МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
вычислительной техники	МО и ПЭВМ

Направление подготовки 09.03.02 «Информационные системы и технологии»

БАКАЛАВРСКАЯ РАБОТА

на тему

Мобильное электронное приложение «Медицинский справочник»

Студент _____ Попченков Антон Сергеевич
(подпись, дата) (ФИО полностью)

Руководитель _____ Шашков Б.Д.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____№_____)

Заведующий кафедрой _____ Макарычев П.П.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____№____)

Секретарь ГЭК _____ _____
(подпись) (фамилия, инициалы)

Пенза, 2021

Федеральное государственное бюджетное образовательное учреждение высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
Вычислительной техники	МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____П.П. Макарычев
«__» _____2021 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент Попченков Антон Сергеевич гр. 17ВИ1 факультета ВТ
<Фамилия, имя, отчество полностью>
2. Тема работы Мобильное электронное приложение «Медицинский справочник»

Тема утверждена приказом ПГУ №_____от "_____"_____2021 г.

2. Руководитель работы к.т.н., профессор кафедры «МО и ПЭВМ» Шашков Б.Д.

3. Задание на работу (назначение разработки, исходные данные и т.п.)

1) Назначение: мобильное электронное приложение медицинского справочника предназначено для базового изучения и освоения нормальной анатомии человека.

2) Среда выполнения: операционная система Windows, начиная с версии Windows 7.

Средства разработки: язык программирования Java.

Инструментальные средства: Microsoft Visual Studio, Android Studio , DB Browser SQLite.

Технологии разработки: ООП, UML.

3) Основные функции мобильного приложения «медицинский справочник»:

а) вход в приложение;

б) просмотр информации об анатомии человека;

в) просмотр аномалий развития;

г) просмотр фактов из разных областей жизни и развития человека;

д) просмотр терминологического словаря.

4. Перечень подлежащих разработке вопросов

Анализ предметной области

Анализ требований

Проектирование

Реализация

Контроль качества ПО:
– Тестирование
– Анализ качества кода

План разработки и оценка бюджета

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований	10	14.04.2021 – 25.04.2021	
2	Проектирование	20	26.04.2021 – 02.05.2021	
3	Реализация	40	03.05.2021 – 18.05.2021	
4	Контроль качества	15	19.05.2021 – 24.05.2021	
5	План разработки бюджета	5	25.05.2021 – 03.06.2021	
6	Оформление пояснительной записки	10	04.06.2021 – 14.06.2021	

Дата выдачи задания « » _____ 20___ г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан факультета _____
(подпись, дата)

Реферат

Пояснительная записка содержит 82 листа, 26 рисунков, 7 таблиц, 17 использованных источников и 2 приложения.

Приложение для ознакомления с анатомией человека спроектировано и реализовано с помощью актуальных средств разработки.

Цель разработки: изучить предметную область и провести её анализ, рассмотреть конкурентные продукты, выбрать основные средства проектирования и разработки, разработать логическую и физическую модель базы данных, спроектировать пользовательский интерфейс, рассчитать экономическую выгоду, а также реализовать и протестировать серверное и клиентское приложение для информационной системы.

Результаты разработки: в процессе разработки была исследована и проанализирована предметная область образовательных приложений по анатомии, проанализированы конкурентные приложения, выбраны основные средства проектирования и разработки, разработаны логическая и физическая модель базы данных, спроектирован пользовательский интерфейс, рассчитана экономическая выгода, а также реализована и протестирована серверная и клиентская часть приложения для информационной системы.

Средства разработки: ERWINDataModeler, Android Studio, DBSQLite.

Содержание

1 Анализ технического задания 10

2 Анализ предметной области и постановка задачи 12

2.1 Развитие информационных технологий 12

2.2 Современные решения 14

2.3 Обзор аналогов разрабатываемой системы 15

2.4 Создание прототипа системы 17

3 Проектирование программного обеспечения 19

3.1 Обоснование средств разработки 19

3.2 Проектирование системы 21

3.3 Анализ прецедентов 25

3.4 Диаграмма состояний 27

3.5 Диаграмма деятельности 28

3.6 Диаграмма последовательности 29

3.7 Диаграмма развертывания 29

4 Проектирование пользовательского интерфейса 31

5 Планирование разработки и оценка бюджета 40

6 Тестирование программного обеспечения 46

6.1 Цели и виды тестирования 46

6.2 Конфигурационное тестирование под разнообразные модели 48

6.3 Тестирование методом функциональных диаграмм 50

Заключение 54

Список использованных источников 55

Приложение А. Глоссарий 57

Приложение Б. Листинг программы 59

Введение

Медицина – система научных знаний и практической деятельности, целями которой являются укрепление и сохранение здоровья, продление жизни людей, предупреждение и лечение болезней человека [1]. Понятие зародилось еще в первобытные времена. Об этом мы узнаем из письменных источников, найденных во время археологических раскопок, данных антропологии и результатов многолетних исследований. Мумифицированные останки первобытных людей зачастую могут дать информацию о том, какие болезни переносились, что могло послужить причиной смерти и какую медицинскую помощь они получали при жизни.

Основными центрами древней медицины были Египет, Индия, Китай и Греция. Именно там образовалась медицина, которая очень напоминает современную. Первоначально возникновение заболевания рассматривалось как некое внешнее и совершенно враждебное человеку живое существо, которое каким-либо образом проникает в его тело и способно вызвать болезненное состояние. Принципы лечения были примитивны в связи со множеством факторов: нехватка знаний, отсутствие каких-либо технологий и опыта.

Постепенно, с появлением технологий, увеличивались и знания людей, и, следовательно, варианты лечения. В средние века у ацтеков существовало уже около нескольких сот терминов, использовавшихся для обозначения частей человеческого тела. В те времена, многие способы лечения были тесно связаны с таким понятием, как «магия», но на деле все это основывалось на многовековом эмпирическом опыте человечества. Активно изучались растения, их лекарственные и целебные свойства. Появились такие науки, как анатомия, гистология и физиология. Умы того времени начали совершать грандиозные открытия, повлиявшие на дальнейшее развитие медицины. Одним из важнейших было открытие Гарвеем кровообращения, явления дыхания и многих других, которые напрямую повлияли на развитие медицины.

В XIX – XX века медицина приблизилась к естествознанию. Активно начали использоваться знания из химии, биологии и физики в различных направлениях – от определения болезней до понимания явлений, происходящих в организме человека.

В это время многие отечественные ученые совершили величайшие открытия. Одним из них был хирург Николай Иванович Пирогов, который создал топографическую анатомию и был основателем военно-полевой хирургии. Илья Ильич Мечников считается основателем теории иммунитета. Он доказал существование в организме клеток, способных поглощать патогенные макроорганизмы. Все это, с наступлением научно – технического прогресса, дало резкий скачок в сторону активного развития всех отраслей медицины.

Основная и самая массовая волна компьютеризации медицины проходила во второй половине XX века. Тогда технологии позволили упорядочить работу с поставщиками и наладить работу персонала внутри медицинских учреждений. Все это дало возможность создать медицинскую инфраструктуру, которая и сейчас активно используется во многих странах мира.

В наше время модернизация здравоохранения является одним из основных и приоритетных направлений долгосрочного развития страны. Основной акцент направлен на повышение технологического уровня, внедрение новых стандартов лечения и профилактики, а также повышение квалификации медицинских работников. Понятие медицины сегодня тесно связано с IT-технологиями. Все чаще и чаще в разнообразных направлениях медицины появляются новые системы, инструменты.

Как показывает практика, чаще всего, IT-решения в медицине необходимы в таких направлениях, как диагностика заболеваний, поддержке медицинских исследований, снижении нагрузки на специалистов и онлайн-образовании. Например, сейчас, в период пандемии коронавируса, IT-технологии необходимы, как никогда. Это и телемедицина. В ней по сути

нет ничего нового. Последние 100 лет тем или иным способом телекоммуникации (телеграф, телефон, видеосвязь, факсимильная передача данных, компьютерные сети, интернет) применялись и применяются в медицинских целях. Телемедицина – инструмент здравоохранения, представляющий собой использование цифровых информационных и телекоммуникационных технологий для дистанционного предоставления медицинской помощи и услуг.

Очень важно медицинское онлайн-образование. Чем выше уровень врача, чем больше он осведомлен о последних достижениях, тем легче ему справедливо оценить действенность клинических решений, сделанных на основе новейших разработок. Теперь, когда в обиход врача вошли такие понятия, как вебинары, онлайн-трансляции, видеолекции, курсы дистанционного образования, словари нет нужды тратить время и ресурсы на частые поездки на конгрессы для повышения своего уровня. Достаточно иметь лишь доступ в интернет и телефон. То же самое и со студентами медицинских вузов. Сейчас, с активным использованием смартфонов, у обучающегося всегда под рукой весь необходимый ему материал.

На сегодняшний день мир мобильных приложений кардинально изменился. Многие аппараты могут без проблем заменять стационарные ПК, не ограничивая пользователя. Сегодня, мобильное приложение – это специально разработанное под функциональные возможности телефонов программное обеспечение. Это может быть что угодно – от онлайн помощников до крупных сервисов.

Рынок мобильных приложений очень развит и неуклонно растет. Согласно прогнозам Statista, в 2020 году валовой годовой доход в отрасли мобильных приложений превысит $189 млрд.

Был произведен анализ области мобильных приложений, изучены основные, наиболее популярные медицинские справочники. В результате было выявлено, что существует не так много действительно полезных справочников по тому или иному профильному предмету. Некоторые

приложения носят информационный характер, другие – выдают подробные инструкции по использованию конкретных препаратов в борьбе с определенным заболеванием. В связи с этим, было решено создать удобный в использовании медицинский справочник для быстрого доступа к информации по нормальной анатомии человека.

1 Анализ технического задания

Техническое задание, или ТЗ – это документ, в котором фиксируются требования к проекту [2]. Техническое задание используется в огромном количестве областей, таких как машиностроение, разнообразное производство, бизнес и т.д. В данном документе фиксируются определенные требования к проекту и отношения заказчика и исполнителя, поскольку в нем прописаны следующие пункты:

- основное назначение разрабатываемого объекта;
- показатели качества;
- технические характеристики;
- специфические требования;
- технические требования;
- экономические требования;
- сроки выполнения работ.

Разработкой технического задания занимается исполнитель, чаще всего, во время беседы с самим заказчиком. Происходит обсуждение грядущей работы и учет всех требований, чтобы добиться максимального результата, который будет в полной мере устраивать обе стороны. Но нередко бывает и то, что все происходит наоборот. В таких ситуациях часто встречается недопонимание исполнителя и заказчика, в связи с тем, что второй не имеет достаточных знаний по требуемой области и не может подробно описать задание. Тогда на помощь приходит техписатель, который нанимается заказчиком.

Согласование и утверждение дополнений к ТЗ проводят в установленном порядке. Изменения и дополнения, не нарушающие условия выполнения обязательных требований, при согласии заказчика и разработчика могут быть внесены на любом этапе разработки, но не после представления изделия на приемо-сдаточные испытания. Регистрация, учет и

хранение ТЗ и дополнений к нему проводят в соответствии с требованиями ГОСТ 2.501-88 (ЕСКД. Правила учета и хранения).

В качестве ТЗ может быть использован и другой документ, содержащий достаточные требования для разработки продукции и одобряемый заказчиком и разработчиком. В случае инициативной разработки продукции ТЗ (или заменяющий его документ), базируется на результатах исследования рынка продукции.

Не допускается включать в ТЗ требования, которые противоречат законам Российской Федерации.

К сожалению, на практике, нередким случаем бывает то, что потребитель, после выполнения работ, требует от исполнителя изменения или добавления определенных функций или какого-либо рода иные изменения, которые ранее не обсуждались. Или, напротив, сам исполнитель некачественно выполнил свою работу, не соблюдая все пункты полученного им задания. Именно поэтому, техническое задание – это, прежде всего, защита прав потребителя и исполнителя.

Разрабатываемая система должна предусматривать следующие функциональные возможности:

– дружественный интерфейс;
– хранение текстовой информации;
– авторизация пользователя.

Техническое задание – это юридический документ, на основании которого каждая из сторон имеет полное право обратиться в суд, если какой-либо пункт не был корректно выполнен или не выполнен вовсе.

2 Анализ предметной области и постановка задачи

2.1 Развитие информационных технологий

Развитие IT-технологий и распространение мобильных устройств дали толчок к формированию нового рынка – рынка мобильных приложений**.** Растет количество разработчиков, увеличивается количество приложений, а также все больше компаний понимают необходимость создания собственных приложений для повышения конкурентоспособности. Для активного распространения приложения пользователям, необходимо использовать определенные платформы и реклама.

На данный момент существует лишь небольшое количество медицинских приложений справочного характера. Был изучен рынок и сделан вывод – существует лишь несколько действительно полезных приложений, напрямую связанных с профессиональной деятельностью. Но, к сожалению, сейчас на рынке крайне трудно найти действительно полезное, справочное приложений для обучающихся в школе, студентов, или пользователей, которые хотят освоить что-то новое и познакомиться с основами анатомии человека.

Мобильные приложения пишутся на различных языках программирования (Java; PHP/JavaScript; ActionScript; Swift и Objective-C). В процессе создания приложение проходит несколько этапов:

– определить, для чего нужно приложение и какие задачи оно будет выполнять;
– проектирование и дизайн;
– разработка;
– тестирование;
– мониторинг.

Мобильное приложение можно разделить на два больших блока – это frontend и backend. Соответственно, в часть frontend входят компоненты и

опции программы, с которой взаимодействует пользователь. Основными из них являются:

– панель выбора;
– информационная панель;
– настройки опций.

Backend – это скрытая часть, «задник». С этими компонентами взаимодействует разработчик посредством серверного софта. Иными словами, мобильное приложение напоминает сплит-систему, в которой одна часть находится на стороне пользователя – это frontend, а другая на стороне разработчика – это backend [3].

На сегодняшний день существует множество приложений, направленных на изучение или освоение той или иной области нашей жизни – для ботаников созданы приложения-книги с огромнейшим числом растений, их описанием и значением, для программистов – множество IT-приложений с разнообразием языков программирования для их базового изучения и множество других. Для любой области из жизни можно найти полезное мобильное приложение.

Основными задачами научного исследования являются:

– анализ основных типов мобильных приложений;
– анализ конкурирующего ПО и рынка;
– анализ функциональных требований;
– выбор средств разработки.

Целью дипломной работы является выявление необходимости данного приложения для определенного круга лиц, непосредственно связанного с тематикой работы, а также поиск и анализ аналогичных приложений для выделения самых важных качеств и характеристик каждого из них, с целью в дальнейшем реализовать это для благоприятной и комфортной работы пользователей.

2.2 Современные решения

Существует два различных способа технической разработки и реализации проектов для мобильных устройств: мобильный web-сайт и мобильное приложение. На первый взгляд, оба способа выглядят очень похожими, но, на деле, они отличаются и каждый предлагает что-то свое.

Мобильный сайт работает только через браузер. В отличие от обычных web-сайтов, он разработан специально для мобильных устройств. Его преимущества это:

– совместимость – удобен во время работы с различными типами смартфонов и планшетов;

– более широкий охват целевой аудитории;

– поддержка и обслуживание – web-сайты позволяют гибко вносить изменения.

Ограничения:

– удобство – в отличие от приложения, адаптивный web-сайт не может эффективно использовать все функции смартфона;

– размер экрана устройства – из-за небольших размеров экраны смартфонов/планшетов отображают намного меньше контента по сравнению с монитором настольного ПК или экраном ноутбука;

– автономный доступ – автономный режим функционирования мобильного сайта возможен только при использовании кэшированных страниц.

В отличие от адаптивных мобильных web-сайтов, работающих через браузеры, приложения должны быть загружены с определенных порталов, таких как Google Play Market, App Store или других. Мобильные программы разрабатываются отдельно для каждой операционной системы.

Плюсы:

– удобство – анализ показывает, что приложения более популярны, чем аналогичные web-сайты, поскольку более удобны;

– персонализация – мобильные программы являются отличным решением для служб, которые требуют регулярного использования и позволяют пользователям создавать личные учетные записи, а также хранить важную информацию под рукой;

– работа в автономном режиме – поскольку приложения требуют установки, они могут предоставлять доступ к своим функциям и контенту даже без подключения к Интернету.

Минусы:

– совместимость – для каждой платформы (iOS, Android, Windows) нужна отдельная рабочая версия программы;

– поддержка и обслуживание – тогда приложение разрабатывается для нескольких различных платформ, его поддержка требует больше времени.

Несмотря на все плюсы адаптивного web-дизайна, мобильные приложения кажутся более популярными. И действительно, в последнее время появилось множество приложений, заменяющих web-сайты и позволяющие работать без ПК. Такими можно назвать разнообразные мессенджеры и электронные журналы. В связи с вышеперечисленными фактами, для разработки было выбрано мобильное приложение под систему Android.

2.3 Обзор аналогов разрабатываемой системы

Развитие технологий не обошло стороной и медицину – существует достаточно приложений на медицинскую тематику. Среди них выделяют: игры, справочники, методические указания и тесты. Однако, из всего вышеперечисленного на рынке преобладают приложения развлекательного характера.

Для подробного и детального сравнения аналогичных продуктов, воспользуемся специальным Android-приложением, используемым для поиска и установки необходимого ПО на телефон – Play Market. Данная

площадка бесплатна и доступна абсолютно каждому пользователю смартфона с операционной системой Android. Помимо потенциального поиска приложений и их последующей установки, Play Market позволяет разработчикам выходить на мировой рынок мобильных приложений и продвигать свою разработку. На данной площадке можно найти практически все – от классических мобильных игр и программ для занятия спортом до разнообразных web-приложений, календарей, развивающих игр, журналов или познавательных приложений.

Целью поиска были разнообразные справочники, напрямую связанные с медициной и ее областями. После детального отбора, было выяснено, что на рынке мобильных приложений проблематично найти непосредственно какого-либо рода анатомический справочник. В связи с этим, были выделены три основных приложения, которые максимально приближенно направлены на изучение анатомии человека: «Нормальная анатомия человека», «Тест по анатомии – Анатомия человека» и «Справочник врача».

Приложение «Нормальная анатомия человека» разработано компанией «Фармакоша» и направлено на изучение анатомии человека. Оно является достаточно сложным в использовании, предоставляет научную информацию и предназначено исключительно для людей, обладающих медицинской базой.

Приложение «Тест по анатомии – Анатомия человека» представлено от разработчика Student Doctor. Приложение включает в себя более 500 разнообразных тестов по анатомии с пояснениями после их прохождения. Оно направлено исключительно на проверку уже полученных знаний, поэтому не содержит ознакомительной, научной информации. После изучения, был обнаружен один большой недостаток – недостоверность в пояснениях к некоторым вопросам.

Приложение «Справочник врача» на площадке Play Market представлено компанией «ООО Медицинские информационные решения». Это одна из самых популярных медицинских IT-компаний, которая

находится на рынке уже более 8 лет. Данная организация занимается разнообразными изучениями и исследованиями медицинских областей, является участником проекта «Сколково» и предоставляет мобильные приложения для поддержки как специалистов, так и пациентов. Главной особенностью приложения «Справочник врача» является профессиональная помощь врачам – присутствуют разнообразные калькуляторы, стандарты лечения, лекарственные препараты и многое другое. Приложение направлено на помощь квалифицированным специалистам и упрощение их работы в медицинских учреждениях.

Таблица 1 – Сравнительный анализ

Наименование	Наличие научной информации	Интуитивно понятный интерфейс	Достоверность информации	Возможность offline-использования	Отсутствие рекламы	Комфортное пользование
Создаваемое приложение	Да	Да	Да	Да	Да	Да
Нормальная анатомия человека	Да	Нет	Да	Нет	Нет	Да
Тест по анатомии – анатомия человека	Нет	Да	Нет	Да	Нет	Нет
Справочник врача	Да	Да	Да	Нет	Да	Да

Рассмотренные продукты в полной мере не предоставляют удобный доступ пользователю. При разработке были учтены недостатки представленных на рынке аналогов.

2.4 Создание прототипа системы

Прототипирование – это процесс быстрого создания приблизительной модели проекта, для того чтобы скорее показать заказчику, клиенту или пользователю [8]. Идея, которую можно увидеть и иногда потрогать.

Прототипирование является одним из этапов проектирования и позволяет проверить на практике пригодность различных конструкторских решений.

Главной идеей и целью данного этапа разработки является демонстрация заказчику своего взгляда на создание того или иного продукта. Прототипирование позволяет сделать это с наименьшим количеством затраченных сил, времени и денег.

Прототип создается в несколько этапов:

– создание эскиза;
– обсуждение;
– создание прототипа;
– тестирование;
– улучшение прототипа.

Создание эскиза – главной задачей данного этапа является формулировка и описание всех идей, которые приходят на ум. Чем больше вариантов сгенерирует мозг, тем лучше.

Обсуждение – необходимо путем диалога с кем-либо отобрать все лучшие идеи.

Создание прототипа – углубленная проработка деталей проекта и появление его в свет.

Тестирование – прототип можно протестировать не только на заказчике, но и на любом другом человеке для того, чтобы выявить некоторые ошибки прототипа перед непосредственной презентацией его клиентам. Во время тестирования прототипа с клиентами решаются различные организационные моменты, заказчик может выразить свои претензии к шаблону продукта, а так же дать множество различных новых идей касательно разрабатываемого программного продукта.

Улучшение прототипа – с учетом тестирования прототипа на клиентах необходимо будет сделать прототип более совершенным.

3 Проектирование программного обеспечения

3.1 Обоснование средств разработки

Для реализации было решено использовать Android Studio с языком программирования Java и встраиваемую СУБД SQLite.

Java – объектно-ориентированный язык программирования, основной особенностью которого является виртуальная машина Java Virtual Machine, которая переводит Java-код, понятный человеку, в код, который будет понятен процессору [4]. Эта возможность подразумевает то, что Java-код можно запустить, где удобно.

SQLite – это C-библиотека, встраиваемая кроссплатформенная СУБД. Данные понятия означают, что ПО не использует модель «клиент-сервер», то есть сама подсистема хранения SQLite не является отдельным работающим процессом, с которым взаимодействует программа [5]. Данный подход упрощает временные расходы на выполнение той или иной операции, уменьшает время отклика и, вместе с этим, значительно упрощает программу. SQLite хранит всю информацию о таблице, о ее полях, индексах, данных и определениях. Все данные хранятся в одном единственном файле на компьютере, где, в данный момент времени, исполняется программа. Сервера нет, ведь само приложение является сервером. Доступ к данным, хранящимся в БД, происходит посредством «подключений», которые открываются через вызов функции DLL. При открытии просто указывается имя файла базы данных. В случае, если конкретный файл не найден, то происходит создание и сохранение нового.

SQLite – это реляционная СУБД, и, по сравнению со своими аналогами, такими как MySQL и PostgreSQL, имеет как ряд преимуществ, так и ряд недостатков.

Основным недостатком, наряду со своими более мощными конкурентами, является полное отсутствие пользовательского управления. К примеру, MySQL предоставляет очень обширный функционал и в разы

больший список поддерживаемых типов файлов. SQLite – же, напротив, поддерживает лишь небольшой список типов данных, таких как null, integer, real, text и blob. В связи с этим, SQLite стоит применять, например, в случаях, когда используются какие-либо однопользовательские приложения. В приложениях, где доступ к БД будет осуществляться одновременно для нескольких пользователей, лучше использовать полнофункциональную реляционную СУБД, например MySQL.

Возможности SQLite очень обширны. Основными из них являются:

– открытость;

– наличие системы отката, защищающая целостность данных;

– бесплатное пользование без ограничений;

– отсутствие каких-либо потребностей в администрировании;

– высокая производительность при достаточном объеме оперативной памяти.

Помимо всего вышеперечисленного, в SQLite есть некоторые свои особенности:

– каждая из записей в таблице содержит свой уникальный столбец rowed, равный 64-битному номеру;

– легко переносить, так как, по умолчанию, база данных является одним файлом;

– в любой столбец можно занести любое значение, в связи с тем, что тип столбца не определяет тип хранимого значения в текущем поле.

Для работы с SQLite используется DBBrowser – браузер баз данных для SQLite. Он представляет собой инструмент для создания, разработки, редактирования файлов БД.

Android Studio – интегрированная среда разработки для работы с платформой Android, находящая в свободном доступе. Она основана на программном обеспечении Intellij IDEA – интегрированной среде разработки ПО для разнообразных языков программирования, таких как Python, Java, JavaScript [6].

Основными возможностями Android Studio являются:

– сборка приложений, основанная на Gradle (система автоматической сборки);

– рефакторинг кода, т.е. переработка кода;

– шаблоны основных макетов и компонентов Android.

Среда Android Studio предназначена как для небольших команд разработчиков, так и для крупных организаций. Опытные разработчики имеют возможность выбирать те инструменты, которые больше подходят для масштабных проектов [7]. Решения для Android разрабатываются в Android Studio с использованием таких языков программирования, как C++ и Java. В основе рабочего процесса Android Studio заложен концепт непрерывной интеграции, т.е. наличие возможности сразу обнаруживать имеющиеся проблемы. Достаточно долгая проверка кода позволяет выполнять эффективную обратную связи с разработчиками и быстрее опубликовать версию мобильного приложения в Google Play.

3.2 Проектирование системы

Для хранения информации используется SQLite. SQLite хранит в себе всю информацию о таблице, обо всех ее полях, индексах, данных и определениях. Все данные хранятся в одном единственном файле на компьютере, где, в данный момент времени, исполняется программа. Как такового понятия «сервера» нет, так как само приложение является сервером. Доступ к данным, хранящимся в БД, происходит посредством «подключений», которые открываются через вызов функции DLL-библиотеки, содержащей код и данные. Для доступа к базе необходимо просто указывать имя файла нужной базы данных. В том случае, если конкретный, искомый файл, не найден, то происходит создание нового и его сохранение.

На рисунке 1 представлена структура базы данных.

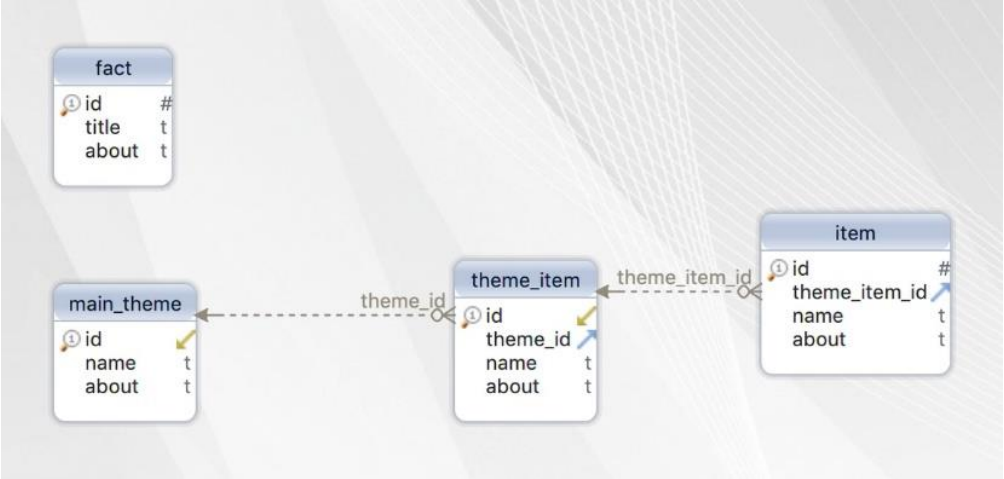


Рисунок 1 – Структура базы данных

Существует основная тема – main_theme. Это главное меню приложения со списком всех тем. На main_theme ссылается пункт theme_item, в котором хранится первоначальная информация об этапе, выбранном на предыдущем шаге. Завершает эту цепочку пункт item – он ссылается на theme_item и несет в себе подробную информацию о конкретном, выбранном объекте.

Помимо этого, существует объект fact, отвечающий за факты, который является отдельной сущностью, модулем.

Вся работа с данными происходит посредством DB Browser SQLite – браузер баз данных для SQLite. Он представляет собой инструмент для создания, разработки, редактирования файлов БД.

Основные возможности DB Browser это:

– создавать файлы базы данных;

– работать с таблицами, индексами и записями, а именно создавать, изменять, просматривать и удалять;

– выполнять SQL – запросы.

В DB Browser SQLite были созданы 5 таблиц, каждая из которых хранит информацию из разных активностей приложения. Данные таблицы представлены на рисунке 2.

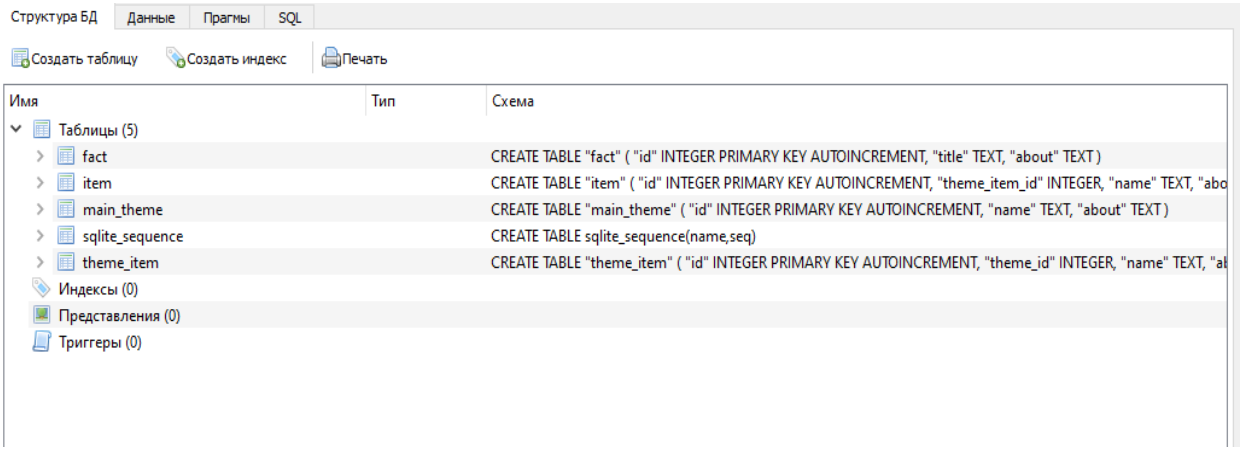


Рисунок 2 – Структура базы данных

У каждой таблицы существуют свой набор данных, которые необходимы для взаимодействия друг с другом. Такими являются title, name, about. Основным считается идентификатор – id, так как база данных используется в Android. Структура таблиц представлена на рисунке 3.

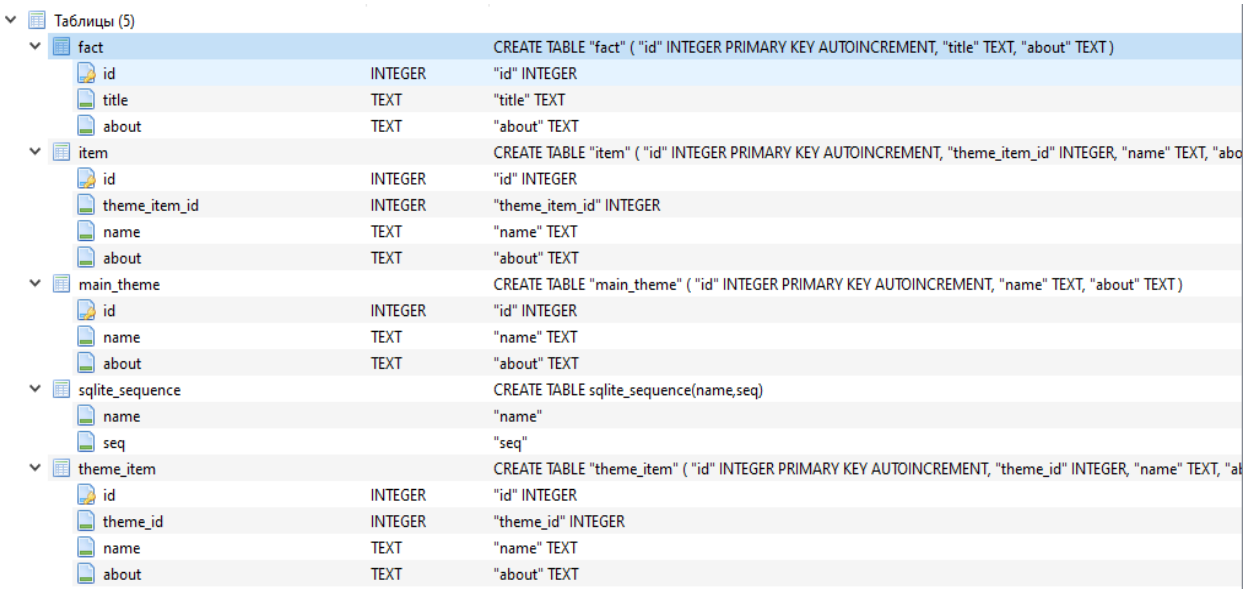


Рисунок 3 – Структура таблиц

В результате создаются несколько взаимосвязанных таблиц, отвечающих за ту или иную часть хранящихся данных, представленные на рисунке 4.

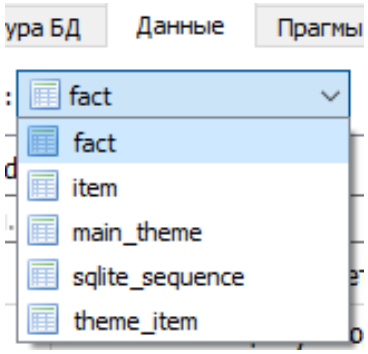


Рисунок 4 – Данные базы данных

После выбора определенного пункта, открывается список с данными, которые можно модернизировать – добавить, изменить или удалить. Здесь предложен довольно простой табличный интерфейс, исключающий сложности SQL – запросов. Основное преимущество – простой и быстрый поиск нужного фрагмента текста. Пример – таблица данных «fact», представлена на рисунке 5.

Таблица: fact

Filter in all columns

	id	title	about
	Фи...	Фильтр	Фильтр
1	1	Красные кровяные клетки	Постоянно в теле человека находится 2,5 ...
2	2	Отпечатки пальцев у плода	Плод приобретает отпечатки пальцев еще в ...
3	3	Борода	Волосы бороды — наиболее быстрорастущи...
4	4	Манту нельзя мочить?	Внутрикожную туберкулиновую пробу, или ...
5	5	Железо	Тело содержит достаточно железа, чтобы ...
6	6	Кровь	Организм человека имеет примерно 5,6 ...
7	7	Дюйм руки	Квадратный дюйм руки содержит почти 3 ...
8	8	Выработка клеток	Все органы и ткани тела ежесекундно ...
9	9	Скорость воздуха при чихании	Во время чихания воздух движется со ...
10	10	24 часа	За 24 часа кровь в организме проходит в ...
11	11	Укус паука	При укусе ядовитого паука многие раны ...
12	12	Нервы руки	Человеческая рука содержит три основных ...
13	13	Мышцы лица	У людей 22 мышцы на каждой стороне лица....
14	14	Перекачка крови	Во время перекачки крови сердце создает ...
15	15	Зеркальное отражение внутренних органов	Примерно один из десяти тысяч человек ...
16	16	Выращивание кожи	Современные технологии позволяют ...
17	17	Перекись водорода	Для лечения небольших поверхностных ран ...
18	18	Анестезия древних греков	Способность электрических скатов вызывать...
19	19	Безымянный палец	Кровь чаще всего берут из безымянного ...
20	20	"Синдром белого халата"	У значительной части пациентов с ...
21	21	Электрокардиограф из игрушки	Первый вариант конструкции ...
22	22	Муравьи-хирурги	Жвалы обитающих в Африке муравьёв рода ...
23	23	Срок годности лекарств	Согласно исследованиям американского ...
24	24	Сам себе хирург	29 апреля 1961 года врач шестой советской ...
25	25	Аппендикс	В прошлом веке в некоторых странах у всех ...

Рисунок 5 – Таблица фактов

В данной таблице хранится вся информации о фактах – их название и полное описание. Пользователю предоставляется полный список, в котором он уже самостоятельно может выбрать интересующий его интересный факт и ознакомиться с ним.

3.3 Анализ прецедентов

Для определения функциональности системы строятся диаграммы вариантов использования или диаграммы прецедентов. Диаграммы вариантов использования описывают взаимоотношения и зависимости между группами вариантов использования и действующих лиц, участвующими в процессе [9].

Диаграммы вариантов использования не предназначены для отображения самого проекта и не могут как-либо описать внутреннее устройство системы. Они предназначены для описания системы, для взаимодействия с будущими пользователями и определения каких-либо характеристик самой системы.

На диаграммах этого вида изображают актеров системы и варианты использования, с которыми они взаимодействуют. Построенная для разрабатываемого приложения диаграмма представлена на рисунке 6.

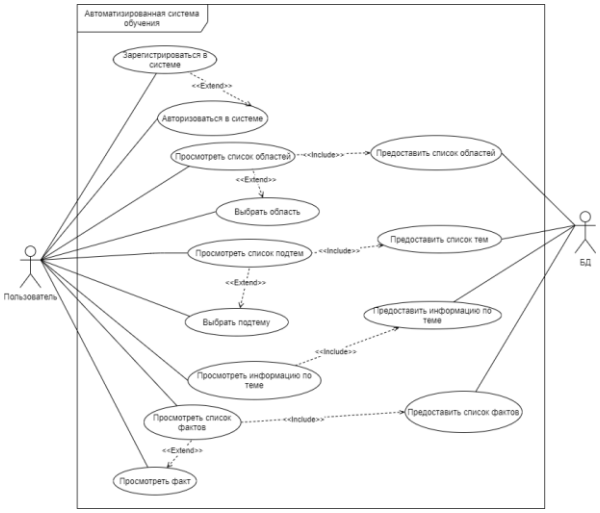


Рисунок 6 – Диаграмма вариантов использования

Описание актеров и всех вариантов использования представлены в таблице 2.

Таблица 2 – Описание актеров и вариантов использования

Актер	Краткое описание	Цель	Краткое описание
Пользователь	Потенциальный пользователь системы	Зарегистрироваться в системе	Пользователь может завести учетную запись
		Авторизоваться в системе	При наличии учетной записи Пользователь может авторизоваться в системе
		Просмотреть список областей	Пользователь может посмотреть все области, информация по которым хранится в системе
		Выбрать область	Пользователь выбирает область, информацию по которой он хочет получить
		Посмотреть список подтем	Пользователь может получить список подтем выбранной области
		Выбрать подтему	Пользователь вбирает подтему, информацию по которой он хочет получить
		Посмотреть информацию по теме	Пользователь просматривает информацию по выбранной теме
		Просмотреть список фактов	Пользователь может просмотреть список фактов
		Просмотреть факт	Пользователь может просмотреть информацию о конкретном факте
БД	База данных, в которой хранится справочная информация	Предоставить список областей	БД возвращает список хранящихся в ней областей
		Предоставить список подтем	БД возвращает список подтем выбранной пользователе области
		Предоставить информацию по теме	БД возвращает информацию по выбранной пользователем теме
		Предоставить список фактов	БД возвращает список фактов

В результате проведенного анализа были выявлены и описаны прецеденты системы.

3.4 Диаграмма состояний

Для определения возможных состояний системы строится диаграмма состояний. Каждый объект на этой диаграмме представляет собой набор атрибутов, выполняемых функций и имеет взаимосвязи с другими объектами. Состояние системы определяется значениями вышеперечисленных параметров в определенный момент времени ее работы. Состояние изменяется посредством наступления какого-либо события.

Диаграмма состояний, которая была построена для разработки данного приложения, изображена на рисунке 7.

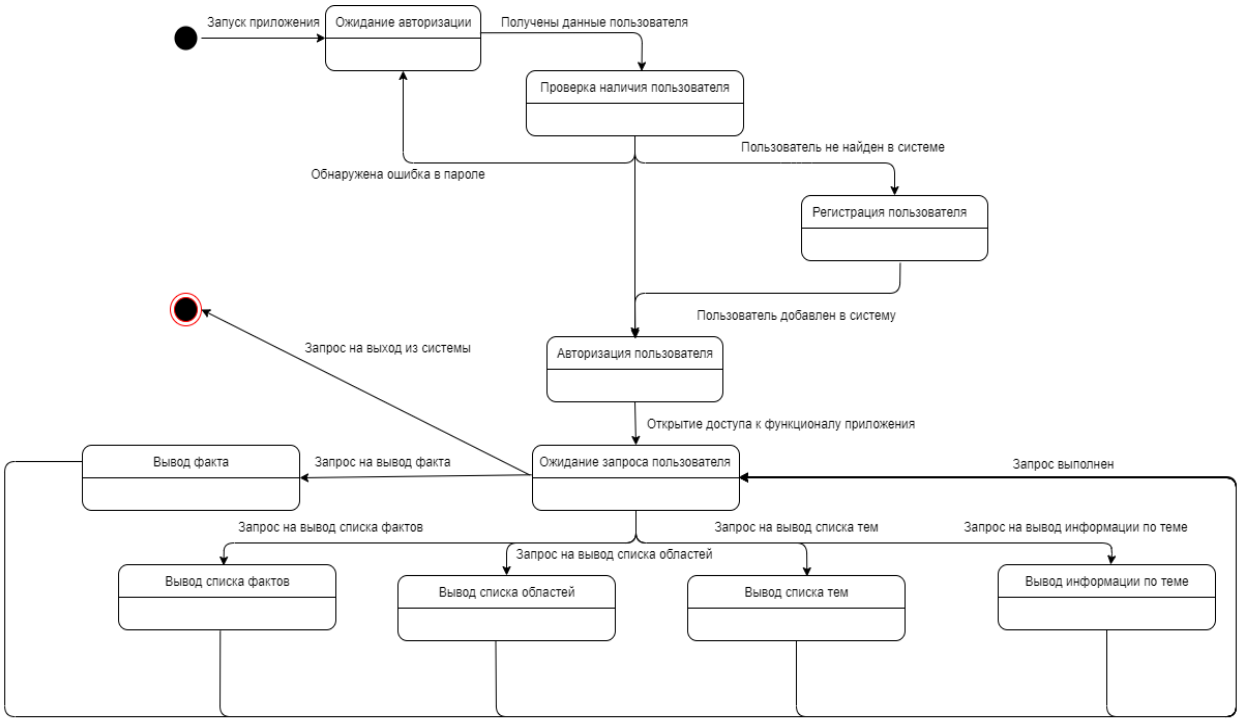


Рисунок 7 – Диаграмма состояний

Данная диаграмма отображает взаимосвязи между состояниями системы и условия их переходов. Были выделены следующие состояния: ожидание авторизации, проверка наличия пользователя, регистрация пользователя, авторизация пользователя, ожидание запроса пользователя, а

так же выводы такой информации, как факты и их списки, списки областей и тем с информацией.

3.5 Диаграмма деятельности

Для определения составных частей какой-либо деятельности используется диаграмма деятельности. При этом под деятельностью понимают спецификацию исполняемого поведения в виде координированного последовательного и параллельного выполнения подчиненных элементов – вложенных видов деятельности и отдельных действий, соединенных между собой потоками, которые идут от выходов одного узла к входам другого. Диаграмма деятельности полезна для описания алгоритма действий, но она не дает представления о поведении определенного объекта в рамках отдельного варианта использования или системы в целом.

Приведем диаграмму деятельности для варианта использования «Получить список подтем». Диаграмма представлена на рисунке 8.

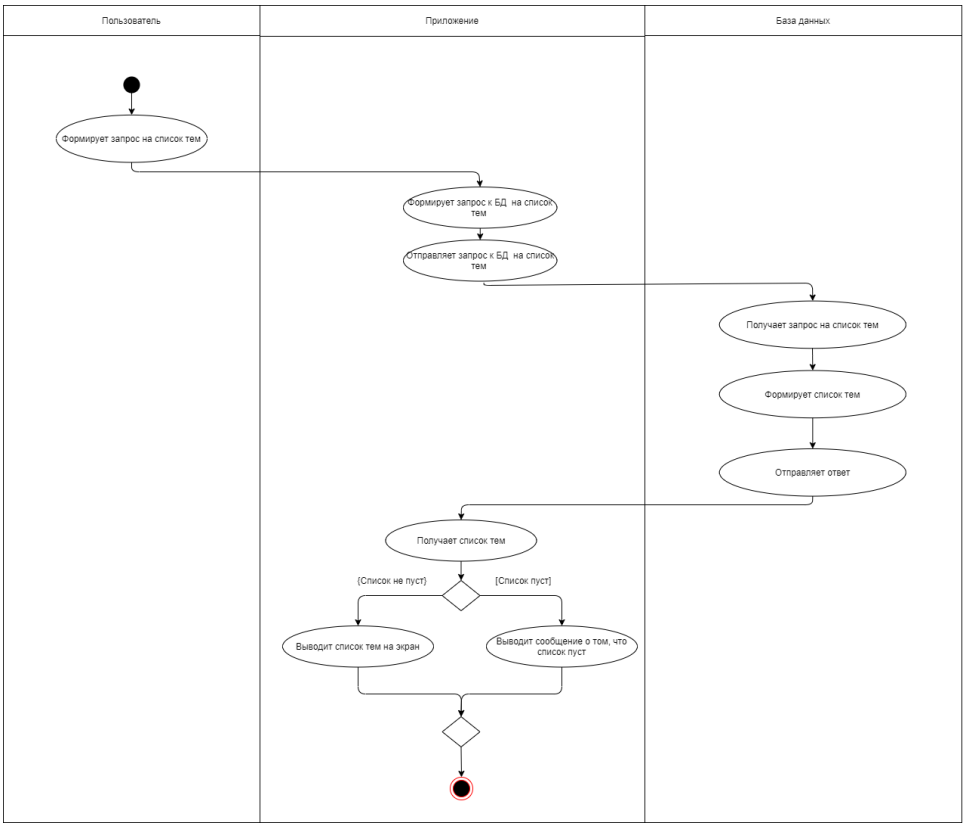


Рисунок 8 – Диаграмма деятельности

Сначала пользователь формирует запрос на получение списка подтем выбранной ранее области. После этого приложение формирует запрос к базе данных, которая его обрабатывает и возвращает результат. На основе этого результата система принимает решение – показать список тем или вывести сообщение об ошибке.

3.6 Диаграмма последовательности

Для описания сценария работы приложения используется диаграмма последовательности. Поскольку большая часть вариантов использования пользователя проходят по схожему сценарию, было принято решение построить обобщенную диаграмму последовательности для этих прецедентов, представленная на рисунке 9.

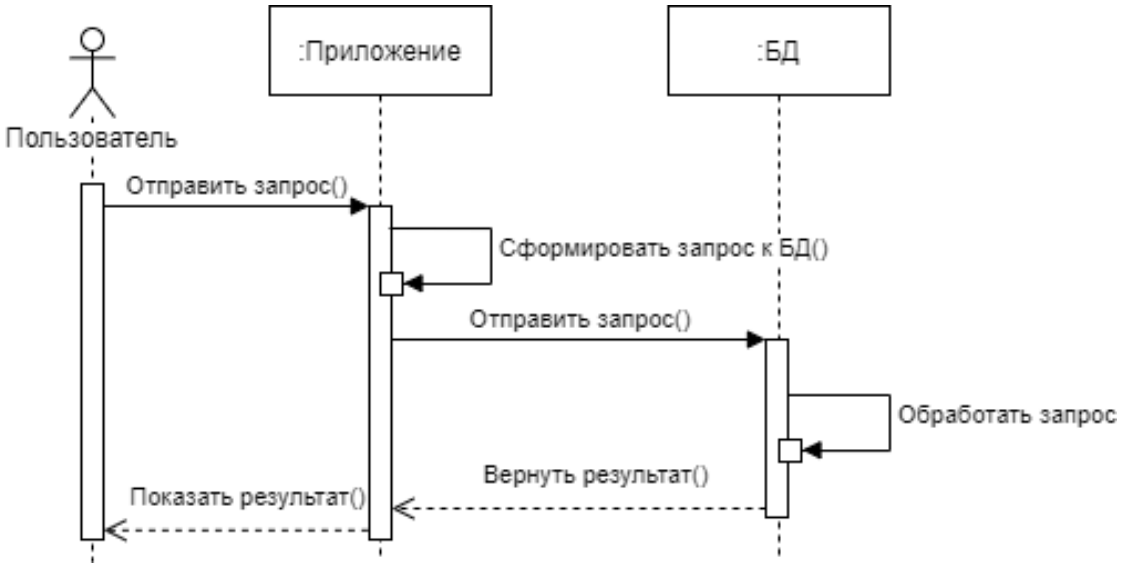


Рисунок 9 – Диаграмма последовательности

Все представленные на диаграмме объекты существуют на протяжении всего сеанса работы приложения.

3.7 Диаграмма развертывания

Диаграмма развертывания показывает, на каких устройствах и в каких средах будет исполняться готовое приложение. На рисунке 10 изображена диаграмма развертывания для разрабатываемого продукта.

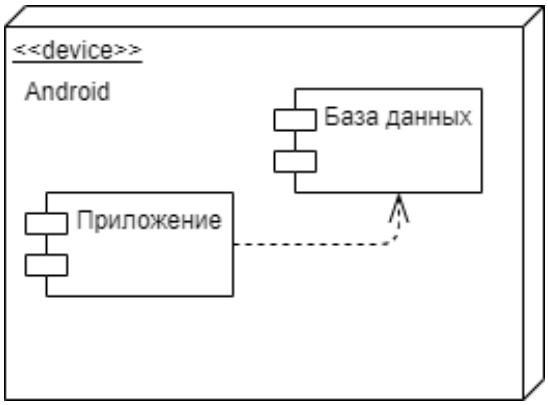


Рисунок 10 – Диаграмма развертывания

Из этой диаграммы видно, что приложение и используемая им БД будут находиться на одном Android-устройстве.

4 Проектирование пользовательского интерфейса

Пользовательский интерфейс – разновидность интерфейсов, в котором одна сторона представлена человеком (пользователем), другая – машиной или устройством. Представляет собой совокупность средств и методов, при помощи которых пользователь взаимодействует с различными, чаще всего сложными, машинами, устройствами и аппаратурой [11].

Так как уровень владения компьютерами у всех людей разный, а также зачастую пользователям приходится иметь дело с новым программным обеспечением, удобно при проектировании использовать дружественный интерфейс.

Дружественный интерфейс – это интерфейс, основанный на таких правилах (принципах), которые позволяют пользователю освоить работу с программой за минимальное время, и требующий от него (пользователя) минимальных усилий при решении его (пользователя) задач [12].

Пользовательские интерфейсы делятся на процедурно-ориентированные и объектно-ориентированные.

Процедурно-ориентированные интерфейсы примитивны, так как они организуют взаимодействие с пользователями только через консоль. Процедурно-ориентированные интерфейсы акцентируются на выполнении задач, а так же обеспечивают пользователя взаимодействием с объектами системы.

Объектно-ориентированные интерфейсы предоставляют пользователям возможность управления элементами предметной области.

К объектно-ориентированным интерфейсам относится графический интерфейс, с помощью которого пользователь может напрямую управлять объектами.

Основным элементом программного управления в объектно-ориентированном интерфейсе является меню, через которое и происходит взаимодействие пользователя и приложения.

Для удобного взаимодействия пользователя и приложения был разработан дружественный интерфейс.

Рассмотрим спроектированный интерфейс. При первом запуске приложения, пользователю представляется окно регистрации. Для пароля уставлены определенные требования:

– длина пароля должна быть не менее 8 символов;
– содержать хотя бы одну цифру;
– содержать хотя бы один нижний и верхний символ;
– содержать хотя бы один специальный символ.

Меню авторизации представлено на рисунке 11.

Рисунок 11 – Форма меню авторизации

Основные компоненты, представленные на рисунке 11:

1 – два компонента edittext, в которые пользователь должен ввести свои логин и пароль;

2 – компонент button, который пользователь должен нажать, чтобы войти в систему.

При попытке входа, система проверяет условия ввода пароля: если не выполнено хотя - бы одно из условий, выдается ошибка. Спроектированные для этого случая сообщения представлены на рисунке 12.

Рисунок 12 – Форма меню авторизации с ошибкой

Основные компоненты, представленные на рисунке 12:

1 – компонент toast, в котором указываются требования к паролю;

2 – компонент setError.

Компонент toast отвечает за всплывающие уведомления. В нашем случае это сообщение «Пароль недействителен», появляющееся, если введенный пароль не соответствует одному или более требованиям.

Компонент setError уведомляет пользователя о наличии ошибки.

После очередного ввода пароля вновь происходит его проверка на заданные к паролю требования. Если пароль подходит под все из условий, пользователь перенаправляется на главное меню приложения. Здесь ему становится доступен весь функционал медицинского справочника.

Главное меню представлено на рисунке 13.

Рисунок 13 – Главное меню

Оно представляет собой список тем, доступных пользователю для ознакомления, интересные факты и терминологический словарь.

Реализована функция сохранения данных пользователя – в верхнем правом углу установлена кнопка «Выйти». После первого входа, данные пользователя сохраняются и при всех последующий открытиях приложения, сразу будет доступно основное меню. При нажатии кнопки «Выйти», пользователя направляет в меню авторизации и происходит удаление последних введенных данных.

Интересные факты подразумевают список фактов, напрямую связанных с медициной. Каждое открытие приложения или выход из темы генерирует пять новых фактов в случайном порядке. Всего в базе данных их хранится 25 штук. На рисунке 14 представлены факты и их случайная генерация.

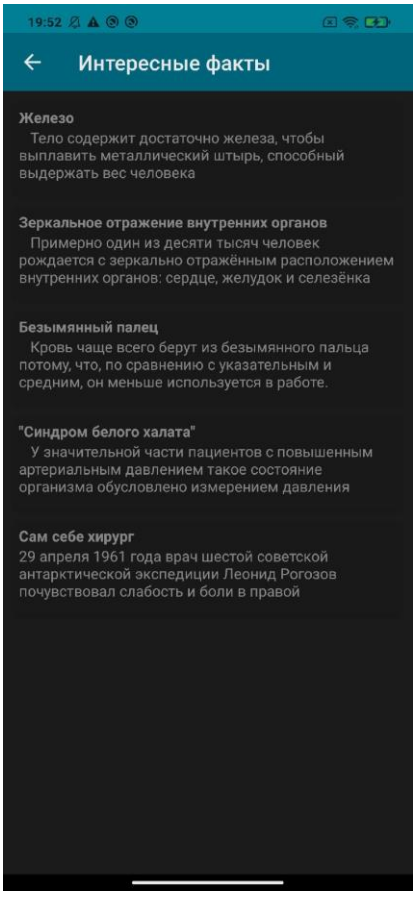


Рисунок 14 – Список фактов

Медицина – поистине интересная наука, хранящая в себе то, о чем многие и не могли догадываться. После изучения определенных научных

областей, было решено добавить в приложение раздел с разнообразными фактами, которые вполне могут удивить простого пользователя. Они были собраны не только по анатомии человека, но и по медицине в целом. Небольшие истории могут рассказать об удивительных особенностях организма человека, затрагивают события жизни некоторых выдающихся в мире медицины людей, раскрывают закономерности методов лечения. Что-то новое среди этих фактов сможет найти не только школьник или абитуриент, но и профессионалы своего дела. После обновления или перезапуска приложения, отображаются новые, сгенерированные факты. Обновленный список представлен на рисунке 15.

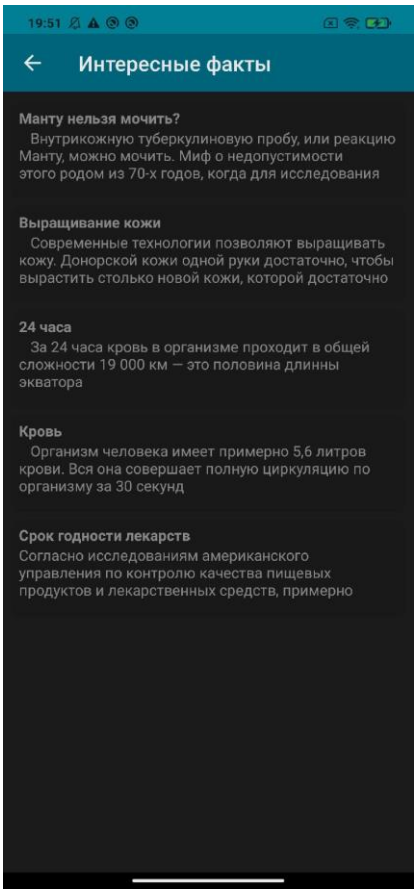


Рисунок 15 – Список новых фактов

Выбрав интересующий пункт, отобразится полная хранящаяся информация. Наглядное отображение одного из фактов представлено на рисунке 16.

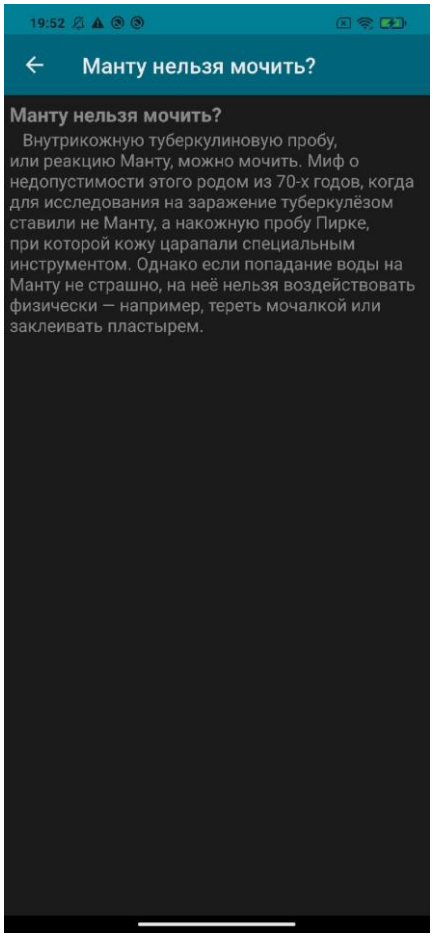


Рисунок 16 – Наглядное представление факта

Несмотря на то, что латынь является языком мертвым, без неё представить медицину невозможно. Заболевания, симптомы и синдромы, а также патологические новообразования в человеческом организме – всё это имеет свое название во всех языках мира. Квалифицированные врачи из разных стран прекрасно понимают друг друга благодаря латыни. И поэтому каждый человек, как-либо связанный с медициной, должен знать перевод. Исходя из этого, в создаваемое приложение была добавлена одна из самых важных функций – словарь наименований на русском и латинских языках. В нем представлен дифференцированный по темам перечень основных терминов различных разделов анатомии человека, что наглядно продемонстрировано на рисунке 17.

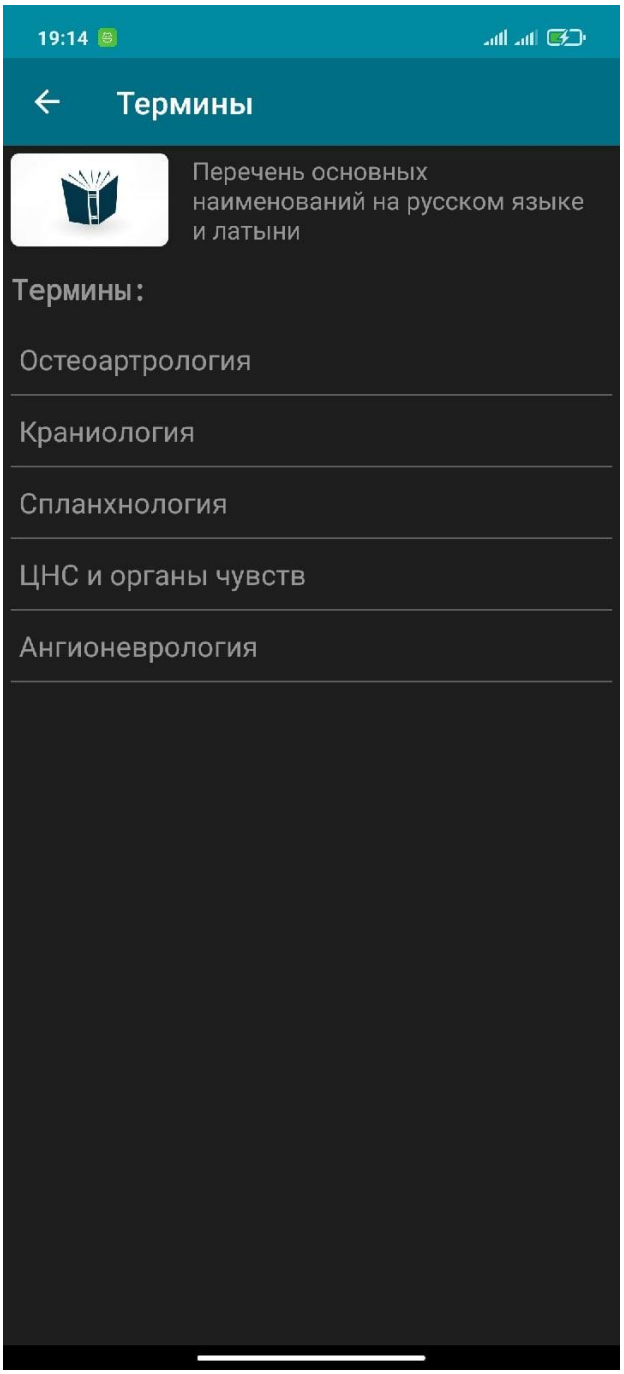


Рисунок 17 – Темы терминов

Для предотвращения путаницы, все термины делятся на области, к которым они относятся – остеоартрология, краниология, спланхнология, ЦНС и органы чувств и ангионеврология. Выбрав необходимую, пользователю предоставляется список основных терминов выбранной области на русском и латыни. Данный список представлен на рисунке 18.

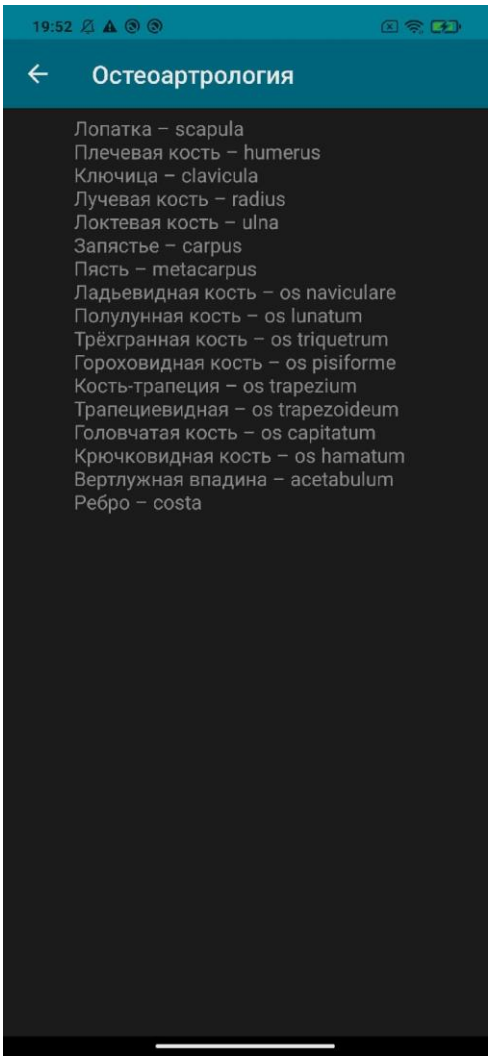


Рисунок 18 – Термины

Таким образом, пользователю предоставляется возможность поиска основных терминов каждого комплекса научных дисциплин анатомии человека.

5 Планирование разработки и оценка бюджета

Одной из самых важных частей создания проекта является проведения планирования разработки и оценки бюджета. От того, на сколько профессионального и грамотно будут проведены данные операции, будет завесить экономическая целесообразность проекта. Для этого был определены списки ресурсов и задач, решение которых необходимо для создания приложения, и перечень ресурсов, с помощью которых эти задачи будут решаться.

В ходе планирования было выделено 42 задачи, представленные в таблице 3.

Таблица 3 – Задачи проекта

№	Название задачи	Длительность	Предшественники
1	Начало проекта	_	_
2	Анализ предметной области	_	_
3	Поиск литературы	2 дня	1
4	Изучение литературы	1 день	3
5	Анализ предметной области завершен	_	4
6	Анализ продуктов-аналогов	_	_
7	Поиск продуктов-аналогов	2 дня	1
8	Сравнение продуктов-аналогов	1 день	7
9	Анализ продуктов-аналогов завершен	-	8
10	Постановка задачи	_	_
11	Формирование целей и задач проекта	1	5,9
12	Постановка задачи завершена	_	11
13	Анализ требований	_	_
14	Анализ функциональных требований	1 день	12
15	Анализ нефункциональных требований	1 день	12
16	Анализ требований завершен	_	14,15
17	Выбор средств реализации проекта	_	_
18	Анализ возможных программных средств	2 дня	16
19	Выбор программного средства для реализации	1 день	18
20	Выбор средств реализации проекта завершен	_	19

Продолжение таблицы 3

	Название задачи	Длительность	Предшественники
21	Проектирование	_	_
22	Проектирование структуры данных	2 дня	20
23	Проектирование интерфейса	2 дня	20
24	Проектирование структуры программного средства	2 дня	23
25	Проектирование завершено	_	22,24
26	Реализация проекта	_	_
27	Реализация БД	5 дней	25
28	Реализация пользовательского интерфейса	4 дня	25
29	Написание кода программы	12 дней	27,28
30	Реализация завершена	_	29
31	Тестирование	_	_
32	Тестирование методом причинно-следственных связей	2 дня	30
33	Автоматизированное тестирование	1 день	32
34	Структурное тестирование	1 день	33
35	Тестирование интерфейса	1 день	34
36	Вычисление метрик кода	2 дня	35
37	Тестирование завершено	_	36
38	Документирование	_	_
39	Составление глоссария	1 день	37
40	Составление отчета	2 дня	39
41	Документирование завершено	_	40
42	Конец проекта	_	41

Все необходимые для выполнении поставленных задач ресурсы представлены в таблице 4.

Таблица 4 – Список ресурсов проекта

№	Название ресурса	Тип	Стандартная ставка	Ставка сверхурочных
1	Программист1	Трудовой	30 000 Р\мес	200 Р\час
2	Программист2	Трудовой	30 000 Р\мес	200 Р\час

Продолжение таблицы 4

№	Название ресурса	Тип	Стандартная ставка	Ставка сверхурочных
3	Тестировщик	Трудовой	25 000 Р\мес	200 Р\час
4	Документовед	Трудовой	20 000 Р\мес	
5	Интернет	Затраты		
6	Компьютер	Материальный		
7	Android-устройство	Материальный		

После выделения ресурсов они были распределены по задачам. Результаты представлены в таблице 5.

Таблица 5 – Назначения ресурсов задачам

Номер ресурса	Номера задач
1	3, 4, 11, 14, 18, 19, 23, 24, 28, 29
2	7, 8, 11, 15, 22, 27, 29
3	32, 33, 34, 35, 36
4	39, 40
5	3, 4, 7, 8, 11, 14, 15, 18, 19, 22, 23,24,27, 28, 29, 32 , 33, 34, 35, 36, 39, 40
6	3, 4, 7, 8, 11, 14, 15, 18, 19, 22, 23,24,27, 28, 29, 32, 33, 34, 35, 36, 39, 40
7	32, 33, 34, 35, 36

В качестве среды для планирования бюджета была выбрана Microsoft Project 2010 – программа для управления проектами, помощи в разработке планов и распределения ресурсов по определенным задачам, отслеживания прогресса и анализа работы. В нее были введены описанные выше списки задач и ресурсов. Занесенные в нее данные, а так же вычисленные даты начала и завершения задач, представлены на рисунках 19 и 20. За начало проекта была принята дата 13.04.2020.

	Режим задачи	Название задачи	Длительно	Начало	Окончание	Предшественники	Названия ресурсов
1		Начало проекта	0 дней	Пн 13.04.20	Пн 13.04.20		
2		**Анализ предметной области**	**3 дней**	**Пн 13.04.20**	**Ср 15.04.20**		
3		Поиск литературы	2 дней	Пн 13.04.20	Вт 14.04.20	1	Программист1;Компьютер[1];Интернет
4		Изучение литературы	1 день	Ср 15.04.20	Ср 15.04.20	3	Интернет;Компьютер[1];Программист1
5		Анализ предметной области завершен	0 дней	Ср 15.04.20	Ср 15.04.20	4	
6		**Анализ продуктов-аналогов**	**3 дней**	**Пн 13.04.20**	**Ср 15.04.20**		
7		Поиск продуктов аналогов	2 дней	Пн 13.04.20	Вт 14.04.20	1	Программист2;Компьютер[1];Интернет
8		Сравнение продуктов аналогов	1 день	Ср 15.04.20	Ср 15.04.20	7	Программист2;Компьютер[1];Интернет
9		Анализ продуктов-аналогов завершен	0 дней	Ср 15.04.20	Ср 15.04.20	8	
10		**Постановка задачи**	**1 день**	**Чт 16.04.20**	**Чт 16.04.20**		
11		Формирование целей и задач проета	1 день	Чт 16.04.20	Чт 16.04.20	5;9	Программист1;Программист2;Компьютер[2];Интернет
12		Постановка задачи завершена	0 дней	Чт 16.04.20	Чт 16.04.20	11	
13		**Анализ требований**	**1 день**	**Пт 17.04.20**	**Пт 17.04.20**		
14		Анализ функциональных требований	1 день	Пт 17.04.20	Пт 17.04.20	12	Программист1;Компьютер[1];Интернет
15		Анализ нефункциональныз требований	1 день	Пт 17.04.20	Пт 17.04.20	12	Программист2;Компьютер[1];Интернет
16		Анализ требований завершен	0 дней	Пт 17.04.20	Пт 17.04.20	14;15	
17		**Выбор средств реализации проекта**	**3 дней**	**Пн 20.04.20**	**Ср 22.04.20**		
18		Анализ возможный программных средств	2 дней	Пн 20.04.20	Вт 21.04.20	16	Программист1;Компьютер[1];Интернет
19		Выбор программного средства для реализации	1 день	Ср 22.04.20	Ср 22.04.20	18	Интернет;Компьютер[1];Программист1
20		Выбор средств реализации проекта завершен	0 дней	Ср 22.04.20	Ср 22.04.20	19	
21		**Проектирование**	**4 дней**	**Чт 23.04.20**	**Вт 28.04.20**		
22		Проектирование структуры данных	2 дней	Чт 23.04.20	Пт 24.04.20	20	Программист2;Компьютер[1];Интернет
23		Проектирование интерфейса	2 дней	Чт 23.04.20	Пт 24.04.20	20	Интернет;Компьютер[1];Программист1
24		Проектирование структуры программного средства	2 дней	Пн 27.04.20	Вт 28.04.20	23	Программист1;Компьютер[1];Интернет
25		Проектирование завершено	0 дней	Вт 28.04.20	Вт 28.04.20	22;24	
26		**Реализация проекта**	**17 дней**	**Ср 29.04.20**	**Чт 21.05.20**		
27		Реализация БД	5 дней	Ср 29.04.20	Вт 05.05.20	25	Интернет;Компьютер[1];Программист2
28		Реализция пользовательского интерфейса	4 дней	Ср 29.04.20	Пн 04.05.20	25	Интернет;Компьютер[1];Программист1
29		Написание кода программы	12 дней	Ср 06.05.20	Чт 21.05.20	27;28	Интернет;Компьютер[2];Программист1;Программист2
30		Реализация завершена	0 дней	Чт 21.05.20	Чт 21.05.20	29	
31		**Тестирование**	**7 дней**	**Пт 22.05.20**	**Пн 01.06.20**		
32		Тестирование методом причинно-следственных связей	2 дней	Пт 22.05.20	Пн 25.05.20	30	Тестировщик;Компьютер[1];Android-устройство[1];Интернет
33		Автоматизированное тестирование	1 день	Вт 26.05.20	Вт 26.05.20	32	Android-устройство[1];Интернет;Компьютер[1];Тестировщик
34		Структурное тестирование	1 день	Ср 27.05.20	Ср 27.05.20	33	Android-устройство[1];Интернет;Компьютер[1];Тестировщик
35		Тестирование интерфейса	1 день	Чт 28.05.20	Чт 28.05.20	34	Android-устройство[1];Интернет;Компьютер[1];Тестировщик
36		Вычислегние метрик кода	2 дней	Пт 29.05.20	Пн 01.06.20	35	Android-устройство[1];Интернет;Компьютер[1];Тестировщик
37		Тестирование завершено	0 дней	Пн 01.06.20	Пн 01.06.20	36	
38		**Документирование**	**3 дней**	**Вт 02.06.20**	**Пт 05.06.20**		
39		Составление глоссария	1 день	Вт 02.06.20	Вт 02.06.20	37	Документовед;Компьютер[1];Интернет
40		Составление отчета	2 дней	Ср 03.06.20	Чт 04.06.20	39	Документовед;Интернет;Компьютер[1]
41		Документирование завершено	0 дней	Пт 05.06.20	Пт 05.06.20	40	
42		Конец проекта	0 дней	Пт 05.06.20	Пт 05.06.20	41	

Рисунок 19 – Список задач проекта

Внесенные данные необходимы для построения последующих графиков и диаграмм.

	Название ресурса	Тип	Единицы измерения материалов	Краткое название	Группа	Макс. единиц	Стандартная ставка	Ставка сверхурочных	Затраты на использ.	Начисление	Базовый календарь	Код
1	Программист1	Трудовой		П1		100%	30 000,00 ₽/мес	200,00 ₽/ч	0,00 ₽	Пропорциональнс	Стандартный	
2	Программист2	Трудовой		П2		100%	30 000,00 ₽/мес	200,00 ₽/ч	0,00 ₽	Пропорциональнс	Стандартный	
3	Тестировщик	Трудовой		Т		100%	25 000,00 ₽/мес	200,00 ₽/ч	0,00 ₽	Пропорциональнс	Стандартный	
4	Документовед	Трудовой		Д		100%	20 000,00 ₽/мес	200,00 ₽/ч	0,00 ₽	Пропорциональнс	Стандартный	
5	Интернет	Затраты		И						Пропорциональнс		
6	Компьютер	Материальный		К			200,00 ₽		0,00 ₽	Пропорциональнс		
7	Android-устройство	Материальный		А			200,00 ₽		0,00 ₽	Пропорциональнс		

Рисунок 20 – Ресурсы проекта

По введенным результатам, система Microsoft Project 2010 автоматически построила диаграмму Ганта. Данный тип диаграмм предназначен для иллюстрации плана или графика работ по какому-либо проекту. Диаграмма Ганта представлена на рисунке 21.

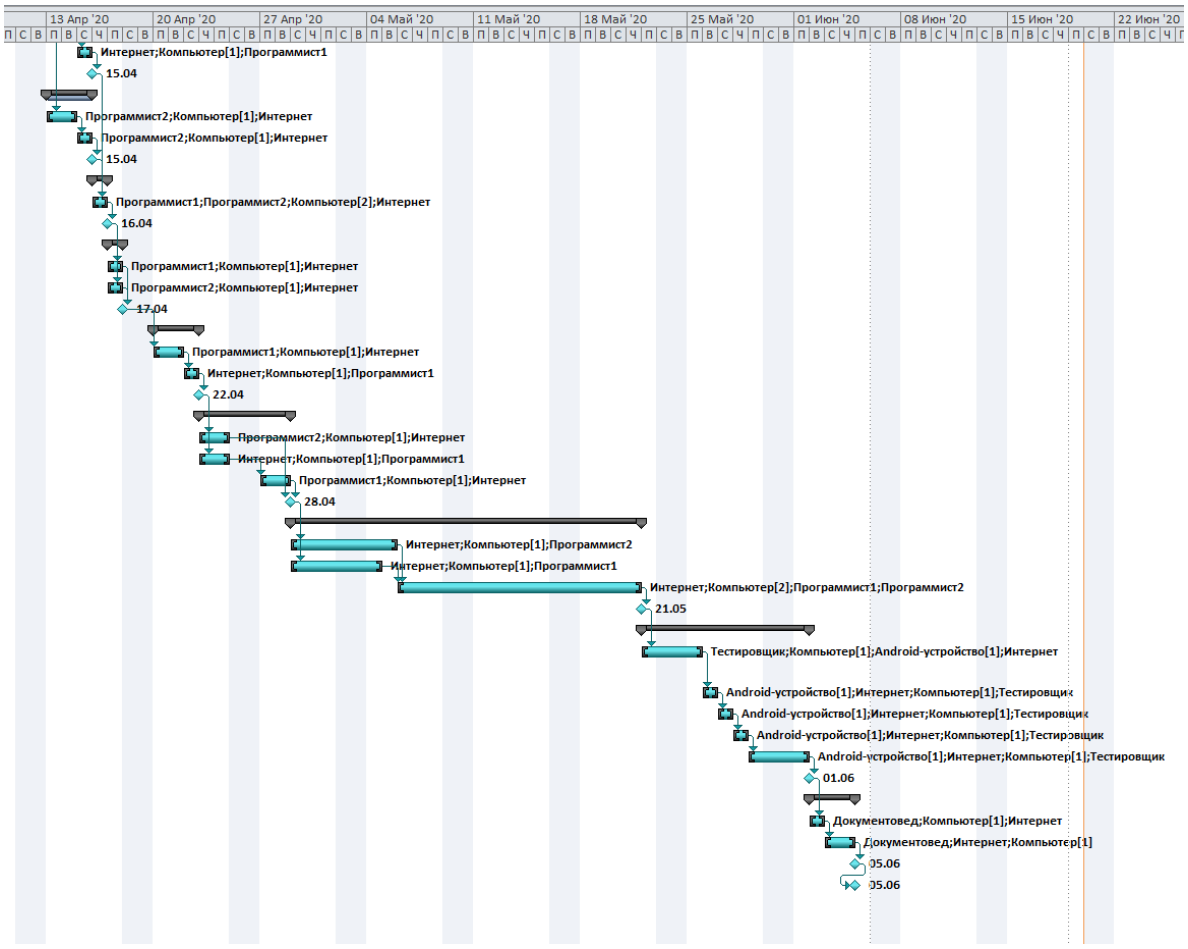
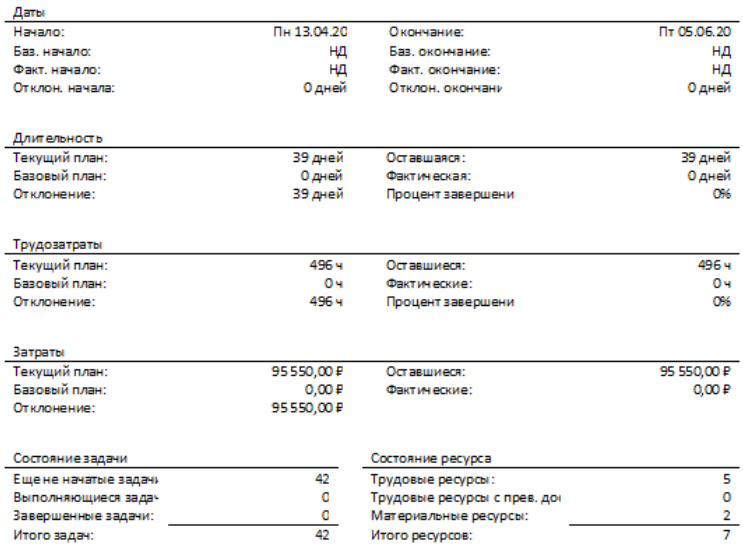


Рисунок 21 – Диаграмма Ганта

Так же, с помощью этой системы, был построен график трудозатрат, представленный на рисунке 22.

Рисунок 22 – График трудозатрат

На рисунке 23 изображен обзор проекта, составленный Microsoft Project 2010.



Даты			
Начало:	Пн 13.04.20	Окончание:	Пт 05.06.20
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончани	0 дней

Длительность			
Текущий план:	39 дней	Оставшаяся:	39 дней
Базовый план:	0 дней	Фактическая:	0 дней
Отклонение:	39 дней	Процент завершени	0%

Трудозатраты			
Текущий план:	496 ч	Оставшиеся:	496 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	496 ч	Процент завершени	0%

Затраты			
Текущий план:	95 550,00 ₽	Оставшиеся:	95 550,00 ₽
Базовый план:	0,00 ₽	Фактические:	0,00 ₽
Отклонение:	95 550,00 ₽		

Состояние задачи	
Еще не начатые задачи	42
Выполняющиеся задач	0
Завершенные задачи:	0
Итого задач:	42

Состояние ресурса	
Трудовые ресурсы:	5
Трудовые ресурсы с прев. до	0
Материальные ресурсы:	2
Итого ресурсов:	7

Рисунок 23 – Обзор проекта

Из этого рисунка видно, что длительность реализации проекта составляет 39 дней, и на его реализацию необходимо затратить 95 550 руб. Из этих данных можно сделать вывод об экономической и трудозатратой целесообразности проекта.

6 Тестирование программного обеспечения

6.1 Цели и виды тестирования

Тестирование программного обеспечения – процесс анализа программного средства и сопутствующей документации с целью выявления дефектов и повышения качества продукта [13].

Тестирование – неотъемлемая часть разработки любого программного обеспечение. На данном этапе происходит полная проверка приложения и определение всех имеющихся проблем. Любая из таких проблем должна быть устранена, после чего процесс тестирования необходимо повторить еще раз.

Тестирование невозможно без верификации и валидации. Эти понятия схожи по звучанию и подразумевают проверку системы и того, что она обладает ожидаемыми свойствами, но все же означают разные типы проверок.

Верификация – это процесс оценки того, на сколько система или её компоненты удовлетворяют результатам текущего этапа разработки и условиям, которые были сформированы в начале этого этапа, выполняются ли цели, сроки и задачи по разработке проекта.

Валидация – это определение соответствия разрабатываемого программного обеспечения ожиданиям и потребностям пользователя, требованиям к системе [14].

Верификация:

– проверяет, выполняет ли исполнитель программный;

– продукт верно, в соответствии с исходными требованиями к разрабатываемой системе;

– проверяет, правильно ли работают функциональные возможности разрабатываемого продукта;

– проверяет соответствие требованиям технического задания.

Валидация:

– проверяет, выполняет ли исполнитель программный;

– продукт так, как этого хочет заказчик;

– проверяет соответствие поведения функциональных возможностей разрабатываемого приложения поведению, которое подразумевает заказчик [15];

– дает субъективную оценку продукта.

Существуют пять основных методик тестирования: модульное, интеграционное, системное, регрессионное и приемочные испытания.

Модульное тестирование – тестирование, при котором происходит проверка определенных модулей или каких-либо компонентов системы.

Интеграционное тестирование – направлено на поиск неисправностей в работе интерфейса. Тестирование проводится на тех компонентах системы, которые прошли модульное тестирование. Эти модули интегрируются между собой и проверяются на наличие ошибок.

Системное тестирование – вся система проверяется на наличие любого рода неисправностей. Проверяются требования к приложению.

Приемочные испытания – финальный этап проверки разработанной системы перед передачей заказчику. Данный тип тестирования необходим для гарантии того, что весь функционал работает исправно и верно. Приемочные испытания могут проводиться среди тех, кто разрабатывал данную систему или непосредственно самим клиентом [16].

Регрессионное тестирование – это набор разнообразных тестов, которые направлены на поиск и обнаружение дефектов в уже протестированных участках приложения. Делается это не для того, чтобы окончательно убедиться в отсутствии проблем и ошибок, а для поиска и исправления регрессионных ошибок. Регрессионные ошибки – недоработки программы, ошибки, но появляются они не при написании программы, а при добавлении в существующий код нового участка программы или исправлении других, найденных ранее ошибок в коде программы, что и

могло стать причиной возникновения новых дефектов в уже протестированном продукте.

Таким образом, мы можем сказать, что цель регрессионного тестирования – убедиться, что исправление одних ошибок не стало причиной возникновения других и что обновление кода не создало новых дефектов в проверенном ранее.

6.2 Конфигурационное тестирование под разнообразные модели

Конфигурационное тестирование (Configuration Testing) – специальный вид тестирования, направленный на проверку работы программного обеспечения при различных конфигурациях системы. Цель Тестирования: определить оптимальную конфигурацию оборудования, обеспечивающую требуемые характеристики производительности и времени реакции тестируемой системы [17].

Для тестирования приложения в Android Studio были применены виртуальные машины с различными характеристиками. Конфигурации виртуальных машин приведены на рисунке 24.

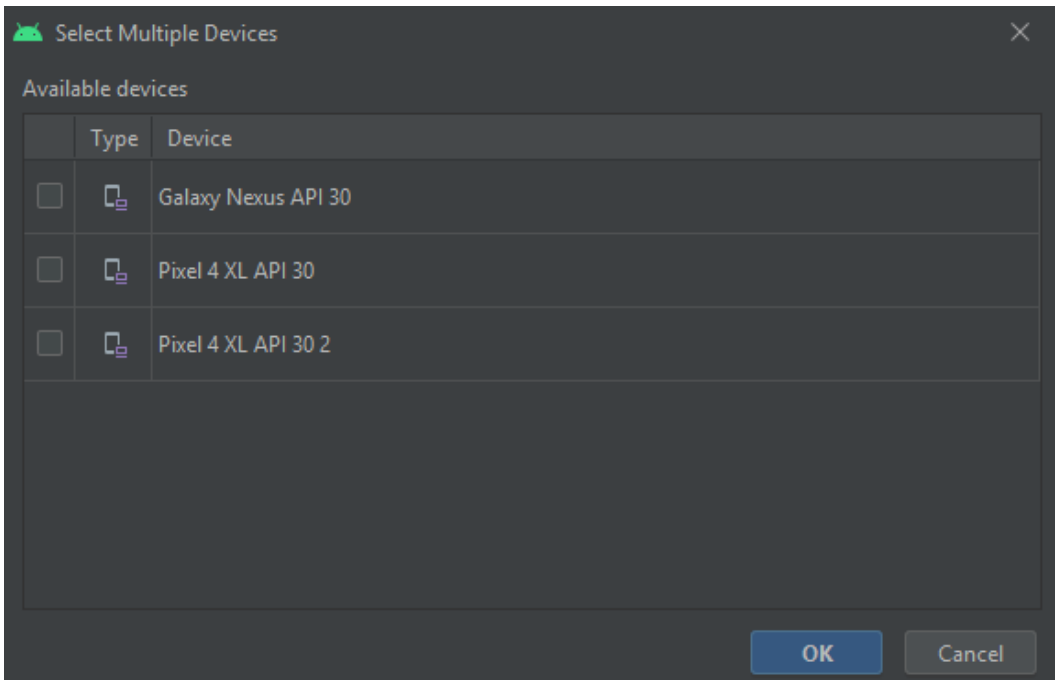


Рисунок 24 – Конфигурации виртуальных машин

Android Studio имеет возможность отображения пользовательского интерфейса для разнообразных конфигураций экрана. Все они приведены на рисунке 25.

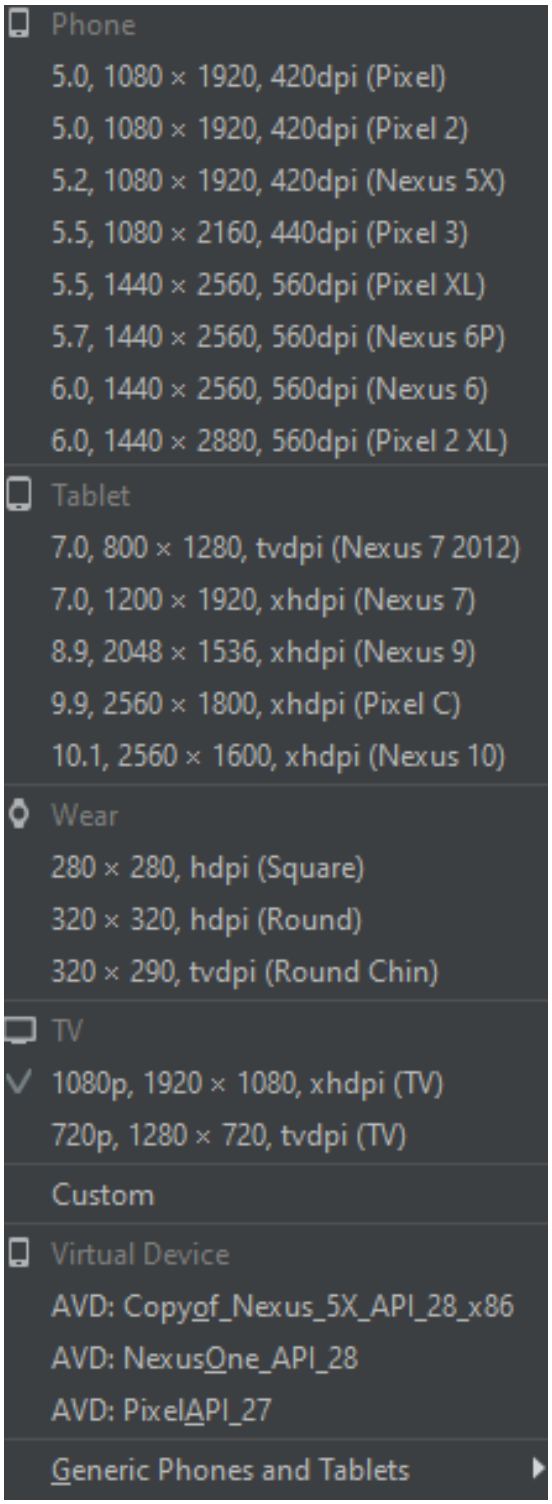


Рисунок 25 – Разнообразные конфигурации экрана

Пользовательский интерфейс разрабатываемой системы был сделан методом резиновой верстки. В данной случае размеры задаются в процентах от ширины экрана, на котором, соответственно, открывается приложение. Такой подход позволяет оптимизировать приложения под любые размеры экранов смартфонов пользователей. Помимо резиновой верстки, существуют еще фиксированная и адаптивная. В первом случае, размеры страниц четко указываются в пикселях и являются статичным значением вне зависимости от размеров экрана. Во втором же случае, может происходить полное изменение интерфейса и различных блоков в зависимости от разрешения экрана, на котором будет открыт ресурс.

6.3 Тестирование методом функциональных диаграмм

Рассмотрим данный вид тестирования на примере функции, проверяющей введенный пользователем пароль на соответствие следующим требованиям:

- должен содержать не менее 8 символов;
- обязательно наличие минимум 1 цифры;
- обязательно наличие минимум 1 прописной буквы;
- обязательно наличие минимум 1 заглавной буквы;
- обязательно наличие минимум 1 спецсимвола.

Тестирование проводилось методом функциональных диаграмм. Этот метод проверяет правильность распознавания приложением всех параметров вместе и их совокупное влияние на работу функции.

Функциональное тестирование – один из видов тестирования, направленный на проверку соответствий функциональных требований программного обеспечения к его реальным характеристикам. Основной задачей данного тестирования является подтверждение того, что разрабатываемый программный продукт обладает всем функционалом, требуемым заказчиком.

Функциональные диаграммы – это диаграммы, создающиеся на ранних этапах проектирования, которые, в первую очередь, наглядно отображают все взаимосвязи функций разрабатываемого программного обеспечения и его составные части.

На рисунке 26 представлена функциональная диаграмма, на которой видно, что для того, чтобы введенный пользователем пароль был зарегистрирован в системе, он должен пройти по всем ограничениям.

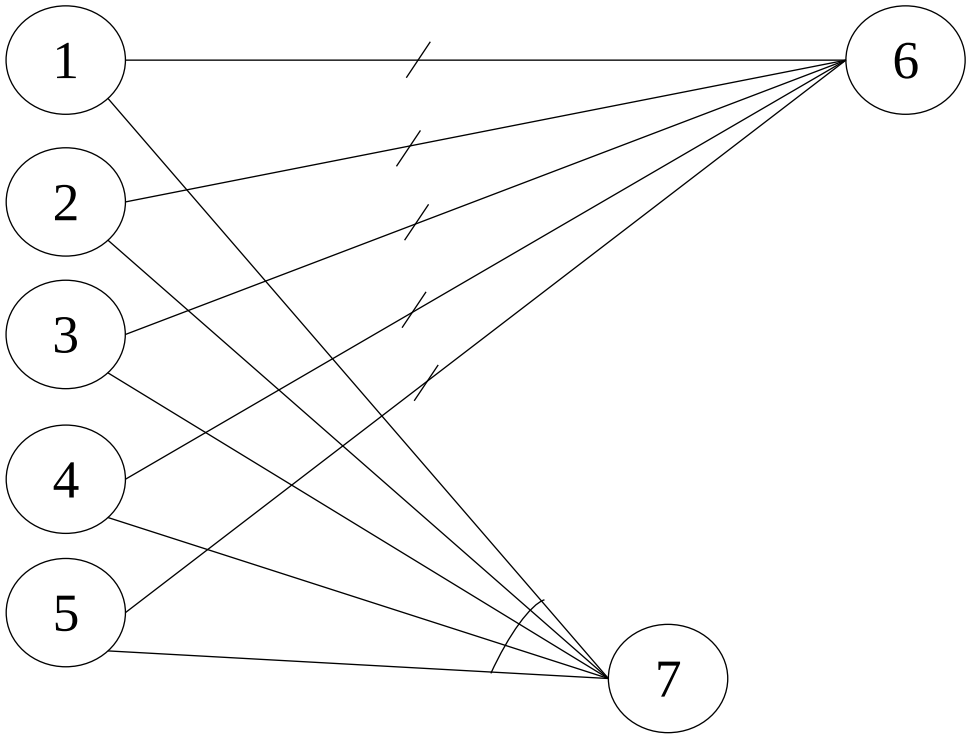


Рисунок 26 – Функциональная диаграмма

Цифры, использованные на этом рисунке, имеют следующие значения:

1 – длина пароля больше или равна 8 символам;

2 – пароль содержит минимум 1 цифру;

3 – пароль содержит минимум 1 прописную букву;

4 – пароль содержит минимум 1 заглавную букву;

5 – пароль содержит минимум 1 спецсимвол;

6 – вывод сообщения «Пароль недействителен»;

7 – принятие пароля.

В таблице 6, построенной на основе предыдущей диаграммы, указаны решения, на основе которых уже непосредственно строятся тесты, представленные в таблице 7.

Таблица 6 – Таблица решений

	1	2	3	4	5	6
1	0					1
2		0				1
3			0			1
4				0		1
5					0	1
6	1	1	1	1	1	0
7	0	0	0	0	0	1

Таблица 7 – Тесты, полученные в результате применения функционального подхода

№		Длина пароля меньше 8 символам	Пароль не содержит 1 цифру	Пароль не содержит 1 прописную букву	Пароль не содержит 1 заглавную букву	Пароль не содержит 1 спецсимвол	Ошибок в параметрах нет
1	Длина пароля меньше 8 символам	Вывод сообщения «Пароль недействителен»	Вывод сообщения «Пароль недействителен»	Вывод сообщения «Пароль недействителен»	Вывод сообщения «Пароль недействителен»	Вывод сообщения «Пароль недействителен»	Не учитывается
2	Пароль не содержит 1 цифру	Вывод сообщения «Пароль недействителен»	Вывод сообщения «Пароль недействителен»	Вывод сообщения «Пароль недействителен»	Вывод сообщения «Пароль недействителен»	Вывод сообщения «Пароль недействителен»	Не учитывается

Продолжение таблицы 7

3	Пароль не содержит 1 прописную букву	Вывод сообщения «Пароль недействителен»	Вывод сообщения «Пароль недействителен»	Вывод сообщения «Пароль недействителен»	Вывод сообщения «Пароль недействителен»	Вывод сообщения «Пароль недействителен»	Не учитывается
4	Пароль не содержит 1 заглавную букву	Вывод сообщения «Пароль недействителен»	Вывод сообщения «Пароль недействителен»	Вывод сообщения «Пароль недействителен»	Вывод сообщения «Пароль недействителен»	Вывод сообщения «Пароль недействителен»	Не учитывается
5	Пароль не содержит 1 спецсимвол	Вывод сообщения «Пароль недействителен»	Вывод сообщения «Пароль недействителен»	Вывод сообщения «Пароль недействителен»	Вывод сообщения «Пароль недействителен»	Вывод сообщения «Пароль недействителен»	Не учитывается
6	Ошибок в параметрах нет	Не учитываетсся	Не учитывается	Не учитывается	Не учитывается	Не учитывается	Принятие пароля

Применение этого метода тестирования позволило минимизировать количество тестов для проверки работы системы со всеми параметрами благодаря тому, что он выявляет самые важные зависимости параметров друг от друга, что, в свою очередь, позволяет избежать прямого перебора всех возможных комбинаций их значений.

Заключение

В результате выпускной квалификационной работы было разработано мобильное приложение «медицинский справочник».

Клиентская часть работы была реализована на языке Java в интеграционной системе AndroidStudio. В серверной части ПО с помощью DB Browser SQLite была реализована БД, хранящая всю информацию.

В процесее выполнения работы было проведено функциональное тестирование. На основе анализа результатов тестирования можно сделать вывод о корректноти и надежности ПО.

В ходе работы было произведено планирование и оценка бюджета разработки проекта с помощью программного продукта для управления проектами Microsoft Project 2010, в процессе которого были выяснены длительность разработки проекта, равная 39 дням, и на его реализацию необходимо затратить 95 550 руб. Результат оценки бюджета и длительности разработки проекта свидетельствует о его экономической выгодности.

Таким образом, в результате выполнения выпускной квалификационной работы было разработано мобильное приложение медицинского справочника для изучения нормальной анатомии человека.

Список использованных источников

1. Медицина. [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Медицина (дата обращения: 11.04.2021).

2. Что такое техническое задание [Электронный ресурс]. URL: https://kontur.ru/articles/5945#1 (дата обращения: 14.04.2021).

3. Мобильные приложения. [Электронный ресурс]. URL: https://www.calltouch.ru/glossary/chto-takoe-mobilnoe-prilozhenie-i-zachem-ono-mozhet-potrebovatsya/ (дата обращения: 16.04.2021).

4. Лафоре, Р. Структуры данных и алгоритмы в Java / Пер. с англ. под ред. Е.А. Матвеева – СПб: Питер, 2018. – 704 с.

5. SQLite [Электронный ресурс]. URL: https://habr.com/ru/post/149356/ (дата обращения: 17.04.2021).

6. Android studio IDE от Google [Электронный ресурс]. URL: https://wnfx.ru/android-studio-ide-ot-google/ (дата обращения: 19.04.2021).

7. Дейтел, П «Android для разработчиков» / П. Дейтел, Х.Дейтел. – СПб: Питер 2016. – 250 с.

8. Варфел, Т. В18 Прототипирование. Практическое руководство / Тодд Заки Варфел; пер. И. Лейко. – М.: Иванов и Фербер, 2013. – 240 с.

9. UML – диаграмма вариантов использования (use case diagramm) [Электронный ресурс]. URL: http://habrahabr.ru/post/47940 (дата обращения 08.04.2021).

10. Плюсы и минусы мобильных приложений. [Электронный ресурс]. URL: http://artismedia.by/blog/plyusy-i-minusy-mobilnyh-sajtov-i-prilozhenij/ (дата обращения: 17.04.2021).

11. А.П. Фисун Теоретические и практические основы человеко-компьютерного взаимодействия: базовые понятия человекокомпьютерных систем в информатике и информационной безопасности / Деп. в ВИНИТИ 15.10. 2004 г. № 1624. В 2004. Орел: Орловский государственный университет, 2004. – 169 с.

12. Дружественный интерфейс [Электронный ресурс]. URL: http://artlib.osu.ru/Docs/piter/bookchap/978594723921.html (дата обращения: 25.05.2021).

13. Куликов, С. Тестирование программного обеспечения. Базовый курс / С. С. Куликов. – Минск: Четыре четверти, 2017. – 312 с.

14. Тестирование ПО [Электронный ресурс]. URL: http://www.protesting.ru/testing/ (дата обращения: 21.05.2021).

15. Тестирование [Электронный ресурс]. URL: https://matlab.ru/solutions/mbd/test (дата обращения: 21.05.2021).

16. Майерс, Г. Искусство тестирования программ. / Г. Майерс, Т. Баджетт, К. Сандлер. – Москва: Издательство Вильямс, 2016. – 272 с.

17. Конфигурационное тестирование [Электронный ресурс]. URL: http://www.protesting.ru/testing/types/configuration.html (дата обращения 22.05.2021).

Приложение А
ГЛОССАРИЙ
(справочное)

ТЗ – Техническое задание.

ГОСТ – Государственный стандарт.

ПО – Программное обеспечение.

Play Маркет (Google Play, Android Market) – магазин приложений, игр, книг, музыки и фильмов компании Google и других компаний, позволяющий владельцам устройств с операционной системой Android устанавливать и приобретать разнообразные приложения.

Android Studio – интегрированная среда разработки для работы с платформой Android.

Java – строго типизированный объектно-ориентированный язык программирования общего назначения.

SQLite – компактная встраиваемая СУБД.

Приложение Б

ЛИСТИНГ ПРОГРАММЫ НА ЯЗЫКЕ JAVA

(рекомендуемое)

Листинг Б.1 – Код класса «MAINACTIVITY»

```
package com.medical.dictionary.Activities;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.app.Activity;
import android.app.ActivityOptions;

import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.os.Build;
import android.os.Bundle;

import android.view.View;
import android.widget.Button;

import com.medical.dictionary.Adapters.RecyclerViewClickListener;
import com.medical.dictionary.Adapters.RecyclerViewTouchListener;
import com.medical.dictionary.Adapters.RvMainThemeAdapter;
import com.medical.dictionary.Models.MainTheme;
import com.medical.dictionary.Models.ThemeItem;
import com.medical.dictionary.R;
import com.medical.dictionary.Utils.DataBaseHelper;

import java.io.IOException;
import java.util.ArrayList;

/* Главнаяактивитистемами
* - Остеартрология
* - Краниология
* - Спланхнология
* - ЦНСиорганычувств
* - Ангионеврология
*  */

public class MainActivityextends AppCompatActivity {

private Context context; //позволяетсозданнымобъектампонятьчтопроисходит
private RecyclerViewrvMain;  //выделениеэлементов
private Button btnFacts; //кнопка(событиеввода)
private ArrayList<MainTheme>mainThemes; //список (темы)

private DataBaseHelpermDBHelper; //подготовкаБД
private SQLiteDatabasemDb;//открываетдоступначтениеБД

@Override//аннотацияjava. След. Методпереопределяетметодsuperclass
protected void onCreate(Bundle savedInstanceState) {(1)

// oncreate – вызывается при создании или перезапуске
активности

super.onCreate(savedInstanceState);// супер- экземпляр родител-гокласса

setContentView(R.layout.activity_main);// R-ресурс.Связаносxml (виз.дизайн)

context = MainActivity.this;
```

```
// ui
rvMain= findViewById(R.id.rv_main);     // ui- дизайн. И зxml

btnFacts= findViewById(R.id.btn_facts);

// темы
mainThemes= new ArrayList<>();

dataBaseConnection();

        Cursor cursorThread;
cursorThread = mDb.rawQuery("select * from main_theme", null);//rawquery-
выводобъединения
таблиц

cursorThread.moveToFirst();
while (!cursorThread.isAfterLast()) {

mainThemes.add(new MainTheme(
cursorThread.getInt(0),
cursorThread.getString(1),
cursorThread.getString(2)
            ));

cursorThread.moveToNext();
        }
cursorThread.close();

// отображениетемвсписок

rvMain.setHasFixedSize(true); (2,3)
RecyclerView.LayoutManagerlayoutManager = new LinearLayoutManager(context);
rvMain.setLayoutManager(layoutManager);
RvMainThemeAdapter adapter = new RvMainThemeAdapter(context, mainThemes);
rvMain.setAdapter(adapter); // спец.компонент, связывающийисточникданныхс
элементомсписка

// обработчикнажатиянаэлементсписка
rvMain.addOnItemTouchListener(new RecyclerViewTouchListener(context, rvMain,
new RecyclerViewClickListener() {
@Override
public void onClick(View view, intposition) {
                Intent intent = new Intent(context, ThemeActivity.class); (4)

intent.putExtra("THEME", mainThemes.get(position));
ActivityOptions options = (5)

ActivityOptions.makeSceneTransitionAnimation((Activity) context); // метод,
который принимает в качестве параметров контекст приложения и
Столько общих элементов, сколько нужно.

startActivity(intent,options.toBundle());
            }


@Override
public void onLongClick(View view, intposition) { // прослушиватель
длинногоклика ( нажатие )
            }
        }));
```

```
btnFacts.setOnClickListener(new View.OnClickListener() { //прослушив-льклика
@Override

public void onClick(View view) {
                Intent intent = new Intent(context, FactsActivity.class);
ActivityOptions options =

ActivityOptions.makeSceneTransitionAnimation((Activity) context);
startActivity(intent,options.toBundle());
            }
        });
    }

private void dataBaseConnection(){

mDBHelper= new DataBaseHelper(context);
try {
mDBHelper.updateDataBase();
        } catch (IOExceptionmIOException) {
throw new Error("UnableToUpdateDatabase");
        }

try {
mDb= mDBHelper.getWritableDatabase();
        } catch (SQLExceptionmSQLException) {
throw mSQLException;
        }
    }
}
```

Листинг Б.2 – Код класса «THEMEACTIVITY»

```
package com.medical.dictionary.Activities;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.app.Activity;
import android.app.ActivityOptions;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.os.Build;
import android.os.Bundle;
import android.transition.Slide;
import android.view.Gravity;
import android.view.MenuItem;
import android.view.View;
import android.view.animation.DecelerateInterpolator;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import com.medical.dictionary.Adapters.RecyclerViewClickListener;
import com.medical.dictionary.Adapters.RecyclerViewTouchListener;
import com.medical.dictionary.Adapters.RvMainThemeAdapter;
import com.medical.dictionary.Adapters.RvThemeItemsAdapter;
import com.medical.dictionary.Models.MainTheme;
import com.medical.dictionary.Models.ThemeItem;
import com.medical.dictionary.R;
```

```
import com.medical.dictionary.Utils.DataBaseHelper;


import java.io.IOException;
import java.util.ArrayList;

public class ThemeActivityextends AppCompatActivity { // внедрениеновых
«фишек» в
старыевещи

private Context context;

private MainThemetheme;

private RecyclerViewrvMainThemeDetails;
private ImageViewivThemePic;
private TextViewtvThemeTitle;
private TextViewtvThemeAbout;

private DataBaseHelpermDBHelper;
private SQLiteDatabasemDb;

private ArrayList<ThemeItem>themeItems;

public void setAnimation(){
            Slide slide = new Slide();
slide.setSlideEdge(Gravity.LEFT);
slide.setDuration(400);// продолжительностьанимации
slide.setInterpolator(new DecelerateInterpolator()); // постоянная
скорости
getWindow().setExitTransition(slide);// выход
getWindow().setEnterTransition(slide); // переходокна
    }

@Override

protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState); //сохранениеданныхприудалениииactivity
setAnimation();
setContentView(R.layout.activity_theme);
context = ThemeActivity.this;


//ui
rvMainThemeDetails= findViewById(R.id.rv_main_theme_details);
ivThemePic= findViewById(R.id.iv_main_theme_pic);
tvThemeAbout= findViewById(R.id.tv_main_theme_about);
tvThemeTitle= findViewById(R.id.tv_main_theme_title);

// полученивыбраннойтемыизпредыдущейактивити
theme = (MainTheme) getIntent().getExtras().getSerializable("THEME");

// отображениеназваниетемывтулбаре
setTitle(theme.getName());

tvThemeTitle.setText(theme.getName() + ":");

tvThemeAbout.setText(theme.getAbout());

switch (theme.getId()){
case 1:
ivThemePic.setImageDrawable(context.getResources().getDrawable(R.drawable.mai
n_theme_1));
```

```
break;

case 2:
ivThemePic.setImageDrawable(context.getResources().getDrawable(R.drawable.mai
n_theme_2));
break;

case 3:
ivThemePic.setImageDrawable(context.getResources().getDrawable(R.drawable.mai
n_theme_3));
break;

case 4:
ivThemePic.setImageDrawable(context.getResources().getDrawable(R.drawable.mai
n_theme_4));
break;

case 5:
ivThemePic.setImageDrawable(context.getResources().getDrawable(R.drawable.mai
n_theme_5));
break;

case 6:
ivThemePic.setImageDrawable(context.getResources().getDrawable(R.drawable.mai
n_theme_6));
break;

case 7:
ivThemePic.setImageDrawable(context.getResources().getDrawable(R.drawable.mai
n_theme_7));
break;
        }

// кнопка "назад" втулбаре
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
getSupportActionBar().setDisplayShowHomeEnabled(true);

// полученияэлементовтемыизбазыданных
dataBaseConnection();
themeItems= new ArrayList<>();

        Cursor cursorThread;
cursorThread = mDb.rawQuery("select * from theme_item where theme_id = " +
theme.getId(), null);
cursorThread.moveToFirst();
while (!cursorThread.isAfterLast()) {

themeItems.add(new ThemeItem(
cursorThread.getInt(0),
cursorThread.getInt(1),
cursorThread.getString(2),
cursorThread.getString(3)
            ));

cursorThread.moveToNext();
        }
cursorThread.close();

rvMainThemeDetails.setHasFixedSize(true);

RecyclerView.LayoutManagerlayoutManager = new LinearLayoutManager(context);
```

```
rvMainThemeDetails.setLayoutManager(layoutManager);

RvThemeItemsAdapter adapter = new RvThemeItemsAdapter(context, themeItems);

rvMainThemeDetails.setAdapter(adapter);

// обработчикнажатиянаэлементсписка
rvMainThemeDetails.addOnItemTouchListener(new
RecyclerViewTouchListener(context, rvMainThemeDetails, new
RecyclerViewClickListener() {
@Override
public void onClick(View view, intposition) {
                Intent intent = new Intent(context, ThemeItemActivity.class);
intent.putExtra("THEME_ITEM", themeItems.get(position));

ActivityOptions options =
ActivityOptions.makeSceneTransitionAnimation((Activity) context);
startActivity(intent,options.toBundle());
            }

@Override

public void onLongClick(View view, intposition) {
            }

        }));

}

private void dataBaseConnection(){

mDBHelper= new DataBaseHelper(context);

try {
mDBHelper.updateDataBase();
        } catch (IOExceptionmIOException) {
throw new Error("UnableToUpdateDatabase");
        }

try {
mDb= mDBHelper.getWritableDatabase();
        } catch (SQLExceptionmSQLException) {
throw mSQLException;
        }
    }

@Override

public booleanonOptionsItemSelected(MenuItem item) {
switch (item.getItemId()) {
case android.R.id.home:

onBackPressed();

return true;

default:
return super.onOptionsItemSelected(item);// метод обработки activity
        }
    }
}
```

Листинг Б.3 – Код класса «THEMEITEMACTIVITY»

```
package com.medical.dictionary.Activities;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.app.Activity;
import android.app.ActivityOptions;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.text.method.ScrollingMovementMethod;
import android.transition.Slide;
import android.view.Gravity;
import android.view.MenuItem;
import android.view.View;
import android.view.animation.DecelerateInterpolator;
import android.widget.ImageView;
import android.widget.TextView;

import com.medical.dictionary.Adapters.RecyclerViewClickListener;
import com.medical.dictionary.Adapters.RecyclerViewTouchListener;
import com.medical.dictionary.Adapters.RvItemsAdapter;
import com.medical.dictionary.Adapters.RvThemeItemsAdapter;
import com.medical.dictionary.Models.Item;
import com.medical.dictionary.Models.MainTheme;
import com.medical.dictionary.Models.ThemeItem;
import com.medical.dictionary.R;
import com.medical.dictionary.Utils.DataBaseHelper;

import java.io.IOException;
import java.util.ArrayList;

public class ThemeItemActivityextends AppCompatActivity {

private DataBaseHelpermDBHelper;
private SQLiteDatabasemDb;
private Context context;
private ThemeItemthemeItem;
private RecyclerViewrvThemeItem;
private TextViewtvThemeItemAbout;
private ArrayList<Item>items;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setAnimation();
setContentView(R.layout.activity_theme_item);
context = ThemeItemActivity.this;

themeItem= (ThemeItem) getIntent().getExtras().getSerializable("THEME_ITEM");

// отображениеназваниетемывтулбаре
setTitle(themeItem.getName());

// ui
rvThemeItem= findViewById(R.id.rv_theme_item);
tvThemeItemAbout= findViewById(R.id.tv_theme_item_about);
```

```
//
tvThemeItemAbout.setText(themeItem.getAbout());
tvThemeItemAbout.setMovementMethod(new ScrollingMovementMethod());

// кнопка "назад" втулбаре
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
getSupportActionBar().setDisplayShowHomeEnabled(true);


dataBaseConnection();

items = new ArrayList<>();

        Cursor cursorThread;
cursorThread = mDb.rawQuery("select * from item where theme_item_id = " +
themeItem.getId(), null);
cursorThread.moveToFirst();
while (!cursorThread.isAfterLast()) {

items.add(new Item(
cursorThread.getInt(0),
cursorThread.getInt(1),
cursorThread.getString(2),
cursorThread.getString(3)
            ));


cursorThread.moveToNext();
        }
cursorThread.close();

rvThemeItem.setHasFixedSize(true);
RecyclerView.LayoutManagerlayoutManager = new LinearLayoutManager(context);
rvThemeItem.setLayoutManager(layoutManager);

RvItemsAdapter adapter = new RvItemsAdapter(context, items);

rvThemeItem.setAdapter(adapter);


// обработчикнажатиянаэлементсписка

rvThemeItem.addOnItemTouchListener(new RecyclerViewTouchListener(context,
rvThemeItem, new RecyclerViewClickListener() {

@Override
public void onClick(View view, intposition) {
                Intent intent = new Intent(context, ItemActivity.class);
intent.putExtra("ITEM", items.get(position));
ActivityOptions options =
ActivityOptions.makeSceneTransitionAnimation((Activity) context);
startActivity(intent,options.toBundle());
            }

@Override
public void onLongClick(View view, intposition) {
            }

        }));

    }
```

```
private void dataBaseConnection(){
mDBHelper= new DataBaseHelper(context);
try {
mDBHelper.updateDataBase();
        } catch (IOExceptionmIOException) {
throw new Error("UnableToUpdateDatabase");
        }
try {
mDb= mDBHelper.getWritableDatabase();
        } catch (SQLExceptionmSQLException) {
throw mSQLException;
        }
    }

public void setAnimation(){
        Slide slide = new Slide();
slide.setSlideEdge(Gravity.LEFT);
slide.setDuration(400);
slide.setInterpolator(new DecelerateInterpolator());
getWindow().setExitTransition(slide);
getWindow().setEnterTransition(slide);
    }

@Override
public booleanonOptionsItemSelected(MenuItem item) {// когдапользователь
выбираетпунктменюпараметров, системавызываетэтотметод
switch (item.getItemId()) {
case android.R.id.home:  // поймать нажатиекнопкихоум , case – eeid
onBackPressed();  // возврат
return true;
default:
return super.onOptionsItemSelected(item);
        }
    }
}
```

Листинг Б.4 – Код класса «ITEMACTIVITY»

```
package com.medical.dictionary.Activities;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;
import android.os.Bundle;
import android.text.method.ScrollingMovementMethod;
import android.transition.Slide;
import android.view.Gravity;
import android.view.MenuItem;
import android.view.animation.DecelerateInterpolator;//скоростьилиграницы…
import android.widget.TextView;

import com.medical.dictionary.Models.Item;
import com.medical.dictionary.Models.ThemeItem;
import com.medical.dictionary.R;

public class ItemActivityextends AppCompatActivity {

private Context context;
private Item item;

private TextViewtvItemAbout;

@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setAnimation();
setContentView(R.layout.activity_item);
context = ItemActivity.this;

item = (Item) getIntent().getExtras().getSerializable("ITEM");//передачад-ых

// отображениеназваниетемывтулбаре
setTitle(item.getName());

// кнопка "назад" втулбаре

getSupportActionBar().setDisplayHomeAsUpEnabled(true);
getSupportActionBar().setDisplayShowHomeEnabled(true);


tvItemAbout= findViewById(R.id.tv_item_about);
tvItemAbout.setText(item.getAbout());
tvItemAbout.setMovementMethod(new ScrollingMovementMethod());

    }
public void setAnimation(){
        Slide slide = new Slide();
slide.setSlideEdge(Gravity.LEFT);
slide.setDuration(400);
slide.setInterpolator(new DecelerateInterpolator());
getWindow().setExitTransition(slide);
getWindow().setEnterTransition(slide);
    }

@Override
public booleanonOptionsItemSelected(MenuItem item) {
switch (item.getItemId()) {
case android.R.id.home:
onBackPressed();
return true;
default:
return super.onOptionsItemSelected(item);
        }
    }
}
```

Листинг Б.5 – Код класса «FACTSCTIVITY»

```
package com.medical.dictionary.Activities;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.app.Activity;
import android.app.ActivityOptions;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.transition.Slide;
import android.view.Gravity;
import android.view.MenuItem;
import android.view.View;
```

```
import android.view.animation.DecelerateInterpolator;

import com.medical.dictionary.Adapters.RecyclerViewClickListener;
import com.medical.dictionary.Adapters.RecyclerViewTouchListener;
import com.medical.dictionary.Adapters.RvFactsAdapter;
import com.medical.dictionary.Adapters.RvMainThemeAdapter;
import com.medical.dictionary.Models.Fact;
import com.medical.dictionary.Models.MainTheme;
import com.medical.dictionary.R;
import com.medical.dictionary.Utils.DataBaseHelper;

import java.io.IOException;
import java.util.ArrayList;

public class FactsActivityextends AppCompatActivity {
private Context context;
private RecyclerViewrvFacts;

private ArrayList<Fact>facts;
private DataBaseHelpermDBHelper;
private SQLiteDatabasemDb;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setAnimation();
setContentView(R.layout.activity_facts);

context = FactsActivity.this;

setTitle("Интересныефакты");

// кнопка "назад" втулбаре
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
getSupportActionBar().setDisplayShowHomeEnabled(true);


// ui
rvFacts= findViewById(R.id.rv_facts);

facts = new ArrayList<>();

getFacts();

    }

private void getFacts(){

// полученияфактовсбазыданных
dataBaseConnection();
        Cursor cursorThread;
cursorThread = mDb.rawQuery("select * from fact ORDER BY RANDOM() LIMIT 5",
null);
cursorThread.moveToFirst();
while (!cursorThread.isAfterLast()) {  // курсор; достижениеконцазаписи
facts.add(new Fact(
cursorThread.getInt(0),
cursorThread.getString(1),
cursorThread.getString(2)
            ));
cursorThread.moveToNext();
        }
cursorThread.close();
```

```
// отображениепятирандомныхфактов
rvFacts.setHasFixedSize(true);
RecyclerView.LayoutManagerlayoutManager = new LinearLayoutManager(context);
rvFacts.setLayoutManager(layoutManager);
RvFactsAdapter adapter = new RvFactsAdapter(context, facts);
rvFacts.setAdapter(adapter);

// обработчикнажатиянаэлементсписка
rvFacts.addOnItemTouchListener(new RecyclerViewTouchListener(context,
rvFacts, new RecyclerViewClickListener() {
@Override
public void onClick(View view, intposition) {
                Intent intent = new Intent(context, FactActivity.class);
intent.putExtra("FACT", facts.get(position));
ActivityOptions options =
ActivityOptions.makeSceneTransitionAnimation((Activity) context);
startActivity(intent,options.toBundle());
            }

@Override
public void onLongClick(View view, intposition) {
            }
        }));
    }
public void setAnimation(){
        Slide slide = new Slide();
slide.setSlideEdge(Gravity.LEFT);
slide.setDuration(400);
slide.setInterpolator(new DecelerateInterpolator());
getWindow().setExitTransition(slide);  // полноэкранныйрежимиличто-тотипа
getWindow().setEnterTransition(slide);
    }

@Override
public booleanonOptionsItemSelected(MenuItem item) {
switch (item.getItemId()) {
case android.R.id.home:
onBackPressed();
return true;
default:
return super.onOptionsItemSelected(item);
        }
    }

private void dataBaseConnection(){
mDBHelper= new DataBaseHelper(context);
try {
mDBHelper.updateDataBase();
        } catch (IOExceptionmIOException) {
throw new Error("UnableToUpdateDatabase");
        }

try {

mDb= mDBHelper.getWritableDatabase();
        } catch (SQLExceptionmSQLException) {
throw mSQLException;
        }
    }
}
```

Листинг Б.6 – Код класса «FACTACTIVITY»

```
package com.medical.dictionary.Activities;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.text.method.ScrollingMovementMethod;
import android.transition.Slide;
import android.view.Gravity;
import android.view.MenuItem;
import android.view.animation.DecelerateInterpolator;
import android.widget.TextView;

import com.medical.dictionary.Models.Fact;
import com.medical.dictionary.R;
public class FactActivityextends AppCompatActivity {
private Fact fact;

private TextViewtvTitle;
private TextViewtvAbout;

    @Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setAnimation();
setContentView(R.layout.activity_fact);

        fact = (Fact) getIntent().getExtras().getSerializable("FACT"); (7)

setTitle(fact.getTitle());

// кнопка "назад" втулбаре

getSupportActionBar().setDisplayHomeAsUpEnabled(true);

getSupportActionBar().setDisplayShowHomeEnabled(true);

// ui
tvTitle = findViewById(R.id.tv_fact_title);
tvAbout = findViewById(R.id.tv_fact_about);

// отображение
tvTitle.setText(fact.getTitle());
tvAbout.setText(fact.getAbout());
tvAbout.setMovementMethod(new ScrollingMovementMethod());
    }

public void setAnimation(){
        Slide slide = new Slide();
slide.setSlideEdge(Gravity.LEFT);
slide.setDuration(400);
slide.setInterpolator(new DecelerateInterpolator());
getWindow().setExitTransition(slide);
getWindow().setEnterTransition(slide);
    }

    @Override
public booleanonOptionsItemSelected(MenuItem item) {
switch (item.getItemId()) {
case android.R.id.home:
onBackPressed();
return true;
```

```
default:
return super.onOptionsItemSelected(item);
        }
    }
}
```

Листинг Б.7 – Код класса «AUTHACTIVITY»

```
package com.medical.dictionary.Activities;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.medical.dictionary.R;

public class AuthActivity extends AppCompatActivity {

    private Context context;
    private EditText edtLogin;
    private EditText edtPassword;
    private Button btnAuth;
    private static final String APP_PREFERENCES = "SETTINGS";
    private static final String LOGIN = "LOGIN";
    private SharedPreferences sharedPreferences;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_auth);
        context = AuthActivity.this;
        sharedPreferences = getSharedPreferences(APP_PREFERENCES,
MODE_PRIVATE);
        edtLogin = findViewById(R.id.edt_login);
        edtPassword = findViewById(R.id.edt_password);
        btnAuth = findViewById(R.id.btn_login);
        btnAuth.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String password = edtPassword.getText().toString();
                String pat = "^.*(?=.{8,})(?=..*[0-9])(?=.*[a-z])(?=.*[A-
Z])(?=.*[@#${}()<>*+=!?^|%&]).*$";
                if (password.matches(pat)) {
                    String login = edtLogin.getText().toString();
                    SharedPreferences.Editor editor =
sharedPreferences.edit();
                    editor.putString(LOGIN, login);
                    editor.commit();
                    startActivity(new Intent(context, MainActivity.class));
                }
                else {
                    edtPassword.setError("Ошибка");
                    Toast toast = Toast.makeText(context, "Пароль должен быть
не менее 8 символов,содержать 1 заглавную букву, 1 цифру и 1 спец.символ.",
Toast.LENGTH_LONG);
```

```
                    toast.setGravity(Gravity.CENTER, 0, 0);
                    toast.show();
                }
            }
        });
    }
}
```

Листинг Б.8 – Код класса «SPLASHACTIVITY»

```
package com.medical.dictionary.Activities;

import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.os.Handler;


public class SplashActivity extends AppCompatActivity {

    private int SPLASH_TIME_OUT = 200;
    private static final String APP_PREFERENCES = "SETTINGS";
    private static final String LOGIN = "LOGIN";
    private SharedPreferences sharedPreferences;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        sharedPreferences = getSharedPreferences(APP_PREFERENCES,
MODE_PRIVATE);
        final String login  = sharedPreferences.getString(LOGIN, null);

        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                if(login != null){
                    Intent intent = new Intent(SplashActivity.this,
MainActivity.class);
                    startActivity(intent);
                    finish();
                }else {
                    startActivity(new Intent(SplashActivity.this,
AuthActivity.class));
                    finish();
                }
            }
        }, SPLASH_TIME_OUT);
    }
}
```

Листинг Б.9 – Код класса «DataBaseHelper»

```
package com.medical.dictionary.Utils;

import android.content.Context;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import java.io.File;
import java.io.FileOutputStream;
```

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

public class DataBaseHelper extends SQLiteOpenHelper {
    private static String DB_NAME = "dictionary.db";
    private static String DB_PATH = "";
    private static final int DB_VERSION = 1;

    private SQLiteDatabase mDataBase;
    private final Context mContext;
    private boolean mNeedUpdate = false;

    public DataBaseHelper(Context context) {
        super(context, DB_NAME, null, DB_VERSION);
        DB_PATH = context.getApplicationInfo().dataDir + "/databases/";
        this.mContext = context;
        copyDataBase();
        this.getReadableDatabase();
    }

    public void updateDataBase() throws IOException {
        if (mNeedUpdate) {
            File dbFile = new File(DB_PATH + DB_NAME);
            if (dbFile.exists())
                dbFile.delete();
            copyDataBase();
            mNeedUpdate = false;
        }
    }

    private boolean checkDataBase() {
        File dbFile = new File(DB_PATH + DB_NAME);
        return dbFile.exists();
    }

    private void copyDataBase() {
        if (!checkDataBase()) {
            this.getReadableDatabase();
            this.close();
            try {
                copyDBFile();
            } catch (IOException mIOException) {
                throw new Error("ErrorCopyingDataBase");
            }
        }
    }

    private void copyDBFile() throws IOException {
        InputStream mInput = mContext.getAssets().open(DB_NAME);
        //InputStream mInput =
mContext.getResources().openRawResource(R.raw.info);
        OutputStream mOutput = new FileOutputStream(DB_PATH + DB_NAME);
        byte[] mBuffer = new byte[1024];
        int mLength;
        while ((mLength = mInput.read(mBuffer)) > 0)
            mOutput.write(mBuffer, 0, mLength);
        mOutput.flush();
        mOutput.close();
```

```
        mInput.close();
    }

    public boolean openDataBase() throws SQLException {
        mDataBase = SQLiteDatabase.openDatabase(DB_PATH + DB_NAME, null,
SQLiteDatabase.CREATE_IF_NECESSARY);
        return mDataBase != null;
    }

    @Override
    public synchronized void close() {
        if (mDataBase != null)
            mDataBase.close();
        super.close();
    }

    @Override
    public void onCreate(SQLiteDatabase db) {

    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
{
        if (newVersion > oldVersion)
            mNeedUpdate = true;
    }
}
```

Листинг Б.10 – Код класса «RecyclerViewTouchListener»

```
package com.medical.dictionary.Adapters;

import android.content.Context;
import android.view.GestureDetector;
import android.view.MotionEvent;
import android.view.View;

import androidx.recyclerview.widget.RecyclerView;


public class RecyclerViewTouchListener implements
RecyclerView.OnItemTouchListener{

    private GestureDetector gestureDetector;
    private RecyclerViewClickListener clickListener;

    public RecyclerViewTouchListener(Context context, final RecyclerView
recyclerView, final RecyclerViewClickListener clickListener) {
        this.clickListener = clickListener;
        gestureDetector = new GestureDetector(context, new
GestureDetector.SimpleOnGestureListener() {
            @Override
            public boolean onSingleTapUp(MotionEvent e) {
                return true;
            }

            @Override
            public void onLongPress(MotionEvent e) {
```

```
                View child = recyclerView.findChildViewUnder(e.getX(),
e.getY());
                if (child != null && clickListener != null) {
                    clickListener.onLongClick(child,
recyclerView.getChildPosition(child));
                }
            }
        });
    }

    @Override
    public boolean onInterceptTouchEvent(RecyclerView rv, MotionEvent e) {

        View child = rv.findChildViewUnder(e.getX(), e.getY());
        if (child != null && clickListener != null &&
gestureDetector.onTouchEvent(e)) {
            clickListener.onClick(child, rv.getChildPosition(child));
        }
        return false;
    }

    @Override
    public void onTouchEvent(RecyclerView rv, MotionEvent e) {
    }

    @Override
    public void onRequestDisallowInterceptTouchEvent(boolean
disallowIntercept) {

    }
}
```

Листинг Б.11 – Код класса «RvFactsAdapter»

```
package com.medical.dictionary.Adapters;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.recyclerview.widget.RecyclerView;

import com.medical.dictionary.Models.Fact;
import com.medical.dictionary.Models.MainTheme;
import com.medical.dictionary.R;

import java.util.ArrayList;


public class RvFactsAdapter extends
RecyclerView.Adapter<RvFactsAdapter.ViewHolder> {

    private Context context;
    private ArrayList<Fact> items;

    public RvFactsAdapter(Context context, ArrayList<Fact> items) {
```

```
        this.context = context;
        this.items = items;
    }

    @Override
    public RvFactsAdapter.ViewHolder onCreateViewHolder(ViewGroup viewGroup,
int i) {
        View view =
LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.list_fact,
viewGroup, false);
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(final ViewHolder viewHolder, int i) {
        viewHolder.tvTitle.setText(items.get(i).getTitle());
        viewHolder.tvAbout.setText(items.get(i).getAbout());
    }

    @Override
    public int getItemCount() {
        return items.size();
    }

    public class ViewHolder extends RecyclerView.ViewHolder{

        TextView tvTitle;
        TextView tvAbout;

        public ViewHolder(View view) {
            super(view);
            tvTitle = view.findViewById(R.id.tv_fact_title);
            tvAbout = view.findViewById(R.id.tv_fact_about);
        }
    }
}
```

Листинг Б.12 – Код класса «RvItemsAdapter»

```
package com.medical.dictionary.Adapters;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.recyclerview.widget.RecyclerView;

import com.medical.dictionary.Models.Item;
import com.medical.dictionary.Models.ThemeItem;
import com.medical.dictionary.R;

import java.util.ArrayList;


public class RvItemsAdapter extends
RecyclerView.Adapter<RvItemsAdapter.ViewHolder> {
```

```
    private Context context;
    private ArrayList<Item> items;

    public RvItemsAdapter(Context context, ArrayList<Item> items) {
        this.context = context;
        this.items = items;
    }

    @Override
    public RvItemsAdapter.ViewHolder onCreateViewHolder(ViewGroup viewGroup,
int i) {
        View view =
LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.list_theme_item,
viewGroup, false);
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(final ViewHolder viewHolder, int i) {
        viewHolder.tvMainThemeItemName.setText(items.get(i).getName());
    }

    @Override
    public int getItemCount() {
        return items.size();
    }

    public class ViewHolder extends RecyclerView.ViewHolder{

        TextView tvMainThemeItemName;

        public ViewHolder(View view) {
            super(view);
            tvMainThemeItemName =
view.findViewById(R.id.tv_main_theme_item_name);
        }
    }
}
```

Листинг Б.13 – Код класса «RvMainThemeAdapter»

```
package com.medical.dictionary.Adapters;

import android.content.Context;
import android.graphics.drawable.Drawable;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import androidx.recyclerview.widget.RecyclerView;

import com.medical.dictionary.Models.MainTheme;
import com.medical.dictionary.R;

import java.util.ArrayList;


public class RvMainThemeAdapter extends
```

```
RecyclerView.Adapter<RvMainThemeAdapter.ViewHolder> {

    private Context context;
    private ArrayList<MainTheme> items;

    public RvMainThemeAdapter(Context context, ArrayList<MainTheme> items) {
        this.context = context;
        this.items = items;
    }

    @Override
    public RvMainThemeAdapter.ViewHolder onCreateViewHolder(ViewGroup
viewGroup, int i) {
        View view =
LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.list_main_item,
viewGroup, false);
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(final ViewHolder viewHolder, int i) {
        viewHolder.tvMainThemeName.setText(items.get(i).getName());
        //viewHolder.ivMainThemePic
        switch (items.get(i).getId()){

            case 1:

viewHolder.ivMainThemePic.setImageDrawable(context.getResources().getDrawable
(R.drawable.main_theme_1));
                break;
            case 2:

viewHolder.ivMainThemePic.setImageDrawable(context.getResources().getDrawable
(R.drawable.main_theme_2));
                break;
            case 3:

viewHolder.ivMainThemePic.setImageDrawable(context.getResources().getDrawable
(R.drawable.main_theme_3));
                break;
            case 4:

viewHolder.ivMainThemePic.setImageDrawable(context.getResources().getDrawable
(R.drawable.main_theme_4));
                break;
            case 5:

viewHolder.ivMainThemePic.setImageDrawable(context.getResources().getDrawable
(R.drawable.main_theme_5));
                break;
            case 6:

viewHolder.ivMainThemePic.setImageDrawable(context.getResources().getDrawable
(R.drawable.main_theme_6));
                break;
            case 7:

viewHolder.ivMainThemePic.setImageDrawable(context.getResources().getDrawable
(R.drawable.main_theme_7));
```

```
                break;

        }

    }

    @Override
    public int getItemCount() {
        return items.size();
    }

    public class ViewHolder extends RecyclerView.ViewHolder{

        TextView tvMainThemeName;
        ImageView ivMainThemePic;

        public ViewHolder(View view) {
            super(view);
            tvMainThemeName = view.findViewById(R.id.tv_main_theme_name);
            ivMainThemePic = view.findViewById(R.id.tv_main_theme_pic);
        }
    }
}
```

Листинг Б.14 – Код класса «RvThemeItemsAdapter»

```
package com.medical.dictionary.Adapters;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.recyclerview.widget.RecyclerView;

import com.medical.dictionary.Models.MainTheme;
import com.medical.dictionary.Models.ThemeItem;
import com.medical.dictionary.R;

import java.util.ArrayList;


public class RvThemeItemsAdapter extends
RecyclerView.Adapter<RvThemeItemsAdapter.ViewHolder> {

    private Context context;
    private ArrayList<ThemeItem> items;

    public RvThemeItemsAdapter(Context context, ArrayList<ThemeItem> items) {
        this.context = context;
        this.items = items;
    }

    @Override
    public RvThemeItemsAdapter.ViewHolder onCreateViewHolder(ViewGroup
viewGroup, int i) {
        View view =
LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.list_theme_item,
```

```
viewGroup, false);
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(final ViewHolder viewHolder, int i) {
        viewHolder.tvMainThemeItemName.setText(items.get(i).getName());
    }

    @Override
    public int getItemCount() {
        return items.size();
    }

    public class ViewHolder extends RecyclerView.ViewHolder{
                TextView tvMainThemeItemName;
                public ViewHolder(View view) {
            super(view);
            tvMainThemeItemName =
view.findViewById(R.id.tv_main_theme_item_name);
        }
    }
}
```