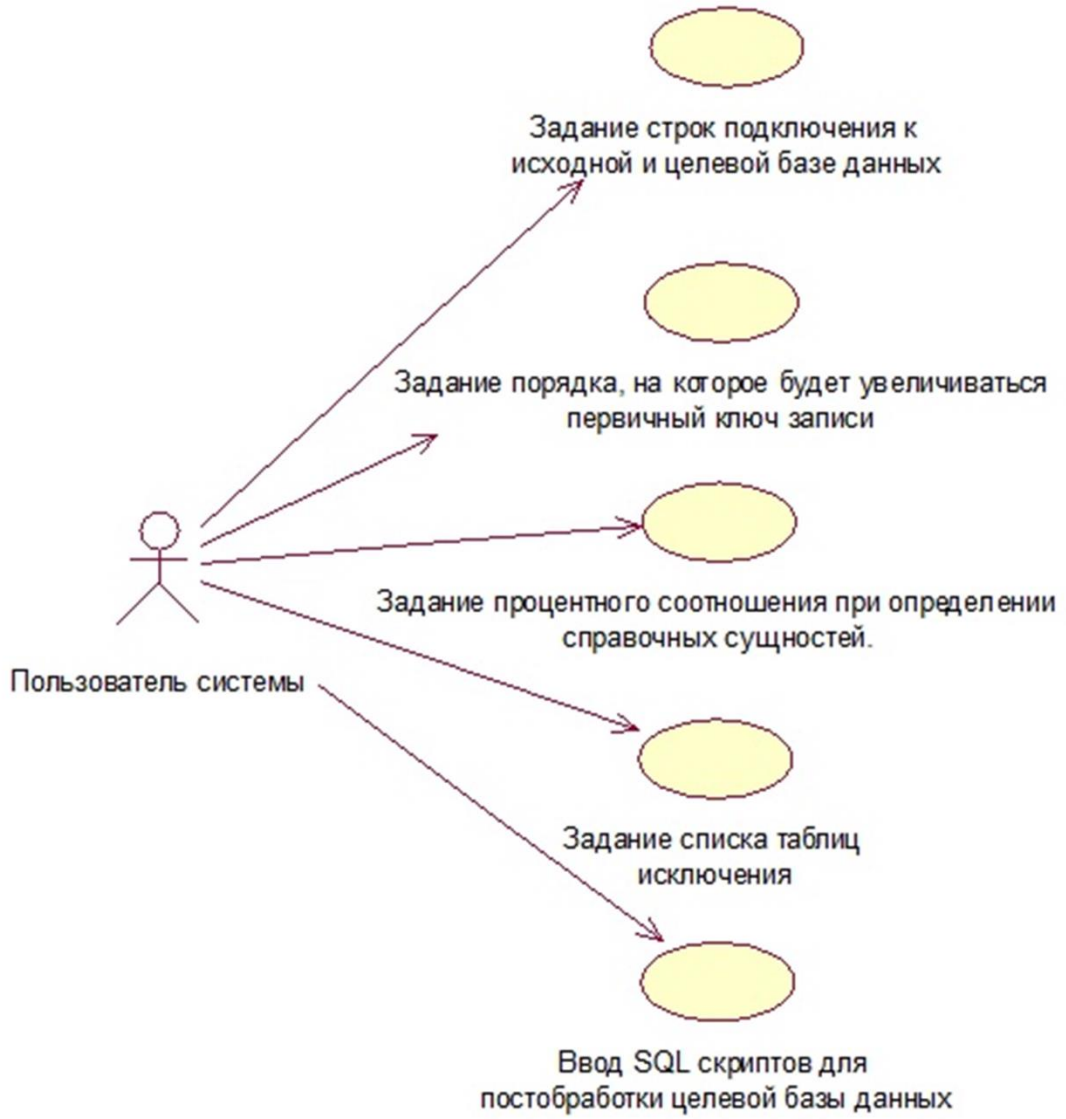
2) Диаграмма развертывания системы

3) Диаграмма последовательности работы пользователя с системой объединения распределённых баз данных

4) Диаграмма деятельности процесса определения справочных сущностей

МД-02069964-09.04.01-02-20

Диаграмма вариантов использования

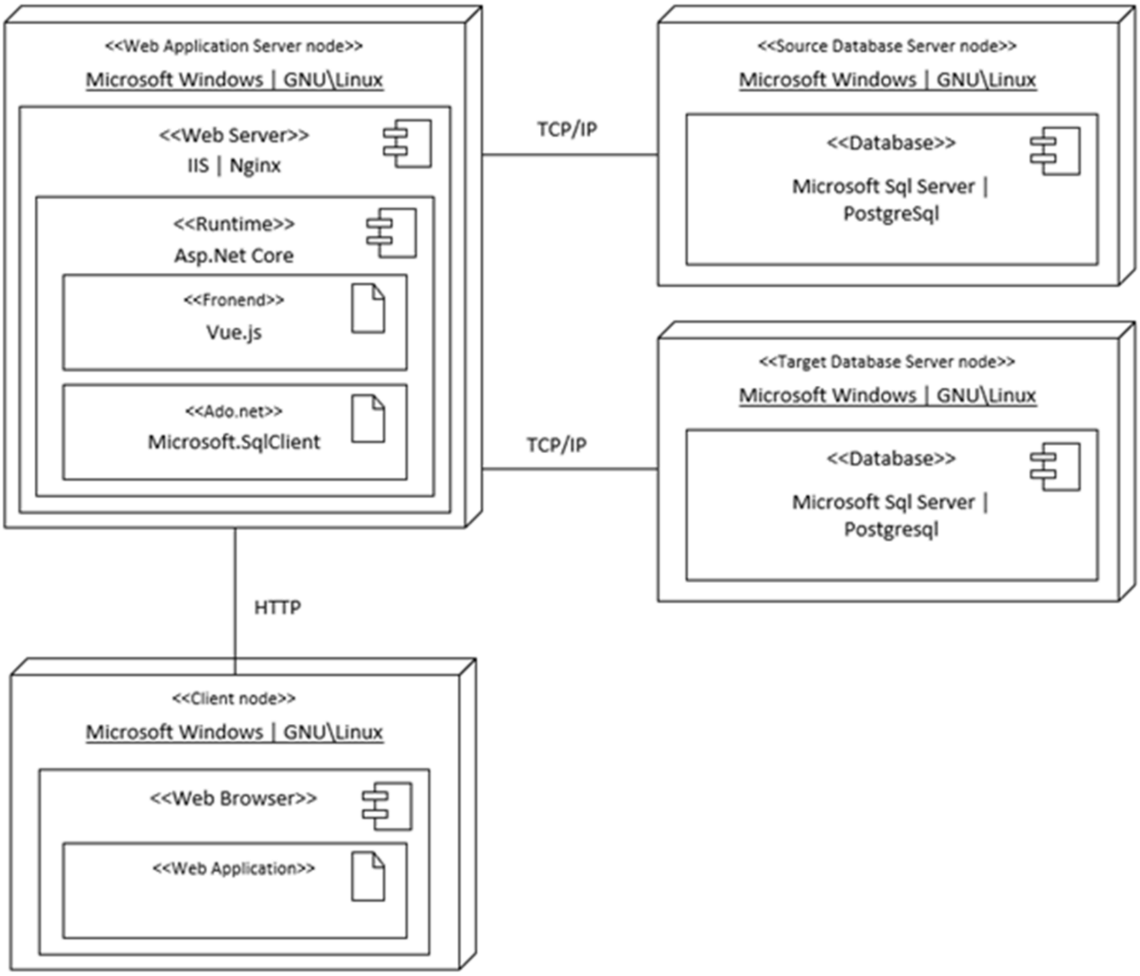


						МД-02069964-09.04.01-02-20	
Изм.	Лист	№ докум.	Подп.	Дата	Разработка алгоритма объединения распределенных баз данных	Лит.	Масса	Масштаб
Разраб.		Ельцанов						
Провер.		Федосин				Лист 1	Листов 4	
Т.контр.								
Н.контр.		Абдрахимов			Диаграмма вариантов использования	НГУ им. Н.П. Огарева ИЭК, АСОиУ ИВТ Группа ИВА-1		
Утв.		Федосин						

					МД-02069964-09.04.01-02-20	
Изм	Лист	№ докум.	Подпись	Дата		132

Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв.№ подл.

Диаграмма развертывания системы

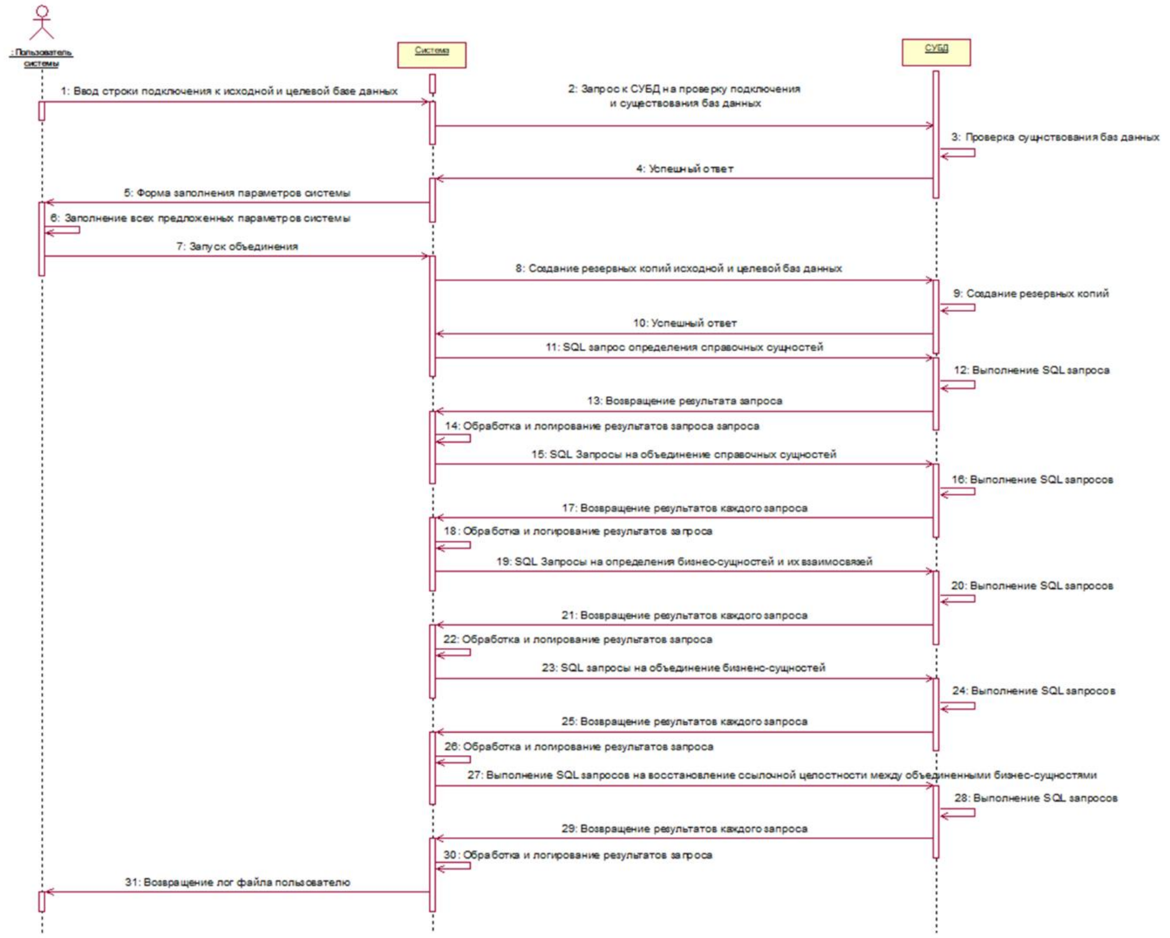


					МД-02069964-09.04.01-02-20		
Изм	Лист	№ докум.	Подп.	Дата	Разработка алгоритма объединения распределенных баз данных	Лит.	Масса / Масштаб
Разраб.		Бальзамов					
Проверил		Федосин					
Т.контр.						Лист 2	Листов 4
Н.контр.		Аббакумов			Диаграмма развертывания системы	МГУ им. Н.П. Огарева ИЭС АСОИУ ИВТ Группа №641	
Утв.		Федосин					

Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв.№ подл.

Изм	Лист	№ докум.	Подпись	Дата

Диаграмма последовательности



					МД-02069964-09.04.01-02-20		
Изм	Лист	№ докум.	Подп.	Дата	Разработка алгоритма объединения распределенных баз данных	Лит. / Масса / Масштаб	
Разраб.		Бальзанов					
Проверил		Федосин					
Т.контр.						Лист 3	Листов 4
Н.контр.		Аббакумов			Диаграмма последовательности	МГУ им. Н.П. Огарева ИЭС АСОИУ ИВТ Группа №641	
Утв.		Федосин					

Подп. и дата

Инв. № дубл.

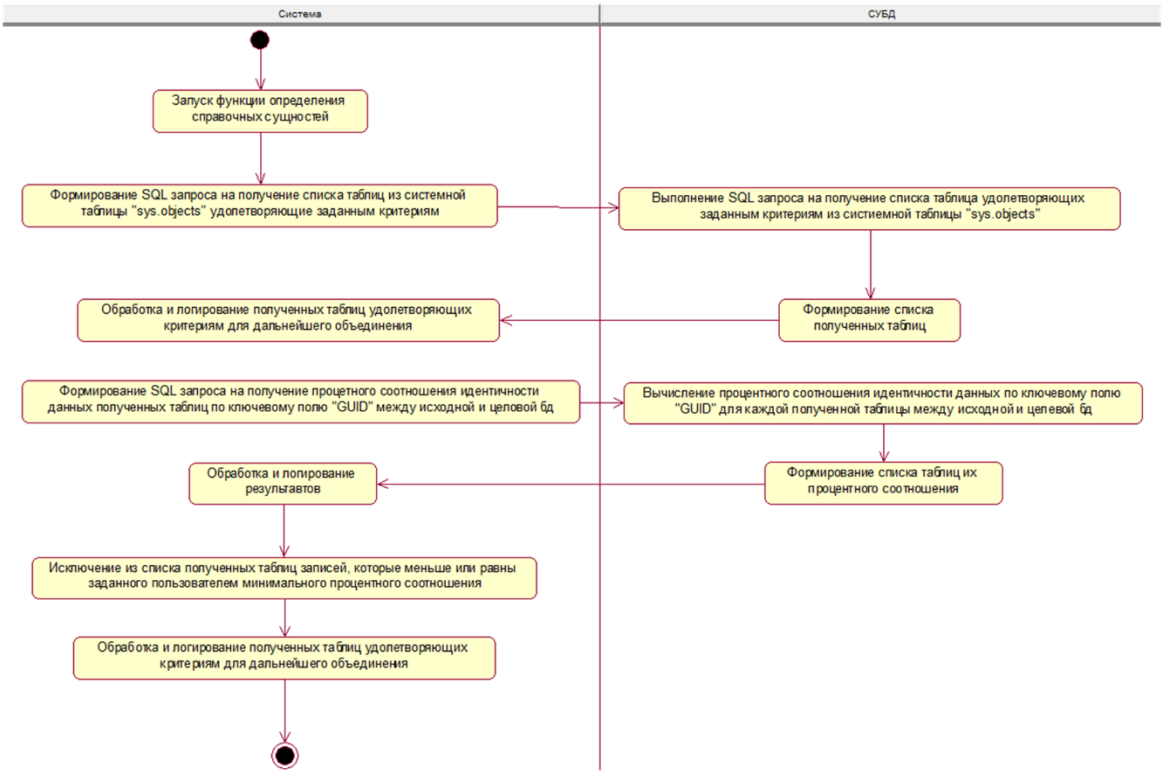
Взам. инв. №

Подп. и дата

Инв.№ подл.

Изм	Лист	№ докум.	Подпись	Дата

Диаграмма деятельности процесса определения справочных сущностей



								МД-02069964-09.04.01-02-20	
Изм.	Лист	№ докум.	Подп.	Дата					
Разраб.		Балыкин						Разработка алгоритма объединения распределенных баз данных	Лит. / Масса / Масштаб
Провер.		Федосин							
Т.контр.									Лист 4 / Листов 4
Н.контр.		Абакумов						Диаграмма деятельности процесса определения справочных сущностей	НГУ им. Н.Э. Баумана ИУК ИСМЭО НВТ Группа ИУ4-41
Утв.		Федосин							

Подп. и дата	Инв. № дубл.	Взам. инв. №	Подп. и дата	Инв.№ подл.

Изм	Лист	№ докум.	Подпись	Дата	МД-02069964-09.04.01-02-20	135

ПРИЛОЖЕНИЕ В

Акт о внедрении научно-технических результатов работы

МИНИСТЕРСТВО
СОЦИАЛЬНОЙ ЗАЩИТЫ, ТРУДА
И ЗАНЯТОСТИ НАСЕЛЕНИЯ
РЕСПУБЛИКИ МОРДОВИЯ
(Минсоцтрудзанятости Республики Мордовия)

Мордовия Республикань эряйхнень эрямань-ащемань араламаснон, покамань и тевонь улемашиснон коряс министерствась

Мордовия Республикань эрицятнень эрямонь-аштемань ванстомань, важодемань ды тевень улемачист коряс министерствась

ул. Титова, 133, г. Саранск, 430027
Тел./факс (8342) 77-71-12/77-72-91
E-mail: minszrm@moris.ru
http://minsoc.e-mordovia.ru

21.05.2020 № 13-2477

На № _____ от _____

ФГБОУ ВО

"МГУ им. Н.П. Огарёва"

АКТ

О внедрении научно-технических результатов работы
Бальзамова Александра Витальевича

Настоящий акт подтверждает, что Обществом с ограниченной ответственностью «Электронные и программные системы» в соответствии с Государственным контрактом № №0809500000318002227_162695 от 22.10.2018 г. были выполнены услуги по объединению сегментов автоматизированной информационной системы "Электронный социальный регистр населения Республики Мордовия" (АИС ЭСРН РМ) подведомственных министерству социальной защиты, труда и занятости РМ учреждений Инсарского и Кадошкинского районов Республики Мордовия (ГКУ «Социальная защита населения по Инсарскому району Республики Мордовия (межрайонная)».)

Ответственным исполнителем работ по данному контракту являлся инженер-программист ООО «Электронные и программные системы» Бальзамов Александр Витальевич.

Полученная в результате выполненных работ объединенная база данных АИС ЭСРН РМ позволила существенно облегчить и ускорить работу Государственного казенного учреждения «Социальная защита населения по Инсарскому району Республики Мордовия (межрайонная)» по назначению и выплате пособий, компенсаций, субсидий, различных мер социальной поддержки пожилых, одиноко проживающих граждан, семей с несовершеннолетними детьми, т.е. самых уязвимых слоев населения проживающих в указанных районах Республики Мордовия.

Заместитель Министра социальной защиты, труда и занятости населения Республики Мордовия С.И. Шувалова

№ 013229

ТКО. ИНН 1327030703, г. Саранск, ул. Титова, 2а. Заказ 10. 2018 г. Тираж 2650