Московский государственный университет имени М.В.Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра автоматизации систем вычислительных комплексов

Князев Егор Иванович

Развитие сервиса сбора и обработки медицинской телеметрии

Курсовая работа

Научный руководитель:
к.ф.-м.н.
Бахмуров Анатолий Геннадьевич
Научный консультант:
Любицкий Александр Андреевич

Москва, 2022

Аннотация

Развитие сервиса сбора и обработки медицинской телеметрии

Князев Егор Иванович

Данная работа нацелена на создание нового и доработку существующего функционала сервиса сбора и обработки медицинской телеметрии на базе платформы интернета медицинских вещей. Платформа представляет собой сервис с веб-интерфейсом для операторов (врачи, администраторы системы) и приложение, предоставляющее интерфейс для пользователей (пациенты, наблюдаемые).

- Ключевой особенностью обновленной системы является возможность подключения датчиков с разным набором сенсоров. Таким образом, пользователи могут подключать к системе различные носимые устройства без привязки к конкретному типу датчиков.

- Также была переработана система операторов: появилась иерархия "администратор-оператор", позволяющая более гибко управлять правами на создание и удаление новых операторов и доступных типов устройств.

- Другим новшеством, призванным улучшить конфиденциальность данных пользователей, стала функция предоставления доступа к собранным показателям определённым операторам.

- Финальным изменением сервиса стало отображение данных пользователей в графическом виде. Такой формат вывода данных намного репрезентативнее и читабельнее, нежели табличный формат CSV, который был реализован до этого.

- Для оценки эффективности системы были определены метрики и проведены замеры производительности.

Содержание

1 Введение

1.1 Используемые понятия

Разрабатываемая платформа построена на базе технологии интернета вещей, инфраструктуры "умных" устройств, способных взаимодействовать друг с другом посредством сети без присутствия человека на каком либо этапе обмена данными. Сервис реализует сужение технологии интернета вещей на область работы с медицинскими данными, называемое интернетом медицинских вещей, эта подсфера отличается спецификой обрабатываемых данных и объемом поступающей информации с датчиков.

В рамках работы введены некоторые дополнительные понятия.

- Платформа или сервис — веб-приложение хранящее, обрабатывающее, а также предоставляющее доступ к собранным данным,
- Пользователь — пациент или наблюдаемый, который генерирует медицинские данные для хранения и обработки,
- Оператор — работник медицинского учреждения, зарегистрированный в сервисе для анализа показателей полученных от пользователя,
- Администратор — оператор, имеющий расширенные права для администрирования системы, например создание и удаление новых операторов или датчиков,
- Датчик — специализированное носимое устройство, предназначенное для снятия физиологических показателей пользователя,
- Сенсор — часть датчка, отвечающая за снятие показателей определённого типа.

1.2 Актуальность

Результатом работы является отечественная система итернета медицинских вещей с открытым исходным кодом, позволяющая в случае медициской телеметрии разворачивать массовые эксперементы по наблюдению за физиологическими показателями жизнедеятельности человека, а также проводить исследования с использованием различных

типов датчиков. Кроме того, у работы есть потенциал для создания платформы нацеленной на более широкую область интернета вещей, от "умных" домов, до систем "умного" города, что является не менее актуальной задачей на сегодняшний день.

1.3 Базовая реализация

В качестве фундамента для развития сервиса была предоставлена платформа разработанная в рамках выпускной квалификационной работы Аникевич Юлии Вадимовны [1]

2 Постановка цели и задач

2.1 Цель

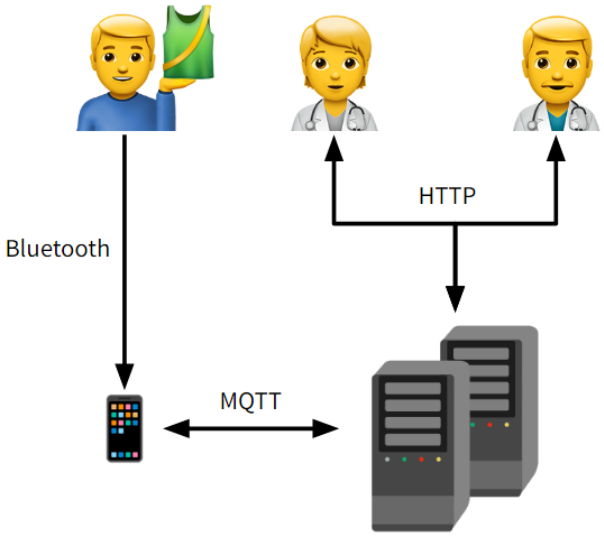
Необходимо модифицировать сущестующую платформу для операторов, посредством добавления нового и улучшения уже реализованного функционала.

2.2 Задачи

- Предусмотреть возможность создания и удаления ключевых рабочих объектов системы, таких как операторы и типы датчиков, без специфичных навыков работы с коммандной строкой и знания программирования,
- Реализовать систему защиты собранных конфиденциальных пользовательских данных и интерфейс предоставления доступа к ним,
- Предоставить операторам корректную визуализацию данных пользователей посредством графиков,
- Унифицировать платформу в направлении поддержки различных типов датчиков с разными наборами сенсоров.

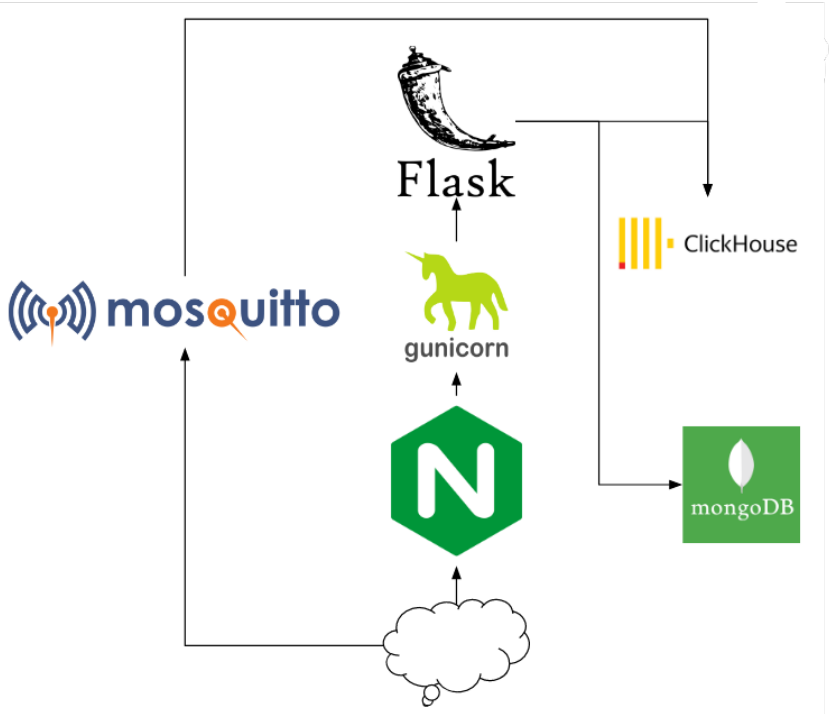
3 Обзор используемых технологий

3.1 Общая архитектура работы сервиса



Архитектура системы предусматривает, как минимум, двух действующих лиц: пользователя и оператора. Пользователь, посредством носимых датчиков, коими являются различные фитнес браслеты, "умные" жилетки и другие устройства, способные считывать физиологические показатели человека, передает собранные данные на смартфон с установленным приложением по протоколу Bluetooth[2]. Смартфон, в свою очередь, передаёт предобработанную информацию на сервер по протоколу MQTT[3], где она обрабатывается и записывается в базу данных, для последующего представления оператору. Процесс работы оператора выстроен следующим образом: оператор подключается к сервису через веб-интерфейс, вводит свои учетные данные в системе, после чего ему доступен вывод информации о пользователях, давших разрешение на просмотр их данных.

3.2 Архитектура платформы для операторов



Рассмотрим подробнее внутреннюю архитектуру платформы для операторов. Сервис реализует взаимодействие с пользовательским приложением и операторами посредством двух протоколов: MQTT и HTTP. Рассмотрим каждый из них по отдельности.

3.2.1 MQTT

MQTT[3] - сетевой протокол прикладного уровня, работающий поверх стека TCP/IP и ориентированный на обмен сообщениями между устройствами по принципу издатель-подписчик, в основном применятеся для обмена данными между устройствами интернета вещей. Данный протокол используется в системе как основной для обмена физиологическими данными между приложением и сервисом. Так как протокол основан на взаимодействии клиентов посредством брокера, сервис реализует сразу оба типа сущностей протокола. В качестве брокера выбран Eclipse Mosquitto[4], так как он распространяется по схеме open source и имеет большое и активное сообщество. Клиент реализован в формате демона написанного на языке Python с использованием библиотеки Paho-mqtt[5], так как она является рекоммендуемой для работы с Eclipse Mosquitto. Логика клиента заключается в том, чтобы записывать полученные по протоколу MQTT данные

в базу данных ClickHouse.

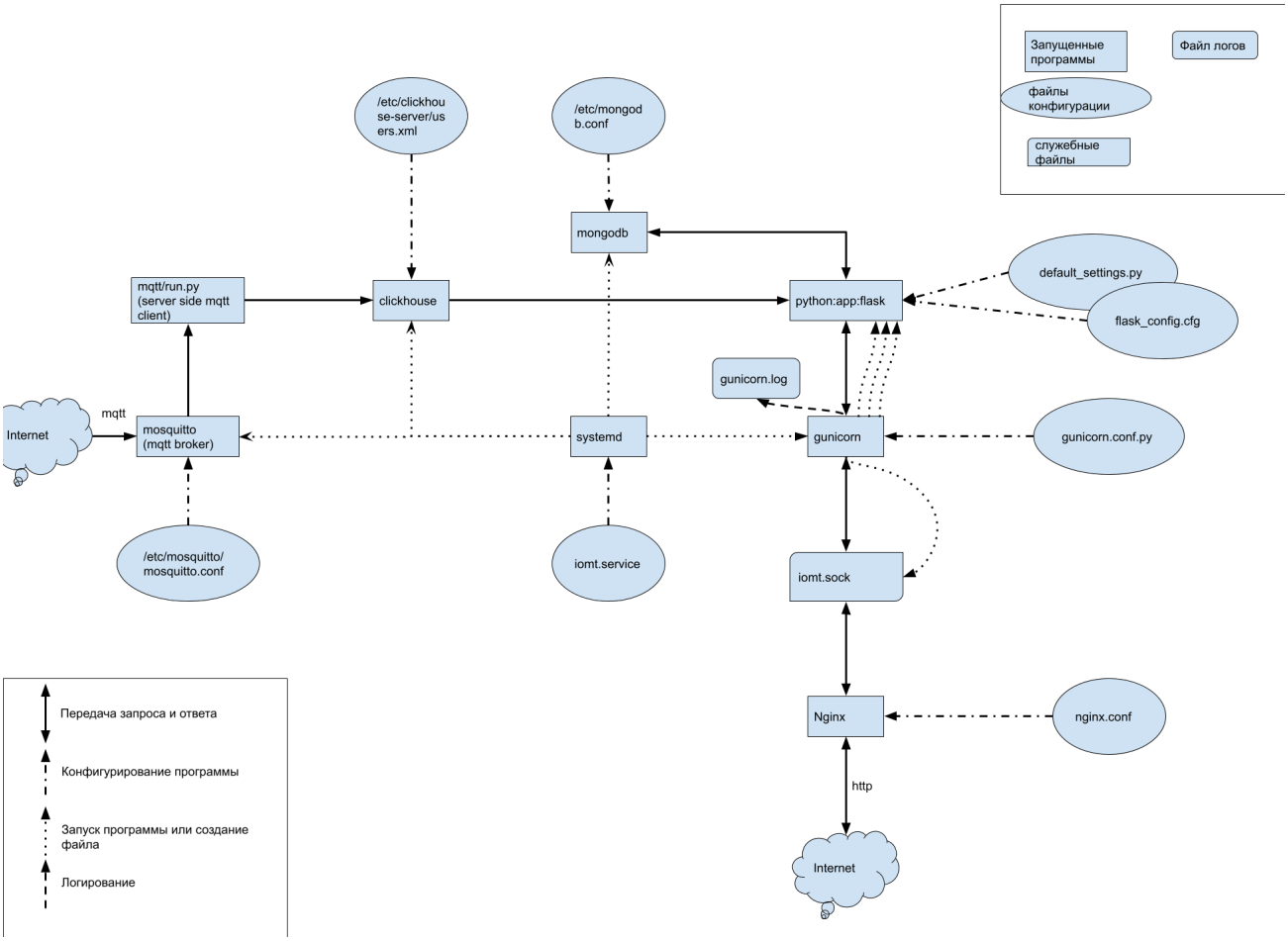
3.2.2 HTTP

HTTP - основной протокол прикладного уровня используемый в сети интернет. При получении HTTP запроса операционная система перенаправляет его на слушающий соответствующий порт веб-сервер. В качестве веб-сервера был выбран nginx, так как он является наиболее современным, производительным и надёжным инструментом для обработки запросов. Более того этот веб-сервер имеет открытый исходный код. Nginx передаёт все входящие запросы в специальный unix-сокет, откуда информация читается инструментом под названием gunicorn. Gunicorn[6] - это так называемый "WSGI сервер" предназначенный для обеспечения взаимодействия Python программы и веб-сервера, также в задачи wsgi-сервера входит запуск нескольких интерпретаторов приложений на Python, делается это в связи с Global Interpreter Lock языка Python. Вся логика сервиса реализована в программе на языке Python с применением фреймворка Flask. В ней происходит рендеринг возвращаемого HTML кода с помощью библиотеки Jinja2, выполняются запросы к базам данных ClickHouse и MongoDB, описан интерфейс взаимодействия с пользовательским приложением и другие действия.

3.2.3 Базы данных

Как уже было упомянуто, в системе используется связка из двух баз данных MongoDB[7] и ClickHouse[8]. Такое разбиение свзано с разнородностью данных, которые требуется хранить. В первой хранится классическая для платформы информация, такая как учётные данные пользователей и операторов, типы поддерживаемых датчиков и так далее. В то время как в последней хранится информация собранная с датчиков, привязанная к временному ряду. Детальный обзор и критерии выбора СУБД приведен в курсовой работе Юлии Аникевич[9].

Ниже приведена подробная схема внутреннего устройства серверной части разрабатываемой системы.


Запущенные программы
Файл логов
файлы конфигурации
служебные файлы
/etc/clickhouse-server/users.xml
/etc/mongodb.conf
mongodb
mqtt/run.py (server side mqtt client)
clickhouse
python:app:flask
default_settings.py
flask_config.cfg
gunicorn.log
Internet
mqtt
mosquitto (mqtt broker)
systemd
gunicorn
gunicorn.conf.py
/etc/mosquitto/mosquitto.conf
iomt.service
iomt.sock
Передача запроса и ответа
Конфигурирование программы
Запуск программы или создание файла
Логирование
Nginx
nginx.conf
http
Internet

4 Исследование и построение решения задачи

4.1 Интерфейс администратора

Решением задачи предоставления возможнасти создания или удаления операторов или датчиков стал интерфейс администратора. Была введена иерархия операторов, где оператор с максимальными правами был назван администратором. Под максимальными правами подразумевается возможность создания или удаления таких объектов как оператор или датчик. На главной странице платформы у администратора доступна кнопка перехода в панель администратора, эта панель отображает всех зарегистрированных в системе операторов и все доступные типы датчиков. Для удаления и создания соответствующих объектов доступны отдельные кнопки. Данная иерархия предлагает надёжный фундамент для добавления более гибкой системы прав, с различными ролями.

4.2 Конфиденциальность данных пользователей

Конфиденциальность данных чрезвычайна важна при хранении пользовательской информации. Таким образом, кроме внешних утечек необходимо предотвращать и внутренние утечки, ситуации, когда злоумышленник тем или иным способом получил доступ к сервису и через него пытается получить информацию. Для предотвращения таких случаев был разработан интерфейс предоставления разрешения на доступ к пользовательским данным определенным операторам. Для этого пользователю необходимо лишь перейти по уникальной ссылке нужного оператора. После этого в интерфейсе оператора появится соответствующий пользователь.

4.3 Визуализация собранной информации

В базовой версии сервиса была раелизована выдача информации в виде CSV файла. Это удобно для хранения и переноса информации, но работать с таким представлением весьма неудобно. В дополнение к такому способу получения данных был реализован раздел представляющий собой графики отображающие значения определённого показателя от времени.

4.4 Поддержка разных типов датчиков

Для унификации платформы была реализованна поддержка различных датчиков и возможность добавления нового устройства. Для добавления устройства администратору необходимо из панели администратора добавить новое устройство, ввести информацию о датчике, такую как поддерживаемые сенсоры, название и так далее. После сохранения новый датчик появится в списке доступных для подключения устройств.

5 Описание практической части

5.1 Администраторы

Так как администратор является расширением такой сущности как оператор, реализован он аналогично, как унаследованный от оператора класс. Таким образом для проверки является ли оператор администратором необходимо лишь проверить соответствие оператора типу "Администратор". Уровень оператора записывается в базе данных как обычное поле, но работы с этим полем на прямую не происходит благодаря поддержке драйвером базы данных такого рода объектов.

5.2 Предоставление оператору доступа к данным

В качестве реализации интерфейса предоставления доступа к данным оператору была создана новая конечная точка. Конечная точка — это логика работы сервиса соответствующая одному конкретному URL. Эта конечная точка доступна исключительно для аутенфицированных пользователей и принимает один аргумент в виде уникального идентификатора оператора, после чего добавляет соответствующего оператора в список людей допущенных к данным пользователя.

5.3 Разные типы датчиков

Для реализации возможности добавления новых типов датчиков были дабавлены новые представления в базе данных такие как "Device"и "Sensor". Рассмотрим их подробнее. Датчик имеет следующий набор полей:

- name — имя устройства,
- nameRegex — регулярное выражение соответствующее имени датчика, это поле необходимо для определения типа датчиков даже если они имеют разные идентификаторы в названии,
- device_type — один из существующих типов устройств, на данный момент доступны жилет и браслет.

Сенсор имеет следующий набор параметров:

- name — название сенсора,
- serviceUUID и characteristicUUID — параметры необходимые для корректного распознавания сенсора приложением,
- devices — список ссылок на датчики в которые входит данный сенсор.

5.4 Оптимизации

В процессе работы были проделаны различные оптимизации существующего кода. Например для систематизиции кодовой базы единый файл со всей логикой работы сервиса был разбит на файлы-чертежи (blueprints), Python файлы реализующие отдельный конкретный функционал, в случае данной платформы был создан blueprint с реализацией протокола взаимодействия с приложением и отдельный blueprint для логики панели администратора. Остальная часть кода осталась в объемлющем все чертежи файле.

Также была произведена реорганизация конфигурационных файлов. Для более гибкой настройки системы. Был добавлен отдельный файл конфигурации фреймворка Flask, где указываются небоходимые для работы константы, этот файл зависит от разворачиваемого окружения. Стандартный конфигурационный файл с незаполненными полями также создан для более удобного распространения и развёртывания данной системы. Также вместо задачи конфигурации WSGI-сервера через аргументы командной строки был создан файл со всеми необходимыми параметрими gunicorn.

6 Дальнейшее развитие

Разрабатываемая платформа готова к запуску в реальных условиях на стадии альфа-тестирования. Для полноценного запуска необходимо найти и исправить макимальное количесвто неисправностей, а также добавить добполнительный функционал.

- Предлагается расширить работу клиентской стороны платформы с серверной посредством API, там самым избавиться от статической генерации страниц и приблизиться к одностраничному приложению [10]. Данная архитектура реализует быстрое и интерактивное обновление контента на странице,
- Реализовать поддержку адаптивного веб-дизайна[11], в первую очередь для корректной работы платформы на мобильных устрйствах,
- Внутренние оптимизации: введение в рабочий функции flask_mongoengine.model_form позволит писать компактный код избегая повторения, реализовать более сложную логику аутенфикации пользователей с разбиением на уровни поераторов.
- Необходимо внедрить более строгие правила документирования программного кода с последующим ковертированием его в один из доступных форматов: PDF, HTML и другие. Также предлагаетсяя разделить внутреннюю документацию по работе сервиса и внешнюю по работе с API.

Предложенный вектор развития платформы способствует улучшению как пользовательского опыта, так и удобства разработки под данную систему.

7 Заключение

Полученный результат удовлетворяет поставленным задачам. Интерфейс администратора предоставляет все необходимые инструменты для добавления и удаления ключевых объектов сервиса. Система доступа к данным предотвращает неконтролируемый доступ к данным пользователей от любого оператора и гарантирует доступность данных только определённым пользователем операторам. Графическое отображение информации о показателях пользователей способствует более глубокому анализу данных со стороны оператора. Датчики и сенсоры оказались гибким и надежным решением для унификации платформы.

Список литературы

[1] *Аникевич, Юлия.* Разработка и реализация серверной части платформы для сбора и обработки медицинской телеметрии / Юлия Аникевич.

[2] *Group, Core Specification Working.* Bluetooth Core Specification. — 2021. — Дата обращения: 23.05.2022. https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=521059.

[3] *Coppen, Richard.* MQTT Version 5.0 OASIS Standard. — 2019. — Дата обращения: 23.05.2022. https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.pdf.

[4] *Light, Roger A.* Mosquitto: server and client implementation of the MQTT protocol / Roger A. Light // *Journal of Open Source Software.* — 2017. — Vol. 2, no. 13. — P. 265. https://doi.org/10.21105/joss.00265.

[5] *Craggs, Ian.* Eclipse Paho. — 2021. — Дата обращения: 23.05.2022. https://projects.eclipse.org/projects/iot.paho.

[6] *Chesneau, Benoit.* Gunicorn documentation page. — 2022. — Дата обращения: 23.05.2022. https://docs.gunicorn.org/en/stable/index.html.

[7] *Merriman, Dwight.* MongoDB. — 2022. — Дата обращения: 23.05.2022. https://www.mongodb.com/.

[8] *Katz, Aaron.* ClickHouse. — 2022. — Дата обращения: 23.05.2022. https://clickhouse.com/.

[9] *Аникевич, Юлия.* Разработка и реализация серверной части платформы для сбора и обработки физиологических данных о человеке / Юлия Аникевич.

[10] *Santamaria, Jose Maria Arranz.* The Single Page Interface Manifesto. — 2015. — Дата обращения: 23.05.2022. http://itsnat.sourceforge.net/php/spim/spi_manifesto_en.php.

[11] *Gustafson, Aaron.* Adaptive Web Design: Crafting Rich Experiences with Progressive Enhancement / Aaron Gustafson. — 2015.