МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Физико-технический факультет

Кафедра теоретической физики и компьютерных технологий

К У Р С О В О Й П Р О Е К Т

РАЗРАБОТКА ПРОТОТИПА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ РЕГИСТРАЦИИ БИОЛОГИЧЕСКИХ ПАРАМЕТРОВ

Работу выполнила _____Данилова Виолетта Валерьевна

Курс 2

Направление 09.03.02 Информационные системы и технологии

Научный руководитель
канд. биолог. наук, преподаватель_____Н. Н. Куликова

Нормоконтролер инженер_____Г.Д. Цой

Краснодар 2017

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сегодняшний день является временем научно-технического прогресса, очень сложно представить себе жизнь и быт современного общества без использования мобильных устройств. Ускоряется ритм жизни, вместе с ней ускоряется процесс создания обществом технических новинок для своего удобства. К ним относятся мобильные телефоны, которые уже есть практически у каждого человека. Очевидно, что мобильный телефон уже давно перестал быть просто средством общения. Обыкновенный разговор по телефону постепенно становится второстепенной функцией, пропадая в огромном наборе функций, реализуемых мобильным телефоном. Слушать музыку, фотографировать, играть... Этот список можно продолжать бесконечно. Таким образом, мобильный телефон стал многофункциональным устройством, позволяющим человеку пользоваться практически всеми современными технологиями.

На данный момент разработано большое количество приложений для мобильных устройств, решающих различные задачи в различных сферах .Сюда также относятся приложения для области здравоохранения.

Актуальность и практическая значимость разработки мобильного приложения для области здравоохранения состоит в увеличении комфорта и удобства для пользователя-пациента, для удобства хранения и обработки биологических данных- гомеостатических констант [1].

Гомеостатические константы - это контролируемые гомеостатической системой параметры (показатели), отражающие её функциональное состояние. Внутренняя среда организма представляет собой совокупность жидкостей организма, омывающих все органы и ткани и принимающих участие в обменных процессах, и включает плазму крови, лимфу и др. Кровь называют универсальной жидкостью, так как для поддержания нормального функционирования организма в ней должны содержаться все необходимые вещества,

т. е. внутренняя среда обладает постоянством – гомеостазом. Гомеостаз определяет динамическое постоянство внутренней среды и ее колебания в допустимых пределах. Хорошо известны биологические константы, при которых возможно полноценное существование организма: температура тела, кровяное давление, концентрация глюкозы и кислорода в крови, частота сердечных сокращений. Организм человека - открытая система, причем внешние воздействия постоянно дестабилизируют внутреннюю среду, нарушая ее постоянство, столь необходимое для полноценной жизнедеятельности. Гомеостаз поддерживается благодаря сложным скоординированным механизмам саморегуляции, среди которых важную роль играет обратная связь.

Отклонение любой гомеостатической константы от заданных пределов (нормы или оптимума) побуждает систему к восстановлению прежнего значения данной константы. Следовательно, для анализа общего физического состояния человека необходимы комплексные сведения об этих константах и анализ их отклонения от нормального значения.

Целью данной работы является разработка прототипа мобильного приложения для регистрации биологических параметров.

В задачи разработки мобильного приложения входят:

- прототип в виде схем и оконных интерфейсов;
- выбор инструментария для реализации приложения;
- разработка каркаса приложения;
- разработка модуля авторизации пользователя.

Разработка ведется для мобильной операционной системы Android - самой распространенной мобильной ОС на сегодняшний день, и на языке программирования высокого уровня Java.

1 Выбор инструментария для реализации приложения

1.1 Основные сведения о системе Android

Архитектура Android состоит из четырех уровней: уровень ядра, уровень библиотек и среды выполнения, уровень каркаса приложений (application framework) и уровень приложений. Система Android основана на ядре Linux версии 2.6. На уровне ядра происходит управление аппаратными средствами мобильного устройства. На этом уровне работают драйверы дисплея, камеры, клавиатуры, WiFi, аудиодрайверы. Особое место занимают драйверы управления питанием и драйвер межпроцессного взаимодействия (IPC).

Следующий уровень — это уровень библиотек и среды выполнения. Данный уровень представлен библиотеками libc (в Android она называется Bionic), OpenGL (поддержка графики), WebKit (движок для отображения Web-страниц), FreeType (поддержка шрифтов), SSL (зашифрованные соединения), SGL (2D-графика), библиотеки поддержки SQLite, Media Framework (нужна для поддержки мультимедиа). На этом же уровне работает Dalvik Virtual Machine — виртуальная машина Java, предоставляющая необходимую функциональность для Java-приложений [2].

Следующий уровень — уровень каркаса приложений. На уровне приложений работает большинство Android-приложений: браузер, календарь, почтовый клиент, навигационные карты и т. д. (рисунок 1).

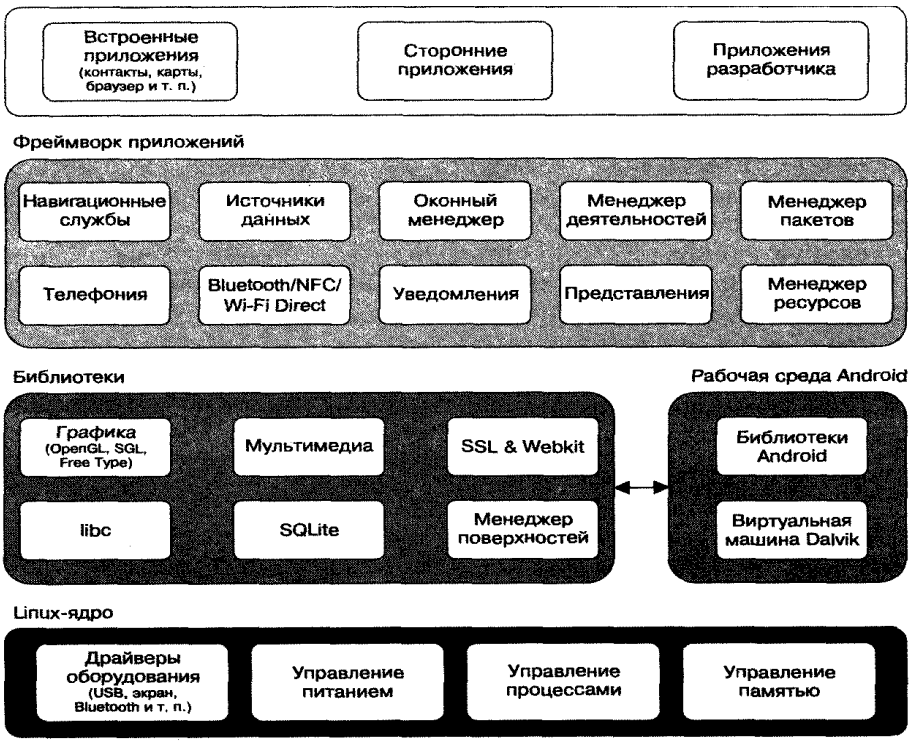


Рисунок 1 - Элементы, формирующие программный стек Android

Ключевым компонентом для создания визуального интерфейса в приложении Android является activity (активность). Нередко activity ассоциируется с отдельным экраном или окном приложения, а переключение между окнами будет происходить как перемещение от одной activity к другой. Приложение может иметь одну или несколько activity.

Все объекты activity представляют собой объекты класса android.app.Activity, которая содержит базовую функциональность для всех activity. В приложении из прошлой темы мы напрямую с этим классом не работали, а MainActivity наследовалась от класса AppCompatActivity. Однако сам класс AppCompatActivity, хоть и не напрямую, наследуется от базового класса Activity.

Все приложения Android имеют строго определенный системой жизненный цикл. При запуске пользователем приложения система дает этому приложению высокий приоритет. Каждое приложение запускается в виде отдельного процесса, что позволяет системе давать одним процессам более

высокой приоритет, в отличие от других. Благодаря этому, например, при работе с одними приложениями не блокировать входящие звонки. После прекращения работы с приложением, система освобождает все связанные ресурсы и переводит приложение в разряд низкоприоритетного и закрывает его.

Все объекты activity, которые есть в приложении, управляются системой в виде стека activity, который называется back stack. При запуске новой activity она помещается поверх стека и выводится на экран устройства, пока не появится новая activity. Когда текущая activity заканчивает свою работу (например, пользователь уходит из приложения), то она удаляется из стека, и возобновляет работу та activity, которая ранее была второй в стеке (рисунок 2).

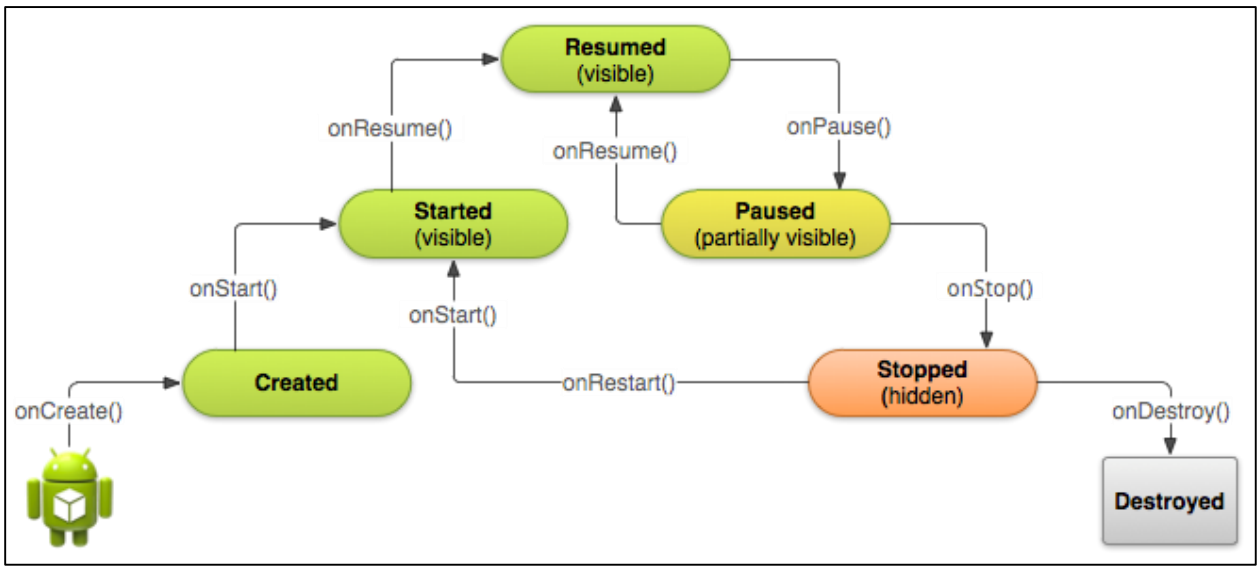


Рисунок 2 – Схема жизненного цикла приложения на Android

1.2 Сведения о языке программирования Java

Java-интерпретируемый, сильно типизированный язык программирования высокого уровня. Разработан компанией Sun Microsystems (в последую-

щем приобретённой компанией Oracle). Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре, с помощью виртуальной Java-машины. Дата официального выпуска — 23 мая 1995 года.

Ключевой особенностью языка Java является то, что его код сначала транслируется в специальный байт-код, независимый от платформы. А затем этот байт-код выполняется виртуальной машиной JVM (Java Virtual Machine). В этом плане Java отличается от стандартных интерпретируемых языков как PHP или Perl, код которых сразу же выполняется интерпретатором. В то же время Java не является и чисто компилируемым языком, как C или C++.

Подобная архитектура обеспечивает кроссплатформенность и аппаратную переносимость программ на Java, благодаря чему подобные программы без перекомпиляции могут выполняться на различных платформах - Windows, Linux, Solaris и т.д. Для каждой из платформ может быть своя реализация виртуальной машины JVM, но каждая из них может выполнять один и тот же код [3].

Java является языком с Си-подобным синтаксисом и близок в этом отношении к C/C++ и C#.

Еще одной ключевой особенностью Java является то, что она поддерживает автоматическую сборку мусора. Это значит, что в не нужно освобождать вручную память от ранее использовавшихся объектов, как в C++, так как сборщик мусора это сделает автоматически.

Java является объектно-ориентированным языком. Он поддерживает полиморфизм, наследование, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений.

В данной работе используется версия платформы Java SE — Java Standard Edition, основное издание Java, содержит компиляторы, API, Java Runtime Environment; подходит для создания пользовательских приложений.

1.3 Анализ и выбор программного обеспечения для разработки мобильного приложения

В данный момент для создания мобильных приложений на платформе Android существует два типа средств разработки: средства разработки для создания нативных мобильных приложений и средства создания web–приложений адаптированных под мобильные приложения. Для анализа нами были выбраны средства разработки нативных приложений так как в данной среде они более популярны и востребованы. Мы рассмотрели три наиболее популярных средства разработки: AndroidStudio, Eclipse, NetBeans IDE. Для анализа данных программных продуктов были выбраны следующие критерии:

-функциональность;

-удобство интерфейса;

-возможность подключения дополнительных модулей;

-требовательность к системе;

-наличие встроенных компонентов тестирования приложения.

Android Studio – продукт компании Google. Основана на программном обеспечении IntelliJ IDEA от компании JetBrains, официальное средство разработки Android приложений. Актуальная на данный момент версия приложения - 2.3.2. Данная среда разработки доступна для Windows, OS X и Linux. Функционал данного приложения использует язык Java для написания программного кода. Разработка интерфейса производится drag–and–drop методом, но так же имеется возможность использовать XML. Интерфейс данного ПО перегружен. Интерфейс библиотек приложения имеет вид выпадающего древа и под него приходится отводить очень много места в общем интерфейсе, в противном случае, информация становится нечитаемой. Та же самая ситуация и с окном отладки. В функционале Android Studio возможность подключения дополнительных плагинов отсутствует. Данное средство разработ-

ки очень требовательно к технической составляющей ЭВМ, по сравнению с другими средствами разработки. Минимальное количество ОЗУ требуемое для данного продукта 2 гигабайта. Но для комфортной работы с данной программой рекомендуемое количество памяти 8 гигабайт, что не является проблемой для современных компьютеров, но на ПК старше 2014 года данная среда функционирует очень медленно, не говоря о параллельном запуске других, даже не очень требовательных, приложений. Имеет встроенный модуль для эмуляции Android-устройства. Данный эмулятор требует отдельных ресурсов, что еще сильнее повышает требовательность ПО к ЭВМ [4].

Eclipse – среда разработки созданная компанией Eclipsefoundation. Актуальная на данный момент версия Eclipse 4.6 (Neon). Используемый язык для написания мобильных приложений – Java. Включает в свой функционал не только средства для разработки мобильных приложений, но и web – приложений, поддерживает языки C++и PHP. Простой и удобный интерфейс. Панель библиотек имеет древовидную структуру, но проблема предыдущего продукта здесь решена просто – существует возможность полностью сворачивать неиспользуемые в данный момент окна. Есть возможность подключения дополнительных плагинов для расширения функционала. Для этого в продукте предусмотрен модуль Eclipse Marketplace, предоставляющий на выбор сразу три «Рынка» плагинов: Eclipse Marketplace, Obeo Marketplace и RedHat [5]. Так же существует возможность написания собственных плагинов и их использования без получения лицензии или обязательного предоставления разработки на рынке. Системные требования для данного ПО не описаны разработчиком, но при использовании на ПК средней производительности 2012 года затруднений замечено не было. Эмуляция устройства отсутствует полностью.

NetBeans IDE – продукт компании NetBeans Community. Актуальная на данный момент версия приложения 8.2. В данном программном продукте функционал полностью реализуется посредствам плагинов. Вследствие этого, ПО поддерживает большое количество используемых языков. Для разра-

ботки нативных мобильных приложений используется Java, но существует возможность создания web–приложения написанного на HTML5 или JS+PHP. Встроенный отладчик полностью отсутствует, но имеется возможность подключения удаленного отладчика через сеть «Интернет». Интерфейс очень похож на интерфейс Eclipse, за исключением отсутствия окна отладки, и наличия разметки номеров строк. Требования к ЭВМ либеральные. Для минимальной работы продукта требуется 512 мегабайт ОЗУ, для более комфортной работы рекомендуется использовать компьютер с 2 гигабайтами. Встроенные компоненты для тестирования приложения полностью отсутствуют [6]. Таким образом, анализ средств разработки мобильных приложений можно свести в таблицу, оценивая рассмотренные критерии по пятибалльной шкале [7] (таблица 1).

Таблица 1 - Сравнительный анализ средств разработки мобильных приложений по 5- балльной шкале

Критерии оценки	Средства разработки мобильных приложений		
Функциональность	AndroidStudio	Eclipse	NetBeansIDE
Удобство интерфейса	5	4	3
Возможность подключения дополнительных модулей	2	5	5
Требовательность к системе	1	5	4
Наличие встроенных компонентов тестирования приложения	3	1	1

На основе анализа средств разработки был выбран Eclipse Indigo for Java Developers,так как Eclipse менее требователен к системным ресурсам компьютера и не менее мощный, чем остальные среды разработки.

Для работы с языком программирования Java необходимо установить JDK (Java Development Kit). На сегодняшний день актуальная версия JDK-версия 1.8.0_021,необходим Java SE –стандартный набор разработчика, именно его поддерживает Android SDK [8].

Следующий инструмент - Android SDK, стандартный набор разработчика Android. Полное название программы- Android SDK and AVD Manager. Основная задача этой программы — загрузить и установить с Интернета все необходимое для разработки приложений для ОС Android выбранной версии. Она содержит следующие компоненты:

-API-библиотеки, необходимые для разработки Android-приложений;

-документацию по Android;

-эмулятор Android-устройства (Android Virtual Device, AVD), позволяющий запускать и тестировать программы без наличия физического мобильного устройства;

-инструментальные средства для разработки, позволяющие компилировать и отлаживать приложения.

Android Development Tools(ADT) - плагин для среды разработки Eclipse[9], позволяет связать воедино собственно среду разработки и Android SDK, содержит следующие модули:

-мастер проектов Android, который помогает создавать новые проекты и содержит готовые шаблоны приложений;

-основанный на формах манифест, шаблон и редактор ресурсов, которые помогают создавать и редактировать XML-ресурсы;

-службу мониторинга отладки Dalvik (DDMS);

-функцию отладки на этапе выполнения, журнал событий и выводимый в консоль список сообщений, а также другие полезные для разработчика средства.

1.4 Методики тестирования мобильного приложения

За термином «тестирования мобильных технологий» скрываются различные виды испытаний, в том числе тестирование приложений, специально предназначенных для мобильных устройств, тестирование мобильных устройств и тестирование мобильных веб-приложений. Под тестированием мобильных приложений понимаются тестовые мероприятия, проводимые по отношению к приложениям с использованием проработанных тестовых методов и инструментов, гарантирующих соответствие заявленным функциям, поведению, производительности, качеству обслуживания и характерным особенностям: мобильности, удобству использования, интероперабельности, связности, безопасности и конфиденциальности.

Тестирование мобильных приложений отличается от тестирования обычного ПО наличием ряда уникальных требований. Прежде всего, мобильные приложения должны правильно выполняться в любое время и в любом месте. Необходимо, чтобы они корректно функционировали на разных платформах, которые отличаются используемыми операционными системами, размерами экрана, вычислительными ресурсами и продолжительностью непрерывной работы от батарей. Мобильные приложения должны поддерживать множество каналов ввода (клавиатура, голос, жесты и т. д.), мультимедийные технологии и обладать другими особенностями, повышающими удобство их использования. Для удержания низких цен на оборудование необходимо широко использовать средства моделирования и виртуализации. И наконец, поскольку большинство мобильных сервисов поддерживают достаточно широкий спектр беспроводных сетей (2G, 3G, 4G, Wi-Fi, WiMax),

мобильные приложения должны нормально функционировать в неоднородной сетевой среде.

С учетом указанных требований при тестировании мобильных приложений необходимо сосредоточиться на следующих целях и мероприятиях:

тестирование функциональности и поведения — позволяет оценить сервисные функции, интерфейсы API для мобильных веб-приложений, поведение внешних систем, интеллектуальность системы и пользовательские интерфейсы;

-тестирование качества обслуживания — позволяет оценить нагрузку на систему, производительность, устойчивость и готовность, масштабируемость и пропускную способность;

-тестирование интероперабельности — дает возможность проверить способность системы работать в условиях различных устройств, платформ, браузеров и беспроводных сетей;

-тестирование удобства использования и интернационализации — позволяет сделать оценку контента пользовательского интерфейса, потоков и сценариев операций пользователей, применения мультимедийных средств и управления при помощи жестов;

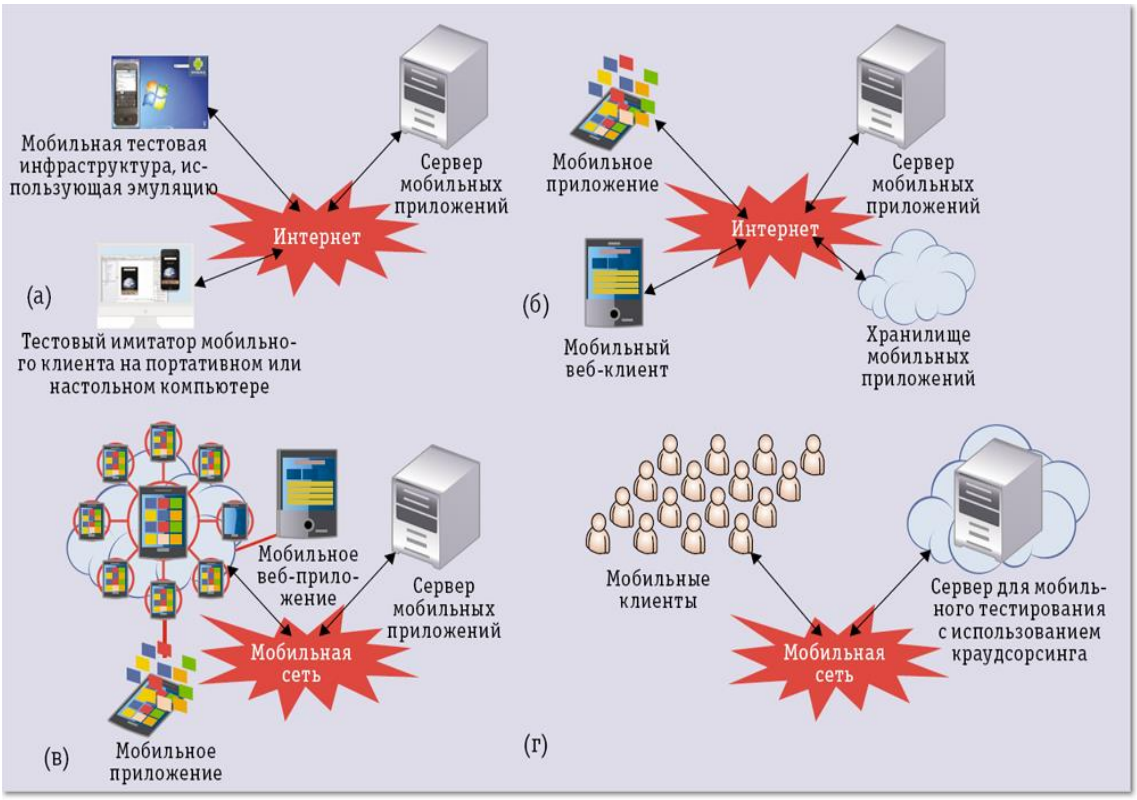
-тестирование безопасности и конфиденциальности — осуществляется для проверки процедур аутентификации пользователей, безопасности устройств, безопасности сеансов работы, возможности проникновения в системы и сети, соблюдения безопасности сквозных транзакций и конфиденциальности пользовательской информации;

-тестирование мобильности — позволяет оценить работу функций, использующих информацию о местоположении, профили клиентов, системные и пользовательские данные;

-тестирование совместимости и связности — позволяет проверить совместимость с мобильными браузерами и платформами и оценить возможность диверсификации соединений беспроводных сетей;

-тестирование мультиарендности — проводится для проверки функций, связанных с множественной арендой приложений, оценки поведения системы, системных данных и пользовательских интерфейсов.

Для создания среды тестирования мобильных приложений применяется несколько различных стратегий. Можно выделить четыре популярных подхода к тестированию мобильных приложений, которые основаны на использовании базовой клиент-серверной инфраструктуры (рисунок 3).



а - эмуляция б- облако, в - устройства, г – краудсорсинг;

Рисунок 3 - Инфраструктуры мобильного тестирования

Тестирование на основе эмуляции. Этот способ тестирования предусматривает использование эмулятора мобильного устройства, имитирующего его поведение в виртуальной машине. Обычно такие эмуляторы бывают включены в состав комплекта инструментов для разработчика, прилагаемого к мобильной платформе (например, в состав SDK Android). Цена такого ре-

шения относительно невелика, потому что в этом случае нет необходимости в создании тестовой лаборатории, покупке или аренде физических устройств. Однако эмуляцию можно использовать только для оценки функциональности системы в весьма ограниченном контексте. Например, проверить обработку всех жестов в полном объеме довольно затруднительно, потому что большинство эмуляторов поддерживают лишь ограниченный набор жестов и лишь специфичные для конкретного устройства функции. Другой недостаток связан с ограниченными масштабами тестирования качества обслуживания приложения (QoS, Quality of Service).

Тестирование на базе устройств. Для такого тестирования требуется создание тестовой лаборатории и покупка реальных мобильных устройств, поэтому и стоит оно гораздо дороже эмулирующего подхода. Но зато в этом случае можно проверить функции и поведение конкретных устройств, а также параметры качества обслуживания. Кроме того, можно оценить возможности базовых мобильных сетей, выбирая и настраивая их конфигурацию в тестовой среде. Одна из главных проблем такого подхода заключается в том, что мобильные устройства и платформы меняются очень быстро. Трудности обусловлены также ограниченными возможностями проверки QoS, поскольку для нее требуется много мобильных устройств.

Тестирование в облаке. При таком подходе обычно используется облако, поддерживаемое поставщиками инструментов тестирования — например, такими как NTT Data. Основная идея подхода этой компании к облачному тестированию на базе устройств заключается в том, чтобы построить облако из мобильных устройств, которое поддерживало бы сервисы тестирования на крупномасштабной основе. Мобильные пользователи получают требуемую тестовую среду на условиях аренды. Этот подход отличается от других более высокой эффективностью при масштабном использовании приложений, а также при проведении разнообразных тестовых мероприятий на мобильных устройствах.

Тестирование с использованием краудсорсинга. Краудсорсинговый подход предполагает привлечение фрилансеров, инженеров, работающих по контракту, или сообществ конечных пользователей , а также формирование краудсорсинговой тестовой инфраструктуры и установку сервера управления сервисами для поддержки неоднородных пользователей. Сегодня провайдеры сервисов по реализации такого подхода предлагают базовое управление тестами, сервис тестирования и выдачу отчетов об ошибках. Однако большинство тестовых мобильных операций управляются инструментами автоматизации мобильного тестирования с весьма ограниченными возможностями — тестировщики получают преимущества стихийного тестирования, не требующего проведения инвестиций в лабораторию и покупку или аренду устройств, но возникает риск низкого качества тестирования и неопределенности сроков его проведения [10].

2 Разработка прототипа мобильного приложения для регистрации биологических параметров

2.1 Концепция мобильного приложения

Мобильное клиентское приложение (далее – мобильное приложение) должно обеспечивать авторизацию гражданина в системе, например, по адресу электронной почты, мобильного телефона или номеру СНИЛС, ОМС или ИНН, для чего должен осуществляться соответствующий запрос к базе данных.

После авторизации в мобильном приложении пользователь видит на экране смартфона главное меню, обеспечивающее возможность добавлять, удалять, редактировать данные медико-биологических констант и персональные данные, смотреть предупреждения [11].

Существенной особенностью мобильного приложения является контроль вхождения биологических параметров в допустимый коридор безопасных для жизни и здоровья значений. При отклонении полученных результатов от допустимых (например, скачок артериального давления) мобильное приложение посылает сигнал тревоги на и выводит предупреждение для пользователя, предлагая обратиться к специалисту.

Основные функции разрабатываемого мобильного приложения:

-хранение данных клиента – анализы, ЧСС, АД и т.д;

-ввод данных – ручной режим;

-отслеживание критических значений биологических параметров;

-история анализов и измеренных параметров;

Меню клиентского приложения должно обеспечивать возможность:

- ввести данные анализов вручную;

- посмотреть историю введенных анализов и отследить динамику изменения измеряемых величин на графике;

- изменить настройки мобильного приложения.

2.2 Модель мобильного приложения в нотациях UML и IDEF

Исходя из требований функциональности разрабатываемого прототипа была разработана use case диаграмма [12] (вариантов использования), отражающая взаимодействие пользователя с системой (рисунок 4).

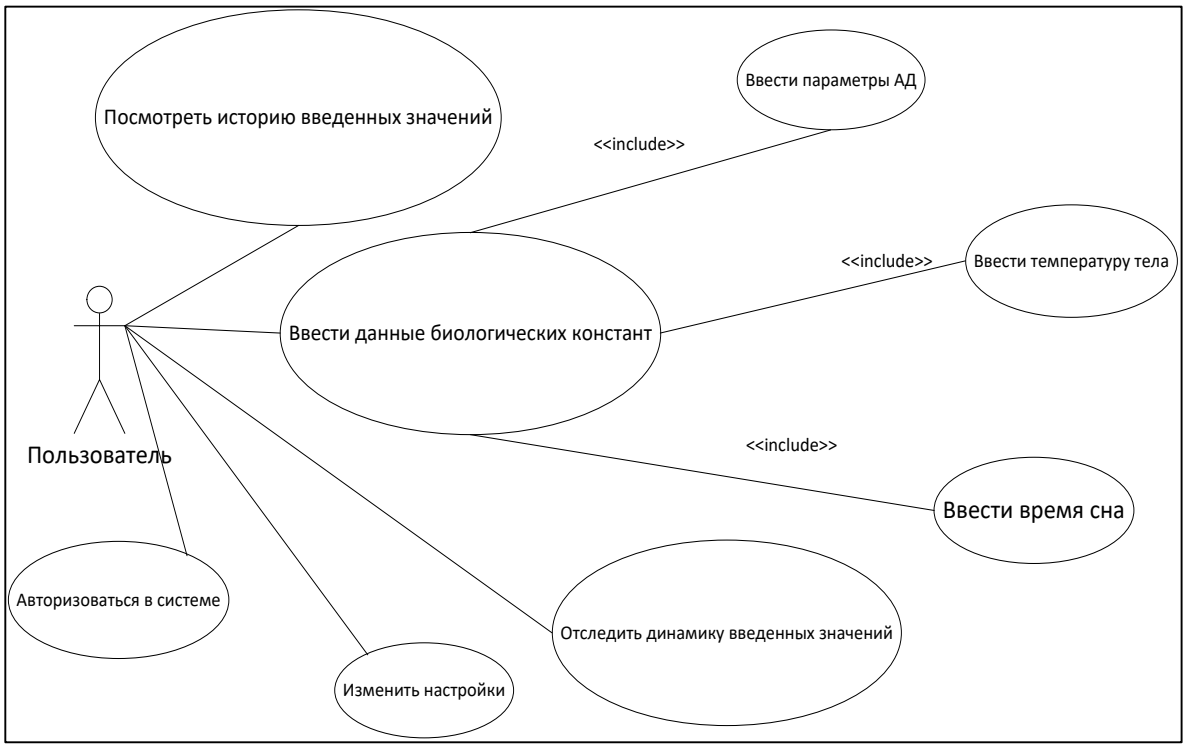


Рисунок 4- Диаграмма вариантов использования мобильного приложения

На данной диаграмме отображены возможности действий пользователя как внешнего элемента по отношению к проектируемой системе. Является важной деталью при проектировании пользовательского интерфейса.

Следующая диаграмма отображает потоки данных в системе (DFD Diagramm). В соответствии с данным методом модель системы определяется как иерархия диаграмм потоков данных, описывающих асинхронный процесс преобразования информации от ее ввода в систему до выдачи потребителю [13]. Источники информации (внешние сущности) порождают информационные потоки (потоки данных), переносящие информацию к подсистемам или процессам. Те, в свою очередь, преобразуют информацию и порождают новые потоки, которые переносят информацию к другим процессам или подсистемам, накопителям данных или внешним сущностям — потребителям информации. Диаграмма разработана методом Гейна-Сарсона (рисунок 5).

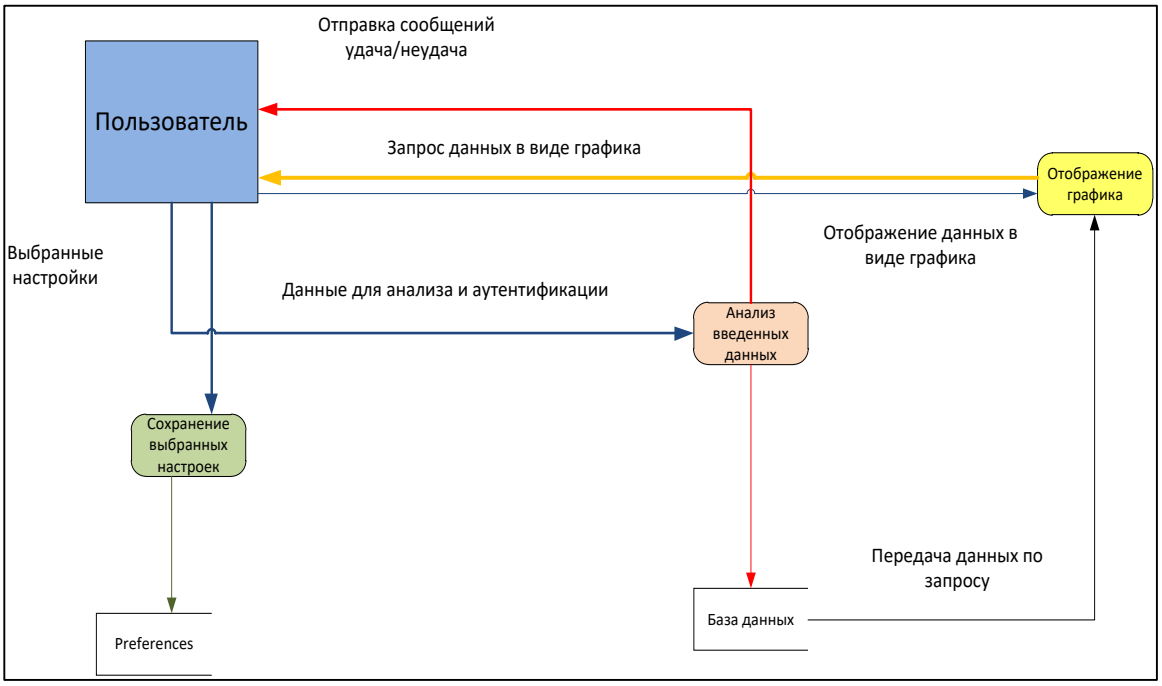


Рисунок 5-Диаграмма потоков данных

На основе функциональных требований и вышеизложенных диаграмм была разработана система классов прототипа приложения с учетом особенностей реализации для системы Android с использованием класса Фрагмент.

Фрагмент (класс Fragment) представляет поведение или часть пользовательского интерфейса в операции (класс Activity). Разработчик может объ-

единить несколько фрагментов в одну операцию для построения многопанельного пользовательского интерфейса и повторного использования фрагмента в нескольких операциях. Фрагмент можно рассматривать как модульную часть операции. Такая часть имеет свой жизненный цикл и самостоятельно обрабатывает события ввода. Кроме того, ее можно добавить или удалить непосредственно во время выполнения операции. Это нечто вроде вложенной операции, которую можно многократно использовать в различных операциях.

Фрагменты впервые появились в Android версии 3.0 (API уровня 11), главным образом, для обеспечения большей динамичности и гибкости пользовательских интерфейсов на больших экранах, например, у планшетов. Поскольку экраны планшетов гораздо больше, чем у смартфонов, они предоставляют больше возможностей для объединения и перестановки компонентов пользовательского интерфейса. Фрагменты позволяют делать это, избавляя разработчика от необходимости управлять сложными изменениями в иерархии представлений. Разбивая макет операции на фрагменты, можно модифицировать внешний вид операции в ходе выполнения и сохранять эти изменения в стеке переходов назад, которым управляет операция.

Например, новостное приложение может использовать один фрагмент для показа списка модулей, а другой – для отображения страницы модуля справа. Оба фрагмента отображаются за одну операцию рядом друг с другом, и каждый имеет собственный набор методов обратного вызова жизненного цикла и управляет собственными событиями пользовательского ввода. Таким образом, вместо применения одной операции для выбора раздела, а другой — для операций с интерфейсом раздела, пользователь может выбрать раздел и взаимодействовать с интерфейсом модуля в рамках одной операции.

Каждый фрагмент разрабатывается как модульный и повторно используемый компонент операции. Поскольку каждый фрагмент определяет собственный макет и собственное поведение со своими обратными вызовами жизненного цикла, можно включить один фрагмент в несколько операций.

Поэтому необходимо предусмотреть повторное использование фрагмента и не допускать, чтобы один фрагмент непосредственно манипулировал другим. Это особенно важно, потому что модульность фрагментов позволяет изменять их сочетания в соответствии с различными размерами экранов. Если приложение должно работать и на планшетах, и на смартфонах, можно повторно использовать фрагменты в различных конфигурациях макета, чтобы оптимизировать взаимодействие с пользователем в зависимости от доступного размера экрана. Например, на смартфоне может возникнуть необходимость в разделении фрагментов для предоставления однопанельного пользовательского интерфейса, если не удается поместить более одного фрагмента в одну операцию (рисунок 6).

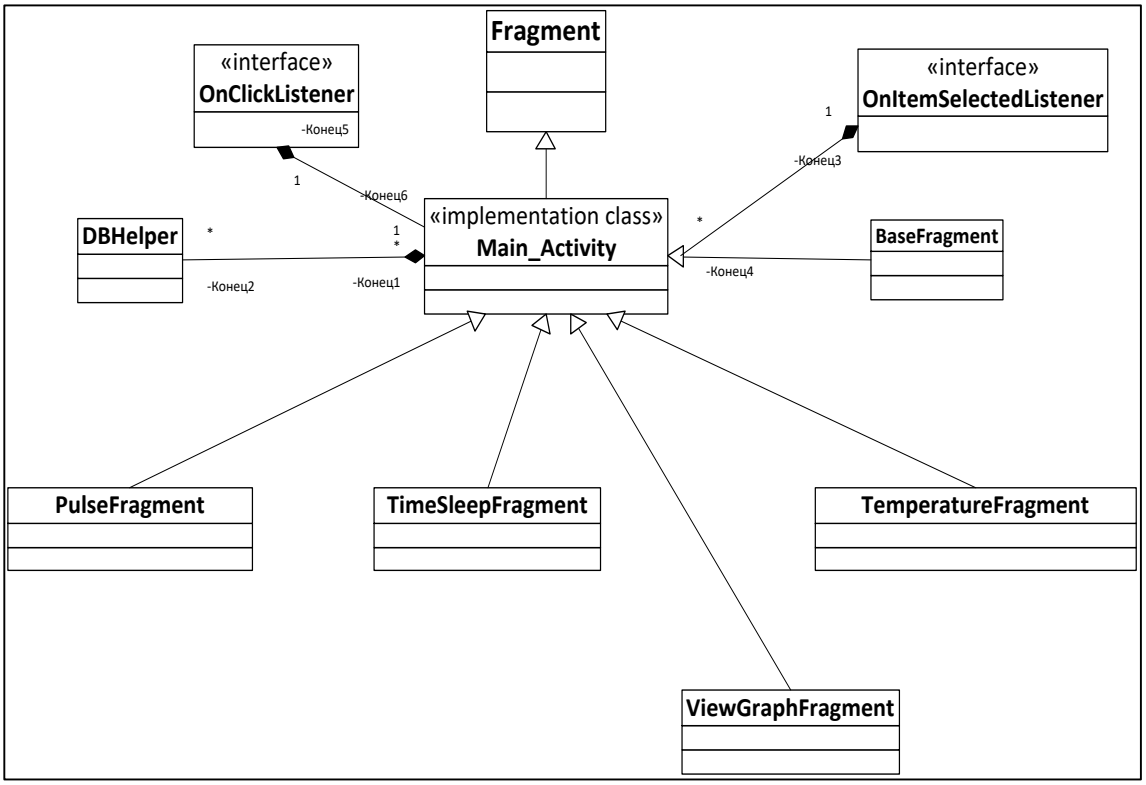


Рисунок 6-Диаграмма классов

Приложение предназначено для длительного хранения упорядоченных массивов данных. Для реализации этой функции разработана модель реляционной базы данных.

БД представлена в виде сущностей, их атрибутов и связей между ними. Каждая сущность представляет множество подобных объектов, называемых экземплярами. Каждый экземпляр индивидуален и должен отличаться от всех остальных. Атрибут выражает определенное свойство объекта. С точки зрения физической модели БД сущности соответствует таблица (например: “Пользователь”, “Время сна ”), экземпляру сущности – строка в таблице, а атрибуту – колонка таблицы (например, строка “Номер СНИЛС” в таблице “Пользователь ”). В результате проектирования были выделены 4 сущности, и независимой является только одна - сущность “Пользователь”, остальные являются подчиненными (рисунок 7).

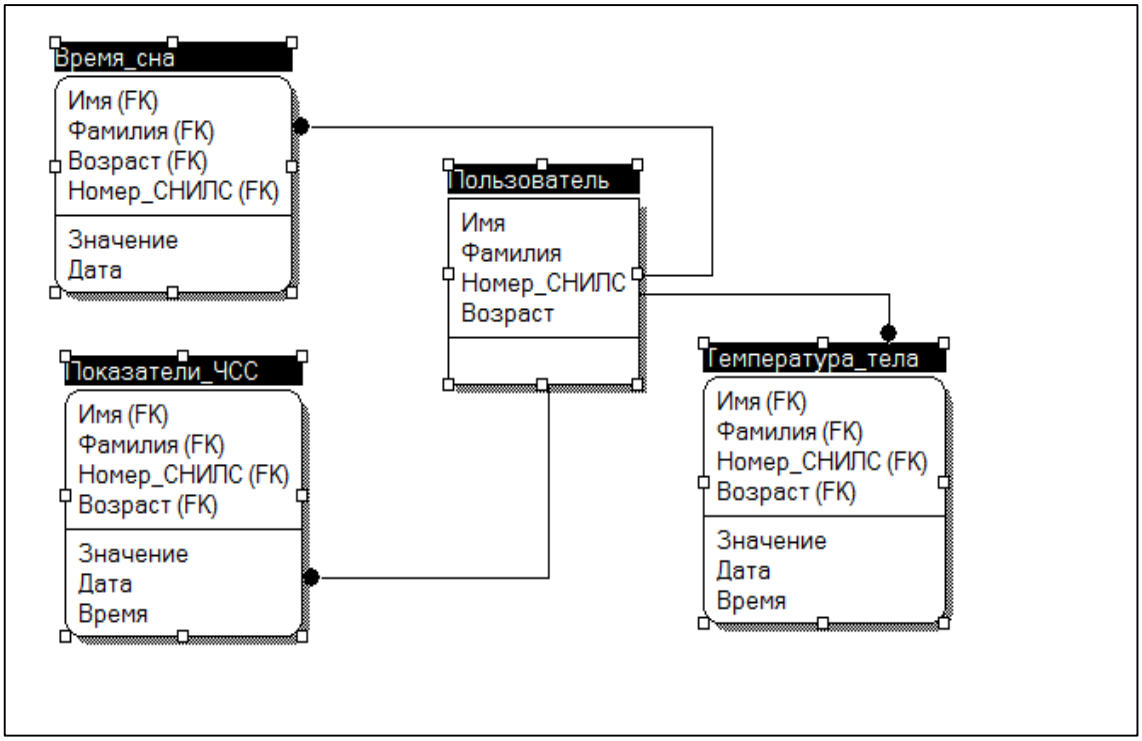


Рисунок 7-Схема базы данных приложения

2.3 Реализация прототипа мобильного приложения

В прототипе приложения для реализации пользовательского интерфейса использован шаблон Navigation Driver, что позволяет создать удобный ,понятный и привычный интерфейс.

Navigation Drawer — это выпадающее меню, которое появляется при клике пользователя на иконку в Action Bar приложения в левом верхнем углу. Такое меню заслоняет лишь часть экрана, накладываясь сверху на его левую часть. В выпадающем списке отображаются пункты меню, позволяющие быстро перейти в нужную часть приложения. Паттерн Navigation Drawer помогает улучшить юзабилити в тех ситуациях, когда приложение содержит множество различных разделов.

Для реализации ввода анализов вручную в приложении предусмотрены модули, различающиеся по типу вводимых параметров. Например, частоту сердечных сокращений можно ввести и сохранить в модуле “Пульс”, для чего предусмотрены поля ввода и кнопки сохранения и перехода (Листинг 1). После получения введенных данных происходит их обработка и уведомление пользователя о результатах, в случае получения критических параметров значений система выводит сообщение.

Листинг1:

```
int Pulse=Integer.parseInt(enterVal.getText().toString());
if(Pulse<=60){
AlertDialog.Builder builder = new          AlertDia-
log.Builder(MainActivity.this);//обработка введенных значений
builder.setTitle("Пульс")
.setMessage("Пульс замедленный,обратитесь к врачу!")
.setCancelable(false)
.setNegativeButton("OK,",
new DialogInterface.OnClickListener() {
     public void onClick(DialogInterface dialog, int id) {
```

```
dialog.cancel();
}
});
AlertDialog alert = builder.create();
alert.show();
```

Для хранения полученных данных создается таблица базы данных SqLight, встроенная база Android ОС (Листинг 2).

Листинг2:

```
public void onCreate(SQLiteDatabase db) {
// создаем таблицу с полями
db.execSQL("create table Pulse ("
+ "id integer primary key autoincrement,"
+ "value,"
+ "date" + ");");
}
SQLiteDatabase db = mDbHelper.getWritableDatabase();

ContentValues values = new ContentValues();
values.put(PULSE.COLUMN_VALUE, value);
values.put(PULSE.COLUMN_DATA, data);

// Вставляем новый ряд в базу данных и запоминаем его
идентификатор
long newRowId = db.insert(PULSE.TABLE_NAME, null, values);

// Выводим сообщение в успешном случае или при ошибке
if (newRowId == -1) {
// Если ID  -1, значит произошла ошибка
Toast.makeText(this, "Ошибка!",
Toast.LENGTH_SHORT).show();
} else {
```

```
        Toast.makeText(this, "Сохранено! ",
Toast.LENGTH_SHORT).show();
        }
        }
```

2.4 Тестирование прототипа приложения

Для тестирования прототипа применялись два подхода: тестирование на основе эмуляции и на основе реальных устройств. Каждый из них следует рассмотреть отдельно.

Тестирование на основе эмуляции. Этот способ тестирования предусматривает использование эмулятора мобильного устройства, имитирующего его поведение в виртуальной машине. Использовался эмулятор, включенный в состав комплекта инструментов для разработчика, прилагаемого к мобильной платформе (в составе SDK Android) (рисунок 8).

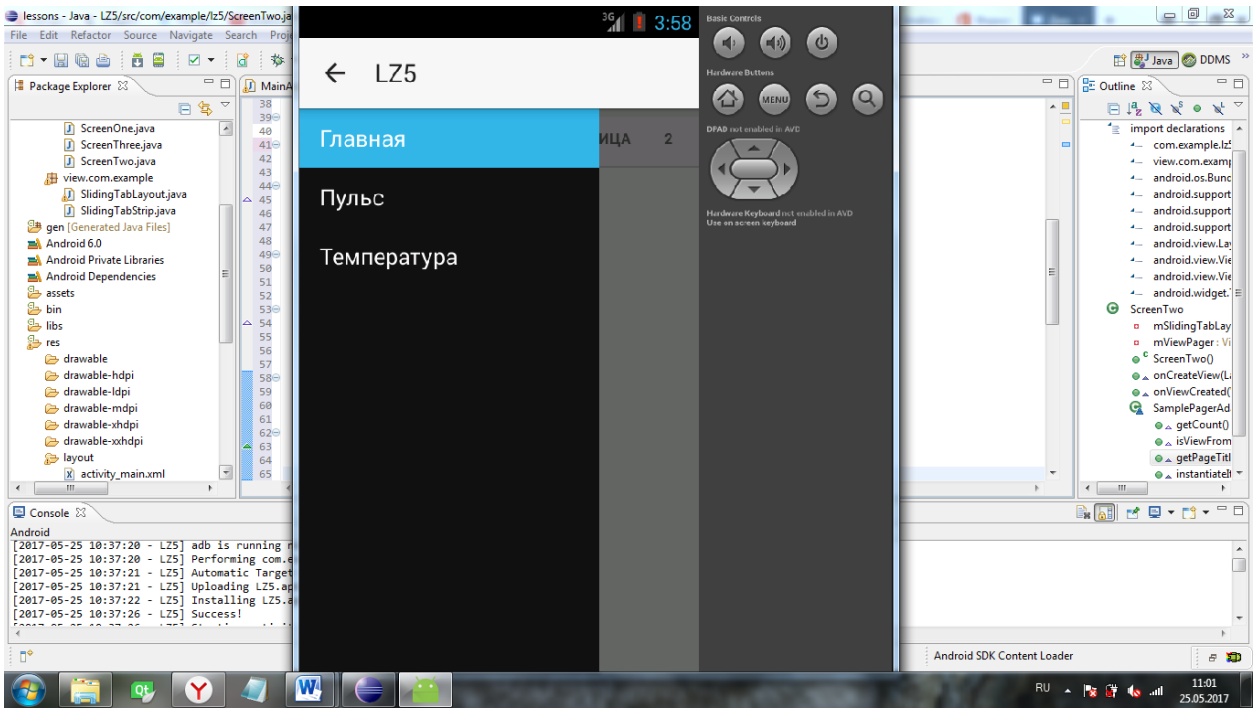


Рисунок 8 – Прототип приложения запущенный на эмуляторе

Тестирование на базе устройств. В результате тестирования было выявлено, что разработанный прототип приложения хорошо работает на различных версиях системы Android, и на устройствах с различными аппаратными характеристиками, однако пользовательский интерфейс и функционал требуют значительной доработки (рисунок 9).

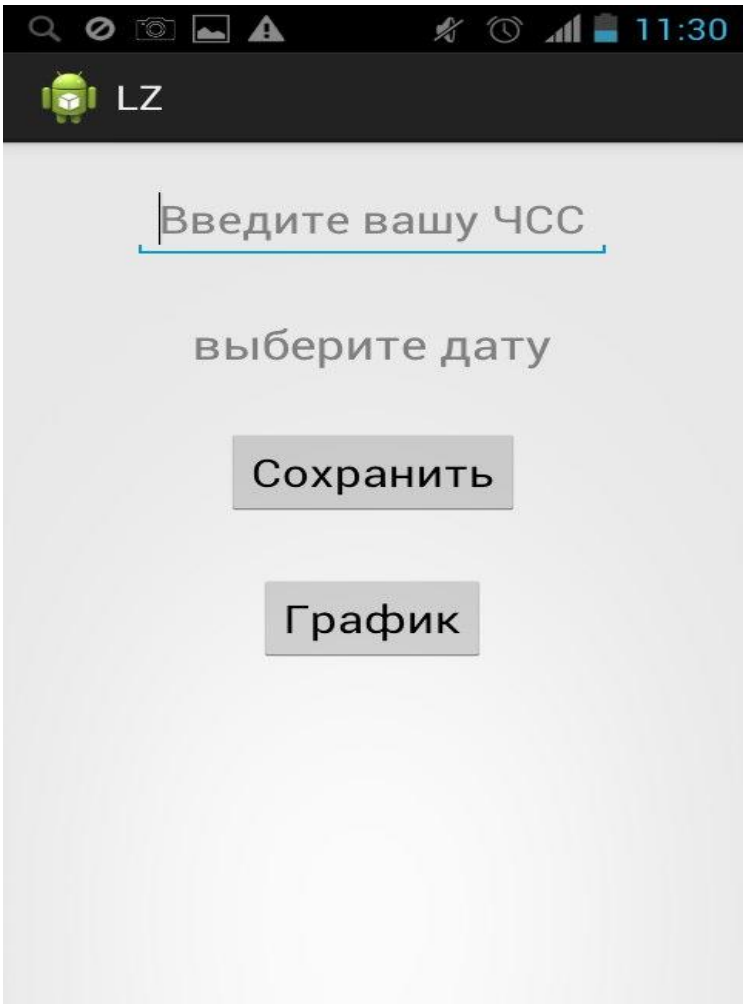


Рисунок 9 – Один из модулей прототипа мобильного приложения на экране смартфона

ЗАКЛЮЧЕНИЕ

Данная курсовая работа содержит сведения о разработке прототипа мобильного приложения для регистрации биологических параметров. Для достижения поставленной цели в качестве инструментального средства разработки были выбраны язык программирования высокого уровня Java и среда программирования Eclipse с установленным и настроенным набором инструментов Android SDK.

Основные результаты данной курсовой работы состоят в следующем:

1. Реализован прототип в виде схем и оконных интерфейсов.

2. Выбран инструментарий для реализации приложения.

3. Разработан каркас приложения.

4. Разработан модуль авторизации пользователя.

В дальнейшей перспективе будут разработаны следующие модули:

-модуль геолокации, позволяющий определить пользователю местоположение ближайшего лечебно-профилактического учреждения;

-модуль получения данных биологических параметров с фитнес-браслетов ;

Также в дальнейшем прототип приложения будет доработан для размещения в Google Play.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ:

1 Фирсова С. С. Конспект лекций по нормальной физиологии./ С. С. Фирсова, С. И. Кузина . – М: Эксмо, 2014. –160 с.

2 Колисниченко Д. Н. Программирование для Android. Самоучитель. / Д.Н. Колисниченко. – СПб.: БХВ-Петербург, 2014. – 272 с.

3 Лафоре Р. Структуры данных и алгоритмы в Java. / Р. Лафоре.– СПб.: Питер, 2013. –704 с.

4 Developer Android.Официальный сайт поддержки инструмента разработчика Android Studio. (Engl). – URL:https:/ developer. android.com/dio/index .html [11 May 2017].

5 Eclipse. Официальный сайт разработчика IDE Eclipse. (Engl).– URL:http:/www.eclipse.org/ide/ [25 April 2015].

6 Дэвид Хеффельфингер. Разработка приложений Java EE 6 в NetBeans 7./Дэвид Хеффельфингер. – ДМК Пресс, 2014. – 230 с.

7 Волков А.И. Анализ средств разработки мобильных приложений // Современная техника и технологии. 2017. № 5 [Электронный ресурс]. (Рус.) – URL: http:/technology.snauka.ru/2017/05/13222 [15 мая 2017].

8 Developer Android. Официальный сайт поддержки инструмента разработчика. (Engl). – URL:https:/ developer.android.com/ studio/intro/update.html [12 May 2017].

9 Фелкер Донн. Android: разработка приложений. /Донн Фелкер. — М.: ООО “И.Д. Вильямс”, 2015. — 336 с.

10 Тестирование мобильных приложений / Джерри Гао, Сяоин Бай, Вей-Тек Цай, Тадахиро Уэхара. // Открытые системы СУБД. – 2015. – № 3. (Рус.) – URL: https://www.osp.ru/os/2015/03/13040836 [09 апреля 2017].

11 Информационная система мониторинга биологических констант жизнедеятельности индивида как платформа для поддержки принятия решений о необходимости врачебной диагностики /Свергун С. В., Куликова Н. Н., Гусев А.А.// Международный научно-исследовательский журнал. – 2017. –

№ 4. (Рус.) – URL: http://research-journal.org/technical/informacionnaya-sistema-monitoringa-biologicheskix-konstant-zhiznedeyatelnosti-individa-kak-platforma-dlya-podderzhki-prinyatiya-reshenij-o-neobxodimosti-vrachebnoj-diagnostiki/ [22 мая 2017].

12 А. В. Леоненков Самоучитель UML - 4 издание./Леоненков А. В. – СПб.: БХВ-Петербург, 2014. — 272 с.

13 В.И. Грекул. Проектирование информационных систем. [Электронный ресурс]. Национальный исследовательский университет "Высшая Школа Экономики" Национальный открытый институт Интуит. (Рус.). – URL: http://www.intuit.ru/ebooks/8197 [21 мая 2017]