ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ И ЦИФРОВЫХ ТЕХНОЛОГИЙ
КАФЕДРА ИНФОРМАЦИОННО-ТЕЛЕКОММУНИКАЦИОННЫХ СИСТЕМ И ТЕХНОЛОГИЙ

РАЗРАБОТКА ТЕХНОЛОГИИ МОНИТОРИНГА ОШИБОК В КАНАЛАХ СВЯЗИ В РАСПРЕДЕЛЁННЫХ БАЗАХ ДАННЫХ

Выпускная квалификационная работа
обучающегося по направлению подготовки 11.04.02 «Инфокоммуникационные технологии и системы связи»
очной формы обучения, группы 12001736
Демченко Василия Джумбериевича

Научный руководитель
доцент, канд. техн. наук,
доцент кафедры
Информационно-
телекоммуникационных
систем и технологий НИУ
«БелГУ» Прохоренко Е.И.

Рецензент
ведущий инженер
департамента разработки
региональных систем ООО
"СофТраст" Чернова Н.А.

БЕЛГОРОД 2019

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время информационные системы могут включать в себя невероятное количество конечных узлов и занимают важное место в нашей жизни. Множество обыденных и привычных вещей, окружающее человека в современном мире, на первый взгляд могут показаться не такими уж и сложными, тем не менее, множество примеров показывает совершенно обратное. Для примера, можно взять светофор. Что представляет из себя это небольшое изобретение? На первый взгляд - ничего сложного, но стоит только задуматься, о настоящей роли светофора в жизни современного города, как станет понятно, что это частица одного большого механизма, по управлению дорожным потоком в огромном мегаполисе, что работоспособность каждого отдельного светофора обеспечивает поддержание транспортной системы всего города. Невозможно представить себе современный город с неуправляемым транспортным потоком внутри, а парализация городского трафика, хотя бы на день, влечёт за собой огромные убытки для всей инфраструктуры города.

В любой целостной системе важен каждый её отдельно взятый компонент, отказоустойчивость системы – один из самых важных критериев при разработке.

Множество факторов постоянно оказывают влияние на работоспособность системы, в следствии чего, мы неизбежно подходим к вопросу о своевременном получение информации о выходе того или иного узла связи.

На данный момент средства тестирования и мониторинга представляют собой отдельные элементы, по тестированию тех или иных узлов, которые часто имеют зависимости друг от друга и не дают полной картины того, что действительно происходит в системе. В связи с огромным набором различных средств разработки и внедрения программного обеспечения, встаёт вопрос об универсальности такого средства. Как пример, рассмотрим стандартные проблемы, которые могут возникнуть при подключении к базе данных от одной машины к другой.

Проблемы с доступностью машин, связанные с неправильной конфигурацией сети (на уровне транспортного протокола, на уровне

конфигурирования подсетей)

1. Проблемы с неактуальной структурой базы данных
2. Проблемы с набором учётных записей для установки соединения с базой данных (пара логин/пароль)
3. Проблемы с неправильным указанием экземпляра базы данных, при установке соединения

Это только малая часть проблем, с которыми встречаются сотрудники, обслуживающие большую распределённую систему, которая требует постоянного мониторинга отдельных её частей.

Целью данной научно-исследовательской работы является разработка технологии мониторинга ошибок в каналах связи в распределённых базах данных.

Основной задачей работы является создание эффективного механизма мониторинга состояния баз данных, отражающего текущее состояние отдельных узлов связи, до конечного момента установления соединения с базой данных и получения информации из неё.

Объект исследования — мониторинг состояния баз данных распределённых на отдельных серверах.

Предмет исследования — создание эффективного инструмента тестирования состояния базы данных и отдельных узлов сети, участвующих в процессе установления соединения с конечной базой данных.

Методы исследования — разработка алгоритма тестирования состояния канала связи и базы данных, а так же размещение средств тестирования на:

1. UNIX-системах (посредством Rancher+Docker),
2. WIN-системах(посредством IIS)

Данная исследовательская работа состоит из двух основных частей: теоретической и практической.

В теоретической части научно-исследовательской работы анализируется возможность создания технологии для обозначенного предмета исследования, а так же обзор возможностей фрэймворка .aspNET по разработке алгоритма для

тестирования баз данных. Обзор основных компонентов системы, обзор методов реализации алгоритмов.

В практической части ведется разработка вышеуказанного технологии, а так же создание инфраструктуры (веб приложения), для получения результатов работы алгоритма и возможности её внедрения с минимальными системными требованиям.

Практическая значимость результатов исследования заключается в экономии ресурсов для мониторинга инфраструктуры целостной системы, использующей множество подключений к распределённым базам данных.

1 ПОСТАНОВКА ЗАДАЧИ ИССЛЕДОВАНИЯ

1.1 Анализ проблемы

При анализе эффективности работы отдела технической поддержки и сопровождения в ООО «Софтраст», были выявлены следующие особенности: первоначальное подключение региона для работы в системе занимает продолжительное время и требует участия множества людей, каждый из которых выполняет свои функции, после момента внедрения, программного комплекса, начинается следующий цикл – сопровождение, который требует внимательного отношения к состоянию программного продукта в рамках региона.

Для примера можно рассмотреть стандартную схему настройки региона, на примере Орловской области. Существует центральная площадка, на которой расположены технический средства для обеспечения работы системы. Так же, там располагается центр обработки данных (ЦОД), состоящий из нескольких баз данных, в которые попадают сведения от каждой отдельной базы данных региона.

Обмен данными между сервера лечебно-профилактических учреждений (ЛПУ) и ЦОД происходит внутри закрытого канала связи, обеспеченного VIPNet координаторами, в соответствии с рисунком 1:

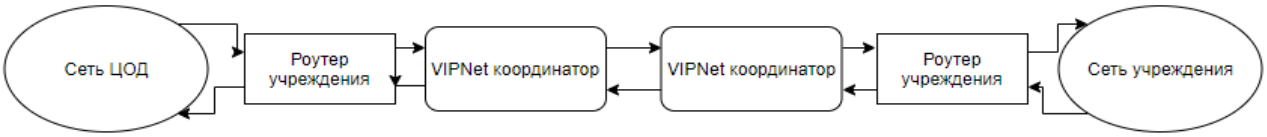

Рисунок 1- Обмен данных между ЦОД и ЛПУ

Не во всех операциях, производимых в ЛПУ, требуется связь с ЦОД, учреждение может жить автономно, но такое состояние ЛПУ считается аварийным и требует вмешательства специалистов для устранения проблем связи. Канал связи от конечного пользователя в ЛПУ до ЦОД состоит из множества отдельных узлов, после первоначальной настройки которых наступает момент их обслуживания и поддержания работоспособности системы.

Зачастую, о факте возникновения проблемы мы узнаём от конечного пользователя системы, посредством обращения в отдел технической поддержки

с банальным вопросом о том, что не работают те или иные функции. Разбор таких заявок иногда может занимать до нескольких дней, а бывает даже недель.

Как пример, можно привести одну из самых нетривиальных задач, которая была решена нашим отделом.

После отключения света ночью, канал связи в ЛПУ переключился с основного на резервный, после чего появились ошибки в некоторых ресурсоёмких (с точки зрения размера передаваемой информации) запросах в базу данных учреждения с площадки ЦОД. Были поставлены задачи в отдел сопровождения и внедрения, по возникающим ошибкам в программном обеспечении. На основе этих задач, были поставлены задачи разработчикам по проверке программного обеспечения в режиме отладки. Оповещены системные администраторы ЛПУ а так же ЦОД. Совместными усилиями множества специалистов было выявлено, что поменялся размер MTU пакета, из-за смены провайдера на резервном канале связи. Отсюда получили, что большие пакеты данных не смогли сегментироваться и если запрос в базу данных отдавал большое количество информации, он «падал» с ошибкой.

На ряду с такими задачами, возникают конечно и стандартные случаи, такие как смена пароля для доступа к удалённой базе данных, ограничение прав, смена IP адреса, редактирование справочников, внесение изменений в структуру БД. Что тоже является частью задачи по мониторингу канала связи, так как может косвенно или напрямую влиять на недоступность учреждения при работе с программным комплексом.

В следствии того, что программный комплекс имеет очень сложную структуру баз данных, компоненты которой могут изменяться, незначительное редактирование тех или иных схем БД может привести к отказу того или иного модуля системы. Только пользовательских таблиц в системе более 2.5 тысяч.

Тем самым, цель создания нашей системы — это сокращение времени на проведение регламентных работ по мониторингу ошибок в каналах связи, а также реализация возможностей дополнительного мониторинга, включающего такие вещи как персональные запросы в базу данных, для проверки структуры

таблиц.

Анализ существующих решений по комплексному мониторингу показывает, что есть все возможности для реализации таких систем на предприятии. В основном это связанно с тем, что нет эталонных решений при выборе средств разработки децентрализованных систем. Есть общие рекомендации к подходу и проектированию системы и инструменты для её создания.

Эти инструменты зачастую могут выполнять схожие функции, более того, они скорее всего будут похожи по производительности и даже будут иметь одинаковый набор выходных данных, при определённых входных. Тем не менее, средства мониторинга работоспособности отдельных частей системы, созданных с помощью таких инструментов, могут быть попросту несовместимы, из-за использования одного из несовместимых компонентов.

Часто, системы мониторинга имеют вполне интуитивно понятный интерфейс, который позволяет произвести анализ для одного отдельного экземпляра. Примером таких систем для SQL Server может являть встроенная система мониторинга Activity Monitor.

Есть системы мониторинга, позволяющие проверять сетевую доступность машин, например Zabbix. Такие системы имеют возможность обратной связи с машинами, находящимися в системе мониторинга. То есть имеют возможность выполнять простые действия для устранения неполадок, например запускать такие процессы как перезагрузка удалённой машины или завершение каких-либо процессов.

При разработке системы будет учтён опыт использования средств мониторинга, выполняющих схожие функции.

1.2 Примеры реализации подобных систем

В настоящее время существует множество различных систем мониторинга, это связано с тем, что существует множество различных инструментов для их создания, а также задачами, которыми продиктовано обоснование этих инструментов.

Пример систем мониторинга:

1. Cacti
2. Ganglia
3. Collectd
4. Graphite
5. Nagios
6. Zabbix
7. Netdata
8. Prometheus

Все они выполняют схожие по своему функционалу функции, отдельно хотелось бы выделить Zabbix.

Zabbix это профессиональная система мониторинга, использующаяся довольно часто. Есть практически всё, включая уведомления на почту. Но Zabbix довольно требовательная система, которая зачастую требует отдельный сервер под свою настройку [28].

Что бы не разбирать каждую систему мониторинга отдельно, достаточно обозначить некоторые правила. Система мониторинга должна:

- потреблять минимум ресурсов
- не значительно нагружать сетевой трафик
- уметь работать как с windows так и linux
- поддерживать работу с SQL server
- не иметь внешних зависимостей
- желательно, быть недорогой или иметь «open source» лицензию
- быть гибко настраиваемой под конкретные задачи

Как видим, список требований обширный, но хотелось бы остановится на некоторых из них поподробнее. В связи с тем, что работа мониторинга проводится внутри закрытой сети и система мониторинга будет иметь доступ к хранилищам с персональными данными, она не может иметь компонентов, которые могут пытаться передавать какие-то данные во внешнюю сеть. По тем же самым причинам нет возможности использовать какие то платные сервисы на

подобии Paessler. Эта система мониторинга имеет хороший функционал, но имеет закрытый код и большую стоимость.

Учитывая всю специфику задачу которые возложены на требования к системе мониторинга, выбрать из существующих систем подходящую под все требования просто не является возможным.

Так же хотелось бы отметить, что сторонние средства мониторинга не используют своих алгоритмов для тестирования нагрузки на SQL экземпляр, а лишь позволяют обращаться к системным таблицам и их записям, которые генерирует SQL экземпляр на основе своей работы. То есть общение системы мониторинга состоит из запросов в системные таблицы SQL server, которые предоставляют необходимую информацию.

Такие встроенные инструменты работы с SQL как Activity Monitor предоставляют хороший набор параметров для оценки работоспособности сервера. Использование Activity Monitor и SQL Server Profiler всегда может предаставить нам полные сведения о проблемах с SQL сервер, если мы говорим о таких параметрах как ресурсоёмкие запросы, блокировки, проблемы записи. Поэтому в системы будет учтён опыт схожих систем и будут реализованы алгоритмы на основе тех данных которые может предоставлять проверяемый компонент.

В алгоритмах системы мониторинга используются проверки доступности SQL экземпляра исходя из опыта работы с SQL server.

Задача создания системы мониторинга появилась как раз из-за невозможности увидеть неработающий по тем или иным причинам сервер. Наличие отклика на ping или открытого порта не говорит о работоспособности сервиса, а только лишь означает, что от места запроса до места назначения есть связь. Поэтому для проверки работоспособности используется другая схема проверок. Эти проверки формируются на основе опыта отдела сопровождения и оформляются в скрипты. Такие скрипты могут включать в себя проверки как настроек сети, конкретных портов так и время выполнения заданий на SQL сервере или в планировщике заданий WINDOWS.

Можно сказать, что каждая проверка, заложенная в систему, это набор правил и алгоритмов, проверяющих ту или иную часть сервера или БД. Система позволяет получить доступ к CMD в режиме администратора и производить любые действия по мониторингу сервера. Система позволяет динамически расширять модель проверок без серьёзных доработок внутри исходного кода системы. В дальнейшем планируется возможность внесения новых проверок в виде SQL скриптов без необходимости редактирования исходного кода приложения [5].

1.3 Исходные данные для реализации системы мониторинга

В качестве исходных данных для проведения исследования используется реальная инфраструктура программного комплекса, действующая в основе системы под названием – «ТМ МИС SaaS», разработанная компанией «СофТраст».

Система включает в себя комплекс программного обеспечения для перевода всех действий работников лечебно-профилактических учреждений в электронный вид. От записи пациента на приём через электронные сервисы, до выписки больничного.

Для хранения данных в системе используются несколько видов баз данных. Основная система хранения данных – SQL Server различных версий (2008-2016 год). Все экземпляры SQL Server размещены на WIN-системах.

Каждый отдельный регион располагает набором отдельных баз данных на каждое учреждение, а также одной центральной площадкой (ЦОД), в которую доставляется информация от баз клиентов (ЛПУ) [18].

Средний размер баз данных учреждений сильно отличается от вида лечебно-профилактического учреждения. Так, поликлиникам и больницам в небольших городах может хватать 10-20 гигабайт, в то время как центральные районные больницы (ЦРБ), могут занимать до 700 гигабайт данных.

Центральное хранилище включает в себя необходимую информацию, собранную от ЛПУ, и может занимать размер от терабайта и выше, в зависимости от количества ЛПУ входящих в область [6].

Каждая отдельная база данных имеет ежедневные задания по созданию резервных копий, а также множество других различных механизмов, для поддержания работоспособности системы.

Эти механизмы чаще всего представлены SQL заданиями, которые имеют своё расписание, а также планировщиком заданий WINDOWS.

Передача данных между ЦОД и ЛПУ происходит закрытой сети, использующей VIPnet координаторы.

Так как система активно развивается под потребности заказчика, она непременно изменяется, что влечёт за собой изменения структуры баз данных. В связи с этим, возникает вопрос в мониторинге множества мелких заданий, которые были сформированы для решений определённого рода уникальных задач. Создание инструмента по мониторингу состояния каналов связи и баз данных позволит сэкономить большое количество временных ресурсов.

Из описания, представленного выше, описания программного комплекса сразу можно сделать несколько замечаний, которые не будут лишними. Хранение заданий на отдельных серверах каждого учреждения, а так же хранение заданий планировщика WINDOWS на каждой отдельной машине накладывает определённые ограничение, а именно невозможность мониторинга за всеми отдельно созданными экземплярами, невозможность их централизованного редактирования.

1.4 Задачи исследования

Задача исследования заключается в исследованиях возможности создания комплексной системы мониторинга по состоянию канала связи до БД, возможности сбора интересующей информации на стороне сервера БД. Исследования возможности оперативного представления информации по состоянию региона в режиме реального времени.

Рассмотрение возможностей повышение безопасности, путём комплексного отслеживания обращений к таблицам с персональной информацией, повышение безопасности за счёт проверок состояния сервера и отдельных его компонентов в режиме реального времени. Исследование

возможности повышения отказоустойчивости сервера, за счёт мониторинга блокировок БД а так же возможных ограничениях БД со стороны сервера на котором она установлена (например мониторинг дискового пространства) [15].

В рамках введения системы мониторинга в продуктивную среду, провести оценочное тестирование системы и пробный запуск. В ходе тестирования, собрать информацию из результатов теста и сделать выводы об эффективности системы с точки зрения полноты информации и производительности [16].

После проведения оценочного тестирования, провести запуск в продуктивной среде. Для этого скомпилировать проект под размещение в службах IIS и разместить на одном из серверов Орловской области.

Предоставить необходимую информацию в пользовательский интерфейс для удобного отображения, которая позволит оценить состояние регионального канала связи, а так же всех узлов входящих в него.

1.5 Выбор средств разработки

Выбор средств разработки продиктован условиями среды, в которых работает программное обеспечение «ТМ МИС SaaS», тем не менее в реализации системы мониторинга будут использованы только актуальные средства разработки. Это позволит использовать один и тот же проект системы для размещения и полной его совместимости.

Обоснование выбора инструментов разработки изначально было продиктовано условиями существующего программного обеспечения, над которым должен проводится мониторинг. Сервера ЛПУ представлены машинами под управлением WINDOWS. Система хранения информации – базы данных на основе SQL Server.

На входе в учреждение стоит VipNet координатор, внутренняя сеть учреждения может сильно различаться, какое-то стандартного оборудования здесь нет, но в основном это один сервер, на котором установлена база данных и подсистема приложений (IIS).

Как мы видим, в работе программного комплекса «ТМ МИС SaaS», представлены в основном продукты Microsoft. Само ядро программы

реализовано на C# и C++ языках программирования.

Средства мониторинга было решено разрабатывать в фреймворке Visual Studio, используя язык программирования C#, это позволит нам использовать современное средство разработки, при этом не уходить от семейства продуктов, уже используемых в программном комплексе. Теоретически, такой подход позволит нам избежать проблем с совместимостью и гарантирует лучшее взаимодействие компонентов системы друг с другом [2].

Современные возможности языка C# поистине впечатляют. В ходе исследования похожих задач было решено использовать платформу разработки веб-приложений - Asp.net, паттерн разработки – MVC (Model-View-Controller). Работа с базой данных реализована с использованием библиотек – Npoco. Выбор этих инструментов разработки позволит нам сделать средство асинхронного опроса баз данных, которые будет собирать необходимую информацию в достаточно оперативные сроки, при этом затраты ресурсов, на разворачивание средств мониторинга будут минимальны[3,7].

Благодаря выбору платформы Asp.net версии 2.1, мы получаем современный проект, который возможно установить на любые современные средства развёртки приложений. Такой подход упрощает и унифицирует разработку приложения, потому как сборку проекта из одних и тех же файлов можно развернуть для IIS, Windows Service, Microsoft Azure или Docker, Rancher, Kubernetes, в зависимости от того, на машине с какой операционной системой будут развёрнуты сервисы [9].

Предоставление результата о работе систем комплексного мониторинга было решено перенести в веб платформу, это позволит предоставить оперативный доступ к имеющейся информации, решит проблему многопользовательского взаимодействия с файлом отчёта, а также позволит получить отчёт из системы, не требуя каких-либо специализированных средств. Для просмотра сведений отчёта подойдёт любой WEB браузер.

Для представления собранных данных было решено использовать такой инструмент как Grafana. Это средство для представления удобного интерфейса

по отображению данных. Grafana имеет ряд гибких настроек, которые полностью удовлетворяют нашим требованиям [10].

Исходя из выше сказанного, был определён следующий набор инструментов:

1. Фреймворк – Visual Studio 2017
2. Язык программирования - C#
3. Платформа разработки веб-приложений - Asp.net
4. Паттерн – MVC (Model-View-Controller)
5. Работа с БД через – Npoco
6. База данных - SQL Server
7. Размещение системы:
   7.1. *UNIX (Docker, Rancher, Kubernetes)
   7.2. WINDOWS (IIS, Windows Service, Microsoft Azure)
8. Вывод результата:
   8.1. в файл на диск
   8.2. в сервис, предоставляющий информацию по определённому порту (обращение возможно из любого браузера или специализированных средств наподобие Postman)
   8.3. Grafana (инструмент для создания панелей с метриками, получаемыми от сервиса)

2 ОПИСАНИЕ КОМПОНЕНТОВ И МЕТОДОВ РАЗРАБОТКИ, ИСПОЛЬЗОВАННЫХ ПРИ РЕАЛИЗАЦИИ СИСТЕМЫ КОМПЛЕКСНОГО МОНИТОРИНГА КАНАЛА СВЯЗИ

2.1 Определение основных компонентов системы

При выборе средств разработки системы комплексного мониторинга, первостепенной задачей стояла оптимизация программного кода так, чтобы не вызывать блокировок ресурсов баз данных, задержек в канале связи.

После первичной инициализации приложения, происходит подключение к серверам ЦОД, где мы получаем системную информацию, необходимую для создания строк подключения и набора инструкций для проверки конечных БД.

Приложение работает в фоновом режиме и делает опрос набора строк подключений по заданному промежутку времени.

Сбор средств о состоянии конечного сервера происходит в три этапа:

1. Проверка канала связи от серверов ЦОД до БД учреждения
2. Проверка конфигурации и состояния базы данных, сбор информации о сервере
3. Проверка канала связи от конечной БД до серверов ЦОД

После получения информации от сервера ЛПУ, система обрабатывает полученные сведения, занося необходимую информацию в модель ЛПУ. После этого формируется список моделей ЛПУ, которые представляет из себя сводную информацию по каждому из опрошенных серверов. Обращение к API допускает возможность представления общего сведения состояния региона, вывод количества доступных/недоступных серверов, предоставления сведения об ЛПУ, по запросу из общего списка используя уникальный код медицинской организации из регионального справочника МО [24,25].

2.2 Методики проверки состояния канала связи

После подробного изучения поставленных задач перед реализуемой системой, был определён достаточный набор функционала, для создания полноценной системы мониторинга. Этот список включает в себя следующие элементы.

1. Проверка канала связи от серверов ЦОД до БД учреждения
   1.1. Проверка корректности внесения строки подключения
   1.2. Проверка сетевой доступности сервера баз данных
   1.3. Проверка учётных данных для подключения к БД.
2. Проверка конфигурации базы данных и сбор данных о сервере
   2.1. Проверка конфигурации безопасности SQL сервера
   2.2. Проверка необходимых прав на доступ к БД под предоставленной учётной записью
   2.3. Проверка целостности схемы таблиц
   2.4. Проверка корректности внесения справочной информации
   2.5. Проверка состояния сервера (блокировки, время чтения/записи)
   2.6. Проверка ежедневных заданий по обслуживанию сервера
   2.7. Проверка свободного места на жёстком диске (с учётом ограничения SQL Express)
   2.8. Проверка минимальных системных требований, для гарантированного взаимодействия с серверами ЦОД. (SQL Server 2008 R2, Windows Server 2008 SP2)
3. Проверка канала связи от конечной БД до серверов ЦОД
   3.1. Проверка ошибок транспортного уровня (размер MTU, сегментирование сетевых пакетов)
4. Представление данных на выходе в виде Json или XML файла
   4.1. Grafana для предоставления удобного интерфейса общения с отчётными формами

Поставленных задач достаточно, что бы оценить эффективность работы системы мониторинга и получить первичные сведения по результатам проверок. Данных результатов будет достаточно, что бы оценить состояние канала связи в регионе, состояние баз данных, работоспособность сервисов системы.

Предполагаемый набор функционала, реализованный в текущей версии системы мониторинга, может дорабатываться и изменятся. Можно отключать некоторые проверки, можно добавлять новые алгоритмы, можно изменять

старые. Данный функционал был заложен при программировании систем. В данный момент реализован только внутри программного кода приложения, но в дальнейшем планируется вынести данный функционал для просмотра в интерфейс, с последующей возможностью редактирования [20,21].

На основании полученных данных можно будет сформировать список тех ресурсов, которым следует получить уведомление о состоянии того или иного счётчика. Данный функционал не реализован, но по необходимости он может быть реализован. Например рассылка email уведомлений или публикация «alert» сообщений в канал мессенджера, например Telegram. Пример такой реализации представлена на рисунке 2:

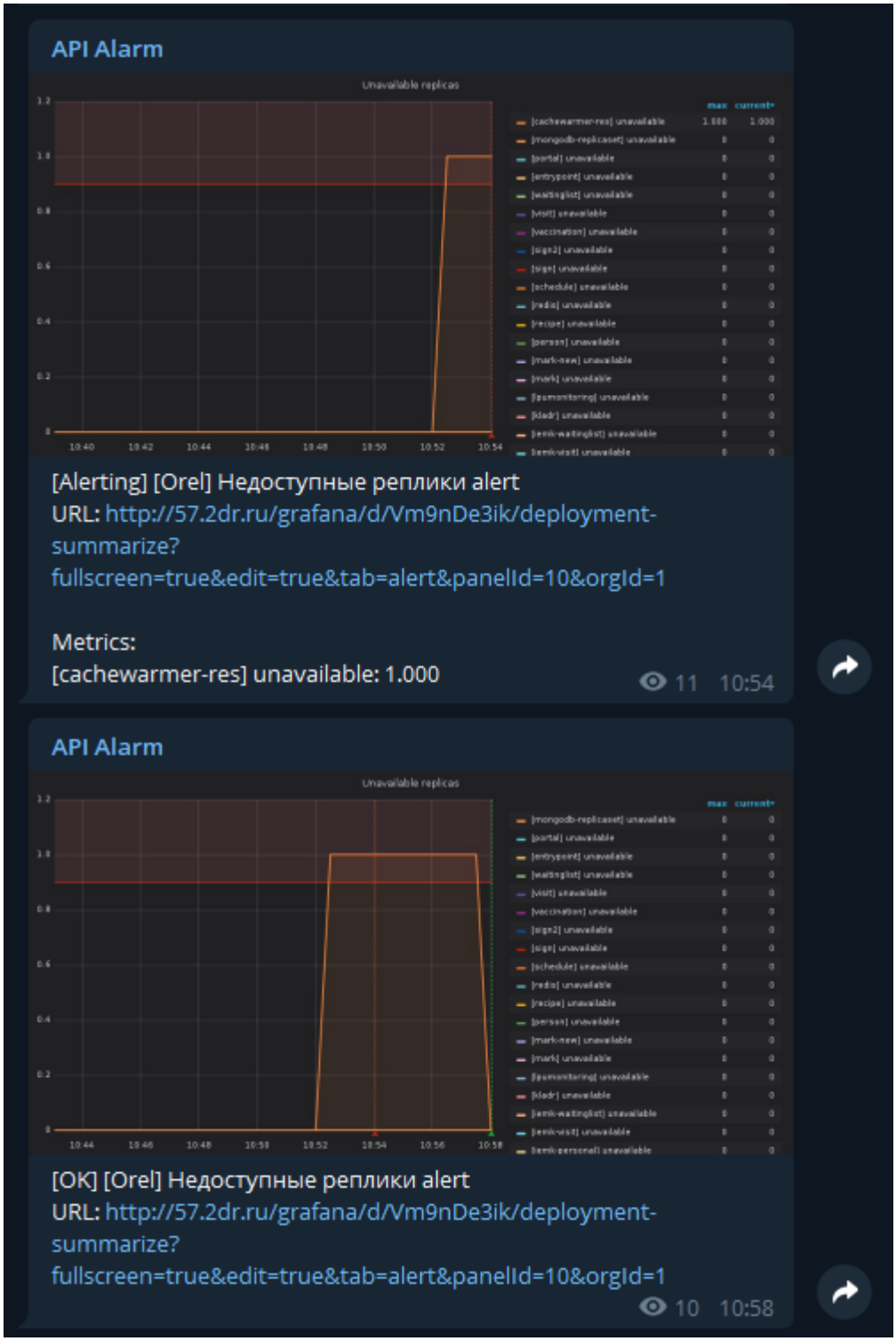

Рисунок 2 - Пример реализации «alert» сообщений в Telegram с использованием Grafana

На скриншоте мы видим первое сообщение о недоступности одного из сервисов, на следующем сообщении видим что проблема не была критичной и сервису удалось самому с ней справиться, вмешательство и фиксация инцидента не требуется.

2.3 Алгоритмы, реализуемые в системе

В системных библиотеках используемых инструментов разработки есть множество возможностей по осуществлению подключения к базам данных. Прежде чем прийти к выбору тех алгоритмов, которые работают в системе, были рассмотрены ещё несколько вариантов.

Так, например, первая версия опроса строк подключения занимала примерно 5 минут. Запрос к базам данных выполнялся последовательно, запросы становились в очередь и ожидали завершения друг друга.

В такой ситуации, на систему в целом, негативно влияли сервера, у которых были проблемы с сетью. А именно задержки в подключении или недоступность по таймауту соединения.

Более того, мониторинг должен опережать вопросы пользователей, поступающих в службу СТП, но описанный выше механизм позволял бы делать запрос с периодичностью не быстрее чем раз в 5 минут. А при сетевых задержках в канале связи, мы могли бы получить полный сбой системы, в итоге только добавив бы проблем сетевому каналу [17].

Поиск решения привёл к гораздо более эффективному методу решения данного вопроса. Таким образом, следующая версия приложения получила асинхронные методы опроса баз данных. Преимущество перед последовательными запросами было очевидно, уже после первых тестов приложения, удалось сократить время опроса всех серверов баз данных до 20-30 секунд.

Тестирование проводилось в абсолютно одинаковых условиях и на одинаковом оборудовании, в продуктивной среде Московской и Орловской области.

Всё тестирование проводилось на компьютере, основные интересующие нас характеристики которого:

- Intel Core i5-6500 (3,20 GHz - 3,60 GHz)
- 16 гигабайт оперативной памяти

Так как степень параллелизма при выполнении асинхронных задач зависит

от количества потоков процессора, а их обработка от свободной оперативной памяти, мы видим, что результаты тестов даже не на серверном оборудовании впечатляют.

Методы, используемые в приложении, максимально быстро проводят опрос и сбор нужной информации о состоянии базы данных, затем соединение закрывается, и обработка результата продолжается уже внутри системы, тем самым сводя время сессии с базой данных к минимальному.

Процесс опроса различных серверов баз данных никак не связан друг с другом. Результаты нагрузочного тестирования системы показали, что основное время тратится на открытие и закрытие соединения с базой данных, что говорит о том, что если бы нам хватало ресурсов открыть число параллельных задач равное числу опрашиваемых баз данных, то мы бы получали почти мгновенный результат по опросу целого региона и предоставлению информации о состоянии всех распределённых баз данных в режиме минимальной задержки, буквально – «онлайн».

Такого результата удалось добиться благодаря использованию встроенного языка запросов (LINQ), который был впервые представлен в .NET Framework 3.5. А если говорить точнее, то Parallel LINQ (PLINQ) являющегося параллельной реализацией шаблона LINQ. Запросы LINQ поддерживают отложенное выполнение, т. е. выполнение только по завершении перечисления запроса. Основное различие состоит в том, что PLINQ пытается задействовать сразу все процессоры в системе. Для этого он разбивает источник данных на сегменты, а затем запрашивается каждый сегмент в отдельном рабочем потоке сразу, используя сразу несколько процессоров. Во многих случаях параллельное выполнение значительно сокращает время выполнения запроса [11.26].

Благодаря параллельному выполнению PLINQ позволяет существенно повысить производительность некоторых видов запросов по сравнению с обычными методами запросов. Тем не менее параллелизм может представлять свои собственные сложности, и не все операции запросов в PLINQ выполняются быстрее. Некоторые запросы при применении параллелизма только

замедляются. В связи с этим необходимо понимать, как влияют на параллельные запросы такие аспекты, как упорядочение [23].

Как пример оптимизации кода с помощью PLINQ можно привести использование оператора «ForALL» вместо «ForEach». Сам оператор «ForEach» параллельно не выполняется, а значит результаты всех параллельных задач необходимо снова объединить с тем потоком, в котором выполняется цикл. В алгоритмах системы, удалось отказаться от оператор «ForEach», так как нам не нужно сохранить окончательный порядок результатов запроса, и последовательную обработку результата [12]. Ускорить выполнение запроса в ситуации, когда сохранение порядка не требуется и обработка результатов допускает параллелизацию, можно с помощью оператора «ForALL». «ForALL» не выполняет этот заключительный шаг слияния, разница в работе двух методов представлена на рисунке 3 [22]:

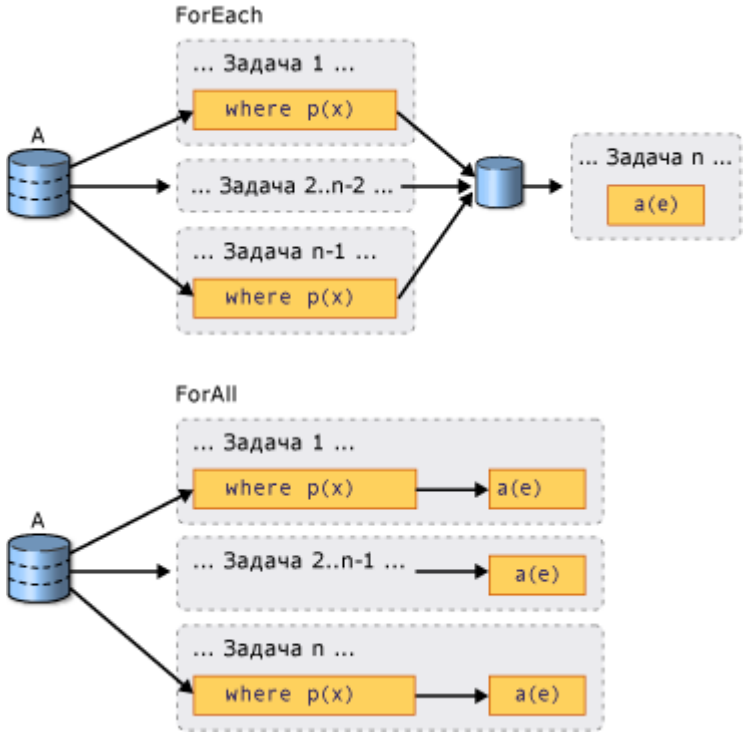

Рисунок 3 - Графическая разница методов работы «ForEach» и «ForALL»

Минусом такого подхода является более сложная структура при разработке методов, а также более внимательное отношения к деталям и методам реализации запросов, но это вполне компенсируется производительностью [14,27].

3 ПРАКТИЧЕСКИЕ РЕЗУЛЬТАТЫ ТЕСТОВ В «ПРОДУКТИВНОЙ» СРЕДЕ

3.1 Описание тестовой среды

Для проведения нагрузочного тестирования в условиях продуктивной среды была выбрана площадка Орловской области.

Площадка Орловской области имеет следующую структуру, показанную на рисунке 4, условно разделённую на две области:

– Площадка ЦОД

– Площадка каждой, отдельно взятой ЛПУ

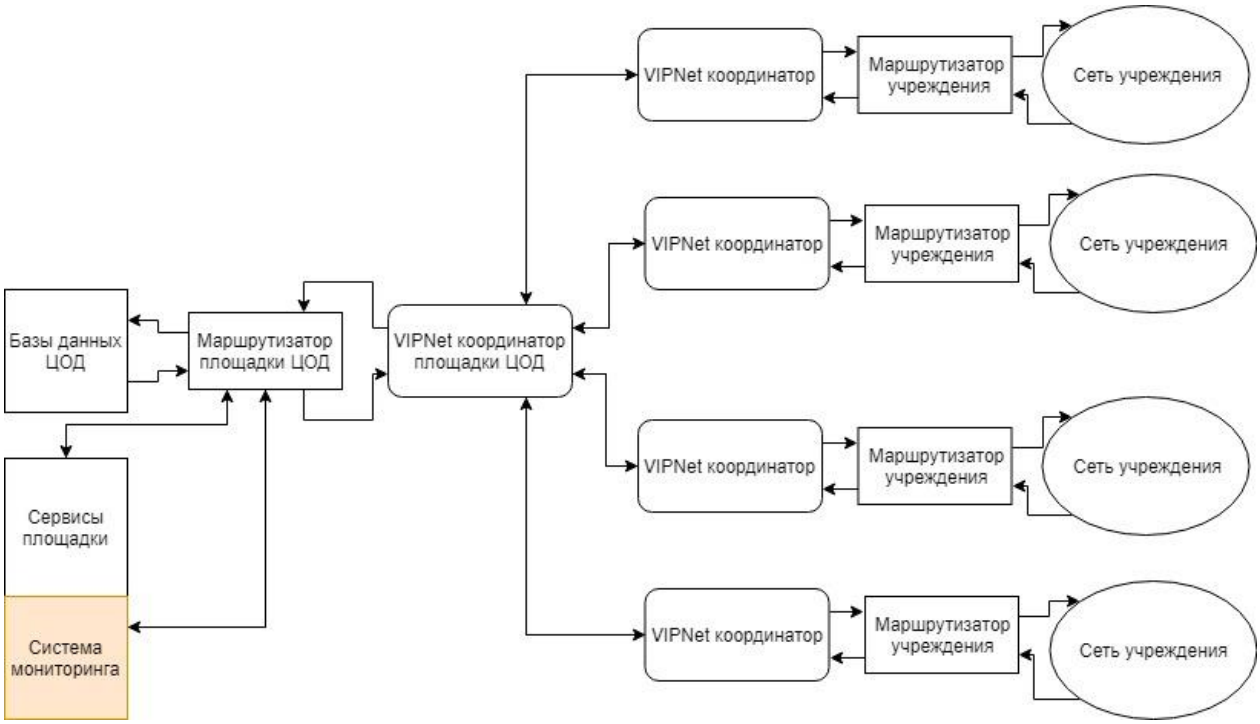

Рисунок 4 - Площадка ЦОД представлена двумя физическими машинами.

Первая - Сервер СУБД ЦОД, включает в себя: региональные и федеральные справочники, региональная ЭМК, региональный справочник пациентов (МИП), регламентные данные для формирования отчетности и прочее [4]. Физические характеристики машины:

- Intel Xeon E5645 2.4 GHz (2 процессора)
- HDD 2 TB (2 накопителя)
- RAM 24 Gb

Вторая - Сервер приложений (сервисов) региональных подсистем, включает в себя: справочник ЛПУ, сервис централизованной авторизации, сервис медзаписей, сервис интеграции региональных систем (UA, сервис поиска данных), сервис формирования медвыписок и прочее. На этой машине будет расположен наш сервис мониторинга. Физические характеристики машины:

- Intel Xeon E5645 2.4 GHz (2 процессора)
- HDD 2 TB
- RAM 24 Gb

Каждый отдельный сервер ЛПУ это физическая машина, с установленной операционной системой Windows Server не ниже 2008 версии, релиза SP2. А также установленным SQL Server не ниже 2008 версии сборки R2.

Метод авторизации каждого конкретного SQL экземпляра – «Проверка подлинности SQL Server».

Орловская область представлена сорока пятью учреждениями (Приложение А, таблица – «Список учреждений Орловской области»). Обмен данных происходит в закрытой сети передачи данных (ЗСПД), организованной посредством VipNet координаторов, установленных перед входом в сеть каждого учреждения и площадки ЦОД [1].

3.2 Размещение сервиса внутри сервера приложений (сервисов) региональных подсистем Орловской области

Размещаем приложение в диспетчере служб IIS. После создания сайта, указываем свободный порт, для нас это «1881», а так же физический путь до файлов приложения, рисунок 5:

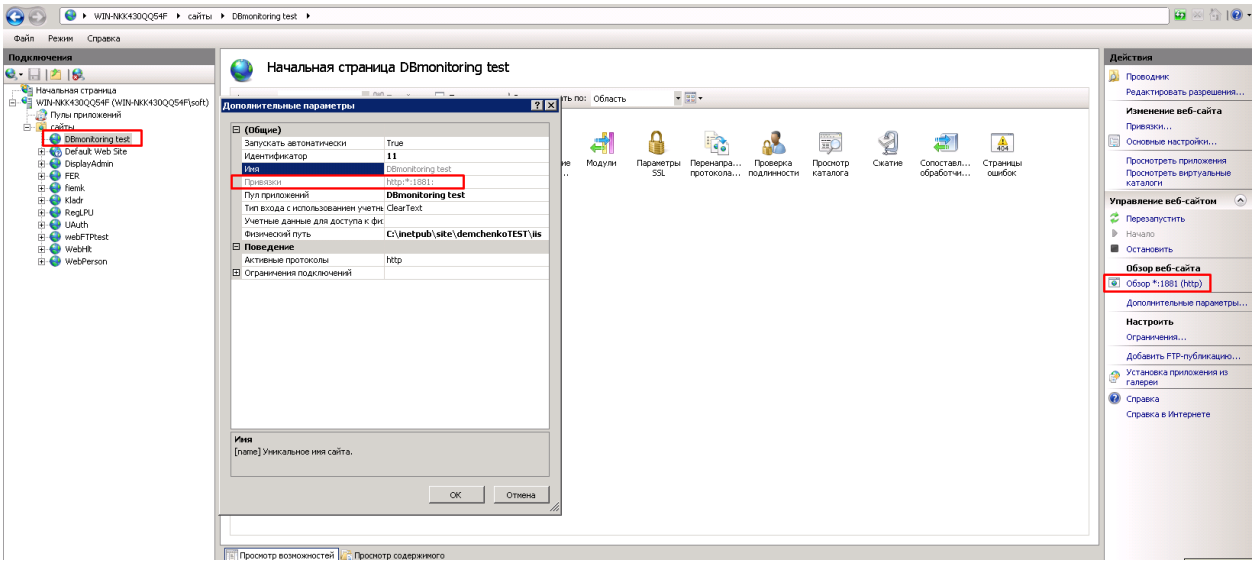

Рисунок 5- Размещение приложение в диспетчере служб IIS

После размещения приложения, попробуем получить ответ от сервиса, указав в браузере выбранный нами порт и локальный ip адрес машины, на которой размещена служба IIS, так же нужно добавить путь до нужного нам метода – «/api/reports», и того получаем - http://localhost:1881/api/reports. Проверим ответ от работающего приложения, рисунок 6:

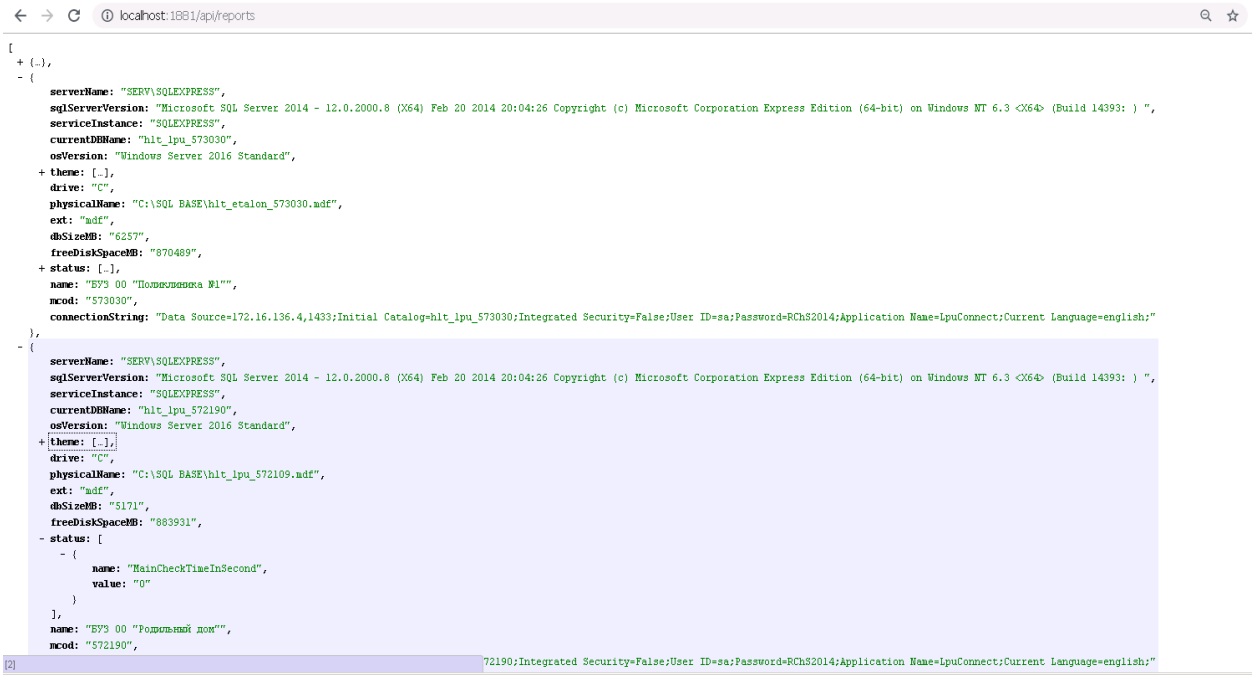

Рисунок 6 - Ответ приложения в виде модели ЛПУ

Как видим, на скриншоте выше модель ЛПУ нам возвращается, а это означает, что сервис работает. Попробуем обратится на страницу со сводной информацией.

За страницу со сводной информацией отвечает такой инструмент как Grafana. Она отвечает за удобное представление информации, полученной от сервиса. Сам сервис Grafana используется в ООО «Софтраст» другими сервисами, поэтому разворачивать новый сервис нет надобности, используемый уже размещённый сервис и обратимся к его панели. Предварительно были настроены счётчики, учитывающие количественные показатели, возвращающиеся в модели ЛПУ. Интерфейс пользователя с использование Grafana можно увидеть на рисунке 7:

Рисунок 7 - Интерфейс пользователя и использованием Grafana

На ней мы видим количество подключений, которые можно идентифицировать по MCOD и сводную информацию о состоянии подключений к ним. Из информации видно, что все 45 учреждений (и 1 тестовый сервис с идентификатором «777777» находятся онлайн и доступны, по результатам опрошенных проверок.

Ещё один пример системы мониторинга, когда один из хостов не прошёл список проверок, рисунок 8:

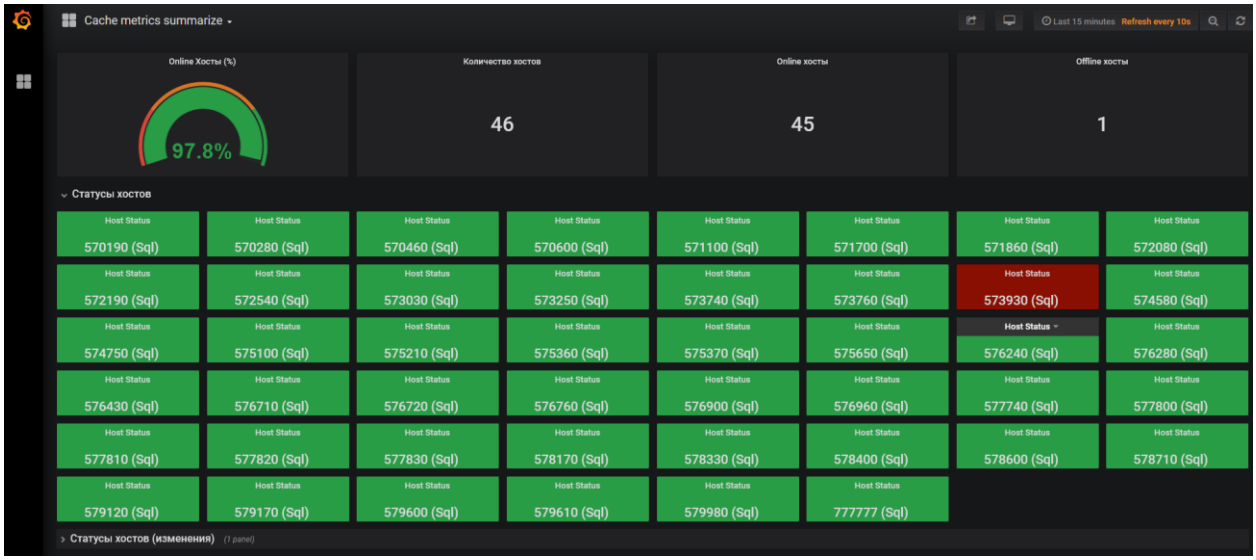

Рисунок 8 - Проблемы с одной из ЛПУ

Ещё один пример из Grafana, где мы видим ответ учреждения на один из методов проверки, рисунок 9:

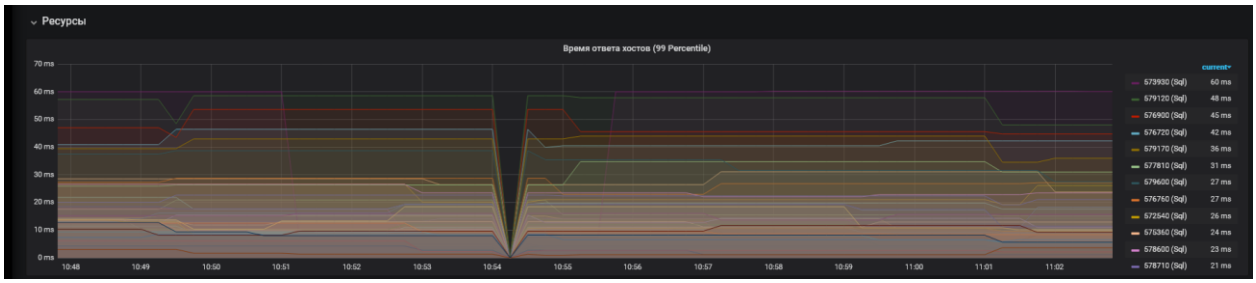

Рисунок 9 - Неудачный результат опроса, указывает что были кратковременные проблемы с сетью площадки

Из сервиса мониторинга видно, что в один из моментов опроса учреждений сервис зафиксировал «провал» в скорости ответа от учреждений. А именно массовый сбой сети, но так как сбой был кратковременный, и не подтвердился при следующей проверке, никакой проблемы из себя это не представляет.

Такой инструмент как Grafana представляет из себя очень удобный интерфейс для вывода информации, поэтому было решено использовать его для отображения информации в более удобном для пользователя виде.

3.3 Измерение скорости работы системы в условиях продуктивной нагрузки

По результатам проверок в продуктивной среде было выяснено, что сервис не создаёт заметной нагрузки на сервера учреждений, а также не создаёт нагрузки в плане сетевого трафика на сеть региона.

Средствами диагностики Visual Studio было установлено, что сервис готов отвечать на запросы спустя 14.9 секунд после запуска, что подтверждает статистика на рисунке 10 [13]:

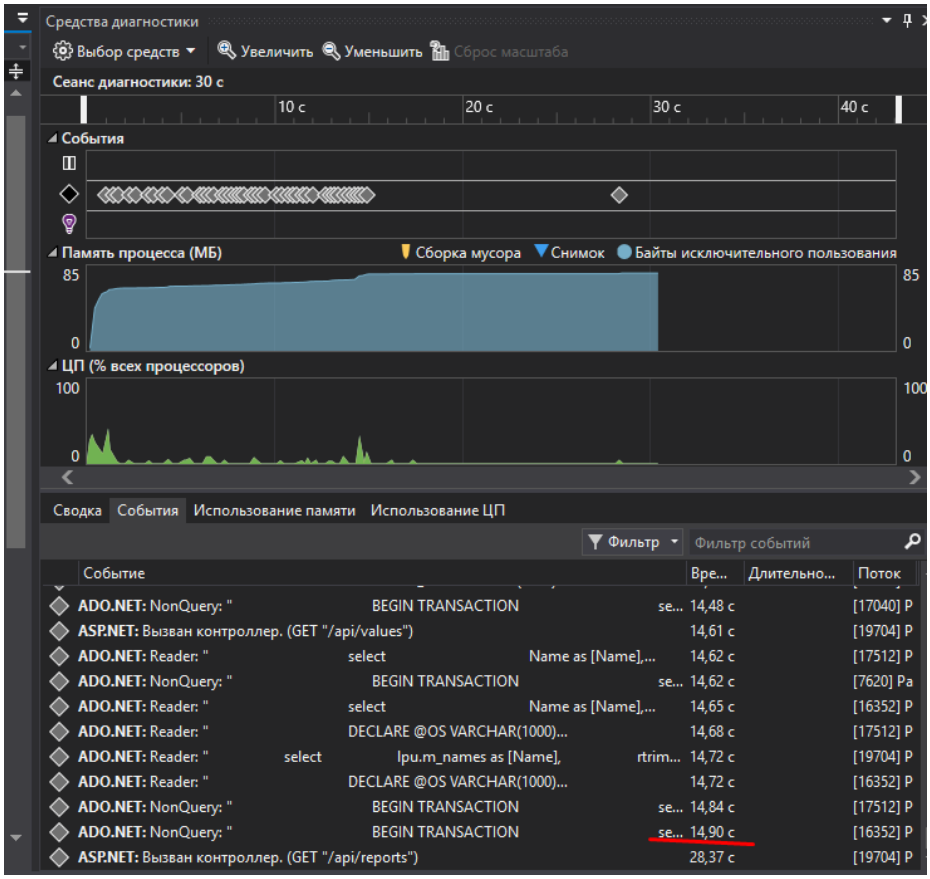

Рисунок 10 - Средства диагностики Visual Studio

Следующий опрос начинается по установке таймера, для тестирования использовался период 1 минута, проверим средствами диагностики, рисунок 11:

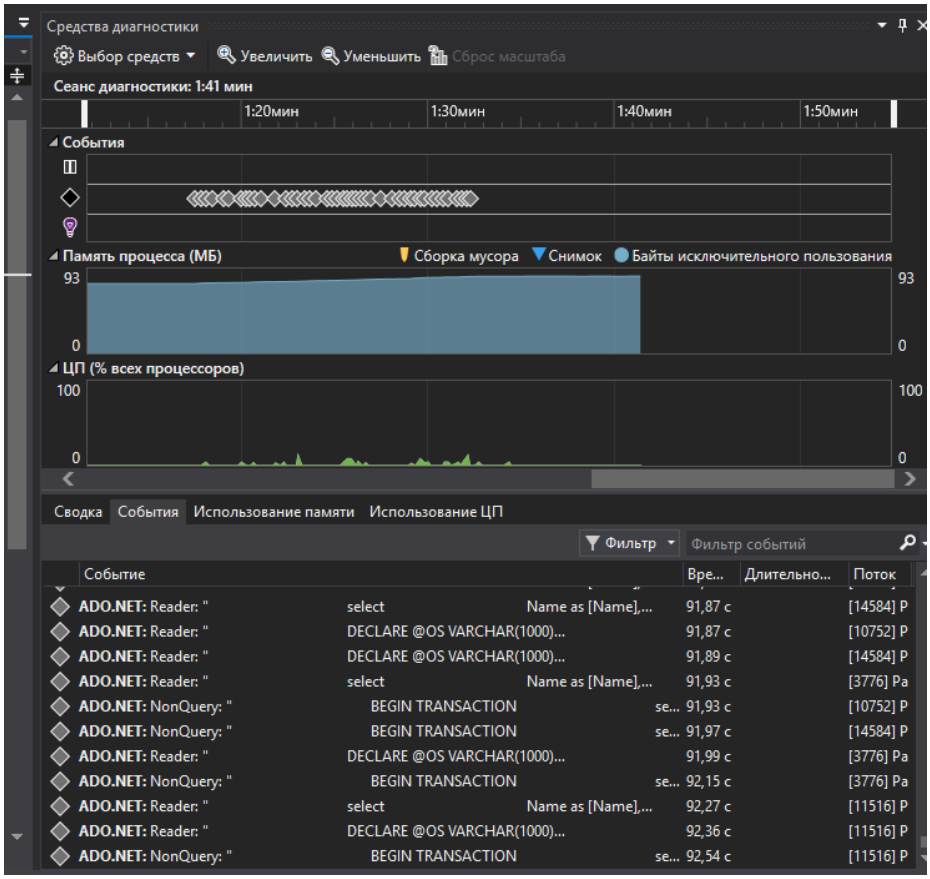

Рисунок 11 - Средства диагностики Visual Studio

Как видим опрос происходит успешно, процессы не зависают, а память приложения очищается после каждого успешно открытого соединения.

Используя асинхронные методы программирования, важно понимать, что вместе с большой производительностью приходят риски утечки памяти и чем больше будет производительность приложения, тем быстрее память может быть израсходована.

Систему удалось реализовать таким образом, чтобы избежать описанных выше проблем. Сервис не создаёт нагрузки в момент опроса учреждений, правильно работает с внутренним хранилищем, не позволяя ему бесконтрольно расти даже после множества цикличных опросов, что подтверждается на рисунке 12:

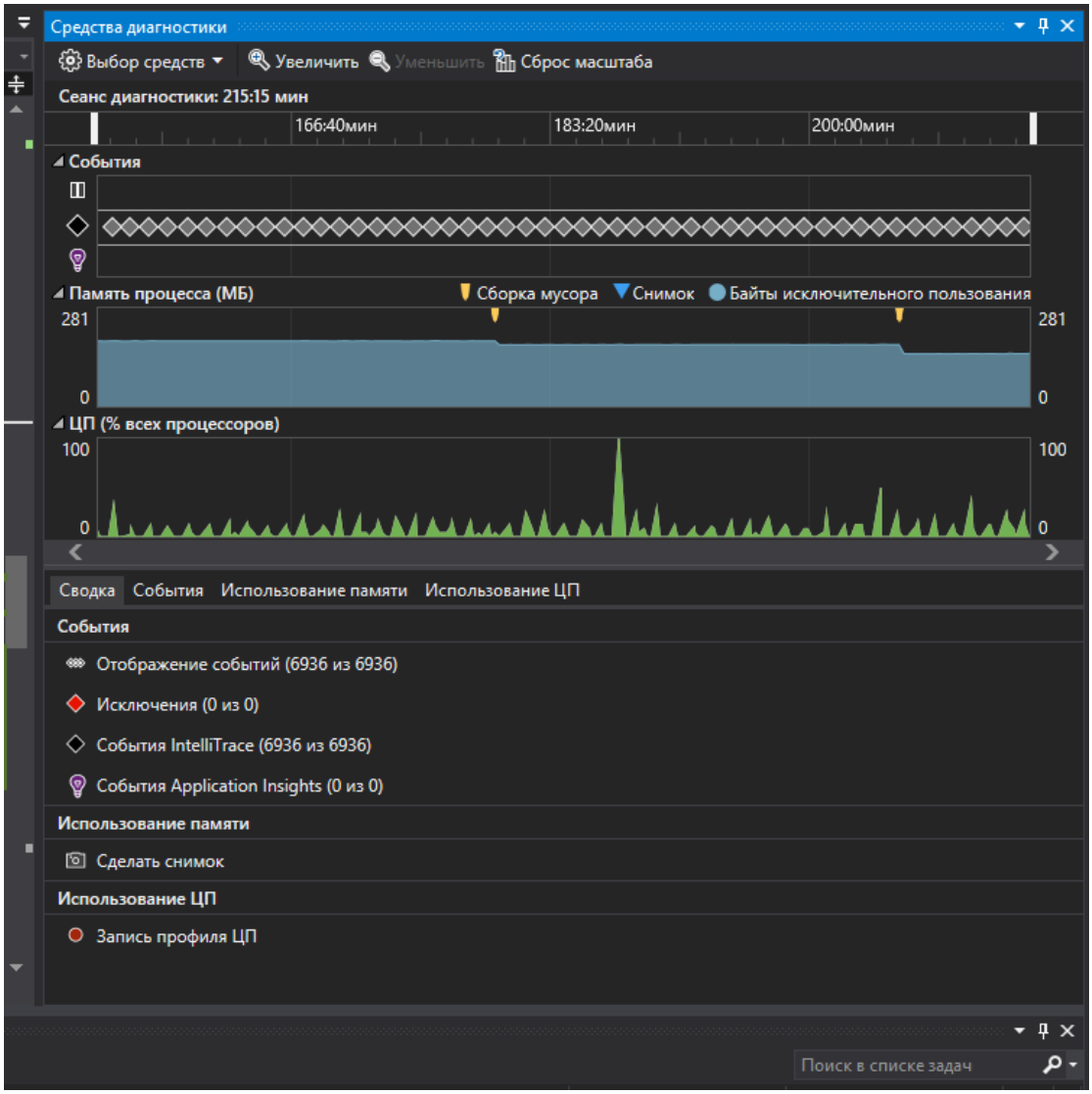

Рисунок 12 - Результат работы сервиса после более чем 4 часов работы в продуктивной среде

Запросы системы были проанализированы, с точки зрения общения приложения с SQL сервером. Для анализа запросов в БД использовались стандартные инструменты SQL Server Management Studio (SSMS), представленные на рисунке 13:

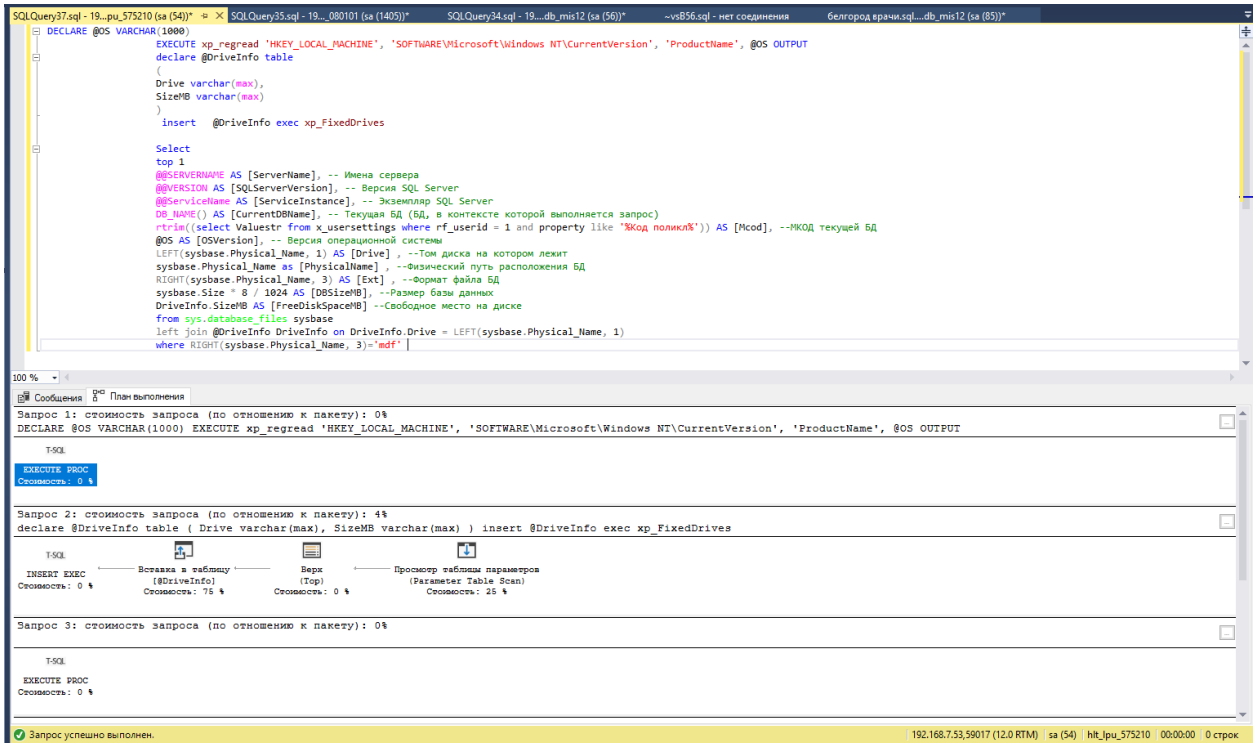

Рисунок 13 - Анализ SQL запросов системы мониторинга

Из скриншота выше видно, что запрос на выбор информации от отдельно взятого сервера составляет менее 1 секунды. Что позволяет говорить о правильной организации схемы взаимодействия приложения и серверов баз данных [18].

Запрос на построение общего списка опрашиваемых строк подключения так же составляет менее 1 секунды:

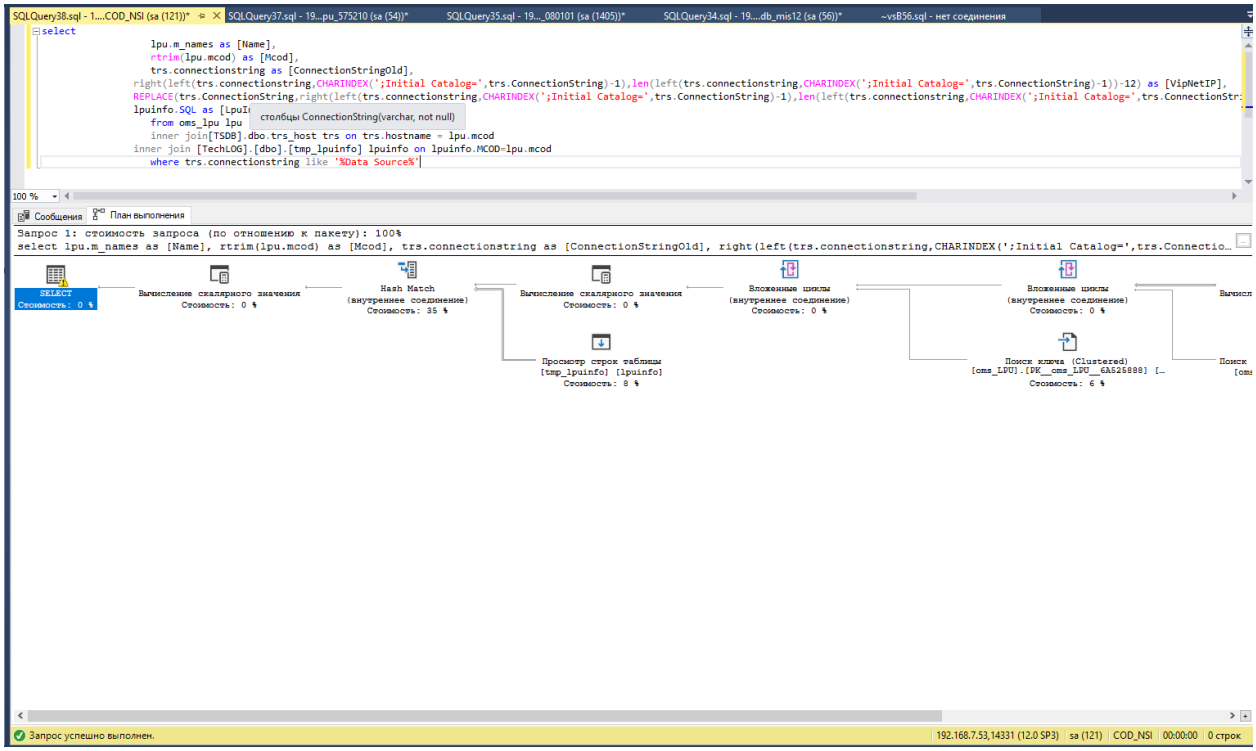

Рисунок 14 - Анализ SQL запросов системы мониторинга

Фактические результаты работы системы мониторинга после тестирования производительности позволяют говорить об успешном запуске сервиса в продуктивной среде.

3.4 Анализ результатов

Анализ результатов тестирования говорит нам о том, что система выполняет поставленные перед ней цели и задачи по предоставлению оперативной информации о состоянии канала связи и конечных баз данных в рамках региональной закрытой сети.

Система стабильно под нагрузкой и может работать в рамках уже существующих процессов, не вызывая блокировок, задержек, помех в уже существующем канале.

Система позволяет вносить изменения в методологию уже существующих условий, добавлять новые алгоритмы проверки.

ЗАКЛЮЧЕНИЕ

Результатом исследовательской работы является полностью готовая для ввода в продуктивную эксплуатацию система асинхронного опроса БД и тестирования канала связи. Результатом работы системы является предоставления отчётных данных в виде графиков или текстовом формате с количественными показателями оценки проверочных алгоритмов.

Система основана на современных средствах разработки, благодаря чему получилось добиться высокой производительности а так же гибкости приложения. Это заключается в возможности размещения на WINDOWS и LINUX системах и работе с базами данных, размещёнными на разных платформах [30].

Набор алгоритмов для тестирования в канале связи является достаточным, для оценки состояния канала связи, поиска ошибок в нём и тестирования конечных компонентов системы. Система готова к расширению моделей алгоритмов, добавлению новых и редактирование старых алгоритмов. Алгоритмический язык, который используется в проверках использует синтаксис SQL запросов объединённых с возможностями утилиты CMD.

После ввода системы в продуктивную эксплуатацию ожидается снижение времени ответа на запросы связанные с неработоспособностью системы. Повышение эффективности отказоустойчивости системы, за счёт локализации проблемы в момент её возникновения. Сокращение времени на поиск проблем в неработоспособности компонентов программного обеспечения «ТМ МИС SaaS».

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Абросимов, Л. И. Базисные методы проектирования и анализа сетей ЭВМ. [Текст] / Л.И. Абросимов. - М.: Университетская книга, 2015. – 248 с.
2. Арсеновски Даниэль Рефакторинг в C# и ASP.NET для профессионалов [Текст] / Вильямс - М., 2010. - 528 с.
3. Анашкина, Н. В. Технологии и методы программирования [Текст] / Academia, 2012. - 384 с.
4. Афонин, В. В. Моделирование систем [Текст] / Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2016. - 231 с.
5. Ботуз, С. П. Управление удаленным доступом. Защита интеллектуальной собственности в сети Internet [Текст] / Солон-Пресс, 2011. - 256 с
6. Гагарина, Л. Г. Разработка и эксплуатация автоматизированных информационных систем [Текст] / Л.Г. Гагарина. - М.: Форум, Инфра-М, 2015. - 384 с.
7. Гросс Кристиан C# 2008 и платформа .NET 3.5 Framework [Текст] / Вильямс - М., 2009. - 480 с.
8. Емельянова, Н. З. Проектирование информационных систем [Текст] / Н.З. Емельянова, Т.Л. Партыка, И.И. Попов. - М.: Форум, 2012. - 432 с.
9. Зыков, С. В. Основы современного программирования [Текст] / С.В. Зыков. - М.: Горячая линия - Телеком, 2016. - 444 с.
10. Иванова, Г. С. Объектно-ориентированное программирование. Учебник [Текст] / Г.С. Иванова, Т.Н. Ничушкина. - М.: МГТУ им. Н. Э. Баумана, 2014. - 320 с.
11. Калихман, И. Л. Динамическое программирование в примерах и задачах. Учебное пособие [Текст] / И.Л. Калихман, М.А. Войтенко. - М.: Высшая школа, 2015. - 125 с.
12. Карпенко, А. П. Современные алгоритмы оптимизации. Учебное пособие [Текст] / А.П. Карпенко. - М.: МГТУ им. Н. Э. Баумана, 2014. - 71 с.
13. Котляров, В. П. Основы тестирования программного обеспечения [Текст] / В.П. Котляров, Т.В. Коликова. - М.: Интернет-университет

информационных технологий, Бином. Лаборатория знаний, 2011. - 288 с.

14. Лупин, С. А. Технологии параллельного программирования [Текст] / С.А. Лупин, М.А. Посыпкин. - М.: Форум, Инфра-М, 2016. – 208 с.
15. Мещеряков, С. В. Эффективные технологии создания информационных систем [Текст] / С.В. Мещеряков, В.М. Иванов. - М.: Политехника, 2015. – 308 с.
16. Орлов, С. А. Технологии разработки программного обеспечения [Текст] / С.А. Орлов, Б.Я. Цилькер. - М.: Питер, 2012. - 608 с.
17. Симан Марк Внедрение зависимостей в .NET Framework [Текст] / Питер - М., 2012. - 276 с.
18. Селко, Джо SQL для профессионалов. Программирование / Джо Селко. - М. Framework [Текст] / ЛОРИ, 2015. - 464 с
19. Семенов, Ю. А. Алгоритмы телекоммуникационных сетей. В 3 частях. Часть 1. Алгоритмы и протоколы каналов и сетей передачи данных [Текст] / Ю.А. Семенов. - М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2015. - 640 с.
20. Семенов, Ю. А. Алгоритмы телекоммуникационных сетей. В 3 частях. Часть 2. Протоколы и алгоритмы маршрутизации в Internet [Текст] / Ю.А. Семенов. - М.: Бином. Лаборатория знаний, Интернет-университет информационных технологий, 2011. - 800 с.
21. Семенов, Ю. А. Алгоритмы телекоммуникационных сетей. В 3 частях. Часть 3. Процедуры, диагностика, безопасность [Текст] / Ю.А. Семенов. - М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2012. - 511 с.
22. Соболь, Б. В. Методы оптимизации. Практикум [Текст] / Б.В. Соболь, Б.Ч. Месхи, Г.И. Каныгин. - М.: Феникс, 2012. - 380 с.
23. Тормасов, А. Г. Параллельное программирование многопоточных систем с разделяемой памятью [Текст] / А.Г. Тормасов. - М.: Физматкнига, 2014. – 208 с.
24. Тузовский, А. Ф. Проектирование и разработка web-приложений. Учебное

пособие [Текст] / А.Ф. Тузовский. - М.: Юрайт, 2016. - 218 с.

25. Чедвик Джесс , Снайдер Тодд , Панда Хришикеш ASP.NET MVC 4. Разработка реальных веб-приложений с помощью ASP.NET MVC [Текст] / Вильямс - М., 2013. - 432 с
26. Bill Wagner Хранилище документации Майкрософт для пользователей, разработчиков и ИТ-специалистов [Электронный ресурс] / https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/concepts/async/ (дата обращения: 11.12.2018)
27. Rick Anderson Хранилище документации Майкрософт для пользователей, разработчиков и ИТ-специалистов [Электронный ресурс] / https://docs.microsoft.com/ru-ru/aspnet/#pivot=aspnet (дата обращения: 01.02.2019)
28. Vladimir Goncharov Обзор систем мониторинга серверов [Электронный ресурс] / https://habr.com/ru/post/331016/ (дата обращения:)
29. Автор неизвестен Асинхронное программирование [Электронный ресурс] / https://metanit.com/sharp/tutorial/13.3.php (дата обращения: 07.11.2018)
30. Автор неизвестен [Электронный ресурс] / https://metanit.com/sharp/mvc5/ (дата обращения: 02.03.2019)

ПРИЛОЖЕНИЕ А

Рисунок 15- Пример использование GRAFANA

Таблица 1 - Список учреждений Орловской области

Столбец 1	Столбец 2	Столбец 3	Столбец 4
Номер	Название учреждения	МКОД	IP в ЗСПД
1	БУЗ ОО "Нарышкинская ЦРБ"	570190	172.16.184.4
2	БУЗ ОО "Верховская ЦРБ"	570280	172.16.144.4
3	БУЗ ОО "Орловская областная клиническая больница ООКБ"	570460	172.16.167.4
4	БУЗ ОО "Детская стоматологическая поликлиника"	570600	172.16.177.4
5	БУЗ ОО "Новодеревеньковская ЦРБ"	571100	172.16.180.4
6	БУЗ ОО "Корсаковская ЦРБ"	571700	172.16.134.4
7	БУЗ ОО "Орловский областной кожно-венерологический диспансер ООКВД"	571860	172.16.139.4
8	БУЗ ОО "Малоархангельская ЦРБ"	572080	172.16.179.4
9	БУЗ ОО "Родильный дом"	572190	172.16.163.4
10	БУЗ ОО "Орловская областная стоматологическая поликлиника ООСП"	572540	172.16.168.4

Продолжение таблицы 1

Столбец 1	Столбец 2	Столбец 3	Столбец 4
11	БУЗ ОО "Поликлиника №1"	573030	172.16.136.4
12	БУЗ ОО "Дмитровская ЦРБ"	573740	172.16.173.4
13	БУЗ ОО "Троснянская ЦРБ"	573930	172.16.182.4
14	БУЗ ОО "Знаменская ЦРБ"	574580	172.16.132.4
15	БУЗ ОО "Орловский онкологический диспансер ООД"	574750	172.16.141.5
16	БУЗ ОО "Орловский областной врачебно-физкультурный диспансер ООВФД"	575100	172.16.195.4
17	БУЗ ОО "Детская поликлиника №2"	575210	172.16.137.4
18	БУЗ ОО "Плещеевская ЦРБ"	575360	172.16.162.4
19	БУЗ ОО "Новосильская ЦРБ"	575370	172.16.196.4
20	БУЗ ОО "Сосковская ЦРБ"	575650	172.16.181.4
21	БУЗ ОО "Должанская ЦРБ"	576240	172.16.191.4
22	БУЗ ОО "Залегощенская ЦРБ"	576280	172.16.143.4
23	БУЗ ОО "Поликлиника №5"	576430	172.16.131.4
24	БУЗ ОО "Поликлиника №3"	576720	172.16.165.4
25	БУЗ ОО "Ливенская ЦРБ"	576760	172.16.153.4
26	БУЗ ОО "Городская больница им. С.П. Боткина"	576900	172.16.161.4
27	БУЗ ОО "Больница скорой медицинской помощи им. Н.А. Семашко"	576960	172.16.140.4
28	БУЗ ОО "Свердловская ЦРБ"	577740	172.16.147.4
29	БУЗ ОО "Детская поликлиника №3"	577800	172.16.138.4
30	БУЗ ОО "Шаблыкинская ЦРБ"	577820	172.16.183.4
31	БУЗ ОО "Хотынецкая ЦРБ"	577830	172.16.133.4
32	БУЗ ОО "Глазуновская ЦРБ"	578170	172.16.145.4

Окончание таблицы 1

Столбец 1	Столбец 2	Столбец 3	Столбец 4
33	БУЗ ОО "Колпнянская ЦРБ"	578400	172.16.178.4
34	БУЗ ОО «НКМЦ им. З. И. Круглой»	578600	172.16.135.4
35	БУЗ ОО "Болховская ЦРБ"	578710	172.16.172.4
36	БУЗ ОО "Детская поликлиника №1"	579120	172.16.164.4
37	БУЗ ОО "Мценская ЦРБ"	579170	172.16.160.4
38	БУЗ ОО "Поликлиника №2"	579600	172.16.130.4
39	БУЗ ОО "Покровская ЦРБ"	579980	172.16.146.4
40	БУЗ ОО "Орловский наркологический диспансер ОНД"	579610	172.16.188.4
41	БУЗ ОО "Орловский центр СПИД"	573760	172.16.170.4
42	БУЗ ОО "Орловский психоневрологический диспансер ОПНД"	573250	172.16.187.4
43	БУЗ ОО "Орловский противотуберкулезный диспансер ОПТД"	576710	172.16.169.4
44	БУЗ ОО "Краснозоренская ЦРБ"	577810	172.16.176.4
45	БУЗ ОО "Кромская ЦРБ"	578330	172.16.129.4

Код одного из файлов приложения, отвечающих за первичную инициализацию приложения и формирование списка строк подключения.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Test3.Models;
using NPoco;
using System.Data.Common;
using System.Data;
using System.Threading;
using System.Data.SqlClient;
using System.Diagnostics;
```

```
using System.Threading;



namespace Test3.Services
{

   public class LpuService
   {
      public List<Lpu> GetLpu()
      {
         //SqlConnection sqlCod = GetConnectionString("Data Source=192.168.7.48,14335;Initial Catalog=Cod_NSI; User id = sa; Password = 4s7Nv0878nxh344;MultipleActiveResultSets=True;");
         //SqlConnection sqlCod = GetConnectionString("Data Source=192.168.7.188,5000 ;Initial Catalog=COD_NSI;Integrated Security=False;User ID=sa;password=Sentr-18;Language=english;Encrypt=False;Application Name=2dr;Current Language=english;");
         //SqlConnection sqlCod = GetConnectionString("Data Source=10.16.0.13,1433;Initial Catalog=Cod_NSI; User id = sa; Password = 1ZX18dcf9;MultipleActiveResultSets=True;");
         SqlConnection sqlCod = GetConnectionString("Data Source=192.168.7.53,14331;Initial Catalog=Cod_NSI; User id = sa; Password = 1ZX18dcf9;MultipleActiveResultSets=True;");
         try { sqlCod.Open(); }
         catch (Exception e) { Console.WriteLine("Error: нет связи с ЦОД " + e.Message); }

         List<Lpu> lpuList;

         using (var dbLpu = new Database(sqlCod, DatabaseType.SqlServer2008, default(IsolationLevel?), false))
         {
            //Получение списка ЛПУ, Орловская область через пробросы
            lpuList = dbLpu.Fetch<Lpu>(@"
              select
              lpu.m_names as [Name],
              rtrim(lpu.mcod) as [Mcod],
              trs.connectionstring as [ConnectionStringOld],
            right(left(trs.connectionstring,CHARINDEX(';Initial Catalog=',trs.ConnectionString)-1),len(left(trs.connectionstring,CHARINDEX(';Initial Catalog=',trs.ConnectionString)-1))-12) as [VipNetIP],
```

```
REPLACE(trs.ConnectionString,right(left(trs.connectionstring,CHARINDEX(';Initia
l                                                    Catalog=',trs.ConnectionString)-
1),len(left(trs.connectionstring,CHARINDEX(';Initial
Catalog=',trs.ConnectionString)-1))-12),lpuinfo.SQL) as [ConnectionString],
            lpuinfo.SQL as [LpuInfoRoute]
              from oms_lpu lpu
              inner join[TSDB].dbo.trs_host trs on trs.hostname = lpu.mcod
            inner join [TechLOG].[dbo].[tmp_lpuinfo] lpuinfo on
lpuinfo.MCOD=lpu.mcod
              where trs.connectionstring like '%Data Source%'
            ");

            ////Получение списка ЛПУ, Орловская область
            //lpuList = dbLpu.Fetch<Lpu>(@"
            //select
            //            lpu.m_names as [Name],
            //            rtrim(lpu.mcod) as [Mcod],
            //            trs.connectionstring as [ConnectionString],
            //                        right(left(trs.connectionstring,CHARINDEX(';Initial
Catalog=',trs.ConnectionString)-
1),len(left(trs.connectionstring,CHARINDEX(';Initial
Catalog=',trs.ConnectionString)-1))-12) as [VipNetIP],
            //
REPLACE(trs.ConnectionString,right(left(trs.connectionstring,CHARINDEX(';Initia
l                                                    Catalog=',trs.ConnectionString)-
1),len(left(trs.connectionstring,CHARINDEX(';Initial
Catalog=',trs.ConnectionString)-1))-12),lpuinfo.SQL) as [ConnectionStringOld],
            //          lpuinfo.SQL as [LpuInfoRoute]
            //            from oms_lpu lpu
            //            inner join[TSDB].dbo.trs_host trs on trs.hostname = lpu.mcod
            //                      inner join [TechLOG].[dbo].[tmp_lpuinfo] lpuinfo on
lpuinfo.MCOD=lpu.mcod
            //            where trs.connectionstring like '%Data Source%'
            //          ");

            ////Получение списка ЛПУ, московская область
            //lpuList = dbLpu.Fetch<Lpu>(@"
            //   select
            //   lpu.m_names as [Name],
            //   rtrim(lpu.mcod) as [Mcod],
            //   trs.connectionstring as [ConnectionStringOld],
            //   prox.orig,
            //   prox.repl,
            //              REPLACE(trs.connectionstring, prox.orig, prox.repl) as
```

```
[ConnectionString]
            //    from oms_lpu lpu
            //    inner
            //    join[TSDB].dbo.trs_host trs on trs.hostname = lpu.mcod
            //    inner join[TechLOG].dbo.proxy prox on trs.ConnectionString like '%' +
prox.orig + '%'
            //    where trs.connectionstring like '%Data Source%'
            //");
            return lpuList;

        }
    }


    public List<LpuMonitor> GetLpuInfo(string id)
    {
        Stopwatch sWatch = new Stopwatch();
        List<Lpu> Lpulist = GetLpu().FindAll(x => x.Mcod == id);
        Lpu found = Lpulist.Find(x => x.Mcod == id);

        SqlConnection sqlLpu = GetConnectionString(found.ConnectionString);

        List<LpuStatus> LpuStatusList = new List<LpuStatus>();
        try { sqlLpu.Open(); }
        catch (Exception e) { Console.WriteLine("Error: " + e.Message); }

        using (var dbLpu = new Database(sqlLpu, DatabaseType.SqlServer2008,
default(IsolationLevel?), false))
        {

            List<Theme> ThemeList = dbLpu.Fetch<Theme>(@"
                select
                Name as [Name],
                NumVersion as [Version]
                from x_theme where ThemeID>0");

            List<LpuMonitor> lpuMonitorList = dbLpu.Fetch<LpuMonitor>(@"
                DECLARE @@OS VARCHAR(1000)
                EXECUTE xp_regread 'HKEY_LOCAL_MACHINE',
'SOFTWARE\Microsoft\Windows NT\CurrentVersion', 'ProductName', @@OS
OUTPUT
                declare @@DriveInfo table
                (
                Drive varchar(max),
                SizeMB varchar(max)
```

```
            )
             insert   @@DriveInfo exec xp_FixedDrives

            Select
            top 1
            @@@SERVERNAME AS [ServerName], -- Имена сервера
            @@@VERSION AS [SQLServerVersion], -- Версия SQL Server
            @@@ServiceName AS [ServiceInstance], -- Экземпляр SQL Server
            DB_NAME() AS [CurrentDBName], -- Текущая БД (БД, в контексте которой выполняется запрос)
            rtrim((select Valuestr from x_usersettings where rf_userid = 1 and property like '%Код поликл%')) AS [Mcod], --МКОД текущей БД
            @@OS AS [OSVersion], -- Версия операционной системы
            LEFT(sysbase.Physical_Name, 1) AS [Drive] , --Том диска на котором лежит
            sysbase.Physical_Name as [PhysicalName] , --Физический путь расположения БД
            RIGHT(sysbase.Physical_Name, 3) AS [Ext] , --Формат файла БД
            sysbase.Size * 8 / 1024 AS [DBSizeMB], --Размер базы данных
            DriveInfo.SizeMB AS [FreeDiskSpaceMB] --Свободное место на диске
            from sys.database_files sysbase
            left join @@DriveInfo DriveInfo on DriveInfo.Drive = LEFT(sysbase.Physical_Name, 1)
            where RIGHT(sysbase.Physical_Name, 3)='mdf' '');

        sWatch.Start();
        dbLpu.Execute(@"
            BEGIN TRANSACTION
            select top 100 * into tmp_mkabLpuService from hlt_MKAB
            select * from tmp_mkabLpuService
            drop table tmp_mkabLpuService
            COMMIT
            select 1 as [Name],2 as [Value]");
        sWatch.Stop();

        var newResult = new LpuStatus()
        {
            Name = "MainCheckTimeInSecond",
            Value = ((int)sWatch.ElapsedMilliseconds / 1000).ToString(),
        };
        LpuStatusList.Add(newResult);

        foreach (var a in lpuMonitorList)
        {
            a.Theme = ThemeList.FindAll(x => x.Name != "");
```

```
                a.Name = found.Name;
                //a.Name = sWatch.ElapsedMilliseconds.ToString();
                a.Mcod = found.Mcod;
                a.ConnectionString = found.ConnectionString;
                a.Status = LpuStatusList.FindAll(x => x.Name == "MainCheckTimeInSecond");
            }
            return lpuMonitorList;

        }
    }


    public List<LpuMonitor> GetLpuReport()
    {
        List<Lpu> Lpulist = GetLpu();

        var LpuReportList = new List<LpuMonitor>();

        Lpulist.AsParallel().WithDegreeOfParallelism(100).ForAll
            (a =>
            {
            try
            {
                SqlConnection sqlLpu = GetConnectionString(a.ConnectionString);

                sqlLpu.Open();
                Stopwatch sWatch = new Stopwatch();
                List<LpuStatus> LpuStatusList = new List<LpuStatus>();

                using (var dbLpuQuery = new Database(sqlLpu, DatabaseType.SqlServer2008, default(IsolationLevel?), false))
                {
                    List<Theme> ThemeList = dbLpuQuery.Fetch<Theme>(@"
                            select
                            Name as [Name],
                            NumVersion as [Version]
                            from x_theme where ThemeID>0");

                    List<LpuMonitor> lpuMonitorList = dbLpuQuery.Fetch<LpuMonitor>(@"
                            DECLARE @@OS VARCHAR(1000)
                            EXECUTE xp_regread 'HKEY_LOCAL_MACHINE', 'SOFTWARE\Microsoft\Windows NT\CurrentVersion', 'ProductName', @@OS OUTPUT
```

```
declare @@DriveInfo table
(
Drive varchar(max),
SizeMB varchar(max)
)
 insert   @@DriveInfo exec xp_FixedDrives

Select
top 1
@@@SERVERNAME AS [ServerName], -- Имена сервера
@@@VERSION AS [SQLServerVersion], -- Версия SQL Server
@@@ServiceName AS [ServiceInstance], -- Экземпляр SQL Server
DB_NAME() AS [CurrentDBName], -- Текущая БД (БД, в контексте которой выполняется запрос)
rtrim((select Valuestr from x_usersettings where rf_userid = 1 and property like '%Код поликл%')) AS [Mcod], --МКОД текущей БД
@@OS AS [OSVersion], -- Версия операционной системы
LEFT(sysbase.Physical_Name, 1) AS [Drive] , --Том диска на котором лежит
sysbase.Physical_Name as [PhysicalName] , --Физический путь расположения БД
RIGHT(sysbase.Physical_Name, 3) AS [Ext] , --Формат файла БД
sysbase.Size * 8 / 1024 AS [DBSizeMB], --Размер базы данных
DriveInfo.SizeMB AS [FreeDiskSpaceMB] --Свободное место на диске
from sys.database_files sysbase
left join @@DriveInfo DriveInfo on DriveInfo.Drive = LEFT(sysbase.Physical_Name, 1)
where RIGHT(sysbase.Physical_Name, 3)='mdf' '');

        sWatch.Start();
        dbLpuQuery.Execute(@"
            BEGIN TRANSACTION
            select top 1000 * into tmp_mkabLpuService from hlt_MKAB
            select * from tmp_mkabLpuService
            drop table tmp_mkabLpuService
            COMMIT");
        sWatch.Stop();

        int ElapsedMilliseconds = 0;
        if ((int)sWatch.ElapsedMilliseconds < 1000) { ElapsedMilliseconds
```

```
= 1; } else { ElapsedMilliseconds = ((int)sWatch.ElapsedMilliseconds / 1000);}

                    //Заполняем список проверок
                        var newResult = new LpuStatus()
                        {
                            Name = "MainCheckTimeInSecond",
                            Value = ElapsedMilliseconds.ToString(), //Конвертирование
из миллисекунд в секунды
                            //Value = sWatch.ElapsedMilliseconds.ToString(),
//Конвертирование из миллисекунд в секунды
                        };
                        LpuStatusList.Add(newResult);
                        //Проверка канала связи от серверов ЦОД до БД учреждения
                        newResult = new LpuStatus()
                        {
                            Name = "CorrectLpuString",
                            Value = "true",
                        };
                        LpuStatusList.Add(newResult);

                        newResult = new LpuStatus()
                        {
                            Name = "LpuServerOnline",
                            Value = "true",
                        };
                        LpuStatusList.Add(newResult);

                        newResult = new LpuStatus()
                        {
                            Name = "ConnectCredentials",
                            Value = "true",
                        };
                        LpuStatusList.Add(newResult);
                        // Проверка конфигурации базы данных и сбор данных о
сервере
                        newResult = new LpuStatus()
                        {
                            Name = "SQLServerSaveConfig",
                            Value = "true",
                        };
                        LpuStatusList.Add(newResult);
                        newResult = new LpuStatus()
                        {
                            Name = "SQLCredentials",
                            Value = "true",
```

```
                    };
                    LpuStatusList.Add(newResult);
                    newResult = new LpuStatus()
                    {
                       Name = "SQLServerHealth",
                       Value = "true",
                    };
                    LpuStatusList.Add(newResult);
                    newResult = new LpuStatus()
                    {
                       Name = "SQLNSI",
                       Value = "true",
                    };
                    LpuStatusList.Add(newResult);
                    newResult = new LpuStatus()
                    {
                       Name = "SQLStatus",
                       Value = "true",
                    };
                    LpuStatusList.Add(newResult);
                    newResult = new LpuStatus()
                    {
                       Name = "SQLJobStatus",
                       Value = "true",
                    };
                    LpuStatusList.Add(newResult);
                    newResult = new LpuStatus()
                    {
                       Name = "SQLFreeDiskSpaceMB",
                       Value = "true",
                    };
                    LpuStatusList.Add(newResult);
                    newResult = new LpuStatus()
                    {
                       Name = "SQLMinimumRequirements",
                       Value = "true",
                    };
                    LpuStatusList.Add(newResult);

                    //LpuStatusList.FindAll(l => l.Name == 
"SQLFreeSpace").ForEach(l => l.Value=a.Name) ;
                    //Проверка канала связи от конечной БД до серверов ЦОД
                    //Рекомендации по исправлению на основе полученных 
ошибок
```

```
                    foreach (var b in lpuMonitorList)
                    {
                        LpuStatusList.FindAll(l => l.Name == "SQLFreeDiskSpaceMB").ForEach(l => l.Value = b.FreeDiskSpaceMB);
                        b.Theme = ThemeList.FindAll(x => x.Name != "");
                        b.Name = a.Name;
                        //a.Name = sWatch.ElapsedMilliseconds.ToString();
                        b.Mcod = a.Mcod;
                        b.ConnectionString = a.ConnectionString;
                        b.Status = LpuStatusList.FindAll(x => x.Name != "");

                        LpuReportList.Add(b);
                    }


                }

                sqlLpu.Close();

            }
            catch (Exception e)
            {
                Console.WriteLine("Error: " + e.Message);
            }

        });

    return LpuReportList;


}

public List<LpuMonitor> GetCurrentLpuReportList()
{
    var LpuReportList = GetLpuReport();
    return LpuReportList;
}

public static SqlConnection GetConnectionString(string newString)
{
    var lpuConn = new SqlConnectionStringBuilder(newString);
    lpuConn.ConnectTimeout = 5;
    var sqlLpuConnection = new SqlConnection(lpuConn.ConnectionString);
    return sqlLpuConnection;
}
```

```
    }
}
```

ПРИЛОЖЕНИЕ А

Магистерская диссертация выполнена мной совершенно самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

«___»_____2019 г.

_____ _____

(подпись) *(Ф.И.О.)*