Министерство образования Московской области

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ
«ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

УДК 621.472
№ госрегистрации АААА-А20-120022590034-8
Инв. №

УТВЕРЖДАЮ
Ректор «МГОТУ»
_____Т.Е. Старцева
«___»_____ 2020 г.

ОТЧЕТ
О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ ОПЫТНО-КОНСТРУКТОРСКОЙ РАБОТЕ

Приказ от 21.01.2020 №01-04/22

по теме:
«Расчет параметров системы управления

обработкой больших данных»

Шифр темы 49.27.31 «Системы и аппаратура цифровой передачи»

Руководитель НИР *Т.С. Аббасова*
к.т.н., доцент
_____
подпись, дата

Королев 2020

СПИСОК ИСПОЛНИТЕЛЕЙ

Должность, ученая степень и звание	**Подпись, дата**	**Фамилия, инициалы и номер раздела выполненной работы**
1	2	3
Руководитель работы, к.т.н., доцент	_____ подпись, дата	Т.С. Аббасова (введение, раздел 1,2,5)
Исполнители темы:		
Заведующий кафедрой информационных технологий и управляющих систем, д.т.н., профессор	_____ подпись, дата	В.М. Артюшенко (раздел 1,2)
Профессор, д.т.н., профессор	_____ подпись, дата	Ю.В. Стреналюк (раздел 2,4)
Доцент кафедры информационных технологий и управляющих систем, к.т.н.	_____ подпись, дата	Г.Н. Исаева (введение, раздел 1)
Аспирант кафедры информационных технологий и управляющих систем	_____ подпись, дата	Э.Э. Акимкина (раздел 1, 5)
Аспирант кафедры информационных технологий и управляющих систем	_____ подпись, дата	А.И. Кузнецов (раздел 3)
Магистрант кафедры информационных технологий и управляющих систем	_____ подпись, дата	С.Н. Польшин (раздел 2)
Начальник отдела организации научных исследований	_____ подпись, дата	А.А. Багдасарян (раздел 5)
Младший научный сотрудник	_____ подпись, дата	А.А. Харитонова (раздел 5)

РЕФЕРАТ

Отчет 64 с., 5 ч., 30 рисунков, 12 таблиц, 55 источников.

ИСТОЧНИКИ ДАННЫХ, ПРИКЛАДНЫЕ ПРОГРАММНЫЕ СЕРВИСЫ, ЦЕНТРАЛИЗОВАННОЕ ХРАНЕНИЕ, ДЕЦЕНТРАЛИЗОВАННОЕ ХРАНЕНИЕ, АРХИВАЦИЯ ДАННЫХ, УПРАВЛЯЕМЫЙ ПАРАМЕТР, ВЫЧИСЛИТЕЛЬНЫЕ ЗАДАЧИ, ВЫЧИСЛИТЕЛЬНЫЕ РЕСУРСЫ, СКОРОСТЬ ОБРАБОТКИ ДАННЫХ, ЦИФРОВАЯ ЗАПИСЬ, ОПЫТНЫЙ ОБРАЗЕЦ ПРОГРАММНОГО ПРИЛОЖЕНИЯ.

Объектом исследования являются логи поведения интернет-пользователей, данные управленческих информационных систем и систем управления базами данных (систем поддержки принятия решений), данные распределенных приложений.

Предметом исследования являются методы и алгоритмы обработки и обеспечения «прозрачности» больших объемов данных в системах поддержки принятия решений.

Целью НИОКР является выработка научно-обоснованных технических решений, полученных на основе расчетов параметров обслуживания, и соответствующей технической документацией для повышения эффективности поддержки решений при обработке больших объемов данных, в том числе данных, предоставляемых для аналитики посредством веб-сервисов.

Результаты работы:

- разработаны алгоритмы архивирования, репликации и секционирования для повышения управляемости, производительности и доступности больших баз данных и действующих на их основе прикладных веб-сервисов;

- создан опытный образец программного распределенного приложения, моделирующего платформу по обработке больших объемов данных с помощью технологии блокчейн;

- рассчитаны параметры системы управления обработкой больших данных с помощью адаптивной к входящему потоку запросов системы поддержки принятия решений, в которой число элементов обслуживания увеличивается в зависимости от интенсивности данного потока; показано достижение эффекта за счет снижения затрат на содержание данных элементов и уменьшения процента потерянных запросов из-за больших очередей;

- разработана техническая документация, описывающая организацию информационного обеспечения для обработки больших объемов данных с использованием технологии блокчейн.

Степень внедрения – результаты исследований могут быть использованы для контроля параметров и повышения помехозащищенности оборудования существующих и вновь разрабатываемых инфокоммуникационных систем.

Область применения – полученные результаты предполагается использовать для решения актуальной научной задачи обработки больших объемов данных и выполнения требований по обеспечению «прозрачности» данных, повышению управляемости, производительности и доступности больших баз данных и действующих на их основе прикладных веб-сервисов.

Содержание

Определения, обозначения и сокращения

В настоящем отчете о НИР применяют следующие термины с соответствующими определениями:

База данных (БД) – представленная в объективной форме совокупность самостоятельных материалов (статей, расчётов, нормативных актов, судебных решений и иных подобных материалов), систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины;

Большие данные (Big Data) – обозначение структурированных и неструктурированных данных огромных объёмов и значительного многообразия, эффективно обрабатываемых горизонтально масштабируемыми программными инструментами, появившимися в конце 2000-х годов и альтернативных традиционным системам управления базами данных и решениям класса Business Intelligence;

Информационная система (ИС) – система, предназначенная для хранения, поиска и обработки информации, и соответствующие организационные ресурсы (человеческие, технические, финансовые и т. д.), которые обеспечивают и распространяют информацию (ISO/IEC 2382:2015);

Многомерные хранилища данных (МХД) – системы хранения, опирающиеся на концепцию многомерных кубов (гиперкубов), ориентированные на аналитическую обработку данных и выполнение сложных нерегламентированных запросов;

Система поддержки принятия решений (СППР) – компьютерная автоматизированная система, целью которой является помощь людям, принимающим решение в сложных условиях, для полного и объективного анализа предметной деятельности;

Элемент обслуживания (ЭО) – служебный элемент; в данной работе элементом обслуживания запросов является контролируемый сервер, с которого собираются данные МХД;

API (от англ. Application Programming Interface – Интерфейс Программирования Приложений) – набор удобных функций, позволяющих получить доступ к какому-либо сервису и запросить у него данные;

APP

JDBC (товарный знак, не аббревиатура, часто расшифровывается как Java Database Connectivity) – промышленный стандарт для независимого от базы данных взаимодействия Java-платформы и широкого диапазона баз данных, определяет API для доступа к базам данных из Java-приложений;

REST (от англ. Representational State Transfer – передача состояния представления) – архитектурный стиль взаимодействия компонентов распределённого приложения в сети, согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы;

Введение

В настоящее время данные, которые требуются обществу, растут в геометрической прогрессии. Только объемы данных, хранимые в сети Интернет, ежегодно увеличиваются примерно на сорок процентов. Такая тенденция обеспечивается, с одной стороны, бурным развитием информационных технологий (ИТ), а с другой стороны, чтобы обеспечить хранение и обработку таких объемов разнообразных данных должны появляться новые программные продукты и решения – они также обеспечивают рост данных. Большие объемы данных генерируют социальные сети, мобильные устройства, измерительные устройства и датчики, источники бизнес-информации.

Большие данные (Big Data) – это данные, которые не помещаются в оперативную память компьютера. Свойство «быть большим» является не самостоятельным свойством данных, а зависит от характеристики системы, применяемой для их обработки. Многие современные компании прибегают к технологии Big Data, не создавая для этого надлежащей инфраструктуры для надежного хранения огромных массивов данных и их прозрачности, которая означает, что данные доступны всем пользователям, и их невозможно удалить или подделать. Прозрачность работы с данными осуществляет технология блокчейн, но существуют проблемы с совместным использованием этой технологии и технологии больших данных. Актуально создание такой инфраструктуры системы управления большими данными, которая смогла бы обеспечить бесконфликтность и прозрачность работы с данными.

В качестве исходных больших данных могут выступать: логи поведения интернет-пользователей; Интернет вещей; социальные медиа; метеорологические данные; оцифрованные книги крупнейших библиотек; GPS-сигналы (от англ. Global Positioning System – глобальная система позиционирования) из транспортных средств; информация о транзакциях клиентов банков; данные о местонахождении абонентов мобильных сетей; информация о покупках в крупных ритейл-сетях и т.д. В данной работе **объектом исследования** являются логи поведения интернет-пользователей, данные управленческих информационных систем и систем управления базами данных (систем поддержки принятия решений), данные распределенных приложений.

Предметом научно-исследовательской опытно-конструкторской работы (НИОКР) являются методы и алгоритмы обработки и обеспечения «прозрачности» больших объемов данных в системах поддержки принятия решений.

Целью НИОКР является выработка научно-обоснованных технических решений, полученных на основе расчетов параметров обслуживания, и соответствующей

технической документацией для повышения эффективности поддержки решений при обработке больших объемов данных, в том числе данных, предоставляемых для аналитики посредством веб-сервисов.

Основными задачами НИОКР являются:

1. Описание комплекса технических средств системы управления обработкой больших объемов данных с помощью запросов и прикладных сервисов.
2. Повышение управляемости, производительности и доступности для больших баз данных и действующих на их основе прикладных веб-сервисов.
3. Моделирование децентрализованного приложения с использованием технологии блокчейн для обеспечения «прозрачности» больших данных.
4. Расчет параметров адаптивной системы поддержки принятия решений для управления обработкой больших объемов данных.
5. Разработка технической документации для описания информационного обеспечения обработки больших данных с применением технологии блокчейн.

Ожидаемые результаты научно-исследовательской опытно-конструкторской работы:

- разработка алгоритмов архивирования, репликации и секционирования больших данных для повышения управляемости, производительности и доступности больших баз данных и действующих на их основе прикладных веб-сервисов;
- создание распределенного приложения, моделирующего платформу по обработке больших объемов данных с помощью технологии блокчейн;
- улучшение параметров системы управления обработкой больших данных с помощью адаптивной к входящему потоку запросов системы поддержки принятия решений, в которой число элементов обслуживания увеличивается в зависимости от интенсивности данного потока; достижение экономического эффекта за счет снижения затрат на содержание данных элементов и уменьшения процента потерянных запросов из-за больших очередей;
- техническая документация, описывающая организацию программного и информационного обеспечения для обработки больших объемов данных.

Полученные результаты предполагается использовать для решения актуальной научной задачи обработки больших объемов данных и выполнения требований по обеспечению «прозрачности» данных, повышению управляемости, производительности и доступности больших баз данных и действующих на их основе прикладных веб-сервисов.

1. Описание комплекса технических средств системы управления обработкой больших объемов данных с помощью запросов и прикладных сервисов

Современная и быстро развивающаяся технология Big Data появилась более десяти лет назад и быстро стала популярной среди крупнейших компаний бизнеса, экономики, которые и финансируют ее совершенствование [1].

Если рассмотреть временной интервал развития данной технологии, то выявлены следующие тенденции. Пик развития данной технологии приходится на 2012 год, как показывают статистические данные, полученные из российских и зарубежных источников. Так компания Datashift (Бельгия) дает следующую информацию по числу упоминаний Big Data в социальных сетях: в 2012 году термин «Big Data» употреблялся около двух миллиардов раз в постах, которые принадлежат одному миллиону авторов на всей планете, что эквивалентно двести шестидесяти постам в час (пик упоминаний составил 3070 упоминаний в час) [2]. На рисунке 1.1 представлена статистика по кварталам за этот год по ведущим вендорам, взявшим на вооружение Big Data.

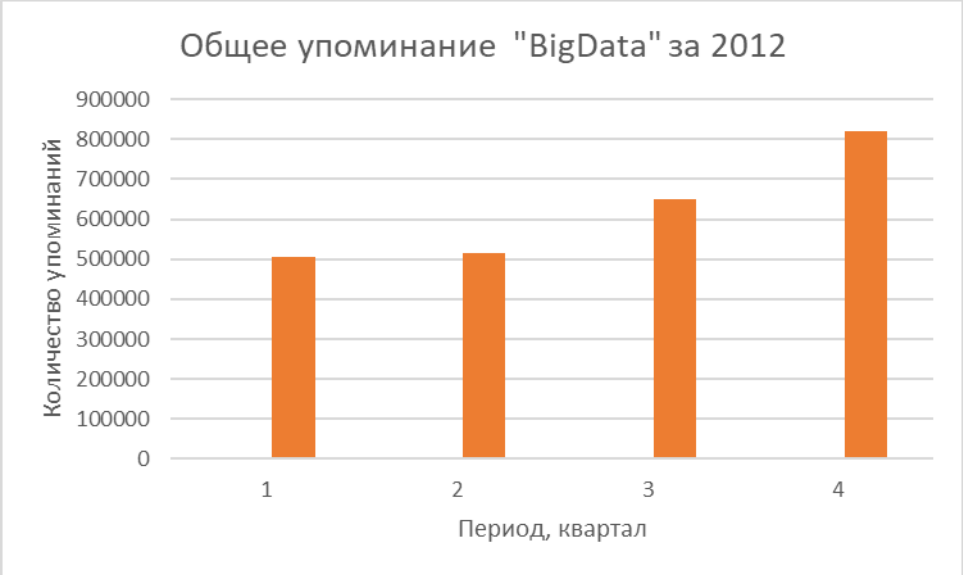

Рисунок 1.1 – Упоминание в сетях интернет термина Big Data [2]

Темы, в которых использовался данный термин, касались таких аспектов: мифы и реальность, опыт использования, человеческий фактор, возврат инвестиций, новые технологии. Среди вендоров, как показано на рисунке 1.2, наиболее активными были ведущие фирмы в области информационных технологий: Apache, 10gen, IBM, HP и Teradata.

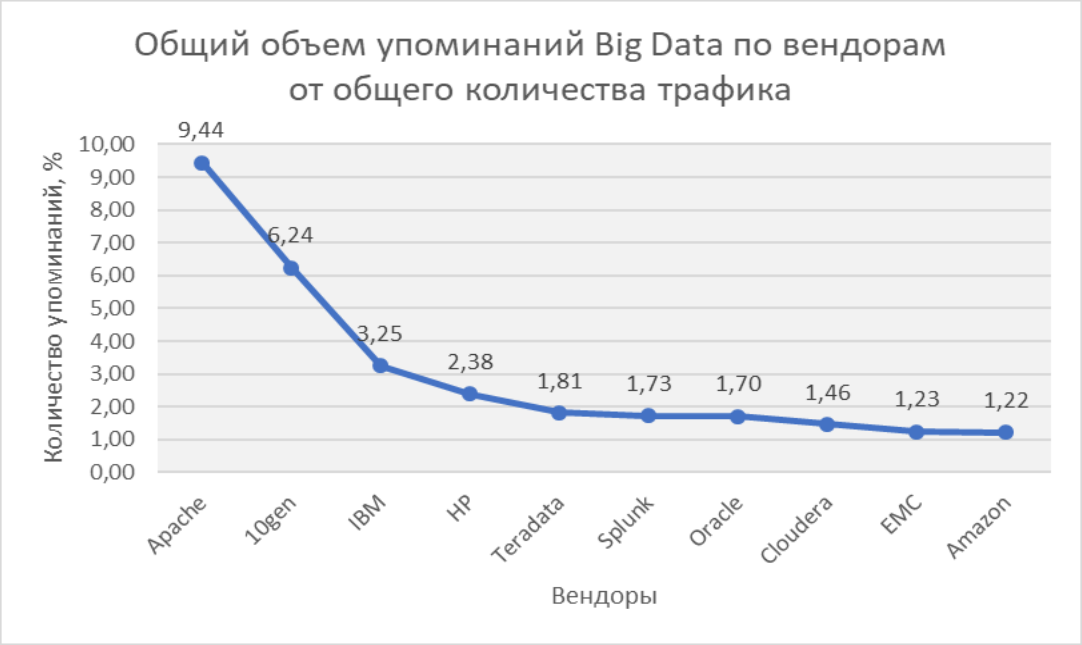

Рисунок 1.2 – Упоминание термина Big Data по фирмам за 2012 год [2]

После пика развития терминологии больших данных, сама суть подхода к технологии стала меняться, выявились составляющие Big Data масса программных продуктов, позволяющих удобно работать с большими данными в разных отраслях экономики.

В 2016-2018 годы интерес к инструментам сбора, обработки, управления и анализа Big Data проявляют все ведущие компании в области ИТ, так как они постоянно работают с огромными объемами информации; использование феномена больших данных позволяет данным компаниям динамично развиваться в бизнесе и искать новые рынки сбыта своих разработок и услуг. В таблице 1.1 приведен перечень основных компаний, задействованный по исследованиям на 2016 год в пространстве Big Data. Кроме того, в данной таблице указаны фирмы- стартапы, которые трудятся в области обработки огромных массивов данных и используют готовые облачные решения от известных крупных разработчиков, таких как Amazon.

Согласно International Data Corporation (IDC) — компании, занимающейся аналитикой в различных областях ИТ-индустрии, рост данных во всемирной сети Интернет, интерес к Интернет-вещей, подключение к Интернет все большего числа абонентов приведут к увеличению доходов компаний, использующих Big Data, от ста тридцати миллиардов долларов к более чем двухсот миллиардов долларов в 2020 году.

Таблица 1.1 – Ведущие компании в области Big Data [2]

№ п/п	ИТ-компании	Стартапы (Startup Company)
1.	Amazon	Acunu
2.	Dell	Apigee
3.	eBay	Aspera
4.	EMC	Aster Data (поглотила Teradata)
5.	Facebook	Cloudera
6.	Fujitsu	Couchbase
7.	Google	Datameer
8.	HDS (Hitachi Data Systems Corporation)	DataStax
9.	HP	Factual
10.	IBM	GoodData
11.	LinkedIn	Greenplum (поглотила EMC)
12.	Microsoft	Hortonworks (имеет общие корни с Yahoo)
13.	NetApp	MapR Technologies
14.	Oracle	Metaweb Technologies (поглотила Google)
15.	SAP	Netezza (поглотила IBM)
16.	SAS	nPario
17.	SGI (Silicon Graphics Inc)	Palantir Technologies
18.	Teradata	ParAccel
19.	VMware	ParStream
20.	Yahoo	SenSage
21.		Socrata
22.		Splunk
23.		Sybase (поглотила SAP)
24.		TellApart
25.		Vertica (поглотила HP)

Известно, что в накопленных больших данных для предприятий может содержаться много полезной информации о конкурентах, о клиентах, о ранках сбыта. Однако, компании, желающие использовать Big Data, сталкиваются с различными проблемами, что задерживает использование этой новой технологии в настоящее время:

— слабые процессорные мощности;

— старая инфраструктура ИТ;

— нехватка финансовых ресурсов;

— условия безопасности.

На основе данных агентства Accenture [3], по итогам результатов большой выборки опрошенных компаний, проблемы в области внедрения больших данных за последние пять лет выражены в процентном соотношении на рисунке 1.3. Анализ данных рисунка 1.3 позволяет выявить причины, по которым предприятия и компании не могут внедрить технологии Big Data и аналитические современные программные системы. Самой большой проблемой внедрения Big Data, как и для других ИТ-технологий, является проблема безопасности – более 50% процентов опрошенных предприятий.

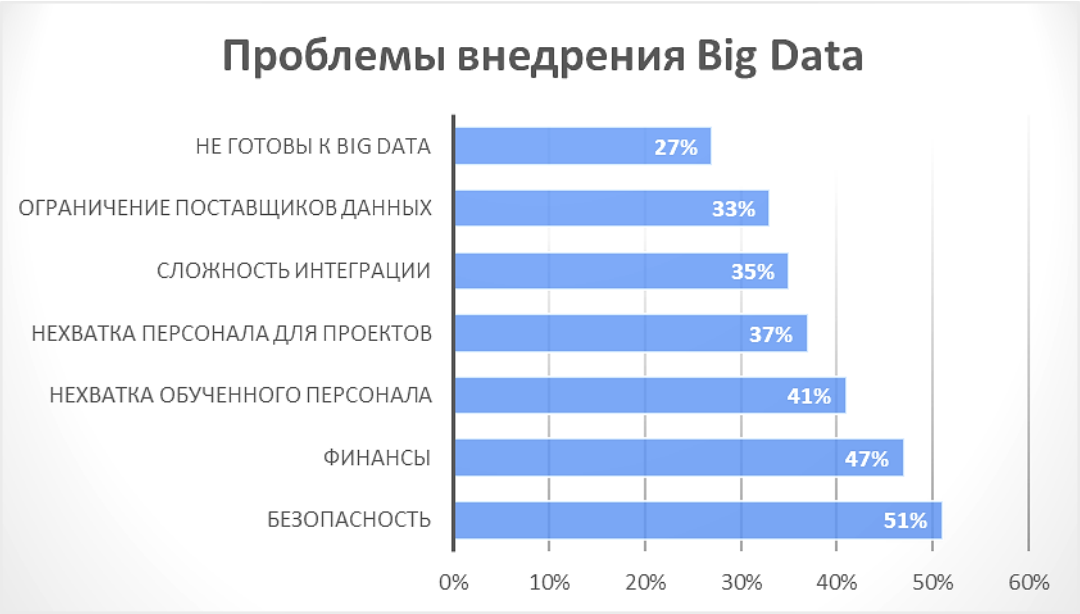

Рисунок 1.3 – Основные проблемы при внедрении больших данных [3]

Для обработки больших объемов данных предназначены традиционные системы управления базами данных и альтернативные им горизонтально масштабируемые программные инструменты, решения класса Business Intelligence, а также многомерные хранилища данных (МХД). Основными инструментами обработки данных являются сортировка, фильтр и запрос. При обработке больших данных также используются XML-преобразования, веб-сервисы, спецификации JDBC (товарный знак) и JMS (от англ. Java Message Service – Служба сообщений Java). Большое влияние на производительность систем с большими данными оказывает время обработки запросов, которые имеют сложный нерегламентированный характер. МХД лучше справляются с обработкой сложных нерегламентированных запросов. На основе МХД строятся современные системы поддержки принятия решений (СППР), которые образованы в результате слияния информационных управляющих систем и систем управления базами (хранилищами) данных.

Во многих работах рассмотрение СППР производится в том ракурсе, что все параметры её работы заранее известны и не изменяются с течением времени. Однако для реально существующих СППР, т.е. наиболее интересных для пристального изучения, такое предположение не всегда является справедливым. В частности, может не учитываться выход из строя отдельных элементов обслуживания (ЭО) запросов в рассматриваемой системе, либо циклическое изменение (повышение или понижение быстродействия отдельных ЭО в зависимости от текущей нагрузки на систему в целом). В данной ситуации естественно при обслуживании запросов и сценариев пользоваться

адаптивным подходом. Таким образом, возникает задача адаптации СППР к изменяющимся условиям внешней и внутренней среды с целью оптимизации функционирования.

В разделе поставлена задача установить в процессе системного анализа системообразующие компоненты (уровни) архитектуры адаптивной СППР, связи и отношения между этими компонентами, а также разработать комплекс технических средств системы управления обработкой больших объемов данных с помощью запросов и прикладных сервисов.

МХД является предметно-ориентированной информационной базой данных (БД), специально разработанной и предназначенной для подготовки отчётов и анализа с целью поддержки принятия решений в организации.

Задачи интеллектуального анализа интегрированных данных различаются по уровню сложности. Обобщенная концептуальная схема ХД, представленная на рисунке 1.4 [4], имеет трехуровневую архитектуру (источники данных – хранилище данных – отчеты) [5] и иллюстрирует зависимость архитектуры ХД от доступности и структуры основной БД. На третьем уровне (визуализация, отчеты) могут также располагаться предметно-ориентированные витрины данных (ВД). СППР может иметь независимые витрины данных, не входящие в архитектуру ХД [6].

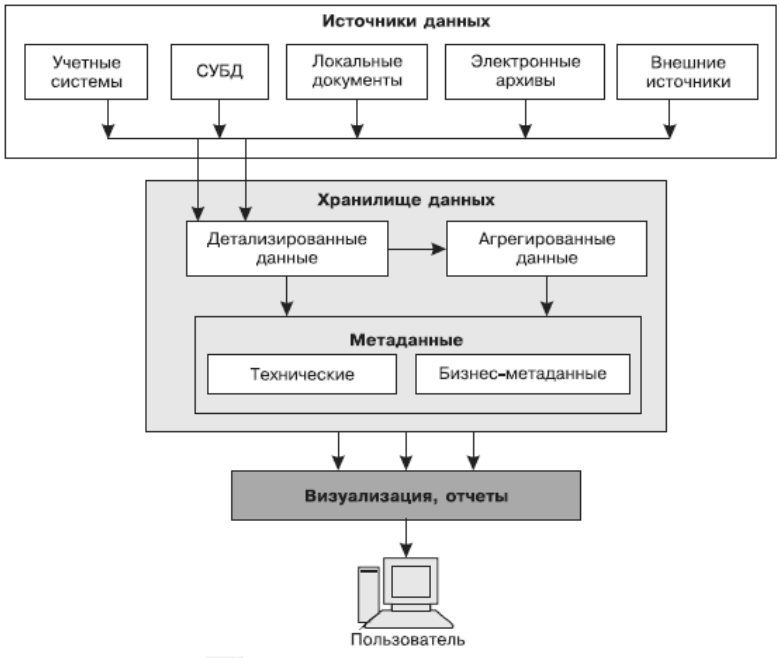

Рисунок 1.4 – Типовая конфигурация хранилища данных [4]

В целях повышения оперативности обработки данных допускаются ненормализованные отношения, трехуровневая архитектура ХД представляется гиперкубом данных [7].

Широко распространенные трехуровневые архитектуры ХД и гиперкубы, основанные на трехуровневой архитектуре, не могут описывать задачи современных ХД, характеризующихся большим объемом обрабатываемых данных разных форматов (до миллиона запросов в день). Архитектура ХД должна отражать средства интеграции данных с другими источниками; производительности; трансформации данных; истории получения данных; чистоты данных. Широко использующиеся при обработке данных XML-преобразования, веб-сервисы, спецификации JDBC и JMS определяют архитектуру ХД как сервисно-ориентированную [8]. Для отображения новейших сервисов интеграции, хранения, доставки, предоставления данных автором разработана сервис-ориентированная шестиуровневая архитектура ХД, структура которой представлена на рисунке 1.5.

Первый уровень «Источники данных»	Документы MS Office
	Унаследованные системы
	Транзакционные системы
	Учетные системы
	Файлы
	Архивы
	Системы управления БД
Второй уровень «Загрузка и преобразование данных»	Извлечение и сбор данных
	Трансформация данных
	Загрузка данных
Третий уровень «Хранение данных»	Ведение метаданных
	Центральное хранилище данных
	Оперативный склад данных
	Зоны временного хранения
	Режимы хранения
Четвертый уровень «Обработка и анализ данных»	Выборка
	Реструктуризация
	Доставка
Пятый уровень «Предоставление данных»	Тематическая витрина данных
	Региональная витрина данных
	Витрина данных подразделения
	Прикладная витрина данных
	Функциональная витрина данных
Шестой уровень «Приложения данных»	Сценарный анализ
	Статистический анализ
	Многомерный анализ
	Отчетность
	Планирование

Рисунок 1.5 – Шестиуровневая архитектура ХД [8]

Передача данных из первого уровня во второй уровень осуществляется на основе подхода ETL (от англ. Extract, Transformation, Load – извлечение, трансформация,

загрузка), включающего этапы. На втором уровне находятся двумерные ASCII-файлы, файлы во внутреннем формате системы или в виде временных промежуточных таблиц в БД (снимки или реплицированные из других источников таблицы). Данные, находящиеся на втором уровне, для анализа не доступны. На втором уровне может быть введен промежуточный слой прикладного программного обеспечения (ПО), которое берет на себя функции нивелирования технических решений разных автоматизированных систем путем реализации новых и поддержки некоторых старых API-интерфейсов (Application Programming Interface – интерфейс прикладного программирования) [9].

При хранении данных на третьем уровне обеспечивается распределенный доступ к ХД, резервирование информации, защита от сбоев программного и аппаратного обеспечения.

При выборке, реструктуризации и доставке данных на четвертом уровне определяется частота обновления данных в ХД и процедуры обработки данных. В соответствии с заданной частотой и процедурами формируются требования к пропускной способности каналов связи и алгоритмам управления трафиком и распределения нагрузки.

Цели визуализации данных и средства восстановления данных на пятом уровне должны быть четко определены [10 – 12]. Пример реализации шестиуровневой архитектуры представлен на рисунке 1.6.

Рисунок 1.6 – Реализация шестиуровневой архитектуры ХД [сделано автором]

В связи с развитием практик гибкой разработки на основе сервис-ориентированной архитектуры разработана микросервисная архитектура для программного обеспечения,

ориентированного на взаимодействие насколько это возможно небольших, слабо связанных и легко изменяемых модулей – микросервисов. Микросервис – часть прикладного сервиса. Архитектура симметричная, а не иерархическая: зависимости между микросервисами одноранговые.

Для повышения эффективности обработки больших данных также рассматривается технология блокчейн. Несмотря на принципиальное различие технологии блокчейн (децентрализованная архитектура для хранения копий информационных цепочек) и технологии больших данных (централизованная архитектура для интеграции информации из различных источников), и разных скоростей работы, уже реализованы проекты с достаточно успешным совмещением технологий блокчейн и больших данных [13]. Основные характеристики этих проектов приведены в таблице 1.2.

Таблица 1.2 – Проекты, совмещающие технологии блокчейн и больших данных [сделано автором]

Название проекта	Задачи проекта	Преимущества
Облачные сервисы хранения больших данных Storj и FileCoin	Обеспечение надежности, абсолютной неизменности и защиты данных от несанкционированного доступа	Сокращение стоимости хранения данных на 90% по сравнению с подобными решениями от Amazon Web Services' Cloud
Omnilytics (информационный сервис)	Объединение блокчейн с аналитикой больших данных по прогнозированию тенденций в различных отраслях, финансам, аудиту	Поддержка smart-контрактов, распределенной идентификации данных, обмен информацией через API и другие протоколы
Rublix (международная торговая платформа)	Проверка подлинности и авторитета трейдеров для криптовалютных инвесторов	Предоставление доступа к рыночной информации для уменьшения текущей путаницы
Datum (децентрализованная сеть хранения информации)	Монетизация индивидуальных данных, управление токеном доступа к данным (DAT, Data Access Token)	Обеспечение технологией блокчейн прозрачности всей цепочки поставок, обеспечение необходимой аналитики технологией больших данных
Provenance (информационный сервис)	Хранение и предоставление данных о происхождении продуктов/изделий	

Технология блокчейн обеспечивает надежное и прозрачное хранение всей истории транзакций, а также целостность информации, а технология больших данных предоставляет новые инструменты для эффективного анализа и прогнозирования.

2. Повышение управляемости, производительности и доступности больших баз данных и действующих на их основе прикладных сервисов

Информационная система (ИС) сервис-ориентированной архитектуры (СОА), обрабатывающая большие объемы данных, состоит из множества сервисов. Актуальна разработка дополнительного веб-сервиса по распознаванию лояльных клиентов к услугам, предоставляемых сервисами ИС, а также БД для хранения и обработки логов поведения интернет-пользователей.

Для выявления нелояльных пользователей сервиса используется скоринговая оценка, основанная на численных статистических методах. Эта оценка также применяется в потребительском (магазинном) экспресс-кредитовании на небольшие суммы, в бизнесе сотовых операторов, страховых компаний и т. д. Скоринг заключается в присвоении баллов по заполнению некоей анкеты, разработанной оценщиками кредитных рисков. По результатам набранных баллов системой принимается решение об одобрении в выдаче услуги веб-сервиса (в том числе в кредит).

В качестве БД выбрана архивная база, т.к. она меньше всего загружена. Разработана схема control и таблица в ней state. Таблица state создана для управляющих параметров, участвует в части ИС СОА для предоставления веб-сервиса по распознаванию лояльных клиентов и имеет структуру, показанную в таблице 2.1.

Таблица 2.1 – Структура таблицы state [сделано автором]

Колонка	Тип поля	Значение по умолчанию	Описание
state_id	serial		Первичный ключ
short_name	character varying		Краткое имя параметра
description	character varying		Описание
value	real	0	Значение с плавающей запятой
is_enable	boolean	false	Он знает, работает этот параметр или нет; необходим для отключения управляющего параметра путем обновления значения в БД

Со временем количество данных в БД увеличивается. В крупных интернет сервисах с логированием данные копятся очень быстро, образуется большое количество данных в БД, благодаря чему увеличивается нагрузка на аппаратную часть серверов. При не столь большом запасе производительности в аппаратной части, время общения с БД (чтение, запись и обновление данных) увеличивается. В свою очередь это влияет на время выполнения REST (от англ. Representational State Transfer — «передача состояния

представления запросов) к API (от англ. Application Programming Interface — Интерфейс Программирования Приложений). Со стороны пользователя веб-сервиса заметно увеличение времени загрузки веб-страниц. Все это чревато большими потерями [14] по следующим причинам:

- пользователей не устраивает скорость загрузки веб-страниц, количество посетителей веб-сервиса сокращается;
- из-за высокого времени отклика API, веб-сервис регистрирует не все заказы, происходит потеря заказов сервиса.

Зачастую, данные логов не хранят долго, а просто удаляют старые, в нашем случае, необходимо сохранять все логи пользователей, независимо от времени. Наступает момент, когда стоит найти правильное архитектурное решение, в нашем случае наиболее подходящим решением было прибегнуть к масштабированию данных в БД – к репликации и шардингу.

Прежде чем приступить к оптимизации, стоит отметить, что у базы данных уже было настроено резервное копирование, оно происходит раз в сутки и хранится на отдельном сервере (недельные данные резервируются). Резервное хранилище хранит три дампа, более старая запись удаляется добавляется новая [15].

Изначально, до того, как основная таблица log стала стала иметь большие данные, ИС имела следующую архитектуру, представленную на рисунке 2.1.

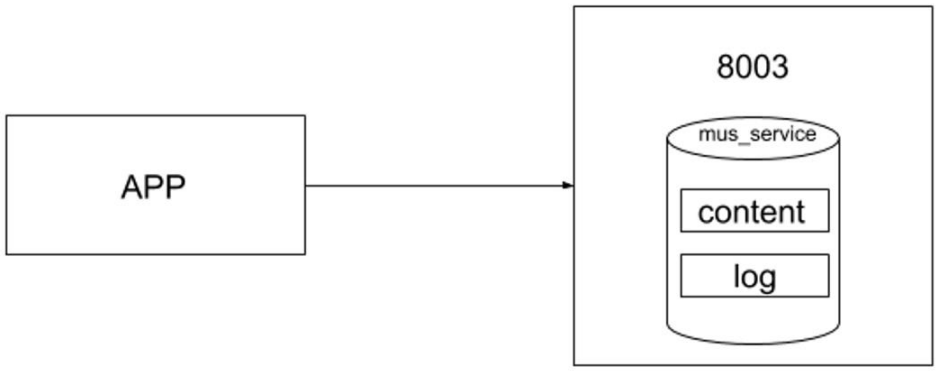

Рисунок 2.1 – Схема первоначальной архитектуры информационной системы [сделано автором]

Приложение APP обращается к БД service на хост сервера 8003 для всех операций – чтение, запись и обновление данных:

- в таблицу log постоянно пишутся данные для аналитики;
- к таблице contentпоступают запросы на чтение данных при каждом обращении к APIMC.

Когда данных в главной таблице БД mus_service – log, становится много, нагрузка на БД увеличивается, скорость отклика уменьшается и приложение начинает задыхаться из-за долгих запросов на чтение к таблице content. Чтобы устранить этот недочёт, стоит прибегнуть к репликации.

Репликация – одна из техник масштабирования БД. Она состоит в том, что данные с одного сервера базы данных постоянно копируются (реплицируются) на один или несколько других (называемые репликами). Для приложения появляется возможность использовать не один сервер для обработки всех запросов, а несколько [16]. Таким образом появляется возможность распределить нагрузку с одного сервера на несколько.

Самая распространенная схема репликации данных – это master-slave. Master – это основной сервер БД, куда поступают все данные. Все изменения в данных (добавление, обновление, удаление) должны происходить на этом сервере. Slave – это вспомогательный сервер БД, который копирует все данные с мастера. С этого сервера следует читать данные. Таких серверов может быть несколько.

Данная схема подходит наиболее лучше из-за архитектуры ИС. Запись в таблицу log ведётся чаще чем чтение данных из таблицы content. Чтение из content необходимо для предоставления услуг и сопровождения заказа пользователя. Данная схема позволяет отделить логирование от функционала заказов сервиса, см. рисунок 2.2.

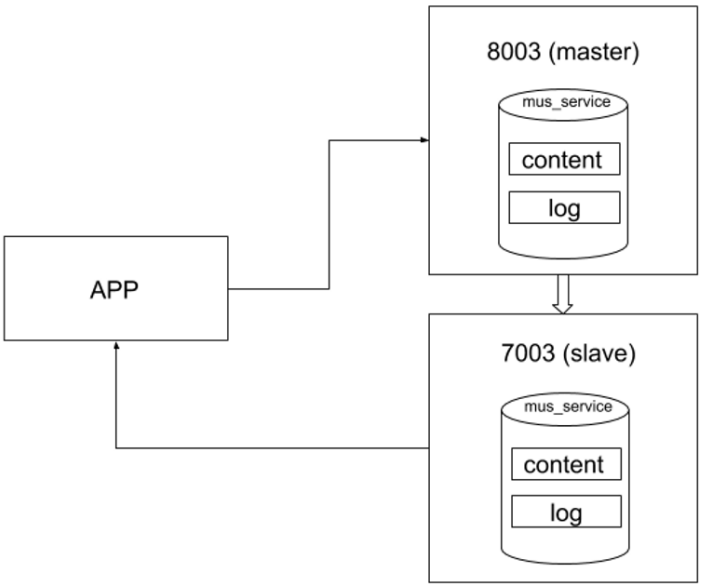

Рисунок 2.2 – Схема ИС с настроенной репликацией master-slave
[сделано автором]

После добавления репликации master-slave, мы настроили чтение и запись из базы следующим образом:

- На мастере 8003 ведётся запись, обновление и удаление данных. Постоянное логирование в таблицу log, а также, периодическое обновление таблицы content через CMS (от англ. Content Management System – Система Управления Контентом);
- На слейве 7003 мы читаем данные из таблицы content, необходимые для заказов интернет-пользователей, так же, из таблицы logданные попадают в аналитическую часть ИС.

Репликация сама по себе не очень удобный механизм масштабирования. Рассинхронизация данных и задержки в копировании с мастера на слейв являются основной причиной этого. Зато это отличное средство для обеспечения отказоустойчивости. Мы всегда можем переключиться на слейв, если мастер ломается и наоборот. Чаще всего репликация используется совместно с шардингом именно из соображений надежности [17].

В нашем случае рассинхронизация данных не критична, поскольку в операциях на чтение со слейва допускается небольшая задержка в копировании, в этой части ИС нет необходимости в горячих данных. К сожалению, репликация не решает все задачи, поставленные перед масштабированием. Со временем записывать и читать данные из таблицы log стало все труднее, будь это мастер или слейв, виной всему большое количество данных. Исходя из этого, было решено дополнительно прибегнуть к другой технике масштабирования данных – шардированию.

Секционирование (от англ. partitioning, часто называют партиционированием) – это разделение хранимых объектов баз данных (таких как таблиц, индексов, материализованных представлений) на отдельные части с раздельными параметрами физического хранения. Используется в целях повышения управляемости, производительности и доступности для больших баз данных. Возможные критерии разделения данных, используемые при секционировании — по предопределённым диапазонам значений, по спискам значений, при помощи значений хэш-функций; в некоторых случаях используются другие варианты. Под композитными (составными) критериями разделения понимают последовательно применённые критерии разных типов.

Секционированием данных называется разбиение одной большой логической таблицы на несколько небольших физических секций, см. рисунок 2.3.

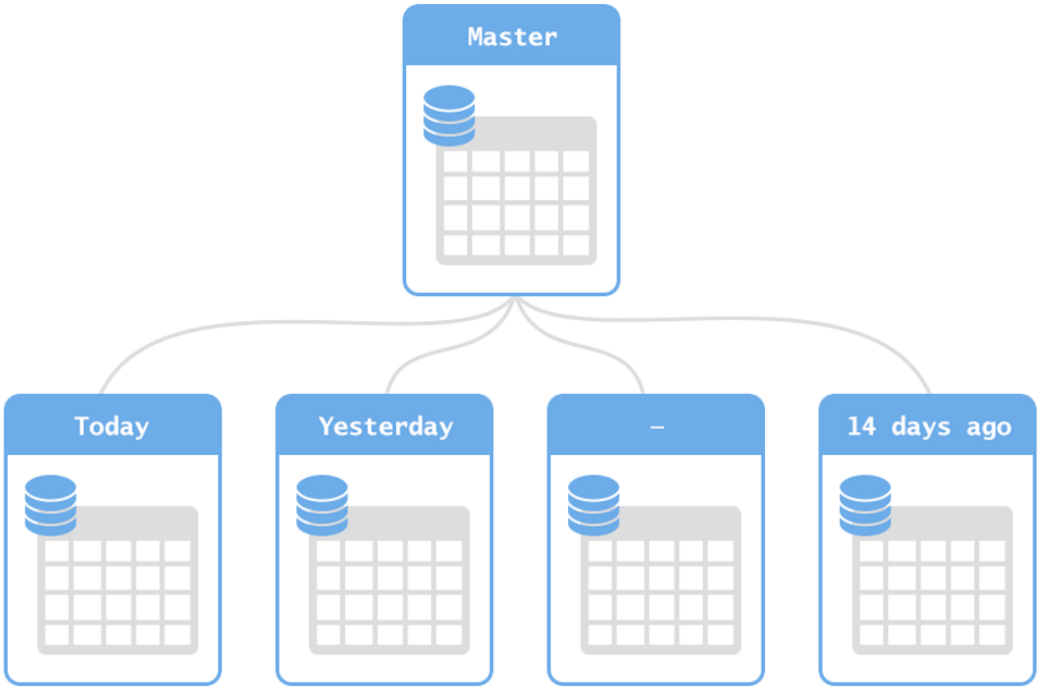

Рисунок 2.3 – Схема секционирования таблицы базы данных [сделано автором]

Секционирование (партиционирование) – это вертикальный шардинг (от англ. sharding (shard) – сервер), основывается на разделении одной большой таблицы на множество маленьких на том же сервере. Применяем данную технику к таблице БД больших данных mus_service log и получаем схему, указанную на рисунке 2.4.

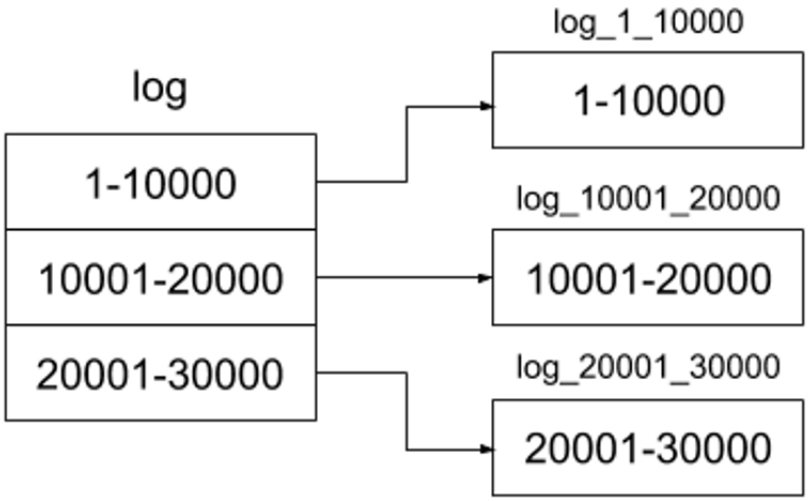

Рисунок 2.4 – Секционирование (вертикальный шардинг) применимо к таблице log [сделано автором]

Преимущества секционирования приведены ниже.

- В определённых ситуациях секционирование кардинально увеличивает быстродействие, особенно когда большой процент часто запрашиваемых строк таблицы относится к одному или небольшому числу секций. Секционирование может сыграть роль ведущих столбцов в индексах, что позволит уменьшить размер индекса и увеличит вероятность нахождения наиболее востребованных частей индексов в памяти.
- Когда в выборке или изменении данных задействована большая часть одной секции, последовательное сканирование этой секции может выполняться гораздо быстрее, чем случайный доступ по индексу к данным, разбросанным по всей таблице.

В данном случае разбиение таблицы происходит в одной базе mus_service на том же самом сервере с репликацией 8003 (master) и 7003 (slave), что нам не подходит из-за остающейся нагрузки на БД

Переходим к другой технике масштабирования данных – горизонтальному шардингу. Горизонтальный шардинг работает так же, как и вертикальный, с единственным отличием – таблицы находятся в разных базах и на разных шардах [15].

Применяем горизонтальный шардинг к нашей структуре и получаем схему, показанную на рисунке 2.5.

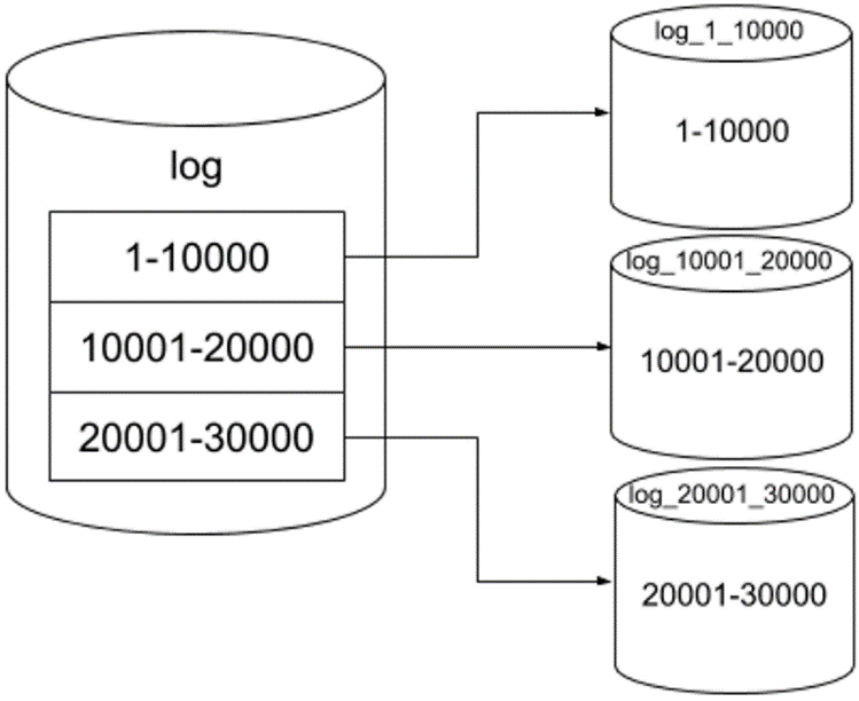

Рисунок 2.5 – Горизонтальный шардинг таблицы log [сделано автором]

Горизонтальный шардинг исправляет не желаемое для нас размещение совокупности таблиц log в одной БД и на одном сервере.

Для того, чтобы выбрать нужные данные из секций их нужно перебрать, что не подходит для аналитики логов пользователей, поэтому продолжаем поиски решения. Необязательно хранить все логи в таблице log на 8003, куда постоянно пишутся данные. На основе аналитического ПО выявлен период в 8 дней, за который данные логов должны хранится в основной таблице. Остальные данные можно архивировать в другой БД и на другом сервере. Такое решение обладает следующими преимуществами:

- снижает нагрузку на основную таблицу;
- аналитическому ПО будет известно место хранения данных.

Вернёмся к основам секционирования – деления основной таблицы на множество небольших (см. рисунок 2.3.) Благодаря этому, данные можно хранить по месяцам, т.е. в каждой таблице будут хранится данные, соответствующие определенному месяцу. Но есть недостатки, первый – это именование таблицы год и месяц, что не очень явно, второй, и главный недостаток, это то, что в одной БД будут инкрементивно добавляться таблицы и, к примеру, в 10-ый год их уже будет 10*12 = 120.

Здесь можно прибегнуть к особенности РСУБД PostgreSQL и обратиться к схемам БД. Каждому году будет соответствовать схема, а каждому месяцу таблица, таким образом, архивные данные за май 2019 года будут прозрачно находится в log_2019.log_05

Приступаем к реализации и чертим схему и выбираем свободный сервер – 6136 (см. рисунок 2.6).

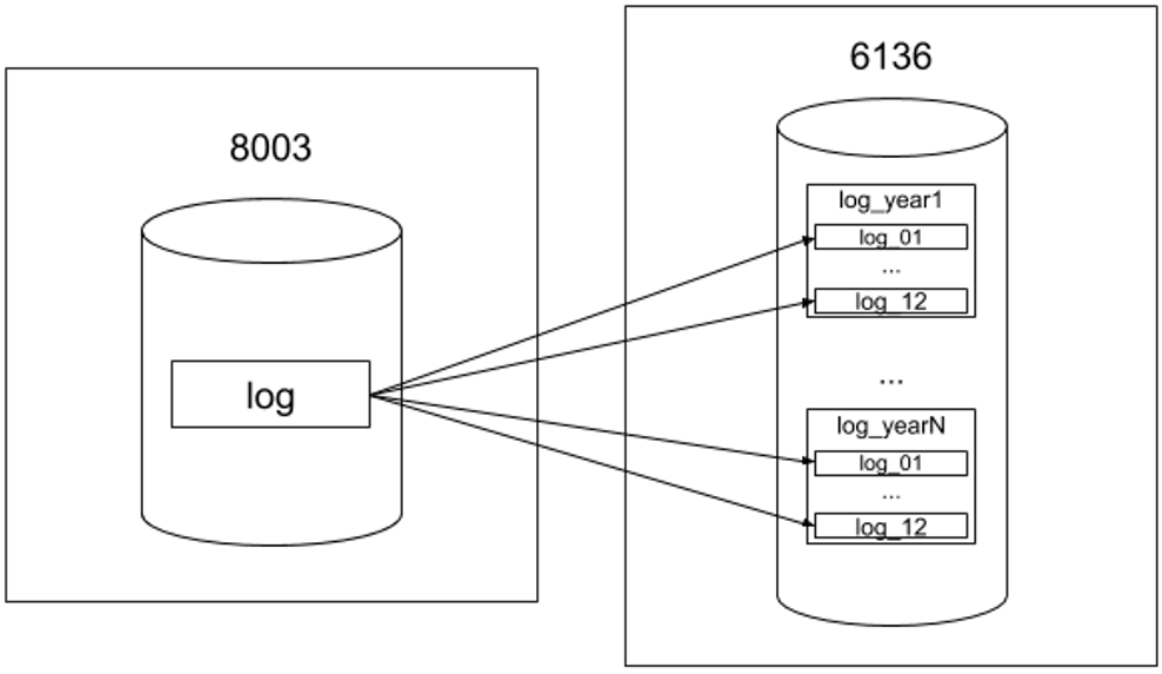

Рисунок 2.6 – Архивирование данных log методом ручного секционирования [сделано автором]

В результате получаем основную таблицу, содержащую в себе записи за 8 дней, и множество таблиц с архивными логами на другом сервере (6136) и собранными в одну БД. Полученное ручное секционирование данных способствует стабильной непрерывной записи логов и чтению их, как их главной таблицы, так и из архивных таблиц с отсортированными данными по годам и месяцам, что создает удобство и прозрачность в использовании данных для аналитики. В связи с тем, что веб-сервис написан языке PHP, реализация архивирования данных была реализована на нем [18].

Архивация запускается по crontab задаче ежедневно два раза в день: 0 7,23 * * * vps8002 archive.php. На рисунке 2.7 представлен фрагмент архиватора (ручного секционирования).

```php
class ArchiveLog extends Daemon_Daemonizeable {
    public function __construct($log, $config, $facade) {
        $this->log       = $log;
        $this->config    = $config;
        $this->facade    = $facade;
        $this->countData = 0;
        $this->curTime   = date('Y-m-d H:i:s');
        $this->data      = null;
        $this->delete    = null;
    }
    public function run() {
        file_put_contents(GIRAR_DATA_DIR . $this->config->startTimeArchiveFile, $this->curTime);
        while($this->data = $this->facade->getLogOldData($this->config->num, $this->curTime))
        {
            try {
                $this->facade->startLogTransact();
                $this->facade->startArchiveTransact();

                $this->facade->insertArchiveOldData($this->data);
                $this->delete = $this->facade->deleteLogOldData($this->data);
                $this->countData += $this->delete;

                $this->facade->commitArchiveTransact();
                $this->facade->commitLogTransact();
            } catch (Exception $e) {
                $this->facade->rollBackArchiveTransact();
                $this->facade->rollBackLogTransact();
                $this->log->err($e);
            }
        }
        $this->log->info('Count archived data: ' . $this->countData);
        return false;
    }
}

$daemon = new Daemon_Daemon(
    new ArchiveLog($log, $config, $facade),
    GIRAR_DATA_DIR . $config->pidFile
);
$daemon->summon(false);
```

Рисунок 2.7 – Листинг реализации на языке PHP ручного секционирования (архивирования) данных [сделано автором]

Пока метод getLogOldData будет возвращать данные, будет выполняться тело цикла while.

Тело цикла whileобернуто в перехватчик ошибок/исключений try-catch:

- В теле trysначала с помощью методов посредника $facade открываются транзакции для текущих данных startLogTransact() и для архивных startArchiveTransact();
- Далее, внутри транзакции в архив записываются полученные в заголовке цикла данные через метод insertArchiveOldData();
- После, из текущей таблицы логов удаляются полученные ранее старые данные с помощью метода deleteLogOldData();
- Производится увеличения счётчика архивируемых данных $countData;
- Транзакции отправляются в обратном порядке открытию, сначала архивная commitArchiveTransact(), а после текущая commitLogTransact();
- В теле catch в случае возникновения ошибки выше происходит откат изменений в обратном порядке открытию транзакции, сначала архивная rollBackArchiveTransact(), а после текущая rollBackLogTransact() и в конце отловленная ошибка записывается в лог файл;
- После выхода из цикла whilев лог записывается количество архивированных данных полученное с помощью счётчика $countData.

В первую очередь приступим к созданию таблицы stop_list для предотвращения предоставления услуги веб-сервиса девиантным (нелояльным) пользователям. Таблица будет иметь структуру, указанную в таблице 2.2.

Таблица 2.2 – Структура таблицы stop_list БД mus_service [сделано автором]

Колонка	Тип поля	Описание
abonent_id	Bigint	Идентификатор полученный от оператора
till_date	timestamp	Дата конца блокировки предоставления услуг

Прежде чем предоставлять услугу пользователю, проверяем, находится ли он в таблице stop_list, если да, то отменяем предоставление [14,18].

Фрагмент программы, предоставленный на рисунке 2.8, включает:

- Перед предоставление услуги вызывается метод checkCurrentStopList() с двумя параметрами $raiseExceptionи $type;
- Далее происходят проверки на установку, в том числе и проверка самого stop_list, в случае если пользователь есть в таблице возвращается false, так же установлен лимит на заказы.

```php
public function checkCurrentStopList($raiseExceptions = false, $type = null) {
    if (RbtWebform_Helper_Test::getInstance()->hasPurchase() !== null) {
        return RbtWebform_Helper_Test::getInstance()->hasPurchase();
    }
    if (RbtWebform_Helper_Test::getInstance()->isTesting()) return false;
    if ($this->checkCacheForPurchase(RbtWebform_Request::getMsisdn(), $type)
    || $this->checkStopList(RbtWebform_Request::getMsisdn(), $type)) {
        $this->setPurchaseCache(RbtWebform_Request::getMsisdn(), $type);
        if (!$raiseExceptions) {
            return true;
        }

        throw new RbtWebform_Exception('Purchase limit exceeded',
        $this->getConfig()->messageTemplate->noDeliverMethod,
        RbtWebform_Response::ERROR_CODE_NOT_TODAY);
    }
    return false;
}
```

Рисунок 2.8 – Листинг программы для проверки пользователя

[сделано автором]

Теперь перейдём к созданной ранее модели и реализованной через значения (баллы) в таблице statec, показанной на Рисунке 2.9.

Файл Правка Вид Инструменты ?

не ограничено

	state_id [PK] serial	short_name character varying(2	description character varying(255)	value real	is_enabled boolean
1	9	score	Скор . .	0.244	TRUE
*					

Рисунок 2.9 – Таблица statec управляющим параметром (баллом)

[сделано автором]

Скоринг – это вероятностное отношение, поэтому необходимо создать управляющий параметр – балл. Балл зависит от приходящих данных от оператора и меняется исходя от количества представленных там пользователей и данных в архивной базе. Изменение балла представлено на рисунке 2.10.

Управляющий параметр (score) имеет вероятностное значение, которое варьируется 0.1 до 0.9.

При реализации ежедневного плана на предоставление услуг (с помощью веб-сервисов), которое в свою очередь базируется на значениях предыдущих дней,

используется данный параметр. К примеру, текущий скор со значение 0.244 является хорошим показателем, т.к. в нашем случае чем меньше значение скора (близко к 0.1), тем лучше. Для того, чтобы было удобно отслеживать эффективность введенной ИС было разработано два представления в компьютерной среде Kibana, которые представлены на рисунках 2.11 и 2.12.

```php
function cheangeScore($delta, $db)
{
    $sth = $db->prepare("
        SELECT value FROM control.state
        WHERE state_id=9
    ");
    $sth->execute();
    $currentScore = $sth->fetch();
    echo "Current value of score is $currentScore[value] \n";
    $newScore = $currentScore['value'] + $delta;
    if ($newScore > 0.1 && $newScore <= 0.7){
        $params = array(
            ':value'    => $newScore);
        $sql = 'UPDATE control.state SET value = :value WHERE state_id = 9';
        $sth = $db->prepare($sql);
        $res = $sth->execute($params);
        return $newScore;
        }
    else
        return $currentScore['value'];
}
$ds = new Datasource_Database(Datasource_PostgreSQL::RBT_WEBFORM_POOL);
$db = new PDO($ds->getConnectionString('pgsql'), $ds->getUser(), $ds->getPass());
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
//
//
//
//
$sth = $db->prepare("
SELECT value FROM control.state
WHERE state_id=4
");//На сколько открыт
$sth->execute();
$traff = $sth->fetch(); //
$delta = 0;
if ($traff['value'] > 0.9 && $traff['value']< 1) //
    echo "\nMaximum traffic is used \n";
else { //
    $delta = controlScore($traff); //Определяем слагаемое
    $newScore = cheangeScore($delta, $db); // Меняем значение
    echo "New value of score is $newScore \n";
}
```

Рисунок 2.10 – Листинг реализации модели по определению score (управляющего параметра) [сделано автором]

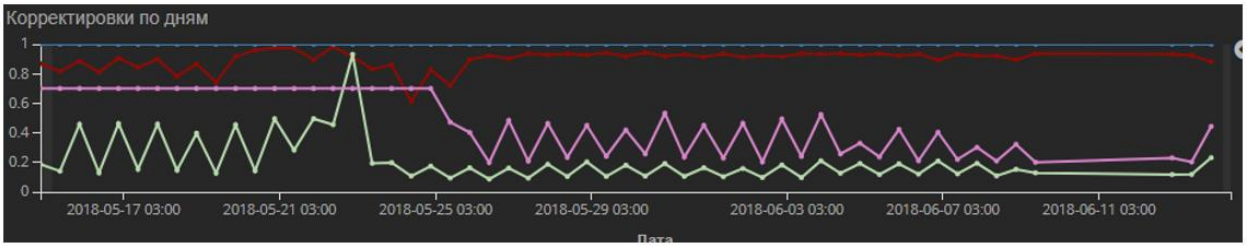

Рисунок 2.11 – Панель kibana с месячным отслеживанием управляющих параметров score [сделано автором]

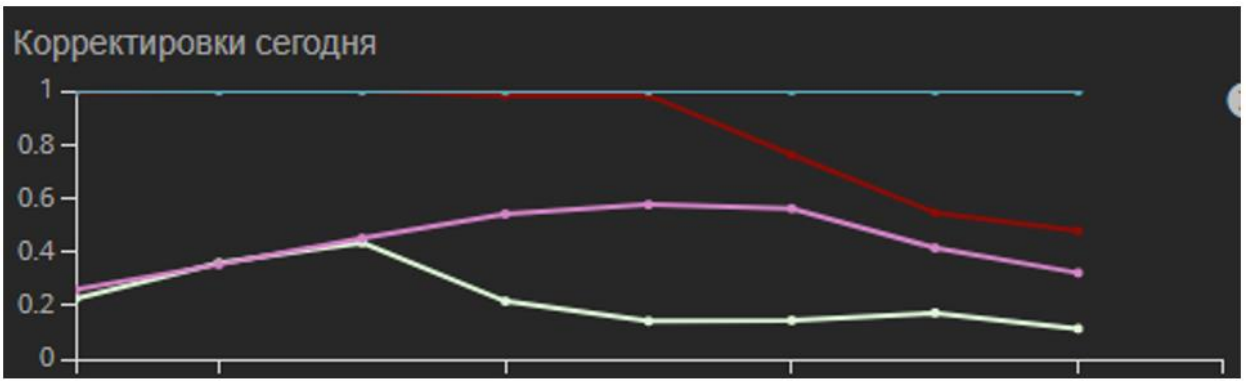

Рисунок 2.12 – Панель kibana с суточным отслеживанием управляющих параметров score [сделано автором]

На Рисунках 2.11 и 2.12 представлены панели с графиками, по оси *X*– время, по оси *Y* – значение управляющего параметра. Розовая линия – линия изменений управляющего параметра скоринга, остальные определяют, сколько раз предоставить услугу, предоставляемую веб-сервисом, в момент времени (в час, на большом графике в день).

Таким образом, были определены недочёты в системе хранения основной и вторичной информации ИС и после устранены путем добавления следующих архитектурных решений в области работы с данными:

- добавлена репликация, повысившая отказоустойчивость прикладного веб-сервиса;
- добавлено секционирование для уменьшения времени отклика и стабильности работы веб-сервиса.

Также на основании ранее определенных показателей лояльного и нелояльного к предоставляемым услугам пользователя была создана ИС по обработке этих показателей (логов пользователей) и добавлена в систему, определяющую количество предоставления услуг в сутки. Проведена оценка эффективности предлагаемых решений путем моделирования графиков зависимостей управляющих параметров от времени и числа предоставляемых услуг.

3. Разработка опытного образца программного приложения с использованием технологии блокчейн для обеспечения «прозрачности» больших данных

Для разработки опытного образца программного приложения с использованием технологии блокчейн необходимо проанализировать средства реализации основного принципа технологии blockchain, который заключается в использовании цифровых документов со штампом времени, чтобы их невозможно было оформить задним числом или подделать. Технологию предложили 1991 году ученые-исследователи Стюарт Хабер и У. Скотт Шторнетта. Blockchain – это тип технологии распределенной структуры, «книги». (Distributed Ledger Technology). Всемирный банк определяет его как «новый и быстро развивающийся подход к записи и обмену данными между несколькими хранилищами данных (или книгами). Эта технология позволяет записывать, совместно использовать и синхронизировать транзакции и данные в распределенной сети различных участников сети». Стоит отметить также, что единого, строгого определения блокчейна не существует. Подробно эту тему освещал в своих трудах Андрианн Джефрис [15]. Автор пишет о том, что блокчейн следует рассматривать как распределенную временную структуру данных только с присоединением.

Изначально технология блокчейн (blockchain) известна как технология, на базе которой действует криптовалюта – с 2008 года на ее базе осуществлялись взаиморасчеты Биткойн. Технология активно внедряется в финансовой сфере Англии, Японии, США, Китая и других стран. В России ПАО Сбербанк активно продвигает данную технологию.

Технология блокчейн нашла широкое применение при покупке недвижимости, ценных бумаг, получении кредита, и прочее, когда данные об этих операциях размещаются и хранятся на серверах разных организаций.

Технология блокчейн использует децентрализацию для поддержания безопасности системы. Централизованная система, какой бы безопасной она ни была, все равно подвержена воздействию «человеческого фактора», в то время как децентрализованная система не имеет общего центра управления и фактического администратора системы. Даже при условии получения доступа к одному из блоков системы, реализовать преступный замысел злоумышленник не сможет.

Злоумышленники могут получить доступ к базам данных на серверах и внести свои коррективы, что повлечет за собой финансовые правонарушения. С помощью технологии блокчейн данные о финансовых операциях, выраженные в цифровой форме, объединяются в хронологическую цепочку, которая потом проверяется с помощью персональных компьютеров и сетевых технологий присваивания уникальной электронной

подписи и минимизации воздействия «человеческого фактора», поскольку большинство информационных систем, построенных на основе технологии блокчейн, требуют подтверждения внесения данных другими участниками. Таким образом обеспечивается «прозрачность» данных. Криптография обеспечивает устойчивость системы к взлому и влиянию извне, также и определенную анонимность, чтобы личные данные об участниках системы и их операциях не появлялись в свободном доступе, и определенную прозрачность, чтобы в кратчайшие сроки пресекать любые нарушения. Применение блокчейн в финансовой сфере имеет значительный потенциал, который все еще находится в нераскрытом состоянии.

Блокчейны позволяют реализовать распределенную одноранговую сеть, в которой не доверяющие участники могут безопасно взаимодействовать с каждым из них без необходимости наличия доверенного органа. Чтобы достичь этого, нужно организовать блокчейн как набор взаимосвязанных механизмов, которые предоставляют специфические функции для инфраструктуры, как показано на Рисунке 3.1.

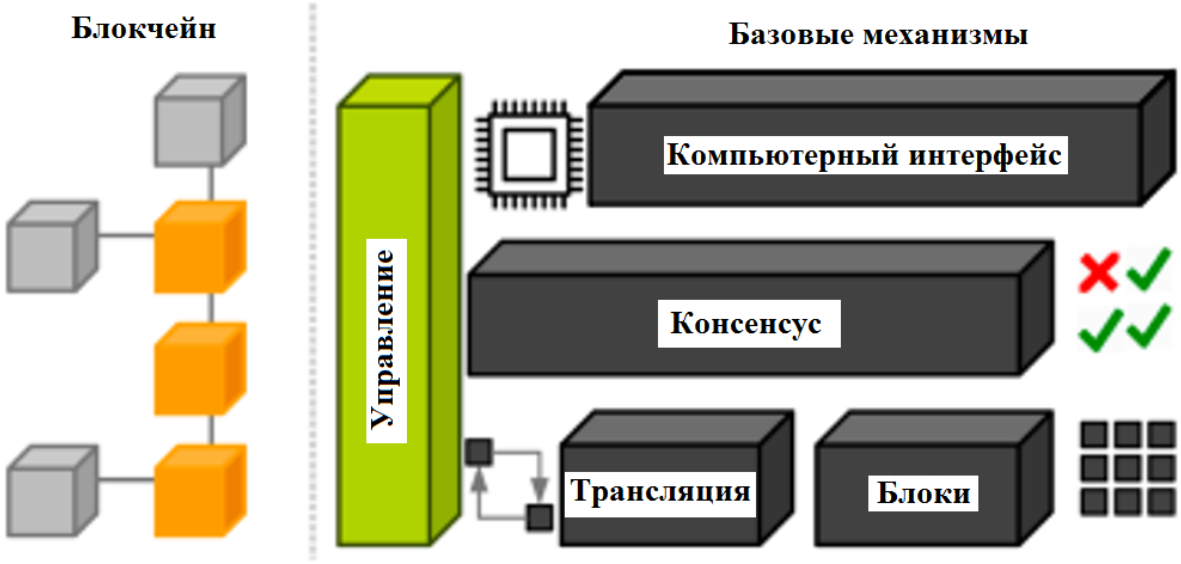

Рисунок 3.1 – Архитектура технологии блокчейн [20]

На самом нижнем уровне инфраструктуры, приведенной на рисунке 3.1, имеются подписанные транзакции между узлами. Эти транзакции означают соглашение между двумя участниками, которое может включать передачу физических или цифровых активов, выполнение задачи и т. д. По крайней мере один участник подписывает эту транзакцию, и она распространяется среди ее соседей. Как правило, любой объект, который подключается к блокчейну, называется узлом. Однако узлы, которые проверяют все правила блокчейна, называются полными узлами. Эти узлы группируют транзакции в

блоки, и они несут ответственность за определение того, являются ли транзакции действительными и должны ли они храниться в цепочке блоков, а какие нет.

В зависимости от типа блокчейна существуют разные механизмы консенсуса [21]. Наиболее известным является PoW (от англ. Proof-of-Work – доказательство работы). PoW требует решения сложного вычислительного процесса, такого как поиск хешей с конкретными шаблонами, например, ведущее число нулей, чтобы обеспечить аутентификацию и проверяемость. Вместо разделения блоков пропорционально относительным показателям хэширования майнеров (то есть их мощности майнинга), протоколы PoS (от англ. Proof-of-Stake – доказательство доли) делят блоки пропорционально текущему богатству майнеров. Таким образом, выбор является более справедливым и препятствует доминированию самого богатого участника в сети [22]. Многие блокчейны, такие как Ethereum, постепенно переходят на PoS из-за значительного снижения энергопотребления и улучшения масштабируемости, о чем писал в своей работе Крис Данен [23].

Современная литература классифицирует блокчейн-сети несколькими способами (В. Бутерин, 2015 [24]; Eris Industries, 2016 [25] и др). Эти категории формируются в соответствии с управлением и разрешениями сети как общедоступные, частные и федеративные.

В общественных блокчейнах (permissionless) любой желающий может присоединиться в качестве нового пользователя. Кроме того, все участники могут выполнять такие операции, как сделки или контракты. В частных блокчейнах, которые наряду с федеративными относятся к разрешенной категории blockchain, как правило, белый список разрешенных пользователей определяется с определенными характеристиками и разрешениями на сетевые операции. Федеративный блокчейн представляет собой гибридную комбинацию публичных и частных блокчейнов [26]. Хотя он имеет схожий уровень масштабируемости и защиты конфиденциальности с private blockchain, их основное отличие заключается в том, что вместо одного объекта для проверки процессов транзакций выбирается набор узлов, называемых узлами-лидерами. Это позволяет частично децентрализовать конструкцию, в которой узлы-лидеры могут предоставлять разрешения другим пользователям.

В таблице 3.1 обобщены основные характеристики каждой блокчейн-сети, касающиеся эффективности, безопасности и механизмов консенсуса.

Известные реализации публичных блокчейнов включают Bitcoin, Ethereum, Litecoin и, в целом, большинство криптовалют. Одним из их главных преимуществ является

отсутствие затрат на инфраструктуру: сеть является автономной и способна поддерживать себя, резко снижая накладные расходы на управление.

Таблица 3.1 – Классификация и основные характеристики блокчейн-сетей [27]

Property	Public	Private	Federated
Consensus Mechanism	• Costly PoW • All miners	• Light PoW • Centralised organisation	• Light PoW • Leader node set
Identity Anonymity	• (Pseudo) Anonymous • Malicious?	• Identified users • Trusted	• Identified users • Trusted
Protocol Efficiency & Consumption	• Low efficiency • High energy	• High efficiency • Low energy	• High efficiency • Low energy
Immutability	• Almost impossible	• Collusion attacks	• Collusion attacks
Ownership & Management	• Public • Permissionless	• Centralised • Permissioned whitelist	• Semi-Centralised • Permissioned nodes
Transaction Approval	• Order of minutes	• Order of milliseconds	• Order of milliseconds

В частных блокчейнах основными приложениями являются управление базами данных, аудит и, в целом, решения, требующие высокой производительности. Multichain является примером открытой платформы для построения и развертывания частных блокчейнов.

Наконец, федеративные блокчейны в основном используются в банковском и промышленном секторах. Это относится к проекту Hyperledger [28], который разрабатывает межотраслевые блокчейн-фреймворки на основе разрешений.

Количество криптовалют иллюстрирует важность блокчейна. Большие темпы роста вскоре могут создать проблемы совместимости из-за неоднородности криптовалютных приложений. Кроме того, ландшафт быстро развивается, поскольку блокчейн используется в других областях за пределами криптовалют, при этом смарт-контракты (Smart-contracts – SCs) играют центральную роль. Смарт-котракт, определенный в Ником Сабо в 1994 году как «компьютеризированный протокол транзакций, который выполняет условия контракта», позволяет нам переводить договорные положения в встраиваемый код, тем самым минимизируя внешнее участие и риски [29] Таким образом, SC - это соглашение между сторонами, которые, хотя и не доверяют друг другу, автоматически выполняют согласованные условия. Таким образом, в контексте блокчейн SCs - это сценарии, выполняемые децентрализованным образом и хранящиеся в блокчейне, не полагаясь на какие-либо доверенные полномочия. В частности, системы на основе блокчейна, поддерживающие SCs, позволяют более сложные процессы и взаимодействия, поэтому они устанавливают новую парадигму с практически безграничными приложениями [30].

Таким образом, блокчейн – это своего рода общая база больших данных, содержание которой проверяется и согласовывается сетью независимых субъектов. Для того, чтобы в блокчейн был добавлен новый блок данных (например, новый владелец переданного имущества), независимые верификаторы должны прийти к единому мнению о его надежности.

Поскольку каждый новый набор транзакций («блок») криптографически связан с предыдущим блоком, крайне сложно изменить данные, хранящиеся в блокчейне, и любое такое изменение будет легко обнаружено. Блокчейны широко признаны неизменяемыми и могут служить доказательством права собственности.

В качестве языка разработки для реализации технологии блокчейн был выбран Solidity – JavaScript-подобный объектно-ориентированный язык для разработки смарт-контрактов (среди прочих языки Serpent и Mutan). Является кроссплатформенным. Один из четырех языков для EVM (от англ. Ethereum Virtual Machine – виртуальная машина платформы Эфириум). Платформа Эфириум предназначена для создания децентрализованных онлайн-сервисов на базе блокчейна.

Средой разработки выступает Remix IDE. Интерфейс среды разработки и команды, реализующие подключение расширения для браузера Google Chrome с целью доступа к распределенным приложениям с поддержкой Ethereum приведены на Рисунке 3.2.

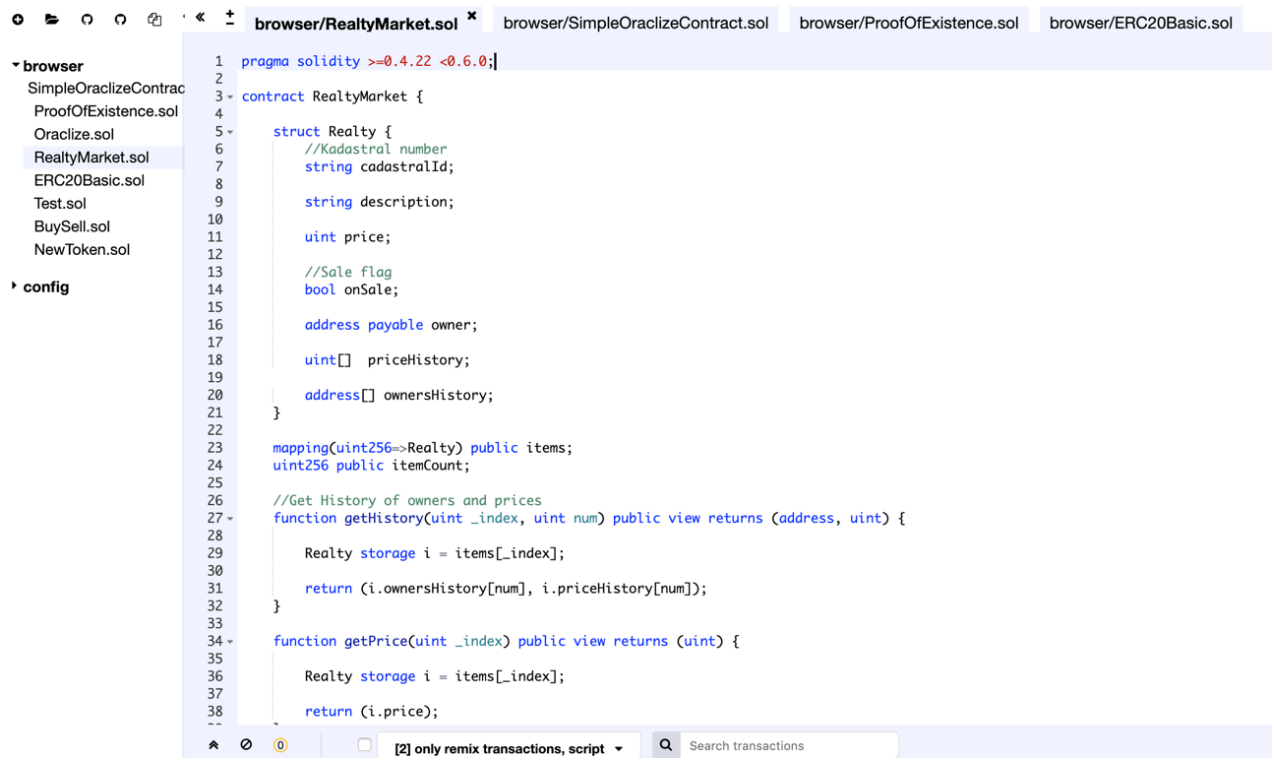

Рисунок 3.2 – Интерфейс среды Remix [сделано автором]

Расширение Metamask внедряет Ethereum web3 API (от англ. Application Programming Interface – Интерфейс Программирования Приложений) в контекст javascript каждого веб-сайта, чтобы Dapp (распределенноое приложение) мог читать из блокчейна. MetaMask, также позволяет пользователю создавать и управлять своими собственными удостоверениями, поэтому. Когда Dapp хочет выполнить транзакцию и выполнить запись в блокчейн, пользователь получает безопасный интерфейс для проверки транзакции перед ее одобрением или отклонением.

В работе расширение выступает в роли личного кабинета пользователя – здесь отображаются баланс и совершенные транзакции (см. Рисунок 3.3). Аналогом логина в сети блокчейн выступает так называемый публичный ключ, являющийся адресом, который участвует в транзакциях. Тестирование и отладка производится в тестовой сети Ropsten.

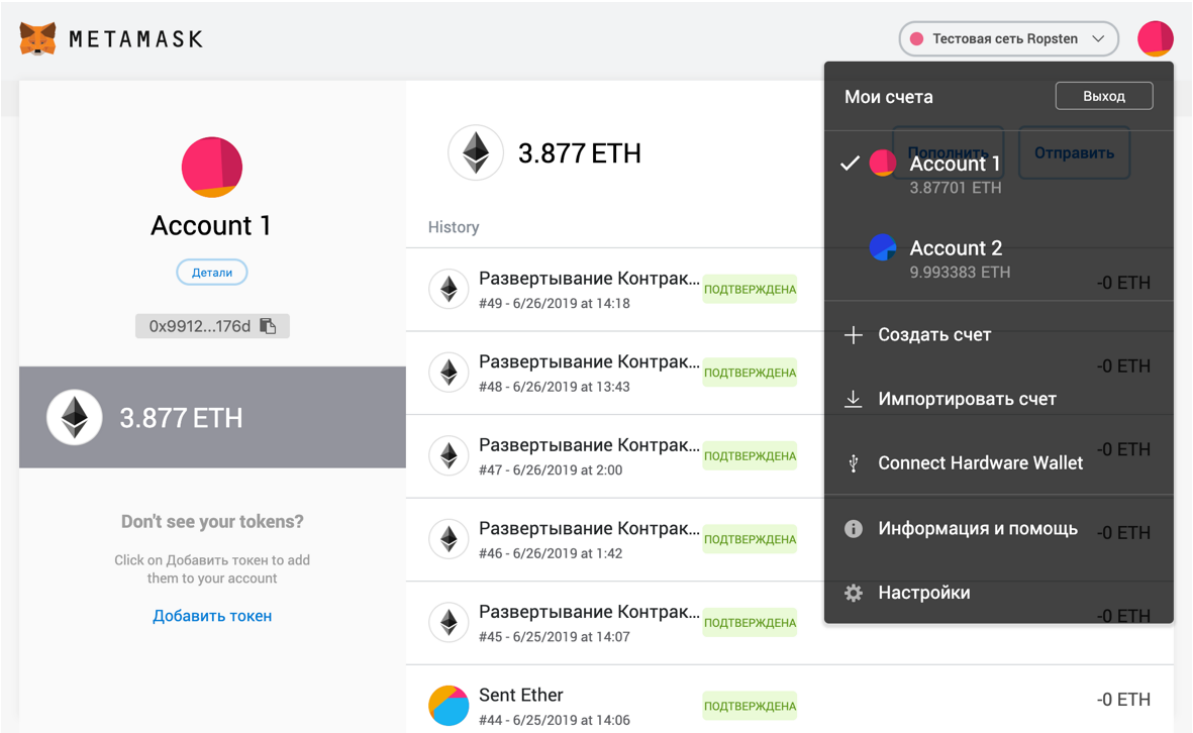

Рисунок 3.3 – Личный кабинет Metamask [сделано автором]

Источник: составлено автором работы

Разработана модель, которая показывает принцип работы децентрализованной платформы по продаже недвижимости.

Распределенное программное приложение должно выполнять следующие функции.

1. Регистрация пользователя в системе. В сети блокчейн доступ к «аккаунту» осуществляется при помощи пары ключей публичный-приватный. Публичный ключ будет являться уникальным идентификатором пользователя в реестре.

2. Регистрация в реестре объекта недвижимости. У каждого объекта так же имеется свой уникальный идентификатор (кадастровый номер) и набор характеристик. При регистрации объекта недвижимости в реестре будет происходить проверка информации посредством Oraclize через API Росреестра.

3. Хранение истории всех операций по одному конкретному пользователю. Таким образом у каждого участника реестра будет свой «рейтинг», на основе которого другие участники могут решить, с кем предпочтительнее провести сделку.

4. Также в блокчейне будет храниться информация по всем произведенным сделкам и изменениям с каждым объектом недвижимости, такими как снятия/выставление на продажу, смена владельца, продажа и т.д.

5. Цифровая передача права собственности в реестр.

Каждая собственность имеет уникальный сертификат о собственности, который служит доказательством владения этой собственностью. Владение недвижимостью обычно отслеживается путём записи таких сертификатов о собственности в организованных реестрах, которые управляются правительственными организациями.

Кульминацией сделки является цифровая передача права собственности в реестр.

Распределенное приложение состоит из нескольких смарт-контрактов, которые взаимодействуют друг с другом и следуют методу архитектуры микросервисов. Каждый контракт несёт ответственность за один вид записи в системе. Каждый контракт содержит функции, которые позволяют создавать и изменять записи, обновлять контракты и другие административные функции. Перечень контрактов приведен в таблице 3.2.

В качестве контракта, отвечающего за сертификат собственности выбран, применяется токен стандарта ERC-721 (от англ. Ethereum Request for Comments – официальный протокол для внесения предложений по улучшению сети Ethereum). Этот стандарт позволяет реализовать стандартный API для несовместимых токенов (далее называемых «NFT» (от англ. NewFoundland Time – Время Ньюфаундленда) в рамках смарт-контрактов. Этот стандарт обеспечивает базовые функции для отслеживания и передачи права собственности на NFT.

Основой этого стандарта является то, что каждый NFT идентифицируется уникальным 256-битным беззнаковым целым номером в рамках своего контракта на отслеживание. Этот идентификационный номер не должен меняться в течение срока

действия контракта. Эта пара (адрес контракта, идентификатор актива) будет уникальным идентификатором для конкретного NFT в экосистеме Ethereum.

Таблица 3.2 – Характеристики контрактов для микросервисов распределенных приложений [сделано автором]

Название контракта	**Функции контракта**
Ownership	Контракт сертификата о собственности (Отвечает за хранение и обновление метаданных на блокчейне). Содержит информацию собственнике и объекте владения. Представляет собой токен стандарта ERC-721.
Tranfer	Контракт документа о передаче права собственности. Отслеживает и инициирует приглашение участников транзакции. Содержит информацию о продавце, покупателе, текущем статусе сделки.
Identity	Контракт идентификации пользователя (Хранит множество записей с информацией о идентификации для всех пользователей системы. Содержит функции для проверки личности физического или юридического лица).

На блок-схеме Рисунка 3.4 представлена архитектура и взаимодействие с внешним Реестром (ЕГРН). На Рисунке 3.5 отображены бизнес-процесс покупки недвижимости в приложении.

В перспективе, платформа будет представлять онлайн рынок недвижимости, позволяющий покупателям, продавцам и брокерам объединиться путем использования набора смарт-контрактов для облегчения транзакций. Система предоставит этим лицам сеть для взаимодействия друг с другом и совершения онлайн покупок недвижимости Peer-to-Peer.

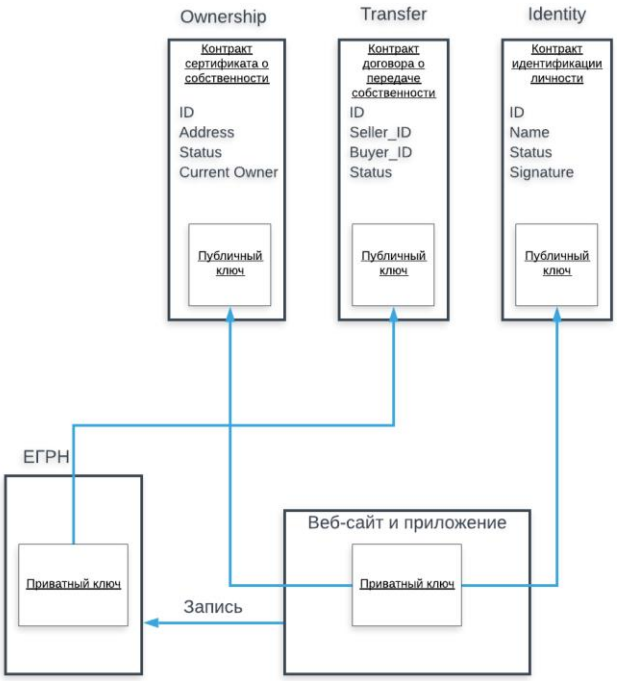

Рисунок 3.4 – Архитектура и взаимодействие с внешним Реестром (ЕГРН) [сделано автором]

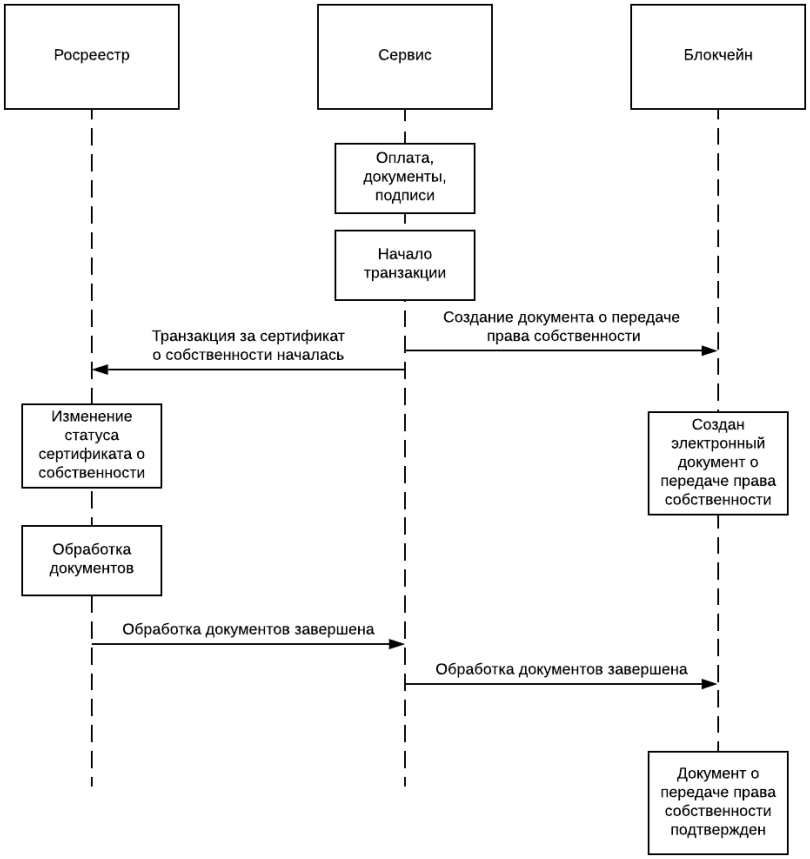

Рисунок 3.5 – Схема бизнес-процессов покупки недвижимости в приложении [сделано автором]

Однако существуют некоторые допущения, которые необходимо учитывать. Ниже представлен краткий обзор предварительных условий для становления на основе созданного программного приложения системы в качестве глобального реестра информации о владениях сертификатами о собственности, который будет доступен для всех участников аналогично системе DNS для доменов веб-сайтов:

1. Цифровая запись в публичном блокчейне должна быть принята в качестве юридически обязывающей.
2. Система идентификации должна быть адаптирована, чтобы она могла ассоциировать личностей с конкретным адресом блокчейна (eID карты с биометрическими данным как один из вариантов).
3. Каждое государственное учреждение, участвующее в совершении сделок в сфере недвижимости, должно иметь API, основанное на смарт-контрактах (Особенно важно для учреждений, отвечающих за оценку и сбор налогов на имущество).
4. Все записи из земельного реестра недвижимости должны быть перемещены на блокчейн.

Технологии больших данных и блокчейн повысят эффективность современных инфокоммуникационных систем [31]. Разработанный опытный образец программного приложения, реализующий принцип работы децентрализованной платформы на основе технологии блокчейн по продаже недвижимости, обеспечивает надежное и прозрачное хранение всей истории транзакций, а также целостность информации. Технология больших данных для обработки сведений о продажах недвижимости предоставляет инструменты для эффективного анализа и прогнозирования.

4. Расчет параметров адаптивной системы поддержки принятия решений для управления обработкой больших объемов данных

Для того чтобы рассчитать параметры СППР, необходимо выявить математические законы, в соответствии с которыми идет управление данными [32]. Элементы обслуживания (ЭО), используемые в СППР, делятся на аппаратные и программные средства. Аппаратные: серверы баз данных и знаний, облачных технологий и др.; рабочие станции в виде персональных компьютеров; интерфейсное оборудование для связи рабочих станций и удаленного управления; устройства, из которых состоит компьютер (центральный процессор, оптические считывающие устройства, накопители на магнитной ленте, дисководы, принтеры, терминалы и т. п.). Программные средства: системные программы (операционные системы, системы управления базами данных, адаптивные базы знаний, сервисные программы); прикладные (пользовательские) программы (наборы машинных команд для обработки данных). Кроме того, в СППР должны участвовать: персонал (работники, которые управляют системой) и процедуры контроля (обеспечивают соответствующую запись операций, предупреждающие или регистрирующие ошибки) [33].

Все перечисленные ЭО и операции с ними представляют собой типичные функции, реализованные дискретными процессами. Таким образом, компьютерные системы и каналы связи для передачи результатов вычисления при удаленном управлении можно представить в виде сетей одно- и многоканальных *систем массового обслуживания* (*СМО*) с отказами и различными дисциплинами обслуживания [34]. Например, «первым пришел, первым обслужен» с заранее заданным входным потоком заявок. Таким образом, при анализе компьютерных систем, сетей передачи информации, баз и банков данных, баз знаний, на которых основаны СППР, особое значение приобрели СМО [35].

Для разработки модели дискретной событийной системы был учтен опыт формирования моделей разных типов дискретных событийных систем, который свидетельствует о том, что приблизительно 80 % таких моделей основаны на СМО [35,36]. Адаптивные системы занимают промежуточное положение между системами *M*/*M*/1/100 (входящий поток требований является Пуассоновским (*M*); распределение времени на обслуживание по экспоненциальному закону (*M*); один ЭО (1), практически бесконечная очередь (100)) и *M*/*M*/3 (три ЭО, нет очередей), обеспечивая максимально сбалансированное и эффективное функционирование системы [37 – 39].

Для построения имитационной модели адаптивной СППР и оценки ее параметров предлагается использовать программную среду GPSS (General Purpose Simulation System)

World Student Version – комплексный моделирующий инструмент, охватывающий области как дискретного, так и непрерывного компьютерного моделирования, обладающий высочайшим уровнем интерактивности и визуального представления информации [40]. Предполагается, что в СППР поступает информация, содержащая сведения от различных входных источников, которые и записываются в базы данных. Информация поступает по каналам связи по запросам поста контроля системы. В теоретической модели СМО все запросы, поступающие в систему (функционирует более одного ЭО), становятся в одну общую очередь, без какой-либо привязки к конкретному элементу. В случае использования *имитационной* модели запрос после поступления в систему сразу прикрепляется к конкретному ЭО, и могут возникнуть ситуации, при которых запрос, поступивший в систему позже предыдущего, получает обслуживание раньше.

Целевую функцию СППР с ЭО, отражающую эффективность использования информационного ресурса, представим следующим образом:

$$M = f(\texttt{AVE.CONT; AVE.TIME; AVE.(-0); MEAN; STD.DEV;}$$
$$\texttt{POTERI,OTKAZ; SR\_DL\_OCH; SUMMA}) \rightarrow \max, \qquad (1)$$

где: AVE.CONT – средняя длина каждой очереди;

AVE.TIME – среднее время нахождения запросов в каждой из очередей;

AVE.(-0) – среднее время нахождения запросов в каждой из очередей без учета тех запросов, которые присоединились к очереди и сразу ее покинули, т.е. время ожидания для них равно нулю;

MEAN – среднее время нахождения запросов в системе;

STD.DEV – среднеквадратичное отклонение;

POTERI,OTKAZ – число и процент потерянных (получивших отказ в обслуживании) запросов;

SR_DL_OCH – средняя суммарная длина очереди;

SUMMA – среднее число задействованных (используемых) ЭО.

Для повышения эффективности используемого ресурса величины, используемые в формуле (1), должны быть минимизированы.

Исследуемая система имеет следующие ограничения: число постоянно функционирующих ЭО – 1; число (лимит) резервных ЭО – 2; максимальное число (лимит) ЭО – $E_{\max} = 3$; максимальное число запросов в каждой из очередей, включая обслуживаемые в настоящее время запросы (при превышении порога включается резервный ЭО) – 3; отказы в обслуживании при отсутствии мест в системе – да; поток

входящих запросов – пуассоновский; нагрузка на систему – Ψ = [0; 5]; время моделирования – 720 мин (12 ч); число осуществляемых прогонов модели – 1000.

Разработанный в соответствии с математической моделью программный код снабжен подробными комментариями, которые позволяют пользователю легко разобраться в особенностях его работы и изменять его параметры (установленные по умолчанию) на требуемые. Фрагмент программного кода приведен на рисунке 4.1.

```
MAX_KL     EQU  2
;Длина очереди (без учета обслуживаемого запроса), ; должна быть >=1
NAK_       STORAGE    1000
;Сбор данных о среднем числе запросов в системе
TIME_TABLE (P9-P8),0,5,20
; Сбор данных о времени пребывания запросов в системе
.....................................................................
VSE_       ADVANCE    (Exponential(1,0,6))
; Интенсивность обслуживания запросов в ЭО
           RELEASE    P2
; Освобождение ЭО, имя которого указано в параметре 2
           LEAVE NAK_
; Окончание сбора данных
; о среднем количестве запросов в системе
           ASSIGN     9,AC1
; Окончание сбора данных
```

Рисунок 4.1 – Фрагмент программного кода [сделано автором]

В стандартном отчете, который выдается программой по окончании моделирования, содержится большое число статистической информации.

Также данные результаты представлены в виде таблицы и гистограммы. Сохраняемые величины (SAVE VALUE) записываются пользовательские данные.

Формула для расчета вероятности начального состояния системы получена путем установки ограничения на максимально возможное число ЭО в СППР [39], т.е.

$$p_0 = \left\{1 + \sum_{S=1}^{E_{max}} \left(\frac{\Psi^{S-1}}{(S-1)!}\right)^K \left(\frac{\Psi}{S}\right)\left[\frac{(\Psi/S)^K - 1}{\Psi/S - 1}\right]\right\}^{-1}, \quad (2)$$

где S – число ЭО; E_{max} – максимальное число элементов ЭО; Ψ – нагрузка на систему; K – число состояний системы, которое зависит от длины очереди.

Полученный результат p_0 соответствует состоянию системы, когда в ней нет ни одного запроса на обслуживание, т.е. система полностью простаивает.

Определим вероятность наличия в системе хотя бы одного запроса

$$p = 1 - p_0. \tag{3}$$

Вычислим вероятность работы нескольких S (в представленном примере – одного, сразу двух и всех трех) ЭО

$$P_S = p_0 \left(\frac{\Psi^{S-1}}{(S-1)!} \right)^K \left(\frac{\Psi}{S} \right) \left[\frac{\left(\frac{\Psi}{S} \right)^K - 1}{\frac{\Psi}{S} - 1} \right]^{-1}. \tag{4}$$

Значения коэффициентов использования ЭО получены с добавлением ограничения на максимально возможное число ЭО [8], т.е.

$$a_n = \sum_{S=n}^{Emax} P_S = p_0 \sum_{S=n}^{Emax} \left(\frac{\Psi^{S-1}}{(S-1)!} \right)^K \left(\frac{\Psi}{S} \right) \left[\frac{(\Psi/S)^K - 1}{\Psi/S - 1} \right] = 1 - p_0 - \sum_{S=1}^{n-1} P_S. \tag{5}$$

где n – количество запросов.

Запросы получают отказ только при одновременном соблюдении двух условий: 1) все имеющиеся ЭО заняты; 2) все места в очередях к ЭО заняты:

$$p_{\text{отказ}} = \frac{\Psi^{max \cdot K}}{(max!)^K} p_0. \tag{6}$$

Более подробно методика расчета параметров СППР с ЭО проведена в работе [41].

Проанализируем результаты исследования способов оценивания параметров.

В рассматриваемой адаптивной СППР постоянно работает только один, первый ЭО. Значение его загрузки записывается в стандартный отчет имитационной модели (в параметр `UTIL`). Значение данного параметра используется для грубой оценки вероятности работы системы (вероятности того, что в системе находится хотя бы один запрос). Грубой оценка является по той причине, что система может находиться в состояниях, когда первый ЭО окажется свободным, а один или несколько резервных ЭО – занятыми (в них в данный момент могут продолжаться процессы обслуживания запросов).

В таблице 4.1 представлены полученные в результате теоретических расчетов значения вероятности наличия в системе хотя бы одного запроса.

Таблица 4.1 – Значения вероятности наличия в системе хотя бы одного запроса [сделано автором]

Рассматриваемая система	**Нагрузка на систему, Ψ**						
	0	**0,5**	**1**	**1,5**	**2**	**2,5**	**3**
M/*M*/3	0	0,334	0,500	0,600	0,666	0,716	0,750
Адаптивная СППР		0,469	0,752	0,877	0,932	0,961	0,974
M/*M*/1/100		0,503	0,996	1			

Проанализируем графики вероятности использования первого ЭО при различной нагрузке для рассматриваемых СППР, полученные в результате проведения имитационных экспериментов и представленные на рисунке 4.2.

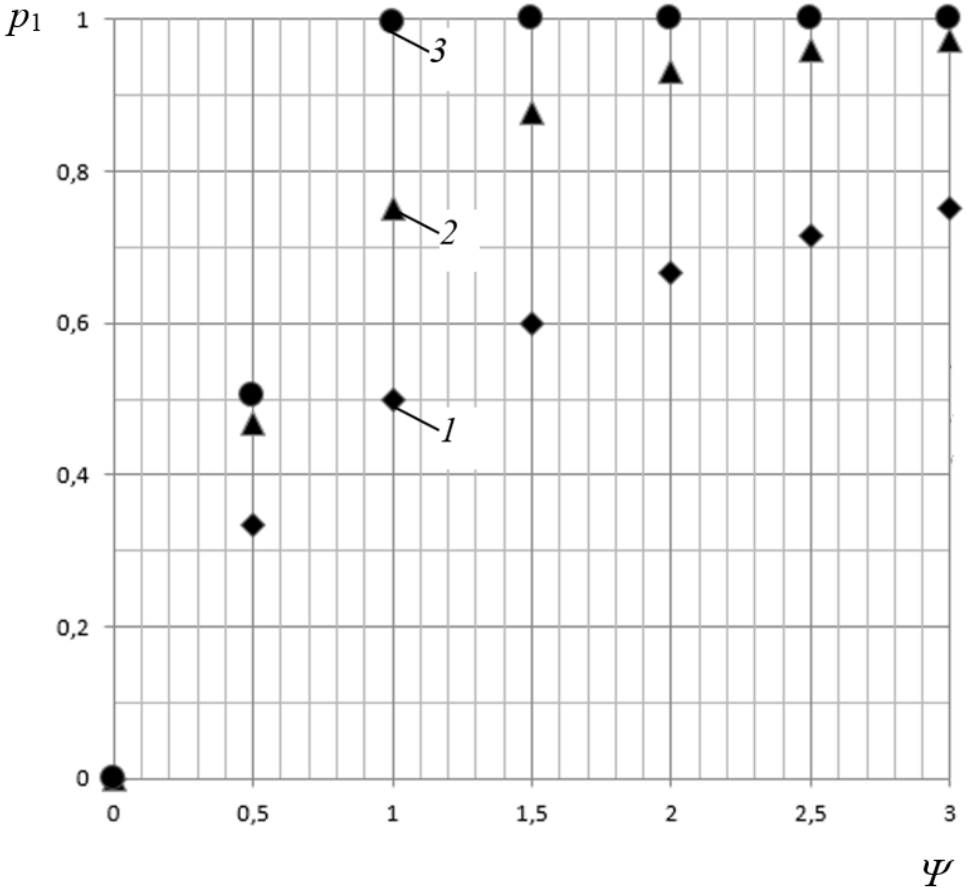

Рисунок 4.2 – Зависимости вероятности использования первого элемента обслуживания при различной нагрузке для систем поддержки принятия решений:
1 – *M*/*M*/3; *2* – адаптивная; *3* – *M*/*M*/1/100 [сделано автором]

Очевидно, адаптивная СППР (точки *2*) занимает промежуточное положение между системой без очередей *M*/*M*/3 (точки *1*) и системой с практически бесконечной очередью *M*/*M*/1/100 (точки *3*), подтверждая выводы теоретических расчетов. Система

функционирует в оптимальном режиме, не допуская ни чрезмерных простоев в работе (как в первом случае), ни работы в авральном режиме (как в случае системы *M*/*M*/1/100).

Рассмотрим коэффициенты использования ЭО в зависимости от нагрузки. Как и в случае предыдущего расчета, они находятся в параметре `UTIL` стандартного отчета – результата имитационного моделирования.

Полученные результаты расчета коэффициентов использования элементов обслуживания сведены в таблице 4.2, графические зависимости данных коэффициентов при различной нагрузке на СППР представлены на рисунке 4.3.

Таблица 4.2 – Коэффициенты использования элементов обслуживания [сделано автором]

Рассматриваемая система	**Нагрузка на систему, Ψ**						
	0	**0,5**	**1**	**1,5**	**2**	**2,5**	**3**
M/*M*/3	0	0,469	0,753	0,877	0,932	0,961	0,974
Адаптивная СППР		0,033	0,230	0,495	0,691	0,820	0,893
M/M/1/100		0	0,021	0,123	0,302	0,501	0,673

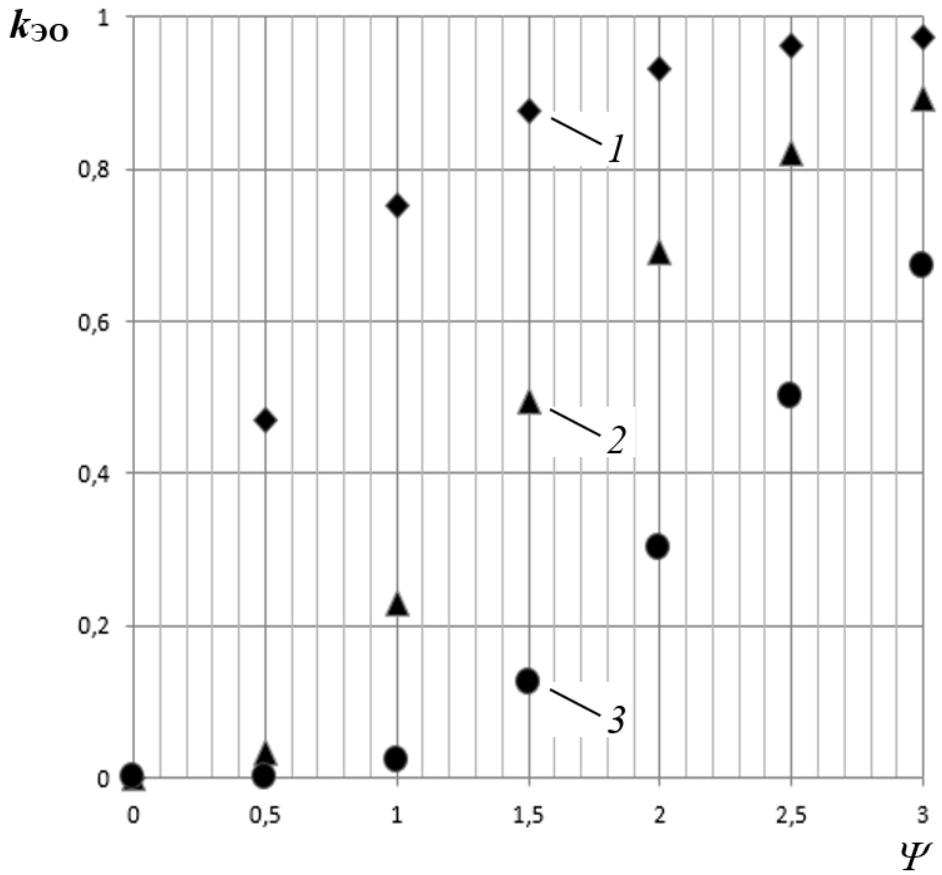

Рисунок 4.3 – Зависимости коэффициентов использования элементов обслуживания при различной нагрузке на систему поддержки принятия решений:
1 – *M*/*M*/3; *2* – адаптивная; *3* – *M*/*M*/1/100 [сделано автором]

Очевидно, что при значительном повышении нагрузки на адаптивную СППР последовательно, (в рамках лимита), включаются резервные ЭО. При исчерпании лимита резервных элементов коэффициенты использования всех ЭО стремятся к единице, т.е. система начинает работать на пределе своих возможностей. В таком случае все ЭО работают с максимальной отдачей, но необходимо знать процент отказов в обслуживании, что и предусматривает рассматриваемая система. Происходит это в те моменты, когда все ЭО и все места в очереди к ним оказываются занятыми. Результаты расчета вероятности отказа в обслуживании $P_{\text{отказ}}$ представлены в таблицу 4.3.

Таблица 4.3 – Вероятности отказа в обслуживании [сделано автором]

Рассматрива-емая система	**Нагрузка на систему, Ψ**										
	0	**0,5**	**1**	**1,5**	**2**	**2,5**	**3**	**3,5**	**4**	**4,5**	**5**
M/*M*/3	0	0,013	0,061	0,137	0,210	0,280	0,348	0,402	0,449	0,49	0,529
Адаптивная СППР		0		0,009	0,037	0,090	0,160	0,230	0,305	0,363	0,418
M/M/1/100		0					0,012	0,146	0,252	0,337	0,399

Зависимости вероятности отказа в обслуживании при различной нагрузке на СППР представлены на рисунке 4.4. Очевидно, что адаптивная система (точки *2*) обеспечивает меньшую вероятность отказа в обслуживании, чем система без очередей *M*/*M*/3 (точки *1*) но большую, чем система с практически бесконечной очередью *M*/*M*/1/100 (точки *3*).

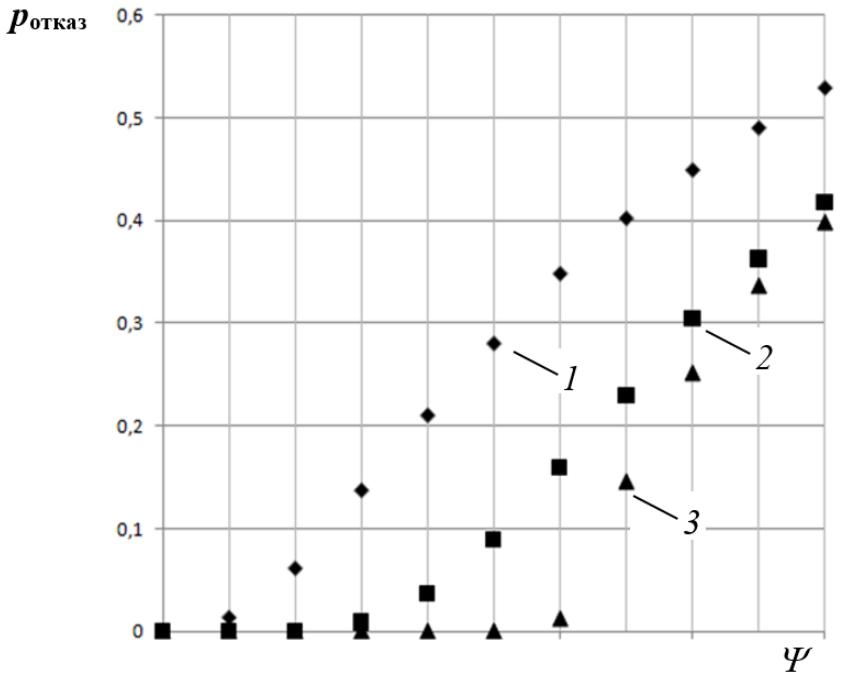

Рисунок 4.4 – Зависимости вероятности отказа в обслуживании при различной нагрузке на систему поддержки принятия решений: *1* – *M*/*M*/3; *2* – адаптивная; *3* – *M*/*M*/1/100 [сделано автором]

Подвергнув результаты испытаний несложной обработке (с использованием сохраняемых величин), получим вероятности работы нескольких ЭО одновременно (таблица 4.4). Графики вероятности использования нескольких ЭО одновременно приведены на рисунке 4.5. Очевидно, что максимумы вероятностей работы первого ЭО (точки *1*) и первых двух (точки *2*) элементов в целом соответствуют значениям нагрузки Ψ, т.е. в первых двух случаях при увеличении нагрузки наиболее вероятно пропорциональное увеличение числа ЭО. Ввиду имеющегося ограничения на общее число ЭО, вероятность одновременной работы всех (в данном случае трех) элементов плавно возрастает (точки *3*), и при отсутствии ограничения на величину Ψ становится равной единице, т.е. система использует все имеющиеся у нее ресурсы.

Таблица 4.4 – Вероятности работы нескольких элементов обслуживания одновременно [сделано автором]

Рассматрива-емая система	**Нагрузка на систему, Ψ**										
	0	**0,5**	**1**	**1,5**	**2**	**2,5**	**3**	**3,5**	**4**	**4,5**	**5**
M/*M*/3	0	0,454	0,567	0,389	0,201	0,086	0,034	0,015	0,006	0,003	0,001
Адаптивная СППР		0,015	0,170	0,381	0,450	0,393	0,285	0,205	0,131	0,091	0,062
M/*M*/1/100		0	0,004	0,053	0,195	0,395	0,585	0,709	0,814	0,871	0,911

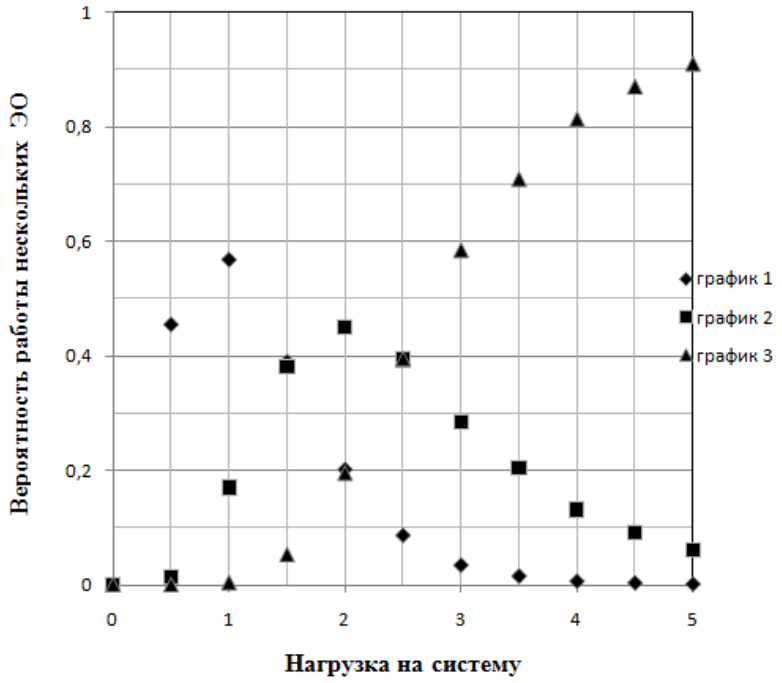

Рисунок 4.5 – Зависимости вероятности использования нескольких элементов обслуживания одновременно в системе поддержки принятия решений: *1* – *M*/*M*/3; *2* – адаптивная; *3* – *M*/*M*/1/100 [сделано автором]

Рассмотрим один из ключевых параметров любой системы обслуживания – среднее число запросов, находящихся в системе. Для сбора статистики данного параметра в GPSS используется общий накопитель. Финальное значение выводится в параметре `AVE.C`. Среднее число запросов в системе показано в таблице 4.5. Зависимости среднего числа запросов от нагрузки в СППР представлены на рисунке 4.6. Очевидно, что адаптивная система (точки *2*) занимает промежуточное положение между системой без очередей *M*/*M*/3 (точки *1*) и системой с практически бесконечной очередью *M*/*M*/1/100 (точки *3*).

Таблица 4.5 – Среднее число запросов в системе [сделано автором]

Рассматриваемая система	**Нагрузка на систему, Ψ**						
	0	**0,5**	**1**	**1,5**	**2**	**2,5**	**3**
M/*M*/3	0	0,494	0,934	1,303	1,579	1,797	1,963
Адаптивная СППР		0,781	1,902	3,112	4,250	5,287	6,125
M/*M*/1/100		1,010	4,580	–			

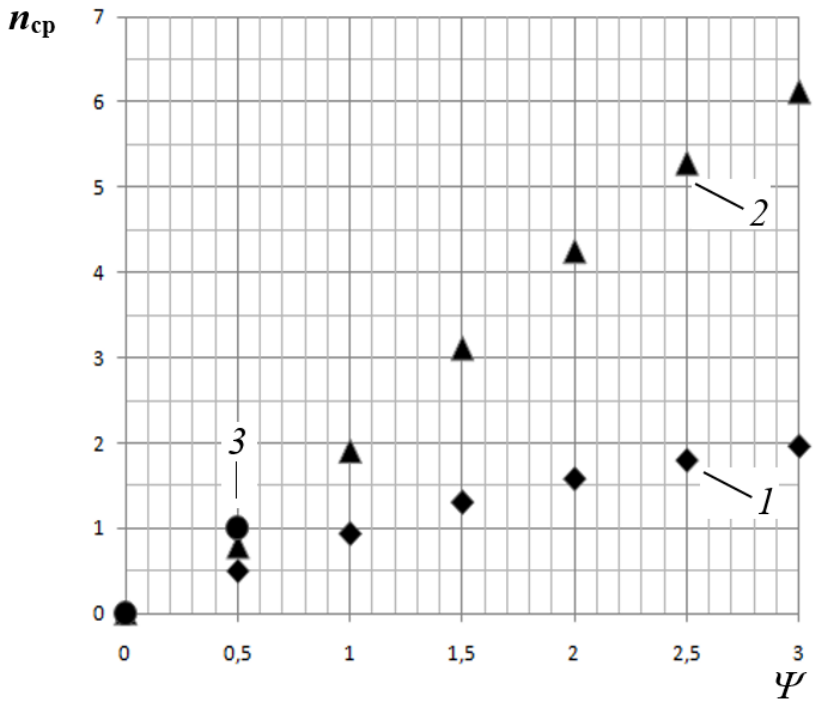

Рисунок 4.6 – Зависимости среднего числа запросов от нагрузки в системе поддержки принятия решений: *1* – *M*/*M*/3; *2* – адаптивная; *3* – *M*/*M*/1/100 [сделано автором]

Рассмотрим генерируемую имитационной моделью диаграмму, показывающую распределение запросов, находящихся в системе, по времени их обслуживания. На рисунке 4.7 показана диаграмма среднего времени нахождения запросов в системе, где представлены все поступившие в систему запросы (без учета потерянных запросов), распределенные по временным (шириной в пять минут) интервалам. По результатам

исследования разработанной модели, среднее время обслуживания составляет 23,249 единиц модельного времени, среднее квадратичное отклонение – 17,255 единиц. В то же время максимальное число запросов обслужено в интервале времени от 10 до 15 единиц, а более половины запросов обработано системой максимум за 20 единиц модельного времени.

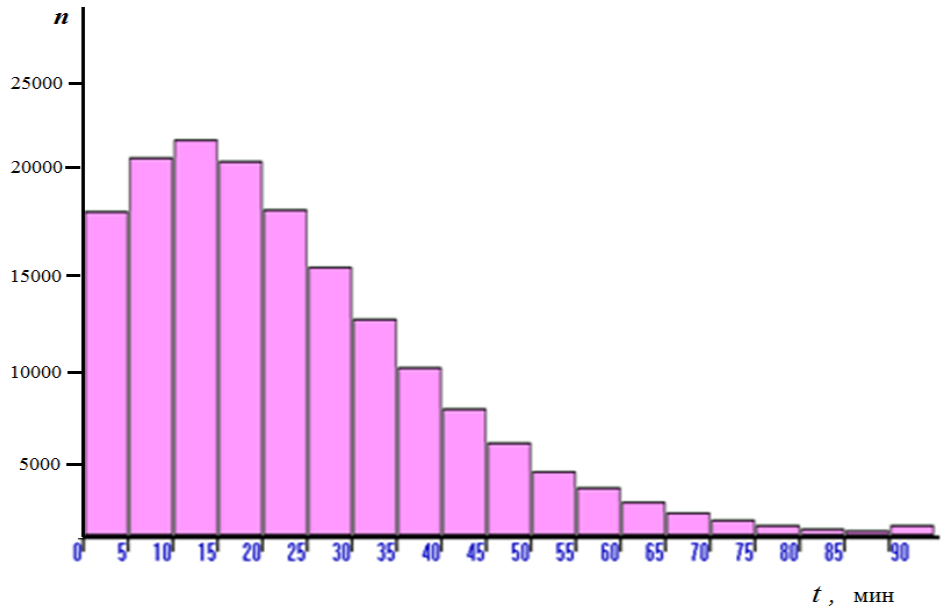

Рисунок 4.7 – Среднее время нахождения запросов в системе [сделано автором]

Полученные результаты выявляют основные преимущества адаптивных систем: большее количество обрабатываемых с помощью ЭО запросов в единицу времени; уменьшение времени простоя системы в целом и отдельных ЭО в частности; сокращение затрат на содержание ЭО, т.е. неиспользуемые в текущий момент ЭО отключаются, что приводит к существенной экономии (рабочего ресурса системы, временных и финансовых затрат, и т.д.); снижение вероятности отказа в обслуживании (система обслуживает бóльшее число поступающих запросов, что приводит к повышению качества их обслуживания); снижение вероятности попадания запросов в очереди; сокращение средней длины очередей.

Среднее число используемых программных ЭО в СППР предлагается снизить путем объединения баз данных знаний в единое информационное пространство, сокращения числа программ и применения вместо них интегрированных программ с расширенными функциями и настройками. Если СППР работает с многомерными хранилищами данных, то важно разработать в качестве инструментального средства для извлечения данных (источников данных может быть более сотни) такое программное

средство, которое позволяет открыть все имеющиеся источники данных. Важно при модернизации инструментальных программных средств переходить не к обновленной версии программы, а использовать модульное построение программы и заменять один из модулей, в зависимости от задачи модернизации [42…46].

5. Разработка технической документации на организацию информационного обеспечения системы управления большими данными

5.1 Состав информационного обеспечения

В таблице 5.1 приведен типичный усредненный состав комплекта технической документации на прикладной сервис (программное приложение) с указанием целевой аудитории каждого документа (ГОСТ 34.201-89). В данной работе составлено описание на информационное обеспечение системы управления большими данными и описание принципов организации работы с технологией блокчейн.

Таблица 5.1 – Состав типовой документации на технический проект

Документы	**Целевая аудитория**				
	Владелец сервиса	**Разработчик сервиса**	**Оператор сервиса**	**Отраслевой регулятор**	**Пользователь сервиса**
Системная документация					
Технический проект					
Описание комплекса технических средств	X	X	X	X	
Схема структурная комплекса технических средств	X	X	X	X	
Описание программного обеспечения	X	X	X	X	
Описание информационного обеспечения	X	X	X	X	
Описание организации информационной базы	X	X	X	X	
Описание организационной структуры	X	X	X	X	
Документация пользователя					
Онлайновая справочная система (веб-хелп)					X

Информационное обеспечение системы управления большими данными включает в себя внутримашинную и внемашинную информационную базу [47…53].

1) Внемашинная информационная база

Внемашинную информационную базу составляют регламенты и инструкции, предназначенные для организации работы обеспечивающих служб функциональных объектов системы.

2) Внутримашинная информационная база

В состав внутримашинной информационной базы системы входит единая БД.

Организация сбора и передачи информации

1) Источники и носители информации

Основными источниками информации для системы служат:

- данные, вводимые пользователями системы в интерактивном режиме;
- данные, автоматически регистрируемые компонентами системы;
- данные внешних и смежных систем, поступающие в систему через информационные веб-сервисы;
- логи поведения интернет-пользователей;
- данные систем поддержки принятия решений;
- данные иных сторонних источников.

Данные, вводимые пользователями системы в интерактивном режиме, включают:

- дополнительную информацию по обращениям и происшествиям, полученную пользователями системы в процессе передачи данных абоненту;
- консультационные данные, обеспечивающие информационную поддержку пользователей системы,
- данные консультационно-справочной поддержки абонентов и пользователей веб-сервисов;
- метаданные, описывающие структуру и информационное наполнение Интернетпортала и его форумов;
- нормативно-справочную информацию, ведущуюся непосредственно в системе;
- данные о пользователях системы и правах их доступа.

Данные, автоматически регистрируемые компонентами системы, включают:

- данные, полученные системой путем обработки предоставляемой веб-сервисами информации;
- данные, регистрируемые подсистемой информационной безопасности, о действиях пользователей системы;
- данные о результатах выполнения автоматических операций. фиксируемые компонентами системы,

Данные смежных и внешних систем, поступающие в систему через информационные веб-сервисы, включают:

- данные, поступающие из документов MS Office;
- данные, поступающие из унаследованных систем;
- данные, поступающие из транзакционных систем;
- данные, поступающие из учетных систем;
- данные файлов;
- данные архивов;
- данные системы управления БД.

Данные внешних источников, поступающие в систему, в электронном или в бумажном виде, включают:

- общероссийскую нормативно-справочную информацию;
- справочную информацию от операторов связи и прочих организаций.

Основные данные для аналитики, накапливаемые в предоставляемых сервисах:

– лог данные,

– файлы с записями или записи в БД с событиями, представленными в хронологическом порядке, обеспечивающим журналирование.

В сервисе присутствует 6 типов логирования:

1) Мгновенное логирование установок
2) Отложенное логирование всех данных
3) Отладочные логи
4) Архив логов
5) Агрегации
6) Логирование с помощью средств программы Kibana

1.2.1. Мгновенное логирование заявок

Данные о покупках и заказах отправляются в таблицу log на сервере 8003 и хранятся там в течении года.

Эти логи используются для проверки наличии недавних заказов в логике сервиса, а также для оперативного мониторинга.

1.2.2. Отложенное логирование всех данных

Сервера приложений сохраняют логи в формате json, в файлы вида /var/log/sender/music-service-20171027123000.json

Эпизодически по крону эти файлы сохраняются в таблицу log на сервере 2190.

На сервере 2190 данные хранятся в течении недели.

1.2.3. Отладочные логи

Полные логи включая отладку хранятся на серверах приложений в файле:

/var/log/sender/ music-service-partner.log

1.2.4. Архив логов

Периодически производится архивация логов. Логи старше недели с сервера 2190 переписываются на сервер 6103. Причем записываются в схему - соответствующую году, а таблицу - соответствующую месяцу.

1.2.5. Агрегации

Агрегации формируются с сервера 2190. Агрегации создаются, как часовые, так и дневные. Агрегации записываются на сервер 6103.

Агреации – это данные собранные и представленные в удобном виде для конечного потребителя, к примеру поискового индекса.

В данном случае, собирается конечная цифра посетителей музыкального сайта в заданный период времени. Существуют часовые и дневные. Все это служит для ускоренного процесса аналитики данных.

1.2.6. Kibana

Агрегации с сервера 6103 импортируются в Kibana. Агрегированные данные отображаются в виде графиков для удобного анализа и мониторинга.

Схема хранения и обработки логов сервиса представлена на рисунке 2. В данную схему входят:

- APP1, APP2 – сервера приложений
- Сервер 8003 – сервер БД хоста 8003
- Сервер 2190 – сервер БДхоста 2190
- Сервер 6103 – сервер БД хоста 6103 с агрегациями

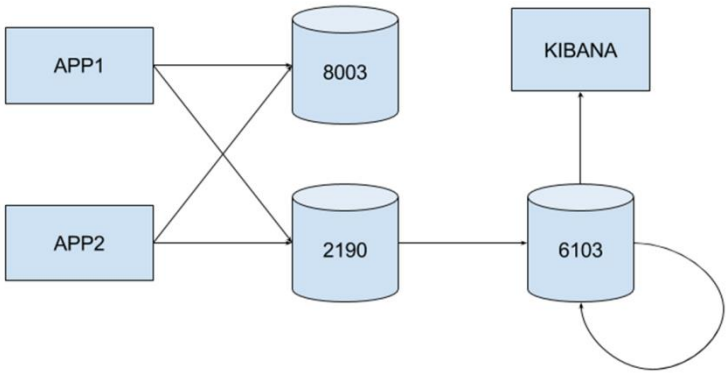

Рисунок 2 – Схема логирования прикладного сервиса

Основные данные необходимые для разработки концептуальной модели модуля ИС по распознаванию нелояльного пользователя сервиса:

1) данные о нелояльных пользователь от оператора;

2) данные о действиях пользователей сервиса из логов ИС, на основе которой функционирует сервис.

Идентификаторы получают от оператора и заносят их в таблицу stop_list для предотвращения повторного предоставления услуги.

Носители данных, размещаемые в специализированном центре обработки данных (ЦОД), позволяют разместить 18 ТБ оперативной и используемой в повседневной деятельности и 20 ТБ архивной информации.

В качестве типа физических носителей данных определены накопители на жестких магнитных дисках:

- для системных блоков – стандарта SATAIII 7200об/мин 32МБ типоразмера 3.5’ емкостью 320ГБ, объединенные в отказоустойчивый массив RAID5;
- для серверов резервирования ЦОД – стандарта SATAIII 7200об/мин 64МБ типоразмера 3.5’ емкостью 500ГБ, объединенные в отказоустойчивый высокоскоростной массив RAID10;
- для серверов и систем хранения данных ЦОД – определяются оператором ЦОД с учетом необходимости удовлетворения требований Технического задания на создание системы.

Общие требования к организации сбора, передачи, контроля и корректировки информации

Извлечение и сбор массивов информации происходит в процессе эксплуатации системы путѐм:

- автоматической регистрации информации компонентами системы;

- импорта структурированных данных формата XML (от англ. eXtensible Markup Language – расширяемый язык разметки), полученных от внешних систем;

- импорта структурированных данных формата XML или JSON (от англ. JavaScript Object Notation – текстовый формат обмена данными, основанный на JavaScript), полученных от смежных и внешних систем;

- формирования пользователями наборов информации в экранных формах и их последующего сохранения в БД.

Контроль целостности данных реализуется прикладным программным обеспечением системы и средствами, встроенными в используемую СУБД (ограничениями, индексами, внешними ключами).

Пополнение и актуализация БД производится в ходе нормального функционирования системы, в соответствии с заложенной в программные компоненты системы процедурной логикой.

Ввод и корректировка данных, ведение метаданных в хранилищах данных, оперативном складе хранения и зонах временного хранения должны осуществляться только через программные компоненты системы. Прямой доступ пользователей к БД не предполагается.

Обработка и анализ данных проводится с помощью операций выборки, реструктуризации и доставки.

Для предоставления доставленных данных используются: тематическая витрина данных, региональная витрина данных, витрина данных подразделения, прикладная витрина данных, функциональная витрина данных.

Сохранность данных системы обеспечивается примененной конфигурацией оборудования и регламентированной процедурой резервного копирования.

5.2 Организация работы с технологией блокчейн

При проведении транзакций на блокчейн-платформе каждый пользователь использует публичный адрес (необходимый другим сетевым объектам для отправки транзакции этому пользователю) и криптографически сопряженный «закрытый ключ». Закрытые ключи используются для цифровой подписи транзакций - формы проверки подлинности, чтобы убедиться, что пользователь на самом деле создал транзакцию.

Существует ряд предварительных условий, которые должны быть на месте, прежде чем реестр может быть интегрирован с технологией блокчейн [54,55]. К ним относятся:

- Решение для идентификации
- Оцифрованные записи: малейшее изменение цифрового файла приведет к совершенно другому хешу - «математическому алгоритму, который отображает данные произвольного размера в битовую строку фиксированного размера (хеш) и предназначен для односторонней функции, то есть функции, которую невозможно инвертировать», а бумажный файл нельзя хешировать, поэтому реестр должен быть полностью цифровым, прежде чем интегрировать блокчейн.
- Множественная подпись кошельков: эти кошельки требуют проверки минимальным количеством ключей, а не одним ключом, прежде чем транзакция будет завершена. Вместо простого нажатия продавцом кнопки «продать» конфигурация реестра может потребовать от продавца и банкира (или регистратора) подписать сделку.
- Частный или гибридный блокчейн

- Точные данные: в идеале, реестр должен быть очищен и обновлен до его установки на неизменяемую платформу.
- Связное и технически осведомленное население: в тех юрисдикциях, где возможности подключения ограничены или потребители не устраивают цифровые транзакции, реестр блокчейнов может быть неоптимальным.
- Обученное профессиональное сообщество: юристы, судьи, риелторы и пр. должны пройти подготовку по новой системе, с тем чтобы она функционировала надлежащим образом.

Заключение

1) Проведено описание комплекса технических средств системы управления обработкой больших объемов данных с помощью запросов и прикладных сервисов.

2) Разработаны алгоритмы архивирования, репликации и секционирования для повышения управляемости, производительности и доступности больших баз данных и действующих на их основе прикладных веб-сервисов.

3) Несмотря на принципиальное различие технологии блокчейн (децентрализованная архитектура для хранения копий информационных цепочек) и технологии больших данных (централизованная архитектура для интеграции информации из различных источников), и разных скоростей работы, потенциал анализа важной стратегической и финансовой информации из блокчейна при помощи больших данных значителен.

4) Разработан опытный образец программного приложения, реализующий принцип работы децентрализованной платформы на основе технологии блокчейн по продаже недвижимости. В качестве инструментальных средств программного приложения выбран объектно-ориентированный язык для разработки смарт-контрактов Solidity, среда разработки Remix IDE. Разработана схема взаимодействия с внешним реестром и схема использования приложения.

5) Рассчитаны параметры системы управления обработкой больших данных с помощью адаптивной к входящему потоку запросов системы поддержки принятия решений, в которой число элементов обслуживания увеличивается в зависимости от интенсивности данного потока; показано достижение экономического эффекта за счет снижения затрат на содержание данных элементов и уменьшения процента потерянных запросов из-за больших очередей.

Полученные результаты выявляют основные преимущества адаптивных систем: бóльшее количество обрабатываемых с помощью ЭО запросов в единицу времени; уменьшение времени простоя системы в целом и отдельных ЭО в частности; сокращение затрат на содержание ЭО, т.е. неиспользуемые в текущий момент ЭО отключаются, что приводит к существенной экономии (рабочего ресурса системы, временных и финансовых затрат, и т.д.); снижение вероятности отказа в обслуживании (система обслуживает бóльшее число поступающих запросов, что приводит к повышению качества их обслуживания); снижение вероятности попадания запросов в очереди; сокращение средней длины очередей.

6) Разработана техническая документация, описывающая организацию программного и информационного обеспечения для обработки больших объемов данных в соответствии с общими требованиями к организации сбора, передачи, контроля и корректировки информации, и организацию работы с технологией блокчейн.

Список использованных источников

1. Коновалов М.В. Big Data. Особенности и роль в современном бизнесе // Технические науки: проблемы и перспективы: материалы VI Междунар. науч. конф. (г. Санкт-Петербург, июль 2018 г.). — СПб.: Свое издательство, 2018. С. 8-10. URL https://moluch.ru/conf/tech/archive/288/14418/ (дата обращения: 08.12.2019).

2. Теория и практика Больших данных в отраслях // Статья. URL http://www.tadviser.ru/index.php (дата обращения: 06.12.2019).

3. Открыть цифровую экосистему // URL https://www.accenture.com/us-en/landing-programmable-network-platforms (дата обращения: 09.12.2019).

4. Орешков В.И. Бизнес-аналитика: от данных к знаниям. Учебное пособие. 2-е издание Издательство: Питер, 2013. 704 с.

5. Кузьмин А.Н. Трёхуровневая архитектура хранилища данных с интерфейсом запросов // Социально-экономические и технические системы: исследование, проектирование, оптимизация, 2006. № 2. С. 3.

6. Егоров С.Л. Разработка архитектуры информационного хранилища данных в процессе проектирования системы поддержки принятия решений // Известия высших учебных заведений. Уральский регион, 2010. № 4. С. 18-21.

4. Григорович А.Г. Архитектура хранилищ данных с ненормализованными отношениями (Data Warehouse Architecture with unnormalized relations) // Технические науки – от теории к практике. – 2013. – № 22. – С. 6-10.

7. Белошицкий Д.А. Интеграция данных в информационных системах // Молодежный научно-технический вестник. – 2013. – № 8. – С. 32.

8. Акимкина Э.Э. Инструментальный подход к организации сбора данных в хранилище систем поддержки принятия решений // Информационные технологии. 2017. №6. С. 424–430.

9. Илюшин Г.Я. Организация управляемого доступа пользователей к разнородным ведомственным информационным ресурсам // Информатика и ее применения, 2010. Т. 4. № 1. С. 24-40.

10. Соханевич С.В. Хранилище данных как базовый элемент построения инструментальных систем поддержки принятия управленческих решений // Известия ЮФУ. Технические науки, 2011. № 11 (124). С. 205-208.

11. Акимкина Э.Э. Развитие и адаптация имитационного и компьютерного моделирования в системах поддержки принятия решений // Современные информационные технологии / сборник трудов по материалам II-ой межвузовской научно-

технической конференции 14 сентября 2016 года, г.о. Королев, «МГОТУ» / Под общей науч. ред. док. техн. наук, проф. В.М. Артюшенко. – М.: Издательство «Научный консультант», 2016. С. 112 – 121 (182 с.).

12. Аббасов Э.М. Акимкина Э.Э. Достижение максимальной производительности при работе с крупными хранилищами данных // Информационные технологии. Радиоэлектроника. Телекоммуникации (ITRT-2016): сб. статей VI международной заочной научно-технической конференции. Ч.1 / Поволжский гос. ун-т сервиса. – Тольятти: Изд-во: ПВГУС, 24-25.03.2016. – С. 7 – 12 (345 с.)

13. Big Data Blockchain Projects You Should Know About [Электронный ресурс]. URL: https://www.smartdatacollective.com/6-big-data-blockchain-projects-you-should-know-about (дата обращения 17.12.2019)/

14. Люк Веллинг. Разработка веб-приложений с помощью PHP и MySQL / Люк Веллинг. М.: Вильямс, 2017. 768 с.

15. Масштабирование баз данных — партиционирование, репликация и шардинг [Электронный ресурс] – Режим доступа https://web-creator.ru/articles/partitioning_replication_sharding (Дата обращения 27.05.2018).

16. Репликация данных [Электронный ресурс] – Режим доступа https://ruhighload.com/%D0%A0%D0%B5%D0%BF%D0%BB%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F+%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85 (Дата обращения 27.05.2018)

17. Партицирование (partitioning) больших таблиц PostgreSQL [Электронный ресурс] – Режим доступа https://romantelychko.com/blog/520/ (Дата обращения 23.05.2018).

18. Дэвид Скляр. PHP. Рецепты программирования / Дэвид Скляр, М.: Питер, 2015. 784 с.

19. Adrianne Jeffries «Blockchain' is Meaningless» [Электронный ресурс], Verge, Mar 7, 2018. URL: https://www.theverge.com/2018/3/7/17091766/blockchain-bitcoin-ethereum-cryptocurrencymeaning (дата обращения 02.03.2019)

20. Düdder, B., Ross, O., 2017. Timber tracking: reducing complexity of due diligence by using blockchain technology. In: CEUR Workshop Proc., vol. 18

21. Артемьев К. Блокчейн: возникновение, особенности использования и регулирования [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/blokcheyn-vozniknovenie-osobennosti-ispolzovaniya-i-regulirovaniya (дата обращения 02.03.2019)

22. Pilkington M., 2016. Blockchain technology: principles and applications. Res. Handbook Digital Transformations, p. 225

23. Dannen C., 2017. Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners. Apress.

24. Buterin, V., 2015. On public and private blockchains, Ethereum Blog 7.

25. Eris Industries, 2016. Explainer — Smart Contracts [Электронный ресурс]. URL: https://docs.erisindustries.com/explainers/smart_contracts/ (дата обращения 02.03.2019)

26. Zheng, Z., Xie, S., Dai, H.-N., Wang, H., 2016. Blockchain challenges and opportunities: a survey. Work Pap.

27. Swanson, T., 2015. Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems [Электронный ресурс]. URL: https://allquantor.at/blockchainbib/pdf/swanson2015consensus.pdf (дата обращения 02.03.2019)

28. Hyperledger Project, 2015 [Электронный ресурс] // Официальный сайт. URL: https://www.hyperledger.org/ (дата обращения 02.03.2019)

29. Szabo, N. Smart Contracts: Building Blocks for Digital Markets, 1994.

30. Равал С. Децентрализованные приложения. Технология Blockchain в действии [Электронный ресурс]. URL: https://www.litres.ru/s-raval/decentralizovannye-prilozheniya-tehnologiya-blockchain-v-deystvii-23892845/ (дата обращения 05.03.2019).

31. Семенов А.Б., Артюшенко В.М., Аббасова Т.С. Пути наращивания эффективности инфокоммуникационных систем: монография / Под научной редакцией А.Б. Семенова. Москва, 2019. 126 с.

32. Ткачук Е. О. Требования к адаптивным системам поддержки принятия управленческих решений // Известия ТРТУ. 2002. № 2. С. 248 – 251.

33. Акимкина Э. Э. Развитие и адаптация имитационного и компьютерного моделирования в системах поддержки принятия решений // Современные информационные технологии / сборник трудов по материалам II-ой межвузовской научно-технической конференции 14 сентября 2016 года, г. Королев, «МГОТУ» / Под общей науч. ред. док. техн. наук, проф. В.М. Артюшенко. – М.: Издательство «Научный консультант», 2016. С. 112 – 121 (182 с.).

34. Клейнрок Л. Теория массового обслуживания. М.: Книга по требованию. 2013. 429 с.

35. Советов В. М., Артюшенко В. М. Основы функционирования систем сервиса: учебное пособие. М.: Издательский Дом «Альфа-М». 2010. 624 с.

36. Зиновьев В. В., Кочетков В. Н. Опыт имитационного моделирования сложных производственных систем // Вычислительные технологии. 2008. № S5.Т. 13. С. 51 – 55.

37. Калиниченко С. В., Хомоненко А. Д. Модель оценки оперативности функционирования распределённых автоматизированных систем при интеграции данных // Бюллетень результатов научных исследований. 2012. № 4 (5). С. 47 – 57.

38. Хлудов О. Н. Моделирование системы сервиса с изменяющимся числом элементов обслуживания // Электротехнические и информационные комплексы и системы. 2011. №3. Т. 7. С. 21 – 24.

39. Советов В. М. Система сервиса с изменяющимся числом элементов обслуживания // Электротехнические и информационные комплексы и системы. 2010. №1. Т. 6. С. 10 – 14.

40. GPSS World Student Version обзор от Joy Download [Электронный ресурс]. URL: http://gpss-world-student-version.joydownload.ru/ (дата обращения: 17.05.2017).

41. Акимина Э. Э. Оптимизация обработки данных в системах поддержки принятия решений с элементами обслуживания // Вестник ВГУ. Серия: Системный анализ и информационные технологии. 2017. №2. С. 90 – 97.

42. Артюшенко В.М., Акимкина Э.Э. Имитационная модель адаптивной системы поддержки принятия решений // Вестник компьютерных и информационных технологий. 2018. № 2. С. 46–56.

43. Акимкина Э.Э., Аббасов А.Э. Анализ инструментальных средств информационных систем для обработки многомерных данных // Информационно-технологический Вестник. 2016. №2(08). С. 61–74.

44. Акимкина Э.Э. Рекомендации по развертыванию многомерных систем аналитической обработки данных // Информационно-технологический вестник. 2017. № 5(11). С. 68–80.

45. Аббасова Т.С., Акимкина Э.Э., Аббасов Э.М. Извлечение данных мониторинга для оперативного анализа / Нано- и биомедицинские технологии. Управление качеством. Проблемы и перспективы. Сборник научных статей. под ред. д.ф.-м.н. проф. С. Б. Венига. – Саратов, СГУ 2019. Вып. 3. С. 9-15 (256 с.: ил.)

46. Акимкина Э.Э. Структуризация и визуализация показателей в многомерных кубах данных // Информационно-технологический Вестник. 2018. № 4 (18). С. 79 –87. ISSN 2409-1650. (ВАК).

47. Польшин С.Н., Косарев Д.И., Хижук А.В. Перспективы развития машинного обучения // Современные информационные технологии: сборник трудов по материалам 3-й межвузовской научно-технической конференции с международным участием 29 сентября 2017 г. / колл. авторов; под общ. науч. ред. док. техн. наук, проф. В.М. Артюшенко. – М.: Издательство «Научный консультант», 2017. С.99-107.

48. Аббасова Т.С., Польшин С.Н. 2017. Чат-боты и нейронные сети / Сборник трудов по материалам II-й межвузовской научно-технической конференции с международным участием «Эволюционные процессы информационных технологий».

49. Исаева Г.Н., Теодорович Н.Н., Харламова Е.С., Польшин С.Н. Использование современных средств программирования в науке о данных // Информационные технологии. Эволюционные процессы / Сборник научных статей под ред. д.т.н., проф. В.М. Артюшенко, колл. авторов. – М.: Издательство «Научный консультант», 2018. С. 17–22.

50. Исаева Г.Н., Теодорович Н.Н. Проблемы передачи и безопасности данных в интернет вещей // Современные информационные технологии/ сборник трудов по материалам 4-й межвузовской научно-технической конференции с международным участием 28 сентября 2018 г под общей науч. ред. док. техн. наук проф В.М. Артюшенко. - М: Издательство «Научный консультант», 2018. - С.89-96.

51. Польшин С.Н., Логачева Н.В., Сидорова Н.П. Возможности и перспективы чат-ботов // Информационные технологии. Эволюционные процессы / Сборник научных статей под ред. д.т.н., проф. В.М. Артюшенко, колл. авторов. – М.: Издательство «Научный консультант», 2018. С. 108–115 (130 с.)

52. Аббасова Т.С. Задачи оптимизации инфокоммуникационных систем // Информационно-технологический Вестник, 2018. № 3 (17). С. 55-65.

53. Стреналюк Ю.В. Концепция и основные требования к ядру территориальной информационной телекоммуникационной сети // Информационно-технологический Вестник, 2018. № 4 (18). С. 109-116.

54. Michael Graglia, Christopher Mellon, and Evan Akin, «Prerequisites for Incorporating Blockchain into a Registry» [Электронный ресурс] // FPR Blog (blog), New America, July 31, 2017. URL: www.newamerica.org/international-security/future-property-rights/blog/prerequisites-incorporating-blockchain-registry/ (дата обращения 05.03.2019).

55. Шукаева А.В. Перспективы внедрения технологий блокчейн в финансовую систему государства// Центральный научный вестник, 2018, №15-16, Т. III [Электронный ресурс] http://cscb.su/n/031501.html (дата обращения 18.08.2018).