Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Школа информационных технологий и робототехники
Направление подготовки 09.04.04 Программная инженерия
Отделение школы (НОЦ) информационных технологий

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Тема работы
Разработка системы мониторинга для облачного приложения (Development of a monitoring system for the cloud application)

УДК 004.353.05:004.62

Студент

Группа	ФИО	Подпись	Дата
8ПМ7И	Васин Максим Алексеевич		

Руководитель

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ	Чердынцев Евгений Сергеевич	к.т.н		

КОНСУЛЬТАНТЫ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Старший преподаватель ОСГН	Потехина Нина Васильевна			

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ООД	Горбенко Михаил Владимирович	к.т.н		

ДОПУСТИТЬ К ЗАЩИТЕ:

Руководитель ООП	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ ИШИТР	Губин Евгений Иванович	к. ф.-м.н.		

Томск – 2019 г.

Запланированные результаты обучения по программе

Код результата	Результат обучения (выпускник должен быть готов)
Общие по направлению подготовки 09.04.04 «Программная инженерия»	
Р1	Проводить научные исследования, связанные с объектами профессиональной деятельности.
Р2	Разрабатывать новые и улучшать существующие методы и алгоритмы обработки данных в информационно-вычислительных системах.
Р3	Составлять отчеты о проведенной научно-исследовательской работе и публиковать научные результаты.
Р4	Проектировать системы с параллельной обработкой данных и высокопроизводительные системы.
Р5	Осуществлять программную реализацию информационно-вычислительных систем, в том числе распределенных.
Р6	Осуществлять программную реализацию системс параллельной обработкой данных и высокопроизводительных систем.
Р7	Организовывать промышленное тестирование создаваемого программного обеспечения
Профиль «Технологии больших данных»/ «Big data solutions»	
Р8	Исследовать и анализировать большие данные, создавать их моделии интерпретировать структуры данных в таких моделях.
Р9	Понимать принципы создания, хранения, управления, передачи и анализа больших данных с использованием новейших технологий, инструментов и систем обработки данных в высокопроизводительных сетях.
Р10	Применять теорию распределенной системы управления базами данных к традиционным распределенным системам реляционных баз данных, облачным базам данных, крупномасштабным системам машинного обучения и хранилищам данных.

Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Школа информационных технологий и робототехники
Направление подготовки 09.04.04 Программная инженерия
Отделение школы (НОЦ) информационных технологий

УТВЕРЖДАЮ:
Руководитель ООП
_____ _____ Губин Е.И._____
(Подпись) (Дата) (Ф.И.О.)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

В форме:

Магистерской диссертации

Студенту:

Группа	ФИО
8ПМ7И	Васин Максим Алексеевич

Тема работы:

Разработка системы мониторинга для облачного приложения (Development of a monitoring system for the cloud application)	
Утверждена приказом директора (дата, номер)	25.02.2019, №1436/с

Срок сдачи студентом выполненной работы:	06.06.2019 г.

ТЕХНИЧЕСКОЕ ЗАДАНИЕ:

Исходные данные к работе	1. Система журналов облачного приложения. 2. Журналы микросервисов. 3. Техническое задание.
Перечень подлежащих исследованию, проектированию и разработке вопросов	1. Анализ особенностей облачных приложений; 2. Анализ системы журналов облачного приложения; 3. Выбор инструментов для мониторинга; 4. Проектирование системы мониторинга; 5. Разработка плагинов для health-мониторинга. 6. Оценка конкурентоспособности разработки, расчет затрат на проведение исследования; 7. Анализ условий труда исполнителей проекта.
Перечень графического материала	1. Концептуальная схема мониторинга. 2. Снимки экранов системы мониторинга. 3. Снимки экранов панелей визуализации.
Консультанты по разделам выпускной квалификационной работы	
Раздел	**Консультант**
Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	Потехина Нина Васильевна

Социальная ответственность	Горбенко Михаил Владимирович
Обязательное приложение на английском языке	Диденко Анастасия Владимировна
Названия разделов, которые должны быть написаны на русском и иностранном языках:	
5 Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	
6 Социальная ответственность	
7 Characteristics and features of the cloud application.	

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	01.03.2019

Задание выдал руководитель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ ИШИТР	Чердынцев Евгений Сергеевич	к.т.н		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8ПМ7И	Васин Максим Алексеевич		

Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Школа информационных технологий и робототехники
Направление подготовки 09.04.04 Программная инженерия
Отделение школы (НОЦ) информационных технологий
Период выполнения весенний семестр 2018/2019 учебного года

Форма представления работы:

магистерская диссертация

КАЛЕНДАРНЫЙ РЕЙТИНГ-ПЛАН
выполнения выпускной квалификационной работы

Срок сдачи студентом выполненной работы:	06.06.2019

Дата контроля	Название раздела (модуля) / вид работы (исследования)	Максимальный балл раздела (модуля)
10.04.2019	Глава 1. Особенности облачного приложения	20
17.04.2019	Глава 2. Проектирование	15
13.05.2019	Глава 3. Программная реализация	20
28.05.2019	Глава 4. Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	20
29.05.2019	Глава 5. Социальная ответственность	15
27.05.2019	Приложение 1. Characteristics and features of the cloud application.	10

Составил преподаватель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ ИШИТР	Чердынцев Евгений Сергеевич	к.т.н		

СОГЛАСОВАНО:

Руководитель ООП	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ ИШИТР	Губин Евгений Иванович	к.ф.-м.н		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ»

Студенту:

Группа	ФИО
8ПМ7И	Васину Максиму Алексеевичу

Школа	**Инженерная школа информационных технологий и робототехники**	**Отделение**	**Информационных технологий**
Уровень образования	магистратура	**Направление/специальность**	09.04.04 программная инженерия

Исходные данные к разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»:	
1. *Стоимость ресурсов научного исследования (НИ): материально-технических, энергетических, финансовых, информационных и человеческих*	1. Оклад инженера – 21 760; 2. Оклад научного руководителя – 33664; 3. Оклад представителя предприятия – 35000;
2. *Нормы и нормативы расходования ресурсов*	Месячная амортизация – 3,1%.
3. *Используемая система налогообложения, ставки налогов, отчислений, дисконтирования и кредитования*	1. Ставки налоговых отчислений во внебюджетные фонды (ст. 426 НК РФ) – 30% 2. Районный коэффициент по г. Томску (ст. 426 НК РФ, Постановление Правительства РФ от 13.05.92. №309) – 1,3
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. *Оценка коммерческого и инновационного потенциала НТИ*	1.QuaD анализ проекта; 2. Диаграмма Исикавы.
2. *Разработка устава научно-технического проекта*	Формирование, цели, задач, ожидаемых результатов, рабочая группа проекта, его ограничений и допущений
3. *Планирование процесса управления НТИ: структура и график проведения, бюджет, риски и организация закупок*	1. Планирование этапов разработки проекта; 2. Определение трудоемкости выполнения работ; 3. Формирование бюджета; 4. Анализ рисков.
4. *Определение ресурсной, финансовой, экономической эффективности*	1. Расчет показателя финансовой эффективности.
Перечень графического материала *(с точным указанием обязательных чертежей):*	
1. *QuaD анализ системы мониторинга для облачного приложения* 2. *Диаграмма Исикавы* 3. *Календарный план-график проекта* 4. *Бюджет затрат* 5. *Матрица вероятности рисков и потерь*	

Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент	Потехина Н.В.			

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8ПМ7И	Васин Максим Алексеевич		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Студенту:

Группа	ФИО
8ПМ7И	Васину Максиму Алексеевичу

Школа	Инженерная школа информационных технологий и робототехники	Отделение	Информационных технологий
Уровень образования	магистратура	Направление/специальность	09.04.04 программная инженерия

Исходные данные к разделу «Социальная ответственность»:	
1. *Описание рабочего места (рабочей зоны, технологического процесса, механического оборудования) на предмет возникновения:* – *вредных проявлений факторов производственной среды (метеоусловия, вредные вещества, освещение, шумы, вибрации, электромагнитные поля, ионизирующие излучения)* – *опасных проявлений факторов производственной среды (механической природы, термического характера, электрической, пожарной и взрывной природы)* – *негативного воздействия на окружающую природную среду (атмосферу, гидросферу, литосферу)* – *чрезвычайных ситуаций (техногенного, стихийного, экологического и социального характера)*	Рабочее место – офисное помещение работодателя. Ширина помещения 6 м, длина – 5 м, высота – 4 м; площадь помещения – 30 м2 ; объём помещения – 120 м3 ; наличие кондиционера; естественная вентиляция помещения: двери, окна, вытяжное вентиляционное отверстие; естественное и искусственное освещение.
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. *Ознакомление с правовыми и организационными вопросами обеспечения безопасности:* – *специальные (характерные для проектируемой рабочей зоны) правовые нормы трудового законодательства;* – *организационные мероприятия при компоновке рабочей зоны*	– изучение специальных правовых норм трудового законодательства при работе с компьютером и орг. техникой (Трудовой кодекс РФ, СанПиН 2.2.2/2.4.1340-03 Гигиенические требования к персональным электронновычислительным машинам и организации работы); – изучение требований к организации рабочих мест пользователей (ГОСТ 12.2.032-78 «ССБТ. Рабочее место при выполнении работ сидя.)

2.1 выявление и анализ вредных факторов проектируемой производственной среды в следующей последовательности: – *физико-химическая природа вредности, её связь с разрабатываемой темой;* – *действие фактора на организм человека;* – *приведение допустимых норм с необходимой размерностью (со ссылкой на соответствующий нормативно-технический документ);* – *предлагаемые средства защиты (сначала коллективной защиты, затем – индивидуальные защитные средства)* *2.2 выявление и анализ опасных факторов проектируемой произведённой среды в следующей последовательности* – *механические опасности (источники, средства защиты;* – *термические опасности (источники, средства защиты);* – *электробезопасность (в т.ч. статическое электричество, молниезащита – источники, средства защиты);* – *пожаровзрывобезопасность (причины, профилактические мероприятия, первичные средства пожаротушения)*	Опасные и вредные факторы: – неблагоприятный климат; – недостаточная освещенность рабочей зоны; – повышенный уровень шума; – умственное перенапряжение; – монотонный режим работы. Мероприятия по защите от вредных факторов включают в себя измерение текущих показателей вредных факторов и обеспечение соблюдения нормативных показателей. Опасные факторы: опасность поражения электрическим током, короткое замыкание, статическое электричество. Для защиты от опасных факторов необходимо проводить организационные и технические мероприятия по предотвращению возникновения опасных ситуаций.
3. Охрана окружающей среды: – *защита селитебной зоны* – *анализ воздействия объекта на атмосферу (выбросы);* – *анализ воздействия объекта на гидросферу (сбросы);* – *анализ воздействия объекта на литосферу (отходы);* – *разработать решения по обеспечению экологической безопасности со ссылками на НТД по охране окружающей среды.*	анализ воздействия объекта на литосферу и атмосферу (отходы, связанные с утилизацией вышедшего из строя ПК, люминесцентных ламп и др.);
4. Защита в чрезвычайных ситуациях: – *перечень возможных ЧС на объекте;* – *выбор наиболее типичной ЧС;* – *разработка превентивных мер по предупреждению ЧС;* – *разработка мер по повышению устойчивости объекта к данной ЧС;* – *разработка действий в результате возникшей ЧС и мер по ликвидации её последствий*	ЧС, которые могут возникнуть в процессе разработки и эксплуатации: - пожар в здании. Требуется следовать инструкциям, чтобы не допустить возникновения ЧС. Однако, если ЧС произошло, требуется следовать протоколу эвакуации из здания, а также вызвать службы для ликвидаций последствий ЧС.

Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент	Горбенко Михаил Владимирович	к.т.н		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8ПМ7И	Васин Максим Алексеевич		

Реферат

Выпускная квалификационная работа содержит 112 страниц, 22 рисунок, 21 таблиц, 23 источников, 10 приложений.

Ключевые слова: визуализация данных, мониторинг, система мониторинга, облачное приложение, бизнес метрики, проектирование системы мониторинга, разработка системы мониторинга.

Объектом исследования является процесс визуализации функционирования облачного приложения. Предметом исследования являются веб сервисы облачного приложения, а также записи журналов этих сервисов.

Цель работы – проектирование и разработки системы мониторинга для облачного приложения, которая нацелена на визуализацию работоспособности приложения, а также отображения текущих показателей, специфичных для данной предметной области.

В магистерской диссертации исследовались особенности визуализации больших данных для отображения бизнес метрик облачных приложений и возможность их реализации программным способом.

В результате исследования была разработана система мониторинга и сконфигурированы к ней панели визуализации метрик и работоспособности приложения.

Область применения: облачные приложения, предназначенные для телемедицины.

Перечень условных обозначений, единиц и терминов

Логи (лог-файлы) – это файлы, содержащие системную информацию работы сервера или компьютера, в которые заносятся определенные действия пользователя или программы.

Дашборд (от англ. Dashboard) – панели управления, отображающие данные в реальном времени.

Метрика – мера, позволяющая получить численное значение некоторого свойства программного обеспечения.

ELK - (ElasticSearch, Logstash, Kibana) Каждый из этих инструментов является полноценным независимым open source продуктом, а все вместе они составляют мощное решение для широкого спектра задач сбора, хранения и анализа данных.

Оглавление

Введение

Актуальность исследования. На сегодняшний день облачные и веб приложения занимают существенную часть в современном обществе. Такие приложения охватывают различные сферы деятельности от продаж до высокотехнологичных предприятий. Некоторые сферы деятельности, критичны к бесперебойности и качеству сервиса. Примером такой сферы является телемедицина – когда врач следит за состоянием пациента или специальных устройств, осуществляющих лечение или диагностику пациентов, в режиме реального времени. Однако ответственность за сбои приложения лежит не на враче, так как он является всего лишь пользователем системы, а на разработчике этого приложения. Чтобы избежать таких проблем, а в следствие и финансовых потерь, следует применять комплекс мер по мониторингу приложения.

Система мониторинга позволяет отслеживать доступность компонентов приложения, собирает показатели в режиме реального времени. Помимо этого, возможно генерировать различные отчеты, например, по использованию ресурсов: загрузка процессора, памяти, жесткого диска и т.д. В случае обнаружения перегрузки какого-либо сервера, можно произвести перераспределение мощностей.

Анализ показателей системы возможен благодаря наличию лог-записей в журнале приложения, в случае облачного приложения с микросервисной архитектурой – в журнале отдельных сервисов. Поэтому каждое приложение должно иметь систему логов. Более того лог-файлы должны быть легко доступны и читаемы. Как правило, обращение к журналу логов происходит после неожиданной ошибки в результате выполнения программного кода, чтобы выяснить проблему и констатировать неисправность, локализовать место (класс, сервис, интерфейс), где код выполнился некорректно. Соответственно, иногда необходимо переключаться между различными журналами, чтобы проследить путь ошибки от начала до конца – для локализации первоначального происхождения.

Стоит отметить что файлы логов это слабо структурированные данные. Поэтому, изучать огромный массив таких файлов нелегкая задача, что следует учитывать при выборе инструментов для анализа логов и мониторинга.

Цель и задачи исследования. Целью магистерской диссертации является проектирование и разработка системы мониторинга облачного приложения. Реализованная система мониторинга нацелена на визуализацию основных бизнес показателей, специфичных для предметной области облачного приложения, доступности сервисов и состояние развертывания приложения. Ожидается, что в рамках реализуемой системы станет легче понимать конечного пользователя приложения, что будет определять необходимость развития нового функционала. Планируется, что после внедрения мониторинга затраты на ресурсы сократятся и будут оптимальными, так как будут доступны точные цифры по каждому параметру.

Для достижения вышеописанной цели были поставлены следующие *задачи*:

1 обоснование необходимости внедрения системы мониторинга;

2 проведение обзора существующих инструментов для мониторинга;

3 построение архитектуры системы мониторинга;

4 конфигурация инструментов для мониторинга;

5 разработка дополнительных плагинов;

6 добавление панелей визуализации;

7 внедрение готового программного модуля.

Основные проблемы, которые затронуты в данной работе:

1 автоматизированная система мониторинга облачного микросервисного приложения в закрытых, частных сетях;

2 обработка большого объема данных журналов микросервисов;

3 анализ воздействия разработанного программного модуля на экономику предприятия (сравнение эффективности работы предприятия до и после внедрения программного модуля);

4 конкурентоспособность разработанного программного модуля на рынке информационных систем (выявление отличительных особенностей программного модуля и сравнение подобных уже существующих систем по необходимым критериям).

Объектом исследования является процесс визуализации бизнес-показателей приложения, и используемых ресурсов. Предметом исследования являются файлы логов сервисов облачного приложения с микросервисной архитектурой.

Методы решения задач и новизна исследования. В данной магистерской диссертации применялся системный подход к изучению функционала приложения. Новизна исследования заключается в применении комплекса современных технологий для визуализации бизнес показателей, специфичных для предметной области и отвечает установленным требованиям облачного приложения.

Научная и практическая значимость работы. Научная значимость исследования заключается в получение новой информации из системы логов и визуализации ее в легкодоступном виде. Внедрение разработанной системы мониторинга позволит минимизировать риски, связанные с внезапным выходом приложения из строя, появлением некорректного поведения. Также разработанная система поможет разработчикам приложения понимать актуальное состояние приложения, чтобы учитывать необходимые трудозатраты при реализации нового функционала.

Глава 1. Особенности облачного приложения

1.1 Описание предметной области.

1.1.1Телемедицина

Телемедицина - инструмент современного здравоохранения, направленный на оптимизацию организации, стандартизацию качества и доступности медицинской помощи [1]. Телемедицина подразумевает использование телекоммуникаций для связи медицинских специалистов с клиниками, больницами, врачами, оказывающими первичную помощь, пациентами, находящимися на расстоянии, с целью диагностики, лечения, консультации и непрерывного обучения [2].

Технологически такого рода телекоммуникация должна обеспечивать прямую передачу медицинской информации в различных форматах (история болезни, лабораторные данные, рентгеновские снимки и результаты КТ, видеоизображения, УЗИ и т.д.), а также видеоконференцсвязь в режиме реального времени между медицинскими учреждениями или врачом и пациентами.

Одной из сфер здравоохранения, где активно применяется телемедицина – это медицина сна. Или, как принято называть данную науку – сомнология. Это очень молодая ветвь медицины, которая исследует различные аспекты и заболевания человеческого сна.

Передаваемая информация должна где-то храниться и обрабатываться. Трендом последних лет является обработка данных в облачных приложениях.

1.1.2 Облачные приложения

Облачные приложения производят все вычисления на удаленных ресурсах, предоставляемы пользователю через интернет как онлайн сервис. Согласно терминологии IEEE, облачные технологии – это парадигма, позволяющая постоянно хранить пользовательскую информацию на интернет-серверах и лишь временно кэшируя ее на стороне пользователя [3].

Существует несколько типов облачных приложений, рассмотрим характерный для данной предметной области – SaaS (от англ. System as a service).

Этот тип предоставляет доступ нескольким клиентам к единому приложение с помощью браузера. Компания разработчик самостоятельно управляет приложением и предоставляет доступ к платформе через интернет [4]. Для обеих сторон есть свои положительные моменты. Для клиента выгода заключается в экономии на оборудование – нет необходимости покупать дорогостоящее высокопроизводительное оборудования и программное обеспечение. Для разработчика модель SaaS позволяет эффективно бороться с нелицензионным использованием программного обеспечения, в силу того, что фактически софт не попадает к конечным заказчикам. Помимо этого, данная концепция снижает затраты на развёртывание и внедрение систем технической и консультационной поддержки продукта, хотя и не исключает их полностью.

Хоть для пользователя это и выглядит как одно единое приложение, зачастую его внутренняя архитектура, с таким типом, выгладит как набор отдельных сервисов, выполняющих строго конкретную задачу, в своем процессе и окружении. Взаимодействие между элементами приложения и внешними компонентами осуществляется, как правило по протоколу HTTP. Данный архитектурный подход называется микросервисным.

Как было сказано выше, идея состоит в том, чтобы разделить приложение на набор небольших взаимосвязанных сервисов, а не разрабатывать одно монолитное приложение. Вместо совместного использования одной схемы базы данных с другими сервисами, каждый микросервис имеет свою собственную базу данных. С одной стороны, этот подход противоречит идее модели данных для всего приложения. Кроме того, это часто приводит к дублированию данных. Однако наличие собственных, отдельных баз данных обеспечивает слабую связь.

На рисунке 1 представлена обобщённая схема приложения с микросервисной архитектурой.

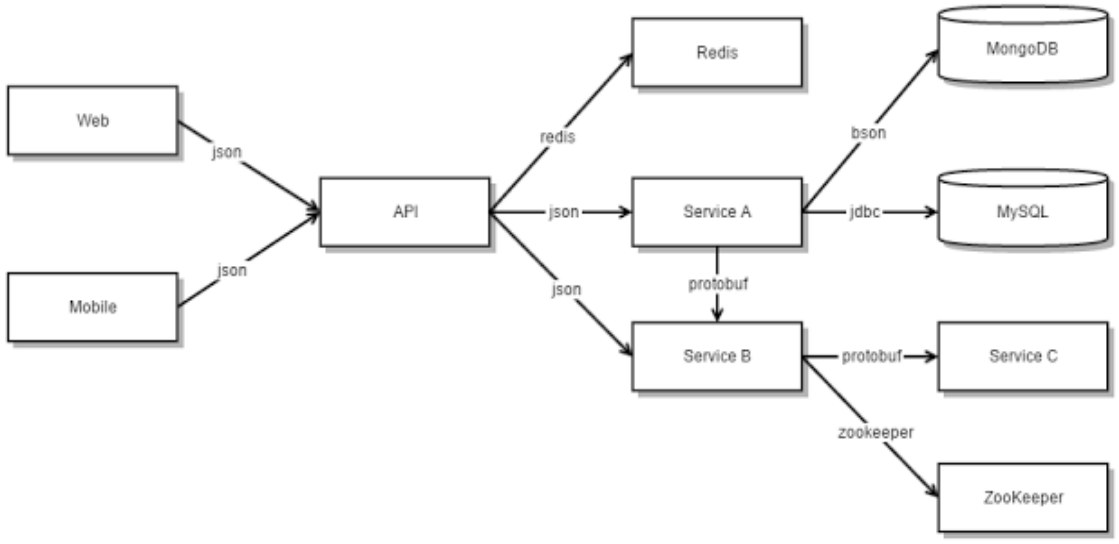


Рисунок 1 Обобщенная схема облачного приложения с микросервисной архитектурой

Конечные пользователи не имеют прямого доступа к внутренним сервисам. Вместо этого связь осуществляется через посредника, известного как шлюз. API-шлюз отвечает за такие задачи, как балансировка нагрузки, кэширование, контроль доступа и мониторинг [5].

1.1.3 Предметная область приложения

Разрабатываемая система мониторинга предназначается для облачного приложения врача-сомнолога. В виду коммерческой тайны и аспектов безопасности пользователей, технические детали и название не раскрывается.

Приложение позволяет доктору следить за стоянием пациентов с расстройствами сна дистанционно. Каждый пациент имеет специальное устройство, которое регистрирует плисомнографические сигналы во время сна. Затем сигналы поступают в систему через интернет, подвергаются анализу и сохраняются, распределяясь по различным микросервисам системы.

Помимо сложного технологического потока обработки медицинских данных, в приложение присутствуют стандартные элементы и функции такие как, поиск, пагинация, фильтры, авторизация и т.д. Эти функции тоже должны подвергаться мониторингу, для выявления востребованности определенных характеристик и оценки нагрузки на приложение в целом.

1.1.4 Система логов облачного приложения

Каждый микросервис записывает лог-информацию в отдельный файл, что в случае с ручной обработкой логов лишь усложняет процесс.

Однако для быстрого реагирования на чрезвычайные ошибки и мониторинг лог – файлов в приложение используется ELK – стек технологий, состоящий из Elasticsearch, Logstash и Kibana [6].

Менеджеры по качеству следят за всплесками ошибок в системе после очередного релиза и передают информацию об этом в отдел разработки. Те, в свою очередь, находят корреляции с ошибками в приложении, сразу видят стеки вызовов функций и прочие детали, необходимые для отладки. На рисунке 2 представлен пример лог записей системы.

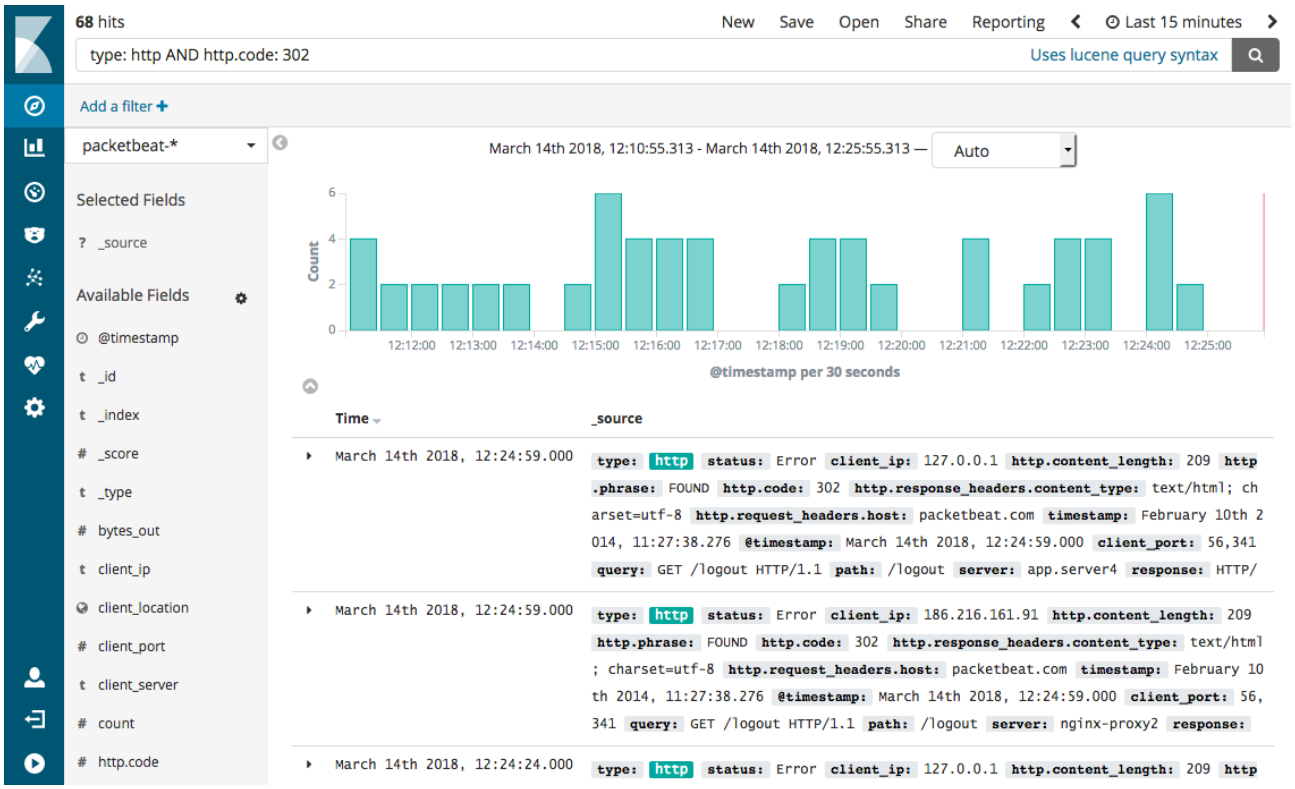


Рисунок 2 Пример логов в сервисе Kibana

1.2 Цели мониторинга

1.2.1 Бизнес-метрики

Данный блок предназначен для отображения фактической информации облачного приложения. Например, количество зарегистрированных пользователей, авторизаций в системе, востребованность того или иного фильтра

при поиске, как и когда была получена статистика с приборов в день и остальные, специфичные для предметной области, показатели.

Конечная цель этого набора метрик необходима для лучшего понимания пользователей системы, чтобы в последующем модернизировать тот или иной бизнес-процесс. То есть бизнес метрики дают информацию об использование конкретных функций приложения, например, авторизация с помощью второго фактора, генерация отчетов и т.д.

Если какая-либо функция пользуется успехом, то стоит задуматься о том, как сделать ее лучше и легче в плане использования. И соответственно наоборот, если функция мало востребована, то не стоит продолжать последующую детальную разработку, а остановится на текущем состоянии.

Помимо этого, бизнес-метрики говорят не только о поведение конечных пользователей, но и характеризуют количественные свойства приложения [7]. К примеру, если мы знаем, что у нас в системе зарегистрировано большое число организаций, то при реализации новой функции стоит задуматься об архитектуре, не добавит ли это дополнительной нагрузки на память во время работы приложения и стоит ли оптимизировать существующие участки кода.

1.2.2 Метрики аппаратных ресурсов

Мониторинг серверов, как физических, так и виртуальных показывает то, насколько корректно и правильно приложение было реализовано с точки зрения памяти, процессорного времени и т.д. Если в приложение есть узкие места, то именно эта группа метрик даст незамедлительный ответ о такого рода проблеме [7]. К примеру, при поиске пациентов может произойти ситуация, когда код выполняется не на стороне СУБД, а весь массив данный загружается в память и в последующем осуществляется работа в оперативной памяти. Такой подход мгновенно поглощает всю доступную память на виртуальной или физической машине и приложение замедляет свою работу, а иногда и вовсе завершается аварийно.

Поэтому чтобы понимать систему с точки зрения ресурсов, необходимо следить за каждым сервисом в отдельности, сколько ресурсов каждого ресурса

он потребляет и при превышение определенного порогового значения оповестить об этом разработчиков.

1.2.3 Жизнеспособность и доступность сервисов (health monitoring)

В случае с микросервисной архитектурой возможны такие исходы событий, когда приложение может частично не работать, потому что каждый микросервис запущен отдельным процессом, то есть набор каких-либо функций может быть недоступен.

Рассмотрим пример. Предположим, что сервис генерации отчетов перестал функционировать. Пользователь может открыть приложение, искать, создавать, редактировать пациентов, изменять различные настройки системы, но когда он захочет сгенерировать отчет, то в лучшем случае он получит уведомление, что не удалось сгенерировать отчет, или же вовсе визуально ничего не произойдёт.

Чтобы избежать таких проблем достаточно периодически опрашивать каждый сервис через API. В случае успешного ответа – сервис доступен и работоспособен, иначе сервис не запущен [8]. И причин может быть несколько: неудачный процесс развертывания приложения после очередного релиза, критическое завершение, сервер, на котором располагается сервис, физически недоступен и т.д. Ответ на этот вопрос могут дать логи приложения. Однако, чтобы это понять раньше пользователей системы, стоит отображать такую информацию или даже присылать уведомление разработчикам. Примером успешного мониторинга доступности сервисов является компания Майкрософт и ее приложение Azure DevOps. Для каждого сегмента приложения они отображают доступность своих сервисов в открытом доступе. Пример переставлен на рисунке 3.

1.3 Текущее состояние мониторинга системы

Конечно без какого-либо мониторинга невозможно поддерживать облачное приложение. Поэтому приложение, для которого проектируется система мониторинга в данной диссертации, имело панель доступности сервисов.

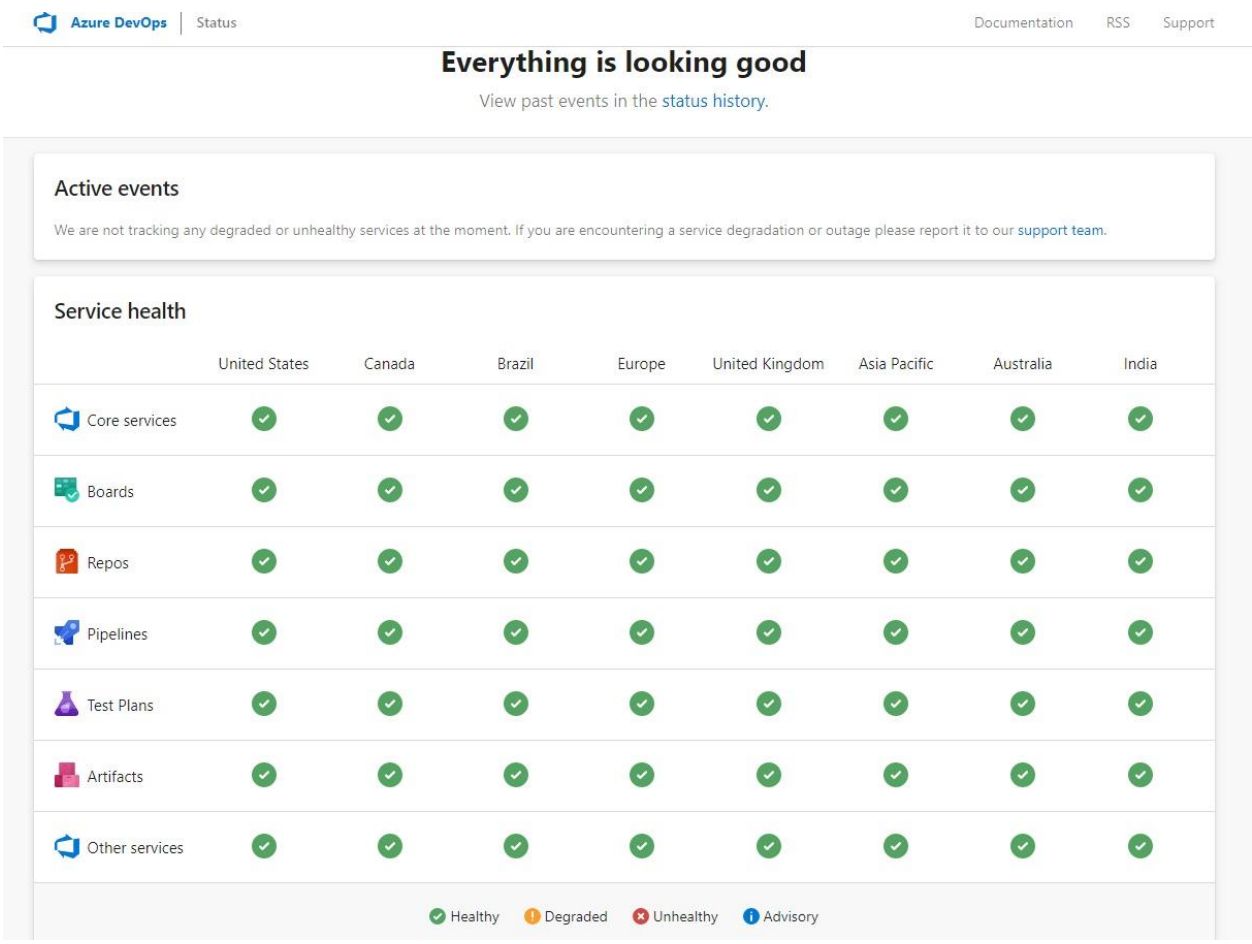


Рисунок 3 Панель доступности сервисов Azure DevOps

Что касается бизнес-метрик и аналитики аппаратных ресурсов – то мониторинг осуществлялся системными администраторами по запросу и с помощью ручного доступа. Что говорит о неудобстве и небезопасности данного подхода. Более того монитор доступности имел проблему с переполнением кэша и зависанием, что выражалось отображением неактуальной информации.

Однако система логов настроена и функционирует стабильно с использование ELK – стека.

ELK включает в себя такие компоненты как:

- Elastic Search. NoSQL-хранилище и поисковая система, основанная на языке Lucene;
- Logstash. Место импорта, с огромным количеством конфигураций, через которое данные попадают в хранилище Elastic Search;
- Kibana. Веб-интерфейс визуализации данных из Elastic Search.

Добавление логов в Logstash происходит через дополнительную утилиту из набора ELK – logstash-forwarder (beats). Данный инструмент является отдельной службой, которая следит за изменениями в файле на диске и добавляет их в Logstash. После этого каждая строчка файла распознается и отправляется в виде полей для индексирования и тегов в Elastic Search [6]. Результат такого преобразования можно посмотреть на рисунке 4.

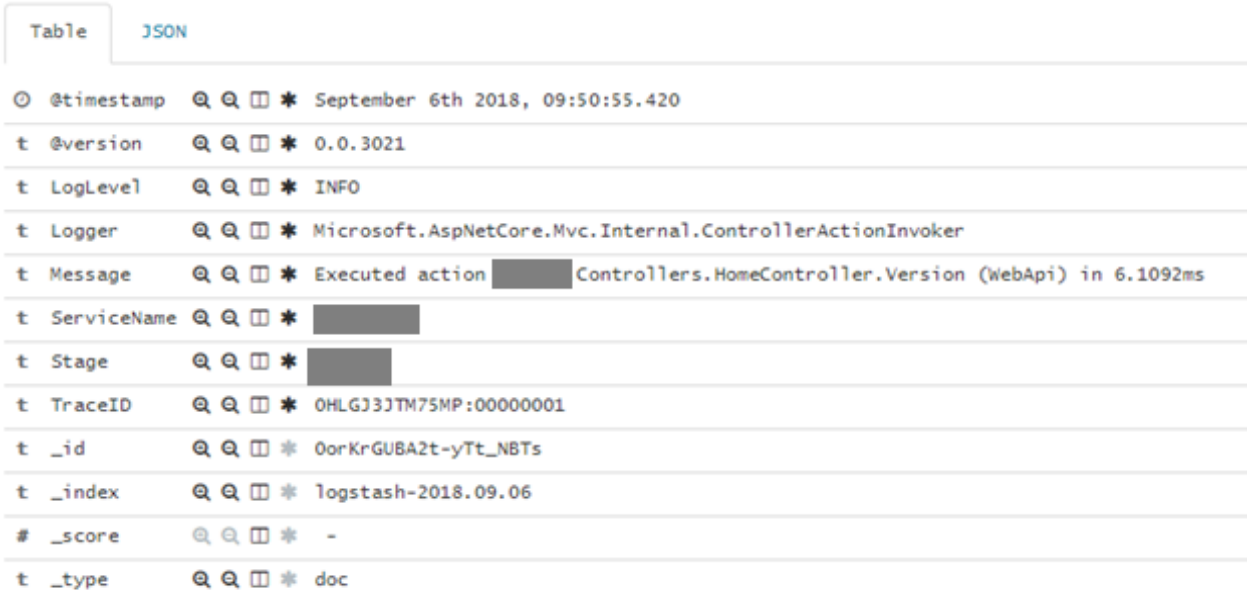


Рисунок 4 Документ в Elastic search

Глава 2. Проектирование

Во-первых, определимся с критериями мониторинга, чтобы выделить характерные черты, позволяющие определить набор инструментов.

Система мониторинга и визуализация данных должна соответствовать следующим требованиям:

- работать в режиме реального времени;
- быть гибкой в настройке новых метрик;
- использовать текущие лог – файлы (возможность подключения/импорта данных из Logstash, ElasticSearch, filebeat, kibana);
- масштабироваться на несколько серверов;
- стабильно работать;
- уведомлять об аномалиях в мессенджер компании;
- иметь возможность мониторинга Docker окружения и системных ресурсов;
- иметь Time Series базу данных;
- иметь возможность написания собственных плагинов на одном из знакомых языков (javascript, C#, F#);
- быть бесплатной и с открытым исходным кодом.

Метрики должны отображать:

- мониторинг аппаратных ресурсов;
- количество пользователей в системе;
- количество пациентов в системе;
- количество организаций в системе;
- количество устройств (возможно разделение на терапевтические, вентиляционные);
- попытки авторизации;
- общую статистику обращения к системе;
- отображать работоспособность сервисов;

– актуальное состояние сервера непрерывной интеграции и результатов развертывания сервисов.

2.1 Выбор инструментов для мониторинга

Были рассмотрены следующие инструменты для средств мониторинга: Cacti, Ganglia, Collectd, Graphite, Zabbix, Nagios, Icinga.

Вышеперечисленные компоненты достойны уважения, благодаря легкой установке и поддержке windows и linux – контейнеров, огромному проценту использования в реальных приложениях на многих серверах и компаниях. Но все они являются устаревшими, хотя все еще поддерживаются, но очень слабо развиваются и имеют устаревший интерфейс.

В большинстве из них используются sql-базы данных, что является не оптимальным для хранения исторических данных (метрик). Это кажется универсальным, но с другой стороны — таки базы данных создают большую нагрузку на диски. При этом размерность данных велика. Для таких задач больше подходят современные базы данных временных рядов, такие как ClickHouse [8].

Системы мониторинга нового поколения используют базы данных временных рядов, одни из них включают их в свой состав как неотделимую часть, другие используют как отдельный компонент, а третьи могут работать и вовсе без баз данных [9].

К последней группе, современных средств мониторинга, следует отнести такие продукты как NetData, Prometheus, InfluxDb, Grafana, Telegraph.

Несмотря на то что в компании уже используется ELK – стек, который позволяет настроить визуализацию в сервисе Kibana, выбор был сделан в пользу связки nodex_exporter, Prometheus, Grafana.

Kibana, входящая в стек ELK, хорошо позволяет искать детальную информацию по лог – файлам, визуализировать данные за последние несколько дней, но при реализации небольшой панели метрик, с агрегацией – система стала ощутимо медленней работать. Поэтому решили выделить отдельную инфраструктуру под визуальный мониторинг в виде Grafana и Prometheus.

2.1.1 Prometheus

Prometheus – это комплексное решение, включающие в себя фреймворк для мониторинга и собственную темпоральную база данных. Это достаточно молодой продукт, первый релиз которого состоялся в 2016 году.

Архитектура. Prometheus включает в себя набор следующих компонентов:

– Сервер. Функция которого это получение и сохранение метрик в темпоральной (time series) базе данных;

– Набор клиентских библиотек для различных языков программирования (Haskell, Java, Python, Go, Ruby, Node.js, .NET/C#);

– Pushgateway — компонент для приёма метрик кратковременных процессов;

– PROMDASH — дашборд для метрик;

– инструменты для экспорта данных из сторонних приложений (Statsd, Ganglia, HAProxy и других);

– менеджер уведомлений;

– клиент командной строки для выполнения запросов к данным.

Все компоненты Prometheus взаимодействуют между собой по протоколу HTTP. Схема коммуникации изображена на рисунке 5.

Ядром системы является сервер Prometheus. Его работа осуществляется автономно. Присутствует локальная база данных для хранения. Обнаружение остальных сервисов происходит автоматически, что существенно упрощает процедуру конфигурирования и развёртывания: для наблюдения за одним сервисом достаточно установить только сервер и необходимые компоненты для сбора и экспорта метрик. Существует набор уже готовых компонентов под мониторинг существующих сервисов таких, как: Haproxy, Docker, PostrgreSQL и т.д.

Есть два способа получения метрик в Prometheus – push и pull. Названия говорят сами за себя. Pull – сервер Pometheus делает запрос к сервису, то есть

Рисунок 5 Схема компонентов Prometheus

подтягивает данные. Однако, предусмотрена и другая возможность получения механизм push. Для такого подхода необходим специально предназначенный компонент – pushgateway. Это необходимо в тех случаях, когда получение метрик методом pull по каким-либо причинам невозможно. Например, когда сервис защищен фаерволом или находится в частной закрытой сети. Также механизм push может оказаться полезным при наблюдении за сервисами, подключающихся к сети периодически и на непродолжительное время, данные будут поступать по мере поступления на сервер автоматически [10].

Модель данных Prometheus требует от поступающих метрик представления в виде временных рядов (time series). Поэтому все метрики хранятся в собственной темпоральной БД. Для хранения индексов используется LevelDB.

2.1.2 Grafana

Grafana представляет собой открытый (Apache 2.0) веб-интерфейс к различным темпоральным СУБД, таким, как Graphite, InfluxDB, и, само собой разумеется, Prometheus. В общем, Grafana строит графики, используя информацию из Prometheus либо из Elastic Search [11]. Несмотря на то, что у

Prometheus есть собственный веб-интерфейс, который весьма минималистичен и довольно неудобен, используется визуализация с помощью Grafana.

Термины Grafana. Панель — базовый элемент визуализации выбранных показателей. Grafana поддерживает панели с графиками, единичными статусами, таблицами, тепловыми картами кликов и произвольным текстом, а также интеграцию с официальными и созданными сообществом плагинами (например, карта мира или часы) и приложениями, которые также можно визуализировать. Можно настроить стиль и формат каждой панели; все панели можно перетаскивать на новое место, перестраивать и изменять их размер.

Дашборд — набор отдельных панелей, размещенных в сетке с набором переменных (например, имя сервера, приложения и датчика). Изменяя переменные, можно переключать данные, отображаемые на дашборде (например, данные с двух отдельных серверов). Все дашборды можно настраивать, а также секционировать и фрагментировать представленные в них данные в соответствии с потребностями пользователя. В проекте Grafana участвует большое сообщество разработчиков кода и пользователей, поэтому существует большой выбор готовых дашбордов для разных типов данных и источников.

В дашбордах можно использовать аннотации для отображения определенных событий на разных панелях. Аннотации добавляются настраиваемыми запросами в Elasticsearch; на графике аннотация отображается вертикальной красной линией. При наведении курсора на аннотацию можно получить описание события и теги, например, для отслеживания ответа сервера с кодом ошибки 5xx или перезапуска системы. Благодаря этому можно легко сопоставить время, конкретное событие и его последствия в приложении и исследовать поведение системы.

2.2 Архитектура

Архитектура системы мониторинга, благодаря поддержке контейнеров позволяет разместиться на одной виртуальной машине. Это является огромным

плюсом и экономит ресурсы, как физические, так и материальные. Более того, при необходимости и возможности можно разместить на одной виртуальной машине как само приложение, так и систему мониторинга. Так как это только этап проектирования, то связи являются условными. Вполне вероятно, что в ходе реализации добавятся новые компоненты или виды связей заменятся по необходимости.

Из ранее описанных элементов системы мониторинга новым является компонент cadvisor. Он необходим для описания состояния docker – контейнеров [10].

На рисунке 6 представлена схема взаимодействия компонентов системы мониторинга между друг другом и внешними элементами, такими как сервер непрерывной интеграции и облачное приложение.

Архитектура приложения получилась довольно экономичной и компактной, несмотря на огромное количество компонентов. Однако каждый из них необходим и выполняет конкретную задачу.

Так как приложение содержит персональные данные, то доступ ограничен авторизацией. Тип используемой авторизации basic auth, то есть авторизация происходит по логину и паролю.

Справедливо было и закрыть доступ к мониторингу тоже посредством авторизации, несмотря на то, что система будет находится в частной закрытой сети – авторизация по паролю не будет лишней, а даже необходимой мерой защиты коммерческих и персональных данных.

Еще один аспект, касающийся проектирования архитектуры – это выделение системы мониторинга на отдельную персональную машину. Это является важным критерием для данного приложения. Все дело в том, что приложение является медицинским продуктом, то есть имеет влияние на процесс лечения пациента. Поэтому по регламенту хостинг приложений такого вида осуществляется на специальных площадках, доступ к которым ограничен.

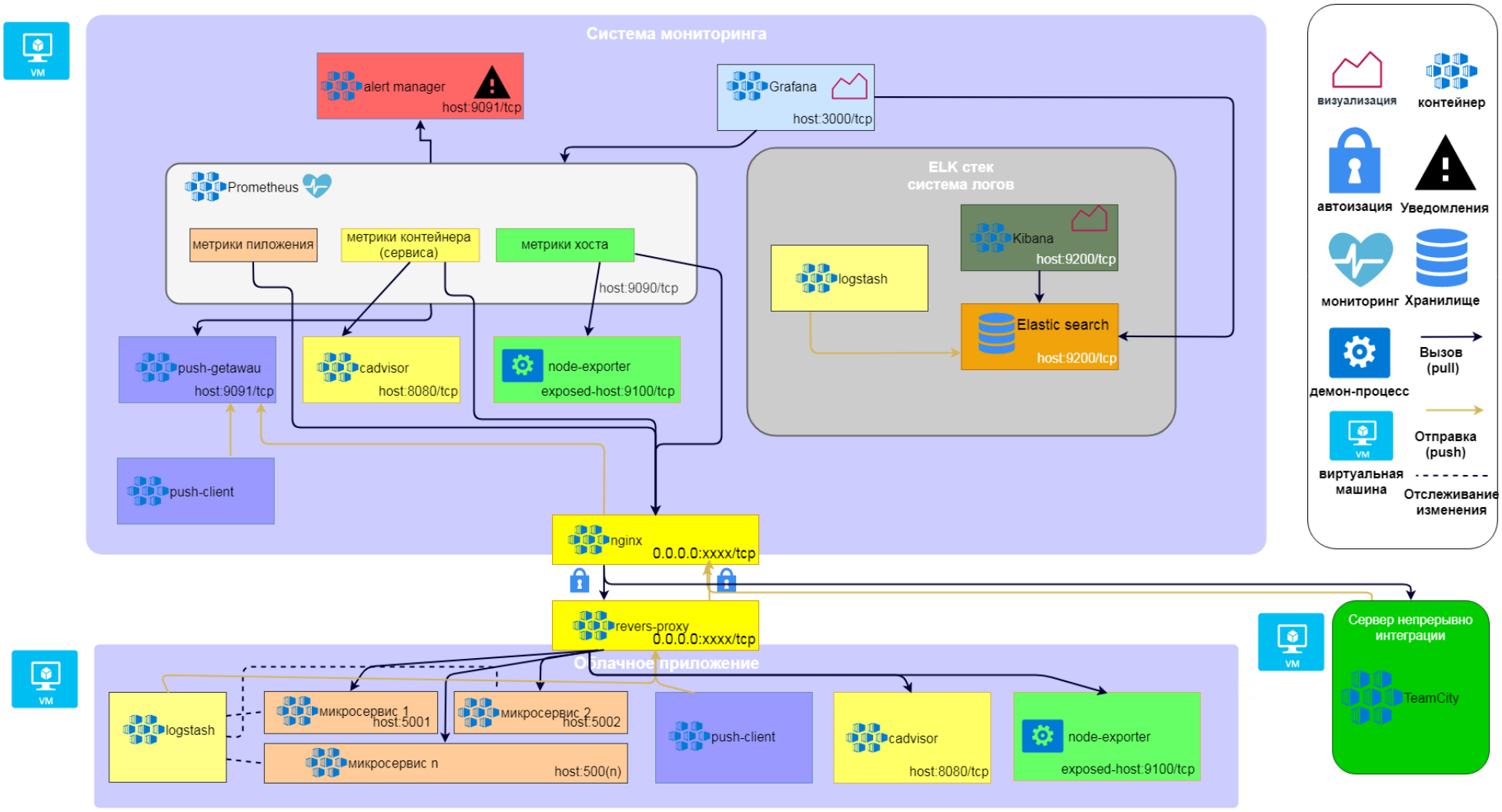


Рисунок 6 Схема системы мониторинга и компонентов

Отсюда вытекает проблема мониторинга большей части приложения – сервисов, находящихся за фаерволом. Такую проблему позволит решить комбинация методов доставки метрик в Prometheus – push и pull [10, 11].

2.3 Конфигурация и установка

2.3.1 Конфигурация Prometheus

Воспользуемся средствами Docker и запустим, как и планировали, согласно архитектуре, Prometheus в контейнере. Для этого воспользуемся базовым образом и соберем собственный контейнер на основе yml – файла конфигурации. Неполное содержимое yml – файла представлено в приложении 2.

После это мы можем собрать собственный образ с помощью команды:

```
docker run -t <image name> .
```

А затем запустить контейнер с помощью docker-compose.yml [12]. Содержимое docker-compose представлено в приложении 4.

Чтобы проверить результат достаточно открыть веб интерфейс Prometheus. Если в спеске сервисов отображаются сконфигурированные в yml файле сервисы, то все сделано корректно. Пример проиллюстрирован на рисунке 7.

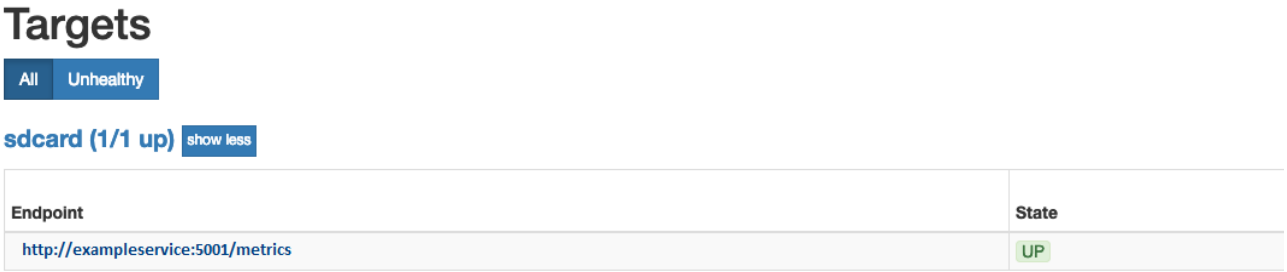


Рисунок 7 веб интерфейс Prometheus

Остальные компоненты добавляются путем редактирования файла конфигурации. Их конфигурация не будет отображена в данной работе, в виду того, что содержит много персональной и коммерческой информации.

2.3.2 Настройка Grafana

Снова воспользуемся публичным образом для Docker контейнера и запустим сервис визуализации метрик Grafana. Для этого в уже имеющийся конфигурационный файл добавим еще одну секцию следующего содержания:

```
graphana:
    environment:
      - GF_INSTALL_PLUGINS=grafana-piechart-panel
    image: grafana/grafana
    ports:
      - "3000:3000"
```

Измененный конфигурационный файл представлен в приложении 3.

Первый раз система попросит авторизоваться с дефолтной учетной записью admin/admin. После первого успешного входе следует сменить пароль и добавить новых пользователей.

Все что необходимо для настройки Grafana – это всего лишь задать источник данных [13]. Дальнейшая работа заключается в выборке значений, аналогично sql запросам, только используя другой язык - lucene. Это можно сделать через web-интерфейс в режиме администрирования, как показано на рисунке 8.

2.3.3 Добавление панелей визуализации

Если оперировать терминами grafana, то для визуализации необходимо добавить дашбоард [11]. Боле того, можно воспользоваться уже готовыми решениями, лишь немного подкорректировав их под свои нужды. На панель визуализации можно добавить различные виды графической информации: диаграммы, индикаторы нагрузки, текст, таблицы, лист метрик, лист предупреждений.

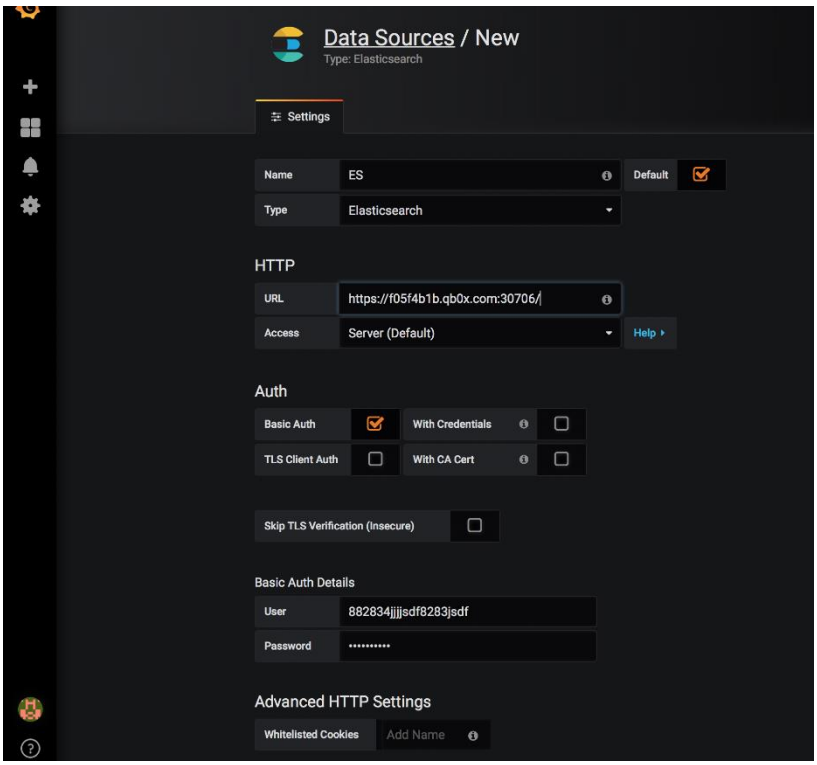


Рисунок 8 Добавление источника данных в grafana

Кроме того, можно написать собственный плагин. Поддерживаемые языки – javascript, причем даже с поддержкой typescript, и html, css. Доступные типы визуализации продемонстрированы на рисунке 9.

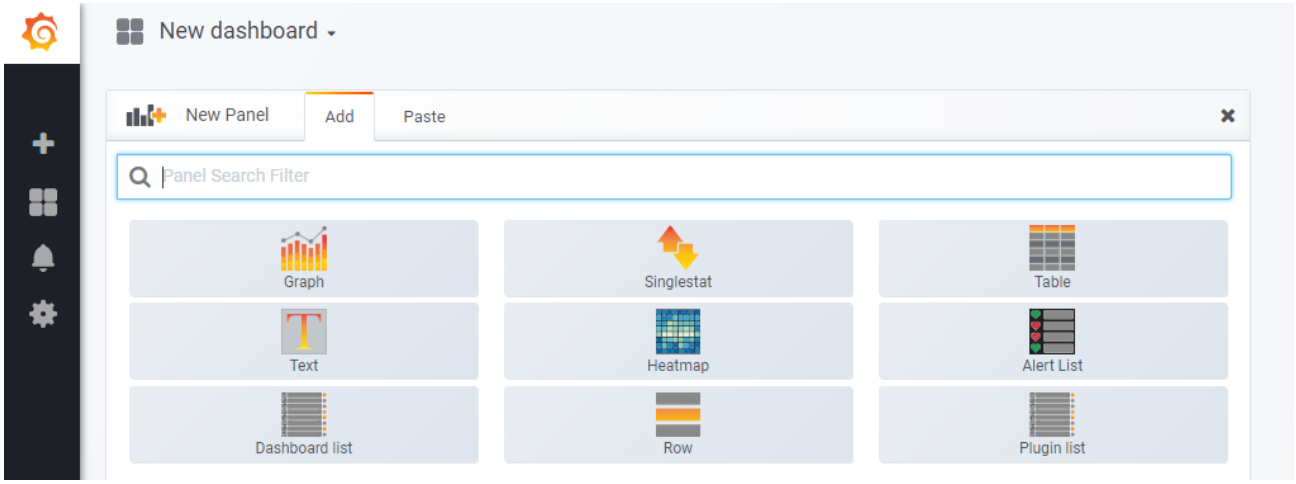


Рисунок 9 Добавление новой панели

Создадим простой график, отображающий количество авторизаций с использованием второго фактора.

Запросы можно отлаживать в встроенном редакторе запросов.

Результат такой отладки показан на рисунке 10.

Рисунок 10 Построение метрики в Grafana

Воспользуемся готовым шаблоном, для отображения аппаратных ресурсов сервера из библиотеки dashboard’ов с сайта Grafana Labs [13].

Результат представлен на рисунке 11.

Рисунок 11 Мониторинг аппаратный ресурсов

Добавим некоторые количественные метрики, такие как количество пользователей системы, количество пациентов, организаций, среднее время запросов для каждого сервиса и сервера, чтобы оценить процесс добавления количественных метрик. А также добавим критические значения, такие как количество долгих запросов, чтобы знать о вероятных задержках системы и общее количество совершенных запросов к системе. Результат представлен на рисунке 12.

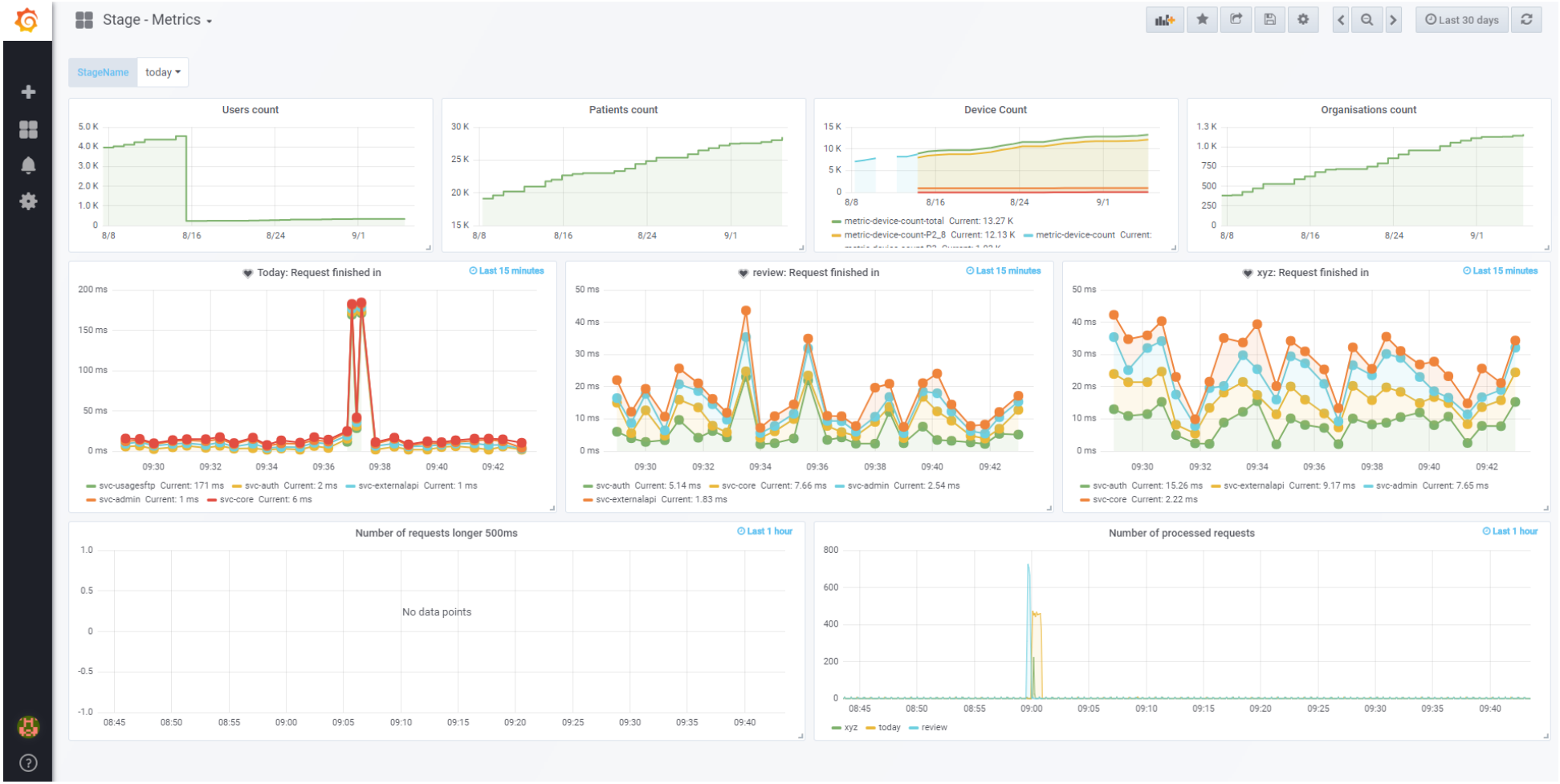


Рисунок 12 количественные метрики системы

Глава 3. Реализация программной части

Конфигурирование Prometheus и подготовка микросервисов к мониторингу займет некоторое время. Поэтому для интеграции сервера непрерывной интеграции и временного решения для health – мониторинга необходимо реализовать модуль экспорта данных в Grafana.

3.1 Анализ вариантов реализации

Для начала определим список задач.

1. Необходимо, каким-то образом, доставить данные в Grafana из TeamCity и состояние веб-сервисов, используя http-запрос service.baseAddress.domain/api/version.

2. После этого данные, по необходимости, преобразовать в timeseries объект с цифровым значением.

3. Визуализировать данные с помощью встроенных компонентов визуализации Grafana.

Проанализируем готовые решения для импорта и экспорта данных в Grafana.

3.1.1 Плагин, для TeamCity teamcity-graphite

Данный плагин позволяет отправлять данные в graphite в момент выполнения сборок в TeamCity [14].

Проработка решения:

3.1.1.1 Устанавливаем и конфигурируем graphite. Для этого воспользуемся docker-контейнером и развернем его одной командой:

```
docker run -d\
--name graphite\
 --restart=always\
 -p 80:80\
 -p 2003-2004:2003-2004\
 -p 2023-2024:2023-2024\
 -p 8125:8125/udp\
 -p 8126:8126\
 graphiteapp/graphite-statsd
```

3.1.1.2 Устанавливаем teamcity-graphite плагин в TeamCity. Результат на рисунке 13.

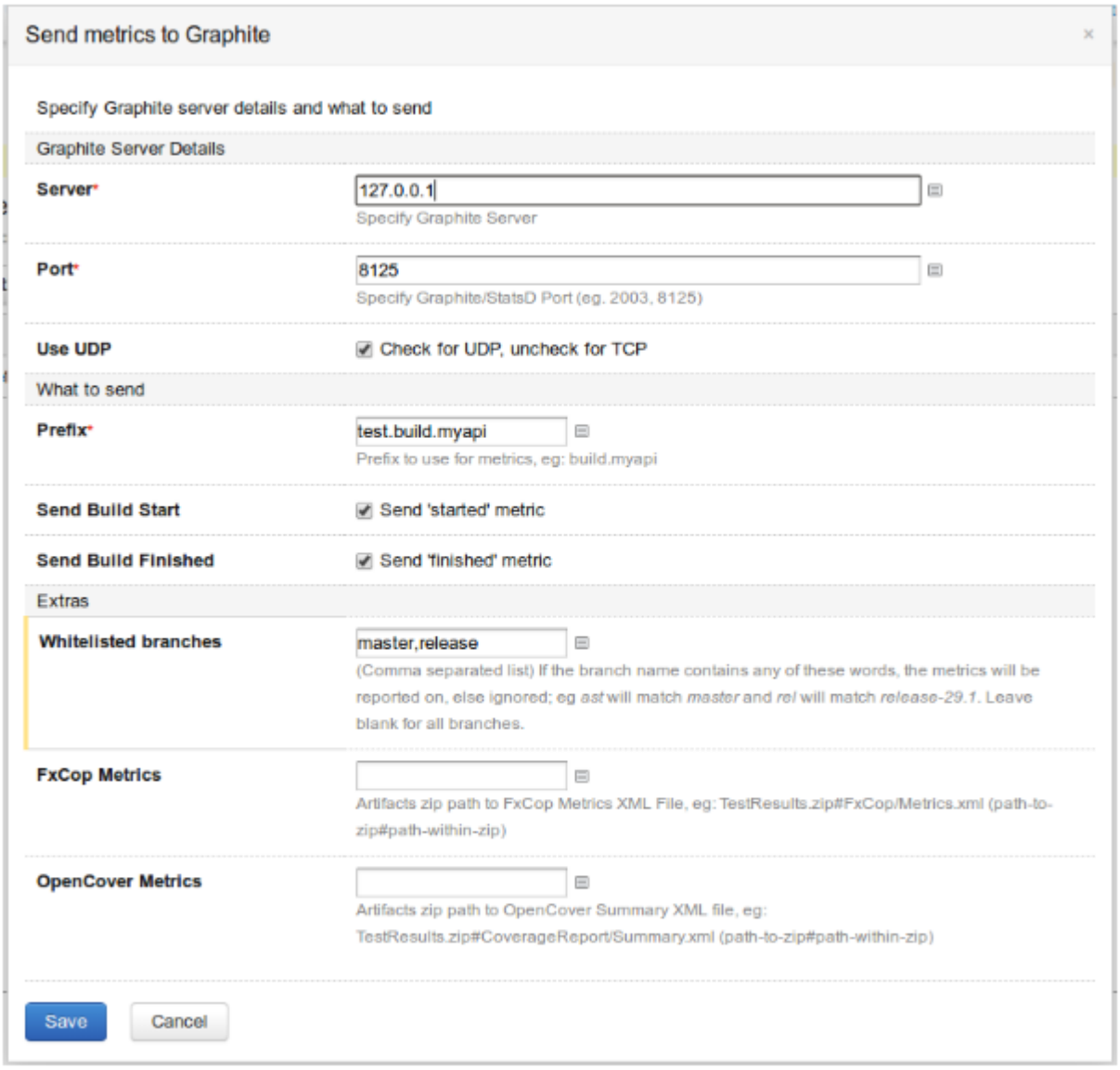


Рисунок 13 Конфигурация плагина в TeamCity

3.1.1.3 Конфигурируем плагин для отправки данных в графит.

3.1.1.4 Настраиваем в Grafana источник данных для Graphite сервера. Результат представлен на рисунке 14.

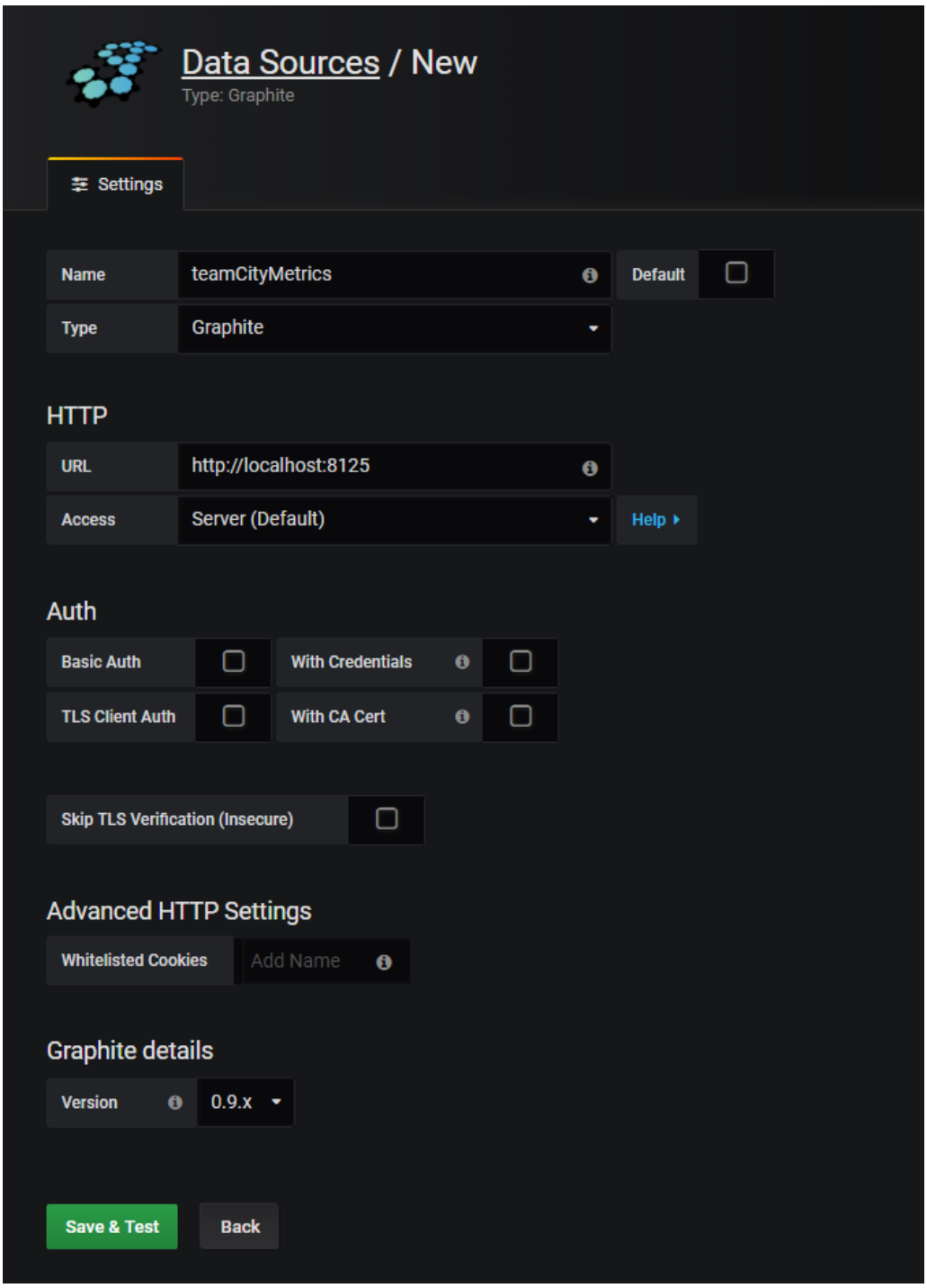


Рисунок 14 Конфигурация источника данных в Grafana

3.1.1.5 Визуализируем полученные метрики.

К сожалению, данное решение не работает, по неизвестной причине данные из Teamcity не отправляются в сервер Graphite. Варианты с доставкой по UDP тоже не увенчались успехом. **Результат**: решение не работает.

Посмотрим на данные с другой стороны. Если не удалось их отправить, то возможно, получится их запросить.

3.1.2 Использование плагина teamcity-datasource для Grafana.

Проработка решение:

3.1.2.1 Установка плагина в Grafana.

3.1.2.2 Конфигурация плагина, подключение teamcity-сервера.

3.1.2.3 Визуализация полученных данных.

На рисунке 15 можно увидеть, что успешно получен список действующих сборок.

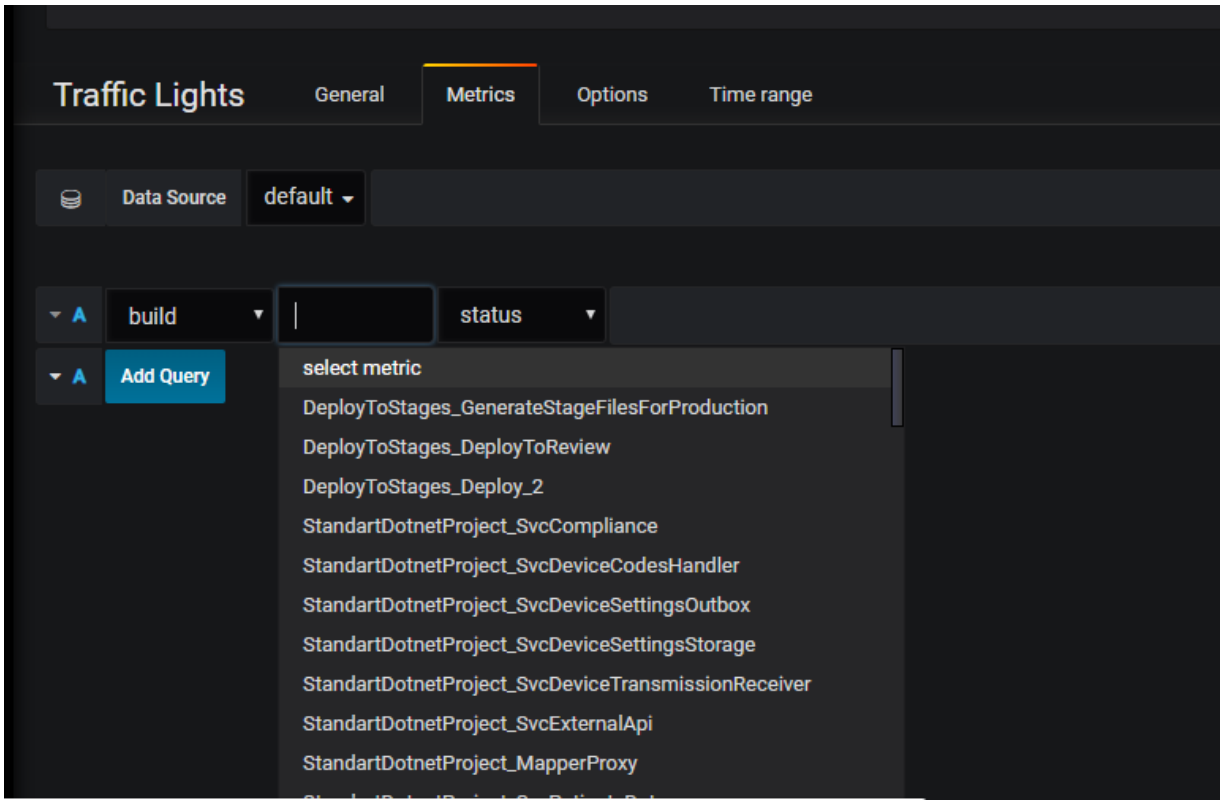


Рисунок 15 Демонстрация работы плагина

Результат: решение работает, но есть проблема. Данный плагин работает по API с использованием времени. То есть если у нас в Grafana выбранный период - последние 6 часов, а сборка прошла 2 дня назад, то мы не получим никаких данных. При этом ответ из плагина приходит строкой, что не позволяет легко визуализировать что-либо. Плагин не удовлетворяет условиям системы.

3.2 Анализ использованных решений

Вариант с Graphite, был хорош, потому что Graphite может использовать несколько сторонних источников данных, в частности один из продуктов,

который так же используется — AppMetrics[15]. Для отображение внутреннего состояние сервисов.

Вариант с плагином для Grafana понравился больше, потому что были видны результаты работоспособности данного подхода, но отображение сборок по временному интервалу абсолютно бесполезно, т. к. релиз в приложения происходит раз в 2-3 месяца и получить информацию корректно для основного приложения не получится.

3.3 Генерация нового решения

Остаётся единственно верное решение – написать свой плагин.

Только существует один вопрос – для какой системы писать плагин, для TeamCity с использованием Graphite, или для Grafana плагин — источник данных?

После анализа и оценки собственных навыков было принято решение писать плагин — источник данных для Grafana, ввиду того, что:

1. Язык реализации javaScript, в отличие от Java. (Навыков и опыта больше).

2. Возможно, повторное использование части кода, для написания плагина – источника данных для опроса веб-сервисов через Http.

3. В силу того, что навыков в js больше, то это существенно ускорит процесс разработки.

4. Документация для Grafana четко и структурированно описана.

5. Огромное количество примеров таких плагинов.

После скачивания шаблона garafana-datasource-typescript[16] был написан плагин, использующий TeamCity API.

Возможности плагина:

– запрос списка всех билдов, доступных на сервере авторизированному юзеру.

– Отображение информации по каждому билду, такой как: статус (числовое значение от 0 до 100), строковое представление статуса, номер билда, имя, имя проекта, и дата выполнения.

Результат построения запроса – представлен на рисунке 16.

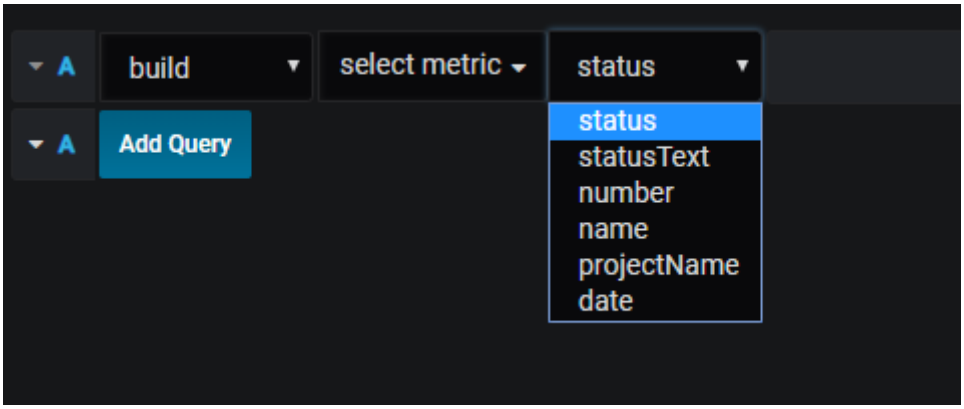


Рисунок 16 Построение запроса

Статус представляет из себя набор констант:

```
export const BuildStates: any = {
    SUCCESS: 100,
    PENDING_AND_SUCCESS: 65,
    PENDING: 50,
    PENDING_AND_FAILED: 35,
    QUEUED: 25,
    FAILED: 0
}
```

Описание работы плагина. Первоначально необходимо сконфигурировать источник данных, передав путь до сервера и имя учетной записи пользователя с паролем.

Вся коммуникация между плагином и конечным сервером происходит через внутренние функции Grafana, поэтому передача конфиденциальных данных защищена.

Когда пользователь сконфигурировал источник данных и пытается построить запрос, выполняется загрузка списка всех доступных сборок. При выборе конкретной сборки из списка становятся доступными метрики. Результат работы плагина — это значение определённой метрики для конкретных сборок.

Получение метрик в числовых константах, позволяет легко настроить панели визуализации в зависимости от значения.

Часть кода представлена в приложение 5. Весь проект доступен в свободном доступе в github-репозитории https://github.com/mav10/grafana-teamcity-datasource.

Аналогично был написан плагин для опроса сервисов по HTTP протоколу.

Описание работы плагина: пользователь конфигурирует источник данных, добавляя базовый адрес сервера, без дополнительных путей и сабдоменов. Пример представлен на рисунке 17.

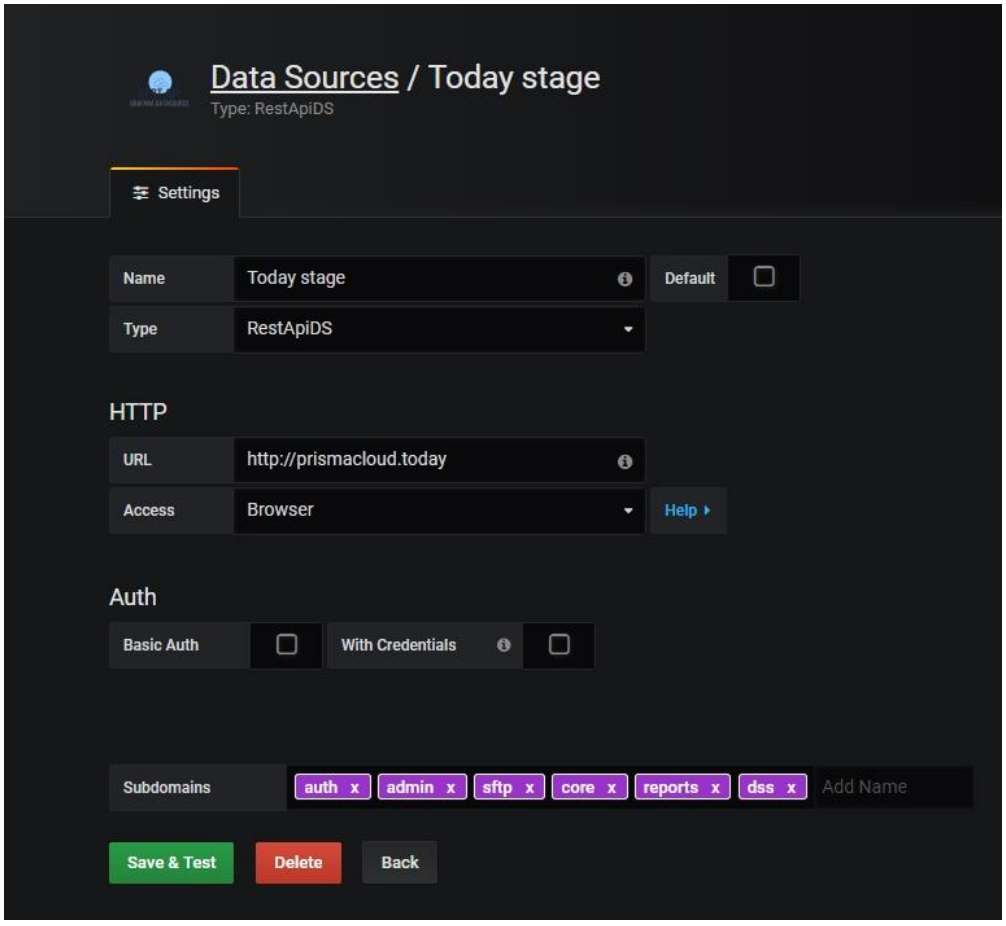


Рисунок 17 Конфигурация источника данных по REST-API

Плагин был написан таким образом, чтобы не создавать много источников данных под каждый сервис в отдельности. Для этого была реализована поддержка сабдоменов и конфигурация путей.

После конфигурации пользователь выбирает необходимый источник данных, как в примере на рисунке 10 – это Today Stage, затем, сабдомен и при необходимости добавляет дополнительный путь, например, /api/version.

Результат выполнения плагина — это числовой ответ, соответствующий HTTP-статус коду (200, 400, 404, 503…) а уже эти значения конвертируются в конечный результат.

Часть написанного кода доступна в приложении 6. Весь проект доступен в свободном доступе в github-репозитории https://github.com/mav10/grafana-restapi-datasource.

3.4 Добавление панелей визуализации

В соответствие с работой плагинов была сконфигурирована панель health-мониторинга и утилит. Как можно заметить на одном экране сразу доступна статистика с 3 разных серверов.

В Верхней части дашборда отражены важные сервисы и результат последнего развертывания. Так, например, на рисунке 18 для третьего стейджа результат неудачный, чему соответствует надпись и броский цвет.

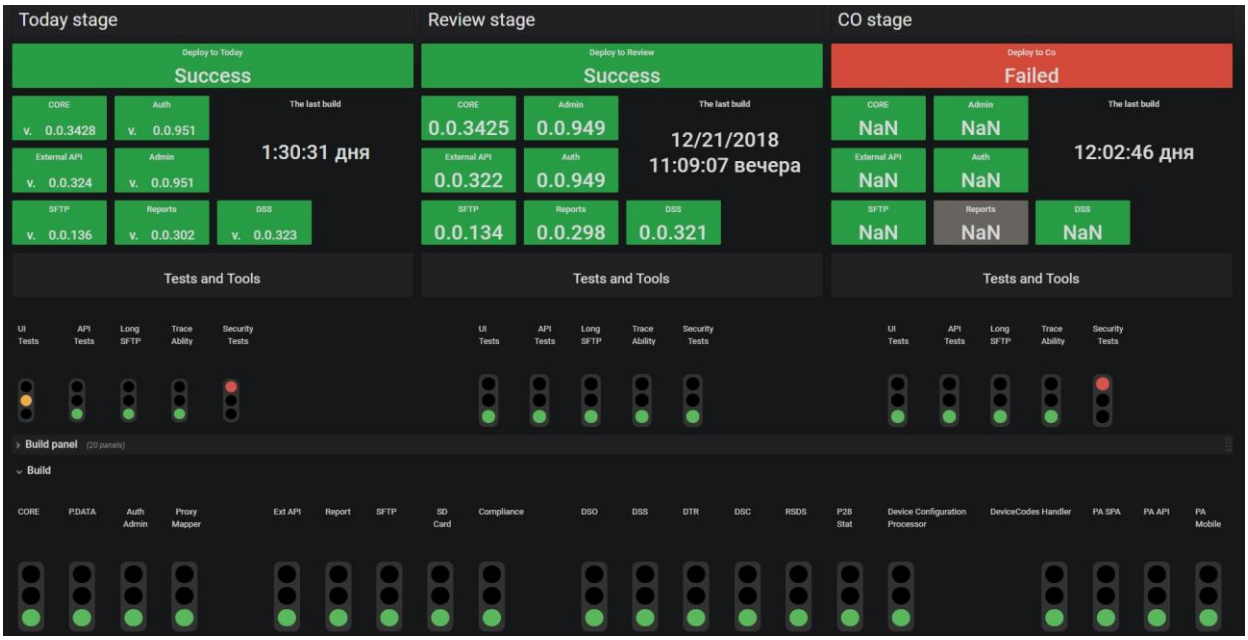


Рисунок 18 Панель визуализации с плохим итогом развертывания

В средней части экрана отражены утилиты. На конкретном примере ля каждой из реплик показаны UI, интеграционные тесты, тесты на безопасность.

Внизу располагаются результаты последней сборки для каждого из микросервисов.

В качестве элемента визуализации выбран графический элемент светофор, отражающий 3 различных состояния:

1. зеленый цвет – сборка завершена и прошла успешна;
2. желтый цвет – в данный момент происходит сборка сервиса или выполнение операции;
3. красный цвет – сборка завершилась с ошибкой.

Каждый из таких элементов кликабельный и по нему можно перейти в конкретный раздел сервера непрерывной интеграции.

Глава 4. Финансовый менеджмент, ресурсоэффективность и ресурсосбережение

В рамках данной диссертации была спроектирована и разработана система мониторинга для облачного приложения врача – сомнолога, предназначенная для отображения актуальной информации работы приложения. Мониторинг имеет панели визуализации для мгновенной аналитики состояния системы.

Перед тем, как представить продукт на рынке информационных систем, необходимо оценить данную разработку с точки зрения ее востребованности, а также ресурсоэффективности и ресурсосбережения. Для достижения данной цели решались следующие задачи:

Для достижения цели были поставлены следующие задачи:

- произвести QuaD анализ;
- построить причинно-следственную диаграмму Исикавы;
- определить цели проекта, его организационную структуру, а также ограничения и допущения проекта;
- спланировать структуру работ, их трудоемкость;
- сформировать бюджет проекта;
- определить потенциальные риски;
- определить финансовой эффективности.

Проектируемая система мониторинга разрабатывается под нужды конкретного облачного приложения с определённой спецификой и архитектурой.

4.1 Предпроектный анализ

4.1.1 Технология QuaD

Так как разрабатываемая система мониторинга уникальна и предназначена для конкретного приложения, то справедливо использовать технологию QuaD (QUality ADvisor) для измерения характеристик, описывающих качество новой разработки и ее перспективность на рынке.

Данный этап поможет принять решение о целесообразности вложения денежных средств в инженерный проект. Оценочная карта представлена в таблице 4.1.

Таблица 4.1 – QuaD анализ системы мониторинга для облачного приложения.

Критерии оценки	**Вес критерия**	**Баллы**	**Макси-мальный балл**	**Отно-сительное значение**	**Средне-взвешенное значение**
	1	2	3	4	5
Показатели оценки качества					
1. Ресурсопотребность (физическое или виртуальное оборудование)	0,005	70	100	0,7	0,0035
2. Потребность в ресурсах памяти	0,065	83	100	0,83	0,05395
3. Отражение количественных метрик приложения	0,091	100	100	1	0,091
4. Горизонтальное масштабирование (мульти – стеджинг)	0,061	100	100	1	0,061
5. Возможность кастомизации	0,077	90	100	0,9	0,0693
6. Простота добавление новой метрики	0,090	93	100	0,93	0,0837
7. Отражение ресурсных показателей	0,090	97	100	0,97	0,0873
8. Возможность отражения дополнительных утилит (UI и интеграционные тесты, тесы на безопасность, и т.д.)	0,092	100	100	1	0,092
9. Устойчивость системы мониторинга (продолжительность работы без сбоев)	0,096	98	100	0,98	0,09408
10. Отображение всех доступных метрик в одном месте	0,055	100	100	1	0,055

Продолжение таблицы 4.1

11. Обширная способность визуализации метрик	0,083	87	100	0,87	0,07221
Показатели оценки коммерческого потенциала разработки					
12. Конкурентоспособность продукта	0,001	50	100	0,5	0,0005
13. Уровень проникновения на рынок	0,015	20	100	0,2	0,003
14. Цена	0,135	50	100	0,5	0,0675
15. Обслуживание после ввода в эксплуатацию	0,025	50	100	0,5	0,0125
16. Финансовая эффективность разработки	0,015	60	100	0,6	0,009
17. Срок реализации	0,004	60	100	0,6	0,0024
Итого:	**1**	**1308**		**13,08**	**0,85794**

Как можно заметить, результат получился 85%, что говорит о перспективности разработки, потому что система мониторинга имеет сильные стороны в силу своей уникальности, так как разрабатывается под конкретное приложение. То есть реализация системы мониторинга окажет положительный эффект после ее внедрения. Но помимо этого можно наметить пути совершенствования в таких направлениях как финансовая эффективность, используемые аппаратные ресурсы и реализация панелей визуализации метрик в одном сервисе.

4.1.2 Диаграмма Исикавы

Диаграмма причины-следствия Исикавы (Cause-and-Effect-Diagram) – это графический метод анализа, формирования причинно-следственных связей между факторами и проблемами в исследуемой ситуации. Диаграмма представлена на рисунке 19.

Рисунок 19 Диаграмма Исикавы

В диаграмме Исикавы, представленной выше представлены основные проблемы, касающиеся четырех основных факторов: персонал, оборудование, методы, материалы. Стоит отметить, что сервис анализа и обработки документов на сегодняшний день не является чем-то популярным в силу того, что технологии в данной области еще не являются совершенными, поэтому обучение персонала и объяснение принципов систем подобного рода является нетривиальной задачей.

4.2 Инициация проекта

В процессе инициации проекта определяются начальные цели и содержание, а также фиксируются финансовые ресурсы. Определяются внутренние и внешние заинтересованные стороны проекта, которые будут взаимодействовать и влиять на общий результат научного проекта.

4.2.1 Цели и результат проекта

В данном разделе приведена информация о заинтересованных сторонах проекта, целях проекта и критериях их достижения. Информация по заинтересованным сторонам представлена в таблице 4.3.

Таблица 4.3 – Заинтересованные стороны проекта

Заинтересованные стороны	**Ожидание сторон**
Организация-заказчик, пользователь	Готовое система мониторинга облачного приложения с визуализацией достоверной и актуальной информации метрик облачного приложения
Научный руководитель, студент	Готовая магистерская диссертация

В таблице 4.4 представлена иерархия целей проекта и критерии достижения данных целей.

Таблица 4.4 – Цели и результат проекта

Цели проекта:	Разработка системы мониторинга для облачного приложения
Ожидаемые результаты:	Системы мониторинга облачного приложения для внутренних нужд организации-заказчика разработана и введена в эксплуатацию. Доступны панели визуализации, отражающие актуальное состояние приложения.
Критерии приемки результата проекта:	– Система мониторинга реализована и работает без сбоев – Метрики доступны в постоянно и обновляются на основе поступающей информации в приложение; – Настроены панели визуализации, которые можно трактовать однозначно; – Присутствует количественный мониторинг приложения; – Отображен мониторинг ресурсов; Отображен health мониторинг.
Требования к результату проекта:	1. Выполнены все пункты технического задания; Разработанный функционал полностью соответствует проектным решениям.

4.2.2 Организационная структура проекта

В ходе реализации проекта помимо магистра задействованы еще несколько человек. В данном разделе необходим определить команду проекта и роль каждого участника в проекте. В таблице 4.5 представлена информация о рабочей группе проекта.

Таблица 4.5 – Рабочая группа проекта

№	ФИО, место работы, должность	Роль в проекте	Функции
1	Васин Максим Алексеевич, ООО МЦЦ Томск, инженер-программист	Инженер	1. Проектирование 2. Реализация 3. Внедрение
2	Лунев Александр Алексеевич, ООО МЦЦ Томск, руководитель практики	Руководитель проекта от организации	1. Проверка разработки 2. Помощь во внедрении
3	Чердынцев Евгений Сергеевич, ТПУ, кандидат технических наук	Научный руководитель	1. Составление научных целей и задач 2. Проверка документации

4.2.3 Ограничения и допущения проекта

Ограничения проекта – это все факторы, которые могут послужить ограничением степени свободы участников команды проекта, а также «границы проекта» - параметры проекта или его продукта, которые не будут реализованы в рамках данного проекта. Ограничения проекта представлены в таблице 4.6.

Таблица 4.6 – Ограничения проекта

Фактор	Ограничения
Бюджет проекта	350 000 рублей
Источник финансирования	ООО МЦЦ Томск
Сроки проекта	09.01.2019-01.06.2019
Дата утверждения плана управления проектом	09.01.2019
Дата завершения проекта	01.06.2019

В данном разделе были рассмотрены цели и ожидаемый результат, организационная структура, а также ограничения и допущения проекта. К ограничениям относятся бюджет и сроки.

4.3 Планирование проекта

4.3.1 Структура работ в рамках проекта

В данном разделе производится декомпозиция задач и разграничение зоны ответственности между участниками проекта.

В таблице 4.7 представлено распределение исполнителей по видам работ.

Таблица 4.7 – Распределение задач

Основные этапы	**№**	**Содержание работ**	**Должность исполнителя**
Постановка задач	1	Описание требований	Руководитель от организации, научный руководитель
	2	Анализ предметной области	Руководитель от организации, научный руководитель, инженер
	3	Разработка технического задания	Руководитель от организации, инженер
Проектирование	4	Разработка архитектуры системы	Инженер
Разработка	5	Настройка и конфигурирование Grafana	Инженер
	6	Подключение количественных метрик на основе логов	Инженер
	7	Разработка плагина teamcity-data-importer для Grafana	Инженер
	8	Разработка плагина rest-api-importer для Grafana	Инженер
	9	Подключение Prometheus	Инженер
	10	Конфигурирование сервисов под prometheus	Инженер

Продолжение таблицы 4.7

	11	Настройка панелей визуализации	Инженер
Отладка решения и внедрение	12	Развертывание системы	Руководитель от организации, инженер
	13	Внедрение разработки	Руководитель от организации, инженер
Оформление документации	14	Написание отчетной документации	Инженер
	15	Проверка работы	Научный руководитель

4.3.2 Определение трудоемкости выполнения работ

Для определения ожидаемых сроков выполнения проекта необходимо оценить его трудоемкость. Воспользуемся формулой:

$$t_{\text{ож}i} = \frac{3 * t_{\min i} + 2 * t_{\max i}}{5}$$

где $t_{ожi}$ – ожидаемая трудоемкость выполнения i-ой работы чел.-дн.;

$t_{\min i}$ – минимально возможная трудоемкость выполнения заданной i-ой работы (оптимистическая оценка: в предположении наиболее благоприятного стечения обстоятельств), чел.-дн.;

$t_{\max i}$ – максимально возможная трудоемкость выполнения заданной i-ой работы (пессимистическая оценка: в предположении наиболее неблагоприятного стечения обстоятельств), чел.-дн.

Исходя из ожидаемой трудоемкости работ, определяется продолжительность каждой работы в рабочих днях Tpi, учитывающая параллельность выполнения работ несколькими исполнителями:

$$t_{pi} = \frac{t_{\text{ож}i}}{\text{Ч}_i}$$

где t_{pi} – продолжительность одной работы, раб.дн.,

$t_{ож}i$ – ожидаемая трудоемкость выполнения одной работы, чел.-дн,

$Ч_i$ – численность исполнителей, выполняющих одновременно одну и ту же работу на данном этапе, чел.

Для удобства составления календарного плана и графика работ необходимо перевести длительность каждого из этапов из рабочих дней в календарные дни. Для этого воспользуемся следующей формулой:

$$T_{кi} = T_{pi} * k_{кал}$$

где $t_{кi}$ – продолжительность выполнения i-й работы в календарных днях;

t_{pi} – продолжительность выполнения i-й работы в рабочих днях;

$k_{кал}$ –коэффициент календарности.

Коэффициент календарности определяется по следующей формуле:

$$Ткал = \frac{Т_{кал}}{Т_{кал} - Т_{вых} - Т_{пр}} = \frac{365}{365 - 66} = 1{,}22$$

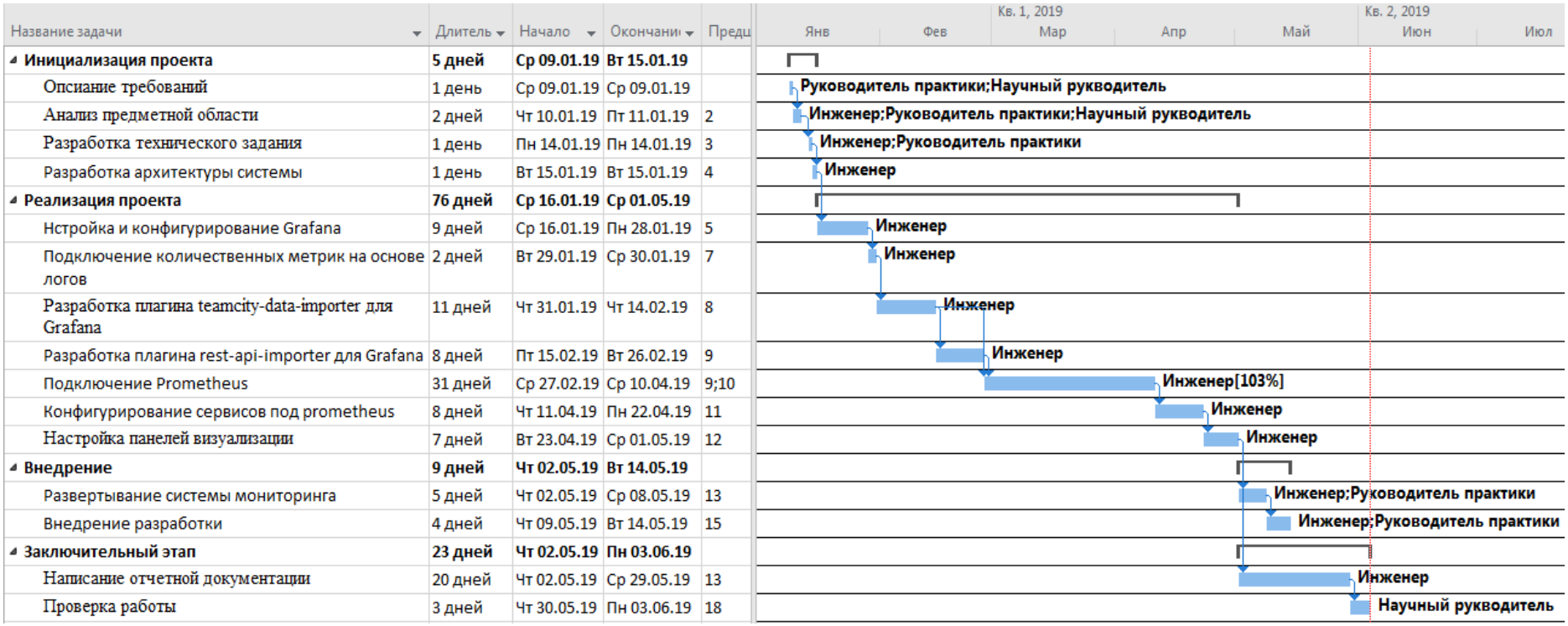
где: $t_{кал}$ – количество календарных дней в году;

$t_{вых}$ – количество выходных дней в году;

$t_{пр}$ – количество праздничных дней в году.

В соответствии с производственным календарем (для 6-дневной рабочей недели) в 2019 году 365 календарных дней, 299 рабочих дней, 66 выходных/праздничных дней.

Для иллюстрирования календарного плана-графика используется диаграмма Ганта, где представлено наглядное отображение графика и распределения работ между участниками проекта. Диаграмма представлена на рисунке 20.



Название задачи	Длитель	Начало	Окончани	Предш
Инициализация проекта	**5 дней**	**Ср 09.01.19**	**Вт 15.01.19**	
Описание требований	1 день	Ср 09.01.19	Ср 09.01.19	
Анализ предметной области	2 дней	Чт 10.01.19	Пт 11.01.19	2
Разработка технического задания	1 день	Пн 14.01.19	Пн 14.01.19	3
Разработка архитектуры системы	1 день	Вт 15.01.19	Вт 15.01.19	4
Реализация проекта	**76 дней**	**Ср 16.01.19**	**Ср 01.05.19**	
Нстройка и конфигурирование Grafana	9 дней	Ср 16.01.19	Пн 28.01.19	5
Подключение количественных метрик на основе логов	2 дней	Вт 29.01.19	Ср 30.01.19	7
Разработка плагина teamcity-data-importer для Grafana	11 дней	Чт 31.01.19	Чт 14.02.19	8
Разработка плагина rest-api-importer для Grafana	8 дней	Пт 15.02.19	Вт 26.02.19	9
Подключение Prometheus	31 дней	Ср 27.02.19	Ср 10.04.19	9;10
Конфигурирование сервисов под prometheus	8 дней	Чт 11.04.19	Пн 22.04.19	11
Настройка панелей визуализации	7 дней	Вт 23.04.19	Ср 01.05.19	12
Внедрение	**9 дней**	**Чт 02.05.19**	**Вт 14.05.19**	
Развертывание системы мониторинга	5 дней	Чт 02.05.19	Ср 08.05.19	13
Внедрение разработки	4 дней	Чт 09.05.19	Вт 14.05.19	15
Заключительный этап	**23 дней**	**Чт 02.05.19**	**Пн 03.06.19**	
Написание отчетной документации	20 дней	Чт 02.05.19	Ср 29.05.19	13
Проверка работы	3 дней	Чт 30.05.19	Пн 03.06.19	18

Рисунок 20 календарный план-график

4.4 Бюджет проекта

Бюджет проекта включает в себя стоимость всех расходов, необходимых для выполнения работ по магистерской диссертации. По окончании формирования бюджета, были выявлены следующие группы затрат:

- основная заработная плата исполнителей;
- дополнительная заработная плата исполнителей;
- отчисления во внебюджетные фонды (страховые отчисления);
- амортизация;
- накладные расходы.

4.4.1 Расчет материальных затрат

Материальные расходы включат только канцелярские принадлежности в сумме 2000 рублей.

4.4.2 Расчет амортизации

При расчете материальных затрат учитывались вся стоимость материалов, которая понадобилась на выполнение данной магистерской работы. В ходе выполнения работы был использован персональный компьютер организации. Срок полезного пользования для офисных машин составляет 2-3 года, персональный компьютер взят на 3 года. Стоимость персонального компьютера составляет 52 000 рублей. Написание работы составляет 10 месяцев. Тогда норма амортизации рассчитывается следующим образом:

$$A_n = \frac{1}{3} \times 100\% = 33.33\%$$

Годовые амортизационные отчисления составляют:

$$A_r = 52000 \times 0.33 = 17\,160 \text{ рублей}$$

Ежемесячные амортизационные отчисления составляют:

$$A_m = \frac{17160}{12} = 1430 \text{ рублей}$$

Итоговая сумма амортизации основных средств составляет:

$$A = 1430 \times 5 = 7150 \text{ рублей}$$

4.4.3 Расчет основной заработной платы исполнителей

Данный раздел включает в себя заработную плату научного руководителя, руководителя от организации и инженера. Расчет выполняется на основе трудоемкости выполнения каждого этапа и величины месячного оклада исполнителя.

Трудоемкость исполнителей на разных стадиях выполнения магистерской диссертации была просуммирована и представлена в виде количества дней. Таким образом, если согласно плану, работа над проектом должна вестись в течение 113 дней, то реально затраченное время каждого исполнителя в днях отличается от данной цифры, а также от того, что можно увидеть из диаграммы Ганта (так как на ней декомпозиция происходит с точностью до дней, а не часов).

Для расчета основной заработной платы необходимо рассчитать среднедневную заработную плату:

$$З_{дн} = \frac{З_м \times М}{F_д}$$

Где:

– $З_м$ – оклад работника за месяц, руб.

– М – количество месяцев работы без отпуска в течение года, 24 дня.

– $F_д$ – действительный годовой фонд рабочего времени персонала, 243 дня.

Основная заработная плата рассчитывается по следующей формуле:

$$З_{осн} = З_{дн} \times Т_р \times К_р$$

Расчет основной заработной платы представлен в таблице 4.8.

Таблица 4.8 - Расчет основной заработной платы

Исполнитель	**Оклад, руб./мес.**	**Средняя дневная ставка, руб.**	**$T_р$**	**$К_р$**	**Основная заработная плата, руб.**
Научный руководитель	33664	1 440,8	6	1,3	11 238,2
Руководитель от организации	35000	1 587,0	9		14 283,0
Инженер	21760	931,3	86		104 119,4
Итого $С_{осн}$					129 640,6

4.4.4 Расчет дополнительной заработной платы исполнителей

В данную статью включается сумма выплат, предусмотренных законодательством о труде. Например, оплата очередных и дополнительных отпусков; оплата времени, связанного с выполнением государственных и общественных обязанностей; выплата вознаграждения за выслугу лет и т. п. (в среднем — 12% от суммы основной заработной платы).

Расчеты дополнительной заработной платы представлены в таблице 4.9.

Таблица 4.9 - Расчет дополнительной заработной платы

Исполнитель	**Основная з/п, руб.**	**К.**	**Дополнительная з/п, руб.**
Научный руководитель	11 238,2	0,12	1 348,6
Руководитель от организации	14 283		1 713,9
Инженер	104 119,4		12 494,3
Итого $С_{доп}$			15 556,8

4.4.5 Расчет итоговой заработной платы

Исходя из расчетов, обозначенных в таблицах 4.8 и 4.9, была рассчитана итоговая заработная плата исполнителей, которая представлена в таблице 4.10.

Таблица 4.10 - Итоговая заработная плата

Исполнитель	**Основная з/п, руб**	**Дополнительная з/п, руб**	**Итоговая з/п, руб**
Научный руководитель	11238,2	1348,6	12 586,8
Руководитель от организации	14283	1713,9	15 996,9
Инженер	104119,4	12494,3	116 613,7
Итого $C_{зп}$			145 197,4

4.4.6 Расчет отчислений во внебюджетные фонды

При составлении расходов на проект, необходимо учитывать обязательные отчисления по установленным законодательством Российской Федерации нормам органам государственного социального страхования (ФСС), пенсионного фонда (ПФ) и медицинского страхования (ФФОМС) от затрат на оплату труда работников, что в сумме составляет 30%. Отчисления представлены в таблице 4.11.

Таблица 4.11 - Отчисления во внебюджетные фонды

Исполнитель	**Зарплата, руб.**	**Сумма отчислений, руб.**
Научный руководитель	12586,8	3776
Руководитель от организации	15996,9	4799,0
Инженер	116613,7	34984,1
Итого $C_{внеб}$		43559,1

4.4.7 Расчет накладных расходов

В данном разделе производится расчет тех затрат, не вошедших в расчеты предыдущих расходов: канцелярия, печать, оплата электроэнергии, пользование услугой интернет.

Такие расходы требуют низких затрат денежных средств относительно заработной платы исполнителей, поэтому величина коэффициента накладных расходов $k_{накл}$ была взята в размере 16%.

Расчет накладных расходов производится по формуле:

$$C_{накл} = k_{накл} \times C_{зп} = 0{,}16 \times 197906{,}5 = 31\,665$$

Где: $k_{накл}$ – величина коэффициента накладных расходов,

$C_{зп}$ – итоговая заработная плата исполнителей.

4.4.8 Формирование бюджета проекта

Исходя из произведенных расчетов, сумма всех расходов была рассчитана и представлена в таблице 4.12.

Таблица 4.12 – Статьи расходов

Расходы	**Сумма, руб.**	**Удельный вес, %**
Материальные затраты	2000	0,9
Основная заработная плата исполнителей	129 640,6	56,4
Дополнительная заработная плата исполнителей	15 556,8	6,7
Отчисления во внебюджетные фонды	43 559,1	18,9
Амортизация	7 150,0	3,1
Накладные расходы	31 665,0	13,7
Итого	**229 571,5**	**100**

Рассчитанный бюджет не превышает бюджета в 350000 рублей, указанного в ограничениях проекта (таблица 4.6, раздел 4.2.3).

4.5 Риски

Риск – это возможность наступления некоторого неблагоприятного события, влекущего за собой возникновение различного рода потерь. Единой классификации рисков проекта не существует. Поэтому выделим основные группы рисков для текущего проекта. Результат представлен в таблице 4.13.

Таблица 4.13 – Определение рисков

№ п/п	Наименование риска	Описание риска
1	Экономические	Выход за рамки бюджета.
2	Социальные	Система может быть невостребованной, в силу того, что интерфейс получится сложным и пользователи будут игнорировать разработанный мониторинг.
3	Технологические	– сложность реализации системы; – неправильно подобранные инструменты; отображение недостоверной (неактуальной) информации;
4	Организационные	Приостановление разработки системы мониторинга, для вовлечения в другой проект.
5	Кадровые	Уход сотрудника, владеющего знаниями в соответствующих технологиях.

Выявление рисков необходимо, чтобы выполнить оценку вероятности рисков и уровня потерь. Результат такой оценки представлен в таблице 4.14.

Таблица 4.14 – Оценка вероятности рисков и уровня потерь.

№ п/п	Наименование риска	Оценка вероятности риска (низкая, средняя, высокая)	Оценка уровня потерь (низкий, средний, высокий)
1	Экономические	Средняя	Высокий
2	Социальные	Средняя	Низкий
3	Технологические	Низкая	Высокий
4	Организационные	Низкая	Высокий
5	Кадровые	Высокая	Средний

В соответствие с определенными ранее рисками и уровнями потерь, заполним матрицу вероятности и потерь. Каждую группу рисков внесем в соответствующую ячейку матрицы. Результат представлен в таблице 4.14.

Таблица 4.14 – Матрица вероятности рисков и потерь.

		Уровень потерь		
		Высокий	Средний	Низкий
Вероятность	Высокая		Кадровые	
	Средняя	Экономические		Социальные
	Низкая	Технологические Организационные		

Красная область – высокий риск;

Желтая область – существенный риск;

Синяя область – умеренный риск;

Зеленая область – незначительный риск.

Для критических значений матрицы необходимо разработать мероприятия по снижению рисков. В такую категорию попадают две группы рисков – экономические и кадровые.

Минимизировать экономические риски можно, спланировав материальный график – рассчитав зарплату работников, покупку дополнительного оборудования амортизацию и т.д. Для этого и предназначен текущий раздел диссертации. Помимо этого, можно вести итеративную разработку с использованием гибких методологий, что добавит прозрачности проекту и обезопасит от экономических, технологических и даже социальных рисков на ранних стадиях.

Для снижения кадровых рисков, а именно уход работника, обладающего рядом знаний в данной предметной области, можно ввести ряд мероприятий по распространению знаний между всеми членами команды. Это позволит обезопасить компанию от утечки интеллекта, повысить компетентность

остальных членов компании в области разработки систем мониторинга. Для этого можно устраивать периодически хакатоны, доджи и технические обзоры внутри компании. Дополнительно можно отправлять разработчиков на различные тематические курсы и конференции.

4.6 Определение потенциального эффекта проекта

По описанным критериям, разобранных в данном разделе, можно сформировать итоговый результат эффективности исследования.

Потенциальными потребителями данной работы являются компании, разрабатывающие облачные и мобильные приложения с большим количеством потенциальных пользователей, и сложной спецификой предметной области, требующей мониторинг.

Конечным потребителем системы является компания ООО «МЦЦ Томск», для внутренних нужд которой и вводится система мониторинга. Однако это не исключает коммерциализацию проекта методом инжиниринга, путем предоставления услуг по консультации, проектированию и разработке аналогичных систем для других организаций на договорной основе.

QuaD анализ показал, что разработка является перспективной, о чем свидетельствует значение 85%. Были обнаружены слабые места, такие как финансовая эффективность и долгая окупаемость, однако это может исправить благодаря выбранному методу коммерциализации.

Общая длительность исследования составила пять месяцев. Были задействованы три исполнителя, для которых был произведен расчет основной и дополнительной заработной платы.

Потенциальная стоимость исследования не превысила бюджета исследования, а именно, не более 350 000 рублей.

Вычислим интегральный финансовый показатель разработки, как отношение затрат к финансовым ограничениям проекта – выделенному бюджету по следующей формуле:

$$I_{\text{ф}} = \frac{I_{\text{р}}}{I_{\text{огр.}}} = \frac{229\,571{,}5}{350\,000} = 0{,}65$$

Где: $I_ф$ – интегральный финансовый показатель,

$I_р$ – финансовые затраты на проект,

$I_{огр.}$ – финансовые ограничения проекта.

Значение равное 0,65 отражает численное удешевление стоимости разработки.

Одним из основных положительных эффектов исследования является то, что благодаря метрикам удалось понять востребованность конкретных функции, что скорректировало курс разработки. Иными словами, это может сократить затраты на разработку бесполезного функционала, и эффективно прогнозировать затраты на физические ресурсы (физическая и оперативная память, количество серверов и т.д.). Помимо этого, удалось обнаружить критичные области приложения с точки зрения использования ресурсов. Что снижает риски простоя приложения. Кроме того, благодаря тому, что все метрики приложения доступны в одном месте и каждому члену команды разработки приложения – появилась экономия времени. Теперь нет необходимости делать запрос к системным администраторам для поиска количественных характеристик системы – достаточно открыть панель с метриками и увидеть результат самостоятельно.

Глава 5. Социальная ответственность

5.1 Введение

Система мониторинга, разработанная в процессе написания магистерской диссертации, предполагается для облачного приложения врача-сомнолога. Разработанная система позволяет отобразить актуальное состояние приложения, включая метрики аппаратных ресурсов, количественные характеристики системы, такие как число зарегистрированных организаций, пациентов, пользователей, телеметрических приборов и т.д. Мониторинг предназначается для крупных облачных приложений с микросервисной архитектурой с медицинской тематикой, отражающей специфичные для данной предметной области характеристики.

Выполняемая работа заключалась в проектировании и разработке системы мониторинга облачного приложения. То есть, данную работу можно классифицировать как работу разработчика программного обеспечения. Пэтому в больше степени следует рассмотреть защиту человека от технических систем и технологий, а именно защиту пользователей компьютерной техники.

Разработка осуществлялась в офисе, в личном кабинете, на территории работодателя, за настольным персональным компьютером. Работодателем является компания ООО “МЦЦ Томск”, специализирующаяся в разработке программного обеспечения на заказ.

5.2 Правовые и организационные вопросы обеспечения безопасности

В процессе написания магистерской диссертации разработка алгоритма проводилась в офисном помещении за персональным компьютером. Для комфортного времяпрепровождения на территории рабочего места, организации необходимо соблюдать ряд правил для офисного помещения и использованию персональных компьютеров, изложенных в трудовом праве российской федерации.

Площадь одного рабочего места с компьютером должна быть не менее 6м^2. При размещении рабочих мест с персональными компьютерами должны

учитываться расстояния между рабочими столами с мониторами. Помещения с компьютерами в обязательном порядке должны быть оборудованы системами эффективной приточно-вытяжной вентиляцией отопления и кондиционирования воздуха. Внутренняя отделка интерьера помещений с компьютерами должна быть сделана при использовании диффузно-отражающих материалов с коэффициентами отражения для потолка от 0,7 до 0,8; для стен от 0,5 до 0,6; для пола от 0,3 до 0,5. В помещениях с эксплуатацией компьютеров поверхность пола должна быть не скользкой, ровной и удобной влажной уборки, а также иметь антистатические свойства.

Работодатели при установке ПК обязаны руководствоваться перечнем требований по отношению к помещению, освещению, организации медицинского обследования пользователей и других.

Также немаловажным фактором при выборе компьютеров для сотрудников является возможность конструкции компьютера изменять положение ПК в различных плоскостях (горизонтальные или вертикальные), с возможной устойчивой фиксацией в положении, которая удобна пользователю. Цвет корпуса ПК должен быть спокойным без блестящих деталей, которые бы вызывали повышенную утомляемость глаз. Экран монитора должен содержать регулировку яркости и контрастности, чтобы каждый работник мог установить нужный режим, которые будет соответствовать чувствительности глаз.

Устройство рабочего стола должно быть использовано для оптимального размещения используемого оборудования. Кроме того, форма рабочего стола должна быть удобна для поддержания рациональной позы пользователя, так, чтобы он мог менять положения своего тела для предупреждения утомления. В соответствии с ГОСТ 12.2.032-78 ССБТ «Рабочее место при выполнении работ сидя. Общие эргономические требования» [17] и СанПин к рабочему месту предъявляются следующие основные требования:

– При организации рабочего места следует учитывать антропометрические показатели женщин (если работают только женщины) и

мужчин (если работают только мужчины); если работают и женщины, и мужчины — общие средние показатели женщин и мужчин;

– Конструкцией рабочего места должно быть обеспечено оптимальное положение работающего, которое достигается регулированием высоты рабочей поверхности, сиденья и пространства для ног.

Продолжительность рабочего дня не должна превышать 40 часов в неделю. Вид трудовой деятельности за компьютерным устройством (компьютер, мобильное устройство), в рамках выполнения магистерской диссертации, соответствует группе «В» — творческая работа в режиме диалога с компьютерным устройством. Категория данной трудовой деятельности соответствует III (до 6 часов непосредственной работы за компьютером).

5.3 Профессиональная социальная ответственность

5.3.1 Анализ вредных и опасных факторов, которые может создать объект исследования

В процессе рассмотрения безопасности в рабочей зоне, необходимо было выявить вредные и опасные факторы, которые могут возникнуть на рабочем месте. Факторы считаются вредными, если его воздействие на человека может привести его к заболеванию. Опасный фактор может привести человека к травме.

Также необходимо описать мероприятия по защите исследователя и пользователей конечного продукта от действия данных факторов.

Факторы, влияющие на работу с компьютером, представлены в таблице 5.1.

Таблица 5.1 — Негативные факторы при работе с компьютером

Наименование видов работ и параметров производственного процесса	**Факторы (ГОСТ 12.0.003-74 ССБТ)**	**Нормативные документы**
Вредные факторы		
Работа с компьютером	Повышенная или пониженная температура воздуха рабочей зоны	СанПиН 2.2.4.548-96
	Повышенная или пониженная влажность воздуха рабочей зоны	СанПиН 2.2.4.548-96
	Повышенный уровень шума на рабочем месте	СанПиН 2.2.4/2.1.8.562-96
	Недостаточная освещенность рабочей зоны	СанПиН 2.2.1/2.1.1.1278-03
	Повышенный уровень электромагнитных излучений	СанПиН 2.2.2/2.4.1340-03
Опасные факторы		
Работа с компьютером	Опасность поражения электрическим током	ГОСТ 12.1.038-82
	Пожаровзрывоопасность	ГОСТ 12.1.041-83

5.3.2 Микроклимат

Одним из необходимых условий труда является обеспечение нормального микроклимата в рабочей зоне, оказывающего значительное влияние на самочувствие человека.

Системы отопления, несмотря на свою пользу, имеют и негативную сторону. Как центральное отопление, так и обогреватели сушат воздух. Пересушенный воздух при критических показателях создает значительную опасность для здоровья человека: способствует возникновению инфекций, провоцирует дерматиты, обострение аллергических заболеваний и астмы. Критической можно считать влажность воздуха менее 20-25%, что и наблюдается зимой в большинстве офисов. По факту, в полностью изолированных от уличного воздуха помещениях (с герметичными окнами,

системами кондиционирования и вентиляции), встречается влажность воздуха, равная 8%. Бороться с этим можно как с помощью дорогих климатических установок, так и используя, увлажнители воздуха. В подобных офисах необходимо установить дополнительную приточно-вытяжную вентиляцию.

Следствием недостаточной вентиляции, является низкое содержание кислорода в воздухе, что ведет к повышенной утомляемости сотрудников, сонливости, а также высокая влажность и конденсация влаги на охлажденных поверхностях (стенах, оконных откосах, стеклах), которая создает благоприятную среду для развития гнилостных грибков и плесени – сильнейших аллергенов.

Так как работа с использованием персонального компьютера является основной, (диспетчерские, операторские, расчетные, кабины и посты управления, залы вычислительной техники и др.) и связана с нервно-эмоциональным напряжением, должны быть обеспечены оптимальные параметры микроклимата для категории работ 1а и 1б в соответствии с действующими санитарно-эпидемиологическими нормативами микроклимата производственных помещений. На других рабочих местах следует поддерживать параметры микроклимата на допустимом уровне, соответствующем требованиям указанных выше нормативов.

Содержание вредных химических веществ в таких рабочих зонах не должно превышать предельно допустимых концентраций загрязняющих веществ в атмосферном воздухе населенных мест в соответствии с действующими гигиеническими нормативами. Также, в помещениях с персональными компьютерами должна проводиться ежедневная влажная уборка.

Нормативным документов, отвечающим за гигиенические требования к микроклимату производственных помещений, является СанПиН 2.2.4.548-96 [18]. В документе указаны все нормативные требования к микроклимату на рабочих местах, у всех видов производственных помещений. Работа

разработчика относится к категории 1а, потому что выполняется сидя и сопровождается незначительным физическим напряжением.

В таблице 5.2 представлены фактические, оптимальные и допустимые показатели микроклимата рабочей зоны.

Таблица 5.2 — Параметры микроклимата на рабочем месте

Период года	**Кат. раб.**	**Температура воздуха, °С**			**Относительная влажность воздуха, %**			**Скорость движения воздуха, м/с**		
		Факт.*	Опт.	Доп.	Факт.*	Опт.	Доп.	Факт.**	Опт.	Доп.
Холодный	Iа	23	22-24	20-25	55	40-60	15-75	0,1	0,1	0,1
Теплый	Iа	23	23-25	21-28	50	40-60	15-75	0,1	0,1	0,1-0,2

Оптимальный микроклимат является необходимым на рабочих местах, так как создает комфортное нахождение человека в рабочей зоне, а также обеспечивает его высокий уровень работоспособности. Такие микроклиматические условия обеспечивают благоприятное состояние организму человека и не вызывают отклонений в состоянии его здоровья.

5.3.3 Шум

Шум – колебания различной физической природы, отличающиеся сложностью спектральной и временной структуры [19]. Шум создает значительную нагрузку на нервную систему человека, оказывая на него психологическое воздействие. Шумовой фон провоцирует увеличение содержания в крови гормонов стресса, таких как, норадреналин и адреналин, кортизол. Шум способен замедлять реакцию человека и угнетать центральную нервную систему (ЦНС), вызывая изменения скорости пульса и дыхания, а также

* Измерения были произведены с помощью датчик температуры и влажности Даджет 8060.
** Измерения произведены с помощью портативного ареометра Smart Sensor AR816.

провоцирует возникновение сердечно - сосудистых заболеваний, гипертонических болезней и язвы желудка.

Среди источников шума выделяют принтеры, систему охлаждения, множительная техника, осветительные приборы дневного света, а также шумы, проникающие извне.

В рассматриваемом рабочем помещение уровень шума является допустимым и не превышает значений, установленных СанПиН 2.2.4/2.1.8.562-96 [19] и составляет не более 50 дБА. Допустимые значения уровней звукового давления представлены в таблице 5.3

Таблица 5.3 — Допустимые значения уровней звукового давления компьютера

Уровни звукового давления в октавных полосах со среднегеометрическими частотами									**Уровни звука в дБА**
31,5 Гц	63 Гц	125 Гц	250 Гц	500 Гц	1000 Гц	2000 Гц	4000 Гц	8000 Гц	
86 дБ	71 дБ	61 ДБ	54 дБ	49 дБ	45 дБ	42 дБ	40 дБ	38 дБ	50

Снижению уровня шума способствует установка звукопоглощающих материалов (плиты, панели), подвесных акустических потолков, а также установка малошумного оборудования.

5.3.4 Монотонный режим работы

Монотонный режим работы заключается в некоторых видов работ, при которых человек долгое время выполняет однообразные элементарные действия либо имеет место быть предельная концентрация внимания на какой-либо деятельности. Работа программиста очень сильно подвержена этому неблагоприятному фактору, так как при написании кода от программиста требуется очень большая концентрация и повышенное внимание. Этот фактор относится к психофизическому типу.

Для борьбы с монотонностью можно использовать следующие методы:

– усложнение рабочих операций, выполняемых действий, объединение их в комплексы;

– увеличение темпа работы или подачи информации (сигналов);

– расчленение общего задания на отдельные части для того, чтобы появились промежуточные (поэтапные) цели;

– прерываться на 5 минутный отдых.

5.3.5 Освещенность

Недостаточная освещенность рабочей зоны оказывает негативное влияние на зрительную систему человека. Происходит снижение концентрации внимания, усталость центральной нервной системы, что приводит к снижению производительности труда.

Уровень освещения на поверхности рабочего стола в зоне размещения документа, согласно СанПиН 2.2.2/2.4.1340-03 [20], должен быть в диапазоне от 300 до 500 лк. Уровень освещенности экрана не должен превышать 300 лк. Яркость осветительных приборов, находящихся в поле зрения, не должна превышать 200 кд/м^2.

Приведем расчет искусственного освещения для прямоугольного помещения, размерами: длина А = 5 м, ширина В = 6 м, высота Н = 4 м, количество ламп N = 12 шт.

Определим расчетную высоту подвеса светильников над рабочей поверхностью (h) по формуле:

$$h = H - h_p - h_c, \tag{5.1}$$

где H – высота потолка в помещении, м; h_p – расстояние от пола до рабочей поверхности стола, м; h_c – расстояние от потолка до светильника, м.

Вычислим расчетную высоту подвеса светильников над рабочей поверхностью по формуле 5.1 для компьютерной аудитории кафедры программной инженерии:

$$h = 4 - 0,8 - 0,01 = 3,19 \text{ м.}$$

Индекс помещения определяется по формуле (5.2):

$$i = \frac{S}{h(A+B)}, \tag{5.2}$$

где S – площадь помещения, м^2; A – длина комнаты, м; B – ширина комнаты, м; h – высота подвеса светильников, м.

Индекс помещения для компьютерной аудитории кафедры программной инженерии:

$$i = \frac{30}{3{,}19(5+6)} = 0{,}83.$$

Исходя из того, что потолок в помещении чистый бетонный, а также свежепобеленные стены без окон, согласно методическим указаниям, примем коэффициенты отражения от стен ρ_c=*70%* и потолка ρ_n=*50%*. По таблице коэффициентов использования светового потока для соответствующих значений i, ρ_c, ρ_n, примем η=0,29.

Освещенность помещения рассчитывается по формуле:

$$E_\phi = \frac{n \cdot \eta \cdot \Phi}{S \cdot k_з \cdot z}; \qquad (5.3)$$

где Φ – световой поток светильника, лм; S – площадь помещения, м^2; $k_з$ – коэффициент неравномерности освещения; n – число светильников; η – коэффициент использования светового потока.

Коэффициент запаса k учитывает запыленность светильников и их износ. Для помещений с малым выделением пыли k = 1,5. Поправочный коэффициент z – это коэффициент неравномерности освещения. Для люминесцентных ламп z = 1,1. В помещении находятся светильники ЛВО 4×18 CSVT, с люминесцентными лампами типа L 18W/640 с потоком F = 1200 лм.

Учитывая все параметры, рассмотренные выше, найдем освещенность по формуле 5.3:

$$E_\phi = \frac{48 \cdot 0{,}29 \cdot 1200}{30 \cdot 1{,}5 \cdot 1{,}1} = 337 \text{ лк}.$$

В рассматриваемом помещении освещенность должна составлять 300 лк согласно СНиП 23-05-95. В данном помещении освещенность находится в пределах нормы, следовательно, дополнительные источники света не нужны.

Вся электротехника, и компьютеры в том числе, производят электромагнитное излучение. В таблице 5.4 представлены временные допустимые уровни электромагнитных полей, создаваемые компьютерами на рабочих местах, согласно СанПиНу 2.2.2/2.4.1340-03 [20].

Таблица № 5.4 — Временные допустимые уровни электромагнитных полей

Наименование параметров		**Временные допустимые уровни электромагнитных полей**
Напряженность электрического поля	в диапазоне частот 5 Гц-2 кГц	25 В/м
	в диапазоне частот 2 кГц-400 кГц	2,5 В/м
Плотность магнитного потока	в диапазоне частот 5 Гц-2 кГц	250 нТл
	в диапазоне частот 2 кГц-400 кГц	25 нТл
Поверхностный видеомонитора электростатический потенциал экрана		500В

5.3.6 Психофизиологические факторы

Продолжительность непрерывной работы за компьютерным устройством, без регламентированного перерыва, не должна превышать 2 часа. Длительность регламентированных перерывов составляет 20 минут (после 1,5-2,0 часа от начала рабочей смены и обеденного перерыва). Также, необходимо уделять время нерегламентированным перерывам (микропаузы), длительность которых составляет 1-3 минуты.

Для снижения воздействия вредных факторов, устанавливаются перерывы в работе для отдыха сотрудников. Суммарное время регламентированных перерывов при работе с персональным компьютером зависит от категории трудовой деятельности и уровня нагрузки за рабочую смену. В таблице 5.5 приведено суммарное время отдыха для каждой категории работ.

Таблица 5.5 — Суммарное время перерывов

Категория работы с ПК	**Уровень нагрузки за рабочую смену при видах работ с ПК**			**Суммарное время регламентированных перерывов при 8-часовой смене, мин.**
	Группа А, количество знаков	Группа Б, количество знаков	Группа В, количество знаков	
I	до 20000	до 15000	до 2	50
II	до 40000	до 30000	до 4	70
III	до 60000	до 40000	до 6	90

5.3.7 Статическое электричество

В помещениях, оборудованных ПЭВМ, токи статического электричества чаще всего возникают при прикосновении персонала к любому из элементов ПЭВМ. Такие разряды опасности для человека не представляют, однако кроме неприятных ощущений могут привести к выходу оборудования из строя.

Для предотвращения образования и защиты от статического электричества в помещении используются нейтрализаторы и увлажнители, а полы имеют антистатическое покрытие в виде поливинилхлоридного антистатического линолеума.

Также в СанПиН 2.2.2/2.4.1340-03 установлен максимальный допустимый электростатический потенциал экрана видеомонитора – 500 В.

В качестве мер уменьшения влияния вредных факторов на пользователя используются защитные фильтры для мониторов, увлажнители воздуха. Должны использоваться розетки с заземлением. Требуется проводить регулярную влажную уборку.

5.3.8 Электрический ток

Среди распространенных опасностей в рабочей зоне находится и поражение электрическим током. Опасность поражения определяется величиной тока проходящего через тело человека I или напряжением прикосновения U. Напряжение считается безопасным при напряжении прикосновения $U < 42$ В.

При получении человеком разряда электрического тока могут быть получены электротравмы, электрические удары и даже летальный исход (согласно ГОСТ 12.1.009-2009 [21]).

Для защиты от поражения электрическим током следует выполнить следующие пункты:

– обеспечить недоступность токоведущих частей от случайных прикосновений;

– электрическое разделение цепей;

– устранить опасность поражения при появлении напряжения на разных частях.

В таблице 5.6 представлены предельно допустимые значения напряжения прикосновения и тока на рабочем месте (согласно ГОСТ 12.1.038-82 [22]).

Таблица 5.6 — Допустимые значения напряжения прикосновения и тока

Род тока	**Напряжения прикосновения, В**	**Ток, мА**
	Не более	
Переменный, 50 Гц	2,0	0,3
Постоянный	8,0	1,0

Согласно электробезопасности, рабочее место относится к помещениям без повышенной опасности поражения электрическим током. Данный фактор характеризуется отсутствием условий, создающих повышенную или особую безопасность от разряда электрическим током.

5.4 Экологическая безопасность

В настоящее время проблема экологической безопасности и охраны окружающей среды стоит под острым вопросом и является приоритетной. Это стало поводом для принятия жестких законов, ограничивающих обычную утилизацию компьютерной техники. В большей мере это обусловливается тем, что в производстве такой техники используется множество различных материалов, которые способны нанести непоправимый вред окружающей среде

и, соответственно, здоровью человека. Утилизация компьютерного оборудования является достаточно сложной. Непосредственная переработка большей части компонентов включает в себя их сортировку, последующую гомогенизацию и отправку для повторного использования, т.е. с предварительным помолом или переплавкой. Люминесцентные лампы представляют собой «чрезвычайно опасные» виды отходов [19]. Содержание ртути в любых люминесцентных лампах составляет от трех до пяти миллиграмм ртути. С учетом этого необходимо обеспечивать определенные условия хранения, их эксплуатации и утилизации. Согласно санитарным нормам хранить ртутесодержащие отходы необходимо в специальных герметичных контейнерах, доступ посторонним лицам к таким контейнерам должен быть запрещен. Транспортировка ламп на полигоны складирования должна выполняться организациями, которые специализируются на утилизации опасных отходов. Категорически запрещено размещение таких отходов, как люминесцентные лампы на полигонах твердых бытовых отходов.

Так как при разработке данной магистерской диссертации использовался персональный компьютер, необходимо описать правильную утилизацию компьютерного лома после его выхода из строя. В соответствии с постановлением правительства юридическим лицам запрещено самостоятельно утилизировать компьютерную технику. Для этого необходимо найти специальную компанию, которая занимается утилизацией в частном порядке.

В нормативном документе СанПиН 2.2.2/2.4.1340-03 [20] даются следующие общие рекомендации по снижению опасности для окружающей среды, исходящей от компьютерной техники:

– применять оборудование, соответствующее санитарным нормам и стандартам экологической безопасности;

– применять расходные материалы с высоким коэффициентом использования и возможностью их полной или частичной регенерации;

– отходы в виде компьютерного лома утилизировать;

– использовать экономичные режимы работы оборудования.

5.5 Безопасность при чрезвычайных ситуациях

5.5.1 Анализ вероятных чрезвычайных ситуаций

Самым распространенным чрезвычайным обстоятельством в офисе является пожар. Такое рабочее место относится к категории “В” (пожароопасные), так как в данном помещении присутствует пыль, вещества и материалы, способные при взаимодействии с воздухом гореть. Возникновение пожара может произойти по нескольким факторам:

– возникновением короткого замыкания в электропроводке вследствие неисправности самой проводки или электросоединений и электрораспределительных щитов;

– возгоранием устройств вычислительной аппаратуры вследствие нарушения изоляции или неисправности самой аппаратуры;

– возгоранием мебели или пола по причине нарушения правил пожарной безопасности, а также неправильного использования дополнительных бытовых электроприборов и электроустановок;

– возгоранием устройств искусственного освещения.

5.5.2 Мероприятия по предотвращению чрезвычайных ситуаций и порядок действия в случае возникновения чрезвычайных ситуаций.

Под пожарной безопасностью понимают состояние объекта, при котором возможность возникновения пожара исключено, но в случае его возникновения предотвращается влияние на людей опасных факторов пожара, а также обеспечивается защита материальных ценностей. Пожарная безопасность обеспечивается системой пожарной защиты и системой предотвращения пожара. Основные причины возникновения пожаров:

– халатность;

– перегрузка электросети;

– поджог;

– разряд молнии и неисправность молниеотвода;

– незащищенность от действия солнечных лучей.

Действия при пожаре в здании:

– при наличии телефона, «112» или «01» и сообщить о пожаре и своем местоположении;

– не входить в места с высокой концентрацией дыма и видимостью менее, чем 10 метров.

Если имеется возможность выйти из помещения (здания) наружу:

– покинуть помещение, используя запасные и основные пути эвакуации;

– попутно отключить электроэнергию;

– передвигаться к выходу на четвереньках, при этом закрывая рот и нос подручными средствами защиты;

– плотно закрыть дверь при выходе;

Если дым и пламя в соседних помещениях не позволяет выйти наружу:

– не поддаваться панике;

– проверить возможности спуститься по пожарной лестнице или выйти на крышу;

– при отсутствии возможности эвакуироваться, для защиты от дыма и тепла необходимо как можно надёжней загерметизировать своё помещение:

  - закрыть плотно двери, заткнуть щели дверей изнутри, используя при этом любую, желательно мокрую, ткань;
  - закрыть окна и форточки.

– при наличии воды, постоянно смачивать двери и пол;

– при задымлении помещения, передвигаться только на четвереньках, прикрыв рот и нос влажным носовым платком или рукавом, в сторону окна и находиться возле окна, при этом привлекать к себе внимание людей на улице.

Для устранения возможных причин возгорания следует проводить следующие мероприятия:

1. Организационные мероприятия:

– противопожарный инструктаж обслуживающего персонала;

– обучение персонала техники безопасности;

– разработка инструкций, планов эвакуаций и пр.

2. Эксплуатационные мероприятия:

– соблюдение эксплуатационных норм оборудования;

– выбор и использование современных автоматических средств пожаротушения.

3. Технические мероприятия:

– профилактический осмотр и ремонт оборудования;

соблюдение противопожарных мероприятий при устройстве электропроводок, оборудования, систем отопления и пр. [23]

Заключение

Результатом написания магистерской диссертации является спроектированная и разработанная система мониторинга для облачного приложения, имеющего микросервисную архитектуру, сложные бизнес процессы, медицинскую специфику и внешнюю телеметрию.

Система наглядно отражает актуальное состояние приложения в визуальной форме, с помощью графиков, диаграмм и различных графических индикаторов. Процесс визуализации бизнес показателей, таких как: количество зарегистрированных пользователей, организаций, полисомнографических устройств и т.д. осуществляется на основе лог – файлов отдельный сервисов приложения. Размер дневного индекса поступающей информации варьируется от гигабайта до десяти, что делает задачу визуализации более трудоемкой, но в тоже время еще более интересной. Так как лог – файлы являются слабоструктурированными данными, и при таких ежедневных объемах, то решение сводится к обработке больших данных.

Инструменты для аналитики и визуализации были подобраны точным образом. За время тестирования система ни разу не отклонялась от фактических значений. При высоких нагрузках не было и единого намека на зависание и медленную работу.

Компонент мониторинга Kibana – позволяет производить мониторинг лог - файлов в конкретный момент, анализировать детали, дает точное описание произошедшего события в системе в конкретный момент времени. Grafana – позволяет увидеть картину в целом. Это значит, что не удастся локализовать конкретное место, но зато будет видна тенденция роста или спада активности пользователей. Например, после внедрения системы мониторинга удалось выявились пиковые часы активности (10 - 12 часов утра). В это время медицинские приборы отсылают телеметрию в облачное приложение. После чего, происходит вторая волна активности, вызванная уже пользователями – врачами, которые проверяют актуальную информацию за день.

Еще один положительный эффект – это разгрузка системных администраторов приложения. Потому что до введения системы мониторинга, эта работа лежала на них. Чтобы получить одну количественную метрику, уходило до получаса, чтобы собрать данные из всех сервисов и проанализировать результаты. Сейчас – достаточно открыть необходимую панель визуализации с метриками и увидеть значение. Пример метрик доступен в приложении 7 - 10.

Health – мониторинг заменил существовавший ранее дашборд. Старая панель визуализации представлена на рисунке 21.

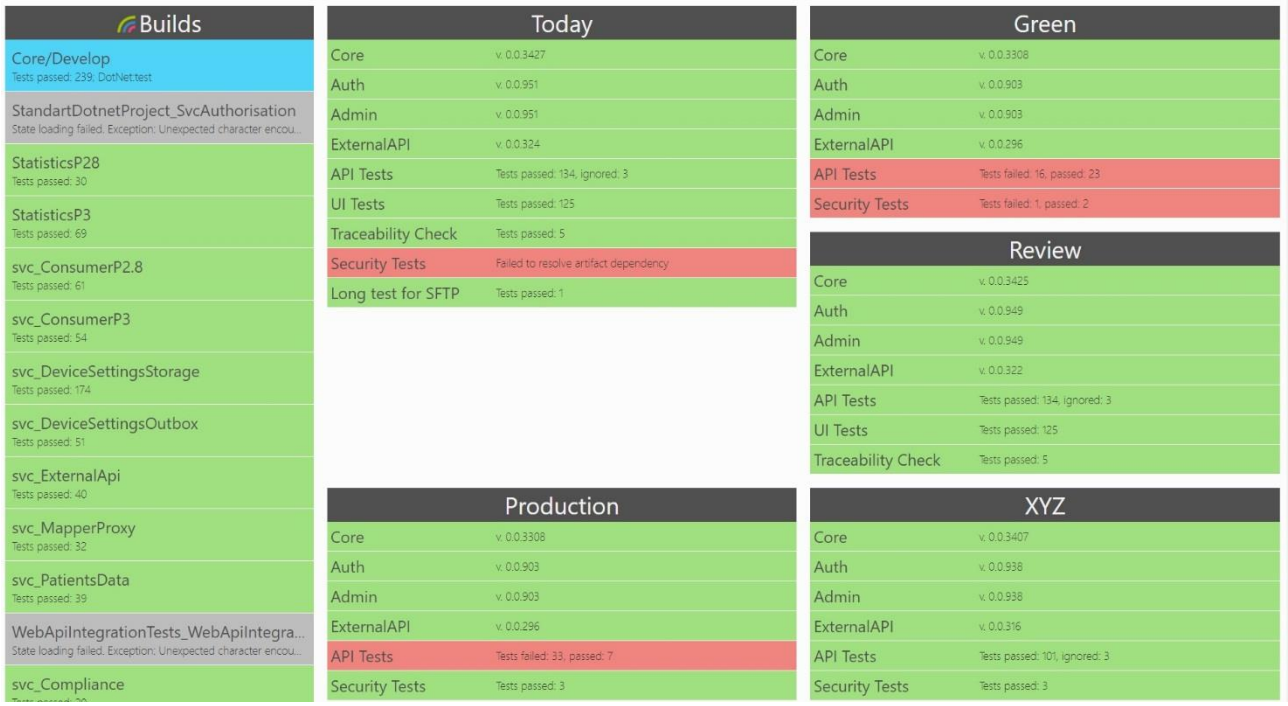


Рисунок 21 старый дашборд health – мониторинга

С появлением нового дашборда отпала необходимость в перекомпоновки элементов, чтобы вместились все сервисы. Исчезли проблемы с зависанием метрик, которые были присуще старому варианту. Это происходило из-за переполнения кеша приложений IIServer, что приводило к отображению ложно информации. Новый дашборд представлен на рисунке 22.

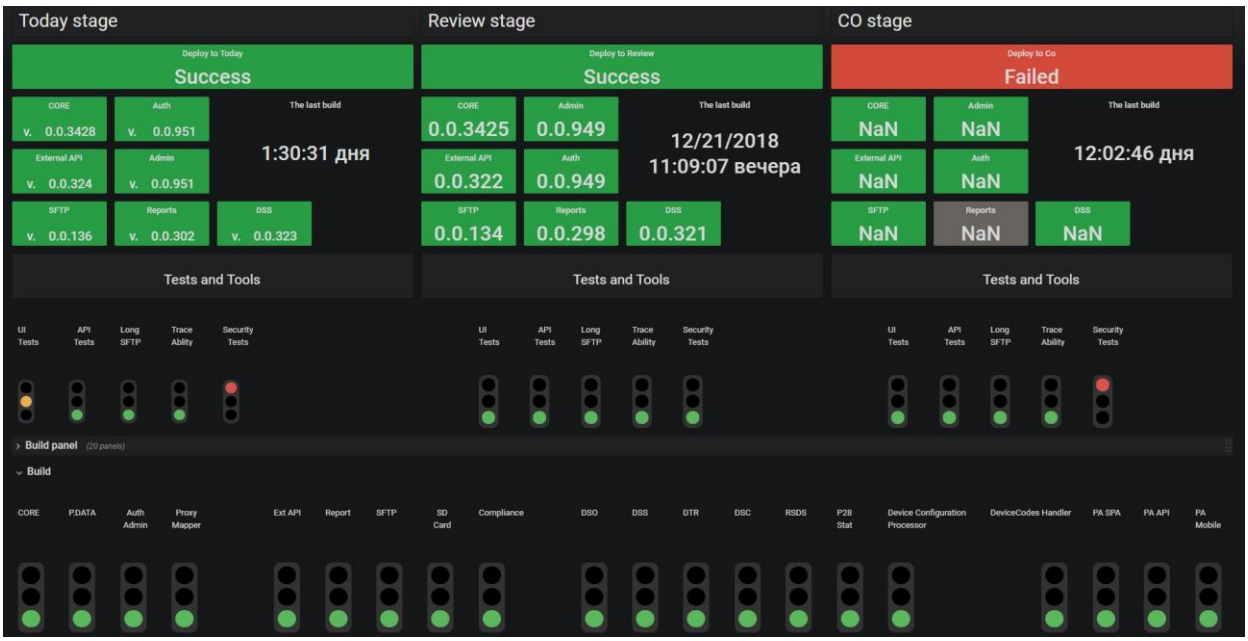


Рисунок 22 health - мониторинг

Система удовлетворяет ожидаемым требованиям, но есть еще идеи для реализации и повышения качества и глубины мониторинга. Так, например, сконфигурировать каждый сервис, на проверку самомониторинга: подключение к базе, очередям mqtt – брокера, service bus. В результате health-monitoring стал бы более осмысленным, чем проверка сервиса на его доступность.

Список публикаций

1. Васин М.А. Микросервисная архитектура как основа облачных технологий. XXII Международная научно-практическая конференция «Российская наука в современном мире» г. Москва, 31 мая 2019 года
2. Васин М.А. Проблемы мониторинга приложений с микросервисной архитектурой. XXVI Международная научно-практическая конференция «Фундаментальные и прикладные исследования в современном мире», г. Санкт-Петербург, 30 мая 2019 года.

Список используемой литературы

1. Владзимирский, Лебедев: Телемедицина. Руководство. ГЭОТАР-Медиа, 2018 г. ISBN: 978-5-9704-4195-4, 576 с.
2. Блажис А.К., Дюк В.А. Телемедицина. – СПб: «СпецЛит», 2000. – 154 с.
3. Гусиос Георгиос, Спинеллис Диомидис. Идеальная архитектура. Ведущие специалисты о красоте программных архитектур. – Символ-Плюс, 2010 г. 580 с.
4. Michael J. Kavis. Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS). – Wiley; 1 edition (January 28, 2014), 224 pages.
5. Сэм Ньюмен. Создание микросервисов. – Бестселлеры O'Reilly (Питер), 2016, 304 с.
6. Open Source Search & Analytics · Elasticsearch, [Электронный ресурс] - Kibana User Guide https://www.elastic.co/guide/en/kibana/6.2/index.html (дата обращения: 20.05.2019)
7. Игорь Сухоруков, Материалы конференции SECR-2014, "Сбор и анализ логов и метрик распределенного приложения с помощью Elasticsearch, Logstash, Kibana"
8. Betsy Beyer, Niall Richard Murphy, David K. Rensin, Kent Kawahara, Stephen Thorne. The Site Reliability Workbook: Practical Ways to Implement SRE. – O'Reilly Media, Inc., 2018 г. 512 с.
9. Betsy Beyer, Chris Jones, Jennifer Petoff, Niall Richard Murphy. Site Reliability Engineering: How Google Runs Production Systems. – O'Reilly Media, Incorporated, 2016 г. 524 с.
10. Gigi Sayfan. Mastering Kubernetes: Master the art of container management by using the power of Kubernetes. – Packt Publishing; 2nd Revised edition edition (April 27, 2018), 468 pages.
11. habrahabr.ru, [Электронный ресурс] - Блог компании 2ГИС, статья "Ещё одна система логирования, теперь на ElasticSearch, Logstash, Kibana и

Prometheus", https://habr.com/company/2gis/blog/329128/ (дата обращения: 20.05.2019)

12. An open-source monitoring system with a dimensional data model, flexible query language, efficient time series database and modern alerting approach, [Электронный ресурс] - Documentation | Prometheus https://prometheus.io/docs/ (дата обращения: 20.05.2019)

13. Grafana Labs, [Электронный ресурс] - docs, http://docs.grafana.org/ (дата обращения: 20.08.2018)

14. TeamCity Graphite Integration, [Электронный ресурс] - .Net logging https://code.mendhak.com/teamcity-graphite/ (дата обращения: 22.05.2019)

15. App Metrics [Электронный ресурс] – docs, https://www.app-metrics.io/ (дата обращения: 22.05.2019)

16. github.com, [Электронный ресурс] - крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки, TypeScript Template Data Source for Grafana, https://github.com/grafana/typescript-template-datasource (дата обращения: 22.05.2019)

17. ГОСТ 12.2.032-78 ССБТ. Рабочее место при выполнении работ сидя. Общие эргономические требования.

18. СанПиН 2.2.4.548–96. Гигиенические требования к микроклимату производственных помещений.

19. СанПиН 2.2.4/2.1.8.562–96. Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории застройки.

20. СанПиН 2.2.2/2.4.1340–03. Санитарно-эпидемиологические правила и нормативы «Гигиенические требования к персональным электронновычислительным машинам и организации работы».

21. ГОСТ 12.1.009-2009. Система стандартов безопасности труда. Электробезопасность.

22. ГОСТ 12.1.038-82 ССБТ. Электробезопасность. Предельно допустимые уровни напряжений прикосновения и токов.

23. ГОСТ Р 22.3.03-94. Безопасность в ЧС. Защита населения. Основные положения.

Приложение 1
Chapter 1. Characteristics and features of the cloud application

Студент:

Группа	ФИО	Подпись	Дата
8ПМ7И	Васин Максим Алексеевич		

Консультант проф. кафедры:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент	Соколова Вероника Валерьевна	к.т.н.		

Консультант – лингвист кафедры ОИЯ ШБИП:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент	Диденко Анастасия Владимировна	к.ф.н.		

Chapter 1. Characteristics and features of the cloud application

1.1 Description of the subject area.

1.1.1 Telemedicine

Telemedicine is an instrument of modern healthcare aimed at optimizing the organization, standardizing the quality and availability of medical care. Telemedicine implies the use of telecommunications to connect medical specialists with clinics, hospitals, primary care physicians, patients at a distance, for the purpose of diagnosis, treatment, counselling, and continuous education. Technologically, this kind of telecommunication should provide direct transmission of medical information in various formats (medical history, laboratory data, X-rays and CT results, video images, ultrasound, etc.), as well as real-time video conferencing between medical institutions or the doctor and patients.

One of the areas of health care where telemedicine is actively used is sleep medicine. Or, as it is customary to name this science – somnology. This is a very young branch of medicine that studies various aspects and diseases of human sleep.

The transmitted information must be stored and processed somewhere. The trend of recent years is data processing in cloud applications.

1.1.2 Cloud applications

Cloud applications perform all calculations on remote resources provided to the user via the Internet as an online service. According to IEEE terminology, cloud technologies are a paradigm that allows constantly storing user information on Internet servers.

There are several types of cloud applications, we will consider the characteristic type for this subject area - SaaS (Eng. System as a service). This type of cloud application allows multiple clients to access a single application using a browser. The developer company controls the application independently and provides access to the platform via the Internet. There are advantages for both sides: for or the customer, the benefit lies in saving on equipment – there is no need to buy expensive high-

performance equipment and software. For the developer, the SaaS model allows efficiently dealing with unlicensed use of software, due to the fact that the software does not actually reach the final customers. In addition, this concept reduces the cost of deployment and implementation of technical and advisory product support systems, however, it does not eliminate them completely.

Although it looks to a user as one single application, the internal architecture of this type of cloud application often looks like a set of separate services that perform a strictly specific task, in its process and environment. The interaction between the elements of the application and external components is usually carried out via HTTP. This architectural approach is called microservice.

As indicated earlier, the idea is to divide the application into a set of small interrelated services. Instead of sharing one database schema with other services, each microservice has its own database. On the one hand, this approach is contrary to the idea of a data model for the entire application. In addition, this often leads to duplication of data. However, on the other hand, the presence of its own separate databases provides a weak link.

End users do not have direct access to internal services. Instead, communication takes place through an intermediary known as a gateway. API Gateway is responsible for tasks such as load balancing, caching, access control and monitoring. Figure 1 presents a generalized diagram of the application with microservice architecture.

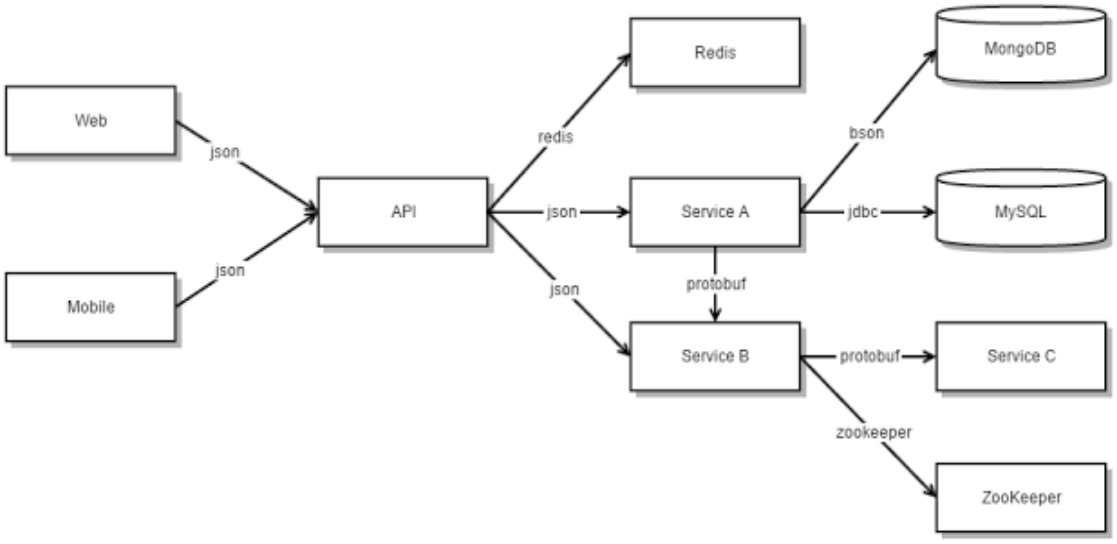


Figure 1. Generalized diagram of the application with microservice architecture

1.1.3 Subject area of the application.

The monitoring system which is being developed is intended for the cloud application of the somnologist. Due to commercial secret and security aspects of users, the technical details and the name will not be disclosed.

The application allows the doctor to monitor the state of patients with sleep disorders remotely. Each patient has a special device that registers the polysomnographic signals during sleep. Then via the Internet the signals enter the system, where they are analyzed and stored after being distributed across various microservices of the system.

In addition to the complex technological flow of medical data processing, the application contains standard elements and functions such as search, pagination, filters, authorization, etc. These functions should also be monitored to identify the relevance of certain characteristics and assess the load on the application as a whole.

1.1.4 Cloud application logging system

Each microservice places log information in a separate file, which only complicates the process in the case of manual log processing. However, for quick

response to extreme errors and log file monitoring in the application, we use ELK - a technology stack consisting of Elasticsearch, Logstash and Kibana.

Quality managers monitor bursts of errors in the system after the next release and transmit this information to the development department. Those, in turn, find correlations with errors in the application, immediately see the function call stacks and other details needed for debugging. Figure 2 shows an example of system log records.

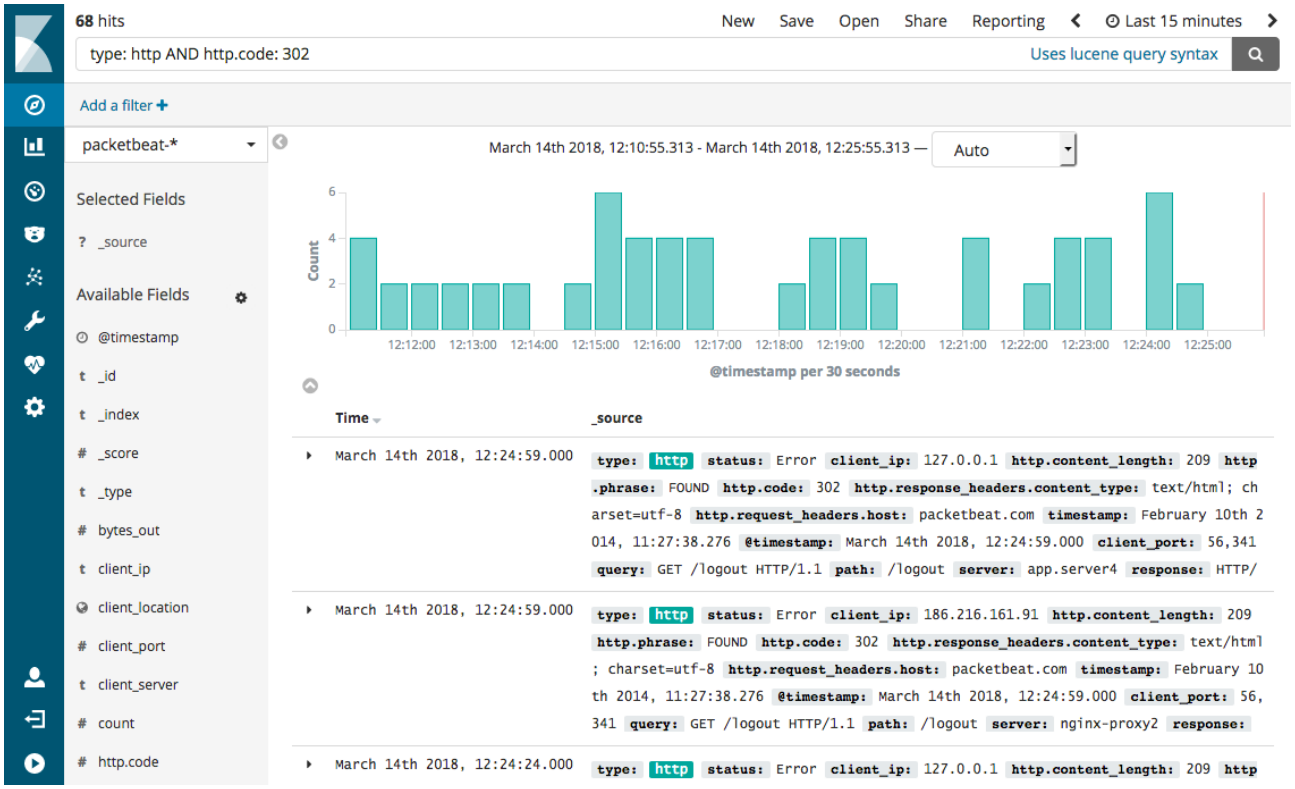


Figure 2. Example of logs in the Kibana service

1.2 Objectives of monitoring

1.2.1 Business metrics

This block is designed to display the actual information of the cloud application. For example, it may display the number of registered users and authorizations in the system, the frequency of using a particular filter when searching, the number of statistics received from the devices per day and other various indicators specific to the subject area.

The ultimate goal of this set of metrics is necessary for a better understanding of the system by its users in order to later upgrade one or another

business process. That is, business metrics provide information on the use of specific functions of the application, for example, authorization using the second factor, generating reports, etc.

If any function is a success, then it is worth reflecting on the ways to improve and simplify it. And accordingly, on the contrary, if the function is of little demand, then its subsequent detailed development should not be continued and has to be stopped at the current state.

In addition, business metrics not only demonstrate the behavior of end users, but also characterize the quantitative properties of the application. For example, if we know that a large number of organizations are registered in our system, then when implementing a new function, it is worth reflecting on the architecture, whether it will add additional load on the memory while the application is running and whether it is worth optimizing existing code sections.

1.2.2 Hardware Resource Metrics

Monitoring servers, both physical and virtual, shows how correctly the application is implemented in terms of memory, processor time, etc. If the application has bottlenecks, then this group of metrics will provide an immediate response to this kind of problem. For example, when searching for patients, a situation may occur when the code is not executed on the side of the DBMS, but the entire data set is loaded into memory and then is performed in RAM. This method instantly occupies all available memory in a virtual or physical machine and the application slows down its work, and sometimes it crashes completely. Therefore, in order to understand the system from the point of view of resources, it is necessary to monitor each service separately, how many resources of each resource it consumes and notify the developers when a certain threshold is exceeded.

1.2.3 Viability and availability of services (health monitoring)

In the case of microservice architecture, such outcomes are possible when the application partially stops working, because each microservice is launched by a separate process. That is, a certain set of functions may not be available.

An example would be that the report generation service has ceased to function. The user can open the application, search, create, edit patients, change various system settings, but when he or she wants to generate a report, at best he or she will receive a notification that the report could not be generated, or nothing will visually happen.

To avoid such problems, it is enough to periodically poll each service through the API. In case of a successful response, the service is available and operational, otherwise the service is not started. There may be several reasons: an unsuccessful application deployment process after the next release; a crash; the server where the service is located, is physically inaccessible, etc. The application logs can give the answer to this question. However, in order to understand this before the users of the system do, it is worth displaying such information or even sending a notification to the developers. An example of a successful service availability monitoring is Microsoft and its Azure DevOps application. For each segment of the application, they display the availability of their services in the public domain. The example is presented in Figure 3.

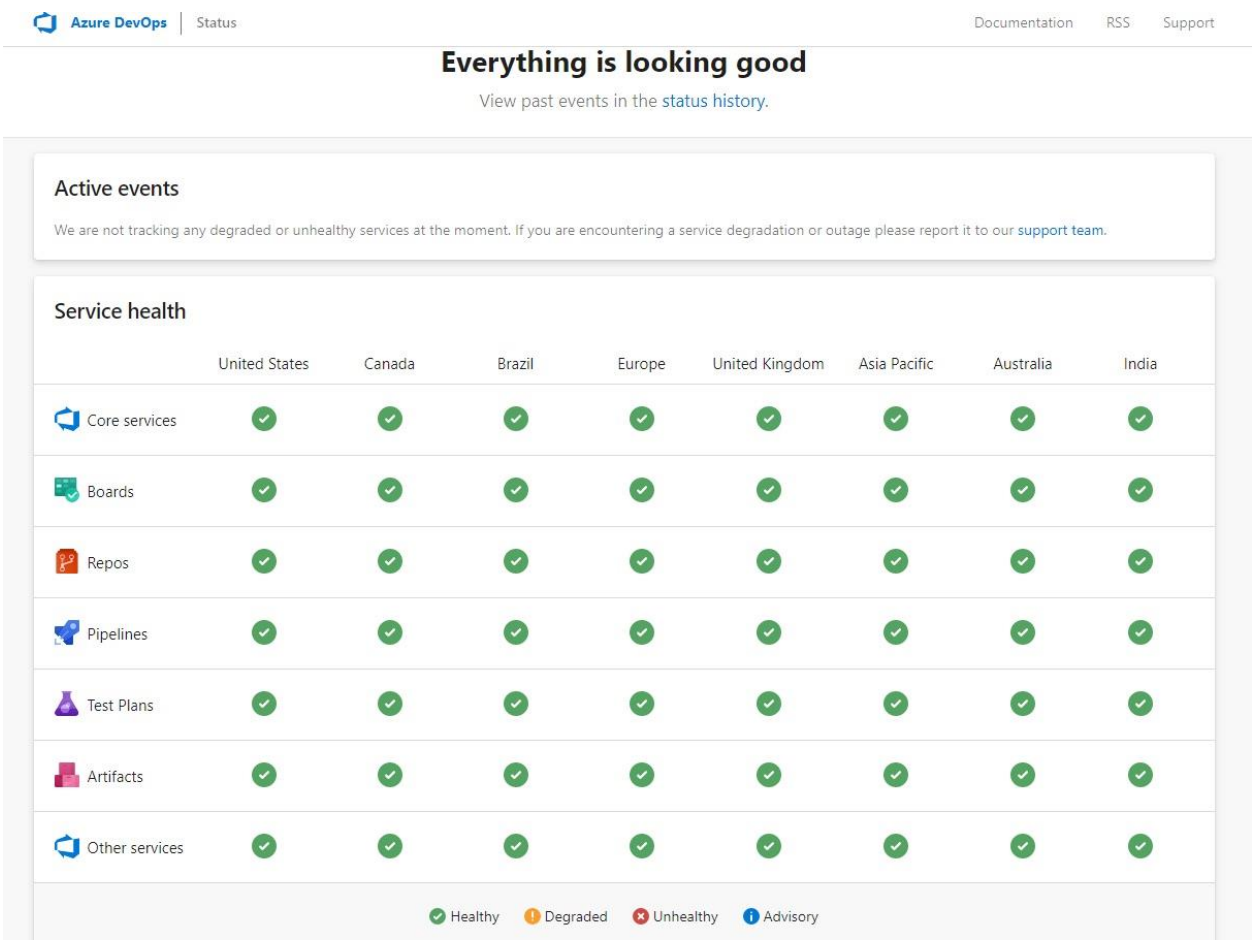


Figure 3. Azure DevOps Services Availability Panel

1.3 The current status of the monitoring system.

Of course, without any monitoring it is impossible to support the cloud application. Therefore, the application for which the monitoring system is designed in this dissertation has a service accessibility panel. As for business metrics and hardware resources analytics, the monitoring has been carried out by system administrators upon request using manual access, which indicates the inconvenience and insecurity of this approach. Moreover, the availability monitor has had a problem with cache overflow and freezing, which has been reflected in displaying irrelevant information.

However, the logging system is configured and functions stably with the use of the ELK stack. ELK includes such components as:

• Elastic Search, NoSQL-repository and search engine based on the Lucene language;

• Logstash, the import location, with a huge number of configurations, through which data enters the Elastic Search storage;

• Kibana, web-based data visualization from Elastic Search.

Logs are added to Logstash through an additional utility from the ELK kit - logstash-forwarder (beats). This tool is a separate service that monitors changes in the file on disk and adds them to Logstash. After that, each line of the file is recognized and sent as fields for indexing and tags to Elastic Search. The result of such transformation can be seen in Figure 4.

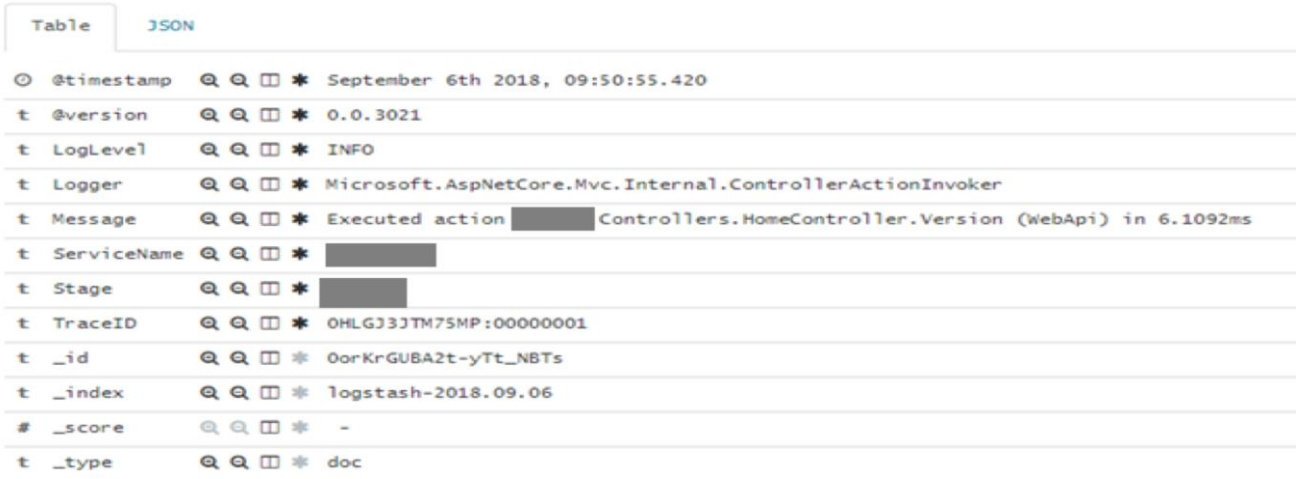


Figure 4 Document in Elastic search

Приложение 2

Файл конфигурации Prometheus

```
global:
  # By default, scrape targets every 15 seconds.
  scrape_interval:     15s

  # Attach these labels to any time series or alerts when
communicating with
  # external systems (federation, remote storage,
Alertmanager).
  external_labels:
    monitor: 'codelab-monitor'

# A scrape configuration containing exactly one endpoint to
scrape:
# Here it's Prometheus itself.
scrape_configs:
  - job_name: 'exampleService'

    # Override the global default and scrape targets from this
job every 5 seconds.
    scrape_interval: 5s

    static_configs:
      - targets: ['exampleService:5001']
```

Приложение 3

Файл docker-compose.yml системы мониторинга

```
version: '3'
  prometheus:
    image: "local/prometheus"
    ports:
    - "9090:9090"
  pgsql:
    image: postgres
  rabbit:
    hostname: rabbit
    image: rabbitmq
  exampleService:
    image: "my.docker.hub/master/exampleService:latest"
    ports:
      - "5000:5000"
      - "5001:5001"
    environment:
      -
ConnectionStrings__DefaultConnection=Server=pgsql;Database=dat
a;Port=5432;Username=postgres;
    depends_on:
      - "pgsql"
      - "rabbit"
    command: >
      /bin/bash -c "
        sleep 30; "
```

Приложение 4

Расширенный docker-compose.yml файл системы мониторинга

```
version: '3'
  graphana:
    environment:
      - GF_INSTALL_PLUGINS=grafana-piechart-panel
    image: grafana/grafana
    ports:
      - "3000:3000"
  prometheus:
    image: "local/prometheus"
    ports:
    - "9090:9090"
  prometheus:
    image: "local/prometheus"
    ports:
    - "9090:9090"
  pgsql:
    image: postgres
  rabbit:
    hostname: rabbit
    image: rabbitmq
  sd:
    image: "my.docker.hub/master/exampleService:latest"
    ports:
      - "5000:5000"
      - "5001:5001"
    environment:
      -
ConnectionStrings__DefaultConnection=Server=pgsql;Database=dat
a;Port=5432;Username=postgres;
    depends_on:
      - "pgsql"
      - "rabbit"
    command: >
      /bin/bash -c "
        sleep 30; "
```

Приложение 5

Исходный код плагина teamcity-datasource

```
///<reference path="../node_modules/grafana-sdk-
   mocks/app/headers/common.d.ts" />

import moment from 'moment';

import {ITarget} from "./typings/queryOptions";
import {
    IBuildFromResponse, BuildFields,
    buildRequests, IExtendedBuildType,
    IFields,
    IHeaders,
    IInstanceSettings,
    IQueryOptions, IRequestOptions, SeriesResult, BuildStates,
   TargetTypes,
} from "./typings/types";

export default class TeamcityDataSource {
    id: number;
    name: string;
    type: string;
    url: string;
    targetTypes: string[];
    fields: IFields;
    headers: IHeaders;
    withCredentials: boolean;
    private backendSrv: any;
    private templateSrv: any;
    private q: any;

    /** @ngInject */
    constructor(instanceSettings: IInstanceSettings,
   backendSrv: any, templateSrv: any, private $q: any) {
        this.name = instanceSettings.name;
        this.id = instanceSettings.id;
        this.type = instanceSettings.type;
        this.url = instanceSettings.url;
        this.withCredentials =
   instanceSettings.withCredentials;

        this.backendSrv = backendSrv;
        this.templateSrv = templateSrv;
```

```
        this.q = $q;

        this.headers = {'Content-Type': 'application/json'};
        if (instanceSettings.basicAuth)
            this.headers['Authorization'] =
instanceSettings.basicAuth;

        this.targetTypes = [TargetTypes.Builds]
        this.fields = {
            build: ['id', 'state', 'status', 'statusText',
'date', 'number', 'name', 'projectName']
        }
    }

    query = (options: IQueryOptions): any => {
        var promises = options.targets
            .map(target => this.queryOneTarget(options,
target));

        return Promise.all(promises)
            .then(results => ({data: [].concat.apply([],
results)}));
    }

    /**
     *
     * @param {IQueryOptions} queryOptions
     * @param {ITarget} target - current target query. @see
collection in IQueryOptions.targets[]
     * @returns {SeriesResult}
     */
    queryOneTarget(queryOptions: IQueryOptions, target:
ITarget): any {
        if (this.targetTypes.indexOf(target.type) == -1) {
            return Promise.reject({status: 'error', message:
`queryOneTarget: Unknown target type ${target.type}`})
        }
        if (this.fields[target.type].indexOf(target.field) ==
-1) {
            return Promise.reject({status: 'error', message:
`queryOneTarget: Unknown field ${target.field}`})
        }
        if (target.type == TargetTypes.Builds) {
            const buildRequestOptions = {
```

```
            buildId: target.target,
            count: queryOptions.maxDataPoints,
            from: queryOptions.range.from,
            to: queryOptions.range.to
        } as buildRequests;

        return this.getBuilds(buildRequestOptions)
            .then(builds => this.mapResult(target,
builds))
    }

    return Promise.reject({status: 'error', message:
`queryOneTarget: Unexpected state`})
 }

 annotationQuery(options) {
    console.log('anotationsOptions: ', options)
    throw new Error("Annotation Support not implemented
yet.");
 }

 /**
  *
  * @param {string} query
  * @returns {any} list view with build names
  */
 metricFindQuery(query: string): any {
    return this.getBuildsList().then(types =>
        types
            .filter(type => type.id.indexOf(query) != -1)
            .map(type => ({text: type.id, value:
type.id}))
    )
 }

 testDataSource = (): Promise<any> => {
    return this.getBuildsList()
        .then((builds: IExtendedBuildType[]) =>
            ({status: 'success', message: `Data source is
working. ${builds.length}`, title: 'Success'})
        )
 }

 /**
```

```
  * Returns last 10 build results for target buildType.
  * @param {buildRequests} request
  * @returns {Promise<IBuildFromResponse>}
  */
 getBuilds = (request: buildRequests):
Promise<BuildFields[]> => {
     var from =
encodeURIComponent(moment(request.from).format("YYYYMMDDTHH
mmssZ"))
     var to =
encodeURIComponent(moment(request.to).format("YYYYMMDDTHHmm
ssZ"))
     var url =
`${this.url}/httpAuth/app/rest/buildTypes/id:${request.buil
dId}`
         +
`/builds?locator=start:0,count:1,defaultFilter:false&`
         +
'fields=build(webUrl,id,state,number,status,statusText,fini
shDate,buildType(name,projectName))'
     return this.doRequest({
         url: url,
         method: 'GET'
     } as IRequestOptions).then(result => {
         return result.data.build.map((build:
IBuildFromResponse) => ({
             id: build.id,
             number: build.number,
             state: build.state,
             status:
this.mapStatusToBuildStates(build.state, build.status),
             statusText: build.statusText,
             date: moment(build.finishDate,
'YYYYMMDDTHHmmssZ').unix() * 1000,
             name: build.buildType.name,
             projectName: build.buildType.projectName
         }) as BuildFields)
     })
 }

 mapStatusToBuildStates = (state: string, status: string):
number => {
     if (state != 'finished') {
```

```
            return state == 'running' ? BuildStates.PENDING :
BuildStates.QUEUED;
        } else if (status != null && status == 'SUCCESS') {
            return BuildStates.SUCCESS;
        }
        return BuildStates.FAILED;
    }

    getBuildsList = (): Promise<IExtendedBuildType[]> => {
        var url =
`${this.url}/httpAuth/app/rest/buildTypes?fields=buildType(
id,name,projectName,projectId)`
        return this.doRequest({
            url: url,
            method: 'GET'
        } as IRequestOptions).then(result => {
            return result.data.buildType.map(type => ({
                id: type.id,
                name: type.name,
                projectId: type.projectId,
                projectName: type.projectName
            }))
        })
    }

    mapResult = (target: ITarget, items: BuildFields[]):
SeriesResult => {
        return {
            target: target.target,
            datapoints: items
                .map(item => [item[target.field], item.date])
        }
    }

    doRequest(options: IRequestOptions) {
        options.withCredentials = this.withCredentials;
        options.headers = this.headers;

        return this.backendSrv.datasourceRequest(options)
    }}
```

Приложение 6

Исходный код плагина rest-api-datasource

```
///<reference path="../node_modules/grafana-sdk-
   mocks/app/headers/common.d.ts" />

import {
    IInstanceSettings,
    IQueryOptions, IRequestOptions, ServiceRequestOptions,
   ServiceResult, ServiceStates,
} from "./typings/types";
import {ITarget} from "./typings/queryOptions";

export default class RestServiceDataSource {
    id: number;
    name: string;
    type: string;
    url: string;
    private backendSrv: any;
    private templateSrv: any;
    private q: any;
    domains: string[];

    /** @ngInject */
    constructor(instanceSettings: IInstanceSettings,
   backendSrv: any, templateSrv: any, private $q: any) {
        this.name = instanceSettings.name;
        this.id = instanceSettings.id;
        this.type = instanceSettings.type;
        this.url = instanceSettings.url;
        this.domains = instanceSettings.jsonData.domains;

        this.backendSrv = backendSrv;
        this.templateSrv = templateSrv;
        this.q = $q;
    }

    /**
     * Return list of domains for dropdown in query editor
     * @returns {string[]}
     */
    getDomains = (): any[] => {
```

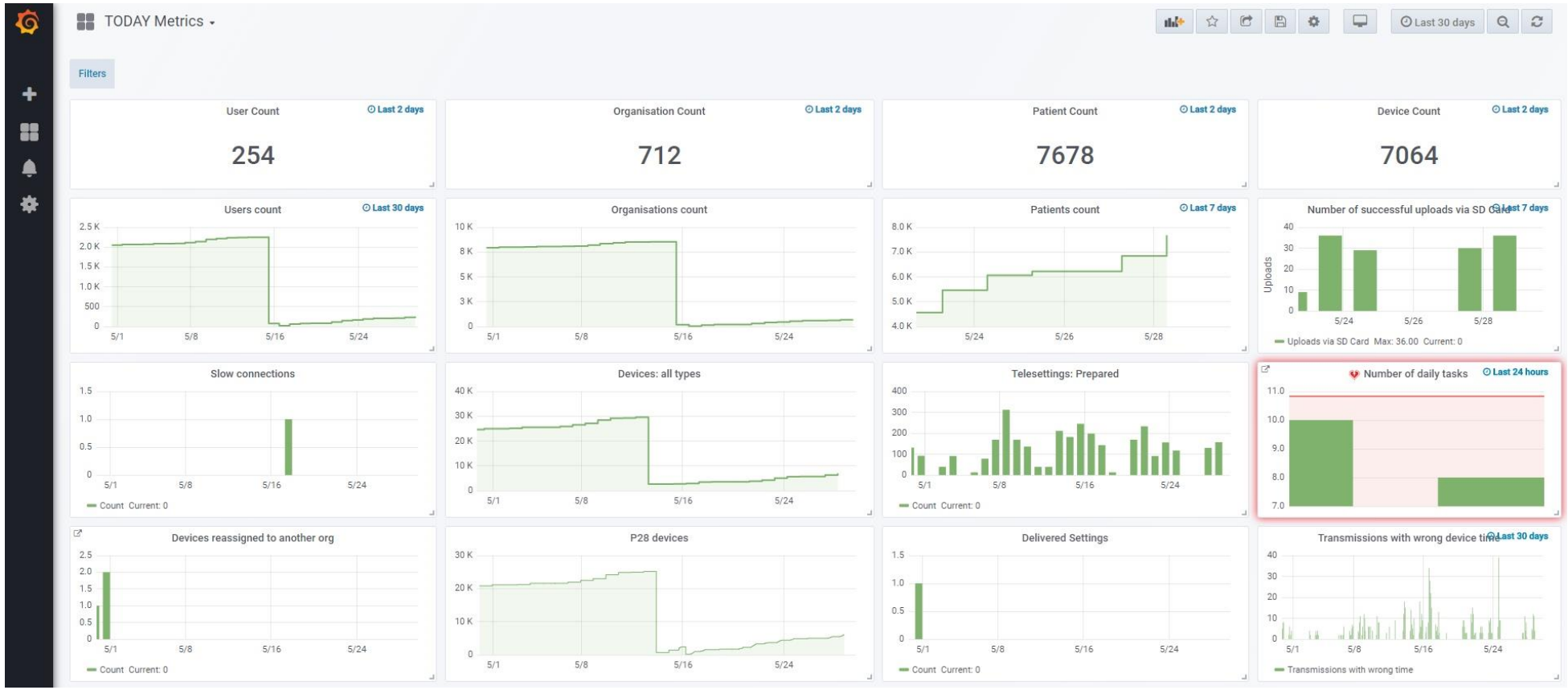
```
        return this.domains
            .map(domain => ({text: domain, value: domain}));
    }

    GetRootAddress = () => {
        return this.url;
    }

    query = (options: IQueryOptions): any => {
        var promises = options.targets
            .map(target => this.queryOneTarget(target));

        return Promise.all(promises)
            .then(results => ({data: [].concat.apply([],
results)}));
    }

    /**
     *
     * @param {IQueryOptions} queryOptions
     * @param {ITarget} target - current target query. @see
collection in IQueryOptions.targets[]
     * @returns {SeriesResult}
     */
    queryOneTarget(target: ITarget): Promise<any> {
        const requestOptions = {
            endpoint: target.endpoint,
            subdomain: target.subdomain
        } as ServiceRequestOptions;

        return this.getServiceVersion(requestOptions)
            .then(result => this.mapResult(target, result))
            .catch(resultError => this.mapResult(target,
resultError));
    }

    mapResult = (target: ITarget, item: ServiceResult): any =>
{
        const httpStatusCode = parseInt(item.status);
        const message = httpStatusCode ===
ServiceStates.SUCCESS
            ? `${item.message}`
            : httpStatusCode;
        return {
```

```
            target: `${target.subdomain} ${item.message}`,
            datapoints: [[message, new Date().getTime()]]
        }
    }

    annotationQuery(options) {
        console.log('anotationsOptions: ', options)
        throw new Error("Annotation Support not implemented
yet.");
    }

    /**
     * Returns last 10 build results for target buildType.
     * @param {buildRequests} request
     * @returns {Promise<IBuildFromResponse>}
     */
    getServiceVersion = (request: ServiceRequestOptions):
Promise<any> => {
        var url = "";
        if (this.url.indexOf("https:") != -1)
            url = `${this.url.replace(/^https:\/\//,
'https://' + request.subdomain +
'.')}/${request.endpoint}`;
        else if (this.url.indexOf("http:") != -1)
            url = `${this.url.replace(/^http:\/\//, 'http://'
+ request.subdomain + '.')}/${request.endpoint}`;
        else
            return Promise.reject({status: 'error', message:
`getServiceVersion: Unknown root protocol ${this.url}`})

        return this.doRequest({
            url: url,
            method: 'GET'
        } as IRequestOptions)
            .then(result => {
                    return {
                        status: result.status,
                        message: result.data
                    } as ServiceResult
                }
            );
    }

    doRequest(options: IRequestOptions) {
```

```
        return this.backendSrv.datasourceRequest(options)
    }
}
```

Приложение 7

Дашборд бизнес-метрик приложения. Количественные показатели

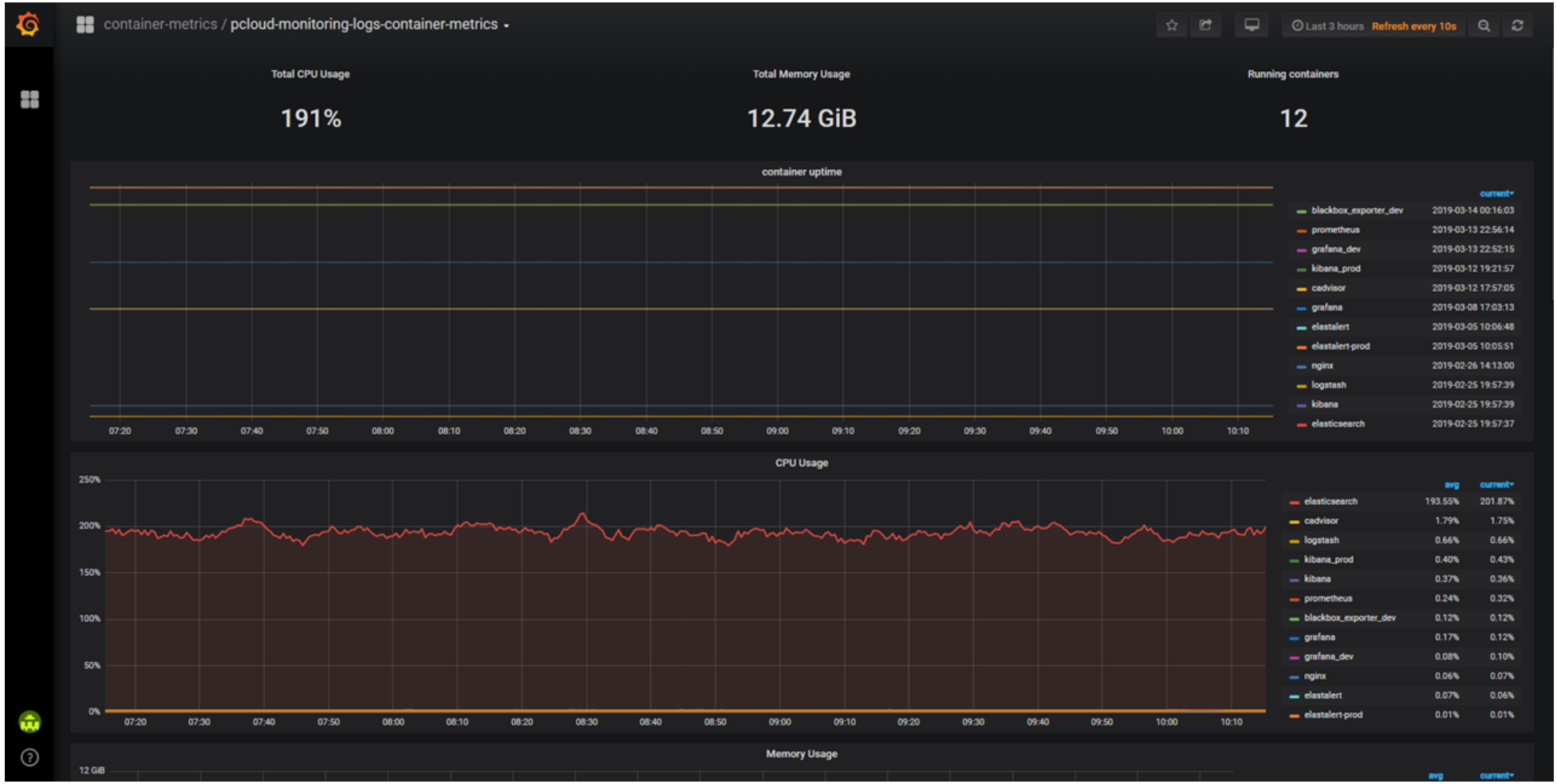


Приложение 8

Метрики использования: количество авторизация со вторым фактором, количество полученных сообщений от теле-устройств, среднее время выполнения запросов

Приложение 9

Дашборд аппаратного мониторинга



Приложение 10

Метрики использования ссылок приложения