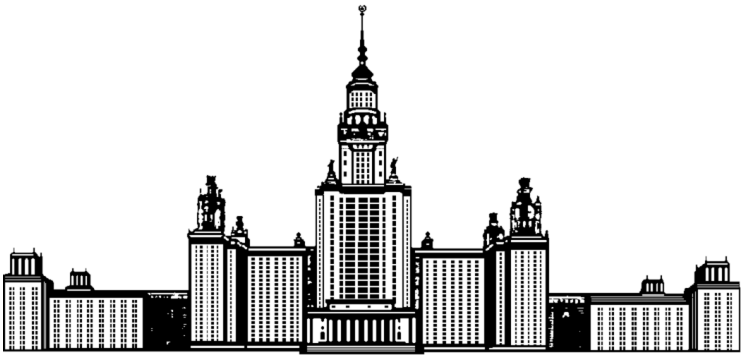

Московский государственный университет им. М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра автоматизации систем вычислительных комплексов

Чайчиц Даниил Александрович

Разработка и реализация клиентской части платформы для сбора и обработки медицинской телеметрии

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:

Бахмуров Анатолий Геннадьевич

Консультант:

Любицкий Александр Андреевич

Москва 2021

Аннотация

В данной дипломной работе решается задача создания клиентской части платформы для сбора и обработки физиологических данных о человеке. Рассматривается приложение, написанное в рамках курсовой работы, исследуются способы экономии энергии и аспекты удобного пользовательского интерфейса. В рамках практической части исследование его потребления энергии в зависимости от режима работы, создается удовлетворяющий стандартам пользовательский интерфейс.

В результате работы получено приложение для обмена данными между медицинским устройством и сервером для обработки телеметрии. Оно обеспечивает стабильную и безопасную передачу личных данных, в нем применены методы экономии энергии, приложение имеет проработанный дизайн. Платформа успешно показала себя в ходе проведенных испытаний.

Содержание

1. Введение

С ростом частоты использования Интернета вещей в разных отраслях, медицинское пространство не стало исключением. На основе связной технологической инфраструктуры операторов беспроводной и проводной связи выстраиваются фрагменты глобального Интернета медицинских вещей (англ. Internet of Medical Things (IoMT)). Появляется всё больше и больше носимых устройств, способных считывать биометрические данные человека. Все эти данные попадают на серверы, а потом – в руки врачей.

Крайне важно получать медицинские показатели не только во время посещения лечебных заведений, но и непрерывно в течение некоторого интервала времени. Эти данные можно использовать для диагностирования заболеваний, которые невозможно выявить при кратком осмотре, прогнозирования состояния человека, составления персональных рекомендаций. Поэтому проблема мониторинга состояния человека является очень важной в наше время. Уже существуют носимые приборы, позволяющие измерять определенные характеристики, но их возможности ограничены лишь небольшим числом параметров, которые можно записать. Кроме того, порой требуется прикреплять к телу достаточно много датчиков, что затрудняет их ежедневное ношение.

В наше время появляется все больше функций слежения за здоровьем в различных умных часах, фитнес браслетах и других привычных для ежедневного ношения устройствах. Но основной проблемой является невозможность получения необработанных данных и закрытость исходного кода. Таким образом, создание открытой платформы сбора и обработки физиологических данных является актуальной задачей на сегодняшний день. Открытость платформы способствует её развитию сообществом, в том числе в форме учебной и исследовательской деятельности.

В данном проекте в качестве умного носимого устройства выступает умная одежда Hexoskin, которая имеет открытый программный интерфейс. Исходный код библиотек для работы с ней находится в общем доступе. В связи с новым, по отношению к привычному формату браслетов и часов, форм-фактору, с помощью этого устройства имеется возможность считывать гораздо больше показателей.

Данная работа посвящена улучшению мобильного приложения для сбора и обработки физиологических данных о человеке, написанного в рамках курсовой работы 3 курса. Проведен обзор способов экономии энергии, создана стратегия улучшения энергопотребления приложения, наиболее подходящая для наших целей. Проведено экспериментальное исследование выбранной стратегии. Проведено исследование свойств функционального и удобного интерфейса.

2. Цель и задачи

2.1. Цель

Требуется реализовать считывание данных с умного жилета Hexoskin по протоколу Bluetooth Low Energy, организовать их хранение на устройстве и отправку на сервер по протоколу MQTT (Рис. 1). Также требуется обеспечить эффективное использование батареи приложением, создать удобный интерфейс.

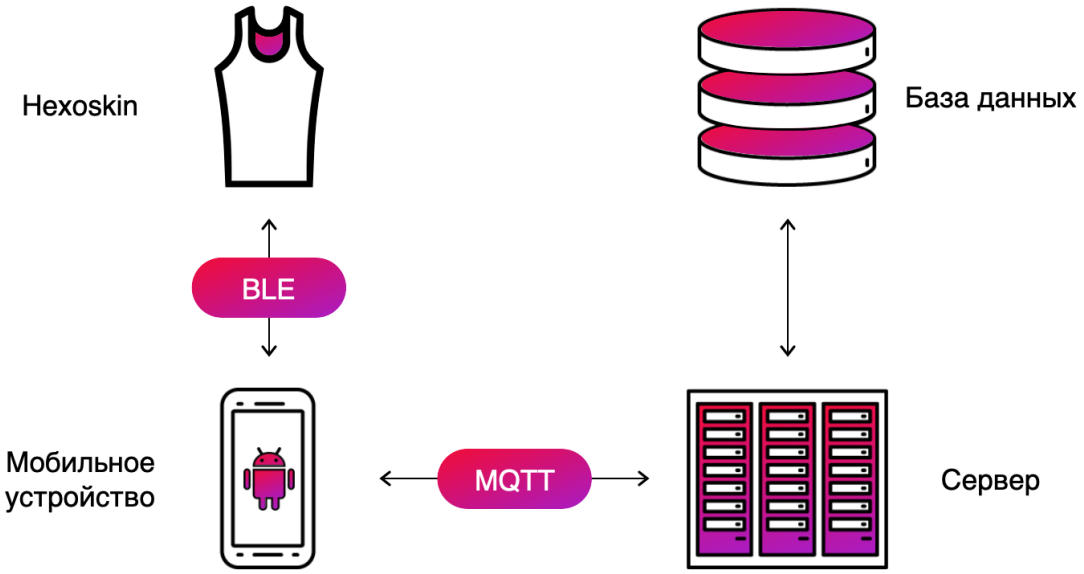

Рис. 1. Схема работы приложения

2.2 Задачи

1. Изучить способы сокращения потребления энергии в ОС Android и их применимость в приложении
2. Реализовать стратегию сокращения энергопотребления
3. Провести экспериментальное исследование выбранной стратегии
4. Изучить качества и свойства качественного интерфейса, создать его макет
5. Реализовать макет интерфейса в приложении для ОС Android

3. Способы сокращения энергопотребления в ОС Android

3.1 Анализ источников повышенного потребления энергии

Согласно [1] основными потребителями энергии являются:

- Экран
- Центральный процессор
- Передача данных по мобильной сети
- GPS позиционирование
- Передача данных по Bluetooth

3.2 Способы повышения эффективности потребления энергии

Рассмотрим способы уменьшить потребление энергии по каждому из пунктов, приведенных выше.

1. Экран. В некоторых современных Android устройствах используются экраны на основе светодиодов, а не ЖК-матриц. В данном типе экранов цвет пикселя определяется суммой цветов палитры RGB трех светодиодов соответствующих цветов. Как известно, белый цвет получается сложением всех трех цветов, а черный – отсутствием света. Самое простое улучшение, которое можно сделать, – перевести цветовую схему приложения в темные тона. В этом случае энергопотребление можно сократить до 53% [2]. Кроме того, в некоторых типах дисплеев энергия, затрачиваемая на подсветку красного, зеленого и синего светодиодов не одинакова. Так, например, в OLED дисплеях синий цвет тратит в среднем на 60% больше энергии, чем красный и зеленый [2]. Соответственно, отказ от цветов, получаемых с помощью синего цвета, также может улучшить потребление энергии. При использовании данной стратегии, получилась бы такая цветовая схема приложения:

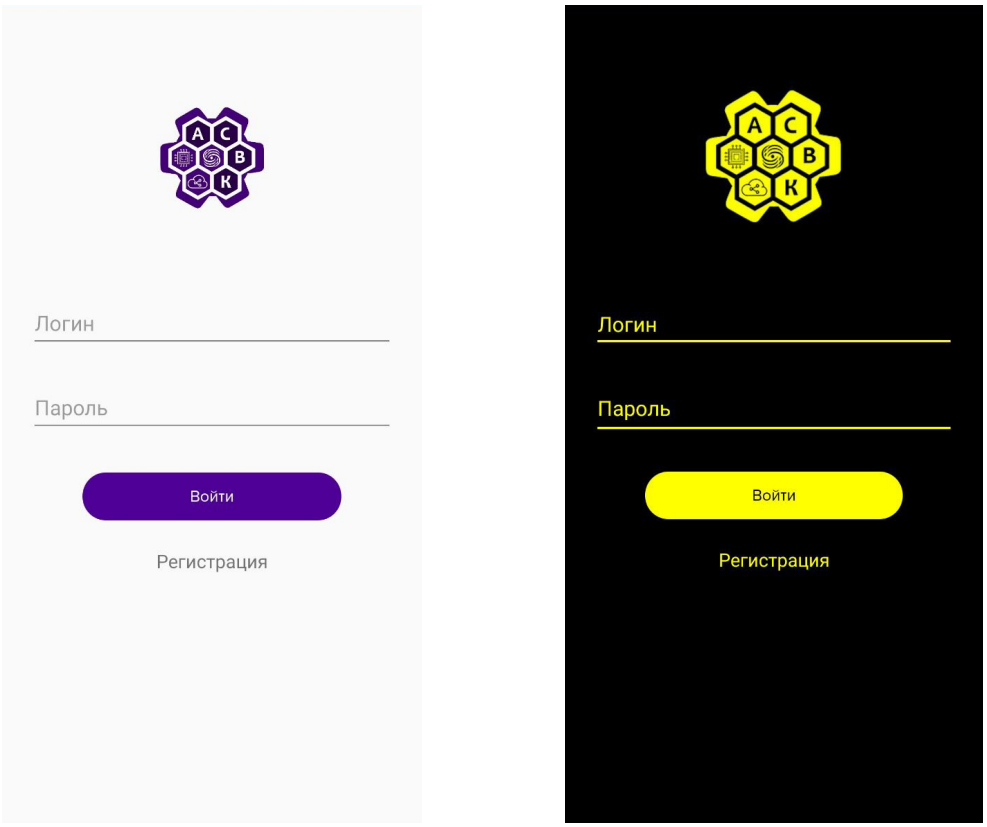

Рис. 2. Обычный и энергоэффективный дизайн

2. Центральный процессор. На нагрузку на центральный процессор напрямую влияет сложность и качество кода программы. Не секрет, что один и тот же фрагмент кода, написанный по-разному, может существенно замедлять или ускорять время выполнения программы. В нашем контексте все такие факторы влияют на расход заряда. В системе существует множество схем оптимизаций, которые не всегда могут сработать, потому что автор программы не сделал соответствующей «подсказки». Вот простой пример [11].

```
static class Foo {
    int splat;
}

Foo[] array = ...

public void zero() {
    int sum = 0;
    for (int i = 0; i < array.length; ++i) {
        sum += array[i].splat;
    }
}

public void one() {
```

```
        int sum = 0;
        Foo[] localArray = array;
        int len = localArray.length;

        for (int i = 0; i < len; ++i) {
            sum += localArray[i].splat;
        }
    }

    public void two() {
        int sum = 0;
        for (Foo a : array) {
            sum += a.splat;
        }
    }
```

В данном фрагменте есть три функции, выполняющие одно и то же действие – суммирование элементов массива.

`zero()` – сама медленная, поскольку на каждой итерации цикла значение размера массива вычисляется заново. Для JIT(just-in-time)-компилятора Java может быть непросто узнать, сколько итераций будет длиться цикл, чтобы его оптимизировать в случае более сложного кода.

`one()` – более быстрая, так как все входные данные преобразуются в локальные, что позволяет избежать поиска объекта, и переменная, отвечающая за длину массива вынесена перед циклом.

`two()`–самая быстрая для устройств, не поддерживающих JIT-компиляцию, и неотличима от `one()` для устройств, поддерживающих её.

Таким образом, можно сказать, что использование цикла, основанного на диапазоне – пример хорошего кода, а вычисление длины массива в объявлении – анти-паттерн. Java – объектно-ориентированный язык, а программы для ОС Android имеют более сложную структуру, чем просто цикл. Даже небольшое приложение представляет из себя сложную систему классов. Поэтому многие примеры плохого кода можно встретить именно в объектном аспекте программ. В [3] выделяются следующие анти-паттерны:

Название	Описание	Способ исправления
Всеобъемлющий класс	Объемный класс, который содержит большую часть функциональности системы с низкой связностью между компонентами	Разбить содержимое на более структурно связанные классы

Ленивый класс	Слишком маленькие классы с очень небольшой функциональностью	Объединить в более крупные классы, не породив при этом всеобъемлющий класс
Излишнее наследование	Наследуемый класс использует только малую часть функциональности родительского класса	Заменить наследование на имплементацию
Ранняя инициализация ресурсов	Инициализация энергозатратных ресурсов, таких как GPS, Bluetooth, Wi-Fi заблаговременно до их использования	Перенести инициализацию максимально близко к моменту использования необходимого ресурса
Использование HashMap	Начиная с API 19 в ОС Android появилась более оптимизированная в плане энергозатрат версия HashMap – ArrayMap. Согласно документации, новая реализация более эффективно распределяет ресурсы, вследствие чего заряд батареи расходуется меньше	Заменить все вхождения HashMap на ArrayMap

Табл. 1. Анти-паттерны, описание и метод решения

3. Передача данных по мобильной сети. При использовании мобильного интернета, помимо затрат на передачу данных, существуют сопутствующие потери энергии на установку и закрытия соединения, так как оно устанавливается только на время использования радиоинтерфейса [4]. Рассмотрим пример соединения (Рис. 3).

После простоя (IDLE) необходимо установить соединение. На данном этапе устройство подключается к базовой станции, энергопотребление растет. Далее идет активное состояние (DCH) – это отправка данных и промежуток из t_1 секунд, когда станция ожидает данные и поддерживает выделенный канал с устройством. Если в течение этого времени устройство не отправляет никаких данных, то базовая станция переключает устройство с выделенного на общий низкоскоростной канал (состояние FACH) и в течение t_2 секунд ожидает данные. В этом состоянии потребление энергии

существенно снижается, но не становится близким к нулевому. Если данных отправлено не было, то соединение разрывается и устройство переходит в режим простоя (IDLE).

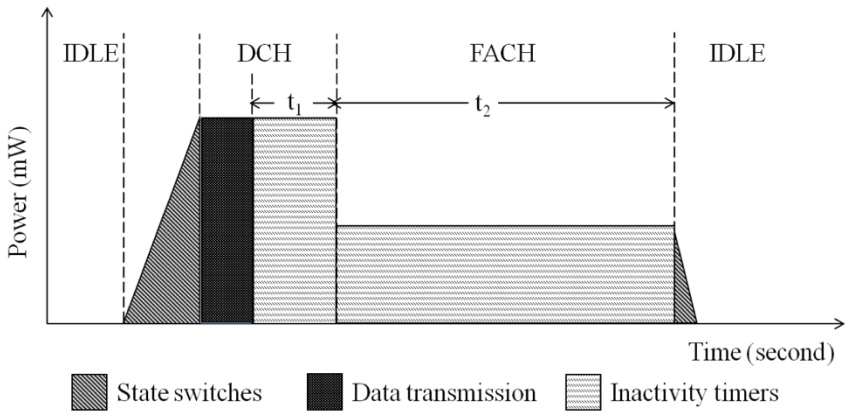

Рис. 3. Потребление энергии при передаче данных

Данная схема показывает работу стандарта 3G. Данный стандарт поддерживают все современные смартфоны, имеющие выход в интернет. Если же используется стандарт 4G (LTE), то данная схема также верна, но с условием, что t_2 становится равным 0, то есть соединение разрывается без перехода в общий канал.

Таким образом, если в программе происходят частые обращения для передачи данных, то затраты на установку/закрытие соединения могут быть значительными (Рис. 4). В первой строке рисунка показан как раз такой случай. Способ решения данной проблемы – передавать данные пакетами, что уменьшит сопутствующие расходы открытия/закрытия соединения. На рисунке это продемонстрировано во второй и третьей строках.

Еще одним способом решения этой проблемы является передача данных по Wi-Fi. При использовании этого способа подключения отсутствуют накладные расходы на подключение и отключения, так как соединение устанавливается на все время использования, кроме того, как правило сигнал Wi-Fi более стабильный и скорость передачи данных выше [7].

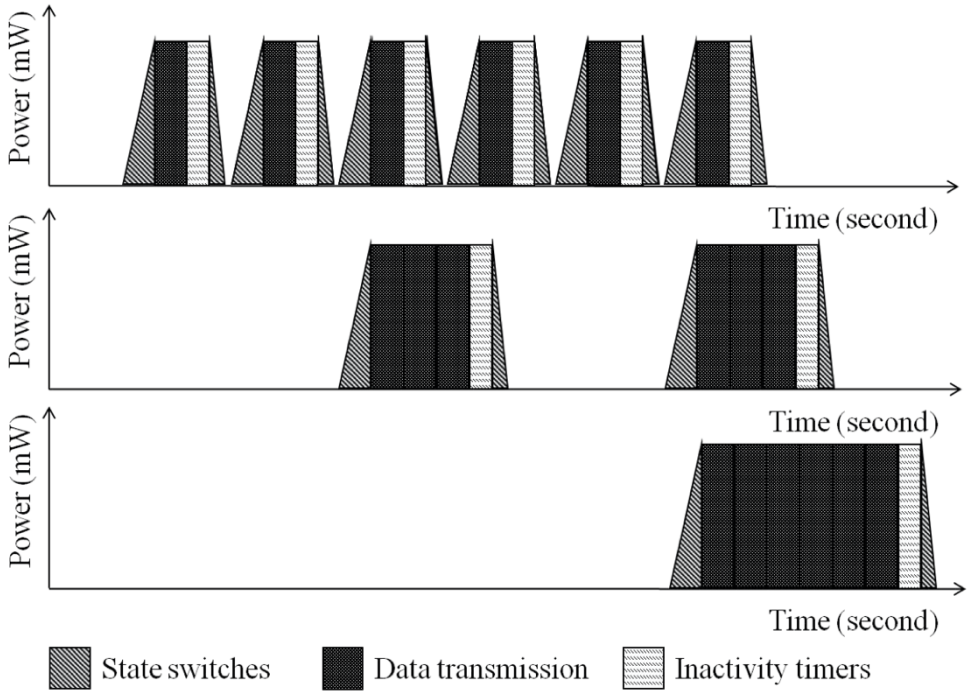

Рис. 4. Потребление энергии при различных стратегиях передачи данных

4. GPS позиционирование. В текущей реализации стандарта Bluetooth Low Energy GPS необходим для поиска других устройств (требования системы). Происходит оно при подключении к устройству, т.е. единожды, при начале работы. В дальнейших действиях, таких как передача данных с носимого устройства на мобильный телефон, с мобильного телефона на сервер, GPS не используется, так что оптимизировать потребление энергии не потребуется.

5. Передача данных по Bluetooth. Потребление энергии классическим протоколом Bluetooth действительно значимо, но в приложении используется его оптимизированная версия – Bluetooth Low Energy. Этот протокол был специально разработан для системы Интернета вещей. Отличием его от обычного Bluetooth является сверхмалое пиковое энергопотребление, среднее энергопотребление и энергопотребление в режиме простоя [11]. Эти параметры являются очень важным критерием для мобильности системы мониторинга показателей человека. Достигается это следующим способом: вещание идет короткими сообщениями с разрывом подключения после передачи информации. Таким образом, большую часть времени устройства «спят», не расходуя энергию на поддержание подключения.

3.3 Выбор стратегии энергосбережения

Для правильного выбора стратегии энергосбережения, нужно проанализировать все вышеперечисленные методы.

1. Экран. Метод позволяет экономить энергию на устройствах с AMOLED/OLED дисплеями. Данный тип дисплеев достаточно распространенный, но не подавляющий на рынке мобильной техники. Чтобы понять, какая часть устройств сможет воспользоваться данным способом, изучим количество моделей с экранами данного типа, предлагаемых к продаже. На сайте Яндекс.Маркет [13] в данный момент представлено 1904 смартфона на ОС Android. Из них матрицы на светодиодах (AMOLED, OLED, Super AMOLED, Super AMOLED Plus) имеют 351 смартфон, что составляет 18.4%. А, например, устройств с матрицей типа IPS представлено 999, что составляет 52.4%. Такое соотношение обусловлено различной ценой матриц разных технологий. Устройства на IPS матрицах стоят дешевле светодиодных аналогов, и чаще используются в устройствах среднего и низкого ценового сегмента, в то время как LED технология присуща премиальным гаджетам.

Данный способ был проверен в действии и было выяснено, что на смартфонах с матрицей IPS типа указанный метод работать не будет. На телефоне со светодиодной матрицей имеет место существенная экономия энергии (более подробно в разделе «Экспериментальное исследование»). Однако при использовании этого способа, дизайн приложения становится слишком отторгающим и резким. Поскольку приложение направлено на использование людьми разных возрастов с разным восприятием, то такое изменение негативно скажется на удобстве использования. К тому же, большую часть времени приложение будет работать в фоновом режиме.

2. Написанное приложение было проверено на наличие анти-паттернов. В коде были обнаружены всеобъемлющие классы, ленивые классы и использование HashMap. Все эти анти-паттерны были исправлены в соответствии с предложенными выше инструкциями.

3. Передавать данные пакетами при использовании мобильного интернета, по возможности передавать данные при подключении к Wi-Fi или при подключенном зарядном. В последнем случае будет выбран основной тип связи, который пользователь выберет в настройках приложения.

3.4 Экспериментальное исследование

Исследуем, насколько повлияют изменения, описанные выше на потребление энергии приложением. Для этого необходимо использовать профилировщик. В исследовании будет использована программа BatteryHistorian [14], которая позволяет посмотреть, сколько процентов потребляет каждая программа.

Для проверки способа сокращения энергопотребления приложения путем изменения цветовой схемы было написано тестовое приложение, которое повторяет начальную страницу основного приложения. Были получены следующие результаты (Рис. 4 – 7):

Устройство: Samsung Galaxy s10e. Тип экрана: Super AMOLED

Ranking	Name	Uid	Battery Percentage Consumed
0	SCREEN	0	0.27%
1	IDLE	0	0.04%
2	ANDROID_SYSTEM	1000	0.03%
3	com.adguard.android	10492	0.02%
4	ROOT	0	0.02%

Рис. 4. Потребление энергии при обычном дизайне

Ranking	Name	Uid	Battery Percentage Consumed
0	SCREEN	0	0.11%
1	IDLE	0	0.04%
2	ANDROID_SYSTEM	1000	0.04%
3	ROOT	0	0.02%
4	BLUETOOTH	0	0.02%

Рис. 5. Потребление энергии при энергосберегающем дизайне

Как видим, экран стал потреблять гораздо меньше энергии в случае эффективного дизайна.

Устройство: TP-link Neffos X1. Тип матрицы: IPS

Ranking	Name	Uid	Battery Percentage Consumed
0	SCREEN	0	0.92%
1	ROOT	0	0.10%
2	GOOGLE_SERVICES	10010	0.09%
3	com.whatsapp	10144	0.08%
4	ANDROID_SYSTEM	1000	0.02%

Рис. 6. Потребление энергии при обычном дизайне

Ranking	Name	Uid	Battery Percentage Consumed
0	SCREEN	0	0.93%
1	ROOT	0	0.07%
2	com.whatsapp	10144	0.05%
3	GOOGLE_SERVICES	10010	0.03%
4	WIFI	0	0.02%

Рис. 7. Потребление энергии при энергосберегающем дизайне

Тестирование проводилось в течение 3 минут. Каждое устройство было настроено на фиксированную яркость экрана. На экране отображался только интерфейс программы. Разницу в показателях между устройствами можно объяснить различными типами экранов и состоянием аккумуляторов.

Теперь перейдем к тестированию различных схем передачи данных. Для начала стоит показать разницу между отправкой данных последовательно и пакетом. В течении пяти минут будем передавать данные с носимого устройства на сервер, как только они будут приходить на смартфон. Получаем следующие результаты (Рис. 8):

Ranking	Name	Uid	Battery Percentage Consumed
0	com.iomt.android	10492	0.31%
1	SCREEN	0	0.19%
2	ANDROID_SYSTEM	1000	0.07%
3	IDLE	0	0.05%
4	ROOT	0	0.04%

Рис. 8. Потребление энергии при потоковой передаче данных

Таким образом, за это время приложение (выделено красным) потребовало энергии больше, чем экран – один из самых энергоемких потребителей.

Теперь в течение пяти минут будем собирать данные, а потом отправим их пакетом. Результаты такой логики работы (Рис. 9):

Ranking	Name	Uid	Battery Percentage Consumed
0	SCREEN	0	0.19%
1	ANDROID_SYSTEM	1000	0.07%
2	IDLE	0	0.05%
3	ROOT	0	0.04%
4	com.iomt.android	10500	0.04%

Рис. 9. Потребление энергии при передаче данных пакетом

Как видим, потребление энергии снизилось в 8 раз. Таким образом, это действительно эффективный способ.

Теперь проверим, сколько энергии потребляет непрерывная и пакетная передача данных через Wi-Fi (Рис. 10, 11).

Ranking	Name	Uid	Battery Percentage Consumed
0	SCREEN	0	0.20%
1	ANDROID_SYSTEM	1000	0.06%
2	IDLE	0	0.05%
3	com.iomt.android	10642	0.05%
4	ROOT	0	0.04%

Рис. 10. Потребление энергии при потоковой передаче данных по Wi-Fi

Ranking	Name	Uid	Battery Percentage Consumed
0	SCREEN	0	0.21%
1	ANDROID_SYSTEM	1000	0.06%
2	IDLE	0	0.05%
3	ROOT	0	0.04%
4	com.iomt.android	10453	0.04%

Рис. 11. Потребление энергии при передаче данных пакетом по Wi-Fi

По результатам экспериментам видно, что разница при передаче данных потоком или пакетом в случае Wi-Fi минимальна, что говорит об отсутствии накладных расходов на запуск передачи.

Если же стоит вопрос, какую сеть предпочтительнее использовать: Wi-Fi или мобильную связь, то ответить сразу будет затруднительно. Все зависит от того, использует ли пользователь Wi-Fi, подключен ли у него тариф с интернетом, или без. Для решения этой проблемы, необходимо реализовать механизм проверки, каким типом подключения есть возможность пользоваться, и в зависимости от результатов логика работы будет меняться по ходу работы программы.

Если рассматривать пользователя, который имеет возможность использовать и мобильную передачу данных, и Wi-Fi, то оптимально будет накапливать данные, когда человек не подключен к интернету или подключен к мобильной сети, и отправлять их на сервер, когда появляется подключение к Wi-Fi или используется зарядное устройство.

4. Пользовательский интерфейс

4.1 Принципы разработки пользовательского интерфейса

Так как разрабатывается прикладная программа, конечным пользователем которой будет обычный человек, желающий следить за своим здоровьем, то необходимо реализовать качественный и понятный интерфейс, который будет способствовать легкому управлению системой. При взаимодействии с графическим интерфейсом пользователь проходит следующие шаги [15]:

1. Формирование цели действий.
2. Определение общей направленности действий.
3. Определение конкретных действий.
4. Выполнение действий.
5. Восприятие нового состояния системы.
6. Интерпретация состояния системы.
7. Оценка результата.

Таким образом, даже перед простым нажатием на элемент интерфейса проходит цепочка шагов, а после выполнения происходит несколько пунктов оценки получившегося действия. Цель хорошего интерфейса – максимально сократить время между каждым шагом. А оно в свою очередь зависит от того, насколько понятно отображается информация, насколько проста в составлении цепочка действий для получения какого-либо результата, насколько удобно расположены элементы для её выполнения.

Для этого нужно исследовать, какие требования предъявляются к современному пользовательскому интерфейсу [6][8]. Сформулируем основные критерии оценки интерфейса пользователем:

- Понятность – насколько легко соотнести элемент управления с действием, которое он реализует.
- Простота иерархии – насколько легко перемещаться по интерфейсу системы, насколько понятны пункты меню и что за ними скрывается.

- Наглядность – насколько легко неопытный пользователь может разобраться с системой.
- Привлекательность – экран не должен быть перегружен элементами, цветовая схема не должна создавать дискомфорт, стиль должен быть единым. В первую очередь должна отображаться важная информация, а далее – вся сопутствующая.
- Использование общепринятых паттернов – во многих приложениях используются схожие элементы управления. Использование таких элементов позволяет «на лету» разобраться с системой, ведь действия уже будут заранее знакомы.
- Предсказуемость – каждый элемент должен выполнять только те действия, которые следуют из его представления.

Хороший интерфейс сам подсказывает пользователю будущие действия, направляет его по меню, отражает все возможности программы и создает уверенность, что все под контролем человека, его использующего. Следует заранее продумать все сценарии использования приложения, обработать все возможные варианты взаимодействия с пользователем.

Кроме того, не стоит забывать, что любой пользователь может не пойти по заранее продуманному сценарию, вместо ожидаемых данных ввести что-то случайное, поэтому необходимо иметь механизмы удержания, которые направят человека в нужное русло и обезопасят код от неожиданных действий.

4.2 Реализация

Определим, какие цели должна выполнять программа с точки зрения конечного пользователя. В программе предусмотрены следующие действия:

- Аутентификация пользователя
- Регистрация пользователя
- Управление личными данными
- Добавление/удаление умных устройств
- Просмотр данных, получаемых с устройства

Для каждого действия нужно придумать красивое и функциональное решение, наглядно отображающее все необходимые данные. Будем создавать дизайн исходя из этих правил.

Для начала необходимо создать карту приложения [9], чтобы понимать, какие элементы будут за что отвечать (Рис. 12.).

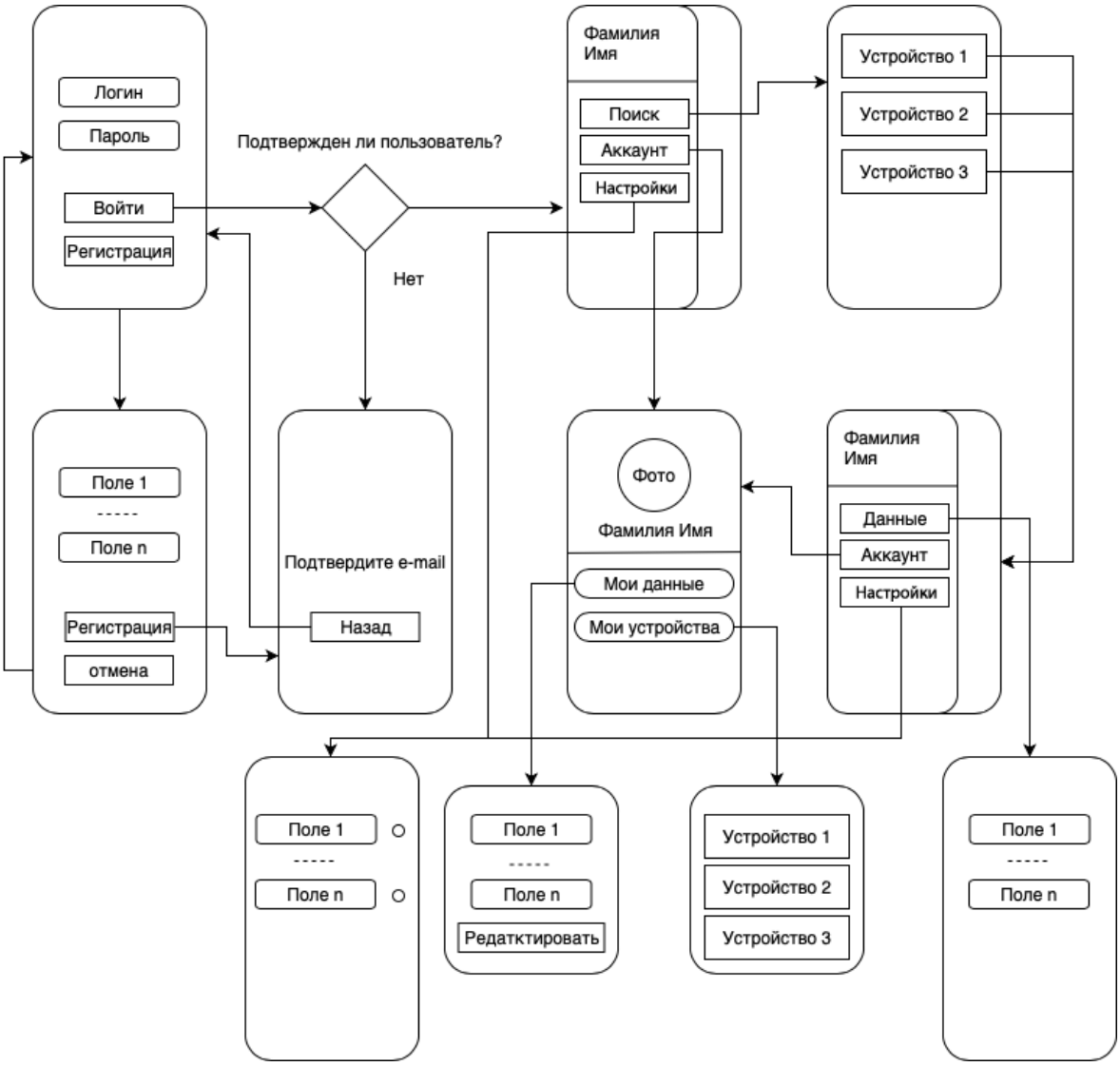

Рис. 12. Схематичная карта экранов приложения

В данной схеме каждая большая фигура представляет из себя экран приложения, прямоугольниками изображены кнопки, прямоугольниками со скругленными углами – поля ввода, малые фигуры – меняющиеся поля одного экрана.

Каждое окно должно отображает необходимый минимум информации, не загружает лишними данными, создает предсказуемый сценарий действий. В

соответствии с правилами, выбрана единая цветовая схема и дизайн приложения. В качестве цветовой схемы, используются бело-фиолетовые тона (основная цветовая схема сайта кафедры). Все схожие элементы дизайна, такие как кнопки, поля ввода, должны быть сделаны в одном стиле. Например, изучая многие современные приложения, можно прийти к выводу, что форму кнопок следует выбирать прямоугольную с полностью закругленными углами:

Рис. 13. Современная кнопка в цветовой схеме приложения

Важно составить простую иерархию окон приложения. Пользователь должен легко понять, как и где можно добавить новое устройство, поменять информацию о себе, в противном случае это грозит получение неактуальных данных. Поэтому, например, все поля, касающиеся пользователя следует сгруппировать на одном экране информации об аккаунте. Кроме того, доступ к данным о себе должен сохраняться на протяжении всего использования программы после прохождения аутентификации. Значит, доступ к аккаунту должен быть предоставлен со всех экранов. Для выполнения этого условия, была использована боковая шторка, присутствующая во всех экранах.

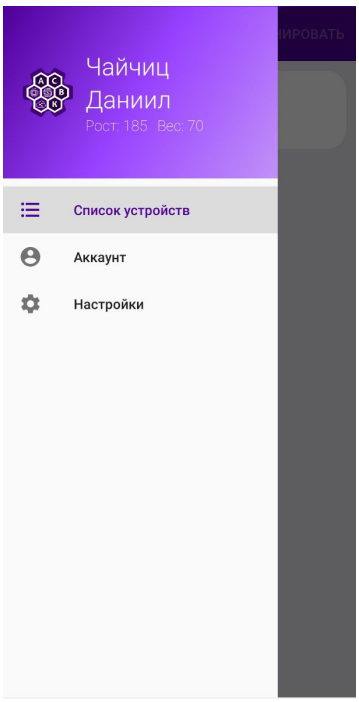

Рис. 14. Боковая шторка в цветовой схеме приложения

Использование боковой шторки для доступа к важным элементам является общепринятой практикой и реализовано во многих приложениях как касающихся темы здоровья (Samsung Health [16]), так и в других повседневных приложениях (ВКонтакте[17], Telegram[18], Яндекс.Go[19]).

Как можно видеть на рисунке выше, рядом с каждой подписью присутствует значок, соответствующий ей по смыслу. Такое дублирование информации позволяет сделать интерфейс более наглядным, позволит «на лету» понять, что ожидает при нажатии на кнопку, какая информация содержится в надписи. Такими значками сопровождается каждое поле, которое содержит важную информацию. Отсутствие их делает восприятие неполным, строка текста выглядит одиноко и незавершенно.

Как было сказано выше, интерфейс должен уметь ограничивать пользователя в действиях. Наиболее велик риск возникновения ошибки в приложении – ввод данных пользователем. Отличным примером такого заградительного механизма является адаптивный ввод данных. Он позволяет предоставить ограничить тип вводимой информации, подсказывая пользователю, что от него ожидается. Таким образом, человеку не нужно задумываться, как именно отформатировать ввод, а программист получает гарантированно правильные данные, что уменьшает вероятность краха приложения. Клавиатура адаптируется под тип вводимой информации, телефон предлагает подсказки. Все это также является неотъемлемой частью удобного интерфейса. Поэтому весь ввод в приложении осуществляется с учетом содержимого.

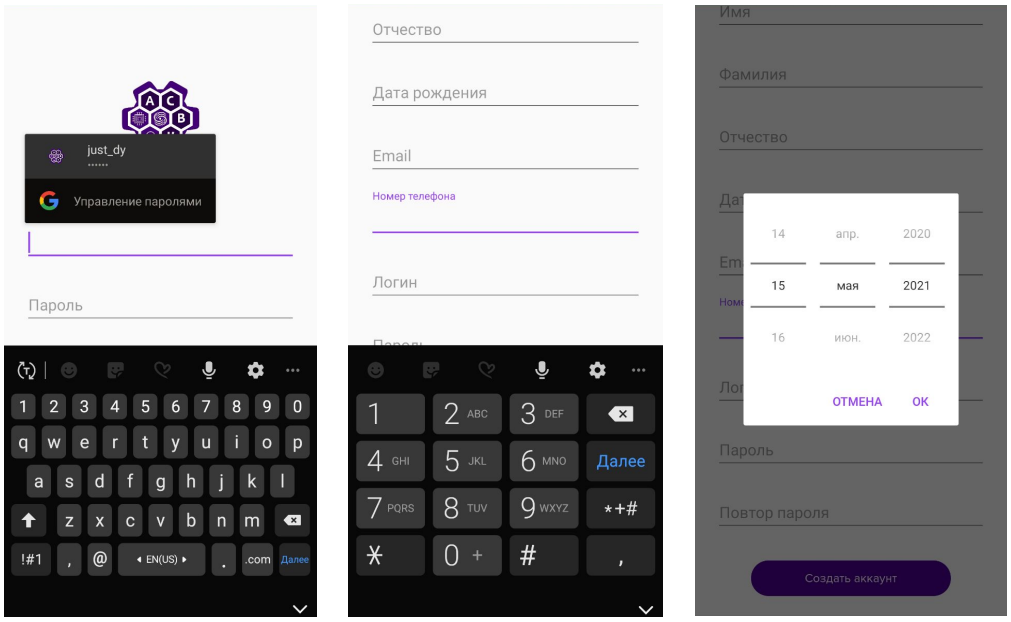

Рис. 15. Умный ввод и подсказки в приложении

4.3 Обзор интерфейса программы

При открытии программы пользователя встречает окно входа в систему (Рис. 16 слева):

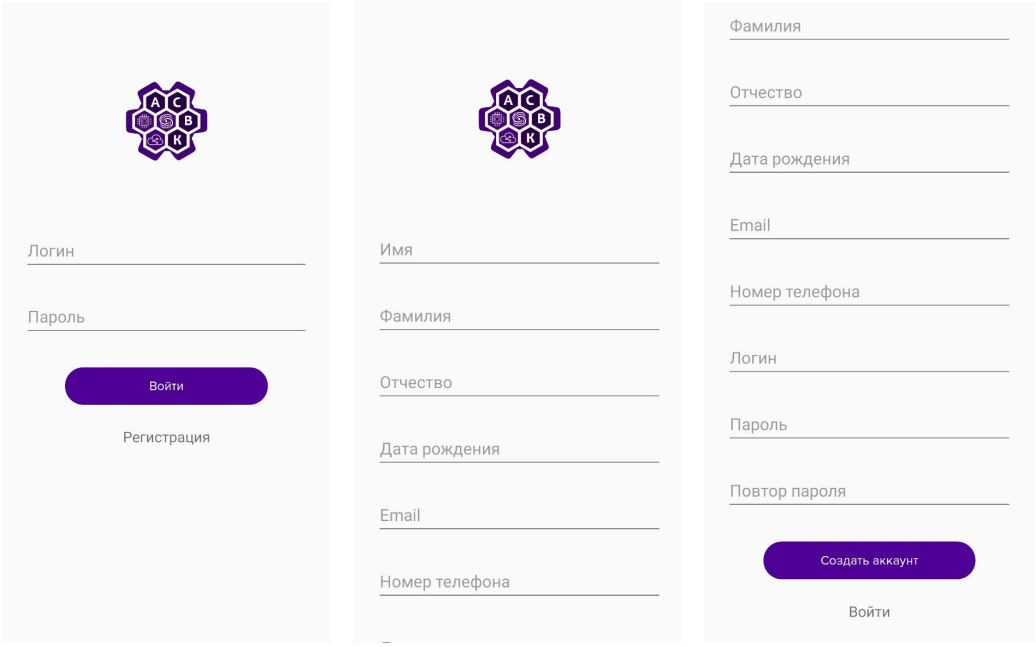

Рис. 16. Окна входа и регистрации

На экране отображены поля для ввода логина и пароля, а также кнопки для входа и регистрации. При нажатии на кнопку регистрации (Рис. 16 по центру и справа), происходит переход в соответствующее окно. В этом окне есть поля для ввода всех необходимых данных с проверкой корректности ввода. Регистрация осуществляется при нажатии на соответствующую кнопку. Также предусмотрена кнопка для возврата к окну входа. После нажатия на кнопку регистрации пользователь переходит на окно, информирующее о необходимости подтвердить свою регистрацию. Сюда же попадает пользователь, если пытается войти, используя данные неподтвержденного аккаунта (Рис. 17).

Если же аккаунт был подтвержден, то после нажатия на кнопку входа пользователь попадает на страницу подключения к устройству (Рис. 18). На данной странице отображаются только те устройства, которые пользователь добавил в свой аккаунт. Если же пользователь этого не сделал, то появляется надпись, уведомляющая о необходимости добавить устройство. Если устройство добавлено и включено, то при нажатии на кнопку «Сканировать», оно отобразится в списке. Слева сверху находится стандартный символ открытия бокового меню (горизонтальные полоски).

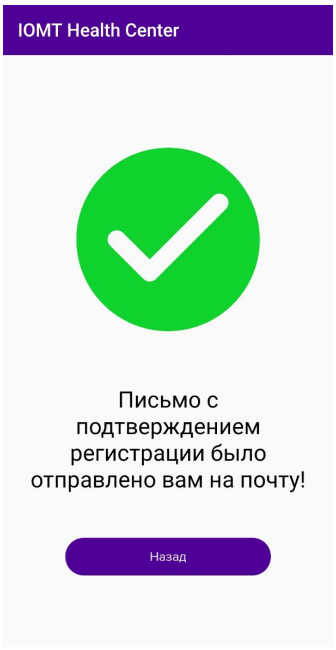

Рис. 17. Окно информирования

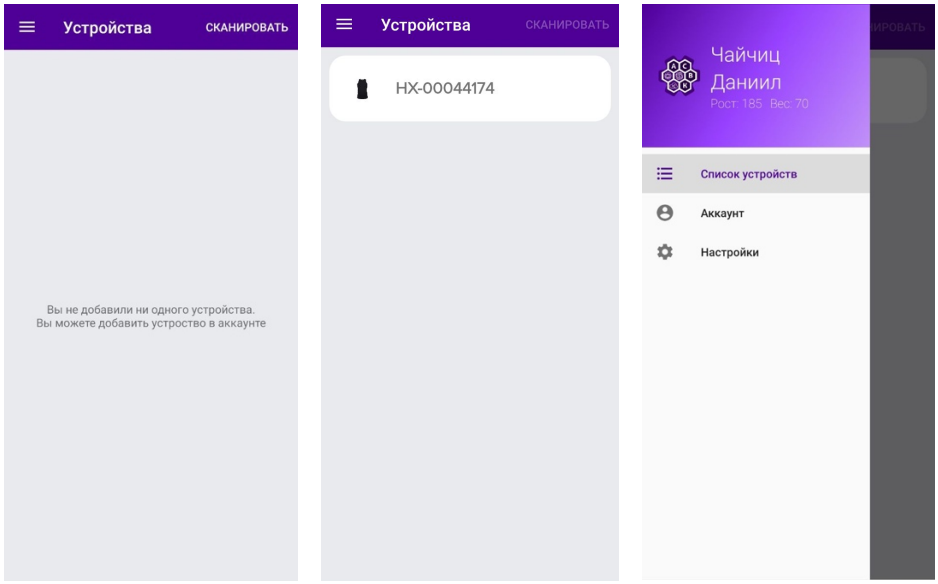

Рис. 18. Окно подключения к устройствам, боковая шторка

Нажатие на отображенное устройство позволяет перейти в режим получения данных с умного устройства (Рис. 19). На данном экране также присутствует вызов шторки меню.

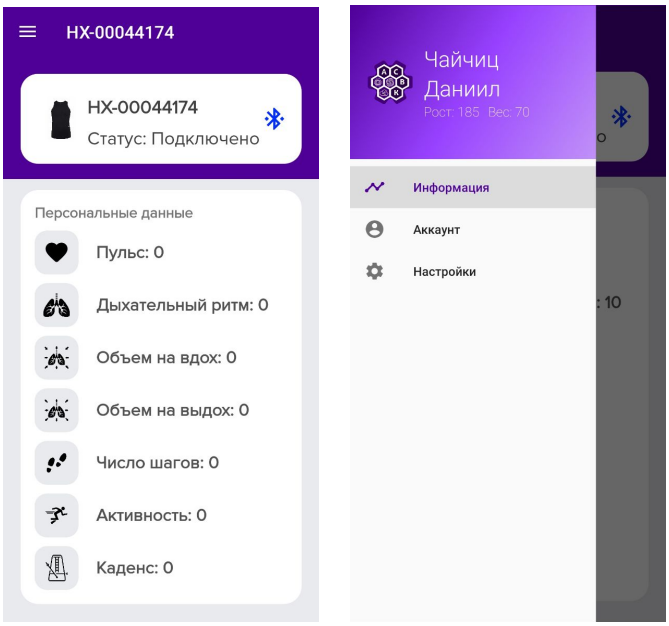

Рис. 19. Окно получения данных с умного устройства, боковая шторка

При переходе в аккаунт, пользователь попадает в следующее окно:

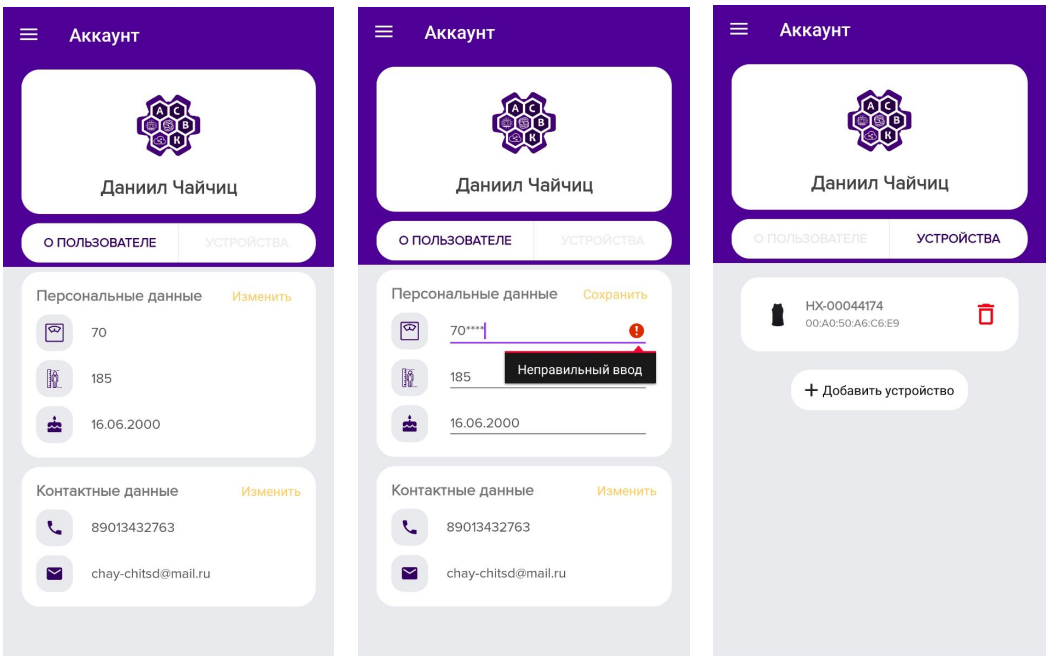

Рис. 20. Окно личного кабинета пользователя

Окно делится на 2 части: верхняя отображает имя и фамилию пользователя, фотографию, а также меню перехода от личных к данным к избранным устройствам. Ярким цветом выделены кнопки изменение информации о человеке. Личная информация разделена на 2 группы – персональные данные и контактные данные. Также на рисунке можно наблюдать как работает проверка корректности ввода. В разделе устройства отображается список добавленных устройств вне зависимости от

того, включены они или нет. Устройство можно добавить или удалить при нажатии на соответствующую кнопку.

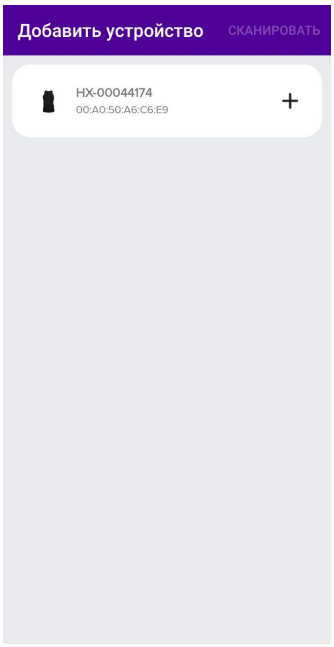

Рис. 21. Окно добавления устройства

И последнее окно, с которым может взаимодействовать пользователь – окно настроек:

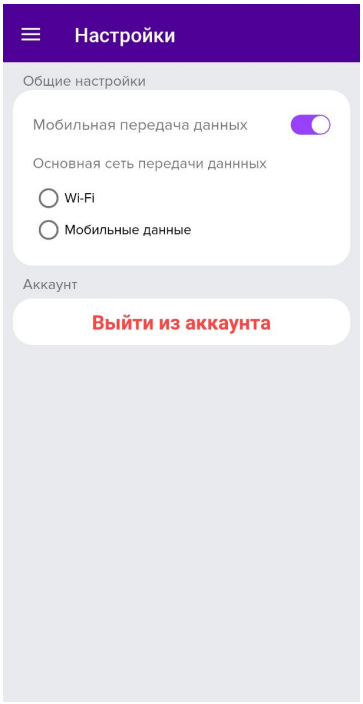

Рис. 22. Окно настроек

В данном окне пользователь может выбрать предпочтения к использованию сети Интернет, а также выйти из аккаунта.

5. Взаимодействие с сервером

5.1 Общее описание

Важной частью системы Интернета вещей является взаимодействие с серверной частью платформы. Общение с сервером можно разделить на 2 категории:

1. Обмен данными о пользователе – аутентификация, регистрация, получении данных о пользователе. В данных сценариях важно двусторонние взаимодействие, так как клиент посылает запрос с определенными параметрами и ожидает получить ответ именно в зависимости от них.

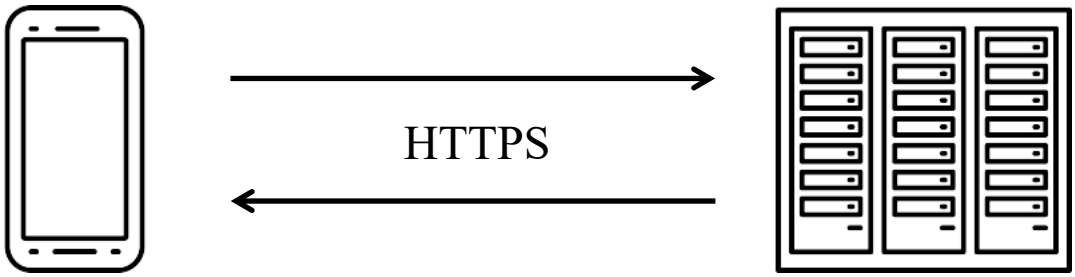

Рис. 23. Схема с HTTPS

2. Обмен телеметрией. В случае отправки данных, полученных с умного устройства, используется протокол MQTT. В этой ситуации приложение просто отправляет данные, не нуждаясь в ответе от сервера.

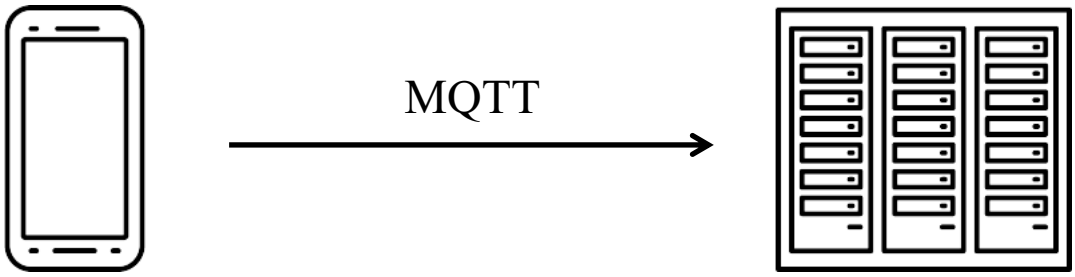

Рис. 24. Схема с MQTT

5.2 Безопасность

Имея дело с личными данными пользователей, необходимо поддерживать должный уровень безопасности. Как видно из обзора интерфейса, существует механизм идентификации и аутентификации пользователей. Для гарантии того, что

на протяжении всего сеанса общения сервер общается с действующим клиентом, используется система токенов JWT (JSON Web Token). Это открытый стандарт (RFC 7519) для создания токенов доступа, основанный на формате JSON. Как правило, используется для передачи данных для аутентификации в клиент-серверных приложениях. Токены создаются сервером, подписываются секретным ключом и передаются клиенту, который в дальнейшем использует данный токен для подтверждения своей личности. Токен JWT состоит из трех частей: заголовка (header), полезной нагрузки (payload) и подписи или данных шифрования. Первые два элемента — это JSON объекты определенной структуры. Третий элемент вычисляется на основании первых двух и зависит от выбранного алгоритма шифрования токена (в случае использования неподписанного JWT может быть опущен) [10].

При входе в систему пользователь вводит логин и пароль, в обмен на которые сервер отправляет ему токен (если данные были введены корректно) и уникальный идентификатор. Теперь при общении с сервером клиент использует эти данные, что позволяет не сохранять логин и пароль. При любом HTTP запросе необходимо прикреплять в качестве параметров эти поля. При общении по MQTT данные шифруются с помощью SSL, для доступа к брокеру необходим пароль, являющийся указанным ранее токеном.

Преимущества системы токенов в сравнении со связкой логин/пароль – сложность токена и ограниченность срока его действия. В отличие от пароля, который человек как правило выбирает не самым надежным способом, токен – длинный и случайный для внешнего человека набор символов, срок взлома которого больше, чем срок жизни самого токена.

6. Апробация платформы

Платформу необходимо было протестировать в реальных условиях. Для этого была промоделирована обычная ситуация тренировки. Испытуемый выполнял комплекс упражнений, находясь в умном жилете Hexoskin, данные с него отправлялись на смартфон, а далее на сервер.

Рис. 25. Сверху вниз: пульс, каденс, объем выдыхаемого воздуха

В результате испытаний был получен набор графиков. Часть из них показана на (Рис. 25), с полным набором графиков можно ознакомиться в Приложении. Эксперимент проводился в следующем режиме: состояние покоя 3 минуты, спокойный шаг 3 минуты, бег 3 минуты, спокойный шаг, покой. Из полученный данных отлично видно, как менялись показания в зависимости от рода занятия.

Покой. Пульс имеет некоторые выбросы, но в целом достаточно низкий, каденс (число шагов в минуту) равен 0, дыхание спокойное: средние выдохи в вдохи (есть выброс в начале – ошибка Hexoskin’a).

Шаг. Возрастает пульс, каденс становится ненулевым, дыхание более активное, объем выдыхаемого воздуха достаточно большой: активность не имеет сильной нагрузки.

Бег. Заметно возрастает пульс, каденс, дыхание становится учащенным, что соответствует бегу. С течением времени видно, как понижается объем выдыхаемого воздуха, что соответствует накоплению усталости.

Шаг. Пульс плавно снижается, каденс резко становится на порядок ниже, начинает расти объем выдыхаемого воздуха.

Покой. Пульс продолжает уменьшаться, каденс становится равным 0, дыхание выравнивается.

Шаг. Снова возрастает пульс, каденс поднимается, частота дыхания увеличивается.

7. Заключение

В рамках дипломной работы была поставлена цель улучшить клиентскую часть платформы для сбора и обработки физиологических данных о человеке, написанную во время курсовой работы.

Было произведено исследование в области экономии энергии, составлен и выполнен план действий, ведущих к сокращению потребления энергии. Экспериментально была подтверждена целесообразность этих действий. Были проанализированы аспекты создания понятного и информативного интерфейса, создан и реализован макет дизайна приложения.

Возможное продолжение действий в этой области – расширение типов поддерживаемых устройств, создание механизма рекомендаций пользователю, создание версии приложения не только для ОС Android.

8. Список литературы

1. Li X. et al. Measurement and analysis of energy consumption on Android smartphones //2014 4th IEEE International Conference on Information Science and Technology. – IEEE, 2014. – С. 242-245.
2. Linares-Vásquez M. et al. Optimizing energy consumption of guis in android apps: A multi-objective approach //Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering. – 2015. – С. 143-154.
3. Morales R. et al. Anti-patterns and the energy efficiency of Android applications //arXiv preprint arXiv:1610.05711. – 2016.
4. Deng S., Balakrishnan H. Traffic-aware techniques to reduce 3G/LTE wireless energy consumption //Proceedings of the 8th international conference on Emerging networking experiments and technologies. – 2012. – С. 181-192.
5. Heydon R., Hunn N. Bluetooth low energy //CSR Presentation, Bluetooth SIG https://www. bluetooth. org/DocMan/handlers/DownloadDoc. ashx. – 2012.
6. Ahmad Z., Ibrahim R. Mobile commerce (M-commerce) interface design: a review of literature //OSR Journal of Computer Engineering (IOSR-JCE) e-ISSN. – 2017. – С. 2278-0661.
7. Balasubramanian N., Balasubramanian A., Venkataramani A. Energy consumption in mobile phones: a measurement study and implications for network applications //Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement. – 2009. – С. 280-293.
8. Этапы разработки пользовательского интерфейса: https://vc.ru/design/58502-etapy-razrabotki-polzovatelskogo-interfeysa-kak-sdelat-tak-chtoby-ui-ne-lishil-vas-pribyli
9. Энергопотребление Android-приложений: https://habr.com/ru/company/appodeal/blog/260095/
10. JSON Web Token: https://ru.wikipedia.org/wiki/JSON_Web_Token
11. Android Developers Community: https://developer.android.com/
12. Hexoskin SDK: https://bitbucket.org/carre/hexoskin-smart-demo/src/master/hexoskin-smart-android/
13. Модели смартфонов на рынке: https://market.yandex.ru/catalog--mobilnye-telefony/54726/filters?text=смартфоны&cpa=0&hid=91491&glfilter=16816515%3A16816517&onstock=0&local-offers-first=0
14. Battary Historian: https://github.com/google/battery-historian

15. Норман Д. Дизайн привычных вещей. – Litres, 2020.
16. Приложение Samsung Health: https://play.google.com/store/apps/details?id=com.sec.android.app.shealth&hl=ru&gl=US
17. Приложение Вконтакте: https://play.google.com/store/apps/details?id=com.vkontakte.android&hl=ru&gl=US
18. Приложение Telegram: https://play.google.com/store/apps/details?id=org.telegram.messenger&hl=ru&gl=US
19. Приложение Яндекс.Go: https://play.google.com/store/apps/details?id=ru.yandex.taxi&hl=ru&gl=US

9. Приложение

В данном приложении приведены все графики, полученные при испытании платформы.

Рис. 26. Сверху вниз: активность, пульс, дыхательный ритм

Рис. 27. Сверху вниз: каденс, объем выдыхаемого воздуха, объем вдыхаемого воздуха