МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

РАБОТА ПРОВЕРЕНА
Исполн. директор ООО «Кит Актив»
_____ Д.С Кульбиков
«___»_____ 2020 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____ 2020 г.

Разработка системы по учету и анализу телеметрии медицинского оборудования и ее интеграция с платформой Китактив

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Руководитель работы,
к.ф-м.н., доцент каф. ЭВМ
_____ Г.И. Радченко
«___»_____2020 г.

Автор работы,
студент группы КЭ-222
_____Я.С. Волков
«___»_____2020 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«___»_____2020 г.

Челябинск-2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____2020 г.

ЗАДАНИЕ

на выпускную квалификационную работу магистра

студенту группы КЭ-222
Волкову Ярославу Сергеевичу
обучающемуся по направлению
09.04.01 «Информатика и вычислительная техника»

Тема работы: «Разработка системы по учету и анализу телеметрии медицинского оборудования и ее интеграция с платформой Китактив» утверждена приказом по университету от 24 апреля 2020 г. №627

Срок сдачи студентом законченной работы: 1 июня 2020 г.

Исходные данные к работе:

- Клеппман М. Высоконагруженные приложения. Программирование, масштабирование, поддержка: Отдельное издание / Клеппман Мартин – Питер, 2018 - 740 с.
- Перри Ли, Архитектура интернета вещей: Отдельное издание / Перри Ли - ДМК Пресс, 2018 – 454 с.

Перечень подлежащих разработке вопросов:

- анализ аналогов, разрабатываемой системы;
- детализация набора требований к приложению;
- выбор среды и средств реализации;
- проектирование архитектуры приложения;
- организация базы данных;
- создание приложений первичной и вторичной обработки входной информации с устройств, подключенных к оборудованию;
- создание веб-приложения для предоставления API;
- создание веб-интерфейса пользователя.

Дата выдачи задания: 3 февраля 2020 г.

Руководитель работы _____/*Г.И. Радченко*/

Студент _____/ *Я.С. Волков* /

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение	04.02.2020	
Обзор аналогов	07.02.2020	
Проектирование архитектуры	14.02.2020	
Организация базы данных	21.02.2020	
Создание приложений обработки входной информации	28.02.2020	
Создание веб-приложения для предоставления api	30.03.2020	
Создание веб-интерфейса пользователя	27.04.2020	

Руководитель работы _____/*Г.И. Радченко*/

Студент _____/*Я.С. Волков*/

Аннотация

Я.С. Волков. Разработка системы по учету и анализу телеметрии медицинского оборудования и ее интеграция с платформой Китактив. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2020, 160 с., 69 ил., библиогр. список 28 наим.

В рамках выпускной квалификационной работы создается система по учету и анализу телеметрии медицинского оборудования для ее дальнейшей интеграции с платформой Китактив. Главное и основное предназначение системы — это предоставление актуальной информации с показателей устройств, подключенных к медицинскому оборудованию, а также контролю выхода значений показателей за пределы нормы, с уведомлением об этом уполномоченных лиц. Ожидаемым результатом работы данной системы будет повышение качества обслуживания клиентов медицинских учреждений, продление срока эксплуатации медицинского оборудования, а также предотвращение денежных потерь медицинских учреждений.

ОГЛАВЛЕНИЕ

ГЛОССАРИЙ

Термины	Определения
Диагноз	Краткое медицинское заключение о том, чем болен человек и каково его состояние.
Инцидент	Отказ или повреждение технического устройства, отклонение от установленного режима технологического процесса.
Инцидент (плановый)	Работы, которые нужно выполнять регулярно, для того чтобы поддерживать оборудование в рабочем состоянии. Эти работы обычно описаны в его паспорте. Данные работы могут выполняться как каждый день, так и раз в два года.
Инцидент (внеплановый)	Инциденты, которые возникают неожиданно. Это может быть какая-либо поломка или выход из строя, также это может быть событием, приходящим с датчиков, подключенных к медицинскому оборудованию и сигнализирующих о том, что какой-то из его показателей вышел за пределы нормы. Данные события могут быть звоночком для инженеров о том, что оборудование нужно осмотреть и проверить на корректную работу.
Медицинское оборудование	Изделие медицинской техники, назначение которого – обеспечить оптимальные условия для больного, а также для медицинского персонала при проведении мероприятий лечебно-диагностического характера и по уходу за больными.
Mosquitto	Брокер сообщений с открытым исходным кодом (лицензированный EPL / EDL), который реализует протоколы MQTT версий 5.0, 3.1.1 и 3.1.
MQTT	Простой и легкий протокол обмена сообщениями, предназначенный для ограниченных устройств и сетей с низкой пропускной способностью, с высокой задержкой или ненадежностью.

Источники измерения	Данные описывающие топики, проходящие с устройств
Точки измерения	Необработанные данные приходящих измерений с устройств, подключенных к медицинскому оборудованию
Каналы измерения	История изменений показателей точек измерения
Типы точки измерения	Данные о характеристиках точек измерения
Точки учета	Обработанные данные приходящих измерений с устройств, подключенных к медицинскому оборудованию
Каналы учета	История изменений показателя точек учета
Типы точки учета	Данные о характеристиках точек учета
Правила учета	Описание правил обработки показателей точек учета
Категории учета	Описывает данные категорий типов точек учета

ВВЕДЕНИЕ

Качественные медицинские услуги на сегодняшний день не могут обойтись без применения профессионального оборудования. Различная техника позволяет диагностировать множество заболеваний, проводить терапию, лечение, и реабилитационные процедуры. Таким образом, на рынке представлен широкий ассортимент данной продукции, которую можно использовать в тех или иных специализациях.

Здоровье иногда имеет свойство подводить, и причину этого способен найти только квалифицированный специалист с богатым опытом. Диагностические центры и современные клиники сегодня оснащаются всем необходимым, чтобы предоставлять сервис на высшем уровне.

Медицинское оборудование бывает разных видов от медицинских мониторов до таких сложных, как аппараты поддержания жизни пациентов. От правильной и надежной работы данных устройств зависит не только его долговечность, но и жизнь пациентов.

Проблема заключается в том, что иногда при неправильной эксплуатации или по мере использования оборудования, могут возникать плановые и внеплановые инциденты.

Существует компания Китактив, которая занимается разработкой узконаправленного программного продукта в сфере медицины, в котором ведется учет инцидентов, ремонтов и других бизнес-процессов, связанных с медицинским оборудованием. Руководство данной компании поставило задачу разработать систему, которая будет вести учет и анализ телеметрии, приходящей от медицинских активов, с целью внедрения в свой программный продукт «Китактив».

В данной работе отражен обзор и анализ существующих систем телеметрии, требования для создания нового решения в области медицинского оборудования, разработка архитектуры системы, проектировка базы дынных, разработка двух приложений первичной и вторичной обработки информации, приходящей от устройств, подключенных к оборудованию, разработка веб-приложения для предоставления API, а также разработка пользовательского интерфейса. Данная система будет вести учет, позволять просматривать актуальную информацию и своевременно анализировать телеметрию, приходящую с медицинского оборудования, тем самым поддерживая его в рабочем состоянии, что является важной и актуальной задачей на сегодняшний день.

Целью данной выпускной квалификационной работы является разработка системы по учету и анализу телеметрии медицинского оборудования, также ее интеграция с платформой Китактив. Система предназначена для предоставления актуальной информации с показателей устройств, подключенных к медицинскому оборудованию, а также контролю выхода значений показателей за пределы нормы, с уведомлением об этом уполномоченных лиц.

Для достижения поставленной цели, необходимо решить следующие задачи:

1) выполнить анализ аналогов, разрабатываемой системы;
2) детализировать набор требований к приложению;
3) выбрать среду и средства реализации;
4) спроектировать архитектуру приложения;
5) организовать базу данных;
6) создать приложения первичной и вторичной обработки входной информации с устройств, подключенных к оборудованию;

7) разработать backend приложение для предоставления информации по получаемым измерениям;
8) разработать интерфейс пользователя;
9) выполнить развертывание и тестирование системы.

Данная работа состоит из 5 глав, заключения и библиографического списка.

В первом разделе, «Анализ предметной области», рассмотрены основные существующие решения на рынке систем, предназначенных для учета и анализа телеметрии медицинского оборудования.

Во втором разделе, «Определение требований», определены основные функциональные и нефункциональные требования, предъявляемые к разрабатываемой системе.

В третьем разделе, «Проектирование», приведены архитектура нового решения и алгоритмы решения задачи.

В четвертом разделе, «Реализация», приведена реализация основных компонентов разрабатываемой системы.

В пятом разделе, «Результаты и тестирование», представлены результаты тестирования, показывающие правильную работоспособность нового решения, и демонстрацию его работы.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. АКТУАЛЬНОСТЬ

Для того, чтобы установить диагноз врач проводит объективный осмотр (осуществляет аускультацию, перкуссию и пальпацию), который дает ему дополнительную информацию. И нередко для точного установления или подтверждения диагноза требуются специальные диагностические процедуры, которые осуществляются с помощью современного медицинского оборудования [1].

Мониторинг медицинского оборудования — это не только очень важный процесс, но иногда даже важный для спасения жизни людей. Чрезвычайные случаи, когда каждая секунда получает данные о здоровье пациента и его статистику, что может быть вопросом жизни или смерти. Надежное решение для мониторинга делает необходимый поток данных максимально быстрым и экономит время специалистов для других жизненно важных процедур.

Каждое медицинское учреждение имеет сотни устройств и оборудования, которые помогают контролировать и показывать текущее состояние здоровья пациента. Само собой разумеется, что мониторинг этих устройств и быстрое уведомление в случае любых сбоев или выпадений имеет жизненно важное значение.

Еще одной важной целью процесса мониторинга в сфере здравоохранения и медицины является предотвращение любых денежных потерь, которые можно сэкономить для развития медицинских учреждений и поддержки нуждающихся людей. В этом случае цель процесса мониторинга направлена на выявление того, были ли запланированные результаты достигнуты, как это определено в плане и стратегии действий в области здравоохранения.

Такой мгновенный контроль над любыми возможными сбоями в общей системе помогает медицинским учреждениям не только правильно распределять свои средства, но и обеспечивать точную работу медицинского оборудования.

1.2. ОБЗОР АНАЛОГОВ

При поиске систем мониторинга телеметрии было найдено несколько примеров. Данные образцы можно получить как на территории Российской Федерации, так и за ее пределами. Достоинства и недостатки некоторых популярных систем будут описаны ниже.

В качестве аналогов были выбраны следующие системы:

- Graphite [2];
- Prometheus [3];
- МТС Телеучет [4];
- Grafana [5].

Graphite – это система агрегации данных и отображения графиков в реальном времени. Пример интерфейса системы приведен на рисунке 1.

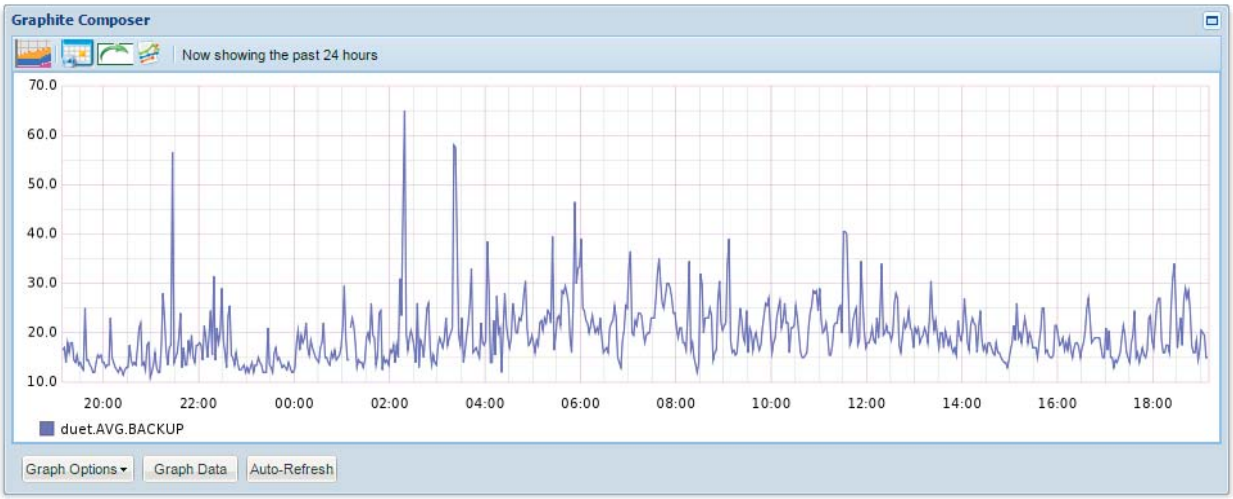

Рисунок 1 – Интерфейс Graphite

Prometheus – система мониторинга различных систем и сервисов, с помощью которой системные администраторы могут собирать информацию о текущих параметрах. Пример интерфейса системы приведен на рисунок 2.

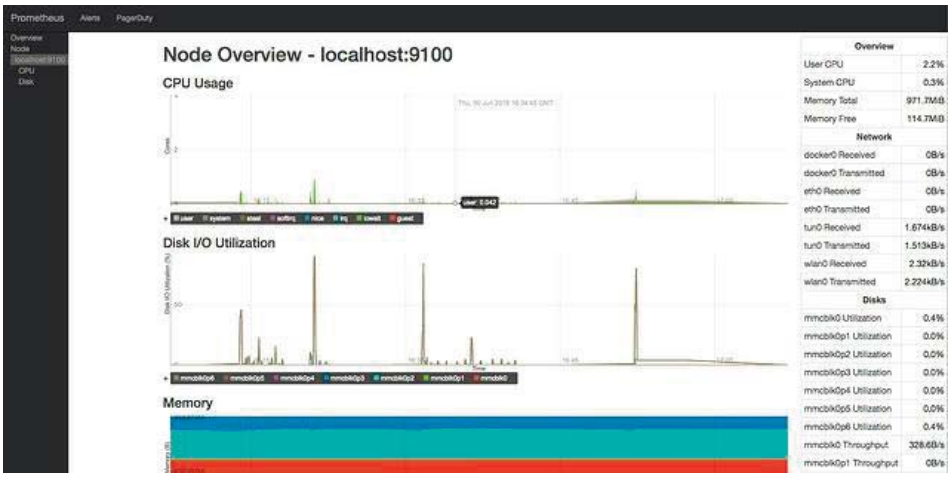

Рисунок 2 – Интерфейс Prometheus

МТС Телеучет – система, позволяющая производить мониторинг удаленных объектов, учет энергоресурсов и дистанционный контроль технологического оборудования. Пример интерфейса системы приведен на рисунок 3.

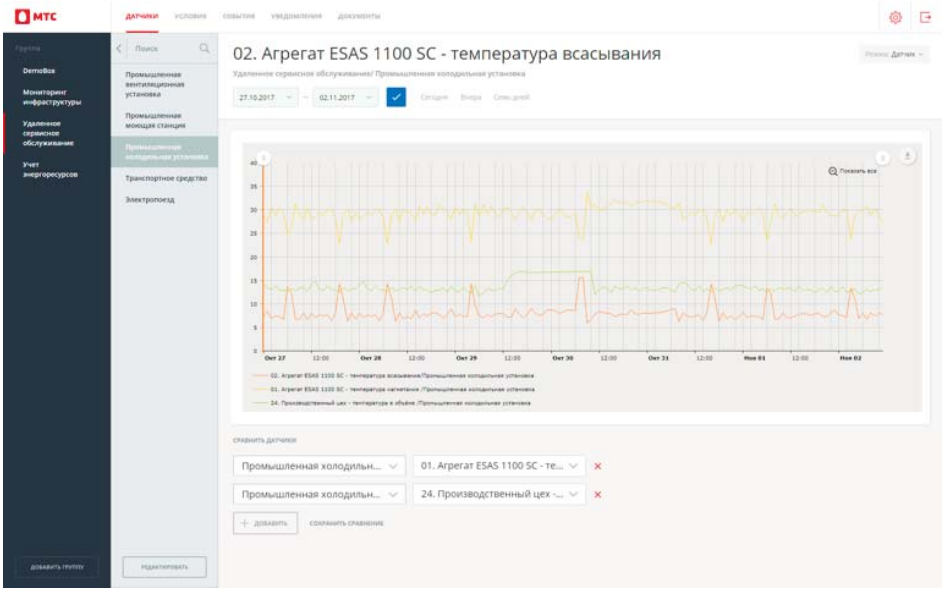

Рисунок 3 – Интерфейс МТС Телеучет

Grafana – это платформа с открытым исходным кодом для визуализации, мониторинга и анализа данных. Пример интерфейса системы приведен на рисунок 4.

Рисунок 4 – Интерфейс Grafana

Авторами [6, 7], проводившими подробный обзор данных систем, были определены следующие достоинства и недостатки данных систем.

Система Graphite имеет хорошие параметры визуализации без возможности редактирования приборной панели, а также имеется возможность хранить временные ряды. Однако у нее отсутствует поддержка сбора данных и прямая сигнализация о тревогах.

Система Prometheus обладает возможностью хранить временные ряды, выполнять сбор данных и сигнализировать о возникновении событий. Однако

присутствует сложность с использованием функций редактирования графиков и приборной панели.

Система МТС Телеучет умеет выполнять сбор данных, хранить временные ряды, сигнализировать о возникновении событий. Однако имеет не богатый функционал по возможностям мониторинга.

Система Grafana способно многофункционально, просто и гибко визуализировать данные. Однако не поддерживает хранение временных рядов, сбора данных и управления тревогами с отслеживанием событий.

По результатам анализа, рассмотренных выше аналогов, была составлена таблица Таблица 1, позволяющая провести сравнительный анализ систем.

Таблица 1 – Сравнение аналогов

Необходимые требования	Graphite	Prometheus	МТС Телеучет	Grafana
Сбор данных	-	+	+	-
Хранение данных	+	+	+	-
Богатое и информативное отображение данных	+	+	-	+
Простота работы с решением	-	-	-	+
Возможность уведомления о критических ситуациях	-	+	+	-

Исходя из обзора аналогов в таблице Таблица 1, видно, что каждый из аналогов имеет свои достоинства и недостатки, имея часть необходимой функциональности. В новом решении необходимо учесть достоинства и недостатки аналогов, а также требования заказчика, которые будут отражены далее в соответствующей главе.

1.3. АНАЛИЗ ОСНОВНЫХ ТЕХНОЛОГИЧЕСКИХ РЕШЕНИЙ

1.3.1. Backend

1.3.1.1. Выбор протокола и сервера обмена сообщениями

Инженеры компании работают с оборудованием, которое в качестве сервера обмена сообщениями используют Mosquitto, который свою очередь построен на протоколе MQTT [8]

MQTT – это MQ Telemetry Transport, который представляет собой простой и легкий протокол обмена сообщениями, предназначенный для ограниченных устройств и сетей с низкой пропускной способностью, с высокой задержкой или ненадежностью. Разработан на принципах минимизации пропускной способности сети и требований к ресурсам устройств, пытаясь в то же время обеспечить надежность и некоторую степень уверенности в доставке. Эти принципы также делают этот протокол идеальным для появления мира подключенных устройств типа «машина-машина» (M2M) или «интернет вещей», а также для мобильных приложений, где крайне важны пропускная способность и заряд батареи [9]. Компания Facebook использует протокол MQTT для своего механизма обмена сообщениями.

Для того, чтобы получать и перенаправлять сообщения с устройств необходимо установить MQTT сервер или брокер. В нашем случае таковым является Mosquitto [10].

Mosquitto - брокер сообщений с открытым исходным кодом (лицензированный EPL / EDL), который реализует протоколы MQTT версий 5.0, 3.1.1 и 3.1. Mosquitto легок и подходит для использования на всех устройствах от одноплатных компьютеров с низким энергопотреблением до мощных серверов.

Также при сравнении показателя пропускной способности Mosquitto с конкурентами, данный брокер становиться победителем [11]. Диаграмма

сравнения показателя пропускной способности Mosquitto с конкурентами представлена на рисунок 5.

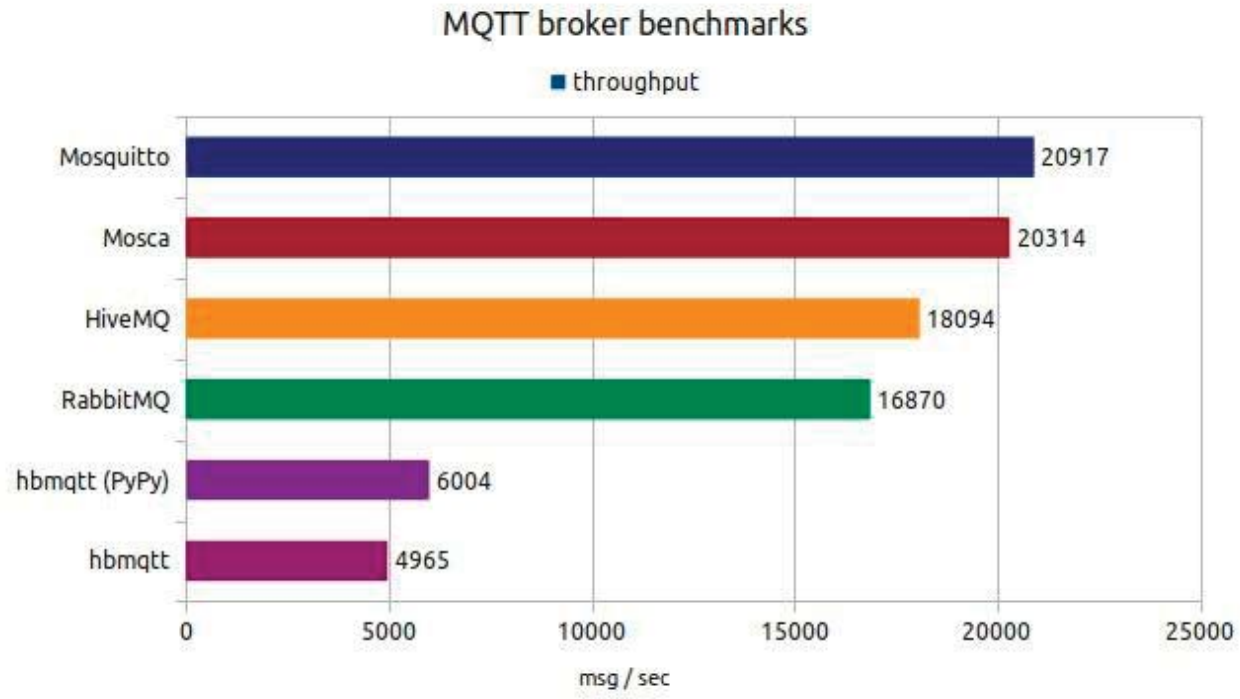

Рисунок 5 – Сравнение MQTT брокеров

При этом хочется отметить его простоту и скорость развертывания сервера Mosquitto. Настройка и запуск сервера при стандартных настройках может занять меньше минуты. Также на сегодняшний день для работы с данным брокером есть поддержка у основных языков программирования, таких как: C/C++, Java, JavaScript, Python, Go и C#, что существенно расширяет область его применимости.

Именно поэтому в качестве сервера обмена сообщениями для связи с устройствами, а также передачи информации между внутренними приложениями будет использоваться Mosquito.

1.3.1.2. Выбор системы управления базой данных

Существует довольно много систем управления базами данных, такие как:

- Microsoft SQL Server [13];
- MongoDB [14];
- PostgreSQL [15];
- MySQL [16].

Автором [17] был проведен подробный анализ ранее перечисленных СУБД. С помощью него можно выделить основные достоинства и недостатки систем управления.

СУБД MS SQL Server является простым в использовании продуктом, с которым можно получить доступ к визуализации на мобильных устройствах, а также хорошо взаимодействовать с другими продуктами компании Microsoft. Однако цена для использования юридическими лицами является неприемлемой для большей части организаций, вместе с этим, даже при тщательной настройке производительности SQL сервер способен занять все доступные ресурсы, а также сообщается о проблемах с использованием службы интеграции для импорта файлов. Данная СУБД идеально подходит для крупных организаций, которые уже используют ряд продуктов Microsoft.

СУБД MongoDB отличается хорошей скоростью и простотой в использовании, поддержкой json и других традиционных документов NoSQL, также данные любой структуры могут быть сохранены/прочитаны быстро и легко. Однако у нее не используется SQL в качестве языка запросов, и сама программа установки может занять много времени. Данная СУБД подходит для организаций, работающих с разнородными данными, которые тяжело поддаются классификации. Для внедрения потребуются высококлассные специалисты.

СУБД PostgreSQL имеет поддержку: формата json, множества предопределенных функций и ряда интерфейсов. Однако вместе с этим и

сложные документацию и конфигурацию для неопытного пользователя. Также скорость работы может падать во время проведения пакетных операций или выполнения запросов чтения. Данная СУБД идеально подходит для организаций с ограниченным бюджетом, но квалифицированными специалистами.

СУБД MySQL включает в себя: бесплатное распространение, простую установку, понятную документацию, большое количество функций, набор пользовательских интерфейсов, работу с другими базами данных. Однако у нее отсутствует встроенная поддержка XML или OLAP, а также для бесплатной версии доступна только платная поддержка. Данная СУБД идеально подходит для организаций, которым требуется надежный инструмент управления базами данных, но бесплатный.

Проанализировав достоинства и недостатки СУБД, хочется отметить сбалансированность решения MySQL, поэтому именно эта база данных будет использоваться в разработке системы.

1.3.1.3. Выбор языка программирования для создания приложений первичной и вторичной обработки

Для данной системы необходимо написать два приложения: первичной и вторичной обработки информации.

Первичная обработка подразумевает обработку входных данных с устройств, записи этих данных в соответствующие таблицы и передача пакета информации во внутреннюю очередь сообщений для вторичной обработки.

Вторичная обработка подразумевает под собой осуществление несложных вычислений со значениями, пришедшими после первичной обработки, по определенным правилам, а также отлавливание ситуаций выхода значения за пределы.

Для сравнения возьмем три популярных языков программирования для создания backend приложений:

- Python [18];
- JavaScript (Node JS) [19];
- Java [20].

Все три языка на сегодняшний день популярны среди компаний и разработчиков.

Заказчиком было поставлено условие при выборе языка программирования обратить внимание на стоимостные показатели, а также на скорость разработки.

Авторами [21, 22, 23] был произведен подробный анализ ранее перечисленных языков программирования. С помощью него можно выделить их достоинства и недостатки.

ЯП Python совмещает в себе: простой синтаксис, широкий спектр инструментов разработки и большое сообщество. Однако он является однопоточным и не подходит для мобильных вычислений.

ЯП JS (Node JS) имеет: высокую производительность, легкость и скорость написания: легковесность, простоту разработки, огромное количество библиотек и постоянное развитие. При этом, в виду быстрого развития, необходимо постоянно следить за обновлениями, соблюдать четкую архитектуру и уметь выбирать необходимую библиотеку из большого количества.

ЯП Java включает: высокую скорость работы, хорошую распространенность и огромный выбор библиотек. Однако он является тяжеловесным и имеет медленное развитие, по сравнению с аналогами.

Подводя итог хочется отметить, что все три языка на сегодняшний день хорошо развиты по своему, если конкретно говорить про данный проект, то учитывая все достоинства и недостатки, а также условия и требования заказчика, то выбор падает на язык программирования JavaScript, с его платформой Node JS.

1.3.1.4. Выбор фреймворка для создания веб-приложения предоставления API

Существует большое количество фреймворков для создания веб-приложений. Для анализа были взяты следующие:

- Yii2 [24];
- Laravel [25];
- Zend framework [26].

Автором [27] был произведен подробный анализ ранее перечисленных фреймворков. С помощью него можно выделить их достоинства и недостатки.

Фреймворк Yii2 имеет: визуальный генератор кода, легкое обучение с нуля, использует стандартные способы решения задач, легкую настройку, нетребовательность к ресурсам. Однако не имеет гибкое формирование роутов, а также работает со слишком склеенными библиотеками для фронтенда с бэкендом.

Фреймворк Laravel включает в себя: удобную систему миграций, интегрированную систему модульного тестирования, встроенный шаблонизатор Blade, очень гибкое формирование роутов, гибкие возможности для написания REST API, быстрое развитие, большое количество документации на любую тему, консоль отладки из коробки со стеком вызовов. Однако большой функционал работает через фасады, и IDE-системы не видят методов и свойств в некоторых классах, показывая предупреждения. Также нет встроенных генераторов интерфейсов.

Фреймворк Zend содержит: наследование классов, объектно-ориентированную сущность, готовые решения для множества задач, хорошую интеграцию, хорошую документацию и поддержку сообщества. Однако он не подходит для быстрого развития, медленнее аналогов (узкое место БД), требует много времени для изучения, а также достаточно ресурсоемкий.

Разобрав и проанализировав достоинства и недостатки фреймворков, хочется выделить Laravel, за счет его быстрого развития, а также гибкого формирования роутов, что является важным для формирования API, поэтому именно этот фреймворк будет использоваться в разработке нового решения.

1.3.1.5. Выбор средства для разработки пользовательского интерфейса

В решении компании Китактив весь пользовательский интерфейс написан с помощью JavaScript библиотеки React JS [28].

На данный момент она является одной из лучших и быстроразвивающихся библиотек для создания огромных интерфейсов веб-приложений, которая включает в себя следующие преимущества: легкость в изучении, высокий уровень гибкости и отзывчивости, виртуальная DOM, хорошая работа при высоких нагрузках, а также невероятно легкий вес.

Просуммировав совокупность выше перечисленных факторов и умножив на количество сокращенного времени при интеграции данной библиотеки с какой-либо еще можно сделать вывод о том, что именно ее мы будем использовать в разработке новой системы для создания пользовательского интерфейса.

1.4. ВЫВОД

Анализ предметной области дал широкое представление о существующих решениях в данной области. Было проведено сравнение 4 систем мониторинга телеметрии, таких как: Graphite, Prometheus, МТС Телеучет и Grafana. Каждое из них имеет свои преимущества и недостатки, которые будут учтены вместе с пожеланиями заказчика, при создании новой системы.

Для реализации системы KitData был произведен подбор технологических решений для реализации следующих компонентов:

- Система управления базой данных. Произведен анализ СУБД: Microsoft SQL Server, MongoDB, PostgreSQL, MySQL;
- Язык программирования для создания приложений первичной и вторичной обработки. Произведен анализ языков: Python, JavaScript (Node JS), Java;
- Фреймворк для создания веб-приложения предоставления API. Произведен анализ фреймворков: Yii2, Laravel, Zend framework.

В результате получился следующий состав технологических решений:

- MySQL – в качестве системы управления базами данных;
- Mosquitto – в качестве сервера обмена сообщениями;
- JavaScript (Node JS) – в качестве языка программирования для разработки приложений первичной и вторичной обработки;
- Laravel в качестве фреймворка для создания веб-приложения предоставления API;
- React JS в качестве библиотеки для создания пользовательского интерфейса.

2. ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

В данной работе необходимо разработать систему по учету и анализу телеметрии, приходящей от медицинского оборудования. После этого необходимо связать ее с системой Китактив, где пользователи смогут знакомиться с актуальной информацией по их парку оборудования, подключенного к системе телеметрии.

Прежде, чем перейти к проектированию системы необходимо определить функциональные и нефункциональные требования. Данные требования описаны ниже.

2.1. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

Построим диаграмму прецедентов для того, чтобы выделить функциональные требования к системе. Данная диаграмма показывает события, возникающие в системе, она хорошо отражает прагматики разрабатываемой системы, а также описывает структуру компонентов, из которых она состоит, ее атрибуты и взаимодействия. Диаграмма в UML нотации, изображена на рисунок 6.

Рисунок 6 – Диаграмма прецедентов системы KitData

Ключевыми акторами, взаимодействующими с системой KitData являются: устройства, подключенные к медицинскому оборудованию, приложение Kitactive и пользователь системы.

Устройства, подключенные к медицинскому оборудованию, взаимодействуют с сервером KitData для того, чтобы передать данные с устройств, подключенных к медицинскому оборудованию.

Взаимодействие *пользователя* с сервером Kitdata происходит с помощью веб-интерфейса *приложения Kitactive*, где он имеет возможность совершить:

- *просмотр реестра оборудования*, подключенного к телеметрии;
- *просмотр актуальной информации* сенсоров конкретного оборудования, подключенного к телеметрии;
- *просмотр учета наработки оборудования*, подключенного к телеметрии;

- *просмотр учета энергопотребления оборудования*, подключенного к телеметрии;
- *просмотр количества работающего оборудования*, подключенного к телеметрии, в момент запроса;
- *создание регламентной задачи* по наработке оборудования.

2.2. НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

2.2.1. Требования к системе в целом

1) Обеспечить актуальность данных показателей с устройств, подключенных к оборудованию;
2) Обеспечить доступ к модулю телеметрии пользователям, имеющим следующие роли:
   - старшая медсестра;
   - инженер;
   - руководитель инженерной службы;
   - главный врач.

2.2.2. Требования к совместимости с другими системами

Разрабатываемая система должна обеспечивать совместимость с программным решением компании Китактив и предоставлять необходимую информацию по запросу.

2.2.3. Требования квалификации персонала

Для работы с системой пользователь должен иметь базовые навыки работы с компьютером.

2.2.4. Требования к надежности и безопасности

Доступ к системе должны иметь только пользователи, зарегистрированные в программе Китактив, в соответствии с правами доступа.

Права доступа должны иметь возможность изменяться для каждой роли в конкретной компании, кроме пользователей с ролью среднего медицинского персонала, которые не должны иметь доступ к модулю телеметрии.

2.2.5. Требования к интерфейсу

Для взаимодействия пользователя с системой будет добавлен пользовательский интерфейс в приложение Китактив. При создании будут учитываться лучшие практики построения пользовательского интерфейса (простота, согласованность элементов, сообщение системы о том, что происходит и т.д.), для того чтобы он однозначно воспринимался пользователями и не требовал дополнительного обучения.

2.2.6. Требования к лингвистическому обеспечению

Пользовательский интерфейс должен быть на русском языке.

2.2.7. Требования к эргономике и технической эстетике

Разрабатываемая система должна уметь быстро принимать, обрабатывать и анализировать информацию, при этом оповещая ответственных лиц о критических ситуациях. Так же с помощью системы должно быть доступно и просто просматривать текущую информацию по медицинскому оборудованию и строить отчеты.

2.3. ВЫВОД

В данной главе были выдвинуты требования к разрабатываемой системе: функциональные и нефункциональные. Для выделения функциональных требований была разработана диаграмма прецедентов. В разделе нефункциональные требования были описаны следующие требования: к системе в целом, к совместимости с другими системами, квалификации персонала, к надежности и безопасности, к интерфейсу, к лингвистическому обеспечению и к эргономике и технической эстетике.

3. ПРОЕКТИРОВАНИЕ

3.1. АРХИТЕКТУРА ПРЕДЛАГАЕМОГО РЕШЕНИЯ

Для описания состава компонентов системы можно представить диаграмму компонентов, которая поясняет как отдельные виды процессов протекают в целостных функциональных блоках. Данная диаграмма приведена на рисунок 7.

На диаграмме компонент *устройства* взаимодействует с компонентом *внешний Mosquitto сервер* системы *KitData*, передавая сообщения о новых измерениях. Данные сообщения прослушивает *компонент первичной обработки данных*, который после обработки сохраняет полученные данные в *Базе данных* и передает их дальше в *компонент вторичной обработки* данных через компонент *внутренний Mosquitto сервер*. После того, как данные были вторично обработаны они сохраняются в *Базу данных* и, при выявлении критической ситуации отправляются в *приложение Kitactive*.

Пользовательские запросы обрабатываются *приложением Kitactive*, затем сами данные запрашиваются на у *компонента предоставления данных по пользовательским запросам* (веб-приложение предоставления API) системы *KitData*.

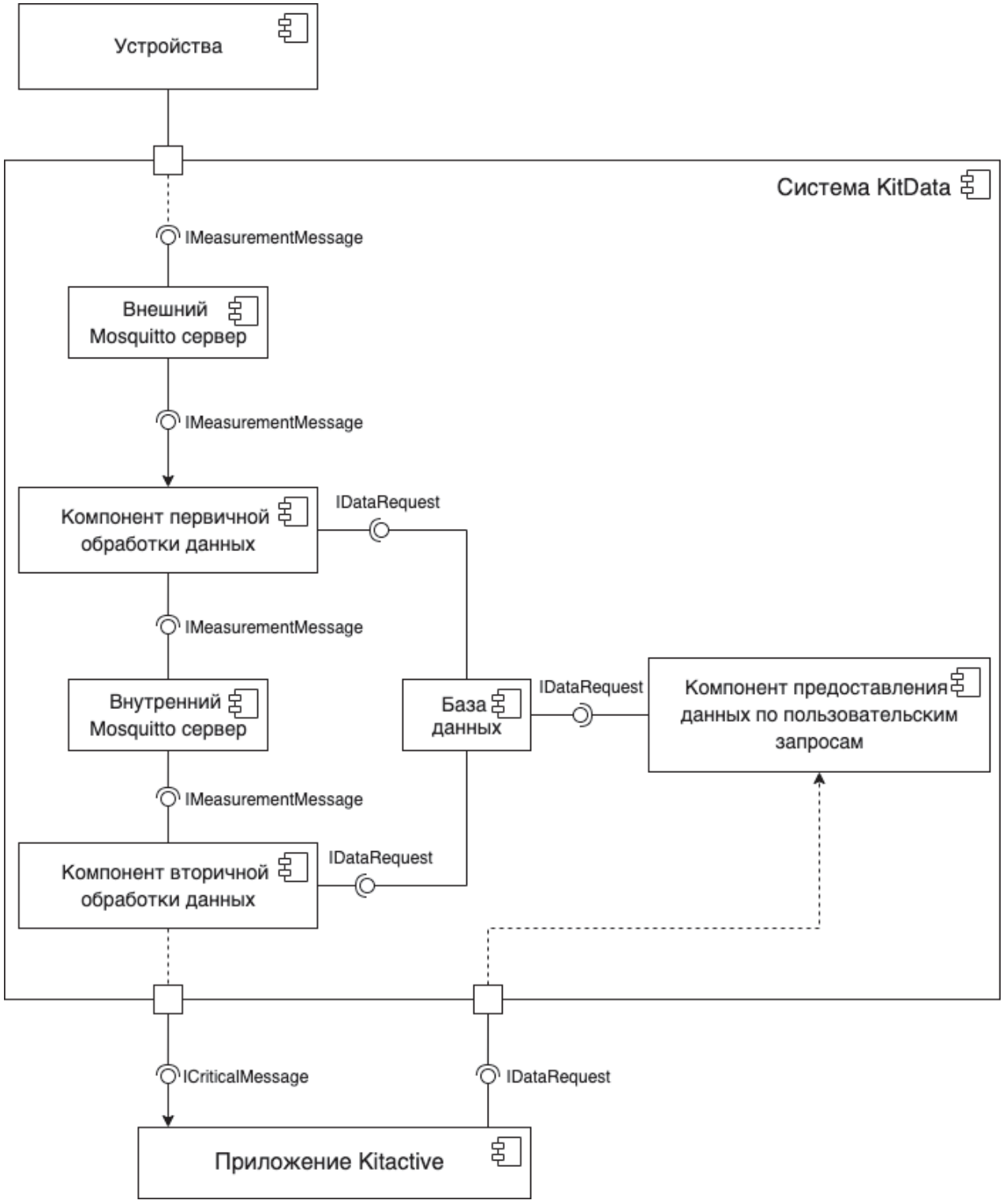

Рисунок 7 – Диаграмма компонентов системы KitData

3.2. АЛГОРИТМЫ ОБЕСПЕЧЕНИЯ УЧЕТА И АНАЛИЗА ТЕЛЕМЕТРИИ

3.2.1.Описание работы приложений первичной и вторичной обработки

Прежде чем начать разрабатывать приложения первичной и вторичной обработки необходимо разработать алгоритмы последовательности действий.

Данные алгоритмы будем представлять в виде блок-схем. Блок-схема алгоритма работы приложения первичной обработки представлена на рисунок 8.

Рисунок 8 – Блок-схема алгоритма работы приложения первичной обработки

На блок-схеме, представленной выше, отображены основные пункты алгоритма первичной обработки данных, таких как:

- получение сообщения о новом измерении от устройств, подключенных к медицинскому оборудованию;
- первичная обработка полученных данных, под которой подразумевается принятие пакета сырых данных и при необходимости разбитие его на отдельные измерения;
- запись данных, полученных в результате первичной обработки, в БД;
- передача данных для вторичной обработки.

Блок-схема алгоритма работы приложения вторичной обработки представлена на рисунок 9.

Рисунок 9 – Блок-схема алгоритма работы приложения вторичной обработки

На блок-схеме, представленной выше, отображены основные пункты алгоритма вторичной обработки данных, таких как:

- получение сообщения о новом измерении от приложения первичной обработки;
- вторичная обработка полученных данных, под которой подразумевается расчет значений измерений по определенным правилам, при необходимости сравнение с определенными уставками;
- запись данных, полученных в результате вторичной обработки, в БД;
- при возникновении критической ситуации отправка об этом сообщения во внешний сервер Kitactive.

3.2.2. Описание вариантов взаимодействия пользователей с системой

На диаграмме прецедентов, изображенной на рисунок 6, пользователь представляют собой человека, взаимодействующего с системой Kitdata через приложение Kitactive. Прежде чем начать разрабатывать приложение предоставления API и пользовательский интерфейс необходимо описать алгоритмы вариантов взаимодействия пользователя с системой.

Вариант использования «Просмотр реестра оборудования, подключенного к телеметрии». Данный вариант использования позволяет просмотреть реестр оборудования, подключенного к телеметрии. Диаграмма последовательностей варианта использования «Просмотр реестра оборудования, подключенного к телеметрии» представлена на рисунок 10.

Рисунок 10 – Диаграмма последовательностей варианта использования «Просмотр реестра оборудования, подключенного к телеметрии»

Основной поток событий состоит из четырех шагов:

1) *пользователь* в *приложении Kitactive* выбирает пункт меню «Телеметрия/Реестр оборудования»;
2) *приложение Kitactive* отрабатывает данное действие *пользователя* и отправляет запрос в *компонент предоставления данных* по пользовательским запросам *системы Kitdata*;
3) *компонент предоставления данных* по пользовательским запросам обрабатывает запрос, запрашивает данные у *базы данных*, формирует JSON объект и отправляет его обратно в *приложение Kitactive*;
4) *приложение Kitactive* принимает JSON объект, парсит его и представляет *пользователю* список реестра оборудования, подключенного к телеметрии.

Вариант использования «Просмотр актуальной информации сенсоров конкретного оборудования, подключенного к телеметрии». Данный вариант использования позволяет просмотреть актуальные данные с сенсоров устройств, подключенных к оборудованию. Диаграмма последовательностей варианта использования «Просмотр актуальной информации сенсоров конкретного оборудования, подключенного к телеметрии» представлена на рисунок 11.

Рисунок 11 – Диаграмма последовательностей варианта использования «Просмотр актуальной информации сенсоров конкретного оборудования, подключенного к телеметрии»

Основной поток событий состоит из четырех шагов:

1) *пользователь* в *приложении Kitactive* из списка реестра оборудования выбирает необходимое оборудование;

2) *приложение Kitactive* отрабатывает данное действие *пользователя* и отправляет запрос в *компонент предоставления данных* по пользовательским запросам *системы Kitdata*;
3) *компонент предоставления данных* по пользовательским запросам обрабатывает запрос, запрашивает данные у *базы данных*, формирует JSON объект и отправляет его обратно в *приложение Kitactive*;
4) *приложение Kitactive* принимает JSON объект, парсит его и представляет пользователю карточку оборудования с информации об самом оборудовании, а также актуальной информации по датчикам, подключенным к выбранному оборудованию.

Вариант использования «Просмотр учета наработки оборудования, подключенного к телеметрии». Данный вариант использования позволяет просмотреть отчет по наработке оборудования в двух вариантах:

- наработка оборудования на дату – данный отчет позволяет просмотреть наработку оборудования на конкретную дату с начала момента подсчета наработки;
- наработка оборудования за период – данный отчет позволяет просмотреть наработку оборудования за конкретный период.

Данные отчеты позволяют также сделать фильтрацию оборудования по подразделению, а также выгружать отчеты в PDF и Excel форматах, для большего удобства просмотра.

Также данный вариант использования позволяет просматривать учет наработки в виде графиков в трех вариантах:

- график фактической работы – данный график показывает два состояния работы оборудования включено и выключено;

- график накопления наработки – данный график показывает работу оборудования с необходимой детализацией (час, день, месяц);
- график в виде таймлайна – данный график позволяет просматривать три состояния оборудования выключено, в ожидании, включено в виде таймлайна, с возможностью просмотра конкретных значений показателя тока на оборудовании.

Данные графики также позволяют сделать фильтрацию по периоду, а также выгружать снимки графиков в форматах SVG и PNG, для большего удобства.

Диаграмма последовательностей варианта использования «Просмотр учета наработки оборудования, подключенного к телеметрии» представлена на рисунок 12.

Рисунок 12 – Диаграмма последовательностей варианта использования «Просмотр учета наработки оборудования, подключенного к телеметрии»

Основной поток событий состоит из четырех шагов:

1) *пользователь* в *приложении Kitactive* выбирает необходимый ему вариант предоставления информации по наработке;
2) *приложение Kitactive* отрабатывает данное действие *пользователя* и отправляет запрос в *компонент предоставления данных* по пользовательским запросам *системы Kitdata*;
3) *компонент предоставления данных* по пользовательским запросам обрабатывает запрос, запрашивает данные у *базы данных*, формирует JSON объект и отправляет его обратно в *приложение Kitactive*;
4) *приложение Kitactive* принимает JSON объект, парсит его и представляет пользователю информацию по наработке в выбранном варианте предоставления.

Вариант использования «Просмотр учета энергопотребления оборудования, подключенного к телеметрии». Данный вариант использования позволяет просмотреть отчет по учету энергопотребления оборудования – «Энергопотребление оборудования за период».

Данный отчет позволяет также сделать фильтрацию оборудования по подразделению, а также выгружать отчеты в PDF и Excel форматах, для большего удобства просмотра.

Также данный вариант использования позволяет просматривать учет наработки в виде графика – «График энергопотребления».

Данный график также позволяют сделать фильтрацию по периоду, а также выгружать снимки графиков в форматах SVG и PNG, для большего удобства.

Диаграмма последовательностей варианта использования «Просмотр учета энергопотребления оборудования, подключенного к телеметрии» представлена на рисунок 13.

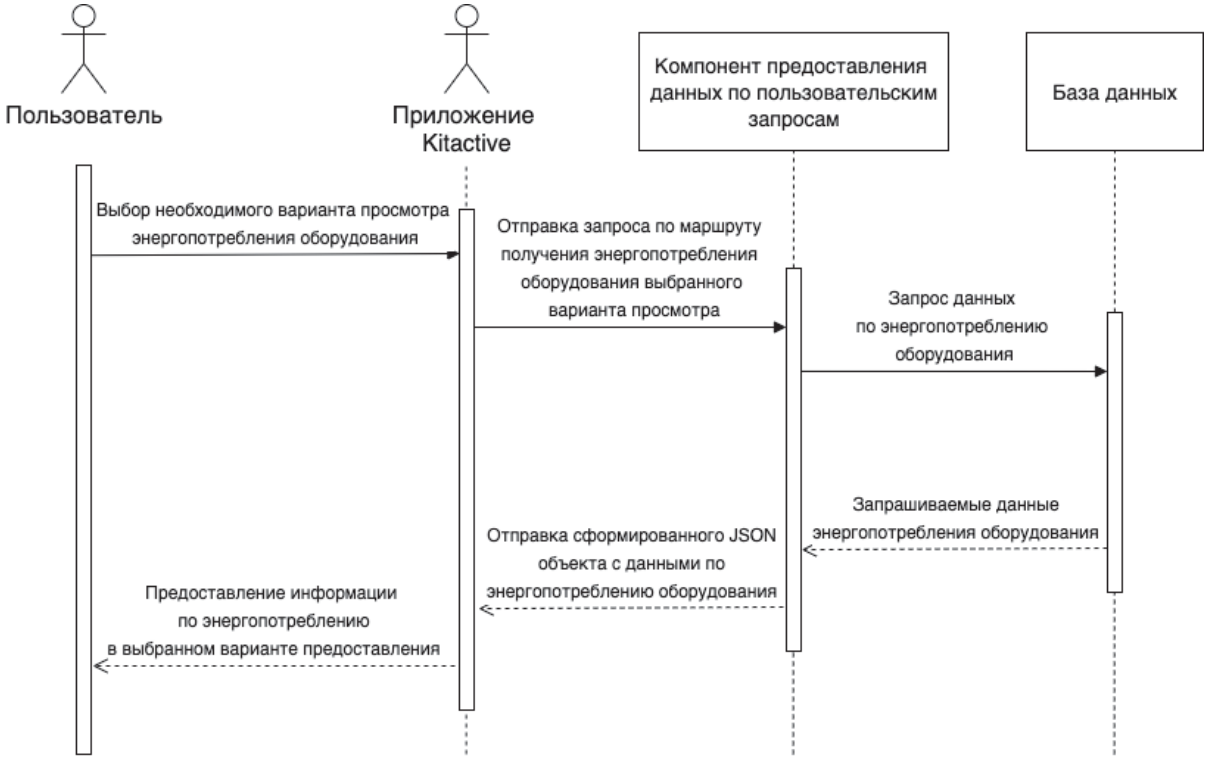

Рисунок 13 – Диаграмма последовательностей варианта использования «Просмотр учета энергопотребления оборудования, подключенного к телеметрии»

Основной поток событий состоит из четырех шагов:

1) *пользователь* в *приложении Kitactive* выбирает необходимый ему вариант предоставления информации по энергопотреблению;
2) *приложение Kitactive* отрабатывает данное действие *пользователя* и отправляет запрос в *компонент предоставления данных* по пользовательским запросам *системы Kitdata*;
3) *компонент предоставления данных* по пользовательским запросам обрабатывает запрос, запрашивает данные у *базы данных*, формирует JSON объект и отправляет его обратно в *приложение Kitactive*;

4) *приложение Kitactive* принимает JSON объект, парсит его и представляет пользователю информацию по энергопотреблению в выбранном варианте предоставления.

Вариант использования «Просмотр количества работающего оборудования, подключенного к телеметрии, в момент запроса». Данный вариант использования позволяет просмотреть количество работающего оборудования, подключенного к телеметрии, в момент запроса.

Диаграмма последовательностей варианта использования «Просмотр количества работающего оборудования, подключенного к телеметрии, в момент запроса» представлена на рисунок 14.

Рисунок 14 – Диаграмма последовательностей варианта использования «Просмотр количества работающего оборудования, подключенного к телеметрии, в момент запроса»

Основной поток событий состоит из четырех шагов:

1) *пользователь* в *приложении Kitactive* выбирает пункт меню «Телеметрия/Монитор», в окне данного пункта есть несколько карточек показателей, одним из которых является показатель количества работающего оборудования;
2) *приложение Kitactive* отрабатывает данное действие *пользователя* и отправляет запрос в *компонент предоставления данных* по пользовательским запросам *системы Kitdata*;
3) *компонент предоставления данных* по пользовательским запросам обрабатывает запрос, запрашивает данные у *базы данных*, формирует JSON объект и отправляет его обратно в *приложение Kitactive*;
4) *приложение Kitactive* принимает JSON объект, парсит его и представляет пользователю количество работающего оборудования из всего оборудования, подключенного к системе телеметрии.

Вариант использования «Создание регламентной задачи по наработке оборудования». Данный вариант использования позволяет создать регламентную задачу по наработке оборудования и отслеживать ее достижения.

Диаграмма последовательностей варианта использования «Создание регламентной задачи по наработке оборудования» представлена на рисунок 15.

Рисунок 15 – Диаграмма последовательностей варианта использования «Создание регламентной задачи по наработке оборудования»

Основной поток событий состоит из четырех шагов:

1) *пользователь* в *приложении Kitactive* создает регламентную задачу по наработке оборудования;
2) *приложение Kitactive* отрабатывает данное действие *пользователя* и отправляет запрос в *компонент предоставления данных* по пользовательским запросам *системы Kitdata*;
3) *компонент предоставления данных* по пользовательским запросам обрабатывает запрос, вставляет данные в *базу данных*, формирует ответ с сообщением об успешном создании и отправляет его обратно в *приложение Kitactive*;
4) *приложение Kitactive* принимает ответ сообщением об успешном создании и предоставляет его *пользователю*.

3.3. ОПИСАНИЕ ДАННЫХ

Для того, чтобы описать данные системы необходимо составить схему базы данных. Схема базы данных – это ее структура, описанная на формальном языке, поддерживаемом на формальном языке системы управления базы данных.

Структурная схема базы данных представлена на рисунок 16.

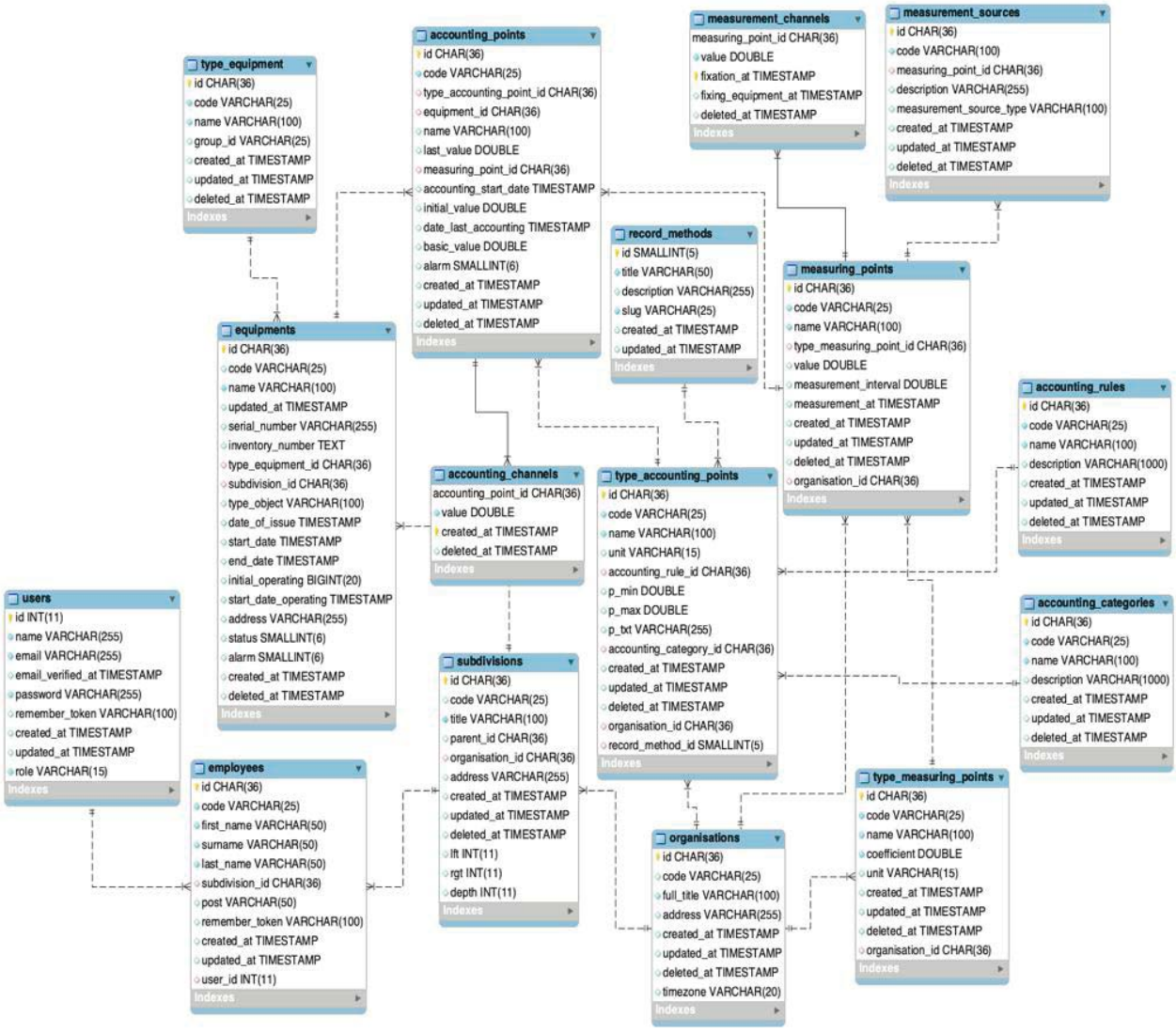

Рисунок 16 – Структурная схема базы данных

Список таблиц базы данных представлен в таблице Таблица 2.

Таблица 2 – Список таблиц базы данных

Наименование	**Назначение**
type_equipment	Виды оборудования
equipments	Оборудование
accounting_channels	Каналы учета
accounting_points	Точки учета
accounting_categories	Категории учета
type_accounting_points	Типы точек учета
record_methods	Методы записи
subdivisions	Подразделения
accounting_rules	Правила учета
measuring_points	Точки измерения
measurement_channels	Каналы измерения
measurement_sources	Источники измерения
type_measuring_points	Типы источников измерения
organisations	Организации
users	Пользователи
employees	Сотрудники

Из всех таблиц на фоне других выделяются несколько, хранящих в себе наибольшее количество данных. Таковыми таблицами являются:

- *equipments*, хранящая в себе описание всего оборудования, находящегося в системе *KitData*;
- *measurement_sources*, хранящая в себе название всех каналов входящих измерений;

- *measuring_points* и *measurement_channels*, хранящие в себе данные, прошедшие первичную обработку;
- *accounting_points* и *accounting_channels*, хранящие в себе данные, прошедшие вторичную обработку.

Описание полей таблиц, представленных на рисунок 16 приведены в приложении А.

3.4. ВЫВОД

В данной главе были разработаны:

- архитектура предлагаемого решения, для этого была разработана диаграмма компонентов;
- алгоритмы решения задачи, для этого были описаны: алгоритмы работы приложений первичной и вторичной работы, а также варианты взаимодействия пользователей с системой;
- описание данных, для этого была составлена схема базы данных.

4. РЕАЛИЗАЦИЯ

В данной работе необходимо разработать систему учета и анализа телеметрии медицинского оборудования. Для ее функционирования необходимо написать: приложения первичной и вторичной обработки, веб-приложение предоставления API и добавить в приложение Kitactive интерфейс для взаимодействия пользователя с системой Kitdata.

Описание реализации компонентов системы, а также сторонних библиотек, использованных при их разработке, будет представлено ниже.

4.1. РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЙ ПЕРВИЧНОЙ И ВТОРИЧНОЙ ОБРАБОТКИ

Для разработки приложений первичной и вторичной обработки были задействованы следующие библиотеки:

- *mqtt*, версии 3.0.0. Для прослушивания и публикации сообщений;
- *redis*, версии 2.8.0. Для передачи сообщений о возникновении: критических ситуаций и наработки оборудования по регламентной задачи;
- *mysql*, версии 2.17.1. Для взаимодействия с базой данных.

Диаграмма модулей приложений первичной и вторичной обработки с указанием основных объектов и функций приведена на рисунок 17.

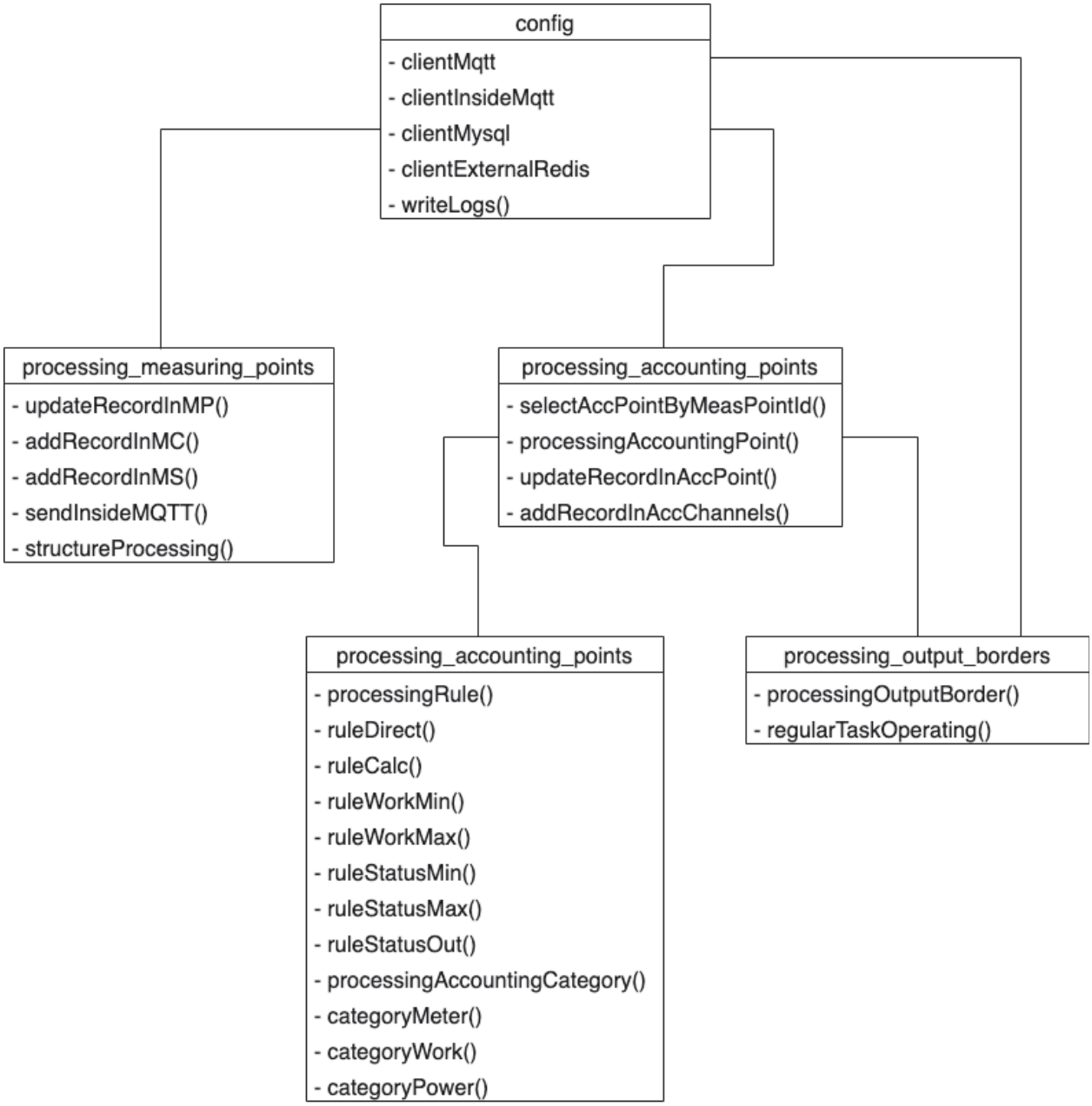

Рисунок 17 – Диаграмма модулей приложений первичной и вторичной обработки

Модуль *config* предназначен для хранения конфигурационных параметров программ обработки данных. К нему обращаются модули «processing_measuring_points», «processing_accounting_points» и «processing_output_borders».

Модуль *processing_measuring_points* предназначен для первичной обработки данных. Он обеспечивает обновление записи точки измерения, добавление записей: канала измерения и источника измерения, а также обработки составного пакета измерений и отправка пакетов для вторичной обработки.

Модуль *processing_accounting_points* предназначен для вторичной обработки данных. Он обеспечивает выборку точек учета по точке измерения, обработки точки учета, обновление точки учета и добавление записи в канал учета.

Модуль *processing_rules* предназначен для определения и обработки правила измерения и категории.

Модуль *processing_output_borders* предназначен для отправки в приложение Kitactive уведомлений о возникновении: критических ситуаций и наступления наработки оборудования по регламентной задачи.

Код основных методов данных модулей приведен в приложении Б.

4.2. РЕАЛИЗАЦИЯ ВЕБ-ПРИЛОЖЕНИЯ ПРЕДОСТАВЛЕНИЯ API

Для разработки веб-приложения предоставления API были задействованы следующие библиотеки:

- *maatwebsite/excel*, версии 3.1. Для экспорта отчетов в Excel;
- *niklasravnsborg/laravel-pdf*, версии 3.1. Для экспорта отчетов в PDF;
- *predis/predis*, версии 1.1. Для хранения кэша в Redis сервере;
- baum/baum, версии 2. Для хранения подразделений организаций в виде упорядоченного дерева;
- *laravel/passport*, версии 7.3. Для аутентификации API.

Веб-приложение предоставления API имеет большое количество созданных директорий и файлов. Ниже будут описаны основные классы приложения, реализующие бизнес-логику.

Для управления основными сущностями (создания, редактирования, удаления), а также предоставления информации о них были созданы следующие классы:

- AccountingCategoriesService – отвечает за *категории учета*;
- AccountingChannelsService – отвечает за *каналы учета*;
- AccountingPointsService – отвечает за *точки учета*;
- AccountingRulesService – отвечает за *правила учета*;
- EquipmentsService – отвечает за *оборудование*;
- MeasurementSourcesService – отвечает за *источники измерений*;
- MeasuringPointsService – отвечает за *точки измерений*;
- SchedulesService – отвечает за *графики работы*;
- SubdivisionsService – отвечает за *подразделения*;
- TypeAccountingPointsService – отвечает за *типы точек учета*;
- TypeMeasuringPointsService – отвечает за *типы точек измерения*.

Для вычисления и предоставления данных для графиков, отчетов и монитора оперативной информации были созданы следующие классы:

- ChartAccumulationOperation – отвечает за *график накопления наработки*;
- ChartActualWork – отвечает за *график фактической работы*;
- ChartPowerUsage – отвечает за *график энергопотребления*;
- ActualOperatingReport – отвечает за *отчет наработка оборудования за период*;
- ActualWorkReport – отвечает за *отчет наработка оборудования на дату*;
- PowerUsageReport – отвечает за *отчет энергопотребление оборудования за период*;

- DashboardsService – отвечает за *показатели и графики монитора оперативной информации*.

Код основных методов данных классов приведен в приложении В.

4.3. РЕАЛИЗАЦИЯ ВЕБ-ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ

Для разработки веб-приложения предоставления API были задействованы следующие библиотеки:

- *redux*, версии 3.6.0. Для хранения глобального состояния приложения;
- *moment*, версии 2.24.0. Для работы с датами;
- *apexcharts*, версии 3.8.3. Для рендеринга графиков.

Для взаимодействия пользователя с системой Kitdata в приложении Kitactive был добавлен новый пункт меню телеметрия, содержащий функционал просмотра: монитора оперативной информации, реестра оборудования, графиков и отчетов. Для его реализации были разработаны следующие классы:

- ChartAccumulationOperationContainer – предназначен для рендеринга графика накопления наработки;
- ChartActualWorkContainer – предназначен для рендеринга графика фактической работы;
- ChartPowerUsageContainer – предназначен для рендеринга графика энергопотребления;
- MonitorContainer – предназначен для рендеринга монитора телеметрии;
- ActualOperationContainer – предназначен для рендеринга отчета накопления наработки;

- ActualWorkContainer – предназначен для рендеринга отчета фактической работы;
- PowerUsageContainer – предназначен для рендеринга отчета энергопотребления;
- RegistryEquipmentsContainer – предназначен для рендеринга списка реестра оборудования;
- RegistryEquipmentContainer – предназначен для рендеринга описания оборудования в карточке оборудования, подключенного к телеметрии;
- AccountingPointsContainer – предназначен для рендеринга списка точек учета в карточке оборудования, подключенного к телеметрии;
- AccountingChannelsContainer – предназначен для рендеринга списка истории изменения показателей с датчиков в карточке оборудования, подключенного к телеметрии.

Код основных методов данных классов приведен в приложении Г.

4.4. ВЫВОД

В данной главе была описана реализация следующих компонентов разрабатываемой системы:

- приложений первичной и вторичной обработки, для этого были приведены задействованные библиотеки и разработанные модули;
- веб-приложения предоставления API, для этого были приведены задействованные библиотеки и основные разработанные классы;
- веб-интерфейс пользователя, для этого были приведены задействованные библиотеки и основные разработанные классы.

5. РЕЗУЛЬТАТЫ И ТЕСТИРОВАНИЕ

В данной главе будет приведено функциональное тестирование вариантов использования пользователя с системой, описанных ранее.

Под функциональным тестированием понимается тестирование программного обеспечения в целях проверки реализуемости функциональных требований, определяющих способность его в определенных условиях решать задачи, нужные пользователям.

5.1. ФУНКЦИОНАЛЬНОЕ ТЕСТИРОВАНИЕ

В приложении Kitactive для каждой роли организации можно настраивать доступные разделы. Для разделов, созданных для взаимодействия с разработанной системой, была также добавлена данная возможность, представленная на рисунок 18.

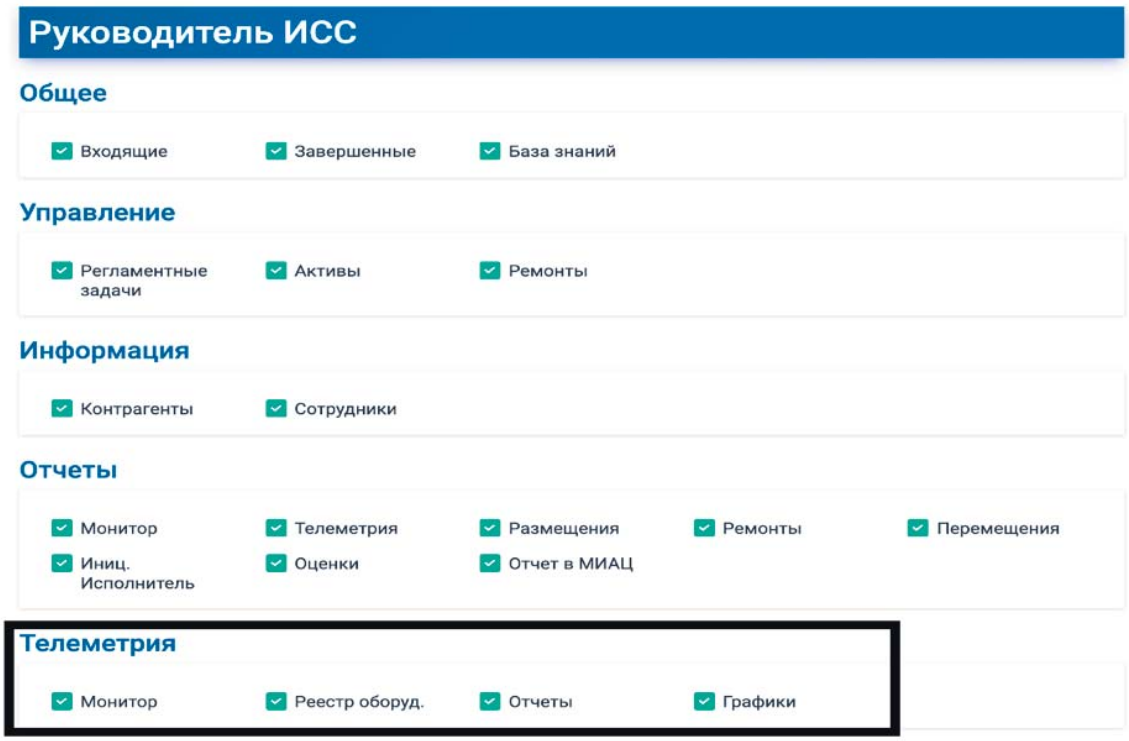

Рисунок 18 – Настройка доступа к разделам Телеметрии

Ниже приведено описание тестов с указанием: ожидаемого результата, шагов тестирования и статуса прохождения.

Тест варианта использования «*Просмотр количества работающего оборудования, подключенного к телеметрии, в момент запроса*». После его прохождения ожидаемым результатом является представление показателя количества работающего оборудования.

Шаги тестирования:

1) авторизация пользователя в приложении *Kitactive* (рисунок 19);

Рисунок 19 – Окно авторизации пользователя

2) переход в пункт меню *«Телеметрия/Монитор»* (рисунок 20);

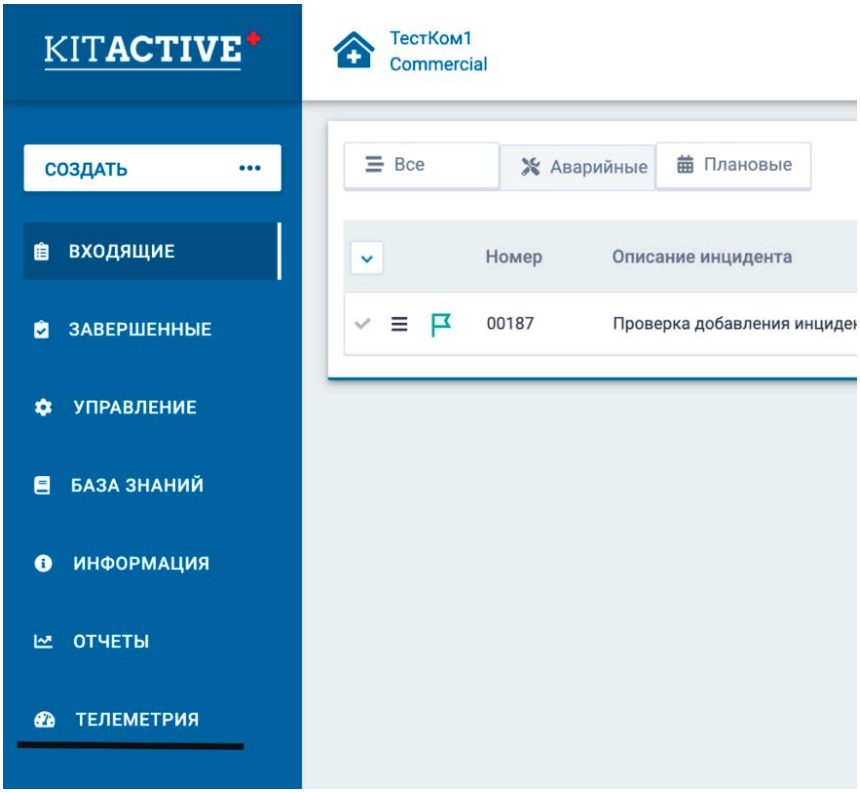

Рисунок 20 – Переход в пункт меню «Телеметрия/Монитор»

3) просмотр показателя количества работающего оборудования (рисунок 21).

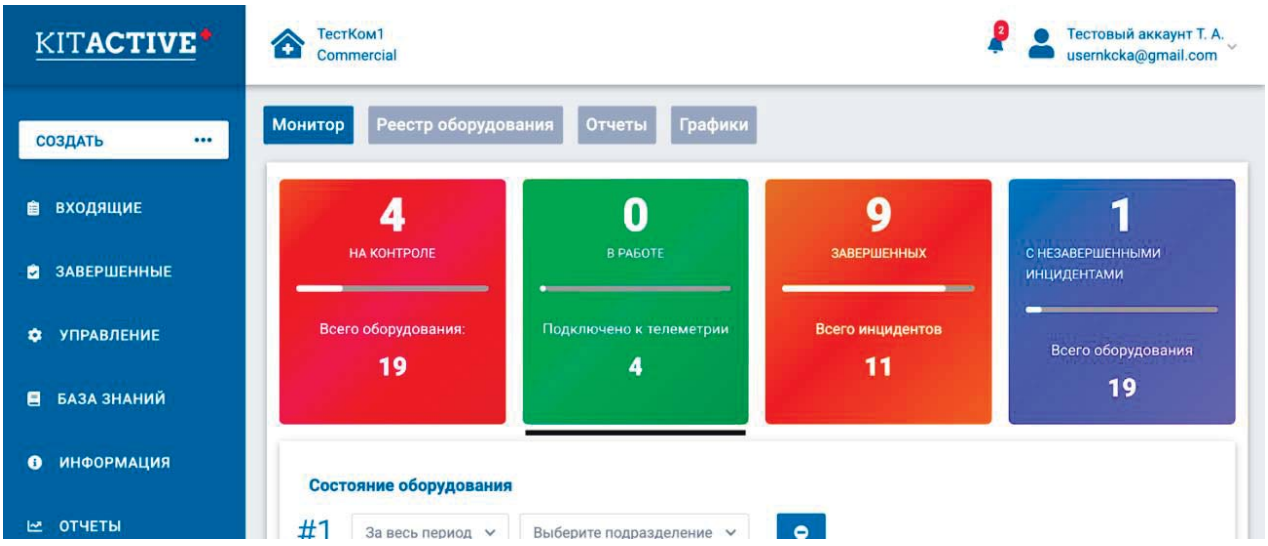

Рисунок 21 – Просмотр показателя количества работающего оборудования

Тест варианта использования «*Просмотр количества работающего оборудования, подключенного к телеметрии, в момент запроса*» пройден.

Тест варианта использования «*Просмотр реестра оборудования, подключенного к телеметрии*». После его прохождения ожидаемым результатом является представление пользователю в виде списка реестр оборудования, подключенного к телеметрии.

Шаги тестирования:

1) переход в пункт меню *«Телеметрия/Реестр оборудования»* (рисунок 22);

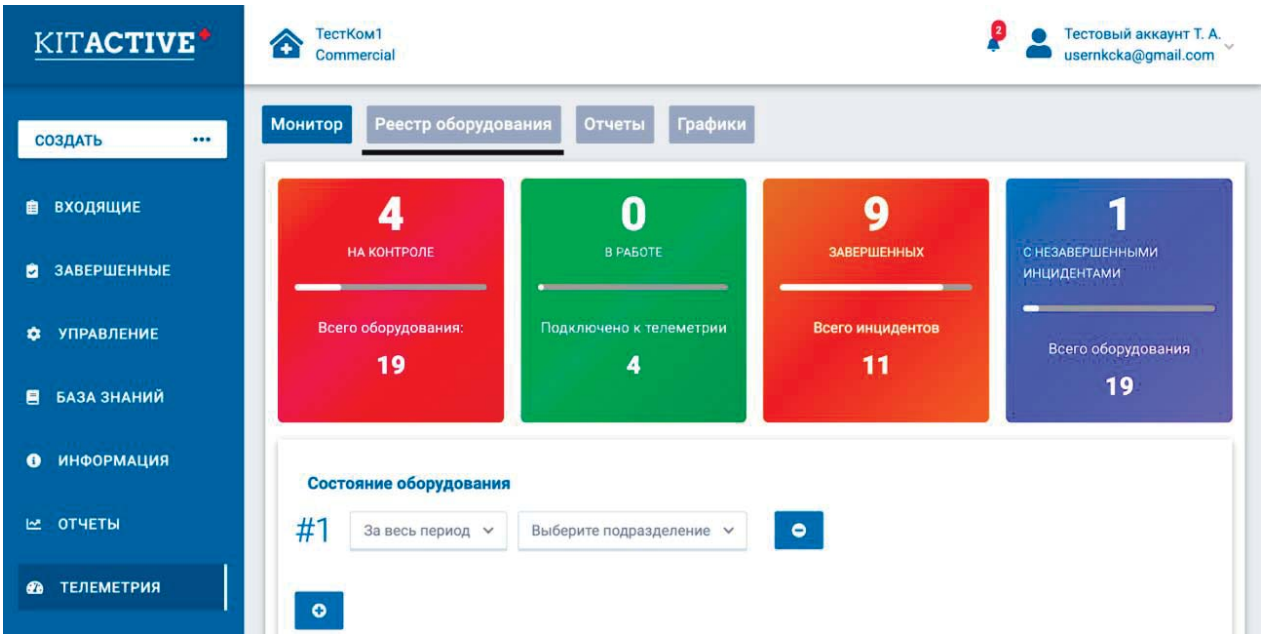

Рисунок 22 – Переход в пункт меню «Телеметрия/Реестр оборудования»

2) просмотр списка оборудования, подключенного к телеметрии (рисунок 23).

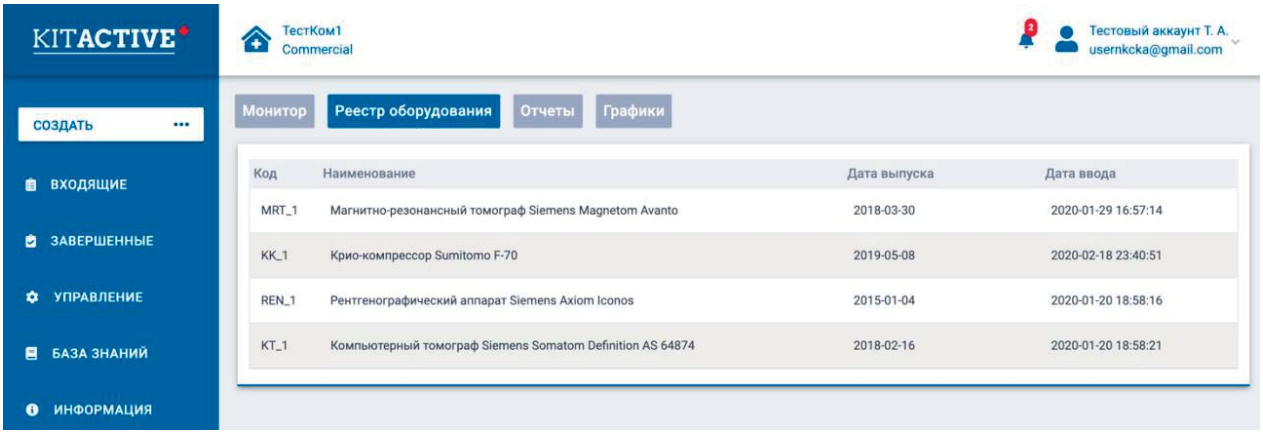

Рисунок 23 – Просмотр списка оборудования, подключенного к телеметрии

Тест варианта использования «*Просмотр реестра оборудования, подключенного к телеметрии*» пройден.

Тест варианта использования «*Просмотр актуальной информации сенсоров конкретного оборудования, подключенного к телеметрии*». После его прохождения ожидаемым результатом является представление актуальной информации сенсоров оборудования, просмотр истории изменения показателей измерений в табличной форме и в виде графика.

Шаги тестирования:

1) пользователь из списка реестра оборудования выбирает необходимое (рисунок 24);

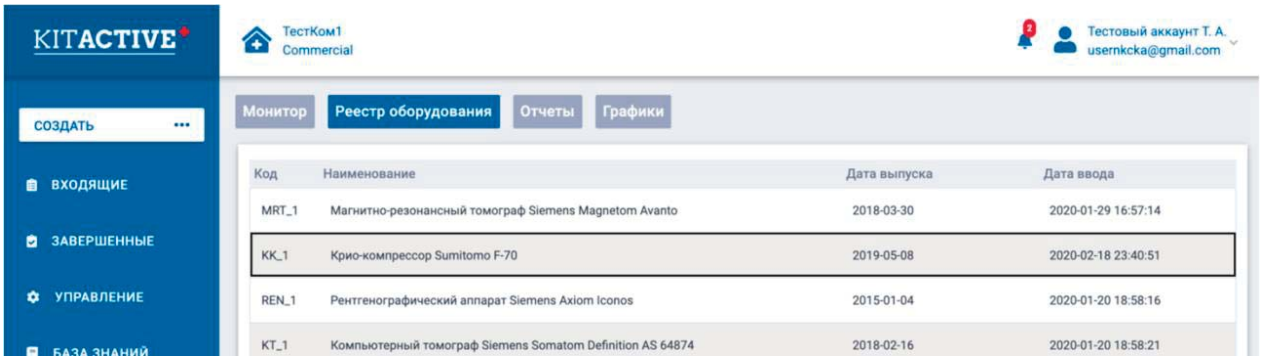

Рисунок 24 – Пользователь из списка реестра оборудования выбирает необходимое

2) переход в раздел «*Датчики*» (рисунок 25);

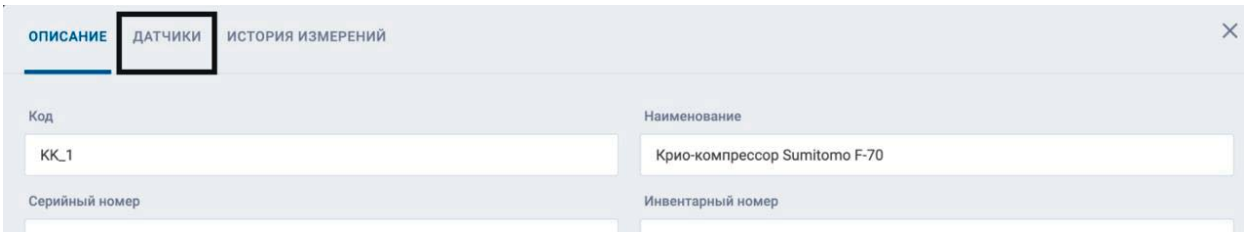

Рисунок 25 – Переход в раздел «Датчики»

3) выбор необходимого *датчика* (рисунок 26);

Рисунок 26 – Выбор необходимого датчика

4) просмотр истории изменения выбранного показателя в *табличной форме* (рисунок 27);

Рисунок 27 – Просмотр истории изменения выбранного показателя в табличной форме

5) выбор просмотра изменения показателя в виде *графика* (рисунок 28);

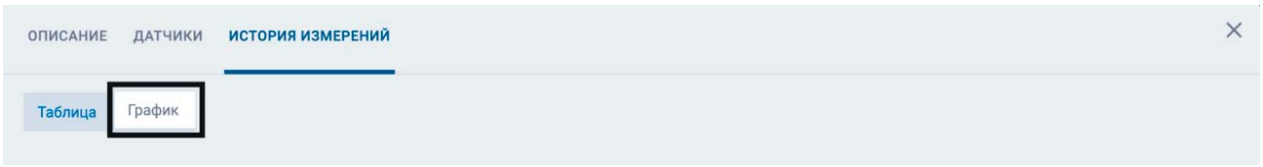

Рисунок 28 – Выбор просмотра изменения показателя в виде графика

6) нажатие кнопки *обновить* (рисунок 29);

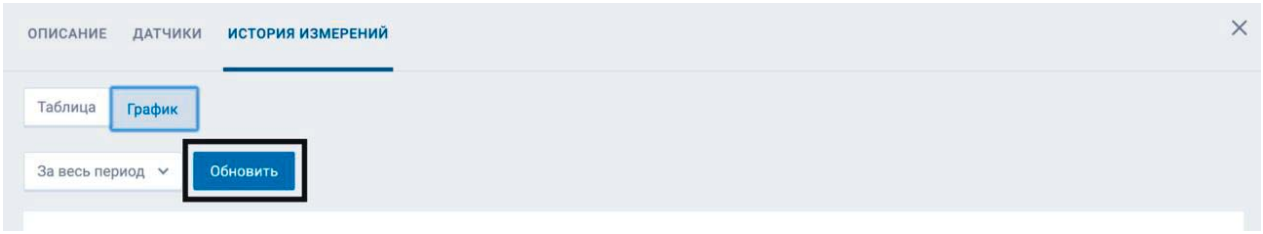

Рисунок 29 – Нажатие кнопки обновить

7) просмотр истории изменения выбранного показателя в виде графика (рисунок 30).

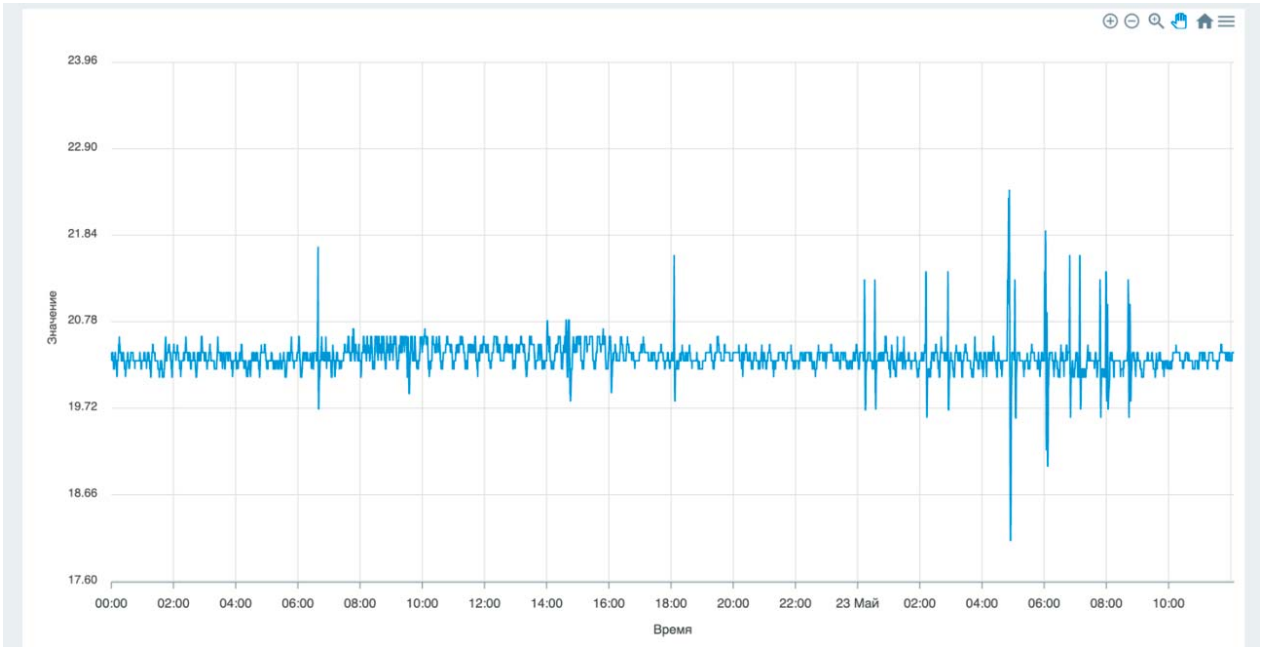

Рисунок 30 – Просмотр истории изменения выбранного показателя в виде графика

Тест варианта использования «*Просмотр актуальной информации сенсоров конкретного оборудования, подключенного к телеметрии»* пройден.

Тест варианта использования «*Просмотр учета наработки оборудования, подключенного к телеметрии»*. После его прохождения ожидаемым результатом является представление в нескольких вариантах учета наработки оборудования.

Шаги тестирования:

1) переход в пункт меню *«Телеметрия/Отчеты»* (рисунок 31);

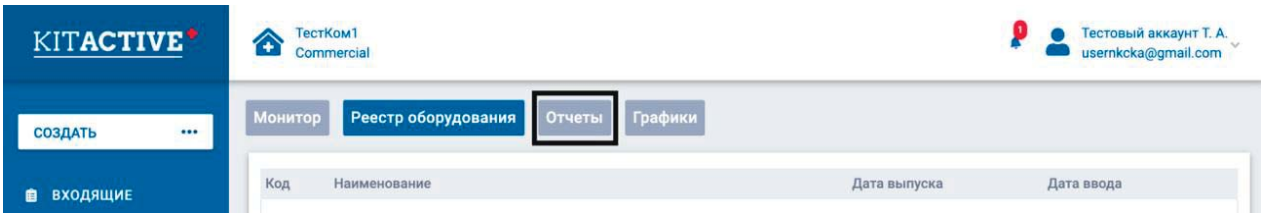

Рисунок 31 – Переход в пункт меню «Телеметрия/Отчеты»

2) выбор необходимого *подразделения* и *даты* (рисунок 32);

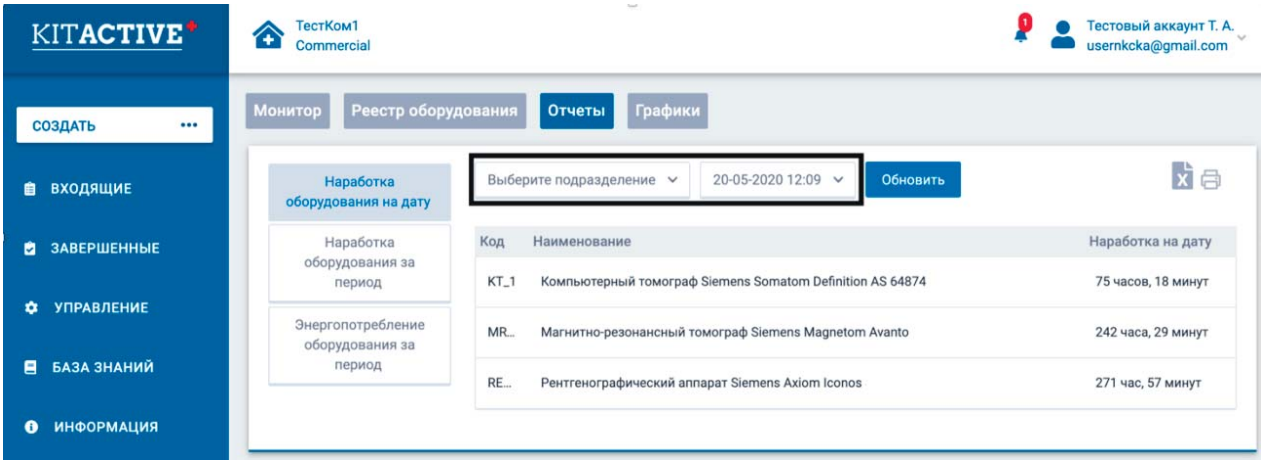

Рисунок 32 – Выбор необходимого подразделения и даты

3) нажатие кнопки *обновить* (рисунок 33);

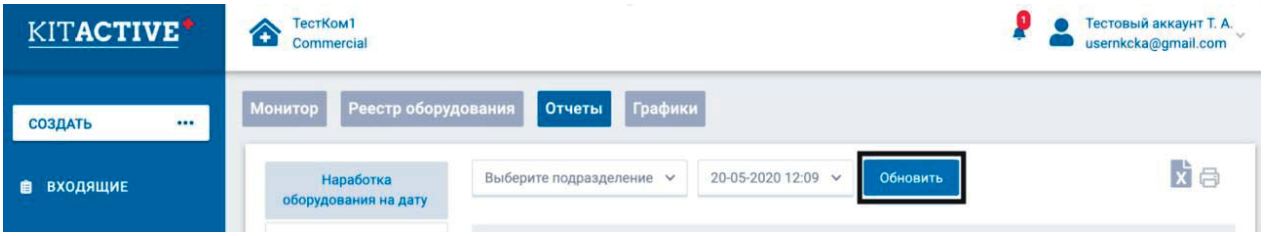

Рисунок 33 – Нажатие кнопки обновить

4) просмотр отчета *наработки оборудования на дату* (рисунок 34);

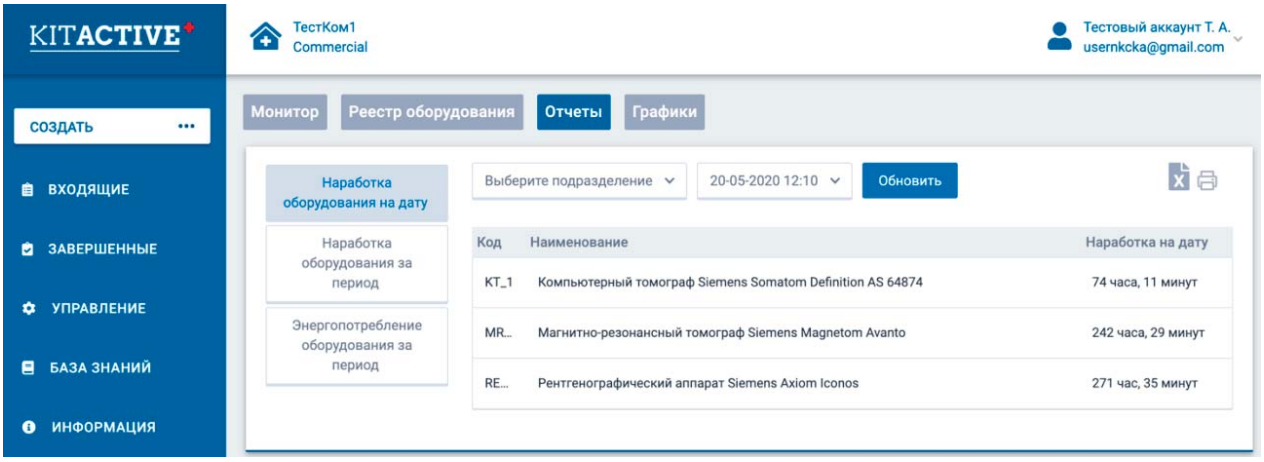

Рисунок 34 – Просмотр отчета наработки оборудования на дату

5) переход в пункт меню *«Телеметрия/Отчеты/Наработка оборудования за период»* (рисунок 35);

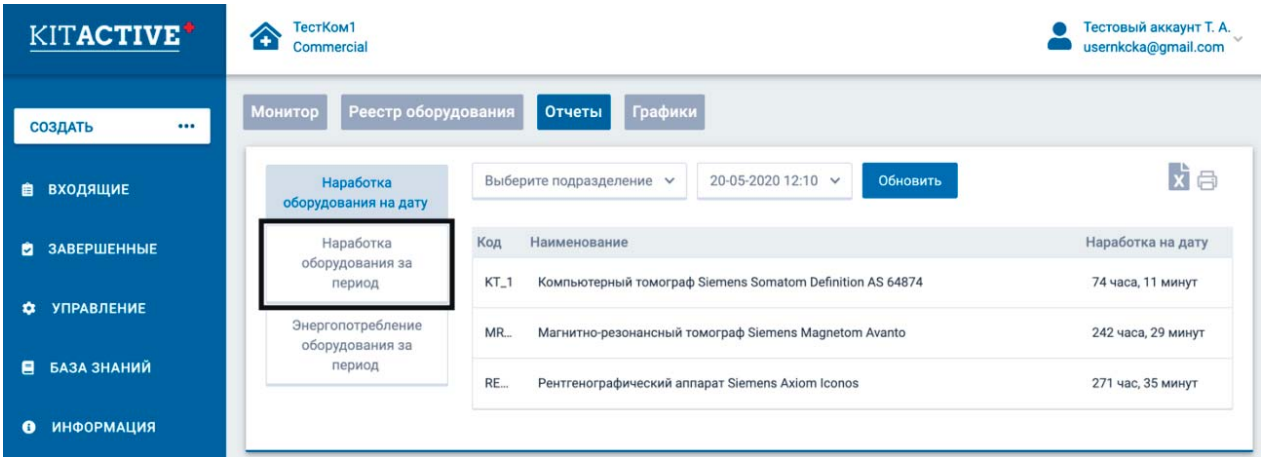

Рисунок 35 – Переход в пункт меню «Телеметрия/Отчеты/Наработка оборудования за период»

6) выбор необходимого *подразделения* и *периода работы* (рисунок 36);

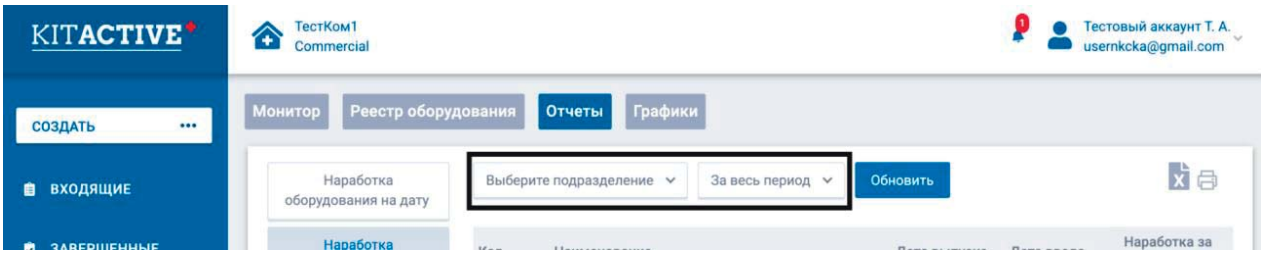

Рисунок 36 – Выбор необходимого подразделения и периода работы

7) нажатие кнопки *обновить* (рисунок 37);

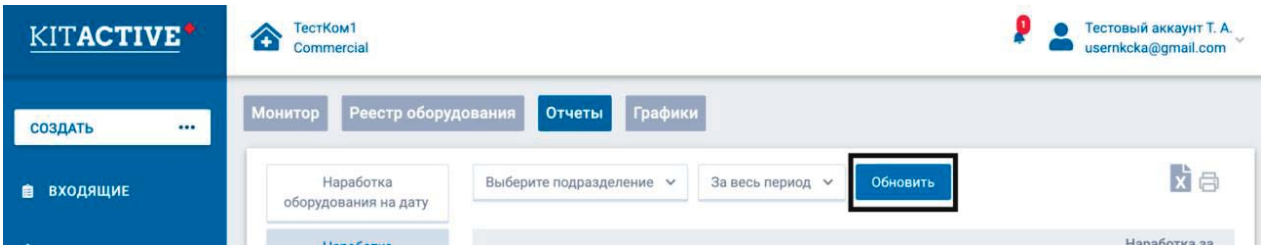

Рисунок 37 – Нажатие кнопки обновить

8) просмотр отчета *наработки оборудования за период* (рисунок 38);

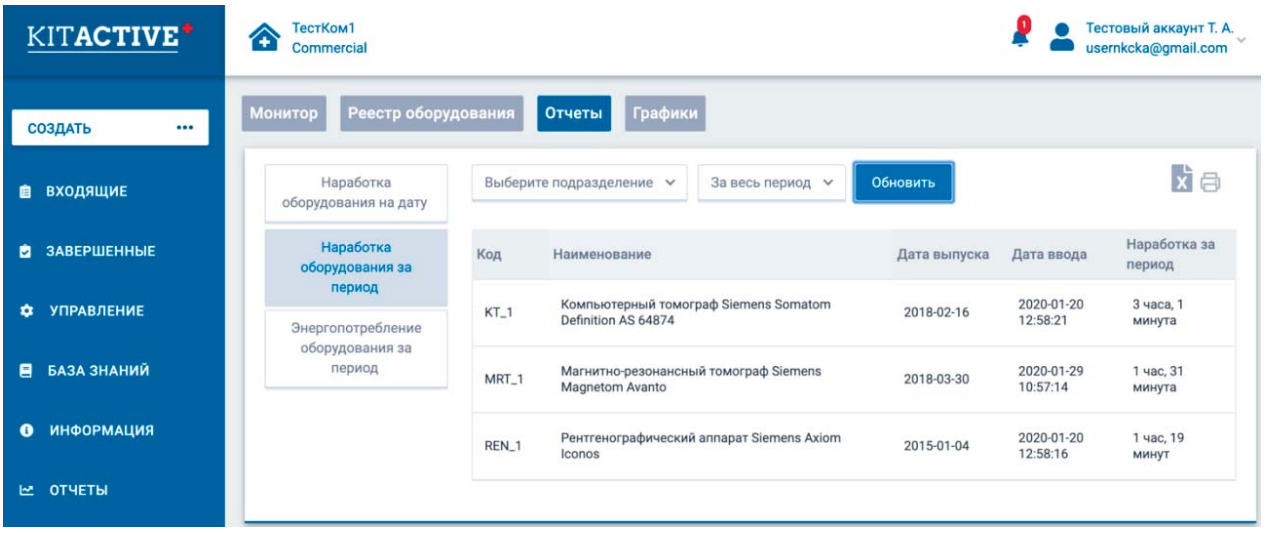

Рисунок 38 – Просмотр отчета наработки оборудования за период

9) переход в пункт меню *«Телеметрия/Графики»* (рисунок 39);

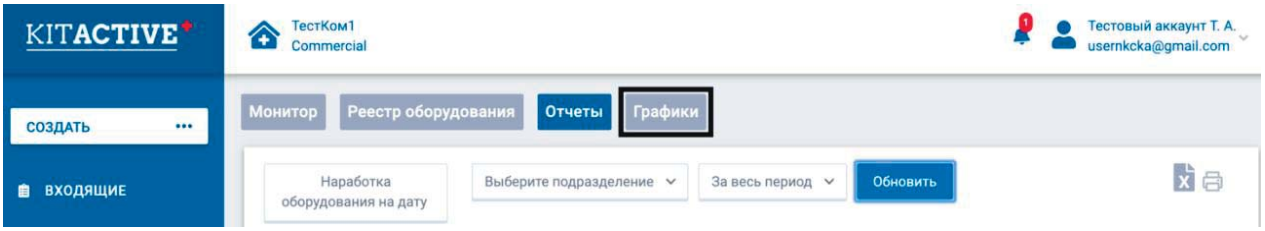

Рисунок 39 – Переход в пункт меню «Телеметрия/Графики»

10) выбор необходимого *периода работы*, *подразделения* и *оборудования* (рисунок 40);

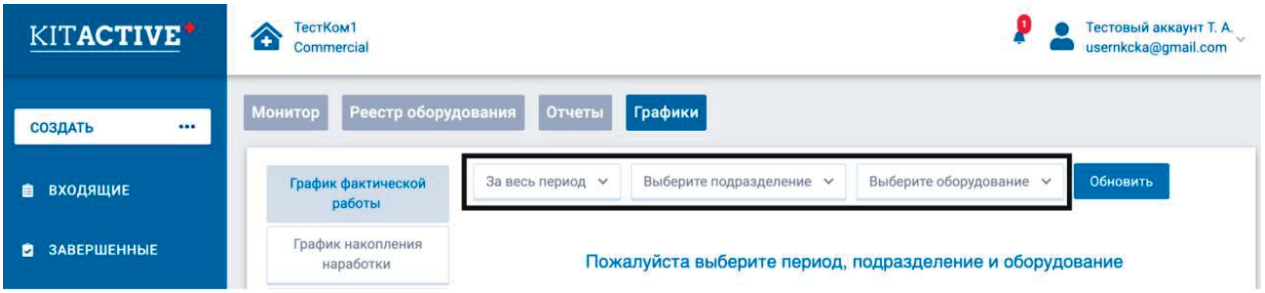

Рисунок 40 – Выбор необходимого периода работы, подразделения и оборудования

11) нажатие кнопки *обновить* (рисунок 41);

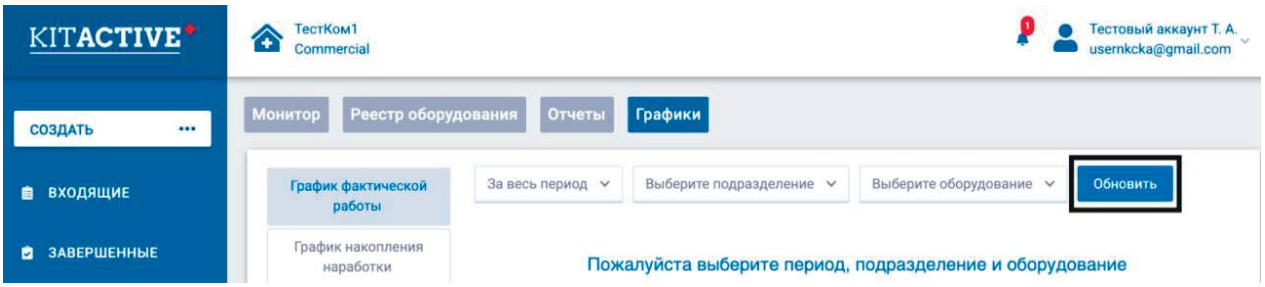

Рисунок 41 – Нажатие кнопки обновить

12) просмотр *графика фактической работы* (рисунок 42);

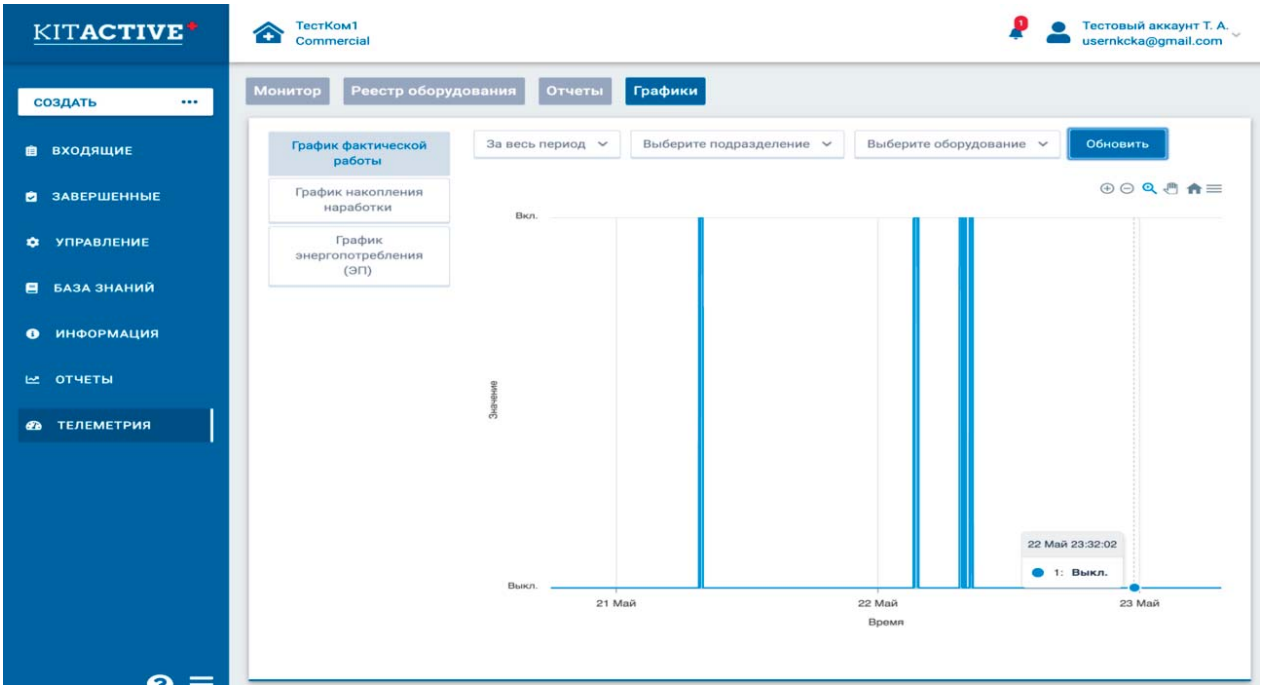

Рисунок 42 – Просмотр графика фактической работы

13) переход в пункт меню *«Телеметрия/Графики/График накопления наработки»* (рисунок 43);

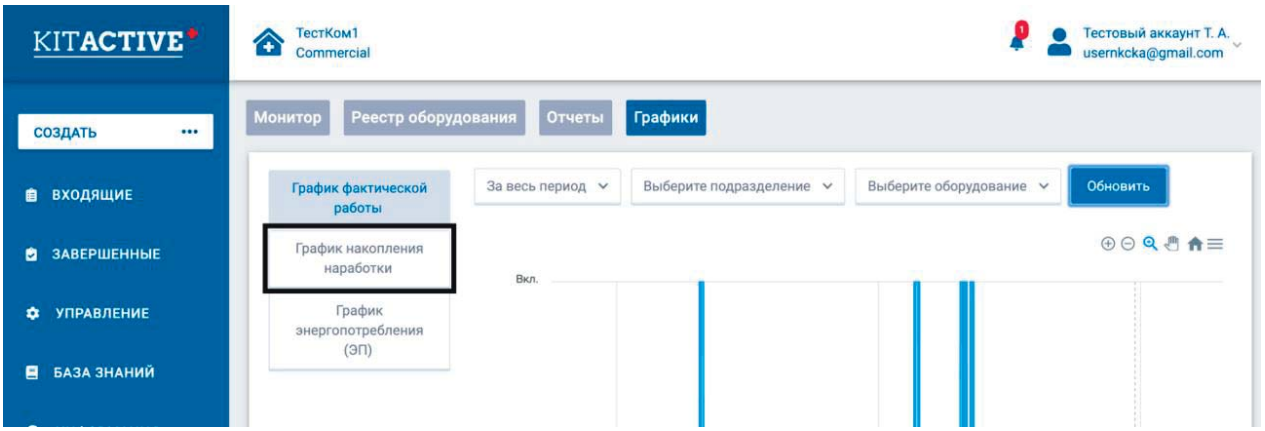

Рисунок 43 – Переход в пункт меню «Телеметрия/Графики/График накопления наработки»

14) выбор необходимой *детализации*, *периода работы*, *подразделения* и *оборудования* (рисунок 44);

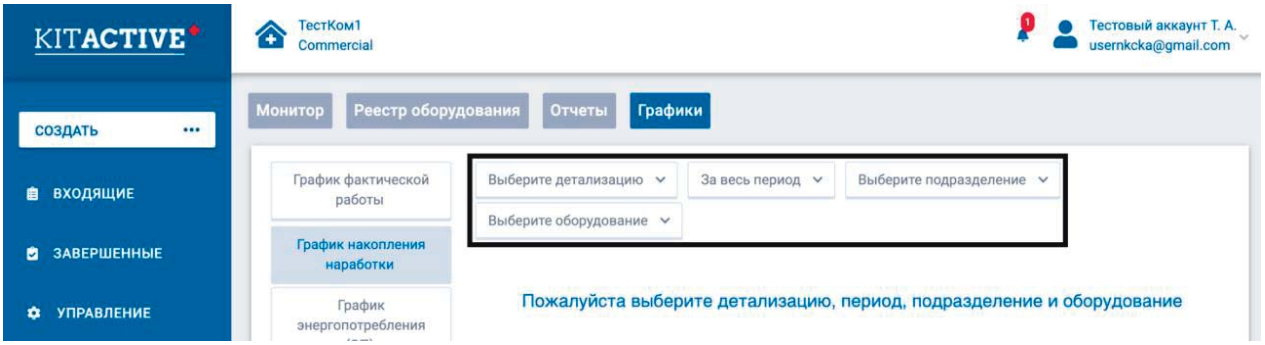

Рисунок 44 – Выбор необходимой детализации, периода работы, подразделения и оборудования

15) нажатие кнопки *обновить* (рисунок 45);

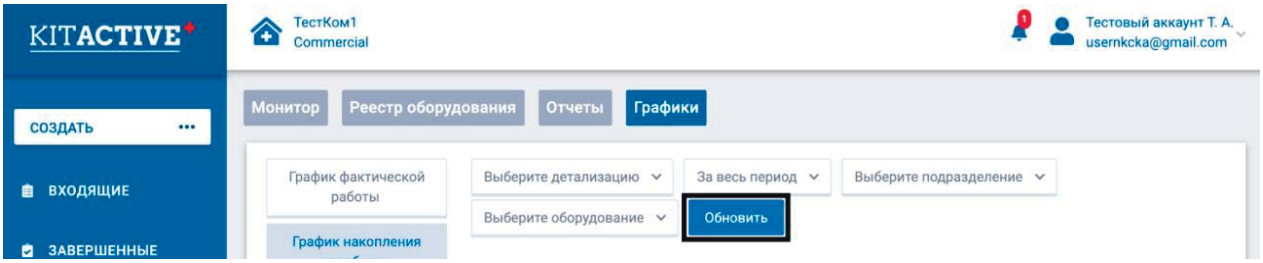

Рисунок 45 – Нажатие кнопки обновить

16) просмотр *графика накопления наработки* (рисунок 46);

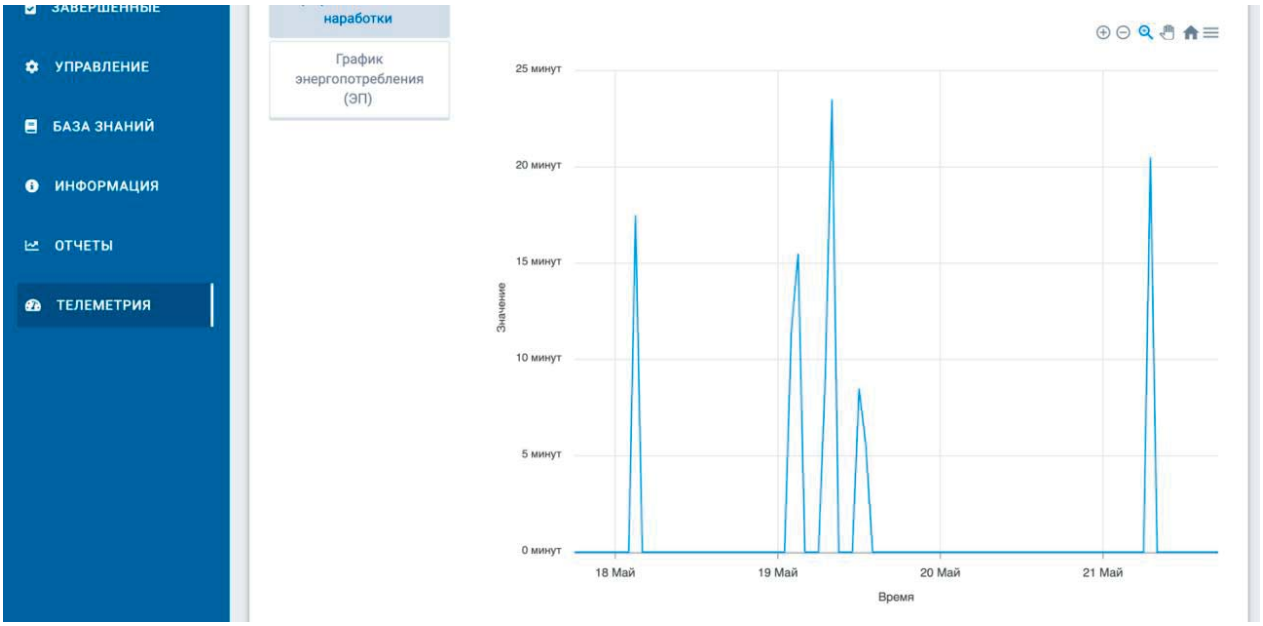

Рисунок 46 – Просмотр графика накопления наработки

17) переход в пункт меню *«Телеметрия/Монитор»* (рисунок 47);

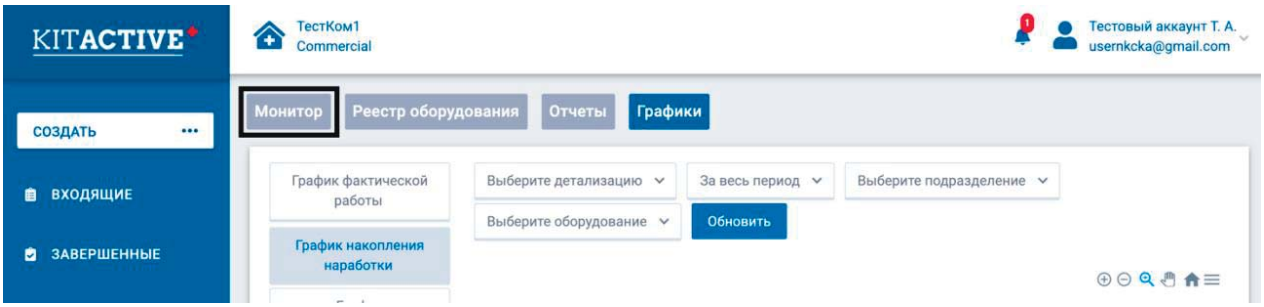

Рисунок 47 – Переход в пункт меню «Телеметрия/Монитор»

18) в области *состояние оборудования* выбор необходимого *периода работы*, *подразделения* и *оборудования* (рисунок 48);

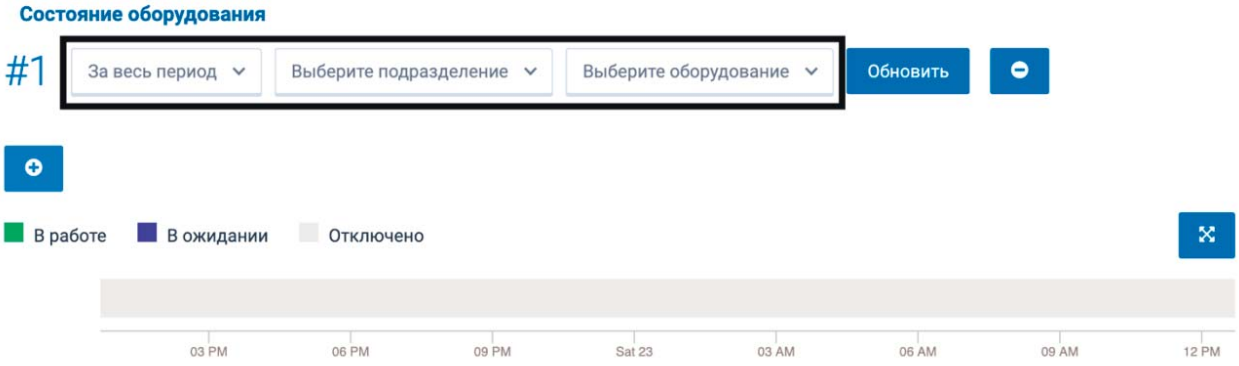

Рисунок 48 – Выбор необходимого периода работы, подразделения и оборудования

19) нажатие кнопки *обновить* (рисунок 49);

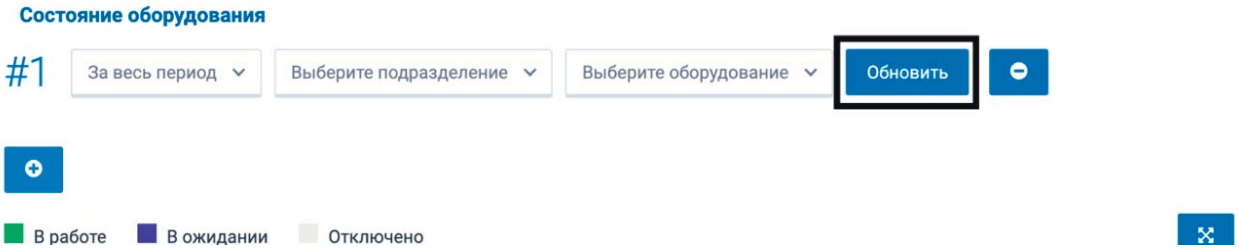

Рисунок 49 – Нажатие кнопки обновить

20) просмотр *состояний работы оборудования* в виде *таймлайна* (рисунок 50);

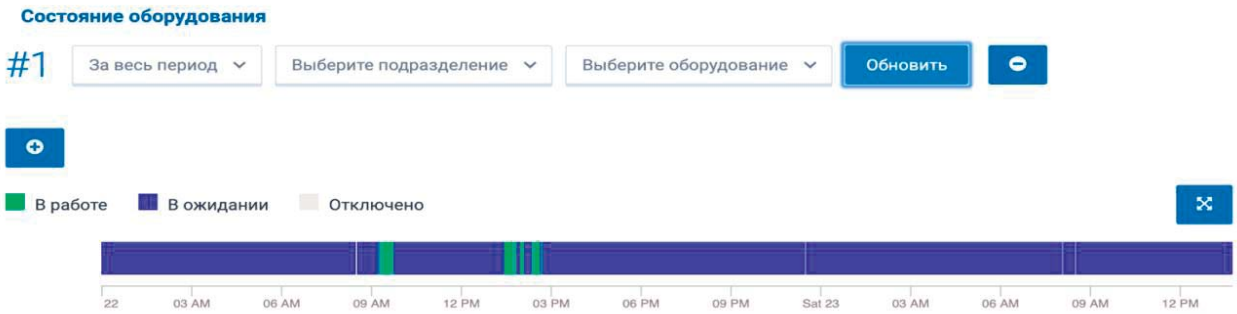

Рисунок 50 – Просмотр состояний работы оборудования в виде таймлайна

21) нажатие кнопки *просмотра подробного графика изменения показателя тока* (рисунок 51);

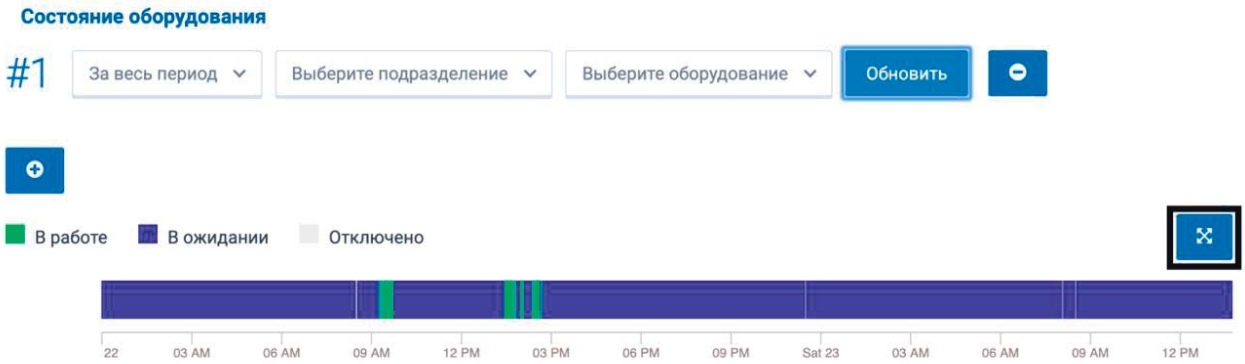

Рисунок 51 – Нажатие кнопки просмотра подробного графика изменения показателя тока

22) просмотр *подробного графика изменения показателя тока* (рисунок 52).

Рисунок 52 – Просмотр подробного графика изменения показателя тока

Тест варианта использования «*Просмотр актуальной информации сенсоров конкретного оборудования, подключенного к телеметрии*» пройден.

Тест варианта использования «*Просмотр учета энергопотребления оборудования, подключенного к телеметрии*». После его прохождения ожидаемым результатом является представление в нескольких вариантах учета энергопотребления оборудования.

Шаги тестирования:

1) переход в пункт меню *«Телеметрия/Отчеты»* (рисунок 53);

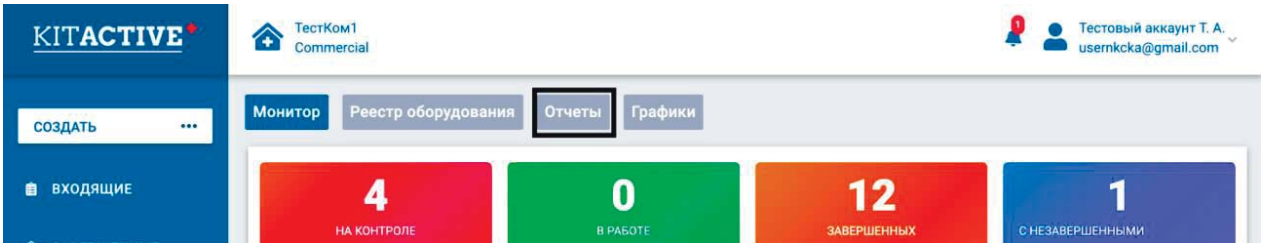

Рисунок 53 – Переход в пункт меню «Телеметрия/Отчеты»

2) переход в пункт меню *«Телеметрия/Отчеты/Энергопотребление оборудования за период»* (рисунок 54)

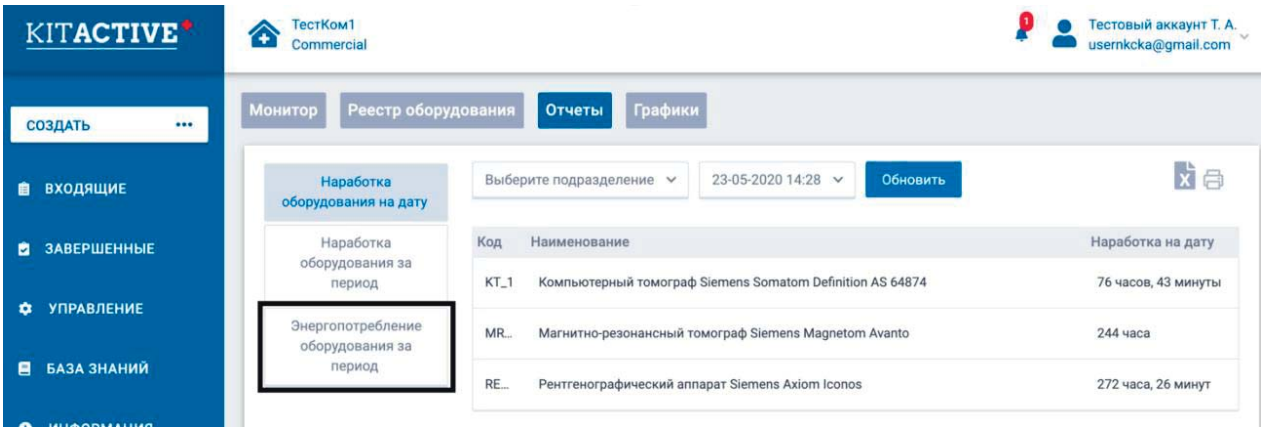

Рисунок 54 – Переход в пункт меню «Телеметрия/Отчеты/Энергопотребление оборудования за период»

3) выбор необходимого *подразделения* и *периода работы* (рисунок 55);

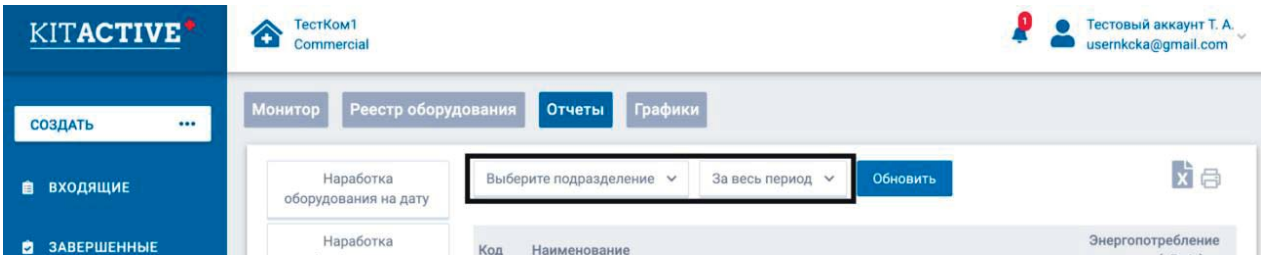

Рисунок 55 – Выбор необходимого подразделения и периода работы

4) нажатие кнопки *обновить* (рисунок 56);

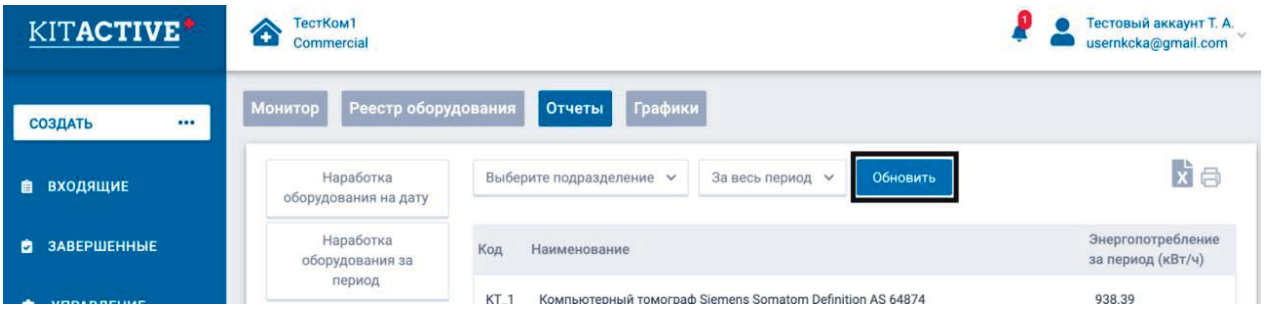

Рисунок 56 – Нажатие кнопки обновить

5) просмотр отчета *энергопотребление оборудования за период* (рисунок 57);

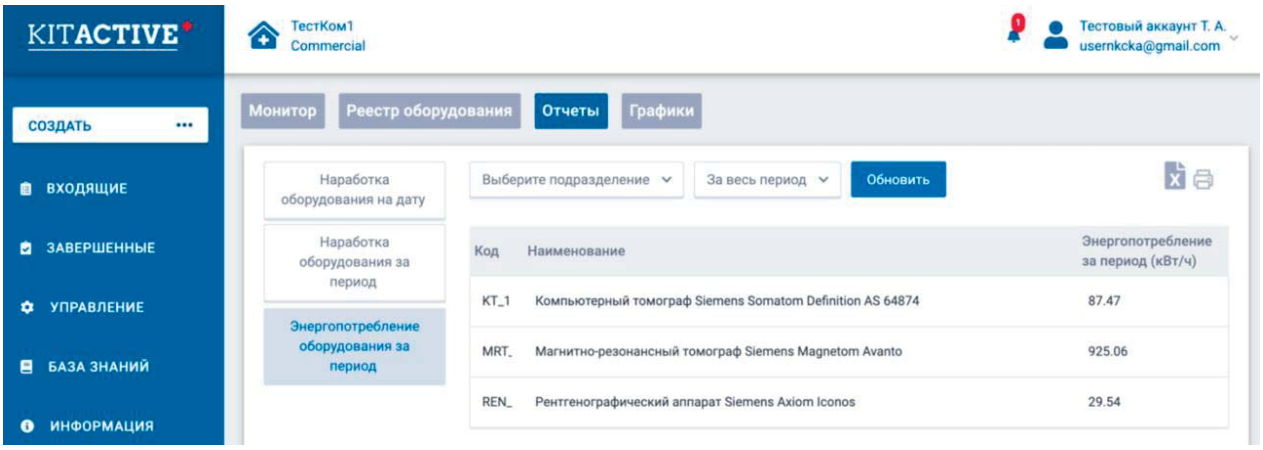

Рисунок 57 – Просмотр отчета энергопотребление оборудования за период

6) переход в пункт меню *«Телеметрия/Графики»* (рисунок 58);

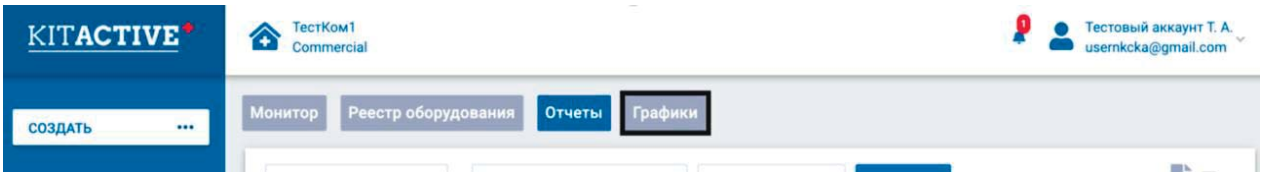

Рисунок 58 – Переход в пункт меню «Телеметрия/Графики»

7) переход в пункт меню *«Телеметрия/Графики/График энергопотребления»* (рисунок 59);

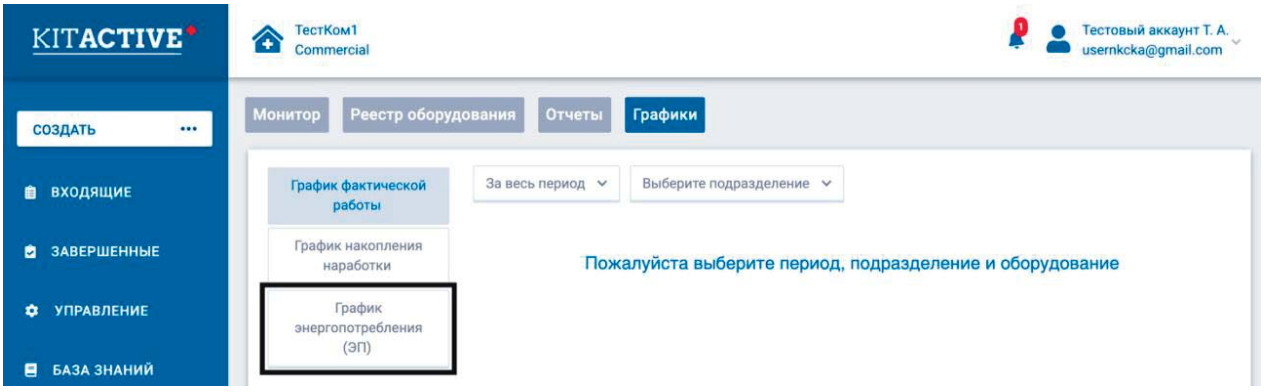

Рисунок 59 – Переход в пункт меню «Телеметрия/Графики/График энергопотребления»

8) выбор необходимой *детализации*, *периода работы*, *подразделения* и *оборудования* (рисунок 60);

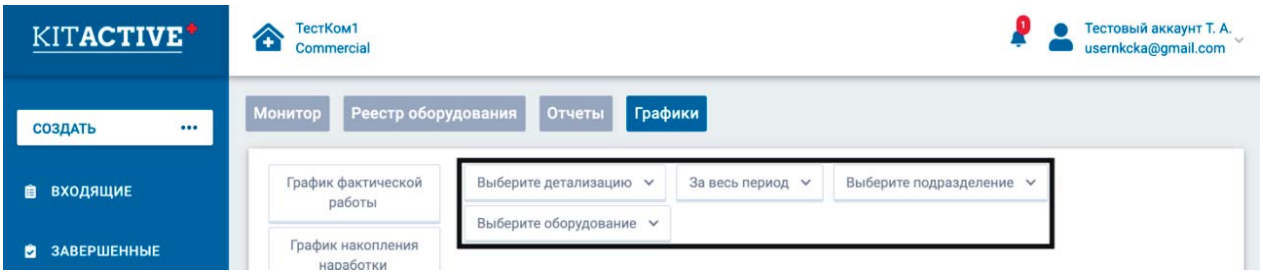

Рисунок 60 – Выбор необходимой детализации, периода работы, подразделения и оборудования

9) нажатие кнопки *обновить* (рисунок 61);

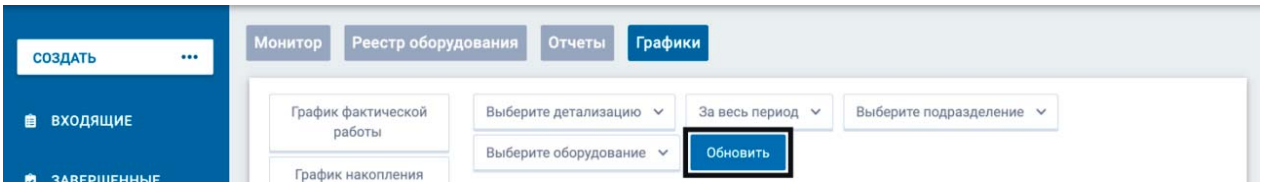

Рисунок 61 – Нажатие кнопки обновить

10) просмотр графика *энергопотребления* (рисунок 62).

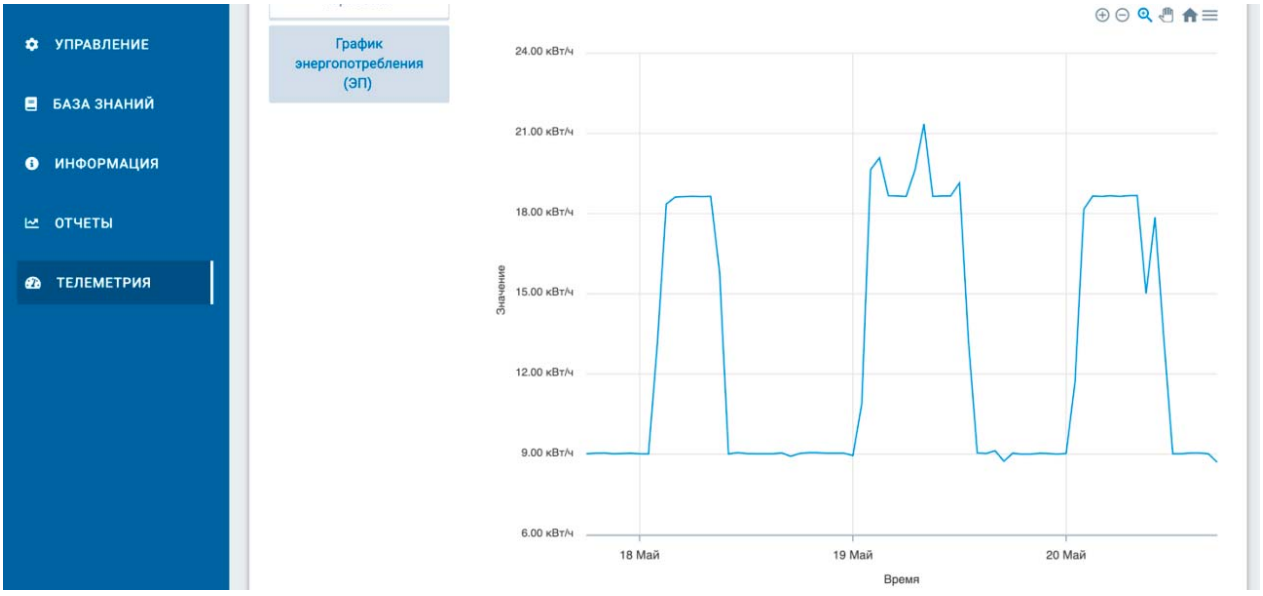

Рисунок 62 – Просмотр графика энергопотребления

Тест варианта использования «*Просмотр учета энергопотребления оборудования, подключенного к телеметрии»* пройден.

Тест варианта использования «*Создание регламентной задачи по наработке оборудования»*. После его прохождения ожидаемым результатом является создание: регламентной задачи по наработке оборудования и инцидента, при ее достижении.

Шаги тестирования:

1) переход в пункт меню *«Управление»* (рисунок 63);

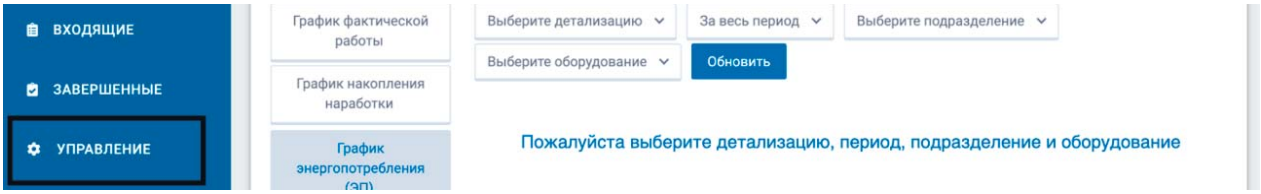

Рисунок 63 – Переход в пункт меню «Управление»

2) переход в пункт меню *«Управление/Регламентные задачи»* (рисунок 64);

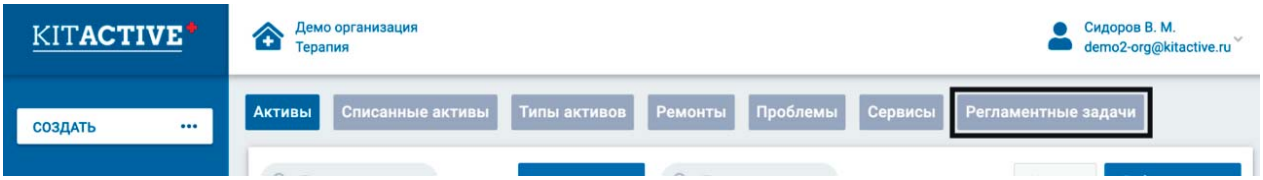

Рисунок 64 – Переход в пункт меню «Управление/Регламентные задачи»

3) нажатие кнопки *добавить регламентную задачу* (рисунок 65)

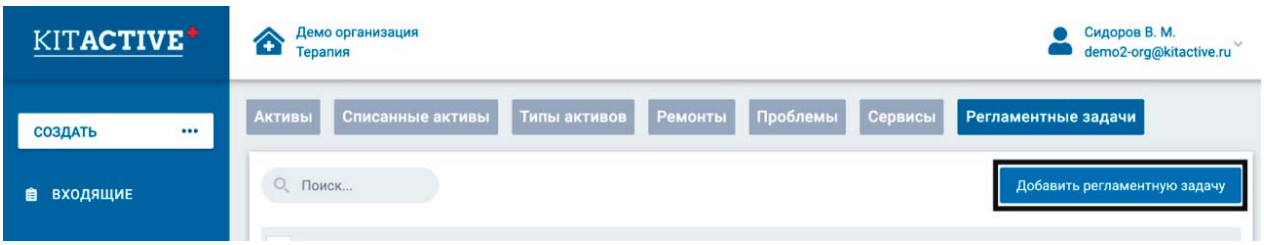

Рисунок 65 – Нажатие кнопки добавить регламентную задачу

4) заполнение необходимых полей для создания регламентной задачи по наработке (рисунок 66);

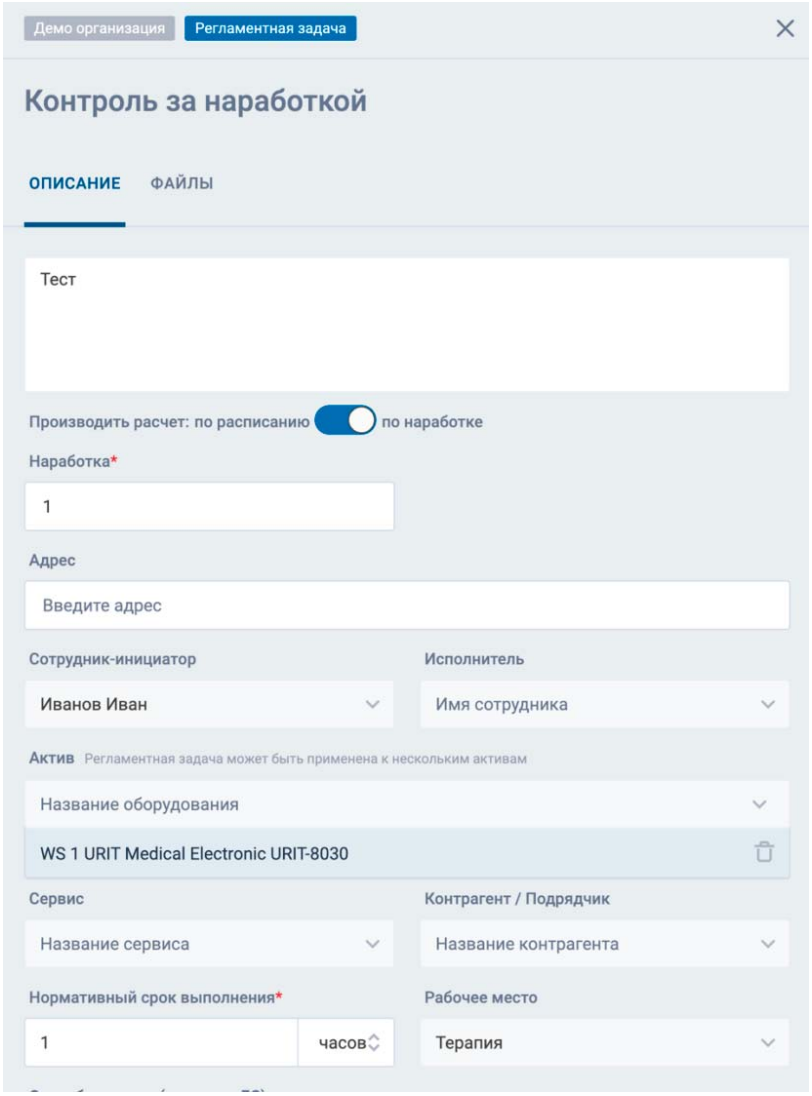

Рисунок 66 – Заполнение необходимых полей для создания регламентной задачи по наработке

5) нажатие кнопки *сохранить* (рисунок 67);

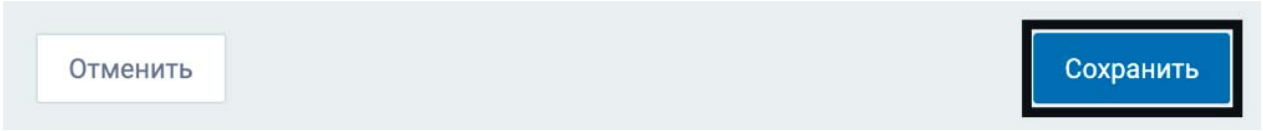

Рисунок 67 – Нажатие кнопки сохранить

6) просмотр созданной *регламентной задачи* (рисунок 68);

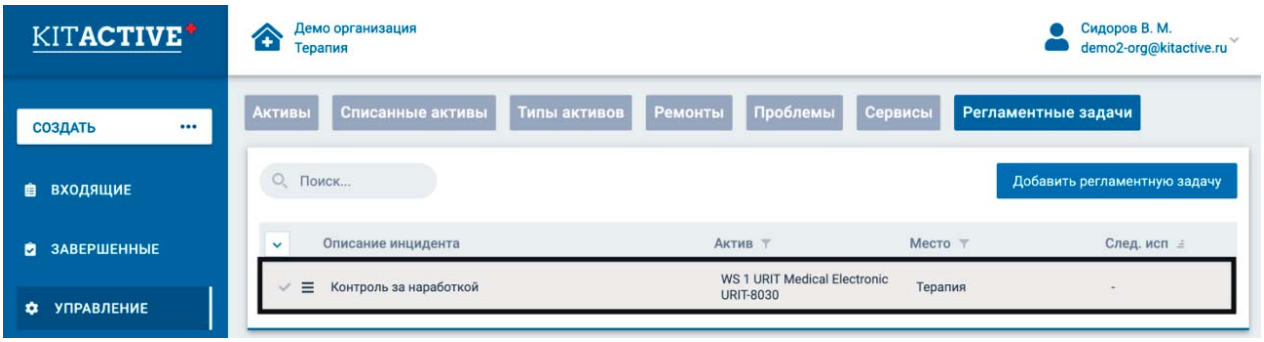

Рисунок 68 – Просмотр созданной регламентной задачи

По достижении наработки создался инцидент, представленный на рисунок 69.

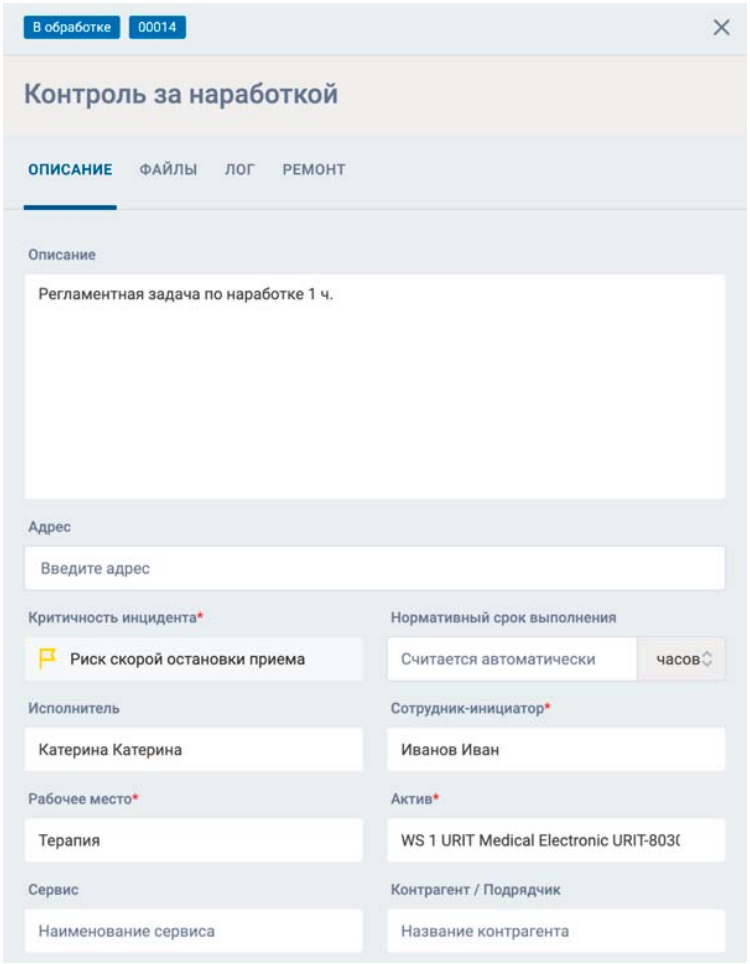

Рисунок 69 – Инцидент, созданный регламентной задачей по наработке

Тест варианта использования «*Создание регламентной задачи по наработке оборудования»* пройден.

5.2. ВЫВОД

В данной главе было проведено функциональное тестирование вариантов использования пользователя с системой. Для каждого теста был описан ожидаемый результат и шаги тестирования, с демонстрацией фрагментов интерфейса пользователя. По результатам тестирования все тесты были успешно пройдены.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы были решены все задачи выпускной квалификационной работы.

Было выполнено следующее:

- произведен анализ аналогов, разрабатываемой системы;
- выполнена детализация набора требований к приложению;
- выбраны среды и средства реализации компонентов системы;
- спроектирована архитектура приложения;
- организована база данных;
- созданы приложения первичной и вторичной обработки входной информации с устройств, подключенных к оборудованию;
- создано веб-приложение для предоставления API
- создан пользовательский интерфейс.

В настоящее время разработанная система внедрена в промышленную эксплуатацию в стоматологии «Белый Кит», акт о данном внедрении приведен в приложении Д, а также вводится в эксплуатацию в нескольких крупных медицинских учреждениях в качестве пилотного проекта, что дает хорошую обратную связь, а также хорошие отзывы.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Абдулин, И. Ш., Медицинские приборы, аппараты, системы и комплексы: учебное пособие. / И.Ш. Абдулин, Е.А. Панкова, Ф.С. Шарифулин. – Казань: Изд-во Казан. гос. Технолог. ун-та, 2011. – 106 с.
2. Веб-сайт «graphiteapp.org» [Электронный ресурс]. URL: https://graphiteapp.org/. (Дата обращения февраль 2020).
3. Веб-сайт «prometheus.io» [Электронный ресурс]. URL: https://prometheus.io/. (Дата обращения февраль 2020).
4. Веб-сайт «teleuchet.io» [Электронный ресурс]. URL: http://teleuchet.io/. (Дата обращения февраль 2020).
5. Веб-сайт «grafana.com» [Электронный ресурс]. URL: https://grafana.com/. (Дата обращения февраль 2020).
6. Веб-сайт «habr.com» [Электронный ресурс]. URL: https://habr.com/ru/company/tinkoff/blog/252907/. (Дата обращения февраль 2020).
7. Веб-сайт «dataenginer.ru» [Электронный ресурс]. URL: https://dataenginer.ru/?p=1454/. (Дата обращения февраль 2020).
8. Hunkeler, Urs, MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks / Hunkeler, Urs, Hong Linh Truong, Andy Stanford-Clark, 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08). IEEE, 2008.
9. Перри, Ли, Архитектура интернета вещей: Отдельное издание / Перри Ли - ДМК Пресс, 2018 – 454 с.
10. Light, Roger, Mosquitto: server and client implementation of the MQTT protocol / Journal of Open Source Software. – 2017. V.2. – №13. – DOI:10.21105/joss.00265.

11. Веб-сайт «muetsch.io» [Электронный ресурс]. URL: https://muetsch.io/basic-benchmarks-of-5-different-mqtt-brokers.html/. (Дата обращения февраль 2020).
12. Веб-сайт «studbooks.net» [Электронный ресурс]. URL: https://studbooks.net/2215550/informatika/bazy_dannyh_subd_prilozheniy/. (Дата обращения февраль 2020).
13. Бондарь, А. Г., Microsoft SQL Server 2014. / А.Г. Бондарь – СПб.: БХВ-Петербург, 2015. – 592 с.
14. Кайл Бэнкер, MongoDB в действии. / Слинкина А.А. – М.: ДМК Пресс, 2012. – 394 с.
15. Джуба, С., Изучаем PostgreSQL 10 / А.А. Слинкина. – М.: ДМК Пресс, 2019. – 400 с.
16. Дюбуа, П. MySQL. Сборник рецептов. / П. Шера – СПб: Символ-Плюс, 2006. – 1056 с.
17. Веб-сайт «drach.pro» [Электронный ресурс]. URL: http://drach.pro/blog/hi-tech/item/145-/. (Дата обращения февраль 2020).
18. Лучано, Рамальо, Python. К вершинам мастерства / Слинкин А.А. – М.: ДМК Пресс, 2016. – 768 с.
19. Сухов, К. К., Node.js. Путеводитель по технологии. / К.К. Сухов – М.: ДМК Пресс, 2015. – 416 с.
20. Нимейер, Патрик, Программирование на Java / Патрик Нимейер, Дэниэл Леук; М. А. Райтмана. – Москва: Эксмо, 2014. – 1216 с.
21. Веб-сайт «nuancesprog.ru» [Электронный ресурс]. URL: https://nuancesprog.ru/p/4281/. (Дата обращения февраль 2020).
22. Веб-сайт «andreyex.ru» [Электронный ресурс]. URL: https://andreyex.ru/python/nodejs-protiv-python-sravnenie-kotoroe-nuzhno-znat/. (Дата обращения февраль 2020).

23. Веб-сайт «medium.com» [Электронный ресурс]. URL: https://medium.com/webbdev/js-9ca13173577b/. (Дата обращения февраль 2020).
24. Сафронов, М., Разработка веб-приложения в Yii 2 / М. Сафронов – М.: ДМК Пресс, 2015. – 392 с.
25. Дронов, В. А., Laravel. Быстрая разработка современных динамических Web-сайтов на PHP, MySQL, HTML и CSS. – СПб.: БХВ-Петербург, 2017. – 768 с.
26. Шасанкар, К., Zend Framework 2.0 разработка веб-приложений / К. Шасанкар – СПб.: Питер, 2014. – 208 с.
27. Веб-сайт «mkdev.me» [Электронный ресурс]. URL: https://mkdev.me/posts/top-5-php-freymvorkov-laravel-vs-yii-vs-zend-vs-phalcon-vs-symfony-plyusy-i-minusy. (Дата обращения март 2020). (Автор б.д.) (Автор б.д.)
28. Стефанов, Стоян, React.js. Быстрый старт. / С. Стефанов – МПб.: Питер, 2017. – 304 с.