**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГООБРАЗОВАНИЯ РФ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«БАЛТИЙСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ им. И. КАНТА»**

**ИНСТИТУТ ФИЗИКО-МАТЕМАТИЧЕСКИХ НАУК**

**И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

|  |  |
| --- | --- |
| Рекомендована к защите:  методический руководитель  направления подготовки  верховный владыка ИФМНиИТ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.О. Синюхин  "\_\_\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. | Допущена к защите:  первый заместитель директора  ИФМНиИТ  к. ф.-м. н., доцент  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.А. Шпилевой  "\_\_\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**Тема: «Разработка мобильного приложения для медицинского устройства»**

**Направление подготовки: 01.03.02 «Прикладная математика и Информатика»**

Квалификация (степень**)**: **бакалавр**

ВКР защищена на оценку: **Выполнил:** студент 4 курса

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_С.А.Зорко

**Руководитель:** верховный владыка ИФМНиИТ

\_\_\_\_\_\_\_\_\_ А.О. Синюхин

Калининград, 2020

Введение

Электронное здравоохранение (e-health англ.) - это использование информационных и коммуникационных технологий (ИКТ) для здравоохранения. За последние 10 лет рынок e-health устройств значительно вырос. Ученые по всему миру совершают тысячи открытий в сфере медицины и компьютерных технологий, которые позволяют реализовать все более функциональные устройства и методы лечения, диагностики пациентов.

Электронное здравоохранение включает в себя такие компоненты как:

* Мобильные медицинские системы (mHealth)
* Информационные системы здравоохранения (HIS)
* Телемедицина
* Обучение на расстоянии (eLearning)

Мобильные медицинские системы – это системы использующие мобильные устройства в медицине и общественном здравоохранении. Этот термин чаще всего используется в отношении использования устройств мобильной связи, таких как мобильные телефоны, планшетные компьютеры и персональные цифровые помощники (КПК), и носимые устройства, такие как умные часы, для упрощения или улучшения качества работы служб здравоохранения, получения информации о состоянии пациентов и сбора данных. Варианты использования мобильных медицинских систем включают использование мобильных устройств для сбора данных о состоянии здоровья населения или пациентов отдельно взятого медицинского учреждения, сбор медицинской информации практикующим врачам, исследователям и пациентам, мониторинг показателей жизнедеятельности пациента в режиме реального времени, прямое предоставление медицинской помощи (с помощью мобильной телемедицины), а также обучение и взаимодействие работников здравоохранения.

В данной выпускной квалификационной работе проводится разработка мобильного приложения для mHealth устройства - Миостимулятора. Само устройство помогает пациентам страдающим болью или неприятными ощущениями в области спины улучшить свое самочувствие. Принцип работы миостимулятора основан на воздействии на мышцы тела с помощью электрических импульсов. К телу прикладываются электроды в непосредственной близости к стимулируемым мышцам. Посылаемые от устройства электрические импульсы похожи на импульсы нервной системы, которые заставляют мышцы сокращаться.

**Актуальность работы** обусловлена необходимостью в мобильном приложении для контроля работы миостимулятора которое будет задавать программу работы, а также собирать различные метрики с устройства.

**Целью** данной выпускной квалификационной работы (ВКР) является

реализация и тестирование нового мобильное приложение на базе имеющегося, которое будет использовать все возможности устройства, работать без сбоев, а также иметь удобный пользовательский интерфейс.

**Задачи ВКР:**

1. Выбрать платформу для разработки
2. Разработать пользовательский интерфейс
3. Расширить функционал модуля взаимодействия с устройством
4. Наладить бесперебойную работу Bluetooth соединения

**Структура работы.** Выпускная квалификационная работа по теме «Разработка автоматизированной информационной системы учета движения товаров ООО «Европласт» состоит из введения, трех глав, заключения, списка использованной литературы, перечня сокращений и приложений.

Во введении раскрывается актуальность темы, а также цель и задачи.

В первой главе рассматриваются общие вопросы касающиеся взаимодействия с устройством. {…} В заключении главы формулируются выводы.

Во второй главе …

В третьей главе …

В заключении подводятся итоги работы, формулируются окончательные выводы.

В приложении …

**Содержание работы.** Полный объем работы составляет X страниц машинописного текста. Всего работа содержит X рисунков и Y таблиц.

**Глава 1**

**Взаимодействие с устройством**

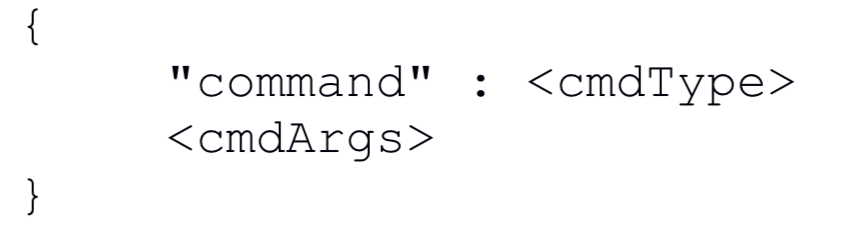
Прикладной уровень стека протоколов взаимодействия устройства с мобильным приложением основан на формате данных JSON. Обмен данными осуществляется по принципу "запрос-ответ", минимальной единицей данных является JSON-документ. Мобильное приложение посылает *документ*, описывающий операцию (команду), которую должно выполнить устройство; получив и обработав её, устройство посылает ответное сообщение, содержащее код выполнения и (для некоторых команд) полезные данные.

Устройство поддерживает следующие операции:

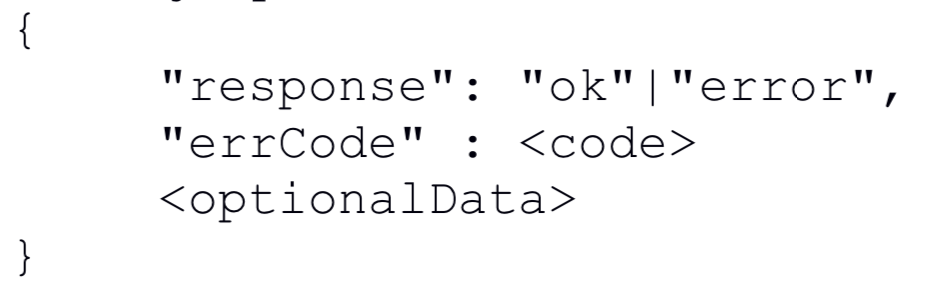
* Синхронизация (установка) времени на устройстве;
* Программирование плана лечения (по расписанию или в ручном режиме);
* Запуск и остановка сеанса лечения вручную;
* Запрос данных о батарее питания (напряжение);

Если операция по той или иной причине не выполнена, возвращается код ошибки, указывающий на причину её возникновения.

Общий вид запроса, посылаемого мобильным приложением:



Ответ устройства:

******

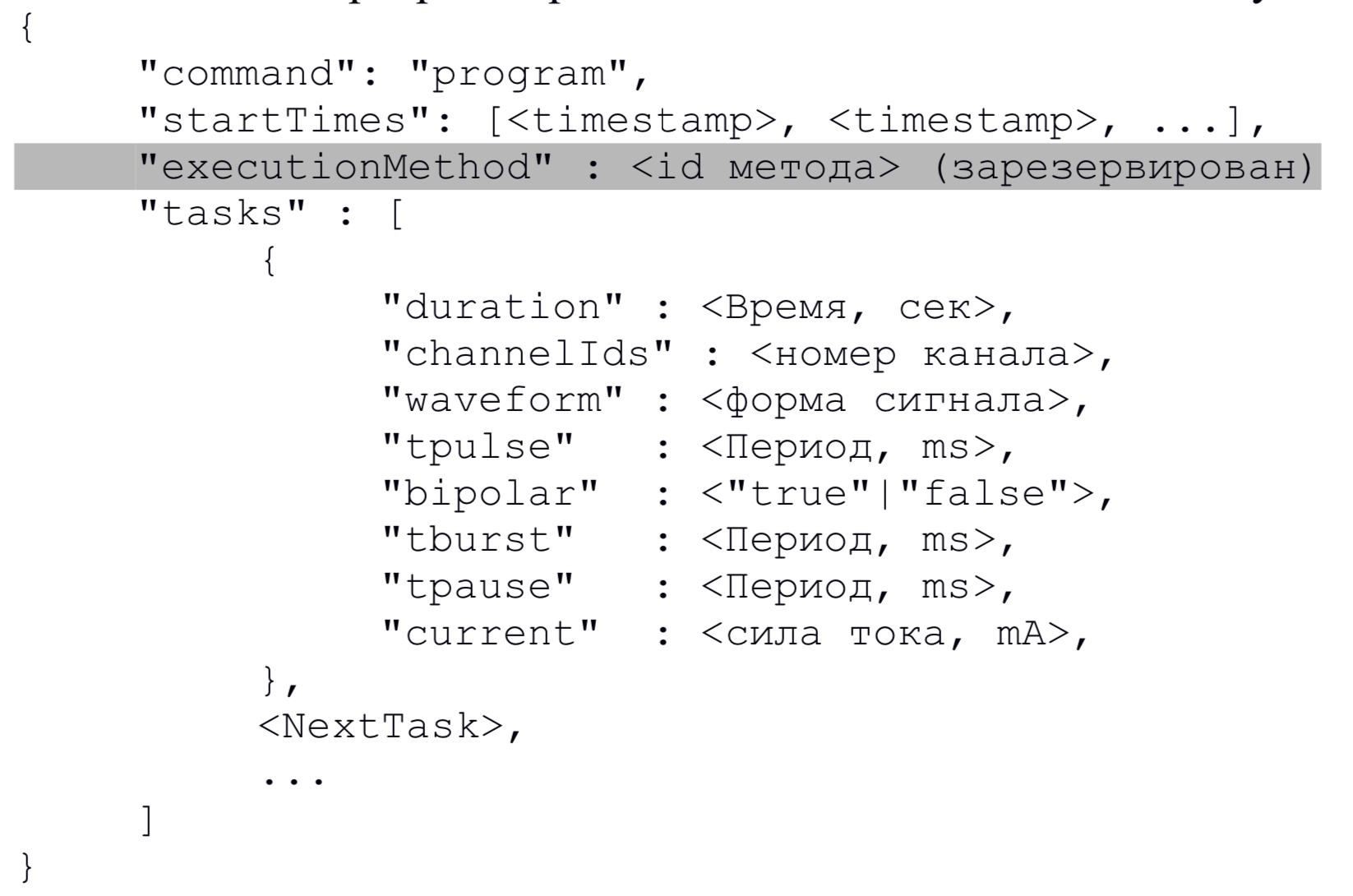
***Перечень кодов ошибок:***

0 – отсутствие ошибок (*зарезервировано*);  
1 – ошибка синтаксиса JSON;  
2 – неверная или неподдерживаемая команда; возникает, когда значение поля "command" не соотвествует ни одной из команд, известных устройству;  
3 – ошибка аргумента команды; возникает, когда пропущен аргумент, являющийся обязательным для команды, а также в случае неверного типа значения аргумента;  
4 – невалидное значение даты или времени;  
5 – невалидный план лечения; возникает, когда по крайней мере один из сеансов плана лечения имеет параметр, выходящий из диапазона допустимых значений (см. п. "Программирование плана лечения");  
6 – таймаут в процессе передачи данных; возникает в случае, если устройство начало принимать команду по Bluetooth, но время ожидания её окончания было превышено (с момента последнего получения данных прошло более 2*сек*)  
7 – 9 – (*зарезервированы*)  
10 – ошибка записи на SD/MMC-карту (*flash*-память заблокирована, что происходит сразу после обновления прошивки; необходимо перезагрузить устройство по питанию)

**Программирования плана лечения**

*План лечения* (называемый также *программой лечения*) представляет собой список *сеансов* ("tasks"), в каждом из которых описываются все необходимые параметры: время начала электромиостимуляции, сила тока, форма импульсов, частота импульса, длительность серии импульсов и длительность паузы между ними, двуполярный или однополярный сигнал. Максимальное количество сеансов в плане лечения равно 40, при этом план лечения может иметь до 10 меток времени, т.е. быть за∞пущен до 10 раз в сутки.

Команда программирования плана лечения имеет следующий формат:

*** Параметры команды:***

• *startTimes* – массив меток времени запуска сеанса миостимуляции. Если массив пуст, то план лечения может быть запущен только вручную. Формат времени "*hh:mm:ss*". Например "10:05:00" – 10 часов 5 минут ровно;

• *tasks* – массив сеансов миостимуляции, каждый сеанс – вложенный документ с JSON-полями.

***Параметры сеанса:***

* *duration* – длительность сеанса в секундах, целое число. ***Валидные значения:*** [0 – 86399]
* *waveform* – идентификатор формы сигнала, целое число из списка: ◦ 0 – *зарезервировано*◦ 1 – нет импульсов  
  ◦ 2 – прямоугольный (меандр)

◦ 3 – трапеция ◦ 4 – синусоида

◦ 5 – пилообразный

Остальные значения невалидны.

* •  *tpulse* – длительность единичного импульса заданной формы, в

милисекундах, с фиксированной точкой, поддерживается до 3х знаков после точки (разрешение до 1us).  
***Валидные значения:*** [0.1 – 100.0]

* •  *bipolar* – "true", если сигнал должен меняет полярность, "false" – в противном случае; текущая версия использует строки;
* •  *tburst* – длительность следования серии импульсов в милисекундах; целое положительное число;
* •  *tpause* – длительность перерывов между сериями импульсов в миллисекундах; целое неотрицательное число (ноль соотвествует непрерывному режиму, без пауз)
* •  *current* – значение тока в *mA*, установленное программой; число с фиксированной точкой, имеющее до 3х знаков после точки (остальные игнорируются).  
  ***Валидные значения:*** [0.0 – 10.0]
* •  *channelIds –* номер выходного канала; целое неотрицательное число. ***Валидные значения:*** 0, 1, 2; значение 0 задаёт сеанс с паузой на обоих

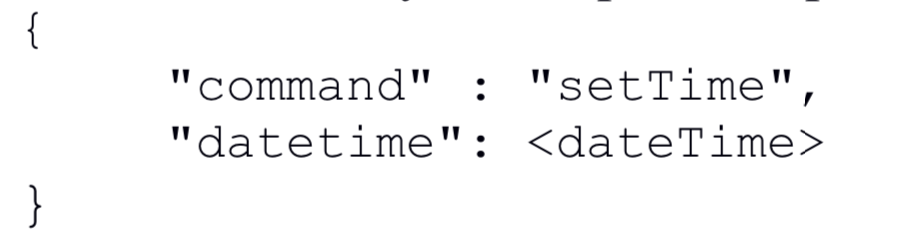
каналах одновременно.

*Примечание:* для того, чтобы задать длительную паузу между сеансами, необходимо создать сеанс с нулевым или отсутствующим параметром *channelIds* и длительностью *duration* равной длительности этой паузы в секундах.

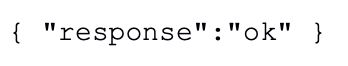
**Синхонизация времени на устройстве**

Синхронизация времени необходима для того, чтобы устройство

включалось в нужное время по расписанию. Команда имеет формат:



Параметр <datetime> должен иметь формат ISO8601, без учёта timezone: "YYYY-MM-DDThh:mm:ss", где YYYY – год, MM – месяц (1 – январь, 12 – декабрь), DD – день, 'T' – разделитель даты и времени (чувствителен к регистру). В случае невалидного значения даты-времени, устройство ответит ошибкой с кодом 4; иначе дата и время будет установлено и вернётся

****

**Ручной запуск/остановка сеанса лечения**

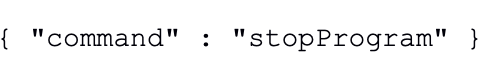
Любая программа лечения может быть запущена принудительно командой с мобильного устройтва, либо кнопкой "Старт/Стоп" на корпусе устройства.

Если в данный момент времени устройтво исполняет план лечения по расписанию, ручной запуск прервёт текущий сеанс. По выходу из ручного режима, устройство продолжит работать в режиме по расписанию.

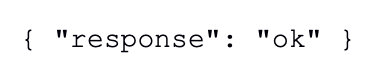
Формат команды запуска:



Формат команды останова:



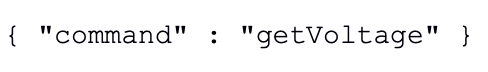
При нормальных условиях функциорирования устройства, всегда возвращается:



Возможна задержка до 1*сек* между моментом окончания обработки команды и моментом начала/окончания электромиостимуляции.

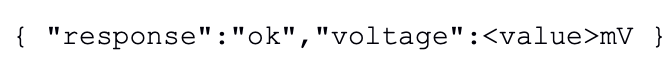
**Запрос напряжения батареи питания**

Команда может быть использована для мониторинга заряда батареи. Формат команды:



аргументы команды отсутствуют.

Формат ответа:



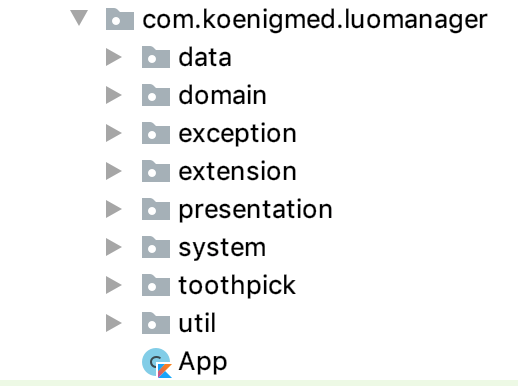
**Структура приложения**

Приложение написано на платформе Android на языке Kotlin. В приложении используется MVP (Model View Presenter) паттерн работы с интерфейсом, (DI) (Dependency injection) паттерн для внедрения зависимостей. Интерфейс приложения построен на принципе – “Одна активность — много фрагментов”. Для хранения данных приложение использует базу данных SQLite, также приложение обменивается данными с сервером через REST API в формате JSON.

Типы классов используемых в приложении:

1. Mapper – преобразует одни объекты в другие
2. Model – модель данных
3. Repository – объект для работы с данными полученными из памяти или из сети
4. TypeConverter – конвертирует одни типы данных в другие. Используется для конвертации объектов для последующей работы с базой данных
5. Dao – интерфейс для общения с базой данных
6. Database – объект ответственный за общение с базой данных
7. Deserializer – преобразователь полученных данных в объекты
8. Serializer – преобразователь объектов в данные для последующей отправки или хранения
9. TypeAdapter – адаптер для преобразования одних типов в другие
10. Interceptor – объект перехватывающий поток данных с возможностью его модификации
11. Api – объект отвечающий за обмен данными с программным интерфейсом – сервером или устройством.
12. Interactor – объект представляющий собой, высокоуровневую прослойку для работы с программным интерфейсом, обычно использует методы Api объекта.
13. Extension – объект описывающий методы дополняющие функционал существующим объектам
14. Handler – объект отвечающий за работу с определенным типом данных
15. Presenter – объект отвечающий за работу с данными графического интерфейса приложения
16. View – интерфейс для работы с графическим интерфейсом приложения
17. State – модель описывающая состояние объекта
18. Presentation – модель хранящая данные с графического интерфейса приложения
19. Delegate – класс берущий на себе некоторые функции другого класса
20. Adapter (Списка) – объект отвечающий за вывод данных списка на экран
21. Fragment – объект работающий на прямую с графическим интерфейсом, обязан быть привязан к активности
22. Activity – объект работающий напрямую с графическим интерфейсом приложения.
23. Callback – объект функции
24. Util – объект представляющий собой утилиту для работы с данными
25. Formatter – объект отвечающий за форматирование данных
26. Widget – объект интерфейса с обособленной логикой
27. Module – объект выполняющий ‘биндинг’ для последующего внедрения зависимостей (DI)
28. Provider – объект создающий различные объекты для последующей передачи в компоненты(DI)
29. Wrapper – объект представляющий собой оболочку для работы с другим объектом
30. Helper – объект содержащий в себе вспомогательные методы

Так выглядит директория проекта



App – представляет собой главный объект приложения. В нем инициализируются основные компоненты приложения.

При старте приложение инициализирует библиотеки Crashlitics - отвечающую за сбор информации об ошибках, Timber – отвечающую за форматирование логов, Toothpick – отвечающую за “инъекцию зависимостей” в приложение, Threeten – отвечающую за работу со временем.

**Внедрение зависимости**

Внедрение зависимости (англ. Dependency injection, DI) — процесс предоставления внешней зависимости программному компоненту. Является специфичной формой «инверсии управления» (англ. Inversion of control, IoC), когда она применяется к управлению зависимостями. В полном соответствии с принципом единственной обязанности объект отдаёт заботу о построении требуемых ему зависимостей внешнему, специально предназначенному для этого общему механизму.

Реализация внедрения зависимостей дает следующие преимущества:

* Повторное использование кода
* Легкость рефакторинга
* Простота тестирования

Библиотеки для работы с внедрением зависимостей в Android приложениях:

* Dagger – библиотека от Google, имеет 15тыс