Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Хакасский технический институт – филиал ФГАОУ ВО
«Сибирский федеральный университет»
институт

Прикладная информатика, математика и естественнонаучные дисциплины
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____Е.Н. Скуратенко
подпись инициалы, фамилия
« _____» _____ 2019г.

БАКАЛАВРСКАЯ РАБОТА

09.03.03 – Прикладная информатика
код – наименование направления

Автоматизация передачи сведений о прикреплении населения к медицинским организациям из РИАМС «ПроМед» в ИС ТФОРМС РХ
тема

Руководитель _____ зав. кафедрой, доцент, к.т.н. Е.Н.Скуратенко
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ И.Н. Барских
подпись, дата инициалы, фамилия

Консультанты по разделам:

Экономический _____ Е.Н. Скуратенко
наименование раздела подпись, дата инициалы, фамилия

Нормоконтролер _____ В.И. Кокова
подпись, дата инициалы, фамилия

Абакан 2019

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Хакасский технический институт – филиал ФГАОУ ВО
«Сибирский федеральный университет»
институт
Прикладная информатика, математика и естественнонаучные дисциплины
кафедра

УТВЕРЖДАЮ

Заведующий кафедрой
_____ Е.Н. Скуратенко
подпись инициалы, фамилия
« ___ » _____ 2019 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту _____Барских Илье Николаевичу_____
фамилия, имя, отчество

Группа 55-1 (ХБ 15-04) Направление (специальность) 09.03.03
номер код

Прикладная информатика_____
наименование

Тема выпускной квалификационной работы Автоматизация передачи сведений о прикреплении населения к медицинским организациям из РИАМС «ПроМед» в ИС ТФОРМС РХ

Утверждена приказом по институту № 279 от 18.04.2019 г._____

Руководитель ВКР Е.Н. Скуратенко, зав. кафедрой, доцент, к.т.н.,___
ХТИ – филиал СФУ
инициалы, фамилия, должность, ученое звание и место работы

Исходные данные для ВКР информация о работе информационных систем РИАМС «ПроМед» и ТФОМС РХ, в том числе технические особенности их работы, технические данные разрабатываемого в ВКР программного продукта.

Перечень разделов ВКР:

1.Анализ предметной области. Выбор средств проектных решений.

2. Описание разработки возможности передачи данных о прикреплении населения к медицинским организациям из РИАМС «ПроМед» в ИС ТФОМС.

3. Оценка экономической эффективности автоматизированной информационной системы для передачи данных о прикреплении населения к медицинским организациям из РИАМС «ПроМед» в ИС ТФОМС РХ.

Перечень графического материала нет

Руководитель ВКР _____ Е.Н. Сукратенко__
подпись инициалы и фамилия

Задание принял к исполнению _____ И.Н. Барских_____
подпись, инициалы и фамилия студента

« 19 » апреля 2019 г.

РЕФЕРАТ

Выпускная квалификационная работа по теме «Автоматизация передачи сведений о прикреплении населения к медицинским организациям из РИАМС «ПроМед» в ИС ТФОМС РХ»содержит 64 страниц текстового документа, 14 формул, 8 таблиц, 9 рисунков, 30 использованных источников.

ГКУЗ РХ «РМИАЦ», ТФОМС РХ, АИС, МИКРОСЕРВИСНАЯ АРХИТЕКТУРА, ПЕРЕДАЧА ДАННЫХ, ПРЯМЫЕЗАТРАТЫ, ЭКСПЛУАТАЦИОННЫЕ ЗАТРАТЫ, КАПИТАЛЬНЫЕ ЗАТРАТЫ.

Целью выпускной квалификационной работы является сокращение временных затрат на обмен информацией путем создания дополнительного функционала для шины «Передача данных о прикреплении (единый пакет с участками)».Для достижения поставленной цели необходимо решить следующие задачи:

1) осуществить поиск информации необходимой для достижения поставленной цели;

2) построить модели процессов предметной области;

3) обосновать выбор средств проектных решений;

4) создать систему позволяющую совершать автоматическую передачу данных в соответствии с поставленной целью ВКР;

5) провести тестирование и подготовить ИС к внедрению системы;

6) оценить планируемую совокупную стоимость разработки проекта;

7) провести оценку экономической эффективности проекта и анализ рисков.

SUMMARY

The theme of the Bachelor's thesis is «Automation of Data Transfer on the Population Registration in Healthcare Organizations from RIAMS «ProMed» to IS of MHIF RKH». It contains 64 pages, 14 formulae, 8 charts, 9 figures, 30 reference items.

SHCPI RKH «RMCIA» (State Healthcare Public Institution of Republic of Khakassia «Republican Medical Center for Information and Analysis»), MHIF RKH (Mandatory Health Insurance Fund of Republic of Khakassia), AIS, MICROSERVICE ARCHITECTURE, DATA TRANSFER, DIRECT COSTS, OPERATIONAL COSTS, CAPITAL COSTS

The purpose of the graduation thesis is to reduce the information exchange time via creating additional bus feature set of «Data transfer on registration». To achieve the stated purpose, it is necessary to consider the following objectives:

1) to search for the necessary information to achieve the purpose;

2) to model the subject area processes;

3) to substantiate the choice of project design tools;

4) to create a system that automates data transfer in accordance with the intended purpose of the Bachelor's thesis;

5) to conduct testing and to prepare the IS for implementation.

6) to assess the planned total cost of project development;

7) to assess the project cost-effectiveness and risk analysis as well.

English language supervisor _____ _____
(signature, date) (full name)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

На сегодняшний день в системе здравоохранения Республики Хакасия создаются информационные системы (далее ИС), которые позволяют систематизировать и автоматизировать процессы сбора, обработки и хранения медицинской информации. Две такие системы представлены ниже: первая поддерживается Республиканским медицинским информационно–аналитическим центром (далее РМИАЦ), а вторая Территориальным фондом обязательного медицинского страхования (далее ТФОМС). Для наилучшего функционирования этих ИС, они должны обмениваться хранимой информацией друг с другом. Но РМИАЦ и ТФОМС разные организации, их информационные системы находятся в разных закрытых сетях. На данный момент изъятие, отправка и загрузка информации из одной ИС в другую делается вручную человеком, что создает большие затраты по времени. Поэтому было принято решение создать программное обеспечение, которое автоматизирует процесс обмена информацией (далее Шина).Одной из возможностей шины является автоматизированная«Передача данных о прикреплении (единый пакет с участками)».

Таким образом, целью данной выпускной квалификационной работы является сокращение временных затрат на обмен информацией путем создания дополнительного функционала для шины «Передача данных о прикреплении (единый пакет с участками)».

Для достижения поставленной цели нужно выполнить следующие задачи:

1) осуществить поиск информации необходимой для достижения поставленной цели;

2) построить модели процессов предметной области;

3) обосновать выбор средств проектных решений;

4) создать систему позволяющую совершать автоматическую передачу данных в соответствии с поставленной целью ВКР;

5) провести тестирование и подготовить ИС к внедрению системы;

6) оценить планируемую совокупную стоимость разработки проекта;

7) провести оценку экономической эффективности проекта и анализ рисков.

1 Анализ предметной области. Выбор средств проектных решений

1.1 Анализ деятельности предприятия Государственное казённое учреждение здравоохранения Республики Хакасия «РМИАЦ».

ГКУЗ РХ «РМИАЦ» осуществляет свою деятельность с целью формирования единой информационной системы здравоохранения Российской Федерации путем организации на базе современных компьютерных технологий межотраслевой системы сбора, обработки, хранения и представлений информации, обеспечивающей динамическую оценку здоровья и информационную поддержку принятия решений, направленных на его улучшение.

Ниже раскрыты ключевые положения учетной политики, определяющей ведение ГКУЗ РХ «Республиканский медицинский информационно-аналитический центр»:

– Разработка концепций и программ информатизации здравоохранения Республики Хакасии.

– Координация работ по созданию единой информационной системы здравоохранения Республики Хакасии.

– Формирование единой системы учета и отчетности медико-статистической информации с применением новых технологий её обработки.

– Прием и обработка статистических отчетов от учреждений здравоохранения.

– Разработка показателей, характеризующих деятельность учреждений здравоохранения Республики Хакасии, состояние здравоохранения в рамках утвержденной статистической отчетности.

– Разработка, внедрение и сопровождение автоматизированных систем сбора, обработки, хранения и передачи информации.

– Анализ полученной информации с привлечением главных штатных и внештатных специалистов Министерства здравоохранения Республики, организационно-методических отделов краевых больниц и диспансеров.

– Осуществление взаимодействия с территориальным фондом обязательного медицинского страхования, территориальными органами государственной статистики, образовательными и научными учреждениями и другими сторонними организациями.

– Осуществление контроля за использованием в работе учреждений здравоохранения международных классификаций при ведении медицинской документации.

– Осуществление контроля за ведением в учреждениях здравоохранения медицинской документации, удостоверяющей случаи смерти.

– Обеспечение учреждений здравоохранения формами медицинской учетной и отчетной документации, утвержденными приказами Министерства здравоохранения Российской Федерации и постановлениями Госкомстата России.

– Изучение и прогнозирование процессов и явлений, связанных со здоровьем человека.

– Выработка управленческих решений по повышению эффективности деятельности учреждений здравоохранения.

– Организация и контроль за состоянием статистического учета и отчетности учреждений здравоохранения.

– Координация деятельности службы медицинской статистики Республики Хакасии.

– Представление в Министерство здравоохранения сводных государственных и отраслевых медицинских статистических отчетов в соответствии с установленным порядком.

– Формирование и сопровождение государственной статистической отчетности Российской Федерации.

– Анализ медико-статистической информации о состоянии здоровья населения и состоянии здравоохранения Республики Хакасии.

– Повышение эффективности использования информационной инфраструктуры здравоохранения Российской Федерации.

– Обеспечение достоверности сведений в учетной и отчетной медицинской документации.

– Внедрение новых технологий сбора и обработки медико-статистической информации.

– Создание и поддержание единого мониторинга здоровья населения.

– Подготовка и публикация аналитических обзоров о состоянии

здоровья и здравоохранения, сборников основных показателей по разделам здравоохранения.

– Проведение мероприятий по повышению деловой и профессиональной квалификации работников медицинской статистики и технического обеспечения при организации семинаров, распространение передового опыта и оказание организационно-медицинской помощи [12].

Министерство здравоохранения контролирует работу ГКУЗ РХ “РМИАЦ”, которое состоит из таких отделов как: отдел медицинской статистики и мониторинга, отдел автоматизированных систем управления, технический отдел, отдел адресно-справочной работы. Структурная схема представлена ниже на рисунке 1.1.

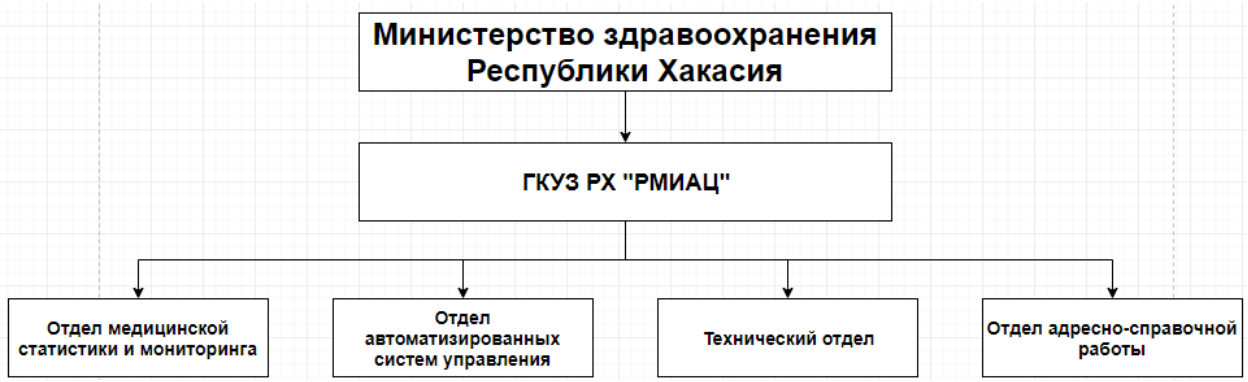

Рисунок 1.1 – Структурная схема РМИАЦ

На данный момент в ГКУЗ РХ «РМИАЦ» функционирует автоматизированный цифровой обмен с медицинскими организациями через e-mail и облачное хранилище.

Запись к специалистам организована через ведение электронного расписания и горячую линию.

Сопровождением программного обеспечения занимается технический

отдел

На данный момент установлено 24 компьютера. Из них:

– 9 машин с процессором Core 2 Duo;

– 15 машин с процессором Core i3.

Приложения, которые используются на предприятии:

– Медстат.

– Веб-приложение ПроМед.

– Касперский endpointsecurity (управляется централизованно с сервера).

– DallasLock — система защиты информации от несанкционированного доступа в процессе её хранения и обработки. Представляет собой программный комплекс средств защиты информации в автоматизированных системах.

Основные информационные системы, используемые для решения задач в сфере здравоохранения Республики Хакасия:

– единый медицинский портал Республики Хакасия;

– телемедицинская система Республики Хакасия;

– глобальная навигационная спутниковая система (ГЛОНАСС) «АвтоГРАФ»;

– региональная информационно-аналитическая медицинская система (РИАМС) «ПроМед».

Региональная информационно-аналитическая медицинская система (РИАМС) «ПроМед»– это специализированный программный комплекс, позволяющий автоматизировать процессы сбора, обработки и хранения медицинской, экономической и статистической информации в системе здравоохранения региона.

РИАМС «ПроМед» функционирует по модели(Softwareas a Services – программное обеспечение как услуга) на едином центре обработки данных для неограниченного числа пользователей. В ЦОД консолидируется вся информация, связанная с персонифицированным учетом оказанной

медицинской помощи и управлением ресурсами здравоохранения региона. «ПроМед» обеспечивает информационный обмен между медицинскими учреждениями, органами управления здравоохранением, ТФОМС, страховыми медицинскими организациями и аптечными учреждениями, участвующими в реализации программы дополнительного лекарственного обеспечения (ДЛО).

Централизованная архитектура РИАМС позволяет масштабировать ее в пределах региона в рекордно короткие сроки и без существенных материальных затрат. Для подключения медицинских учреждений к «ПроМед» необходимо только наличие каналов связи с пропускной способностью не менее 2 Мбит/с и оборудованных вычислительной техникой рабочих мест врачей. Доступ к медицинской информационной системе возможен, как со стационарных, так и с мобильных устройств [21].

На текущий момент РИАМС «ПроМед» принадлежит и обслуживается ГКУЗ РХ «РМИАЦ».

1.2 Анализ деятельности предприятия «Территориальный Фонд обязательного медицинского страхования Республики Хакасия»

Территориальный фонд обязательного медицинского страхования Республики Хакасия (далее – Территориальный фонд) является некоммерческой организацией, созданной Республикой Хакасия для реализации государственной политики в сфере обязательного медицинского страхования на территории Республики Хакасия.

Территориальный фонд является юридическим лицом, созданным в соответствии с законодательством Российской Федерации, в своей деятельности подотчетен Правительству Республики Хакасия и Федеральному фонду обязательного медицинского страхования (далее - Федеральный фонд). Для реализации своих полномочий Территориальный

фонд открывает счета, может создавать филиалы и представительства, имеет бланк и печать со своим полным наименованием, иные печати, штампы и бланки, геральдический знак-эмблему.

Территориальный фонд осуществляет свою деятельность в соответствии с Конституцией Российской Федерации, федеральными законами, указами и распоряжениями Президента Российской Федерации, постановлениями и распоряжениями Правительства Российской Федерации, нормативными правовыми актами федерального органа исполнительной власти, осуществляющего функции по выработке государственной политики и нормативному правовому регулированию в сфере здравоохранения, настоящим Положением и нормативными правовыми актами Республики Хакасия.

Задачами Территориального фонда являются:

- обеспечение предусмотренных законодательством Российской Федерации прав граждан в системе обязательного медицинского страхования;
- обеспечение гарантий бесплатного оказания застрахованным лицам медицинской помощи при наступлении страхового случая в рамках территориальной программы обязательного медицинского страхования и базовой программы обязательного медицинского страхования;
- создание условий для обеспечения доступности и качества медицинской помощи, оказываемой в рамках программ обязательного медицинского страхования;
- обеспечение государственных гарантий соблюдения прав застрахованных лиц на исполнение обязательств по обязательному медицинскому страхованию в рамках базовой программы обязательного медицинского страхования независимо от финансового положения страховщика [23].

У Территориального фонда имеется своя ИС наподобие РИАМС «ПроМед» для сбора, обработки и хранения медицинской информации.

1.3 Сравнение информационных систем ГКУЗ РХ «РМИАЦ» и ТФОМС РХ

Для того, чтобы разработать дополнительный функционал передачи данных о прикреплении населения к медицинским организациям нужно сравнить рабочие области сетей и форматы данных, с которыми работают две существующие информационные системы.

1.3.1 Защищенная сеть передачи данных

VPN (англ. VirtualPrivateNetwork – виртуальная частная сеть) – это безопасное, зашифрованное подключение между двумя сетями или между отдельным пользователем и сетью. Сети VPN позволяют пользоваться Интернетом, сохраняя конфиденциальность.

Технология VPN шифрует все ваши действия в интернете. Все, что вы отправляете и получаете. Если вы входите только через VPN, виден не ваш подлинный источник подключения, а один из многочисленных VPN-маршрутизаторов. Существует три основных вида шифрования: хеширование, симметричное и асимметричное шифрование. У каждого вида свои преимущества и недостатки, но все они шифруют ваши данные так, что в чужих руках они будут бесполезными.

Дополнительный уровень защиты, который есть у большинства служб VPN — их собственная система DNS. DNS — система доменных имен — это "телефонная книга интернета", в которой текстовые URL-адреса отождествлены с соответствующими IP-адресами. Киберпреступники могут

наблюдать за запросами DNS, чтобы отслеживать действия в интернете, но система DNS в службах VPN разработана так, чтобы с помощью дополнительного шифрования помешать им [29]. Чтобы придерживаться Федерального Закона «О персональных данных» – обе ИС работают в разных защищенных сетях.

1.3.2 Форматы данных JSONи XML

XML-файлы представляют собой файлы, используемые в различных приложениях, поскольку они относятся к файлам, которые написаны в ExtensibleMarkupLanguage. Это формат, который содержит теги и использует его для определения объекта, а также атрибуты. Эти файлы немного похож на форматирование по сравнению с HTML документе, но для того, чтобы определить данные, которые используются пользовательские теги. Эти XML-файлы распространенных способов, используемых для хранения и передачи данных в Интернете, которые приходят из разных программ и могут быть отредактированы в любом текстовом редакторе, потому что они приходят в текстовых документах [11].

JSON (JavaScriptObjectNotation) – простой формат обмена данными, удобный для чтения и написания как человеком, так и компьютером. Он основан на подмножестве языка программирования JavaScript, определенного в стандарте ECMA-262 3rd Edition - December 1999. JSON - текстовый формат, полностью независимый от языка реализации, но он использует соглашения, знакомые программистам C-подобных языков, таких как C, C++, C#, Java, JavaScript, Perl, Python и многих других. Эти свойства делают JSON идеальным языком обмена данными [5].

Информационная система Территориального фонда работает с данными формата json в то время, как РИАМС «ПроМед» работает с данными формата XML.

Поэтому при создании дополнительного функционала «Передача данных о прикреплении (единый пакет с участками)» для принятия и отправки данных нужно использовать защищенную сеть. Также нужно преобразовывать данные из одного формата в другой (из XMLв JSON).

1.4 Постановка цели и задач проектирования

На данный момент в «Шине» не существует возможности, которая позволила бы автоматизировать процесс передачи данных о прикреплении из «ПроМед» в ИС Территориального фонда. Данный функционал нужен для того, чтобы связать потоки данных в системах, которые используют разные форматы данных. Таким образом, актуальность выбранной темы выпускной квалификационной работы связана с необходимостью разработки дополнительного функционала передачи данных о прикреплении населения к медицинским организациям.

Целью данной выпускной квалификационной работы является сокращение временных затрат на обмен информацией путем создания дополнительного функционала для шины «Передача данных о прикреплении (единый пакет с участками)».

Для достижения поставленной цели необходимо решить следующие задачи:

1) осуществить поиск информации необходимой для достижения поставленной цели;

2) построить модели процессов предметной области;

3) обосновать выбор средств проектных решений;

4) создать систему позволяющую совершать автоматическую передачу данных в соответствии с поставленной целью ВКР;

5) провести тестирование и подготовить ИС к внедрению системы;

6) оценить планируемую совокупную стоимость разработки проекта;

7) провести оценку экономической эффективности проекта и анализ рисков.

1.5 Обоснование выбора средств для разработки автоматизированной информационной системы для передачи информацией из БД РИАМС «ПроМед» в ТФОМС РХ

Автоматизированная информационная система (АИС) — совокупность программно-аппаратных средств, предназначенных для автоматизации деятельности, связанной с хранением, передачей и обработкой информации [10]. Данная АИС будет разрабатываться с помощью микросервисной архитектуры.

Микросервисная архитектура— принципиальная организация распределенной системы на основе микросервисов и их взаимодействия друг с другом и со средой по сети, а также принципов, направляющих проектирование архитектуры, её создание и эволюцию [20] Просто о микросервисах. Суть данной архитектуры в том, что каждый созданный сервис должен решать одну задачу.

Микросервисная архитектура содержит следующие принципы:

— Сервисы должны быть сфокусированы. Таким образом, из данного принципа следует, что каждый сервис выполняет свою единственную задачу.

— Сервис должен быть небольшим. Из этого принципа следует, что разработка сервиса не требует множества людей.

— Сервисы должны быть малосвязанными. В этом принципе описано то, что изменения в одном сервисе не должно влиять на другой.

Разрабатываемая АИС будет представлять из себя веб-приложение. Веб-приложение – это клиент-серверное приложение, в котором клиентом выступает браузер, а сервером – веб-сервер. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется,

преимущественно, на сервере, обмен информацией происходитпо сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения являются кроссплатформенными сервисами [20].

1.5.1Выбор языка программирования для разработки

Для разработки веб-приложений используются языки программирования, которые в основном предназначены для работы с веб-технологиями. Языки программирования для веб-приложений делятся на клиентские и серверные. В данной разработке будет применяться серверный язык, т.к. работа таких программ полностью зависима от сервера, на котором расположена. Важная сторона работы серверных языков – возможность организации непосредственного взаимодействия с системой управления базами данных.

Python – интерпретируемый язык программирования. С одной стороны, это позволяет значительно упростить отладку программ, с другой – обуславливает сравнительно низкую скорость выполнения. Python поддерживает несколько парадигм программирования, в том числе структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное. Основные архитектурные черты – динамическая типизация, автоматическое управление памятью, полная интроспекция, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных [22].

Преимущества языка программирования Python:

— кроссплатформенность и бесплатность;

— простой синтаксис и богатые возможности позволяют записывать программы очень кратко, но в то же время понятно;

— по простоте освоения язык сравним с бейсиком, но куда более богат возможностями и значительно более современен;

— богатая стандартная библиотека, возможность разработки промышленных приложений (для работы с сетью, GUI, базами данных и т.д.).

Недостатки языка программирования Python:

— ограниченность средств разработки с базами данных;

— низкое быстродействие;

— ограниченность средств для работы с базами данных.

PHP (рекурсивный акроним словосочетания PHP: HypertextPreprocessor) – это распространенный язык программирования общего назначения с открытым исходным кодом. PHP сконструирован специально для ведения Web-разработок и его код может внедряться непосредственно в HTML.

Скрипты, написанные на языке PHP, обычно хранятся в файлах срасширениемphp, которые содержат в себе смесь обычных HTML-тэгов со специальной разметкой: открывающим тэгом <?php и закрывающим ?>[17] DEVACADEMY.

Преимущества языка программирования PHP:

— поддержка многих систем управления базами данных (далее СУБД) (MySQL, PostgreSQL, Sybase, Informixи других);

— доступность для большинства операционных систем (Linux, Unix, MicrosoftWindows, MacOSи других);

— большой количество библиотек и расширений языка;

— прост в освоении.

Недостатки языка программирования PHP:

— не подходит для создания десктопных приложений или системных компонентов;

— веб-приложения, написанные на PHP, зачастую имеют проблемы с безопасностью.

JavaScript — язык сценариев, или скриптов. Скрипт представляет собой программный код — набор инструкций, который не требует предварительной обработки (например, компиляции) перед запуском. Код JavaScript интерпретируется движком браузера во время загрузки веб-страницы. Интерпретатор браузера выполняет построчный анализ, обработку и выполнение исходной программы или запроса.

Изначально JavaScriptможно использовать только, как средство для клиентской стороны, однако с помощью Node.jsJavaScriptможно использовать и для серверной стороны.Node.js — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения.

Преимущества языка программирования JavaScript с использованием Node.js:

— возможность работы, как с клиентской стороны, так и с серверной стороны;

— очень высокая производительность;

— встроенная система управления модулями и зависимостями;

— прост в освоении.

Недостатки языка программирования JavaScript с использованием Node.js:

— молодость системы сказывается на том, что поддержка со стороны пользователей пока существенно ниже, чем у более старших серверных «братьев»;

— асинхронность кода может сильно его усложнить.

В итоге, исходя из сделанного сравнения, можно сказать, что Python, несмотря на множество плюсов, очень сильно проигрывает в

производительности рассмотренным в данном разделе языкам программирования. Исходя из этого, Pythonне будет использоваться в разработке автоматизированной информационной системы для обмена данными о прикреплении населения к медицинским организациям между базами данных ТФОМС РХ и РИАМС «ПроМед».Переходя к PHPи JavaScriptстоит сказать, что несмотря на то, что Node.jsдовольно молодая платформа, в создании АИС будет использоваться именно она по причинам того, что PHPпроигрывает в безопасностии производительности. Таким образом, для разработки автоматизированной информационной системы для обмена информацией между базами данных ТФОМС РХ и РИАМС «ПроМед»был выбран язык программирования JavaScript.

1.5.2 Выбор системы управления базами данных для разработки

Системы управления базами данных (СУБД) позволяют структурировать, систематизировать и организовывать данные для их компьютерного хранения и обработки. Именно базы данных являются основой практически любой информационной системы.

Взаимодействие пользователя с БД обеспечивает специальный язык запросов – StructuredQueryLanguage (SQL) (Язык структурированных запросов).

Существует множество СУБД для управления реляционными базами данных. Каждая из них обладает своими достоинствами и недостатками. Рассмотрим несколько СУБД, которые широко используются в интернете: SQLite, PostgreSQL, MariaDB.

SQLite – это, легко встраиваемая в приложения, база данных. Так как это система базируется на файлах, то она предоставляет довольно широкий набор инструментов для работы с ней, по сравнению с сетевыми СУБД. При работе с этой СУБД обращения происходят напрямую к файлам (в этих файлах хранятся данные), вместо портов и сокетов в сетевых СУБД. Именно

поэтому SQLite очень быстрая, а также мощная, благодаря технологиям обслуживающих библиотек [1].

Преимущества SQLite:

— вся база данных хранится в одном файле, что облегчает перемещение;

— стандартный функционал SQL.

Главным недостатком SQLitedля разрабатываемой системы является то, что данная СУБД не поддерживает формат данных XML. Это является критически важным недостатком данной системы, т.к. РИАМС «ПроМед» пересылает данные в формате XML.

PostgreSQL – свободно распространяемая СУБД с поддержкой объектно-ориентированного подхода к данным. Реализован принцип параллельной обработки. За счет использования встроенных хранимых процедур ускорено использование повторяемых процессов. СУБД отлично справляется и во многом ориентирована на работу со сложными структурами данных –по сравнению с другими свободно распространяемыми СУБД.

Достоинства PostgreSQL:

— открытое ПО, соответствующее стандарту SQL – PostgreSQL – бесплатное ПО с открытым исходным кодом;

— большое количество дополнений;

— расширения – существует возможность расширения функционала за счет сохранения своих процедур;

— объектность – PostrgreSQL это не только реляционная СУБД, но также и объектно-ориентированная с поддержкой наследования.

Недостатки PostgreSQL:

— Производительность – при простых операциях чтения PostgreSQL может значительно замедлить сервер и быть медленнее своих конкурентов, таких как MariaDB;

— малоизвестность.

MariaDB– ответвление СУБД MySQL, разрабатываемое сообществом. Основная цель проекта MariaDB – создание полностью бинарно-совместимой с оригинальной MySQL версии СУБД, которая при этом будет иметь значительное количество улучшений в коде, влияющих на производительность.

Преимущества MariaDB:

— высокая производительность и безопасность;

— может работать с действительно большими объёмами данных, и неплохо подходит для масштабируемых приложений;

— поддерживает большую часть функционала SQL;

— высокая стабильность.

Недостатки MariaDB:

– более низкая поддержка со стороны сообщества, в связи с небольшой популярностью.

Согласно проведенному анализу СУЬД проектирование будет осуществляться с помощью MariaDB, по причине её скорости, защищённости и простоты в использовании.

Также в проекте будет использоваться HTMLиCSS, которым не существует альтернатив.

Таким образом, для разработки автоматизированной информационной системы для передачи данных о прикреплении населения к медицинским организациям, будет использоваться JavaScript, Node.js, СУБД MariaDB, а также HTML и CSS.

1.6 Выводы по разделу «Анализ предметной области. Выбор средств проектных решений»

В данном разделе проанализирована основная деятельность учреждения ГКУЗ РХ «РМИАЦ».

Для соблюдения Федерального закона РФ от 27 июля 2006 года № 152-ФЗ «О персональных данных» использовалась защищенная сеть для передачи данных.

Было выполнено описание организации ТФОМС РХ, что позволило выявить проблему того, что у РИАМС «ПроМед» и ТФОМС РХ различные форматы данных.

Принято решение о разработке новой автоматизированной информационной системы для обмена информацией между базами данных ТФОМС РХ и РИАМС «ПроМед». Для реализации автоматизированной информационной системы для обмена информацией между базами данных ТФОМС РХ и РИАМС «ПроМед»были проанализированы наиболее популярные средства в среде современных IT-разработчиков:

– языки программирования: Phyton, PHP и JavaScript;
– СУБД: Microsoft SQL,SQLite, PostgreSQLиMariaDB.

Для разработки автоматизированной информационной системы для обмена информацией между базами данных ТФОМС РХ и РИАМС «ПроМед»был выбран язык программирования JavaScript.

Согласно проведенному анализу, проектирование СУБД будет осуществляться с помощью MariaDB, по причине её скорости, защищённости и простоты в использовании. Также в проекте будет использоваться HTMLиCSS, которым не существует альтернатив.

2 Описание разработки возможности передачи данных о прикреплении населения к медицинским организациям из РИАМС «ПроМед» в ИС ТФОМС

Для создания дополнительного функционала опишем и создадим модель потоков данных DFD.

2.1 Описание модели DFD реализации автоматизированной информационной системы для передачи данных о прикреплении к медицинской организации из РИАМС «ПроМед» в ТФОМС РХ

DFD — общепринятое сокращение от англ. data flow diagrams — диаграммы потоков данных. Так называется методология графического структурного анализа, описывающая внешние по отношению к системе источники, и адресаты данных, логические функции, потоки данных и хранилища данных, к которым осуществляется доступ. Диаграмма потоков данных (data flow diagram, DFD) — один из основных инструментов структурного анализа и проектирования информационных систем, существовавших до широкого распространения UML. Предназначенная для моделирования информационный систем с точки зрения хранения, обработки и передачи данных.

Исторически синтаксис этой нотации применяется в двух вариантах — Йордана (Yourdon) и Гейна-Сарсона (Gane-Sarson). Различия между ними – на рисунке 2.1.

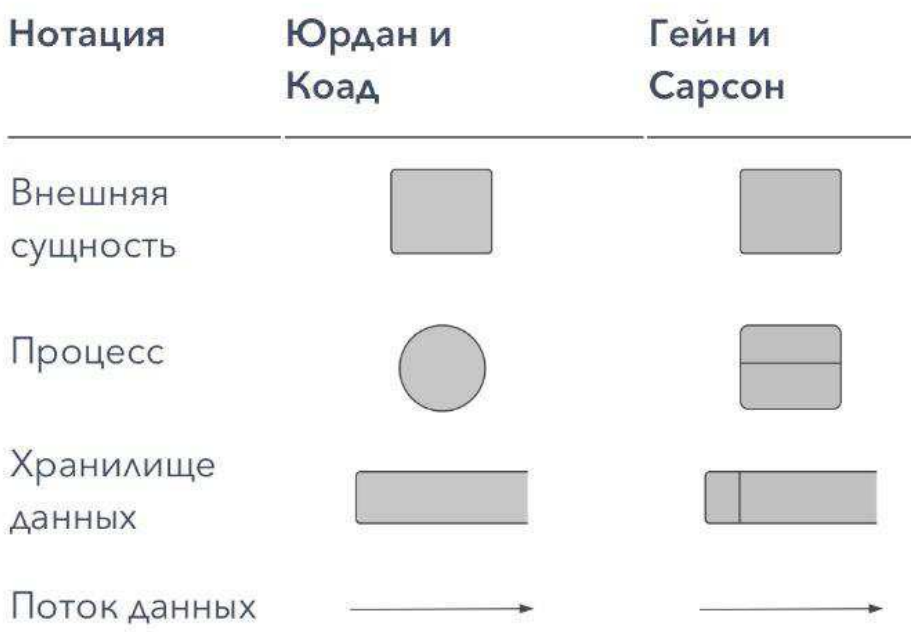

Рисунок 2.1 – Синтаксис диаграммы DFD

- **Процесс (англ. Process)**, т.е. функция или последовательность действий, которые нужно предпринять, чтобы данные были обработаны. Это может быть создание заказа, регистрация клиента и т.д. В названиях процессов принято использовать глаголы, т.е. «Создать клиента» (а не «создание клиента») или «обработать заказ» (а не «проведение заказа»). Здесь нет строгой системы требований, как, например, в IDEF0, где нотации имеют жестко определенный синтаксис, так как они могут быть исполняемыми.
- **Внешние сущности (англ. External Entity).**Это любые объекты, которые не входят в саму систему, но являются для нее источником информации либо получателями какой-либо информации из системы после обработки данных. Это может быть человек, внешняя система, какие-либо носители информации и хранилища данных.
- **Хранилище данных (англ. Data store)**. Внутреннее хранилище данных для процессов в системе. Поступившие данные перед обработкой и результат после обработки, а также промежуточные значения должны где-то храниться. Это и есть базы данных, таблицы или любой другой вариант организации и хранения данных. Здесь будут храниться данные о клиентах, заявки клиентов, расходные накладные и любые другие данные, которые поступили в систему или являются результатом обработки процессов.
- **Поток данных (англ. Data flow)**. В нотации отображается в виде стрелок, которые показывают, какая информация входит, а какая исходит из того или иного блока на диаграмме[19] DFD.

Диаграмма потоков данных для дополнительного функционала передачи данных о прикреплении пациентов к медицинской организации представлена на рисунке 2.2.

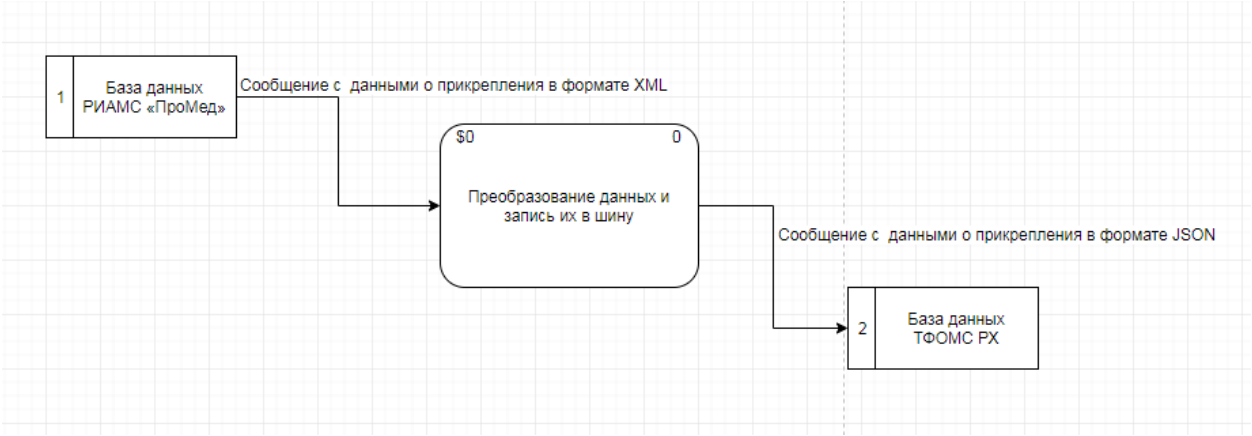

Рисунок 2.2 – Диаграмма потоков данных для передачи данных о прикреплении

На данной диаграмме представлены 1 внешняя сущность "Преобразование данных и запись их в шину" и два хранилища данных: "База данных РИМАС «ПроМед»" и "База данных ТФОМС РХ". Две дуги потоков данных "Данные о прикреплении в формате XML" и "Данные о прикреплении в формате JSON".

На диаграмме потоков данных показано, как данные о прикреплении людей в формате XMLприходят в шину из базы данных РИАМС. После этого обработанные данные в формате JSONотправляются в базу данных ТФОМС РХ.

Декомпозиция шины первого уровня диаграммы потоков данных представлена ниже на рисунке 2.3.

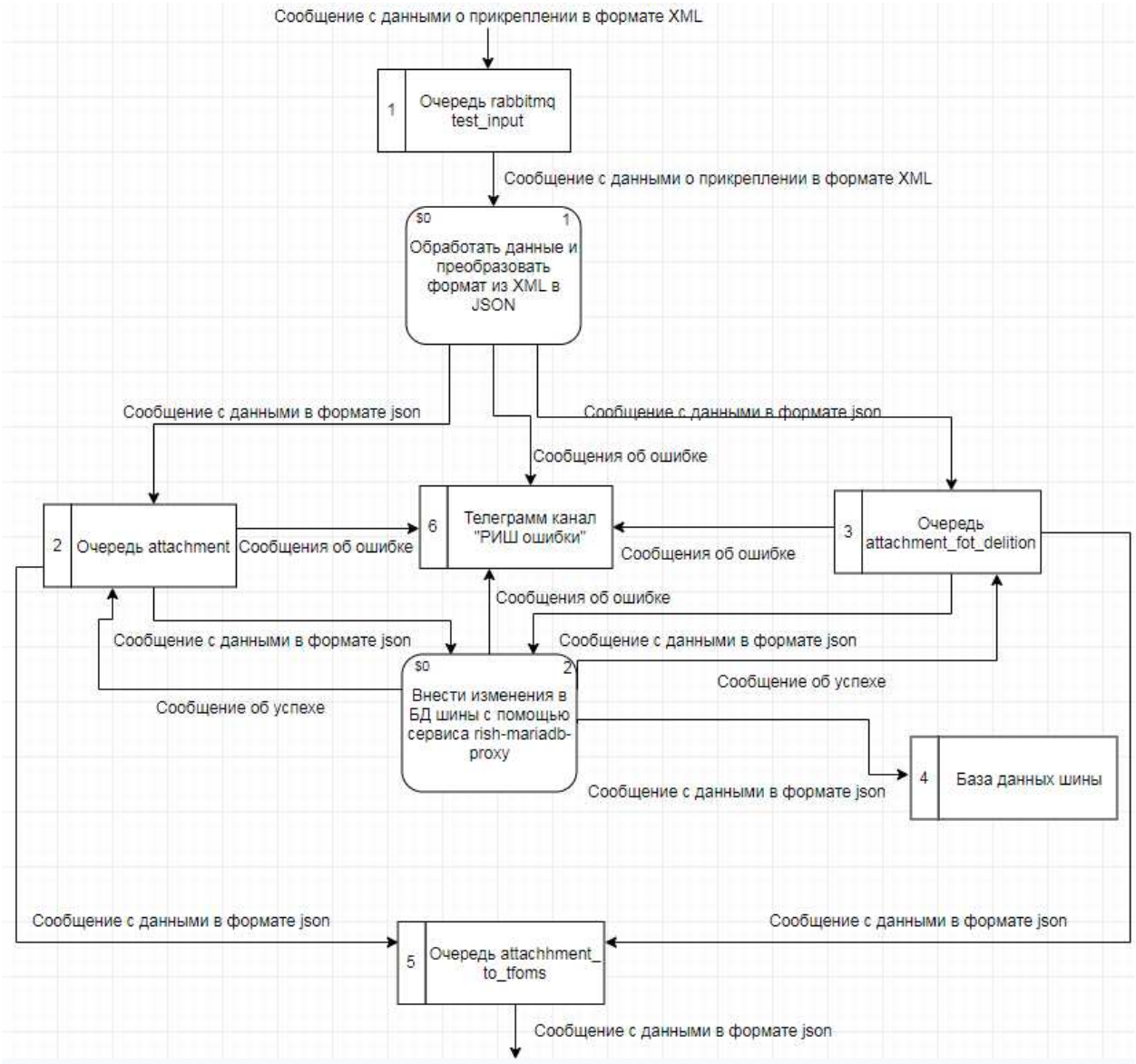

Рисунок 2.3 – Декомпозиция шины первого уровня диаграммы потоков данных

На данной диаграмме показаны 5 хранилищ данных и 2 процесса. Сообщение с данными о прикреплении поступают в очередь шины test_inputиз вне в формате XML. Дальше шина преобразовывает формат данных из XMLв JSON и отправляют либо в очередь attachment, либо в очередь attachment_for_deletion – зависит от запрашиваемого действия. Потом сообщение с данными вносятся в БД шины и при успехе отправляются в очередь attachment_to_tfoms для дальнейшей отправки в

ТФОМС. Также в случае ошибки, на любом из этапов, отправляется сообщение о самой ошибке в телеграмм канал «РИШ ошибки».

2.2 Описание модели IDEF3для реализации автоматизированной информационной системы для передачи данных о прикреплении к медицинской организации из РИАМС «ПроМед» в ИС ТФОМС РХ

IDEF3 является стандартом документирования технологических процессов, происходящих на предприятии, и предоставляет инструментарий для наглядного исследования и моделирования их сценариев. Исполнение каждого сценария сопровождается соответствующим документооборотом, который состоит из двух основных потоков: документов, определяющих структуру и последовательность процесса (технологических указаний, описаний стандартов и т.д.), и документов, отображающих ход его выполнения (результатов тестов и экспертиз, отчетов о браке, и т.д.). Для эффективного управления любым процессом, необходимо иметь детальное представление об его сценарии и структуре сопутствующего документооборота.

Прямоугольники на диаграмме IDEF3 называются функциональными элементами или элементами поведения (Unit of Behavior, UOB) и обозначают событие, стадию процесса или принятие решения. Каждый UOB имеет свое имя, отображаемое в глагольном наклонении и уникальный номер. Стрелки или линии являются отображением перемещения детали между UOB-блоками в ходе процесса. Линии бывают следующих видов:

- Старшая (Precedence) – сплошная линия, связывающая UOB. Рисуется слева направо или сверху вниз.

- Отношения (Relational Link) – пунктирная линия, использующаяся для изображения связей между UOB

- Потоки объектов (Object Flow) – стрелка с двумя наконечниками используется для описания того факта, что объект (деталь) используется в

двух или более единицах работы, например, когда объект порождается в одной работе и используется в другой.

Объект, обозначенный J1 – называется **перекрестком** (Junction). Перекрестки используются для отображения логики взаимодействия стрелок (потоков) при слиянии и разветвлении или для отображения множества событий, которые могут или должны быть завершены перед началом следующей работы. Различают перекрестки для слияния (Fan-in Junction) и разветвления (Fan-out Junction) стрелок. Перекресток не может использоваться одновременно для слияния и для разветвления. При внесении перекрестка в диаграмму необходимо указать тип перекрестка.

На рисунке Е.1 в приложении Е показана IDEF 3 диаграмма работы разрабатываемого функционала шины.

2.3 Подготовка среды разработки

Для создания дополнительного функционала нужно подготовить среду разработки, т.е. установить и настроить программное обеспечение (далее ПО). Сначала скачаем и установим OracleVMVirtualBoxManager, после этого поставим операционную систему Linux на виртуальную машину.

Все прописанные ниже команды со знаком “$” в начале, пишутся в приложении Terminal операционной системыLinux.

2.3.1 Установка nodeJS

Для установки воспользуемся пакетным менеджером apt. Сначала обновим локальный индекс пакетов:

$sudoaptupdate

Теперь установим Node.js из репозиториев:

$sudoaptinstallnodejs

Если пакет из репозиториев удовлетворяет потребностям, то на этом установка Node.js закончена. Однако в большинстве случаев также потребуется установить npm– менеджер пакетов для Node.js. Это нужносделать при помощи следующей команды:

$sudoaptinstallnpm [26].

2.3.2Установка MariaDB

Для Ubuntu 18.04 надо выполнить три команды:

$sudo apt-get install software-properties-common

$sudo apt-key adv --recv-keys --keyserver hkp://keyserver.ubuntu.com:80 0xF1656F24C74CD1D8

$sudo add-apt-repository 'deb [arch=amd64,arm64,ppc64el] http://mirror.mephi.ru/mariadb/repo/10.3/ubuntu bionic main'

Затем обновим списки репозиториев и установим MariaDBUbuntu нужной версии:

$sudo apt install mariadb-server mariadb-client

После того, как установка будет завершена, нужно проверить запущена-ли база данных:

$sudo systemctl status mariadb

Настройкамariadbвubuntu. Сразу после установки MariaDB ещё не готова к работе. Для обеспечения её безопасности необходимо выполнить команду:

$sudomysql_secure_installation [25].

2.3.3 Установка Git

Для установки Git'a понадобятся библиотеки, от которых он зависит: curl, zlib, openssl, expat и libiconv. Например, если в системе менеджер пакетов — yum (Fedora), или apt-get (Debian, Ubuntu), можно воспользоваться следующими командами, чтобы разрешить все зависимости:

```
$ yum install curl-devel expat-devel gettext-devel \
  openssl-devel zlib-devel
$ apt-get install libcurl4-gnutls-dev libexpat1-dev gettext \
  libz-devlibssl-dev
```

Установив все необходимые библиотеки, можно идти дальше и скачать последнюю версию с сайта Git'a:

http://git-scm.com/download

Теперь скомпилируйте и установите:

```
$ tar -zxf git-1.7.2.2.tar.gz
$ cd git-1.7.2.2
$ make prefix=/usr/local all
$ sudo make prefix=/usr/local install
```

После нужно скачать Git с помощью самого Git'a, чтобы получить обновления:

```
$ git clone git://git.kernel.org/pub/scm/git/git.git
```

Установка в Linux

1Установить Git под Linux как бинарный пакет, можно сделать, используя обычный менеджер пакетов дистрибутива:

```
$ yuminstallgit-core [4].
```

2.3.4 Установка Docker

Для начала обновим базу данных пакетов:

```
$sudo apt-get update
```

Теперь установим Docker. Добавьте ключ GPG официального репозитория Docker в систему:

$sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys 58118E89F3A912897C070ADBF76221572C52609D

Добавим репозиторий Docker в список источников пакетов утилиты APT:

$sudo apt-add-repository 'deb https://apt.dockerproject.org/repo ubuntu-xenial main'

Обновим базу данных пакетов информацией о пакетах Docker из вновь добавленного репозитория:

$sudo apt-get update

Далее, установим Docker:

$sudo apt-get install -y docker-engine

После завершения выполнения этой команды Docker должен быть установлен и процесс должен запускаться при загрузке системы. Проверим, что процесс запущен:

$sudo systemctl status docker[28].

Далее удаляем старые контейнеры Docker:

$sudo systemctl start docker
$docker kill $(docker ps -q)
$docker rm $(docker ps -a -q)
$docker rmi $(docker images -q)

2.4 Разработка автоматизированной информационной системы для передачи данных из РИАМС «ПроМед» в шину Rish

Для разработки дополнительного функционала передачи данных из РИАМС «ПроМед» в шину Rishнеобходимо реализовать следующие функции:

- принятие информации из базы данных РИАМС «ПроМед»;
- преобразование данных из формата XMLв JSON;

- отправка данных о прикреплении в очереди attachment_for_deletion и attachment для вставки или удаления из БД шины
- преобразование данных о прикреплении из формата XML в JSON

2.4.1 Разработка функционала принятия сообщения с данными о прикреплении из базы данных РИАМС «ПроМед»

Для того, чтобы создать функцию принятия информации из базы данных РИАМС «ПроМед», первое, что требуется сделать, это начать «слушать» очередь данных, поставляемых из РИАМС «ПроМед». После этого идет принятие данных из РИАМС «ПроМед» в строковом виде и преобразование этих данных из формата XMLв JSON. Однако для преобразования данных требуется создать отдельную функцию. Функция принятия информации из базы данных из РИАМС «ПроМед» представлена в приложении А.

2.4.2 Разработка функционала преобразования данных о прикреплении из формата XML в JSON

Дальше создается функция преобразования XMLданных в формат JSON.Ниже приведены примеры XMLиJSONформатов. ПримерXML-файла представлен на рисунках 2.3 и 2.4.

```
ilya@ilya-VirtualBox:~/rish$ node xmlToJs.js
<?xml version="1.0" encoding="utf-8"?>
<PERSONATTACHDISTRICT>
<HEADER>
<OPERATIONTYPE>Insert</OPERATIONTYPE>
<CODE_MO>190004</CODE_MO>
<PERSONATTACHID>77373602</PERSONATTACHID>
</HEADER>
<BODY><BDZID>400327</BDZID><FAM>Алексеев</FAM><IM>Юрий</IM><OT>Иванович</OT>
<W>2</W>
<DR>1987-03-15</DR>
<DOCTYPE>4</DOCTYPE>
<DOCSER>72 46</DOCSER>
<DOCNUM>470704</DOCNUM>
<INFO_TYPE>1</INFO_TYPE>
<ATTACH_TYPE>2</ATTACH_TYPE>
<ATTACH_DT_MO>1997-10-02</ATTACH_DT_MO>
<DETACH_DT_MO>2019-05-13</DETACH_DT_MO>
<DETACH_CAUSE_MO>1</DETACH_CAUSE_MO>
<PODR>1</PODR>
<OTD>918</OTD>
<UCH>5</UCH>
<UCH_TYPE>01</UCH_TYPE>
<ATTACH_DT>1995-01-12</ATTACH_DT>
</BODY>
</PERSONATTACHDISTRICT>
```

Рисунок 2.3 – XML формат данных

Тот же файл, в формате JSON представлен на рисунке 2.4.

```
parsed-content { meta: { OPERATIONTYPE: 'Insert' },
  data:
   { BDZID: '400327',
     CODE_MO: '190004',
     INFO_TYPE: '1',
     ATTACH_DT_MO: '1997-10-02',
     DETACH_DT_MO: '2019-05-13',
     DETACH_CAUSE_MO: '1',
     PODR: '1',
     OTD: '918',
     UCH: '5',
     UCH_TYPE: '01',
     created_at: '2019-06-12',
     deleted: 0,
     PERSONATTACHID: '77373602' } }
```

Рисунок 2.4 – JSON формат данных

Данное преобразование необходимо в связи с тем, что форматы данных у РИАМС «ПроМед» (XML) и ТФОМС РХ (JSON) не совпадают. Функция преобразования данных из формата XML в JSON представлена в приложении В.

2.4.3 Разработка функционала отправки данных о прикреплении в очереди attachment_for_deletion и attachment для вставки или удаления из БД шины

Дальше делается проверка на запрашиваемое действие (action). Вслучаеaction: InsertилиUpdateданные отправляются в очередь attachmentдля дальнейшей работы с ними.В случае action: Delete данные отправляются в очередь attachment_for_deletion. Если actionуказана не верно дальнейшая работа прекращается, сообщение с данными отбрасывается и идет отправка сообщения об ошибке в телеграмм канал «РИШ ошибки». Код данного функционала представлен в приложении В.

2.4.4 Разработка функционала добавления или удаления данных о прикреплении из БД шины

После того, как данные попали в очереди attachment_for_deletion или attachment – делается прием данных из очереди и внесение изменений в базу данных шины. Код данного функционала представлен в приложении Г.

2.5 Выводы по разделу «Описание разработки возможности передачи данных о прикреплении населения к медицинским организациям из РИАМС «ПроМед» в ИС ТФОМС»

В проектном разделе была построена модель по методологии потоков данных DFD для того, чтобы понять, как будут передаваться данные о

прикреплении пациента к медицинской организации из РИАМС «ПроМед» в ИС ТФОМС РХ.

После этого были рассмотрены программные средства, которые использовались при создании АИС и их установка.

Была разработана автоматизированная информационная система для передачи данных из РИАМС «ПроМед» в шину Rish, которая включала в себя создание таких функций, как:

- принятие информации из базы данных из РИАМС «ПроМед»;
- преобразование данных из формата XMLв JSON.

Также была разработана возможность генерировать преобразованные ненастоящие данные из РИАМС «ПроМед».

3 Оценка экономической эффективности автоматизированной информационной системы для передачи данных о прикреплении населения к медицинским организациям из РИАМС «ПроМед» в ИС ТФОМС РХ

Для того, чтобы провести расчет затрат реализации автоматизированной информационной системы для обмена информацией между базами данных ТФОМС РХ и РИАМС «ПроМед», для начала, нужно разобраться с тем, кто именно будет связан с создаваемой системой и с помощью каких устройств он будет взаимодействовать с этой системой, а также какое программное обеспечение (ПО) необходимо для разработки АИС.

Чтобы показать кто связан с системой, нужно построить таблицу 3.1, в которой будет описана деятельность персонала.

Таблица 3.1 – Персонал, связанный с проектом

Должность	Заработная плата, руб.	Продолжительность работы в проекте
Инженер-программист	13 050	2 месяца

Исходя из таблицы видно, что только один человек связан с системой, поэтому для данного человека нужно определить стоимость оборудования для работы, которая представлено в таблице 3.2. Также этот человек будет сопровождать автоматизированную информационную систему для передачи данных о застрахованных лицах, стоящих на диспансерном наблюдении, из РИАМС «ПроМед» в ИС ТФОМС РХ после внедрения.

Таблица 3.2 – Стоимость оборудования

Наименование оборудования	Стоимость, руб.	Срок эксплуатации
Компьютер (IntelCorei3 3460, 8 GBDDR3, 120gbSSD)	18 750	5 лет
Монитор 23.8" AOC I2480SX	7 899	5 лет
Дополнительное оборудование	1000	5 лет
Итого	27 649	5 лет

В АИС для передачи данных о прикреплении населения к медицинским организациям из РИАМС «ПроМед» в ИС ТФОМС РХ используется такое ПО, как: операционная система Linux, Node.js, MariaDB, Git, RabbitMQ, Docker. Всё ПО распространяется бесплатно.

3.1 Расчет затрат реализации проекта создания ИС

Полученные данные из подраздела 3.1 дают возможность рассчитать ТСО (совокупную стоимость владения). Для расчета затрат реализации проекта была выбрана методика ТСО, которая рассчитывается по формуле:

$$ТСО = DE + IC_1 + IC_2, \quad (1)$$

где DE – прямые затраты;

$IC_{1,2}$ – косвенные расходы первой и второй группы, но в данном проекте они отсутствуют.

3.1.1 Капитальные затраты

Расчет капитальных затрат происходит по следующей формуле:

$$К = К_{пр} + К_{тс} + К_{лс} + К_{по} + К_{ио} + К_{об} + К_{оэ}, \quad (2)$$

где $К_{пр}$ – затраты на разработку информационной системы;

$К_{тс}$– затраты на технические средства управления;

$К_{лс}$– затраты на создание линий связи, а также интернет соединений;

$К_{по}$ – затраты на программные средства для использования готового программного продукта;

$К_{ио}$ – затраты на формирование информационной базы.

$К_{об}$ – заработная плата работника, обучающего персонал работать с новым программным продуктом;

$К_{оэ}$ – состав затрат, соответствующих эксплуатационным затратам.

Расчет затрат на разработку информационной системы ($К_{пр}$) происходит по следующей формуле:

$$К_{пр} = К_{зп} + К_{ипс} + К_{свт} + К_{проч}, \quad (3)$$

где $К_{зп}$ – затраты на заработную плату инженера;

$К_{ипс}$ – затраты на инструментальные программные средства для проектирования;

$К_{свт}$ – затраты на средства вычислительной техники для проектирования;

$К_{проч}$ – прочие затраты на разработку;

Затраты на заработную плату разработчиков.

Для расчета заработной платы проектировщиков ($К_{зп}$) нужно составить таблицу заработной платы и результат умножить на продолжительность работы в проекте и умножить получившуюся сумму на ФОТ (Фонд оплаты труда), который составляет 32%. Расчет зарплаты проектировщиков показан на таблице 3.3.

Районный и северный коэффициенты в сумме равны 60%, т.к. разработка ведётся в Республике Хакасия. Рабочая норма инженеров в месяц составляет 22 дня.

Таблица 3.3 – Расчет заработной платы разработчика

Код	Начисление/Удержание	Начислено	Удержано
003	Оклад	13 050	
181	Районный коэффициент + северный коэффициент	7 830	
202	НДФЛ		2 714
Итого		20880	2 714

Итого: $К_{зп} = 20880 \cdot 1{,}302 \cdot 2 = 54\,371$ (руб.)

Затраты на инструментальные ПО для проектировщика ($К_{ипс}$), исходя из раздела 3.1 составляют 0 руб.

Затраты на средства вычислительной техники для проектирования ($К_{свт}$), ввиду наличия всего необходимого оборудования для разработки системы на рабочем месте проектировщика, включают лишь сумму амортизационных отчислений за период создания проекта ($А_{пэвм}$) и равняются произведению амортизационных отчислений в день на количество дней эксплуатации компьютера при создании системы.

$$А_г = С_б \cdot Н_{ам}, \quad (4)$$

где $А_г$ – сумма годовых амортизационных отчислений;

$С_б$ – балансовая стоимость компьютера, р./шт.;

$Н_{ам}$ – норма амортизации, %.

$А_г = 27\ 649 \cdot 0,20 = 5\ 530$ (руб.)

$А_{пэвм} = (5\ 530 \cdot 60)/264 = 1256$ (руб.)

Итого: $К_{свт} = 1\ 256$(руб.)

Прочие затраты на разработку ($К_{проч}$) обычно представляет из себя резерв на непредусмотренные расходы, который составляет 3% от общей суммы затрат на разработку. Такие расходы формируются из плат за свет, интернет.

Итого: $К_{проч} = (54\ 371 + 1\ 256) \cdot 3\% = 1\ 668$ (руб.)

Затраты на разработку, которые рассчитываются по формуле (3) информационной системы равны:

$К_{пр} = 54\ 371 + 1\ 668 + 1\ 256 = 57\ 295$ (руб.)

Затраты на технические средства управления ($К_{тс}$)отсутствуют, потому что разрабатываемая АИС функционирует отдельно от других систем, использующихся на предприятии.

Затраты на создание линий связи локальных сетей ($К_{лс}$) исключаются, так как на предприятии функционирует собственная локальная сеть, удовлетворяющая новым требованиям с учетом работы разработанной системы.

Затраты на программные средства ($К_{по}$) не учитываются, так как все используемые для разработки системы программные средства являются свободно распространяемыми.

Затраты на формирование информационной базы ($К_{ио}$) включают в себя заработную плату проектировщика за время создания информационной базы (5дней – 1 рабочая неделя), а также стоимость программного обеспечения для её создания. Программное обеспечение для создания информационной базы не учитывается, так как оно является свободно распространяемым. Таким образом данный вид затрат составляет:

Итого: $К_{ио}$= 20880/4 = 5220 (руб.)

Затраты на обучение персонала ($К_{об}$) не учитываются, т.к. автоматизированная информационная система для обмена данными о застрахованных лицах, стоящих на диспансерном наблюдении между базами данных ТФОМС РХ и РИАМС «ПроМед» в будущем будет сопровождаться самим разработчиком.

Затраты на опытную эксплуатацию ($К_{оэ}$)включают в себя заработную плату проектировщика за время тестирования информационной системы (5дней – 1 рабочая неделя). Таким образом данный вид затрат составляет:

Итого: $К_{оэ}$= 20880/4 = 5220 (руб.)

Список капитальных затрат показан в таблице 3.4 и рисунке 3.1.

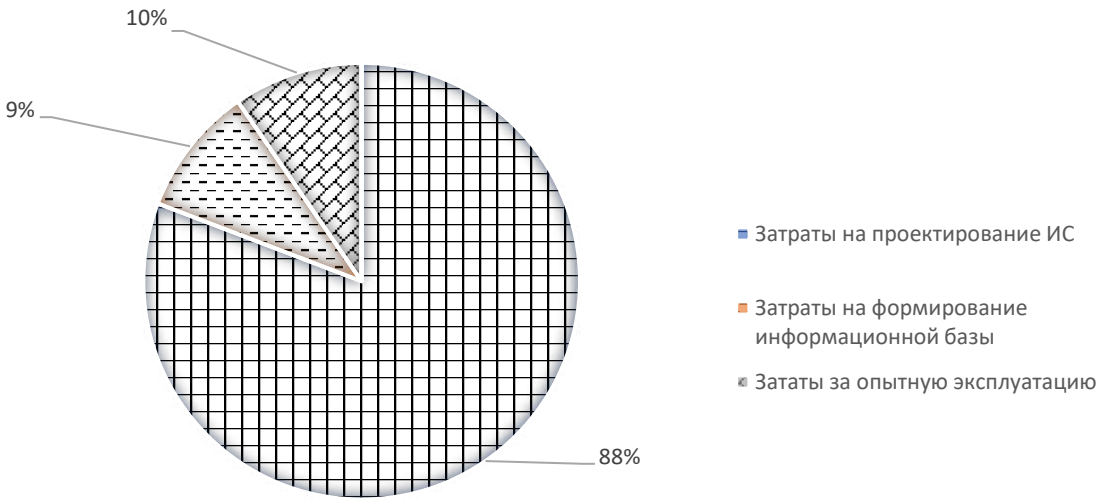

Рисунок 3.1 – Диаграмма капитальных затрат

Таблица 3.4 – Список капитальных затрат

Затраты	Состав затрат	Планируемая сумма, руб.
Затраты на разработку ИС	Затраты на заработную плату разработчика	54 371
	Затраты на инструментальные программы средства для проектирования	0
	Затраты на средства вычислительной техники для проектирования	1 668
	Прочие затраты на разработку	3 695
	Затраты на технические средства управления	0
	Затраты на создание линий связи локальных сетей	0
	Затраты на программные средства	0
	Затраты на формирование информационной базы	5220
	Затраты на обучение персонала	0
	Затраты на опытную эксплуатацию	5220

Капитальные затраты, которые рассчитываются по формуле (2), равны:

К = 54 371 + 5220+ 5220= 64811 (руб.)

3.1.2 Эксплуатационные затраты

Расчет эксплуатационных затрат происходит по следующей формуле:

$$С = С_{зп} + С_{ао} + С_{то} + С_{лс} + С_{ни} + С_{проч}, \quad (5)$$

где $С_{зп}$ – зарплата персонала, работающего с информационной системой:

$С_{ао}$ – амортизационные отчисления;

$С_{то}$ – затрата на техническое обслуживание;

$С_{лс}$ – затраты на использование глобальных сетей;

$С_{ни}$ – затраты на носители информации;

$С_{проч}$ – прочие затраты.

Зарплата управленческого персонала ($С_{зп}$), работающего с использованием ИС (пользователей ИС) не учитывается, так как ввод в эксплуатацию разработанной АИС не влияет существенным образом на их работу.

Амортизационные отчисления ($С_{ао}$) состоят из стоимости используемого для работы системы оборудования (в данном случае – сервера) и не будут учитываться, т.к. система уже размещена на сервере виртуальной машины.

Затраты на техническое обслуживание (Сто) состоят из заработной платы администратора системы за год. Администратору предстоит тратить на обслуживание разработанной АИС около 1 часа рабочего времени в неделю.

Таким образом в год будет уходить 48 часов на обслуживание АИС, а это 6 рабочих дней (0,2 от заработной платы в месяц), а именно:

$C_{то} = 20\ 880 \cdot 0{,}2 = 4\ 176$ (руб.)

Затраты, связанные с использованием глобальных вычислительных сетей ($C_{лс}$),не учитываем ввиду использования на предприятии безлимитного интернета и несущественного трафика, создаваемого системой, по сравнению с общим трафиком предприятия.

Затраты на носители информации ($C_{ни}$) исключаются, так как система располагается на сервере и не требует дополнительных носителей информации.

Прочие затраты ($C_{проч}$) составляют 3 % от общей суммы эксплуатационных затрат.

$C_{проч} = 4\ 176 \cdot 3\% = 125$ (руб.)

Список эксплуатационных затрат показан в таблице 3.5 и рисунке 3.2.

Таблица 3.5 – Список эксплуатационных затрат

Состав затрат	Планируемая сумма рублей в год, руб.
Затраты на заработную плату персонала	0
Затраты на амортизацию отчисления	0
Затраты на техническое обслуживание	4 176
Затраты на использование глобальных сетей	0
Затраты на носители информации	0
Прочие затраты	125

Таким образом, сумма годовых эксплуатационных затрат, рассчитываемых по формуле (5), составляет:

C = 4 176+ 125= 4 301 (руб.)

3.1.3Прямые затраты

Прямые затраты равны:

$$DE = DE_1 + DE_2 + DE_3 + DE_4 + DE_5 + DE_6 + DE_7 + DE_8, \quad (6)$$

где DE_1 – капитальные затраты;

DE_2 – расходы на управление информационными технологиями;

DE_3 – расходы на техническую поддержку;

DE_4 – расходы на разработку прикладного ПО внутренними силами;

DE_5 – расходы на аутсорсинг;

DE_6 – командировочные расходы;

DE_7 – расходы на услуги связи;

DE_8– другие группы расходов.

Капитальные затраты (DE_1), рассчитанные ранее, составляют 62 723 (руб.)

Расходы на управление ИТ (DE_2) в соответствии с эксплуатационными затратами не учитываются.

Расходы на техническую поддержку АО и ПО (DE_3) составляют 4 176 руб.

Расходы на разработку прикладного ПО внутренним силам (DE_4) включены в капитальные затраты.

Расходы на аутсорсинг (DE_5) не учитываются, так как услуги аутсорсеров не использовались.

Командировочные расходы (DE_6) не учитываются ввиду отсутствия командировок.

Расходы на услуги связи (DE_7) исключаются, так как данная статья расходов не относится к проекту.

Другие группы расходов (DE_8) включают в себя прочие эксплуатационные затраты и составляют 125 руб.

Таким образом, сумма прямых расходов, рассчитываемые по формуле (6), составляет:

DE = 64 811+ 4 176 + 125 = 69112 (руб.)

Итого, совокупная стоимость владения информационной системой, рассчитываемая по формуле (1), составляет:

TCO = 69112 (руб.)

3.2 Экономическая эффективность

Автоматизированная информационная система для обмена данными о застрахованных лицах, стоящих на диспансерном наблюдении между базами данных ТФОМС РХ и РИАМС «ПроМед» создаётся с целью сокращения времени обработки данных. В отличии от того, что, если бы врачам пришлось передавать данные по электронной почте, созданная АИС позволяет сделать передачу данных в несколько раз быстрее.

Значимость технических решений(ЗТР) вычисляется по следующей формуле:

$$ЗТР = k_а \cdot k_п \cdot k_с + k_м \cdot k_о \cdot k_ш, \quad (7)$$

где $k_а$–коэффициент актуальности;

$k_п$– коэффициент соответствия программам важнейших работ научно-технического прогресса;

$k_с$ – коэффициент сложности;

$k_м$ – коэффициент места использования;

$k_о$ – коэффициент объема использования;

$k_ш$ – коэффициент широты охвата охранными мероприятиями;

В таблице 3.6 приведены коэффициенты и ЗТР базового и разрабатываемого вариантов проекта.

Таблица 3.6 – Коэффициенты и ЗТР базового и разрабатываемого варианта

Коэффициенты	Базовый вариант	Разрабатываемый вариант
$k_а$	1	2
$k_п$	1	1
$k_с$	1	1
$k_м$	1	1
$k_о$	1	2
$k_ш$	1	2
ЗТР	2	6

Таким образом, из данной таблицы видно, что разрабатываемая АИС имеет более высокий показатель эксплуатационно-технического уровня по сравнению с базовым вариантом. Вычисляем коэффициент эксплуатационно-технического уровня $k_{эту}$ по формуле:

$$k_{эту} = \frac{ЗТРпр}{ЗТРбаз}\ , \qquad (8)$$

где $ЗТР_{пр}$ и $ЗТР_{баз}$ – значимость технического решения для проекта и для базового варианта соответственно.

$k_{эту} = \frac{6}{2} = 3$

$k_{эту} > 1$, следовательно, разработка проекта является оправданной с технической точки зрения.

Вычислим комплексный показатель качества проекта по группе показателей $I_{эту}$ по формуле:

$$I_{эту} = \sum bi * Xi, \quad (9)$$

где b_i – коэффициент весомости iго показателя;

X_i–относительный показатель качества, устанавливаемый экспертным путем по выбранной шкале оценивания.

Для оценки $I_{эту}$ рекомендуется пятибалльная шкала оценивания.

В таблице 3.7 приведен расчет показателя качества.

Таблица 3.7 – Расчет показателя качества

Показатель качества	Весовой коэффициент, bi	Оценка, Xi	
		Разрабатываемый проект	Базовый проект
Удобство работы (пользовательский)	0,2	5	1
Надежность (защита данных)	0,2	4	2
Функциональные возможности	0,1	1	1
Временная экономичность	0,4	4	1
Время обучения персонала	0,1	1	1
Комплексный показатель качества $I_{эту}$		3,4	1,3

Коэффициент технического уровня:

$$k_т = \frac{I\text{этупр}}{I\text{этубаз}}, \quad (10)$$

где $I_{этупр}$ и $I_{этубаз}$ – комплексные показатели качества, разрабатываемого и базового проектов.

$k_T = \frac{3,4}{1,3} = 2,6$

Для расчета экономического эффекта рассчитаем приведенные затраты $З_i$на единицу работ, выполняемых по базовому и разрабатываемому вариантам, по формуле:

$$З_i = C_i + E_i * З_{ппi}, \quad (11)$$

где C_i–текущие эксплуатационные затраты единицы іго вида работ, р.

$З_{ппi}$–суммарные затраты, связанные с внедрением проекта;

$E_н = 0,33$ – нормативный коэффициент экономической эффективности;

Для базового варианта:

До внедрения проекта эта работа проделывалась вручную и занимала 33 часа в месяц.

20880/176 = 119 руб. - стоимость одного часа работы

33 * 119 = 3942,8 - стоимость работы за один месяц

Ci= 12 * 3942,8 =47314 - стоимость работы за один год

$З_{баз}$= 47314+ 0,33 * 0 = 47 314 (руб.)

Для проекта:

$З_{пр}$= 4 301+ 0,33 * 64811= 25689(руб.)

Экономический эффект от использования разрабатываемой системы определяется по формуле:

$$Э = (З_{баз} \cdot k_т – З_{пр}) \cdot V, \quad (12)$$

где $З_{баз}$, $З_{пр}$ – приведенные затраты на единицу работ, выполняемых с помощью базового и проектируемого вариантов процесс обработки информации, р.;

$k_т$ – коэффициент эксплуатационно-технической эквивалентности; V– объем работ, выполняемых с помощью разрабатываемого проекта, натуральные единицы.

Экономический эффект от использования разрабатываемой системы:

$$Э = (47\ 314 \cdot 2{,}6 - 25689) \cdot 1 = 97327$$

Также необходимо рассчитать срок окупаемости затрат на разработку проекта по формуле:

$$Т_{ок} = \frac{Зпп}{Э}, \quad (13)$$

где $З_{пп}$ – единовременные затраты на разработку проекта, р.;

Э – годовая эффективность, р.

Рассчитываемый срок окупаемости затрат на разработку продукта:

$$Т_{ок} = \frac{54\ 964}{97327} = 0{,}57$$

Таким образом, срок окупаемости составляет примерно полгода.

Фактический коэффициент экономической эффективности разработки (Еф):

$$Е_ф = \frac{1}{Ток} \quad (14)$$

Нормативное значение коэффициента эффективности капитальных вложений $Е_н$ = 0,33, если $Е_ф > Е_н$, то делается вывод об эффективности капитальных вложений $Е_н$ = 0,33, если $Е_ф > Е_н$, то делается вывод об эффективности капитальных вложений.

Рассчитаем фактический коэффициент экономической эффективности разработки ($E_ф$):

$$E_ф = \frac{1}{0,57} = 1,75$$

Так как $E_ф = 1,75 > E_н$, то разработка и внедрение разрабатываемого продукта являются эффективными, т. е. эффект от использования данной системы окупает все затраты, связанные с проектированием и эксплуатацией

В таблице 3.8 приведены сводные данные экономического обоснования

Таблица 3.8 – Сводные данные экономического обоснования

Показатель	Величина
Затраты на разработку проекта	22 314
Общие эксплуатационные затраты	4 301
Экономический эффект	97327
Коэффициент экономической эффективности	1,75
Срок окупаемости	6 мес.

Рассчитанная экономическая эффективность дает возможность оценить риски в данном проекте.

3.3 Оценка риска при реализации проекта создания АИС

Наиболее сильно влияющими на реализацию проекта рисками являются:

Организационный риск. Этот риск имеет высокий уровень влияния на проект, так как может существенно увеличить стоимость разработки и внедрения проекта из-за того, что может увеличится время разработки. Однако вероятность данного риска минимальна, так как между ГКУЗ РХ «РМИАЦ» и ТФОМС РХ заключён договор.

Лимитированное время разработки. Данный риск имеет высокий уровень влияния на проект, потому что, если разработка проекта не уложится в срок, то это повлечет за собой дополнительные капитальные расходы. При этом вероятность риска средняя, так этот проект является не основной задачей разработчика на работе, что может повлечь за собой задержки при разработке.

Неучтенные дополнительные расходы. Этот риск имеет средний уровень влияния на проект, так как это может затормозить разработку проекта и увеличить расходы. При этом вероятность риска средняя, так как учесть все дополнительные расходы почти не возможно.

Реализационный риск. Этот риск имеет средний уровень влияния на проект, так как может увеличить стоимость разработки из-за возможного добавления нового функционала. Вероятность риска низкая, потому что список требований, которым должна соответствовать ИС, оглашен в договоре и вряд ли будет изменён.

Дополнительные расходы на доработку проекта. Этот риск имеет низкий уровень влияния на проект, так как он немного повышает операционные расходы на разработку проекта. При этом вероятность риска высокая, потому что все ошибки при разработке найти почти невозможно.

Таблица Д.1 со списком рисков представлена в приложении Д.

3.4 Меры по предотвращению или снижению риска

Шанс срабатывания организационного риска в данном проекте невероятно мал благодаря тому, что между организациями заключён договор.

Лимитированное время разработки. Тщательное продумывание плана перед разработкой поможет более четко сформулировать временные границы для разрабатываемого проекта, а это существенно снизит вероятность данного риска.

Неучтенные дополнительные расходы. Дополнительное соглашение к договору, т.е. увеличение бюджета на дополнительные расходы. Это значительно снизит вероятность риска.

Реализационный риск. Создать дополнительное соглашение к договору, где будут максимально расписаны, возможные непредвиденные расходы. Это существенно снизит шанс срабатывания риска.

Дополнительные расходы на доработку проекта. Регулярные тесты могут значительно снизить вероятность данного риска, однако это повлечет за собой увеличение стоимости проекта.

3.5 Вывод по разделу«Оценка экономической эффективности автоматизированной информационной системы для передачи данных о прикреплении населения к медицинским организациям из РИАМС «ПроМед» в ИС ТФОМС РХ»

В данном разделе были выявлены технические характеристики проекта, также были посчитаны капитальные затраты, которые составляют 54 371(руб.), и эксплуатационные затраты, которые составляют 4176(руб.), что позволило рассчитать совокупную стоимость владения информационной системой (ТСО), сумма которого составляет 69 112 (руб.).Срок окупаемости составляет шесть месяцев, а это показывает, что разработанная АИС экономически эффективна, т.к. количество затрат существенно снизится. Были определены риски и возможность их предотвращения.

ЗАКЛЮЧЕНИЕ

Работа выполнена по заказу ГКУЗ РХ «РМИАЦ». В ходе выпускной квалификационной работы была решена проблема автоматизации передачи данных о прикреплении населения к медицинской организации.

Таким образом, цель данной выпускной квалификационной работы сокращение временных затрат на обмен информацией путем создания дополнительного функционала для шины «Передача данных о прикреплении (единый пакет с участками)» – достигнута, поставленные задачи решены.

В разделе «Анализ предметной области»проведено: выбор средств проектных решений», проанализирована основная деятельность учреждения ГКУЗ РХ «РМИАЦ», которая позволила выяснить особенности разработки информационной системы для обмена данными о прикреплении пациента к медицинской организации между базами данных ТФОМС РХ и РИАМС «ПроМед».

В связи с Федеральным законом РФ от 27 июля 2006 года № 152-ФЗ «О персональных данных» была разработана защищенная сеть.

Для разработки автоматизированной информационной системы для обмена информацией между базами данных ТФОМС РХ и РИАМС «ПроМед» был выбран язык программирования JavaScript.

Согласно проведенному анализу СУБД создание базы данных Шины будет осуществляться с помощью MariaDB, по причине её скорости, защищённости и простоты в использовании.

В проектном разделе была построена модель по методологии потоков данных DFD,объясняющая процесс передачи данных о прикреплении пациентов к медицинской организации из РИАМС «ПроМед» в ТФОМС РХ.

Разработана автоматизированная система для передачи данных из РИАМС «ПроМед» в шину Rish, которая включает в себя создание таких функций, как:

- принятие информации из базы данных из РИАМС «ПроМед»;

- преобразование данных из формата XMLв JSON;
- функционал отправки данных о прикреплении в очереди attachment_for_deletion и attachment для вставки или удаления из БД шины;
- функционал добавления или удаления данных о прикреплении из БД шины.

Проведен расчет показателей экономической эффективности проекта. Капитальные затраты составили 69112 (руб.), и эксплуатационные затраты, которые составили 4301 (руб.), совокупная стоимость владения информационной системой (ТСО) составляет 69112 (руб.). Анализ риска реализации проекта позволяет рекомендовать проект к реализации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. BusinessStudio [Электронный ресурс]: Понятие бизнес-процесса, – Режим доступа: https://www.businessstudio.ru
2. DEVACADEMY – обучение современному программированию онлайн [Электронный ресурс]. – Режим доступа: http://devacademy.ru/
3. GeekBrains – образовательный портал [Электронный ресурс]: Git – распределенная система управления версиями, – Режим доступа: https://geekbrains.ru
4. Git [Электронный ресурс]: Git – распределенная система управления версиями, – Режим доступа: https://git-scm.com/book/ru/v1/Введение-Установка-Git.
5. JSON. Что это такое[Электронный ресурс]. – Режим доступа:https://www.json.org/json-ru
6. MariaDB в сравнении с MySQL [Электронный ресурс]. – Режим доступа: https://mariadb.com/kb/ru/mariadb-vs-mysql-features/
7. Ubuntu [Электронный ресурс]: Программная платформа Node.js, – Режим доступа: https://help.ubuntu.ru/wiki/javascript
8. Wikipedia [Электронный ресурс]: RabbitMQ, – Режим доступа https://ru.wikipedia.org/wiki/RabbitMQ
9. Wikipedia [Электронный ресурс]: VirtualBox, – Режим доступа https://ru.wikipedia.org/wiki/VirtualBox
10. Автоматизированная информационная система [Электронный ресурс]. – Режим доступа: https://dic.academic.ru/dic.nsf/ruwiki/334809
11. Все о xml формате данных[Электронный ресурс]. – Режим доступа: https://www.reviversoft.com/ru/file-extensions/xml
12. ГКУЗ РХ «Республиканский медицинский информационно-аналитический центр» [Электронный ресурс]. – Режим доступа:https://miac.mz19.ru/org/.

13. Как установить и использовать Dockerв ubuntu [Электронный ресурс]: Режим доступа: https://www.digi talocean.com/community/tutorials/docker-ubuntu-16-04-ru

14. Компания IBM [Электронный ресурс]. – Режим доступа: https://www.ibm.com/.

15. Министерство здравоохранения Республики Хакасия [Электронный ресурс]. – Режим доступа: http://mz19.ru/.

16. Национальная библиотека им. Н. Э. Баумана [Электронный ресурс]: MariaDB, – Режим доступа: https://ru.bmstu.wiki

17. Об основах охраны здоровья граждан в Российской Федерации: федер. закон Рос. Федерации от 21 ноября 2011 г. № 323-ФЗ: принят Гос. Думой 1 ноября 2011 г.: одобр. Советом Федерации 9 ноября 2011 г. // Рос. газ. – 2011. – 23 ноября.

18. Основы IDEF3 [Электронный ресурс]: Режим доступа: https://www.cfin.ru/vernikov/idef/idef3.shtml

19. Положение о министерстве здравоохранения Республики Хакасия: Постановление Правительства Республики Хакасия от 11 июня 2009 г. № 260 (в ред. от 07.03.2017 N 93) // Вестник Хакасии. – 2009. – 23 июня.

20. Просто о микросервисах [Электронный ресурс]. – Режим доступа:https://habr.com/ru/company/raiffeisenbank/blog/346380/

21. РИАМС ПроМед [Электронный ресурс]. – Режим доступа: http://swan-it.ru/elektronnoe_zdravoohranenie/riams_promed

22. Сайтостроение от А до Я [Электронный ресурс]: PHP, Ruby, Python– краткая характеристика трёх языков программирования. – Режим доступа:http://www.internet-technologies.ru

23. Территориальный Фонд обязательного медицинского страхования Республики Хакасия [Электронный ресурс]. – Режим доступа:http://tfomsrh.ru/about-us/

24. Устав государственного казенного учреждения здравоохранения Республики Хакасия "Республиканский медицинский информационно-аналитический центр": Постановление Правительства Республики Хакасия от 30 августа 2007 г. № 274 (в ред. от 26.08.2014 N 430) // Вестник Хакасии. – 2007. – 7 сентября.

25. Установка MARIADBв UBUNTU 18.04 [Электронный ресурс]: Режим доступа: https://losst.ru/ustanovka-mariadb-v-ubuntu-18-04.

26. Установка Node.jsв UBUNTU 16.04 [Электронный ресурс]: Режим доступа: https://fornex.com/help/install-nodejs-ubuntu/

27. Центр информационных технологий [Электронный ресурс]: Преимущества Linux для бизнеса. – Режим доступа: https://itvolga.com

28. Что такое DFD (диаграммы потоков данных) [Электронныйресурс]:Понятиебизнес-процесса. – Режимдоступа: https://habr.com/ru/ company/trinion/blog/340064/

29. Что такое VPN? [Электронный ресурс]. – Режим доступа:https://blog.avast.com/ru/chto-takoe-vpn-i-kak-eto-rabotaet-bazovoe-ukovodstvo-avast.

30. Язык программирования JavaScript [Электронный ресурс]. – Режим доступа: https://www.internet-technologies.ru/articles/yazyk programmirovaniya-javascript-informaciya-dlya-nachinayuschih.html.

ПРИЛОЖЕНИЕ А

Программный код файла index.js

```
const AMQP = require('amqplib')
const path = require('path')
const moment = require('moment')
const fs = require('fs')
const sharedFunctions = require('rish-shared-functions')
const SERVICE_NAME = '[rmis-rabbitmq-poller]'
const consumeFromQ = sharedFunctions.consumeFromQ
const publishToQ = sharedFunctions.publishToQ

let localRMQConnectionString,
    rmisRMQConnectionString,
    rmisQData,
    rmisQAnswer,
    localDispObservationQ,
    localAttachmentQ,
    dispObservationDeletionQ

if (process.env.NODE_ENV === 'production') {
  localRMQConnectionString = process.env.RMQ_LOCAL_CONNECTION_PRODUCTION
  rmisRMQConnectionString = process.env.RMQ_RMIS_CONNECTION_PRODUCTION
  rmisQData = process.env.RMQ_Q_RMIS_DATA_PRODUCTION
  rmisQAnswer = process.env.RMQ_Q_RMIS_ANSWER_PRODUCTION
  localDispObservationQ = process.env.RMQ_Q_LOCAL_DISP_OBSERVATION_PRODUCTION
  localAttachmentQ = process.env.RMQ_Q_LOCAL_ATTACHMENT_PRODUCTION
  localAttachmentDeletionQ =
process.env.RMQ_Q_LOCAL_ATTACHMENT_DELETION_PRODUCTION
  dispObservationDeletionQ =
process.env.RMQ_Q_LOCAL_DISP_OBSERVATION_DELETION_Q_PRODUCTION
}
else {
  localRMQConnectionString = process.env.RMQ_LOCAL_CONNECTION_PRODUCTION
  rmisRMQConnectionString = process.env.RMQ_RMIS_CONNECTION_TEST
  rmisQData = process.env.RMQ_Q_RMIS_DATA_TEST
  rmisQAnswer = process.env.RMQ_Q_RMIS_ANSWER_TEST
  localDispObservationQ = process.env.RMQ_Q_LOCAL_DISP_OBSERVATION_TEST
  localAttachmentQ = process.env.RMQ_Q_LOCAL_ATTACHMENT_TEST
  localAttachmentDeletionQ = process.env.RMQ_Q_LOCAL_ATTACHMENT_DELETION_TEST
  dispObservationDeletionQ =
process.env.RMQ_Q_LOCAL_DISP_OBSERVATION_DELETION_Q_TEST
}

const localRMQConnection = AMQP.connect(localRMQConnectionString)
const rmisRMQConnection = AMQP.connect(rmisRMQConnectionString)

CODE_MO_HACK = {
  "190007": 190006,
  "190071": 190070,
  "190072": 190070
}
```

```
const processDispObservation = (json) => {
  const result = {
    meta: {
      OPERATIONTYPE: null
    },
    data: {
      bdz_id: null,
      DISP_ID: null,
      DS: null,
      DS_DETECT: null,
      DS_DETECTTYPE: null,
      SNILS_VR: null,
      DATE_OUT: null,
      RESULT_OUT: null,
      DATES: null,
      CODE_MO: null,
      promed_pacient_id: null,
      created_at: null
    }
  }

  if (json['DISP'] && json['DISP']['HEADER']) {
    //console.log('JSON', json)
    const header = json['DISP']['HEADER'][0]
    const body = json['DISP']['BODY'] ? json['DISP']['BODY'][0] : null
    console.log('OPERATIONTYPE', header.OPERATIONTYPE[0])
    if (body) {
      if (header.OPERATIONTYPE[0] === "Insert" || header.OPERATIONTYPE[0] === "Update") {
        if (header.OPERATIONTYPE && header.OPERATIONTYPE[0]) {
          result.meta.OPERATIONTYPE = header.OPERATIONTYPE[0]
        }
        if (header.CODE_MO && header.CODE_MO[0]) {
          const code_mo = CODE_MO_HACK[header.CODE_MO[0]] ?
CODE_MO_HACK[header.CODE_MO[0]] : header.CODE_MO[0]
          result.data.CODE_MO = code_mo
        }
        if (header.DISP_ID && header.DISP_ID[0]) {
          result.data.DISP_ID = header.DISP_ID[0]
        }
        if (header.DATA && header.DATA[0]) {
          result.data.created_at = header.DATA[0]
        }
        if (body.BDZ_ID && body.BDZ_ID[0]) {
          result.data.bdz_id = body.BDZ_ID[0]
        }
        if (body.ID_PAC && body.ID_PAC[0]) {
          result.data.promed_pacient_id = body.ID_PAC[0]
        }
        if (body.DATE_IN && body.DATE_IN[0]) {
          result.data.DATE_IN = body.DATE_IN[0]
        }
        if (body.DS_DETECTTYPE && body.DS_DETECTTYPE[0]) {
          result.data.DS_DETECTTYPE = body.DS_DETECTTYPE[0]
        }
        if (body.DS_DETECT && body.DS_DETECT[0]) {
```

```
        result.data.DS_DETECT = body.DS_DETECT[0]
       }
       if (body.DS && body.DS[0]) {
        result.data.DS = body.DS[0]
       }
       if (body.SNILS_VR && body.SNILS_VR[0] && body.SNILS_VR[0] !== '00000000000') {
        result.data.SNILS_VR = body.SNILS_VR[0]
       }
       if (body.RESULT_OUT && body.RESULT_OUT[0]) {
        result.data.RESULT_OUT = body.RESULT_OUT[0]
       }
       if (body.DATES && body.DATES[0]) {
        if (body.DATES[0].PLAN_DATE) {
         result.data.DATES = body.DATES[0].PLAN_DATE.map(e => e)
        }
        else {
         console.error(`Ошибка с плановыми датами: ${JSON.stringify(body)}`)
         result.data.DATES = [ moment().format('YYYY-MM-DD') ]
        }
       }
       //console.log(localDispObservationQ, JSON.stringify(result))
      }
      else {
       const _message = `OPERATIONTYPE не найден: ${header.OPERATIONTYPE[0]}, DISP_ID:
${result.data.DISP_ID}`
       sharedFunctions.sendNotificationToDispObservationErrors(SERVICE_NAME, _message)
      }
     }
     else {
      if (header.OPERATIONTYPE[0] === 'Delete') {
       result.meta.OPERATIONTYPE = 'Delete'
       result.data.DISP_ID = header.DISP_ID[0]
      }
     }
    }
    return result
}

const processAttachment = (Json) => {

    const result = {
     meta: {
      OPERATIONTYPE: null
     },
     data: {
      BDZID: null,
      CODE_MO: null,
      INFO_TYPE: null,
      ATTACH_DT_MO: null,
      DETACH_DT_MO: null,
      DETACH_CAUSE_MO: null,
      PODR: null,
      OTD: null,
      UCH: null,
      UCH_TYPE: null,
      created_at: moment().format('YYYY-MM-DD'),
```

```
      deleted: 0,
      PERSONATTACHID: null
    }
  }
  //console.log('JSON v functii processAttachment', JSON.stringify(Json))
  if (Json['PERSONATTACHDISTRICT'] && Json['PERSONATTACHDISTRICT']['HEADER']) {
    const header = Json['PERSONATTACHDISTRICT']['HEADER'][0]
    const body = Json['PERSONATTACHDISTRICT']['BODY'] ?
Json['PERSONATTACHDISTRICT']['BODY'][0] : null
    //console.log(body);
    //console.log(header);
    //console.log('OPERATIONTYPE', header.OPERATIONTYPE[0])
    if (body) {
      if (header.OPERATIONTYPE[0] === "Insert" || header.OPERATIONTYPE[0] === "Update") {
        if (header.OPERATIONTYPE && header.OPERATIONTYPE[0]) {
          result.meta.OPERATIONTYPE = header.OPERATIONTYPE[0]
        }
        if (body.BDZID && body.BDZID[0]) {
          result.data.BDZID = body.BDZID[0]
        }
        if (header.CODE_MO && header.CODE_MO[0]) {
          const code_mo = CODE_MO_HACK[header.CODE_MO[0]] ?
CODE_MO_HACK[header.CODE_MO[0]] : header.CODE_MO[0]
          result.data.CODE_MO = code_mo
        }
        if (body.INFO_TYPE && body.INFO_TYPE[0]){
          result.data.INFO_TYPE = body.INFO_TYPE[0]
        }
        if (body.ATTACH_DT_MO && body.ATTACH_DT_MO[0]){
          result.data.ATTACH_DT_MO = body.ATTACH_DT_MO[0]
        }
        if (body.DETACH_DT_MO && body.DETACH_DT_MO[0]){
          result.data.DETACH_DT_MO = body.DETACH_DT_MO[0]
        }
        if (body.DETACH_CAUSE_MO && body.DETACH_CAUSE_MO[0]){
          result.data.DETACH_CAUSE_MO = body.DETACH_CAUSE_MO[0]
        }
        if (body.PODR && body.PODR[0]){
          result.data.PODR = body.PODR[0]
        }
        if (body.OTD && body.OTD[0]){
          result.data.OTD = body.OTD[0]
        }
        if (body.UCH && body.UCH[0]){
          result.data.UCH= body.UCH[0]
        }
        if (body.UCH_TYPE && body.UCH_TYPE[0]){
          result.data.UCH_TYPE = body.UCH_TYPE[0]
        }
        if (header.PERSONATTACHID && header.PERSONATTACHID[0]){
          result.data.PERSONATTACHID = header.PERSONATTACHID[0]
        }
      }
      else {
        const _message = `OPERATIONTYPE не найден: ${header.OPERATIONTYPE[0]},
PERSONATTACHID: ${result.data.PERSONATTACHID}`
```

```
      sharedFunctions.sendNotificationToDispObservationErrors(SERVICE_NAME, _message)
     }
    }
    else {
     if (header.OPERATIONTYPE[0] === 'Delete') {
      result.meta.OPERATIONTYPE = 'Delete'
      result.data.PERSONATTACHID = header.PERSONATTACHID[0]
     }
    }
   }
  console.log(result);
  return result
}

// слушаем очередь РМИС с данными
console.log(`Слушаем ${rmisQData}, NODE_ENV ${process.env.NODE_ENV}`)
consumeFromQ(rmisRMQConnection, rmisQData, async (ch, rawMsg) => {
 try {
    const content = rawMsg.content.toString()
    const parsedResult = await sharedFunctions.parseXML(content)
    console.log(parsedResult);
    if (parsedResult[1]) {
     //console.log("json posle poluchenya iz ocheredi",JSON.stringify(parsedResult[1])) // kopai tyt ))
     const type = Object.keys(parsedResult[1])[0]
     if (type === "DISP") {
      const result = processDispObservation(parsedResult[1])
      console.log(`результат после обработки: ${JSON.stringify(result)}`)
      if (result.meta.OPERATIONTYPE === 'Insert' || result.meta.OPERATIONTYPE === 'Update') {
       publishToQ(localRMQConnection, localDispObservationQ, JSON.stringify(result), (err, ok) => {
        //console.log(err, ok)
        if (!err) {
         ch.ack(rawMsg)
        }
       })
      }
      else if (result.meta.OPERATIONTYPE === 'Delete') {
       console.log(`DISPS на удаление`)
publishToQ(localRMQConnection, dispObservationDeletionQ, result.data.PERSONATTACHID, (err,
ok) => {
        if (!err) {
         ch.ack(rawMsg)
        }
       })
      }
      else {
       console.log(`Ошибка...`)
      }
     }
     else if (type === "PERSONATTACHDISTRICT") {
      const result = processAttachment(parsedResult[1])
      const PERSONATTACHID = result.data.PERSONATTACHID ? true : false
      if (PERSONATTACHID && (result.meta.OPERATIONTYPE === 'Insert' ||
result.meta.OPERATIONTYPE === 'Update')) {
       publishToQ(localRMQConnection, localAttachmentQ, JSON.stringify(result), (err, ok) => {
        if (!err) {
         ch.ack(rawMsg)
```

```
          }
         })
        }
        else if (PERSONATTACHID && result.meta.OPERATIONTYPE === 'Delete') {
         publishToQ(localRMQConnection, localAttachmentDeletionQ, result.data.PERSONATTACHID,
(err, ok) => {
          if (!err) {
           ch.ack(rawMsg)
}
         })
        }
      }
else {
console.log(`Неизвестный тип сообщения - отмечаем как обработанное...`)
ch.ack(rawMsg)
console.error(`Неизвестный тип: ${type}`)
     }
   }
else {
console.error(`Ошибка при парсинге XML: ${parsedResult[0]}, данные: ${content}`)
process.exit(1)
   }
  } catch (e) {
   const _message = `ошибка ${e.message}`
   await sharedFunctions.sendNotificationToDispObservationErrors(SERVICE_NAME, _message)
   process.exit(1)
  }
})
console.log(`Слушаем ${rmisQAnswer}, NODE_ENV ${process.env.NODE_ENV}`)
// слушаем очередь РМИС с ответами на обновление данных
consumeFromQ(rmisRMQConnection, rmisQAnswer, (ch, rawMsg) => {
 console.log(rawMsg.content.toString())
})
.catch(e => console.warn(e))
```

ПРИЛОЖЕНИЕ Б

Программный код файла xmlGeneration.js

```
var amountXml; // amount of xml queries for xmlGeneration() function
var minRandom; // min for getRandomInt function
var maxRandom; // max for getRandomInt function
var fixedNumber; // The fixed length for getRandomFixedInt function

//The getRandomInt() function calculates a random number between min and max
function getRandomInt(minRandom, maxRandom){
   var rand = minRandom - 0.5 + Math.random() * (maxRandom - minRandom + 1);
   rand = Math.round(rand);
   return rand;
}
// The getRandomFixedInt() function calculates a random fixed(var fixedNumber) number.
//Example 1 is not fixed, but 0001 is fixed 4-digit number
function getRandomFixedInt(fixedNumber){
   var result = "";
   for (j=0;j<fixedNumber;j++){
      result+=getRandomInt(0,9);
   }

   return result;
}

//The getFIO function generates random FIO. Example of an result variable "result =
<FAM>Васильев</FAM><IM>Василий</IM><OT>Сергеевич</OT>"
function getFIO(){
   var result;

   //The massives is random massives of Imën Familiy Otchestv
varmasOfFiNames = ["Василий", "Юрий", "Аркадий", "Иван", "Петр", "Максим",
"Леонид", "Михаил", "Алексей", "Сергей"];//Imena
varmasOfSeNames = ["Васиьев", "Юриьев", "Аркадьев", "Иванов", "Петров", "Максимов",
"Леонидов", "Михаилов", "Алексеев", "Сергеев"];//Familii
varmasOfThNames = ["Васиьевич", "Юриьевич", "Аркадьевич", "Иванович", "Петрович",
"Максимович", "Леонидович", "Михаилович", "Алексеевич", "Сергеевич"];//Otchestvo

result = `<FAM>${masOfSeNames[getRandomInt(0,masOfFiNames.length-1)]}</FAM>`;//this string
adds First name at result variable in xml format
   result += `<IM>${masOfFiNames[getRandomInt(0,masOfSeNames.length-1)]}</IM>`;//this string
adds Second name at result variable in xml format
   result += `<OT>${masOfThNames[getRandomInt(0,masOfThNames.length-1)]}</OT>`;//this string
adds Third name at result variable in xml format
   return result;
}

//The getMoCode() function generates random MO code of availble mo codes
function getMoCode(){
   //The masCodeMO variable is massive of availble MOs
   var masCodeMo = ["190048", "190007", "190085", "190006", "190033", "190036", "190032",
"190037", "190038", "190008", "190001", "190013", "190004", "190002", "190003", "190009",
```

```
"190010", "190022", "190087", "190020", "190070", "190071", "190072", "190047", "190039",
"190040", "190030", "190086", "190025", "190028", "190031", "190045", "190113"];
   var result = masCodeMo[getRandomInt(0,masCodeMo.length-1)];
   return result;
}

//The getBithDate() function generates today's date and records in yyyy-mm-dd format
function getDate(){
   var result;

   result = new Date();
   result =`${result.getFullYear()}-${result.getMonth()}-${result.getDay()}`;

   return result;
}

//The xmlGeneration function generates X xml queries where X is amountXml variable
function xmlGeneration(amountXml){

   var numberOfMo;
   var numberOfPersonattachid;
   var seFeTh;

   for (i=0;i<amountXml;i++)
   {
      var xmlOutput = `<?xml version="1.0" encoding="utf-
8"?>\n<PERSONATTACH>\n<HEADER>\n<OPERATIONTYPE>Insert</OPERATIONTYPE>`;//
XML query variable
      numberOfMo = getMoCode();//A number of MO
      numberOfPersonattachid = getRandomInt(1, 99999999);//A number of operation ID
      numberOfTomId = getRandomInt(1,999999);//A number of a human's TomskDatabaseID
      seFeTh = getFIO();//A random Second First and Third name
      gender = getRandomInt(1,2);// A random gender of 2 available. But all we know that geneders more
then 2:)
      birthDay = getDate();// A pseudo birthday of a human
      docType = getRandomInt(1,20);// A random Document Type
      docSer = `${getRandomFixedInt(2)} ${getRandomFixedInt(2)}`;// A random passport series
      docNum = getRandomFixedInt(6);// A random passport number
      attachType = getRandomInt(1,2);// случайныйспособприкрепления
      attachDateMo = getDate();// A date of attachement
subdivison = `1`;// подразделение основного участка
division = getRandomFixedInt(3);// отделосновного
uch = getRandomInt(0,9);//A random code of a main therapeutic
site(основнойтерапевтическийучасток)
attachDate = getDate();// дата прикрепления к медработнику основного участкак

xmlOutput +=`\n<CODE_MO>${numberOfMo}</CODE_MO>`;// this string adds <CODE_MO> to
XML query.
      xmlOutput
+=`\n<PERSONATTACHID>${numberOfPersonattachid}</PERSONATTACHID>\n</HEADER>`;//
this string adds <PERSONATTACHID> to XML query
      xmlOutput +=`\n<BODY><BDZID>${numberOfTomId}</BDZID>`;//etc
      xmlOutput += getFIO();
      xmlOutput +=`\n<W>${gender}</W>`;
      xmlOutput +=`\n<DR>${birthDay}</DR>`;
      xmlOutput +=`\n<DOCTYPE>${docType}</DOCTYPE>`;
```

```
        xmlOutput +=`\n<DOCSER>${docSer}</DOCSER>`;
        xmlOutput +=`\n<DOCNUM>${docNum}</DOCNUM>`;
        xmlOutput +=`\n<ATTACH_TYPE>${attachType}</ATTACH_TYPE>`;
        xmlOutput +=`\n<ATTACH_DT_MO>${attachDateMo}</ATTACH_DT_MO>`;
        xmlOutput +=`\n<PODR>${subdivison}</PODR>`;
        xmlOutput +=`\n<OTD>${division}</OTD>`;
        xmlOutput +=`\n<UCH>${uch}</UCH>`;
        xmlOutput
+=`\n<ATTACH_DT>${attachDate}</ATTACH_DT>\n</BODY>\n</PERSONATTACH>`;

return xmlOutput;
    }

}
module.exports = () => {
return xmlGeneration(1);
}
//console.log(xmlGeneration(1));
//console.log(getRandomFixedInt(8));
//console.log(getBirthDate());




var parseString = require("xml2js").parseString;

parseString(xml, function(error, result){
    if(error){
        console.log("ERROR: " + error);
        return;
    }
    var kek = JSON.stringify(result);
    console.dir(kek);
   });

delay(2000);
console.log('Rish: Успешно...');

//console.log(xml());
```

ПРИЛОЖЕНИЕ В

Программный код файла services/attachment-to-rish/index.js

```
const AMQP = require('amqplib')
const path = require('path')
const fs = require('fs')
const sharedFunctions = require('rish-shared-functions')
const consumeFromQ = sharedFunctions.consumeFromQ
const publishToQ = sharedFunctions.publishToQ
const config = JSON.parse(fs.readFileSync('config.json').toString())
const SERVICE_NAME = '[attachment-observation-to-rish]'

let localRMQConnectionString,
   localAttachmentQ,
   localAttachmentToTfomsQ,
   attachmentDeletionQ

if (process.env.NODE_ENV == 'production') {
   localRMQConnectionString =
process.env.process.env.RMQ_LOCAL_CONNECTION_PRODUCTION
   localAttachmentQ = process.env.RMQ_Q_LOCAL_ATTACHMENT_PRODUCTION
   attachmentDeletionQ =
process.env.RMQ_Q_LOCAL_ATTACHMENT_DELETION_PRODUCTION
   localAttachmentToTfomsQ =
process.env.RMQ_Q_LOCAL_ATTACHMENT_TO_TFOMS_PRODUCTION
}
else{
   console.log('test');
   localRMQConnectionString = process.env.RMQ_LOCAL_CONNECTION_PRODUCTION
   localAttachmentQ = process.env.RMQ_Q_LOCAL_ATTACHMENT_TEST
   attachmentDeletionQ = process.env.RMQ_Q_LOCAL_ATTACHMENT_DELETION_TEST
   localAttachmentToTfomsQ = process.env.RMQ_Q_LOCAL_ATTACHMENT_TO_TFOMS_TEST
}

console.log(`Инициализациязакончена. Слушаемочереди: ${localAttachmentQ},
${attachmentDeletionQ}`);
//console.log(process.env);
const localRMQConnection = AMQP.connect(localRMQConnectionString)
// очередь с добавлением данных о прикреплении
consumeFromQ(localRMQConnection, localAttachmentQ, async (ch, rawMsg) => {
   const content = rawMsg.content.toString()
   //console.log(content)
   const parsedContent = JSON.parse(content)
   console.log('parsed-content', parsedContent)
   const existence = await sharedFunctions.sendToMariadbProxy({ type: 'attachement', action:
'is_bdz_id_and_personattachid_exists?', data: { bdz_id: parsedContent.data.BDZID, personattachid:
parsedContent.data.PERSONATTACHID } })
   if (existence.personattachid) {
      console.log(`PERSONATTACHID: ${parsedContent.data.PERSONATTACHID} ужеестьвбазе,
обновление...`)
      const updateResult = await sharedFunctions.sendToMariadbProxy({ type: "attachement", action:
'Update', data:  parsedContent.data })
```

```
      console.log('updateResult', updateResult)
      if (!updateResult[0]) {
        publishToQ(localRMQConnection, localAttachmentToTfomsQ,
parsedContent.data.PERSONATTACHID, async (err, ok) => {
          if (!err) {
            ch.ack(rawMsg)
            console.log(`PERSONATTACHID успешнообновлено
${parsedContent.data.PERSONATTACHID}`)
          }
          else {
            const _message = `[attachment-observation-to-rish]
ошибкапридобавлениивочередьдляотправкивТФОМС PERSONATTACHID
${parsedContent.data.PERSONATTACHID}`
            await sharedFunctions.sendNotificationToDispObservationErrors(SERVICE_NAME, _message)
            process.exit(1)
          }
        })
      }
      else {
        const _message = `[attachment-observation-to-rish]
ошибкаприобновлениивочередьдляотправкивТФОМС PERSONATTACHID
${parsedContent.data.PERSONATTACHID}, ${JSON.stringify(updateResult[0])}`
        await sharedFunctions.sendNotificationToDispObservationErrors(SERVICE_NAME, _message)
        process.exit(1)
      }
    }

    else if (!existence.personattachid) {
      const attachInsertResult = await sharedFunctions.sendToMariadbProxy({ type: 'attachement', action:
'Insert',
data: parsedContent.data
      })
      if (attachInsertResult[0]) {
        await sharedFunctions.sendNotificationToDispObservationErrors(SERVICE_NAME,
`${attachInsertResult[0]} PERSONATTACHID ${parsedContent.data.PERSONATTACHID}`)
        process.exit(1)
      }
      else if (attachInsertResult[1]) {
        publishToQ(localRMQConnection, localAttachmentToTfomsQ,
parsedContent.data.PERSONATTACHID, async (err, ok) => {
          if (!err) {
            ch.ack(rawMsg)
            console.log(`PERSONATTACHID в внутренним ID ${attachInsertResult[1]}
(PERSONATTACHID: ${parsedContent.data.PERSONATTACHID}) передано в очередь на передачу
в ТФОМС`)
          }
          else {
            const _message = `Ошибка при добавлении в очередь PERSONATTACHID
${parsedContent.data.PERSONATTACHID}`
            await sharedFunctions.sendNotificationToDispObservationErrors(SERVICE_NAME, _message)
            process.exit(1)
          }
        })
}
      else {
        process.exit(1)
```

```
      }
    }
    else{console.log(`nothing`)}
})

consumeFromQ(localRMQConnection, attachmentDeletionQ, async (ch, rawMsg) => {
  const personAttachId = parseInt(rawMsg.content.toString())
  console.log(`Полученинособщениеобудалении PERSONATTACHID: ${personAttachId}`)
const result = await sharedFunctions.sendToMariadbProxy({ type: "attachement", action: "Delete", data:
{ PERSONATTACHID: personAttachId }})
  console.log(`Получен ответ об удалении PERSONATTACHID: ${personAttachId}:
${JSON.stringify(result)}`)
  if (result[1] && parseInt(result[1])) {
    publishToQ(localRMQConnection, localAttachmentToTfomsQ, result[1].toString(), (err, ok) => {
      if (!err) {
        ch.ack(rawMsg)
}
else {
const _message = `Ошибка при отправке ид (${result[1]}) на пометку как удаленный в очередь для
тфомс ${err}`
sharedFunctions.sendNotificationToDispObservationErrors(SERVICE_NAME, _message)
      }
    })
  }
  else {
    const _message = `ошибка при пометке как удаленный: ${JSON.stringify(result[0])}
(PERSONATTACHID: ${personAttachId})`
    sharedFunctions.sendNotificationToDispObservationErrors(SERVICE_NAME, _message)
  }
})
```

ПРИЛОЖЕНИЕ Г

Программный код файла services/rish-mariadb-proxy/index.js

```
const fs = require('fs')
const isDevelopmentEnv = fs.existsSync("services")
const path = require('path')
const config = JSON.parse(fs.readFileSync("config.json", 'utf8'))
const mysql = require('mysql2')
const net = require('net')
const JsonSocket = require('json-socket')
const sharedFunctions = require('rish-shared-functions')

const pool = mysql.createPool({
  "user": process.env.RISH_MARIADB_PROXY_USER,
  "password": process.env.RISH_MARIADB_PROXY_PASSWORD,
  "database": process.env.RISH_MARIADB_PROXY_DB,
  "connectionLimit": process.env.RISH_MARIADB_PROXY_CONNECTION_LIMIT,
  "timezone": process.env.RISH_MARIADB_PROXY_TZ,
  "dateStrings": true,
  "host": process.env.RISH_MARIADB_HOST
})


// console.log(pool);

const mariadbProxy = net.createServer()
const attachementActions = require(path.join(process.cwd(), isDevelopmentEnv ? "services/rish-mariadb-proxy/actions/attachement" : "actions/attachement"))
const dispObservationActions = require(path.join(process.cwd(), isDevelopmentEnv ? "services/rish-mariadb-proxy/actions/dispObservation" : "actions/dispObservation"))
const mdbActions = require(path.join(process.cwd(), isDevelopmentEnv ? "services/rish-mariadb-proxy/actions/mdb" : "actions/mdb"))
const periodicalsActions = require(path.join(process.cwd(), isDevelopmentEnv ? "services/rish-mariadb-proxy/actions/periodicals" : "actions/periodicals"))
const personDataActions = require(path.join(process.cwd(), isDevelopmentEnv ? "services/rish-mariadb-proxy/actions/personData" : "actions/personData"))
const systemActions = require(path.join(process.cwd(), isDevelopmentEnv ? "services/rish-mariadb-proxy/actions/system" : "actions/system"))
const statsActions = require(path.join(process.cwd(), isDevelopmentEnv ? "services/rish-mariadb-proxy/actions/stats" : "actions/stats"))
const dictionariesActions = require(path.join(process.cwd(), isDevelopmentEnv ? "services/rish-mariadb-proxy/actions/dictionaries" : "actions/dictionaries"))
const usersActions = require(path.join(process.cwd(), isDevelopmentEnv ? "services/rish-mariadb-proxy/actions/users" : "actions/users"))

const TABLES_THAT_ALLOW_INSERTION = Object.keys(config.services["mdb-parser"].columnMatchings).map(k => config.services["mdb-parser"].columnMatchings[k].appendable ? config.services["mdb-parser"].columnMatchings[k].table : null ).filter(e => e !== null)
TABLES_THAT_ALLOW_INSERTION.push("mdbFiles")

const promisedQuery = (query) => {
  return new Promise((resolve, reject) => {
```

```
    pool.getConnection((err, conn) => {
      conn.query(query, (err, results, fields) => {
       if (err) {
         pool.releaseConnection(conn)
         console.error(`Erorr in query: ${query} \n ${err}`)
         resolve([`Erorr in query: ${query} \n ${err}`])
       } else {
         pool.releaseConnection(conn)
         resolve([null, results ])
       }
      })
    })
  })
}

// квотирование строки, если нужно
const normalizeValuesForSql = (array) => {
  return array.map(e => {
    if (typeof(e) === 'string' && e === "NULL") {
      return e
    }
    else if (typeof(e) === 'string' && (!e.startsWith("'") && !e.endsWith("'"))) {
      return `'${e}'`
    }
    else {
      return e
    }
  })
}

const SMOCODES = {}

console.log(`ЗагружаетCMOкодывпамять...`)
promisedQuery(`SELECT code, name FROM SMOCODES`).then((data) => {
  data[1].forEach((e) => {
    SMOCODES[e['name']] = e['code']
})
console.log(`CMO коды загружены.`)
mariadbProxy.listen(process.env.RISH_MARIADB_PROXY_PORT)
  //console.log(SMOCODES);
  mariadbProxy.on('listening', () => console.log('Сервис mariadb-proxy запущен.'))
})
.catch(e => console.warn(e))

const formatFields = (person) => {
  //console.log('formatting person', person);
  const formattedPerson = {}
  const keys = Object.keys(person)
  for (let i = 0; i < keys.length; i++) {
    if (person[keys[i]] instanceof Date) {
      formattedPerson[keys[i]] = sharedFunctions.formatDate(person[keys[i]])
    }
    else if (person[keys[i]] === null) {
      formattedPerson[keys[i]] = 'NULL'
    }
    else {
```

```
      formattedPerson[keys[i]] = person[keys[i]]
    }
  }
  return formattedPerson
}

const diffPeoples = (rawOldPerson, newPerson) => {
  const diffObject = { }
  const oldPerson = formatFields(rawOldPerson)
  const keys = Object.keys(oldPerson)
  console.log('formatted oldperson', oldPerson)
  for (let i = 0; i < keys.length; i++) {
    if(oldPerson[keys[i]] !== newPerson[keys[i]]) {
      diffObject[keys[i]] = newPerson[keys[i]]
    }
  }
  return diffObject
}

const insertPeriodics = async (peopleId,mdbFileId,changes,initial,ENP,socket) => {
  console.log(`Вставка периодик для ${ENP}`)
const insertInPeriodicals = `INSERT INTO periodicals(people_id,mdbFile_id,changes,initial) VALUES
(${peopleId},${mdbFileId},'${JSON.stringify(changes)}',${initial})`
  const insertInPeriodicalsResult = (await promisedQuery(insertInPeriodicals))[1]
  if (insertInPeriodicalsResult && insertInPeriodicalsResult.insertId) {
    console.log(`Измененияздля ${ENP}`)
socket.sendEndMessage([null])
  }
  else {
    console.error(`Ошибкапривставкепериодикдля ${ENP}`)
socket.sendEndMessage([`Ошибка при вставке периодик для ${ENP}`])
}
}

mariadbProxy.on('error', (e) => console.error(e))
mariadbProxy.on('connection', async (socket) => {
  socket = new JsonSocket(socket, { delimeter: process.env.JSON_SOCKET_DELIMETER })
  socket.on('message', async (message) => {
    console.log(`Принято: ${JSON.stringify(message)}`)
    let opts = { socket, message, pool, promisedQuery, formatFields, normalizeValuesForSql }
    if (message.action === "insert" && TABLES_THAT_ALLOW_INSERTION.includes(message.type)
&& message.data) {
      socket.sendEndMessage(await promisedQuery(`INSERT INTO ${message.type}
(${Object.keys(message.data).join(",")})
VALUES(${normalizeValuesForSql(Object.values(message.data)).join(",")})`))
    }
    else if (message.type === 'attachement') {
      attachementActions(opts)
    }
    else if (message.type === 'dispObservation') {
      dispObservationActions(opts)
    }
    else if (message.type === 'mdb') {
      mdbActions(opts)
    }
    else if (message.type === 'periodicals') {
```

```
    periodicalsActions(opts)
   }
   else if (message.type === 'personData') {
    personDataActions(opts)
   }
   else if (message.type === 'system') {
    systemActions(opts)
   }
   else if (message.type === 'stats') {
    statsActions(opts)
   }
   else if (message.type === 'dict') {
    dictionariesActions(opts)
   }
   else if (message.type === 'users') {
    usersActions(opts)
   }
   else if (message.action === 'get_if_not_then_create' &&
TABLES_THAT_ALLOW_INSERTION.includes(message.type) && message.data) {
    const getQuery = `SELECT id FROM ${message.type} WHERE ${Object.keys(message.data)[0]} =
${normalizeValuesForSql(Object.values(message.data))[0]}`
    //console.log(getQuery)
    const getResult = await promisedQuery(getQuery)
    //console.log(getResult);
    console.log(message.action, "after get", message.data, getResult[1], getQuery)
    if (getResult[1].length === 0) {
     const insertQuery = `INSERT INTO ${message.type} (${Object.keys(message.data)[0]}) VALUES
(${normalizeValuesForSql(Object.values(message.data))[0]})`
     //console.log("INSERT QUERY ------ ", insertQuery)
     const insertResult = await promisedQuery(insertQuery)
     //console.log("insertResult ++++++++++++++", insertResult)
     console.log(message.action, 'after insert', message.data, 'insertResult', insertResult, getQuery)
     socket.sendEndMessage([null, insertResult[1].insertId])
    }
    else {
     //console.log([null, getResult[1][0].id]);
     socket.sendEndMessage([null, getResult[1][0].id])
    }
   }
   else if (message.action === "get_code_from_name" && message.type === 'SMOCODES' &&
message.data) {
    console.log(message.data, SMOCODES);
    socket.sendEndMessage([null, SMOCODES[message.data]])
   }
   else {
    socket.sendEndMessage([`Не найден ни один маршрут по вашему запросу:
${JSON.stringify(message)}`])
   }
  })
})
```

Программныйкодфайлаservices/rish-mariadb-proxy/action/attachment.js:

```
const path = require('path')
```

```
const fs = require('fs')
const sharedFunctions = require('rish-shared-functions')

const SERVICE_NAME = '[rish-mariadb-proxy/attachment]'

const attachment_to_tfoms_fields =
[
  "UPDATED_AT",
  "CREATED_AT",
  "DELETED",
  "CODE_MO",
  "INFO_TYPE",
  "BDZID",
  "ATTACH_DT_MO",
  "DETACH_DT_MO",
  "DETACH_CAUSE_MO",
  "PODR",
  "OTD",
  "UCH",
  "UCH_TYPE",
  "PERSONATTACHID"
]

const attachement = async (opts) => {

  console.log('attachement-mariadb-proxy принято сообщение: ' + JSON.stringify(opts.message));
  if (opts.message.action === 'Insert' && opts.message.data) {
    const cleanedFromNull = formatForInsertion(opts.message.data, false)
    console.log('cleanedFromNull ->', cleanedFromNull, typeof(cleanedFromNull.DATES))
    const insertQresult = await opts.promisedQuery(`INSERT INTO ATTACHMENTS
(${Object.keys(cleanedFromNull)}) VALUES
(${sharedFunctions.normalizeValuesForSql(Object.values(cleanedFromNull))})`)
    if (insertQresult[0]) {
      console.error(`Ошибка при вставке: ${JSON.stringify(opts.message.data)}`)
      opts.socket.sendEndMessage(["Ошибка при вставке"])
    }
    else {
      console.log(`Успешно вставлено, id в ATTACHEMENT: ${insertQresult[1].insertId}`)
      opts.socket.sendEndMessage([null, insertQresult[1].insertId])
    }
  }
  else if (opts.message.action === 'Update' && opts.message.data) {
    console.log(opts.message.data)
    const PERSONATTACHID = opts.message.data.PERSONATTACHID
    delete opts.message.data.PERSONATTACHID
    const formatted = formatForInsertion(opts.message.data)
    console.log('formatted', formatted);
    const updateFieldsString = Object.keys(formatted).map(k => `${k} =
${sharedFunctions.normalizeValueForSql(formatted[k])}`).join(", ")
    console.log(", updateFieldsString)
    const updateResult = await opts.promisedQuery(`UPDATE ATTACHMENTS SET
${updateFieldsString} WHERE PERSONATTACHID = ${PERSONATTACHID}`)
    const setTfomsStatus = await opts.promisedQuery(`UPDATE ATTACHMENTS SET
SEND_TO_TFOMS_STATUS_id = 0 WHERE PERSONATTACHID = ${PERSONATTACHID}`)
    if (updateResult[0]) {
      opts.socket.sendEndMessage([updateResult])
```

```
    }
    else {
      opts.socket.sendEndMessage([null])
    }
  }
  else if (opts.message.action === 'Delete' && opts.message.data) {
    const result = await opts.promisedQuery(`UPDATE ATTACHMETNS SET RESULT_OUT = 5
WHERE PERSONATTACHID = ${opts.message.data.PERSONATTACHID}`)
    const setTfomsStatus = await opts.promisedQuery(`UPDATE ATTACHMETNS SET
SEND_TO_TFOMS_STATUS_id = 0 WHERE PERSONATTACHID =
${opts.message.data.PERSONATTACHID}`)
    if (result[0]) {
      console.error(`Ошибка при отметке, как удаленный PERSONATTACHID
${opts.message.data.PERSONATTACHID}`)
      opts.socket.sendEndMessage(result[0])
    }
    else {
      const deleteResult = await opts.promisedQuery(`SELECT id FROM ATTACHMETNS WHERE
PERSONATTACHID = ${opts.message.data.PERSONATTACHID}`)
      console.log('deleteResult', deleteResult)
      if (deleteResult[0]) {
        const _message = `ошибка (${JSON.stringify(deleteResult[0])}) при удалении
PERSONATTACHID ${opts.message.data.PERSONATTACHID}`
        await sharedFunctions.sendNotificationToDispObservationErrors(SERVICE_NAME, _message)
        opts.socket.sendEndMessage([ deleteResult[0] ])
      }
      else {
        opts.socket.sendEndMessage([ null, opts.message.data.DISP_ID ])
      }
    }
    // console.log(`Результат удаления: ${JSON.stringify(result)}`)
    // opts.socket.sendEndMessage(result)
  }
  else if (opts.message.action === 'is_bdz_id_and_personattachid_exists?' && opts.message.data) {
    const result = {
      bdz_id: false,
      personattachid: false
    }
    console.log(`opts.message.data.BDZID = ${opts.message.data.bdz_id}
opts.message.data.PERSONATTACHID = ${opts.message.data.personattachid}`);
    bdzIdQResult = await opts.promisedQuery(`SELECT id FROM peoples WHERE id =
${opts.message.data.bdz_id} LIMIT 1`)
    personattachidQResult = await opts.promisedQuery(`SELECT id FROM ATTACHMENTS WHERE
PERSONATTACHID = ${opts.message.data.personattachid} LIMIT 1`)
    if (bdzIdQResult[1].length !== 0) {
      result.bdz_id = true
    }
    if (personattachidQResult[1].length !== 0) {
      result.personattachid = true
    }
    opts.socket.sendEndMessage(result)
  }
  else{console.log(`nothing`)}
}

module.exports = attachement
```

Выпускная квалификационная работа выполнена мной самостоятельно. Использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

Отпечатано в одном экземпляре.

Библиография 30 наименований.

Один экземпляр сдан на кафедру.

«___» июня 2019 г.
(дата)

_____ Барских И.Н.
(подпись) (ФИО)

ПРИЛОЖЕНИЕД

Таблица Д.1– Список рисков

№ проекта	Группы рисков	Перечень рисков проекта	Уровень влияния риска на проект	Вероятность риска	Возможность предотвращения или снижения риска
1.	Риски, связанные с разработкой проекта	Организационный риск	Средний	Низкий	Обратная связь между организациями, проработка договора
2.		Лимитированное время разработки	Высокий	Средний	Тщательное продумывание плана перед разработкой
3.		Неучтенные дополнительные расходы	Средний	Средняя	Дополнительное соглашение к договору
4.	Риски, связанные с внедрением проекта	Реализационный риск	Низкий	Низкий	Дополнительное соглашение к договору
5.		Дополнительные расходы на доработку проекта	Низкий	Высокая	Регулярные тесты

ПРИЛОЖЕНИЕ Е

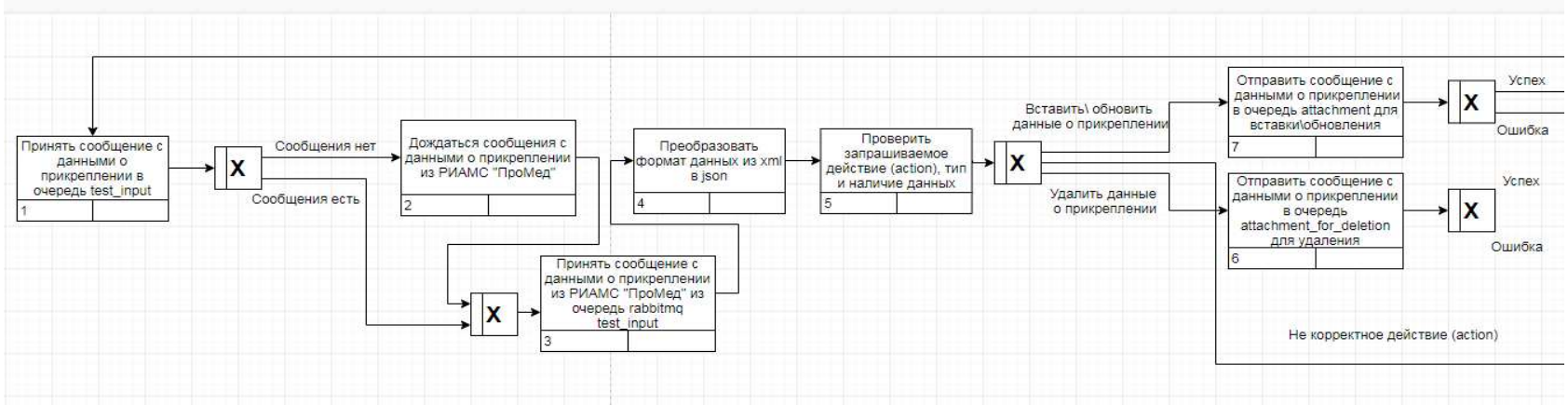

Рисунок 2.4– IDEF 3 диаграмма работы разрабатываемого функционала шины, лист 1

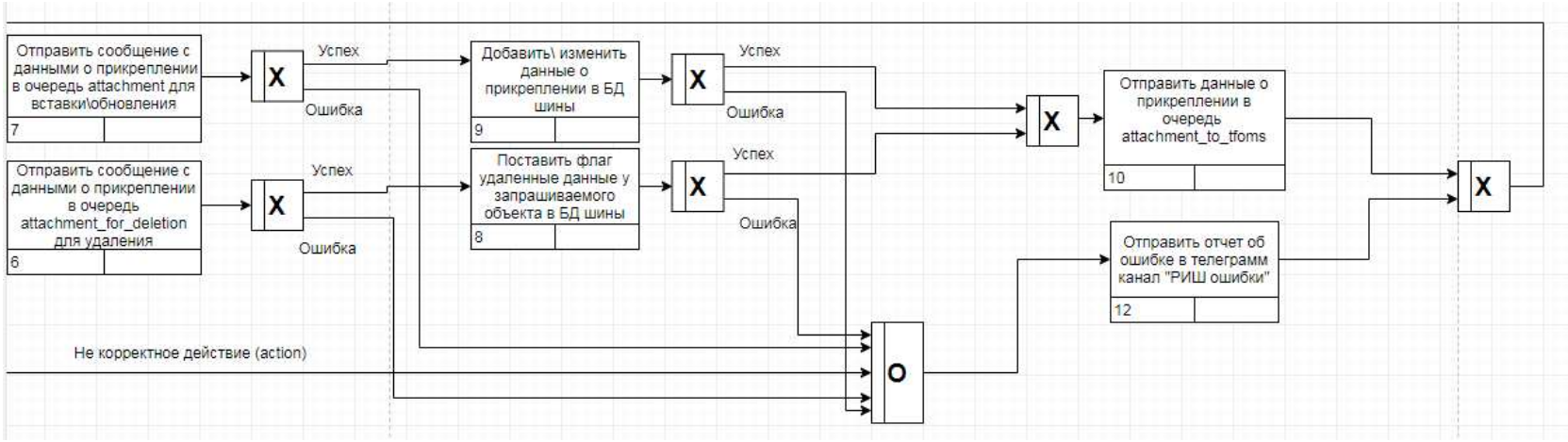

Рисунок 2.4, лист 2

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Хакасский технический институт – филиал ФГАОУ ВО
«Сибирский федеральный университет»
институт

Прикладная информатика, математика и естественнонаучные дисциплины
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ Е.Н. Скуратенко
подпись инициалы, фамилия
« 21 » 06 2019г.

БАКАЛАВРСКАЯ РАБОТА

09.03.03 – Прикладная информатика
код – наименование направления

Автоматизация передачи сведений о прикреплении населения к медицинским организациям из РИАМС «ПроМед» в ИС ТФОМС РХ
тема

Руководитель _____ 21.06.19 зав. кафедрой, доцент, к.т.н. Е.Н.Скуратенко
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ 21.06.19 И.Н. Барских
подпись, дата инициалы, фамилия

Консультанты по разделам:

Экономический _____ 21.06.19 Е.Н. Скуратенко
наименование раздела подпись, дата инициалы, фамилия

Нормоконтролер _____ 21.06.19 В.И. Кокова
подпись, дата инициалы, фамилия

Абакан 2019