Московский государственный университет имени М. В. Ломоносова
Факультет Вычислительной Математики и Кибернетики

Аникевич Юлия Вадимовна

Разработка и реализация серверной части платформы для сбора и обработки физиологических данных о человеке

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:
к.ф.-м.н., в.н.с.
Бахмуров А.Г.

Консультант:
Любицкий А.А.

Москва, 2020

Аннотация

Данная работа посвящена разработке серверной части платформы для сбора и обработки физиологических данных о человеке.

Проведен обзор систем управления базами данных для хранения и обработки временных рядов, на основе которого было проведено экспериментальное сравнение выбранных решений по производительности, также разработан протокол взаимодействия с приложением на мобильном устройстве на основе протокола MQTT. В рамках практической работы разрабатывается программное обеспечение, предоставляющее возможность хранения и обработки данных мониторинга.

Содержание

1 Введение . . . 3
2 Цели и задачи . . . 5
2.1 Цели . . . 5
2.2 Задачи . . . 5
3 Обзор СУБД . . . 6
3.1 Цель обзора . . . 6
3.2 Критерии обзора . . . 6
3.3 Основные понятия . . . 6
3.4 Сценарий использования БД и обусловленные им требования к СУБД . . . 7
3.5 Результаты обзора . . . 8
3.6 Выводы из обзора . . . 9
4 Экспериментальное сравнение СУБД . . . 10
4.1 Сценарий тестирования . . . 10
4.2 Типы запросов . . . 11
4.3 Результаты . . . 11
4.3.1 Загрузка данных . . . 11
4.3.2 Степень сжатия данных . . . 12
4.3.3 Время выполнения запросов . . . 12
4.4 Выводы из сравнения . . . 13
5 Описание реализации . . . 14
5.1 Общая схема работы . . . 14
5.2 Протокол передачи данных . . . 14
5.2.1 MQTT . . . 14
5.2.2 Структура топиков . . . 15
5.2.3 Формат передачи данных . . . 16
5.3 Аутентификация и авторизация . . . 17
5.4 Работа с базой данных . . . 18
6 Заключение . . . 19
7 Список литературы . . . 20

1 Введение

С каждым днем все известнее становится такое направление в здравоохранении, как интернет медицинских вещей (IoMT, Internet of Medical Things). Данная формулировка обозначает концепцию сети, которая позволяет объединять устройства и приборы, способные отслеживать состояние человеческого организма, окружающую его среду и влиять на профилактический, лечебный и реабилитационный процессы.

На сегодняшний день проблема мониторинга состояния человека вне медицинских учреждений чрезвычайно актуальна. Данное направление существует с целью получения детальной информации об организме человека, на основании которой можно принимать обоснованные решения, предотвращающие возникновение и развитие заболеваний. Современная концепция позволяет проводить непрерывный контроль за состоянием человека, и, с помощью оценки диагностических показателей, выявлять случаи отклонения их от нормы.

Персональные мониторные устройства способны выявить нарушения в работе систем организма на более ранней стадии, а также могут послужить дополнительным источником информации для специалистов. Также, такая система позволяет оперативно предупредить критическое состояние пациента, что даст возможность вовремя провести необходимые мероприятия для его стабилизации.

К настоящему времени сформирован обширный рынок портативных носимых устройств и систем для регистрации физиологических параметров человека. Большинство из них имеет свои решения для обработки и хранения собираемых данных. Однако в большинстве своем они не являются open-source проектами, и данные из них получить либо невозможно, либо они выдаются уже в обработанном виде. Таким образом, создание открытой платформы сбора и обработки физиологических данных является актуальной задачей на сегодняшний день.

В данной работе в качестве носимого устройства было взята умная одежда Hexoskin, которая имеет открытый программный интерфейс и является разработкой компании Carre Technologies Inc. Умная одежда - это одежда, которая может взаимодействовать с окружающей средой, воспринимая сигналы, обрабатывая информацию и запуская ответные реакции. Одежда Hexoskin способна производить мониторинг таких показателей, как ЭКГ и сердцебиение, вариабельность сердечного ритма, события QRS, восстановление сердечного ритма, частот дыхания, минутная вентиляция, интенсивность активности, пиковое ускорение, шаги, ритм, позиции и трекер сна (Рис. 1).
https://www.hexoskin.com

Данная работа посвящена созданию серверного программного обеспечения платформы для сбора и обработки физиологических данных о человеке. В раз-

Рис. 1: Умная одежда Hexoskin

деле 2 описаны цели и задачи данной курсовой работы. В разделе 3 проведён обзор систем управления базами данных, на основе которого был сделан выбор СУБД для экспериментального сравнения по производительности, которое описано в разделе 4. В разделе 5 изложено описание реализации разрабатываемого серверного программного обеспечения: описана реализация аутентификации и авторизации, использование протокола передачи данных между клиентом и сервером и работа с базой данных.

2 Цели и задачи

2.1 Цели

1. Создание серверной части платформы для сбора и обработки физиологических данных о человеке.
2. Получение опыта разработки с использованием широко распространенных инструментов разработки и систем управления базами данных с открытым исходным кодом.

2.2 Задачи

1. Изучить порядок работы с протоколом MQTT.
2. Сформулировать критерии сравнения СУБД. Отобрать СУБД для экспериментального сравнения.
3. Провести экспериментальное сравнение отобранных СУБД по производительности, для этого разработать сценарий проведения экспериментов.
4. Разработать протокол взаимодействия с приложением на мобильном устройстве.
5. Реализовать цепочку получения данных с Hexoskin с записью в базу данных.
6. Реализовать аутентификацию и авторизацию пользователей.
7. Реализовать поддержку протокола MQTTS.

3 Обзор СУБД

3.1 Цель обзора

Целью данного обзора является выбор систем управления базами данных для хранения и обработки временных рядов для экспериментального сравнения по заданным критериям.

3.2 Критерии обзора

1. Открытость исходного кода
2. Долговременное хранение
3. Масштабируемость
4. Гранулярность

3.3 Основные понятия

Под временным рядом понимается совокупность значений какого-либо показателя, собранного в различные моменты времени. Например, данные, получаемые с акселерометра или показатели ЭКГ можно рассматривать как временные ряды. Измерения, составляющие временной ряд, упорядочены по временной шкале, что показывает историю изменения данных во времени и может быть очень полезным для медицинской диагностики.

В рамках этой работы система управления базами данных временных рядов будет определена как СУБД, которая может:

- хранить строку, состоящую из метки времени, значений и необязательных тегов,
- хранить несколько строк данных временного ряда, сгруппированных вместе,
- запрашивать строки данных,
- содержать временную метку или временной диапазон в запросе.

СУБД временных рядов более удобны для пользователя и обеспечивают большую скорость записи и более высокую производительность запросов, несмотря на большой объем данных, которые они организуют. В некоторых случаях СУБД временных рядов выполняют те же функции, что и обычные СУБД. Однако попытка использовать реляционную базу данных или базу данных NoSQL для данных временных рядов приведет к гораздо более медленной и более низкой производительности.

Современные решения все чаще испытвают необходимость в больших объемах, более высоких скоростях и более высокой специфичности при поиске данных. В последние годы эти требования привели к сильному увеличению использования систем упарвления данных временных рядов.

Запросы к базе данных временных рядов аналогичны запросам в других типах баз данных, но вместо поиска по значениям пользователи могут осуществлять поиск по прошедшему периоду времени, диапазону дат или конкретному моменту времени, когда произошло событие.

В данном обзоре будут рассмотрены системы управления базами данных, оптимизированные для хранения и обработки временных рядов.

Под масштабируемостью понимается возможность увеличить объем хранилища или производительность за счет добавления дополнительных узлов.

Долговременное хранение необходимо для больших объемов данных, поскольку не все СУБД могут хранить все значения в полном разрешении в течение длительного периода времени. Старые данные могут удаляться или храниться в агрегированном состоянии (например, хранение среднего значения за минуту вместо значений для каждой миллисекунды).

Гранулярность описывает наименьшее возможное расстояние между двумя временными метками. При вставке данных в базу данных степень детализации входных данных может быть выше, чем степень детализации хранилища, для которой СУБД гарантирует безопасное хранение. Некоторые СУБД принимают данные с меньшей степенью детализации, чем они могут хранить, что приводит к агрегированию или потере данных.

3.4 Сценарий использования БД и обусловленные им требования к СУБД

База данных представляет собой набор таблиц, где каждая таблица отведена под конкретного пользователя. В таблицы будет производиться запись семи измерений, в соответствии параметрами, получаемыми с датчиков Hexoskin.

Согласно информации о датчиках Hexoskin, самая высокая частота получения данных имеет значение 256 Гц. Строго говоря, для выбора СУБД важно, чтобы она имела возможность сохранять данные с разрешением в 4 мс. Предполагаем, что гранулярность принимает значения 1 мс, 10 мс, 100 мс и т.д. Таким образом, необходимая гранулярность имеет значение 1 мс.

Необходимость в долговременном хранении обусловлена возможностью хранить данные мониторинга за длительный промежуток времени, что позволяет получить более полную картину о состоянии человека.

Высоким приоритетом при выборе обладает такой критерий, как время выполнения запросов, то есть важно, чтобы чтение из базы данных происходило достаточно быстро. Более низким – время записи в базу данных и степень сжатия записанных данных на диске.

3.5 Результаты обзора

Для обзора были взяты СУБД временных рядов, которые занимают первые десять позиций в рейтинге DB-Engines СУБД временных рядов. DB-Engines [1] - это независимый веб-сайт, который ранжирует системы управления базами данных на основе популярности в поисковых системах, упоминаний в социальных сетях, количества предложений о работе и частоты технических обсуждений. Рейтинг датируется маем 2020 года (Рис. 2)

Rank			DBMS	Database Model	Score		
May 2020	Apr 2020	May 2019			May 2020	Apr 2020	May 2019
1.	1.	1.	InfluxDB	Time Series	20.92	-0.70	+2.84
2.	2.	2.	Kdb+	Time Series, Multi-model	5.37	+0.09	-0.23
3.	3.	↑ 4.	Prometheus	Time Series	4.31	+0.06	+1.20
4.	4.	↓ 3.	Graphite	Time Series	3.46	+0.03	+0.23
5.	5.	5.	RRDtool	Time Series	2.52	-0.09	-0.37
6.	↑ 7.	↑ 7.	Druid	Multi-model	1.91	-0.01	+0.22
7.	↓ 6.	↓ 6.	OpenTSDB	Time Series	1.82	-0.19	-0.65
8.	8.	8.	TimescaleDB	Time Series, Multi-model	1.77	-0.10	+0.61
9.	9.	↑ 11.	FaunaDB	Multi-model	0.95	+0.07	+0.56
10.	10.	↓ 9.	KairosDB	Time Series	0.57	+0.01	+0.02

Рис. 2: uge crowd facing the distant stage during the Woodstock Music Art Fair in August 1969

Также в список обозреваемых СУБД отдельно включается ClickHouse, разрабатываемая компанией Яндекс, в виду хороших результатов, показанных в статье [2], где сравнивается производительность СУБД ClickHouse, InfluxDB и TimescaleDB.

Первым критерием обзора является открытость исходного кода, поэтому далее такие решения, как Kdb+ и FaunaDB не рассматриваются.

Результаты сравнения по заданным критериям приведены в таблице на рис. 3.[1] [2] Таким образом, по результатам сравнения, всем перечисленным критериям удовлетворяют InfluxDB, ClickHouse и Druid.

Из статьи [3] можно сделать вывод, что Druid больше подходит для сценариев, когда имеется большой кластер с большим количеством таблиц и данных.

[1] Доступно в InfluxEnterprise[4]
[2] См. [9]

	Масштабируемость	Долговременное хранение	Наименьшая гранулярность для хранения	Наименьшая гранулярность для безопасного хранения
InfluxDB	(+) [1]	+	1 мс	1 мс
Prometheus	+	-	1 мс	1 мс
Graphite	+	+	1000 мс	1000 мс
RRDtool	-	-	1000 мс	1000 мс
Druid	+	+	1 мс	1 мс
OpenTSDB	+	-	1 мс	> 1 мс (1000 мс) [2]
TimescaleDB	-	+	1 мс	1 мс
KairosDB	+	-	1 мс	1 мс
ClickHouse	+	+	1 мс	1 мс

Рис. 3: Таблица сравнения СУБД

Таблицы и наборы данных периодически появляются в кластере, анализируются и удаляются из него в отличие от ClickHouse, где таблицы и данные находятся в кластере перманентно. Поэтому Druid в дальнейшем сравнении не рассматривается.

3.6 Выводы из обзора

В данном разделе было проведен обзор одиннадцати систем управления базами данных временных рядов по таким критериям, как открытость исходного кода, возможность масштабирования и долговременного хранения, а также наименьшей гранулярности для хранения данных. В результате обзора были выбраны две СУБД InfluxDB и ClickHouse для дальнейшего сравнения по производительности.

4 Экспериментальное сравнение СУБД

4.1 Сценарий тестирования

В сравнении на производительность используются СУБД версий InfluxDB версии v1.8.0 и ClickHouse версии v19.3.3.

InfluxDB

InfluxDB [4] - это нереляционная система управления базами данных временных рядов, разработанная компанией InfluxData, которая имеет открытый исходный код с дополнительными компонентами с закрытыми исходными кодами. Она написана на языке программирования Go и оптимизирована для обработки временных рядов. Поддерживает SQL-подобный язык запросов.

Время является самой важной концепцией в InfluxDB. Столбец времени включен в каждую базу данных InfluxDB и содержит дискретные временные метки, которые связаны с конкретными данными.

ClickHouse

ClickHouse [5] – это аналитическая колоночная база данных с открытым исходным кодом, разработанная компанией Яндекс для OLAP сценариев работы. Язык запросов ClickHouse представляет собой диалект SQL.

Колоночные базы данных хранят записи в блоках, сгруппированных по столбцам, а не по строкам. Поскольку такая БД не загружает данные для столбцов, которых нет в запросе, она тратит меньше времени на чтение данных при выполнении запросов. Потому колоночные базы данных могут вычислять и возвращать результаты для определенных рабочих нагрузок, таких как OLAP, намного быстрее, чем традиционные строчные системы.

Тестовый стенд

- Ubuntu 18.04.3
- Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30GHz
- 3,9 Gb RAM
- L2 cache: 4096K
- L3 cache: 16384K

Тестовый набор данных

Для генерации тестового набора данных была воспроизведена непрерывная запись данных с самой высокой частотой из всех датчиков Hexoskin. Временной

промежуток генерации составил около двух недель. В конечном итоге, набор данных состоит из 398.458.880 строк и содержит 16 показателей. Общий объем тестовых данных составляет 55 Гигабайт.

4.2 Типы запросов

Для сравнения выбранных СУБД по времени выполнения запросов было использовано 8 типов запросов, характерных для работы с временными рядами. Предположим, что v1, v2, ..., vn – значения, соответствующие датчикам 1, 2, ..., n. Также в нижеприведенных запросах func() представляет собой функцию агрегации, такую как avg, min и т.д.

1. Точечный запрос
   SELECT v1 ... FROM data WHERE time = ?
2. Запрос временного диапазона
   SELECT v1 ... FROM data WHERE time >? AND time <?
3. Запрос с фильтром значений
   SELECT v1 ... FROM data WHERE v op ? (op: <,>,=)
4. Запрос временного диапазона с фильтром значений
   SELECT v1 ... FROM data WHERE time > ? AND time < ? AND v1 op ? (op: <,>,=)
5. Запрос с лимитом строк
   SELECT v1 ... FROM data LIMIT ?
6. Запрос с функцией агрегации
   SELECT func(v) FROM data
7. Запрос временного диапазона с функцией агрегации
   SELECT func(v) FROM data WHERE time > ? AND time < ?
8. Агрегация + group by
   SELECT func(v) FROM data GROUP BY time(?)

4.3 Результаты

4.3.1 Загрузка данных

Для загрузки тестового набора данных в базы данных был использован интерфейс командной строки и утилита time для измерения времени загрузки. Загрузка данных в ClickHouse заняла 712 секунд, а в InfluxDB – 2367 секунд, что в 3,3 медленнее, чем время загрузки данных в ClickHouse (Рис.4).

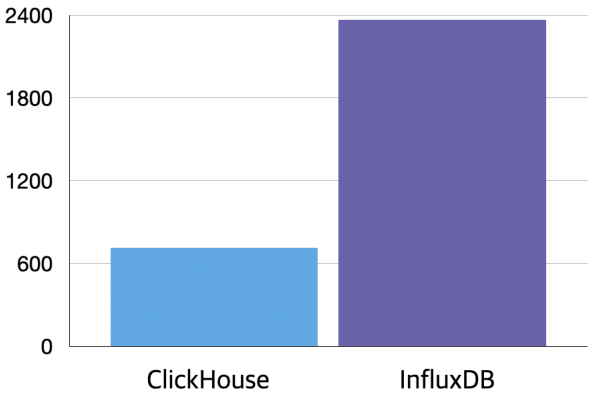

Рис. 4: Время загрузки данных(в секундах)

4.3.2 Степень сжатия данных

Пространство, занимаемое тестовым набором данных в случае ClickHouse, составляет 4.7 Гб, а в случае InfluxDB - 3.1 Гб (Рис. 5). Таким образом, коэффициент сжатия равен 1:12 для ClickHouse и 1:18 для InfluxDB.

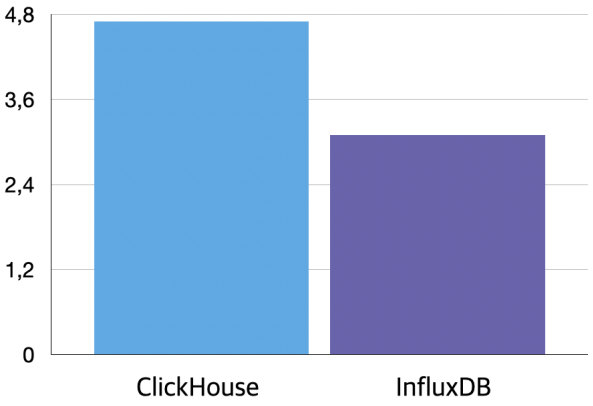

Рис. 5: Степень сжатия данных(в Гигабайтах)

4.3.3 Время выполнения запросов

Для измерения производительности выполнения запросов было использовано 8 типов запросов, приведенных в разделе 3.3.1. Каждый запрос был выполнен 5 раз, после чего для него было вычислено среднее значение. Среднее время ответа на запросы для ClickHouse составляет около 3,5 секунд, а для InfluxDB – около 107 секунд. В среднем, время выполнения запроса в ClickHouse в 77 раз

быстрее, чем в InfluxDB.

Запрос	ClickHouse	InfluxDB
1	0,035	5,733
2	11,901	35,649
3	1,889	277,03
4	7,742	228,447
5	0,061	5,84
6	2,307	237,092
7	0,784	60,623
8	2,629	6,168

Таблица 1: Время выполнения запросов(в секундах)

4.4 Выводы из сравнения

В данном разделе было проведено сравнение по производительности СУБД InfluxDB и ClickHouse по следующим криетриям: время загрузки данных, степень сжатия данных на диске и время выполнения запросов.

По двум из трех критериев ClickHouse показал лучшие результаты. Время загрузки тестового набора в ClickHouse заняло в 3,3 раза меньше времени, чем в InfluxDB и также выполнение тестового набора запросов в среднем вышло в 77 раз быстрее, чем в InfluxDB.

Таким образом, по результатам сравнения для использования в реализации была выбрана СУБД ClickHouse.

5 Описание реализации

5.1 Общая схема работы

На рис. 6 показана упрощенная схема работы разрабатываемого сервиса. Данный сервис включает в себя серверную и клиентскую часть. Для реализации среверной части был выбран язык Python.

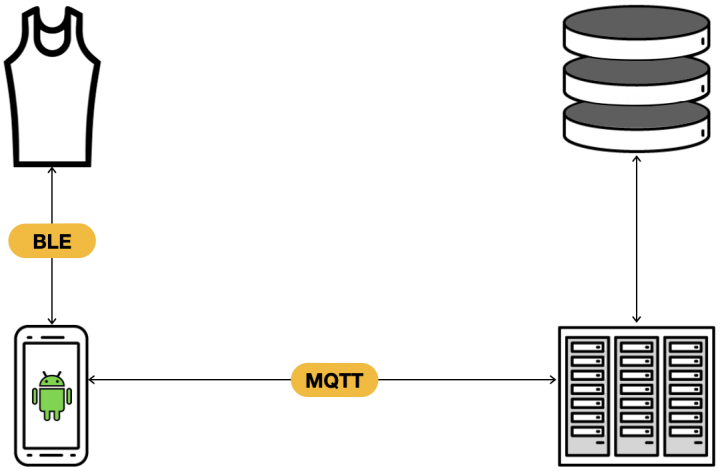

Рис. 6: Общая схема работы

Можно выделить следующие этапы работы:

1. Физиологические показатели пользователя считываются с помощью датчиков умной одежды Hexoskin, после чего отправляются на мобильный телефон на операционной системе Android по протоколу Bluetooth Low Energy (BLE).

2. Мобильный телефон принимает считанные показатели и отправляет их на сервер по протоколу MQTT.

3. Сервер записывает полученные данные в базу данных, откуда их можно взять для медицинской обработки.

5.2 Протокол передачи данных

5.2.1 MQTT

В качестве протокола передачи данных между клиентом и сервером используется протокол MQTT[6].

MQTT — это сетевой протокол, работающий поверх TCP/IP, ориентированный для обмена сообщениями между устройствами по принципу издатель-подписчик. Весь обмен данными происходит через центральный сервер, называемый брокером. Клиент может быть издателем, подписчиком или иметь обе роли сразу. TCP-порт 1883 зарезервирован для протокола MQTT. Если задействуется TLS для защиты связи между клиентом и брокером, то используется порт 8883. Все клиенты должны идентифицировать себя однозначно при подключении к брокеру.

На рисунке 7 изображены два клиента, подключающиеся к брокеру. Один из клиентов является издателем и публикует данные, а другой – подписчиком и подписывается на топик, в который публикует сообщения первый клиент.

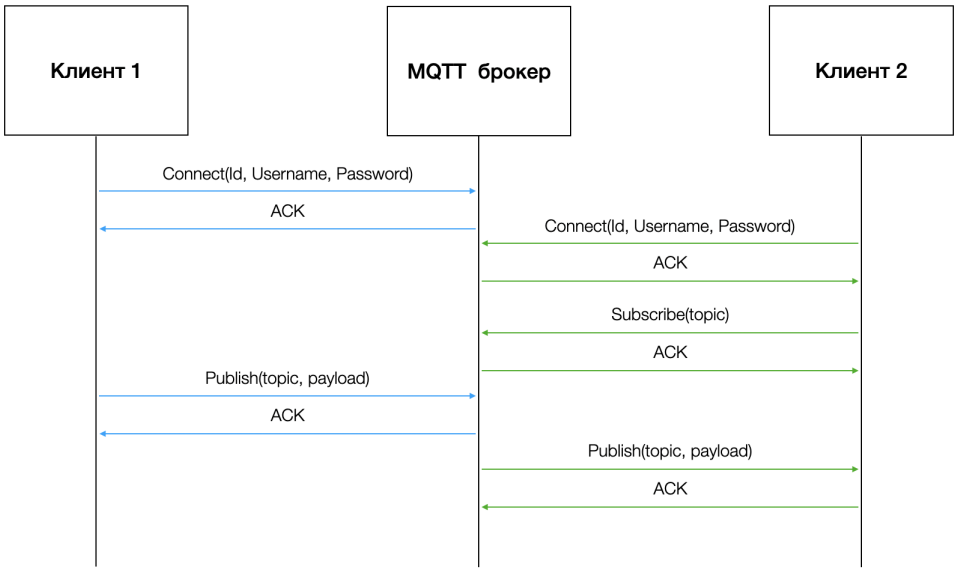

Рис. 7: Пример работы протокола MQTT

5.2.2 Структура топиков

Каждое публикуемое сообщение содержит поле для имени топика, которое идентифицируют публикуемые данные. Аналогичным образом, клиент имеет возможность подписаться на определенные топики, если заинтересован в получении конкретных его данных. Топики имеют иерархическую структуру в формате “дерева”, что упрощает их организацию и доступ к данным.

При передаче данных между клиентом и сервером в разрабатываемом сервисе топики имеют следующую структуру:

/from/id/action

где:

- “from” – часть, которая показывает кто публикует данное сообщение – клиент или сервер. Данный элемент позволяет читать серверу сообщения от клиентов, а клиентам исключительно от сервера.
- “id” - уникальный идентификатор клиента, который выдается каждому клиенту на стороне пользователя при создании клиента MQTT. Данный элемент позволяет клиенту публиковать сообщения в топик, предназначенный для отправки данных только этого клиента, а так же позволяет отправлять сообщения от сервера конкретному клиенту.
- “action” – действие, показывающее цель отправки сообщения. Например, отправка данных или настроек.

Пример: /client/27042020/data – отправка данных на сервер от клиента с идентификационным номером 27042020.

5.2.3 Формат передачи данных

В качестве формата обмена данными между клиентом и сервером был выбран формат JavaScript Object Notation (JSON).

Это известный текстовый формат, полностью независимый от языка реализации, но также он использует соглашения, знакомые программистам C-подобных языков, таких как Java, Python и многих других. Так как при разработке в клинетской и в серверной частях используются различные языки программирования, кроссплатформенность и поддерживаемость различными языками программирования делает JSON удобным форматом передачи данных.

Пример единицы данных в формате JSON, которую сервер получает от клиента:

```
{
    ”Clitime”: ‘2020-04-08 21:07:21’,
    ”Millisec”: 263,
    ”Steps”:45,
    ”RespRate”:2,
    ”Cadence”:229,
    ”Insp”:1914.375,
    ”Exp”: 0.0,
    ”Activity”:5.296875,
    ”HeartRate”:70
}
```

5.3 Аутентификация и авторизация

Протокол MQTT предоставляет поля для имени и пароля пользователя при запросе соединения с брокером. Эти поля используются для аутентификации, как показано на рис. 8.

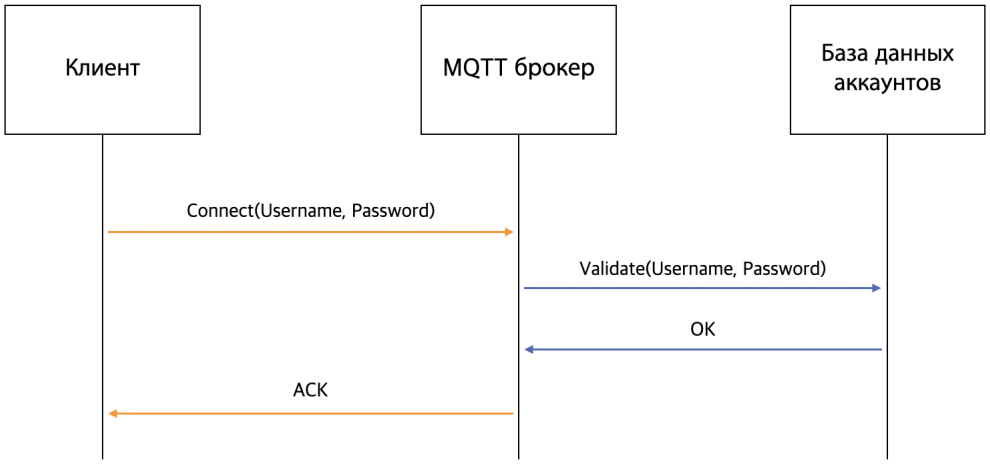

Рис. 8: Процесс аутентификации

При подключении к брокеру клиент предоставляет введенные пользователем логин и пароль в полях сообщения Connect, а брокер сверяет полученные данные с учетными записями, лежащими базе данных акканутов. В данной реализации используется брокер Eclipse Mosquitto. Eclipse Mosquitto — брокер с открытым исходным кодом (лицензии EPL/EDL), который реализует протоколы MQTT различных версий.
https://mosquitto.org

По умолчанию брокер хранит пароли в файле конфигурации. Для хранения учетных записей в базе данных используется плагин mosquitto-auth-plug. https://github.com/jpmens/mosquitto-auth-plug.

Для реализации хранилища аккаунтов была взята NoSQL СУБД MongoDB[7]. MongoDB — документоориентированная система управления базами данных с открытым исходным кодом, не требующая описания схемы таблиц. На рис. 9 приведен пример документа пользователя.

Документ пользователя содержит логин пользователя и пароль, который представлен в виде PBKDF2 хэша, и также встроенный поддокумент, который используется как Access Control List(ACL). ACL ограничивает доступ к топикам, т.е. через ACL предоставляется разрешение на подписку и / или публикацию на определенные топики. Таким образом, провекой ACL реализуется авторизация пользователя.

```
{
    "username": "user1",
    "password": "PBKDF2$sha256$901$8ebTR72Pcmjl3cYq$SCVHHfqn9t6Ev9sE6RMTeF3pawvtGqTu",
    "topics": {
        "/client/%c/#": "w",
        "/server/%c/#": "r"
    }
}
```

Рис. 9: Пример документа пользователя в MongoDB

Расширения %c и %u позволяют использовать в структуре топика уникальный идентификатор клиента и логин пользователя соответственно. На рис. 8 показан пример использования расширения %c: клиенту разрешено публиковать сообщения и подписываться на топики, в которых присутствует его уникальный идентификатор.

5.4 Работа с базой данных

В результате обзора и экспериментального сравнения СУБД в разделах 4 и 5 была выбрана СУБД ClickHouse. Ниже приведен пример создания таблицы в данной СУБД.

```
CREATE TABLE user (
    Clitime DateTime,
    Millisec UInt16,
    HeartRate Nullable(UInt16),
    RespRate Nullable(UInt16),
    Insp Nullable(Float64),
    Exp Nullable(Float64),
    Steps Nullable(UInt32),
    Activity Nullable(Float64),
    Cadence Nullable(UInt32)
)
ENGINE = MergeTree()
PARTITION BY toYYYYMM(Clitime)
ORDER BY (Clitime, Millisec)
```

Каждая таблица соответсвует конкретному пользователю. Столбцы согласованны с параметрами, получаемыми с датчиков Hexoskin, также пристуствуют столбцы для меток времени. ClickHouse не поддерживает тип данных для времени с разрешением более одной секунды, поэтому используется решение хранить время с разрешением до секунд в одном столбце типа DateTime, а миллисекунды в другом.

Для работы с ClickHouse на сервере используются клинетская библиотека clickhouse-driver.
https://github.com/mymarilyn/clickhouse-driver.

6 Заключение

Подводя итог, в рамках настоящей курсовой работы был проведён обзор СУБД временных рядов, целью которого был выбор СУБД для дальнейшего экспериментального сравнения по производительности.

Далее на основании вышеуказанного обзора было проведено сравнение по производительности двух выбранных СУБД, в результате которого была выбрана СУБД для использования в реализации.

Завершающим этапом данной курсовой работы было описание реализации серверной части разрабатываемой открытой платформы для сбора и обработки физиологических данных о человеке.

В качестве возможных направлений для дальнейшей работы можно указать реализацию интерфейса эспорта данных, проработку всех сценариев и механизмов защиты сервиса и реализацию регистрации пользователей.

7 Список литературы

1. DB-Engines [Электронный ресурс]. - режим доступа: https://db-engines.com/en/
2. ClickHouse Crushing Time Series. [Электронный ресурс]. - режим доступа: https://www.altinity.com/blog/clickhouse-for-time-series
3. Comparison of the Open Source OLAP Systems for Big Data: ClickHouse, Druid, and Pinot. [Электронный ресурс]. - режим доступа: https://medium.com/@leventov/comparison-of-the-open-source-olap-systems-for-big-data-clickhouse-druid-and-pinot-8e042a5ed1c7
4. InfluxDB Documentation. [Электронный ресурс]. - режим доступа: https://docs.influxdata.com/influxdb/v1.8/
5. ClickHouse Documentation. [Электронный ресурс]. - режим доступа: https://clickhouse.tech/docs/ru/
6. MQTT Version 3.1.1 documentation. [Электронный ресурс]. - режим доступа: http://mqtt.org/documentation
7. MongoDB Documentation. [Электронный ресурс]. - режим доступа: https://docs.mongodb.com
8. Rui Liu, Jun Yuan, Benchmark Time Series Database with IoTDB-Benchmark for IoT Scenarios , 2019
9. Bader A., Kopp O., Falkenthal O., Survey and Comparison of Open Source Time Series Databases, 2017
10. Naqvi S.N. Yfantidou S., Time Series Databases and InfluxDB, 2017
11. Bader A. Comparison of Time Series Databases, 2016
12. Eclipse Paho MQTT Python. [Электронный ресурс]. - режим доступа: https://github.com/eclipse/paho.mqtt.python