**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 7](#_Toc33617548)

[**1 АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ** 9](#_Toc33617549)

[**СРЕДСТВУ И ПОСТАНОВКА ЗАДАЧ** 9](#_Toc33617550)

[**1.2** **Обзор существующих программных средств по теме** 10](#_Toc33617551)

[**дипломного проекта** 10](#_Toc33617552)

[**1.3 Обоснование выбора языка программирования и средств** 11](#_Toc33617553)

[**разработки** 11](#_Toc33617554)

[**1.4 Постановка задач по разработке программного средства** 19](#_Toc33617555)

[**2 РАЗРАБОТКА ИГРОВОГО АРКАДНОГО** 20](#_Toc33617556)

[**ПРОГРАММНОГО СРЕДСТВА** 20](#_Toc33617557)

[**2.1 Разработка архитектуры программного средства** 20](#_Toc33617558)

[**2.2 Разработка объектной модели программного средства** 22](#_Toc33617559)

[**2.3 Разработка алгоритмов функционирования программного** 25](#_Toc33617560)

[**средства** 25](#_Toc33617561)

[**2.4 Разработка и обоснование пользовательского интерфейса** 38](#_Toc33617562)

[**программного средства** 38](#_Toc33617563)

[**3 ИНЖЕНЕРНЫЕ РАСЧЕТЫ, ИСПОЛЬЗУЕМЫЕ** 42](#_Toc33617564)

[**В ПРОГРАММНОМ СРЕДСТВЕ** 42](#_Toc33617565)

[**3.1 Оценка файловой структуры игрового программного средства** 42](#_Toc33617566)

[**3.2 Анализ памяти, используемой программным средством** 45](#_Toc33617567)

[**4 ЭКСПЛУАТАЦИЯ ПРОГРАММНОГО СРЕДСТВА** 49](#_Toc33617568)

[**4.1 Ввод в эксплуатацию программного средства** 49](#_Toc33617569)

[**4.2 Руководство к пользованию разработанным программным** 51](#_Toc33617570)

[**средством** 51](#_Toc33617571)

[**5 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ** 53](#_Toc33617572)

[**РАЗРАБОТКИ ИГРОВОГО АРКАДНОГО** 53](#_Toc33617573)

[**ПРОГРАММНОГО СРЕДСТВА «RUN OUT» ПОД** 53](#_Toc33617574)

[**ОПЕРАЦИОННУЮ СИСТЕМУ ANDROID** 53](#_Toc33617575)

[**5.1 Краткая характеристика программного продукта** 53](#_Toc33617576)

[**5.2 Расчет затрат на разработку и отпускной цены программного** 53](#_Toc33617577)

[**продукта** 53](#_Toc33617578)

[**5.3 Расчет экономической эффективности реализации на рынке** 56](#_Toc33617579)

[**игрового аркадного программного средства «Run out»** 56](#_Toc33617580)

[**под операционную систему Android** 56](#_Toc33617581)

[**ЗАКЛЮЧЕНИЕ** 59](#_Toc33617582)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 59](#_Toc33617583)

[**ПРИЛОЖЕНИЕ А** 61](#_Toc33617584)

[**ПРИЛОЖЕНИЕ** **Б (обязательное)** 62](#_Toc33617585)

[**ПРИЛОЖЕНИЕ В (обязательное)** 63](#_Toc33617586)

# **ВВЕДЕНИЕ**

На сегодняшний день такое заболевание, как сахарный диабет, приобретает все большее и большее распространение. Это обусловлено изменением стиля жизни современного человека, уменьшением подвижности его тела в связи с преобладанием умственного труда над физическим, потреблением нездоровой пищи и развитием других вредных привычек.

Диабет — это хроническая болезнь, развивающаяся в тех случаях, когда поджелудочная железа не вырабатывает достаточно инсулина или когда организм не может эффективно использовать вырабатываемый им инсулин. Инсулин — это гормон, регулирующий уровень содержания сахара в крови. Общим результатом неконтролируемого диабета является гипергликемия, или повышенный уровень содержания сахара в крови, что со временем приводит к серьезному повреждению многих систем организма, особенно нервов и кровеносных сосудов. [1] Число людей с диабетом возросло со 108 миллионов в 1980 году до 422 миллионов в 2014 году. Наблюдая такую динамику распространения болезни, невозможно легкомысленно относиться к этому явлению.

За последние 10 лет смартфоны прочно закрепились в повседневной жизни людей, так как эти устройства поддерживают огромное количество функций и заменили собой множество других гаджетов, которые ранее приходилось приобретать отдельно, такие как: калькулятор, камера, *mp3*-плеер и множество других устройств. Современные смартфоны способны заменить даже персональный компьютер. Такую универсальность смартфоны имеют благодаря наличию различных приложений и возможности расширять функционал устройства благодаря установке дополнительных программ. Если принять во внимание этот момент, то хорошим решением является использование смартфона и в оздоровительных целях, а именно использование устройства в качестве трекера здоровья, куда человек будет заносить данные о своем здоровье и, анализируя предоставляемую приложением статистику, корректировать свой образ жизни для того, чтобы избегать причин возникновения заболевания.

Объектом исследования являются существующие варианты приложений для мобильных устройств, позволяющие пользователю отслеживать свою статистику здоровья.

Предметом исследования являются способы улучшения качества наблюдения человека, болеющего сахарным диабетом, за своим состоянием здоровья.

Целью данной работы является разработка мобильного приложения для операционной системы *Android*, которое позволит пользователю вести учет потребляемых им лекарственных средств, калорий, количества времени физической активности, уровня глюкозы в крови. Также приложение должно иметь функции напоминания пользователю о времени принятия необходимых лекарственных средств, сигнализирования пользователю о превышении или занижении нормы потребляемых микроэлементов и уровня глюкозы в крови. Также для того, чтобы в процесс наблюдения здоровья было легче вовлечь детей, приложение должно иметь игровую форму с вознаграждением пользователя за постоянный ввод данных.

Для выполнения приведённых целей необходимо создать *RESTful API* серверное приложение с подключением к базе данных для обеспечения возможности получения клиентским приложением данных. Далее реализовать клиентское приложение с удобным пользовательским интерфейсом, которое бы обращалось к серверу за данными, получало их, сохраняло в памяти смартфона и предоставляло возможности для выполнения описанных выше функций.

Мобильное приложение — программное обеспечение, предназначенное для работы на смартфонах, планшетах и других мобильных устройствах. Многие мобильные приложения предустановлены на самом устройстве или могут быть загружены на него из онлайн магазинов приложений, таких как *App Store, Google Play, Samsung Store* и других [2].

Дипломный проект выполнен самостоятельно, проверен в системе «Антиплагиат». Процент оригинальности составляет ИКС%. Цитирования обозначены ссылками на публикации, указанными в «Списке использованных источников».

Скриншот приведен в приложении А.

# **1 АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ**

# **СРЕДСТВУ И ПОСТАНОВКА ЗАДАЧ**

* 1. **Анализ и описание функциональных возможностей**

**программного средства**

Основными функциями разрабатываемого в данном проекте приложения является система учета данных о состоянии здоровья пользователя, сбор статистических данных, вносимых им, осуществление напоминаний пользователю о необходимости принятия лекарственных средств, отображение среднестатистических данных на основе информации, введенной пользователем, выдача подробной детализированной информации, которая была внесена в течение определенного промежутка времени. Также важной функцией является создание учетной записи пользователя с привязкой к адресу электронной почты и номеру телефона, что позволит пользователю осуществлять доступ к приложению на более высоком уровне безопасности. В случае утери пароля пользователь сможет продолжать осуществлять доступ к приложению посредством привязанного номера телефона.

Анализируя существующие аналоги разрабатываемого приложения, можно сделать вывод о том, что вышеописанные функции присутствуют во всех похожих приложениях в том или ином виде. Все рассмотренные аналоги имеют различный интерфейс, но практически идентичны по функциям. Каждое приложение имеет функции сбора данных, напоминания, оценку состояния пользователя на основе введенных данных и т.д.

Все вышеупомянутые функции безусловно очень нужны и приносят большую пользу людям, но если немного проанализировать потенциальную аудиторию пользователей такого рода приложений, то можно обнаружить, что большую долю составляют люди подросткового и даже детского возраста. По данным Международной диабетической федерации на 2016 год, только в Европе проживает более 140 тысяч детей в возрасте до 14 лет, имеющих диабет первого типа. [3] Существующие приложения учета здоровья имеют интерфейс и логику, понятные для взрослого человека. Но далеко не каждые ребенок или подросток сможет разобраться с тем, как пользоваться такого рода приложением. Поэтому в разрабатываемом в данном проекте приложении было решено добавить игровую логику с виртуальным персонажем, взаимодействуя с которым, пользователю будет проще понять суть приложения и выполнять необходимые действия. Было решено создать функцию чат-бота для симуляции общения пользователя и виртуального персонажа, а также систему вознаграждения пользователя за ввод данных.

Приложение, которое должно получиться в результате выполнения дипломного проекта на данный момент не имеет аналогов в магазинах приложений.

## **Обзор существующих программных средств по теме**

## **дипломного проекта**

Если начать поиск похожих программных средств по теме дипломного проекта в магазинах приложений, то можно найти достаточно большое количество продуктов, которые очень схожи по своим функциям. Рассмотрим некоторые из них.

Первое из обнаруженных приложений – «*Diabetes:M*». Скриншот одного из экранов можно видеть ниже на рисунке 1.1.



Рисунок 1.1 – Скриншот приложения «*Diabetes:M*»

Данное приложение имеет интеграцию с сервисом *Google Fit*, который отслеживает данные о здоровье пользователя со смартфона и других носимых устройств. Также можно вводить в нее лабораторные исследования крови и мочи, записывать давление, задавать значения на занятия спортом и данные о болезни. Данное приложение обладает следующими функциями:

* калькулятор. Программа рассчитывает ту дозу инсулина, которую необходимо ввести, опираясь на заданные ей значения
* расчет активного инсулина в организме
* дневник питания
* наглядная визуализация данных дневника в виде графиков и диаграмм за период от недели до трех месяцев.

Рассмотрим другое приложение – «*DiaMeter*». Это не менее удобное приложение для ведения записей об уровне сахара, количестве съеденных вами хлебных единиц, инъекциях короткого и продлённого инсулина, а также о самочувствии в целом. Есть наглядная статистика и возможность синхронизации всех данных в облаке. Дополнительно DiaMeter предлагает ряд интерактивных статей, в которых можно найти немало полезной информации о правильном питании при диабете, физических нагрузках, влиянии алкоголя и другом. Скриншот данного приложения изображен на рисунке 1.2.

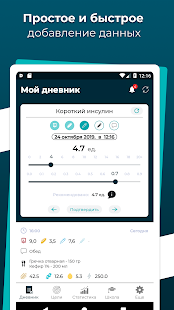


Рисунок 1.2 – Скриншот приложения «*DiaMeter*»

Оба приложения, рассмотренные выше, очень хороши, удобны и полезны для взрослого пользователя, который осознает свое заболевание и достаточно замотивирован в лечении и наблюдении за своим состоянием здоровья. Однако рассмотренные приложения будут очень сложны в восприятии более юной аудиторией.

## **1.3 Обоснование выбора языка программирования и средств**

## **разработки**

При разработке сложных программных средств под мобильные опера-ционные системы следует с особым вниманием подходить к выбору средств разработки, так как от этого зависит скорость и качество процесса создания программного средства.

Для выбора подходящей среды разработки необходимо учесть ряд опре-делённых факторов, которые могут оказать сильное влияние на разработку:

– интерфейс. Этот первый компонент, с которым сталкивается пользо-ватель после установки и который формирует первое впечатление о про-грамме, и, на основании которого, может быть осуществлен окончательный выбор. Здесь оценивается не только общий дизайн, хотя, безусловно, он тоже сказывается определенным образом, но и удобство расположения и настройки таких компонент как окно исходного кода, окно проекта;

– настройка. Соответственно после установки и первого запуска среды разработки выполняется ее настройка, т.е. указываются пути, по которым рас-полагаются установленные *SDK*, *DOCS*, *J*2*EE*. В этом компоненте, как пра-вило, существенных различий не наблюдается. Более того, современные сред-ства, как правило, самостоятельно определяют установленные компоненты;

– редактор кода. Настройка отображения исходных кодов, как правило, тоже не отличается разнообразием, в любом средстве легко можно настроить кеглю и ее размер, а также цвет. Немаловажным преимуществом является наличие помощника, когда, при «зависании» мышки на любой переменной или методе всплывает довольно подробный *ToolTip* (контекстное окно) в ко-тором развернуто, описаны все параметры объекта. Также есть масса прият-ных мелочей, вывод нумерации строк, отображение структуры класса, показ символов абзаца, проверка орфографии.

На сегодняшний день среди мобильных операционных систем получили наибольшее распространение *Android* и *iOS*. Большинство мобильных устройств работают под одной из данных систем. Согласно мировой статистике [4], 74.3% пользователей смартфонов используют операционную систему *Android* и 24.7% – *iOS*. Благодаря тому, что преобладает всего две операционные системы, широкое распространение сегодня получают различные кроссплатформенные фреймворки, позволяющие из одной и той же кодовой базы создать одновременно приложение как под одну операционную систему, так и под другую. Использование такого подхода позволяет сэкономить не только временные ресурсы, но также и финансовые. Производительность таких кроссплатформенных приложений очень зависит от выбранной технологии. Лучшие технологии для кроссплатформенной разработки на сегодняшний день мало чем уступают нативным приложениям в производительности. Исходя из этого было решено для разработки приложения данного дипломного проекта использовать кроссплатформенный фреймворк.

Одним из наиболее популярных фреймворков для разработки мобильных кроссплатформенных приложений является *React Native* [5]. *React Native* – фреймворк с открытым исходным кодом, разработанный для создания кроссплатформенных мобильных приложений для операционных систем *iOS* и *Android*. Данная платформа использует язык *JavaScript* для создания приложений. Структура платформы отображена ниже на рисунке 1.3.

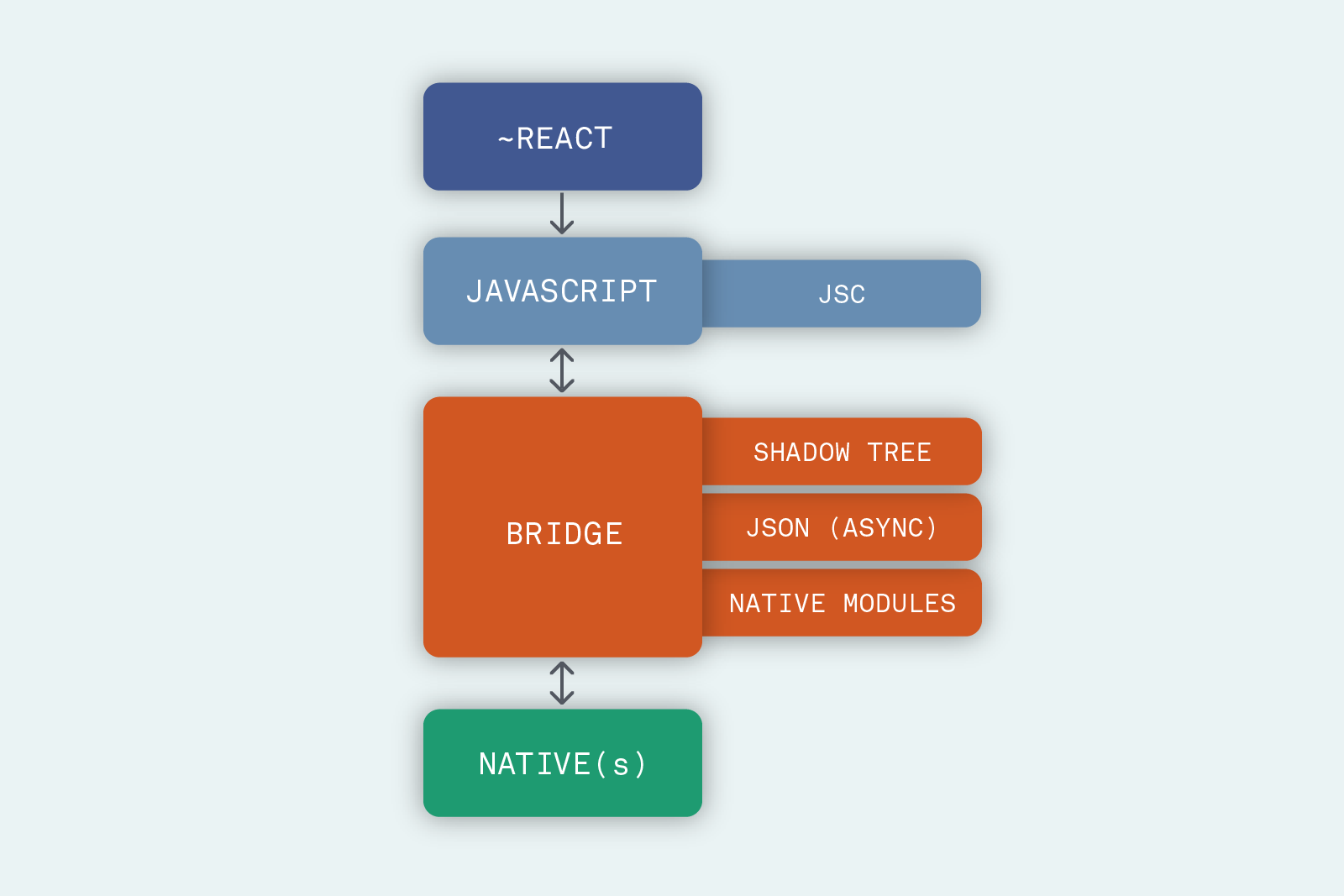


Рисунок 1.3 – Структура платформы *React Native* [5]

*React Native* использует библиотеку *React* в качестве основы для написания исходного кода приложения. Код, написанный с использованием этой библиотеки компилируется в один файл, так называемый *bundle*.

Также на рисунке можно видеть, что собранный *JavaScript*-код взаимодействует с «нативной» платформой посредством моста, который содержит в себе всю логику работы фреймворка.

Приложение, разработанное с использованием *React Native*, очень сложно отличить от разработанного с использованием родных платформе инструментов. Это обусловлено строением архитектуры данного фреймворка и его работы. Полностью готовое *React Native* приложение представляет собой «нативное» *Android* или *iOS* приложение, которое при открытии передает управление *JavaScript*-потоку, содержащему всю бизнес-логику приложения, а также пользовательский интерфейс. Данный поток может управлять работой родных для мобильной платформы компонентов. Например, открыть модальное окно, добавить *Drawer*-меню и так далее. Фреймворк также поддерживает обработку специфических для каждой платформы жестов и анимации.



Рисунок 1.4 – Пример программного кода

С точки зрения разработчика, пишущего непосредственно код приложения, *React Native* практически не отличается от библиотеки *React*. Эта библиотека является отличным примером реактивного программирования, ориентированного на реакцию к определенным событиям. Также особенностью *React* и *React Native* является *JSX*-разметка компонентов, используемая при построении пользовательского интерфейса. Данная разметка похожа на разметку *html*-документа, что облегчает освоение данных платформ разработчиками, которые имели опыт работы в *web*-разработке. На рисунке 1.4 отображен пример программного кода, написанного при использовании платформы *React Native*.

В сущности хакерской культуры Facebook, React Native начинался как проект хакатона летом 2013 года. Подобно React , React Native казался смелой нетрадиционной идеей. Не было ясно, что это действительно сработает. Как работали сенсорные переговоры между JS и собственными ScrollViews? Как насчет производительности, а как насчет отладки? Ни одна из этих проблем не помешала инженерам сосредоточиться и двигаться вперед.

Менеджер рекламы Facebook для iOS вышел в феврале 2015 года, менее чем через шесть месяцев после того, как команда разработчиков начала работать над ним. Одновременно все, кто сосредоточился на JS или iOS, работали над открытым исходным кодом реализации iOS. В январе 2015 года на React.js Conf состоялся первый публичный предварительный просмотр , а на конференции разработчиков F8 в марте 2015 года был открыт код для всех .

Сразу после этого инженеры по продукту Ads Manager начали переносить свой код JavaScript на Android, тесно сотрудничая с командой React Native Android в Лондоне. Facebook не стремился к совместному использованию кода между платформами при работе с iOS Ads Manager, но ожидалось, что это может быть хорошим побочным эффектом использования React Native. Когда Android Ads Manager был готов к поставке было понятно, что около 85 процентов кода было поделено между двумя приложениями.

В июне 2015 года, после трех месяцев разработки и месяца внутренних собеседований, была выпущена первая версия менеджера рекламы на Android . Учитывая, как взлетела React Native для iOS, Facebook сразу же переключили наше внимание на открытый исходный код React Native для Android, и ожидали большой интерес к выпуску Android. В конце концов, необходимость создавать одно и то же приложение отдельно для каждой платформы - проблема в отрасли, и это было известно из опыта работы с Менеджером рекламы, что React Native может помочь с этим. [6]

Подобно запуску iOS, Facebook хотел выпустить React Native для Android как можно раньше, чтобы получить обратную связь. Поэтому они начали только с небольшого количества представлений и модулей (Text, Image, ScrollView, Network, AsyncStorage и некоторых других). 14 сентября было выдвинуто ядро ​​среды выполнения Android и начальный набор модулей Android на GitHub и npm. И именно так React Native версии 0.11 был первым выпуском с поддержкой Android. Были добавлены следующие модули Android с тех пор, как открыли исходный код, приблизив паритет API к iOS: оповещения, AppState, CameraRoll, буфер обмена, указатели даты и времени, геолокация, намерение, модальные, NetInfo, извлечение для обновления представления, средство выбора , слайдер, просмотр пейджера и веб-просмотр.

Само собой разумеется, что принятие, которое последовало за пределами Facebook, очень вдохновило на дальнейшую работу в команде React Native.

На текущий момент последней стабильной версией React Native является 0.61.5 и с момента первого релиза изменилось многое. Но одно можно утверждать точно – React Native развивается и учитывает свои прошлые ошибки двигаясь вперед.

Платформа *React Native* обладает достаточно большим количеством достоинств. Благодаря существованию этих достоинств некоторые популярные приложения, таких как *Instagram, Airbnb, Walmart, UberEats, Tesla, Skype* и другие, уже активно используют этот фреймворк. *React Native* предоставляет все основные и передовые методы разработки гибридных мобильных приложений. Перейдем к рассмотрению некоторых причин, по которым использование *React Native* для разработки мобильных приложений является хорошим решением. [7]

**1.3.1** Достоинства *React Native*

**1.3.1.1** Единая кодовая база для *Android* и *iOS*

*React Native* позволяет писать код один раз и запускать как на *Android*, так и на *iOS*. Кодовая база будет одинаковой для обеих платформ, поэтому вам не придется тратить деньги и время на два проекта разработки.

Поскольку *React Native* имеет открытый исходный код и способствует повторному использованию, вы можете повторно использовать компоненты в любое время на любом уровне, не переписывая его и не перекомпилируя приложение.

**1.3.1.2** Возможность повторного использования

Теперь поговорим о возможности повторного использования. *React Native* приносит блоки, которые состоят из повторно используемых «нативных компонентов». Почти все компоненты *Android* и *iOS* имеют свои аналоги в *React Native*. Все эти компоненты компилируются непосредственно в нативные приложения и, таким образом, позволяют разработчикам поддерживать внешний вид платформы.

Ранее в гибридных приложениях было невозможно получить структуру, специфичную для компонента. Но с *React Native* эта структура позволяет разработчикам создавать приложения, используя гибкие веб-методологии.

С другой стороны, *React Native* также позволяет разработчикам добавлять компоненты *«native»* в код своих существующих приложений. Им не придется кодировать с нуля, чтобы преобразовать существующее мобильное приложение в приложение *React Native*.

**1.3.1.3** Производительность

*React Native* сокращает цикл разработки, позволяет разрабатывать приложения очень быстро и позволяет доставлять приложения максимально быстро. Он использует библиотеку *ReactJS UI*, разработанную *Facebook* для пользовательских интерфейсов. В то время как собственные приложения работают на центральном процессоре, *React Native* использует графический процессор.

Он поставляется с уникальной функцией под названием «живая перезагрузка», которая позволяет просматривать последние изменения кода сразу, разбивая экран на две части. Первый показывает код, а второй показывает результат кода в кадре мобильного телефона. В целом, с помощью *React Native* компании могут сократить расходы на разработку почти на 50% без ущерба для качества или производительности.

**1.3.1.4** Ориентация на пользовательский интерфейс

*React Native* – полностью ориентированный на пользовательский интерфейс. Декларативный *API* значительно упрощает понимание ваших требований и прогнозирование пользовательского интерфейса. Декларативный стиль позволяет вам контролировать поток и состояние в вашем приложении, говоря: «Это должно выглядеть так». Вам больше не придется беспокоиться о деталях реализации.

**1.3.1.5** Совместимость с сторонними плагинами

Добавление аппаратных возможностей устройства в приложение является одним из наиболее распространенных требований в наши дни. В отличие от функций *WebView, React Native* позволяет напрямую связать плагин с собственным модулем через инфраструктуру. Это приводит к более плавному запуску приложения, более быстрой загрузке и меньшим требованиям к памяти.

**1.3.2** Сравнение *React Native* с другими платформами

Для разработки приложения был выбран кросс-платформенный фреймворк *React Native*. По этому для сравнения с данной технологией, был выбран *Flutter*.

***Flutter*** — *SDK* с открытым исходным кодом для создания мобильных приложений от компании *Google*. Он используется для разработки приложений под *Android* и *IOS*, а также это пока единственный способ разработки приложений под *Google Fuchsia*. [8]

*React Native* создан для приложений на *IOS*, *Android* и *Windows*, при этом использует язык программирования *JavaScript* и библиотеку *React.js* как главное средство разработки. Фреймворк был создан и поддерживается компанией *Facebook*, которые разработали *React.js*.

*Flutter* в свою очередь создан для приложений *Android*, *IOS* и *Fuchsia*, в отличии от *React Native*, для разработки используется язык программирования *Dart*, который также служит для веб-программирования.

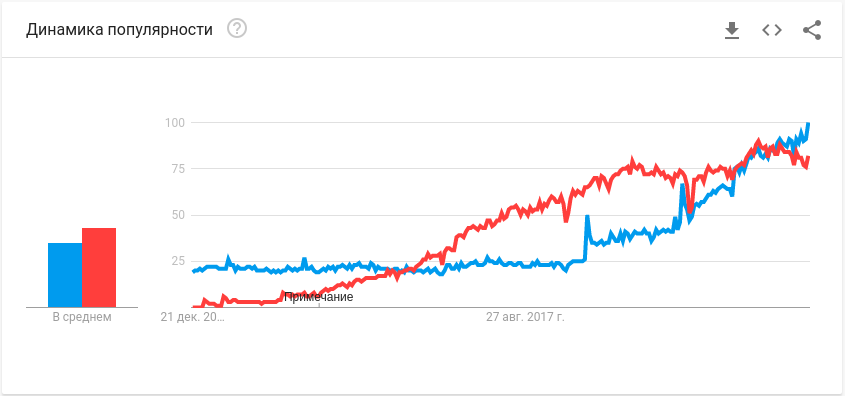


Рисунок 1.5 – График популярности *React Native* и *Flutter* [9]

На рисунке 1.5 приведен график популярности *React Native* (красный) и *Flutter* (синий), согласно статистике *Google Trends.*

Для сравнения *React Native* и *Flutter* компанией *SimbirSoft* была создана таблица в зависимости от характера приложения (рис. 1.6).

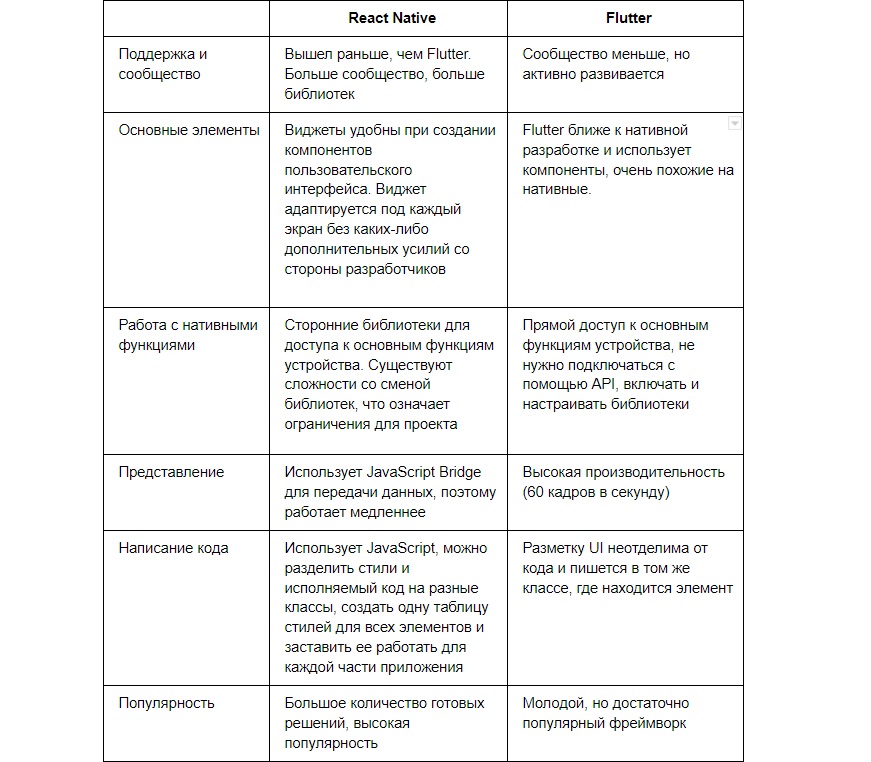


Рисунок 1.6 – Сравнение *React Native* и *Flutter* [10]

Опираясь на выше перечисленные критерии для разработки небольшого приложения с использованием сторонних библиотек хорошим выборам будет *React Native* из-за своей простоты и популярности.

В качестве основной среды разработки была выбрана *WebStorm* [11] от компании *JetBrains.* Данная интегрированная среда разработки является лучшей для написания кода на языках *JavaScript* и *TypeScript* на сегодняшний день.

Так как в теме дипломного проекта обозначена разработка под систему *Android*, то для разработки приложения также будет необходима среда *Android Studio. Android Studio* это официальная интегрированная среда разработки(*IDE*) для разработки *Android*-приложений, основанная среде разработки *IntelliJ IDEA* [12].

## **1.4 Постановка задач по разработке программного средства**

Разрабатываемое приложение должно иметь удобный, интуитивно по-нятный интерфейс, в котором самые важные функции находятся в ближайшей доступности, а более глубокие возможности не должны перегружать пользо-вательский интерфейс.

В результате анализа аналогов на рынке приложений и предметной области сформирован следующий список требований к разрабатываемой системе:

* возможность авторизации в системе;
* возможность просмотра профиля пользователя;
* возможность внесения данных о потребленных пользователем калориях, уровне глюкозы в крови, времени физической активности;
* возможность устанавливать получение уведомлений с напоминанием о принятии лекарственных средств;
* возможность получения пользователем статистики своих данных;
* реализация игровой логики и системы вознаграждения пользователя;
* реализация чат-бота.

# **2 РАЗРАБОТКА** **ПРОГРАММНОГО СРЕДСТВА**

## **2.1 Разработка архитектуры программного средства**

Программный код дипломного проекта будет разрабатываться в архи-тектурном стиле *Model-View-Controller* (*MVC*, «Модель-Представление-Кон-троллер»).

*Activity* —это компонент приложения,который выдает экран,и с кото-рым пользователи могут взаимодействовать для выполнения каких-либо дей-ствий, например набрать номер телефона, сделать фото, отправить письмо или просмотреть карту. Каждой операции присваивается окно для прорисовки со-ответствующего пользовательского интерфейса. Обычно окно отображается во весь экран, однако его размер может быть меньше, и оно может разме-щаться поверх других окон [13].

Для эффективной работы системы *Android*, в ней были созданы жизненные циклы различных элементов программного средства. Жизненный цикл *Android*-приложения жёстко контролируется системой и зависит от нужд пользователя, доступных ресурсов и производительности устройства. Например, когда пользователь хочет запустить браузер, решение о запуске приложения принимает система. Хоть последнее слово и остаётся за системой, она подчиняется заданным и логическим правилам, позволяющим определить, можно ли загрузить или приостановить приложение, прекратить его работу. Когда в определённый момент пользователь работает с конкрет-ным окном, система даёт приоритет соответствующему приложению. И наобо-рот, если окно невидимо, и система решает, что работу приложения необхо-димо остановить, чтобы освободить дополнительные ресурсы, будет останов-лена работа приложения, имеющего более низкий приоритет. В *Android* ресурсы на мобильном устройстве ограниченны, поэтому он более жёстко кон-тролирует работу приложений [14].

У активити есть свой жизненный цикл, который показан на рисунке 2.1.

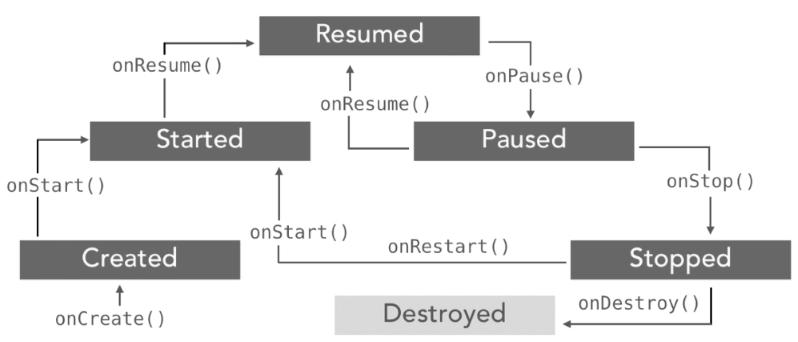


Рисунок 2.1 – Жизненный цикл *Activity*

Метод *onCreate*() вызывается при создании или перезагрузке активно-сти. Система может запускать и приостанавливать текущие окна в зависимо-сти от происходящих событий. Внутри данного метода настраивают статиче-ский интерфейс окна. В методе инициализируются статические данные актив-ности, связывают данные со списками и так далее. Связывает с необходимыми данными и ресурсами. Задаёт внешний вид через метод *setContentView*(). В этом методе загружаем пользовательский интерфейс, размещаем ссылки на свойства класса, связываем данные с элементами управления, создаваем сер-висы и потоки. Метод *onCreate*() принимает объект *Bundle*, содержащий со-стояние пользовательского интерфейса, сохранённое в последнем вызове об-работчика *onSaveInstanceState*. Для восстановления графического интерфейса в его предыдущем состоянии нужно задействовать эту переменную: внутри *onCreate*()или переопределив метод *onRestoreInstanceState*().В методе можносделать проверку, запущено ли приложение впервые или восстановлено из па-мяти. Если значение переменной *savedInstanceState* будет *null*, приложение за-пускается первый раз.

За *onCreate*() всегда следует вызов *onStart*(), но перед *onStart*() не обяза-тельно должен идти *onCreate*(), так как *onStart*() может вызываться и для воз-обновления работы приостановленного приложения (приложение останавли-вается методом *onStop*()). При вызове *onStart*() окно еще не видно пользова-телю, но вскоре будет видно. Вызывается непосредственно перед тем, как ак-тивность становится видимой пользователю. Сопровождается вызовом метода *onResume*(),если активность получает передний план,или вызовом метода *onStop*(),если становится скрытой.

Метод *onResume*() вызывается после *onStart*(), даже когда окно работает в приоритетном режиме и пользователь может его наблюдать. В этот момент пользователь взаимодействует с созданным вами окном. Приложение полу-чает монопольные ресурсы. Запускает воспроизведение анимации, аудио и ви-део. Также может вызываться после *onPause*().

Система вызывает этот метод каждый раз, когда активность идет на пе-реднем плане, в том числе, при первом создании. Таким образом, мы должны реализовать *onResume*() для инициализации компонентов, регистрации любых широковещательных приемников или других процессов, которые освободили или приостановили в *onPause*() и выполнять любые другие инициализации, ко-торые должны происходить, когда активность вновь активна.

Необходимо пытаться размещать относительно быстрый и легковесный код, чтобы приложение оставалось отзывчивым при скрытии с экрана или вы-ходе на передний план.

Когда пользователь решает перейти к работе с новым окном, система вызовет для прерываемого окна метод *onPause*(). По сути происходит свёрты-вание активности. Сохраняет незафиксированные данные. Деактивирует и вы-пускает монопольные ресурсы. Останавливает воспроизведение видео, аудио и анимацию. От *onPause*() можно перейти к вызову либо *onResume*(), либо *onStop*().

Метод *onStop*() вызывается, когда окно становится невидимым для поль-зователя. Это может произойти при её уничтожении, или если была запущена другая активность (существующая или новая), перекрывшая окно текущей ак-тивности. Всегда сопровождает любой вызов метода *onRestart*(), если актив-ность возвращается, чтобы взаимодействовать с пользователем, или метода *onDestroy*(),если эта активность уничтожается.

Для передачи данных между активити, в *Android SDK* существуют ин-тенты. *Intent* представляет собой объект обмена сообщениями, с помощью ко-торого можно запросить выполнение действия у компонента другого прило-жения. Несмотря на то, что объекты *Intent* упрощают обмен данными между компонентами по нескольким аспектам. В разрабатываемом приложении ин-тенты будут использоваться для запуска активностей и получения результатов с других компонентов. Компонент *Activity* представляет собой один экран в приложении. Для запуска нового экземпляра компонента *Activity* необходимо передать объект *Intent* методу *startActivity*(). Объект *Intent* описывает опера-цию, которую требуется запустить, а также содержит все остальные необходи-мые данные.

Если после завершения операции от нее требуется получить результат, вызовите метод *startActivityForResult*(). Ваша операция получит результат в виде отдельного объекта *Intent* в обратном вызове метода *onActivityResult*() операции.

Файл манифеста инкапсулирует всю архитектуру *Android*-приложения, его функциональные возможности и конфигурацию. В процессе разработки приложения приходится постоянно редактировать данный файл, изменяя его структуру и дополняя новыми элементами и атрибутами.

Корневым элементом манифеста является <*manifest*>. Помимо данного элемента обязательными элементами является теги <*application*> и <*uses-sdk*>. Элемент <*application*> является основным элементом манифеста и содержит множество дочерних элементов, определяющих структуру и работу приложе-ния. Порядок расположения элементов, находящихся на одном уровне, произ-вольный. Все значения устанавливаются через атрибуты элементов. Кроме обязательных элементов, упомянутых выше, в манифесте по мере необходи-мости используются другие элементы.

Элемент <*application*> один из главних элементов файла манифеста, он содержит описание компонентов приложения, доступных в пакете: стили, зна-чок, строки и др. Содержит дочерние элементы, которые объявляют каждый из компонентов, входящих в состав приложения. Также в файле может быть только один элемент <*application*>.

Элемент <*activity*> объявляет активность. Если приложение содержит несколько активностей, то нужно обязательно объявлять их в манифесте, со-здавая для каждой из них свой элемент <*activity*>. Если активность не объяв-лена в манифесте, она не будет доступна системе и не будет запущена при вы-полнении приложения или будет выводиться сообщение об ошибке. Каждый тег <*activity*> поддерживает вложенные узлы <*intent*-*filter*>. Элемент <*intent*-*filter*>определяет типы намерений,на которые могут ответить деятельность,сервис или приемник намерений. Фильтр намерений определяет возможности его родительского компонента, в нём описано, что могут сделать деятельность или служба и какие типы рассылок получатель может обработать. Фильтр намерений предоставляет для компонентов-клиентов возможность получения намерений объявляемого типа, отфильтровывая те, которые не значимы для компонента, и содержит дочерние элементы <*action*>, <*category*>, <*data*>.

Элемент <*service*> объявляет службу как один из компонентов приложе-ния. Все службы должны быть представлены элементом <*service*> в файле ма-нифеста. Службы, которые не были объявлены, не будут обнаружены систе-мой и никогда не будут запущены. Этот элемент имеет много атрибутов, опре-деляющих имя, доступность, разрешения, процесс и т. д. Поддерживает вло-женные узлы <*intent*-*fiiter*> [15].



Рисунок 2.2 – Скриншот файла *AndroidManifest.xml*

В архитектуре *React Native* приложений имеется характерная особенность в том, что все приложение находится в единственной активности. Так как вся бизнес-логика приложения написана на языке *JavaScript*, который не является родным для мобильных платформ, то создается одна активность, в которой обрисовывается весь интерфейс приложения, а также создается отдельный поток для выполнения *JavaScript*-кода. Поэтому все вышеперечисленные методы жизненного цикла активности применимы к разрабатываемому приложению в единственном экземпляре.

Несмотря на то, что *React Native* приложения состоят всего из одной активности, паттерн *MVC* все же остается актуален. *JavaScript*-код приложения состоит из компонентов, которые имеют в себе часть, ответственную за отображение, часть – за контроллер и часть – за модель.

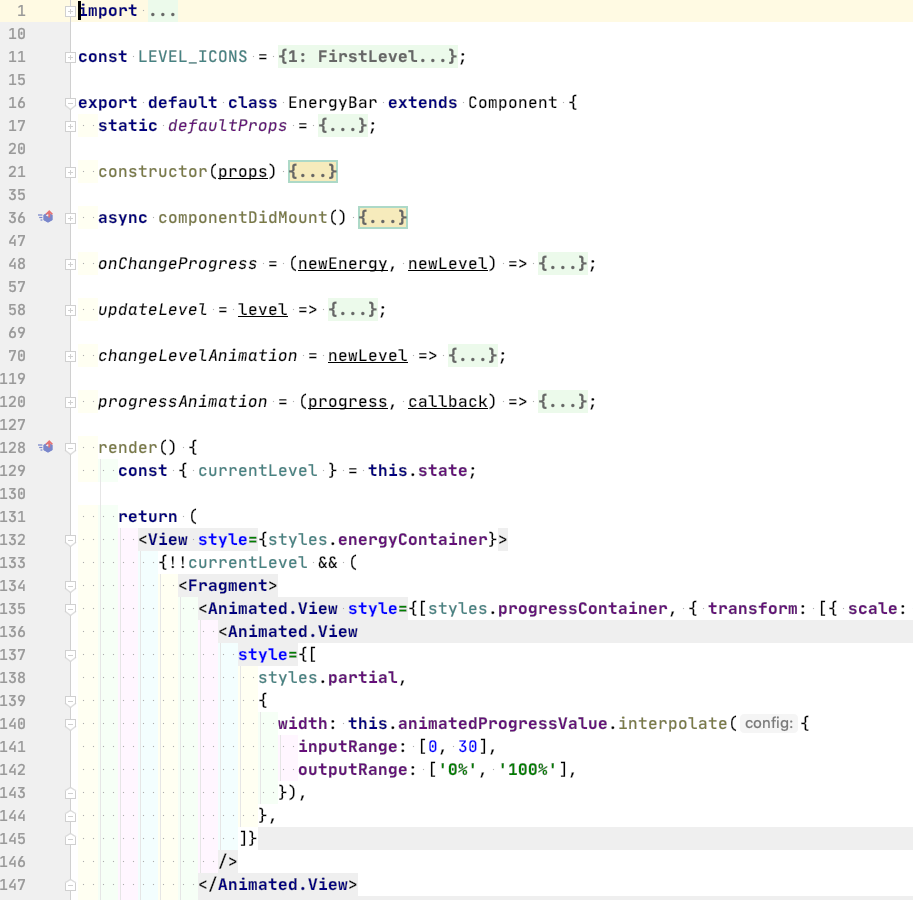


Рисунок 2.3 – Пример компонента *React Native* приложения

На рисунке 2.3 изображен скриншот, показывающий пример компонента *React Native* приложения. В строке 16 кода можно видеть объявление класса *EnergyBar*, который и является компонентом. В строке 128 объявляется метод класса *render()*, выполнение которого приводит к отображению части пользовательского интерфейса, за которую ответственен данный компонент. Именно эта часть компонента и отвечает за составляющую отображения в паттерне *MVC*. В строке 48 приведенного снимка кода присутствует объявление метода *onChangeProgress()*, который выполняет функцию контроллера, а именно при определенном взаимодействии пользователя с интерфейсом приложения изменяется уровень прогресса – и данный метод выполняется как реакция на действия пользователя. Составляющая, ответственная за выполнение функций модели находится в составе конструктора класса. Это состояние компонента. Которое изменяется посредством методов-контроллеров, например, *onChangeProgress()*. Изменение состояния компонента в свою очередь приводит к его перерисовке и обновлении составляющей отображения.

Чтобы вручную не собирать проект и не прописывать множество ко-манд, разработчики создали системы сборки проектов. В нашем случае ис-пользуется *Gradle*.

*Gradle* – система автоматической сборки, построенная на принципах *Apache Ant* и *Apache Maven*, но предоставляющая *DSL* на языке *Groovy* вместо традиционной *XML*-образной формы представления конфигурации проекта.

В отличие от *Apache Maven*, основанного на концепции жизненного цикла проекта, и *Apache Ant*, в котором порядок выполнения задач (*targets*) определяется отношениями зависимости (*depends-on*), *Gradle* использует направленный ациклический граф для определения порядка выполнения задач.

*Gradle* был разработан для расширяемых многопроектных сборок, и поддерживает инкрементальные сборки, определяя, какие компоненты дерева сборки не изменились и какие задачи, зависимые от этих частей, не требуют перезапуска [16].

На рисунке 2.4 приведен снимок экрана, отображающего конфигурацию *build.gradle* файла.



Рисунок 2.4 – Снимок экрана с конфигурацией *build.gradle* файла

В этом файле указаны все необходимые специфичные для запуска *Android* проекта опции, такие как *defaultConfig, compileOtions* и другие. Однако в контексте *React Native* приложения важным моментом конфигурации данного файла является пятая строка, в которой указан входной файл для *JavaScript* кода. Именно с этого файла начнется выполнение приложения после того, как полностью сконфигурируется и запустится его *Android* оболочка.

## **2.2 Разработка информационной модели системы**

В данном дипломном проекте для хранения всех пользовательских данных используется база данных *PostgreSQL*. Никакие пользовательские данные не должны храниться локально на мобильном устройстве, кроме тех токенов и ключей, которые необходимы для обеспечения более удобного входа и авторизации в приложении. Такое решение обусловлено тем, что необходимо обеспечить наименьшую вероятность потери данных пользователем при удалении, переустановке или обновлении приложения.

*PostgreSQL* — это объектно-реляционная система управления базами данных (ОРСУБД). В *PostgreSQL* появилось множество новшеств, которые были реализованы в некоторых коммерческих СУБД гораздо позднее.

*PostgreSQL* — СУБД с открытым исходным кодом, основой которого был код, написанный в Беркли. Она поддерживает большую часть стандарта *SQL* и предлагает множество современных функций:

* сложные запросы
* внешние ключи
* триггеры
* изменяемые представления
* транзакционная целостность
* многоверсионность

Кроме того, пользователи могут всячески расширять возможности PostgreSQL, например создавая свои

* типы данных
* функции
* операторы
* агрегатные функции
* методы индексирования
* процедурные языки

А благодаря свободной лицензии, PostgreSQL разрешается бесплатно использовать, изменять и распространять всем и для любых целей — личных, коммерческих или учебных [17].

Для взаимодействия между мобильным приложением и базой данных необходимо создать серверное приложение, которое будет принимать *REST*-запросы от клиентских приложений, совершать запросы к базе данных и отправлять полученную информацию обратно.

Так как выбранное решение для организации базы данных основано на *SQL*-таблицах, то необходимо тщательно продумать структуру и организацию таблиц данных для того, чтобы избежать совершения неэффективных запросов к базе данных и уменьшения производительности информационной системы в целом.

Для работы приложения был создан набор таблиц, которые покрывают имеющийся функционал и позволяют использовать их для реализации новых функций. Рассмотрим созданные таблицы и дадим краткую характеристику каждой из них.

Таблица *«users»* предназначена для хранения списка пользователей приложения. При окончании регистрации в приложении в эту таблицу вносится новая запись с данными, которые внес пользователь в специальной регистрационной форме. На рисунке 2.5 можно видеть, какие поля входят в состав данной таблицы.

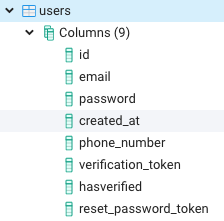


Рисунок 2.5 – Таблица «*users*»

Таблица *«metrics»* предназначена для хранения вводимых пользовательских данных для учета и предоставления последующей статистики. Сохраненные в таблице данные ассоциируются с определенным пользователем посредством поля *user\_id*, в котором хранится идентификационный номер пользователя из таблицы *users*. Данная таблица хранит информацию о количестве потребленных пользователем углеводов, приемов пищи, лекарственных средств и уровне глюкозы в крови, а также о количестве совершенной физической активности.

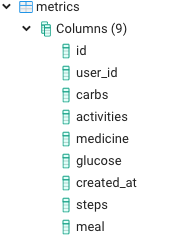


Рисунок 2.6 – Таблица «*metrics*»

Таблица *«achievements\_info»* содержит данные о достижениях пользователя. Так как в приложении должна быть реализована система вознаграждения и достижений, то именно эта таблица будет хранить соответствующие данные. Особенность данной таблицы в том, что она хранит промежуточные данные о прогрессе пользователя.

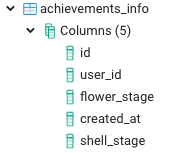


Рисунок 2.7 – Таблица «*achievements\_info*»

Также для слежения за «успеваемостью» пользователя необходимо создать таблицу, которая хранит данные о «наградах» пользователя. В приложении необходимо организовать по крайней мере два уровня. Для каждого уровня достижений характерен свой тип призов и наград пользователя. Например, находясь на первом уровне пользователь награждается цветками за ввод своих данных. При переходе на второй уровень в качестве награды будет использоваться морская ракушка. Ниже на рисунке 2.8 приведены структуры двух типов наград пользователя.

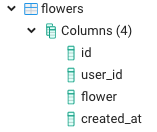
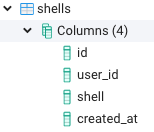
 

Рисунок 2.8 – Таблицы «*flowers*» и «*shells*»

## **2.3 Разработка алгоритмов функционирования программного**

## **средства**

Описать алгоритмы, для которых буду делать схемы.

## **2.4 Разработка и обоснование пользовательского интерфейса**

## **программного средства**

Пользовательский интерфейс является принципиально важной частью продукта. Неудобный пользовательский интерфейс может оттолкнуть пользо-вателя, снизить эффективность использования веб-сервиса. Для разработки пользовательского интерфейса обычно привлекают дизайнера.

В связи с особенностями программного продукта пользовательский интерфейс также должен быть прост в использовании – расположение некоторых элементов, цветовая гамма и т.п. должны соответствовать сценария использования приложения через сенсорный экран мобильного устройства.

На сегодняшний день существует множество рекомендаций по проектированию пользовательского интерфейса, а также готовых решений, которые можно просто настроить под определенное приложение. Например, *Android* предоставляет следующие элементы для построения приложений в соответствии с концепцией *Material Design*:

* новую тему;
* новые виджеты для сложных представлений;
* новые *API*-интерфейсы для нестандартных теней и анимаций [18].

*Material Design* представляет собой комплексную концепцию созданиявизуальных, движущихся и интерактивных элементов для различных плат-форм и устройств. Теперь *Android* включает в себя поддержку приложений с элементами *Material Design*. Чтобы использовать элементы *Material Design* в своих приложениях под *Android*, руководствуйтесь инструкциями в специфи-кации *Material Design*, а также воспользуйтесь новыми компонентами и функ-циями, доступными в *Android* 5.0 (уровень *API* 21) и выше.

Однако такие решения подходят для типового мобильного приложения, которое использует элементы интерфейса, предоставляемые операционной системой. В случае какого-либо приложения с игровой логикой подобные решения далеко не всегда будут являться актуальными. Именно в таком положении и находится приложение, разрабатываемое в данном дипломном проекте. Поэтому в таком случае необходимо прибегнуть к помощи дизайнера.

Интерфейс программного средства должен удовлетворять следующим требованиям:

* интерфейс должен быть интуитивно понятен и не вызывать замеша-тельства;
* в интерфейсе должен присутствовать доступ к наиболее необходимым для пользователя частям программного средства;
* интерфейс должен располагать цветовой гаммой, не раздражающей глаз и не вызывающей чувство отвращения, цвета должны быть спокойных тонов;
* интерфейс должен максимально использовать свободное простран-ство экрана мобильного устройства, но не должно быть нагромождения;
* использование программного средства должно быть понятно пользо-вателю без специальной подготовки и не должно требовать прохождения пред-варительного обучения в использовании программы;
* расположение элементов, стиль иконок и поведение интерфейса должно следовать устоявшимся стандартам разработки программ для системы *Android*,для того чтобы программное средство не выбивалось из визуальногостиля системы.

Для создания пользовательского интерфейса в приложениях на платформе *React Native* используется такой же подход, как и в разработке с библиотекой *React*. Этот подход предполагает написание кода разметки компонентов и последующее добавление стилей по принципу, схожему с веб-разработкой. Ниже будут рассмотрены результаты проектирования пользовательского интерфейса основных экранов приложения.

При первом открытии приложения пользователю необходимо создать учетную запись и зарегистрировать ее, используя адрес электронной почты и номер мобильного телефона. Номер телефона пользователь может не указывать, а использовать только адрес почты. Также необходимо создать пароль для защиты учетной записи от взлома.

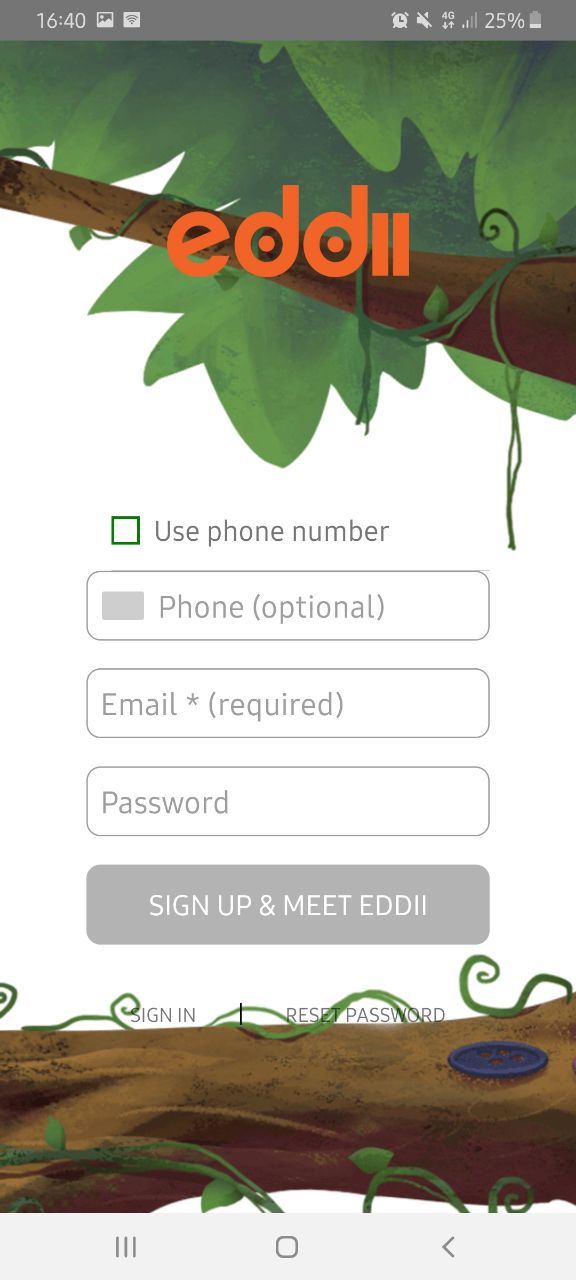


Рисунок 2.9 – Экран регистрации в приложении

На рисунке 2.9 изображен интерфейс экрана регистрации нового пользователя. Так как использование номера мобильного устройства является опциональным, то необходимо было реализовать возможность отмены его использования. Кроме кнопки подтверждения формы на экране можно видеть две дополнительные кнопки, отвечающие за переход к экрану входа, в случае, когда пользователь уже имеет учетную запись, и переход к экрану сброса пароля. Интерфейс данных экранов практически аналогичен экрану регистрации, поэтому их скриншоты не приведены в данном описании.

После прохождения пользователем процесса аутентификации он перенаправляется к главному экрану приложения. Изображение данного экрана можно видеть на рисунке 2.10.



Рисунок 2.10 – Главный экран приложения

Так как в приложении реализована игровая логика, то и интерфейс соответственно будет не стандартным. На рисунке 2.10 изображена игровая локация первого уровня – парк. Также на рисунке присутствует игровой персонаж, сидящий на ветке дерева. Вверху экрана находится панель прогресса пользователя, кнопка переключения локаций и кнопка входа в настройки приложения. Внизу данного экрана находится кнопка перехода в меню навигации. Также внизу экрана располагается круговая диаграмма, которая отображает текущую статистику уровня глюкозы в крови пользователя на основе введенных им данных. При нажатии на данную диаграмму происходит навигация приложения к экрану с более подробной статистикой. А при нажатии на иконку редактирования пользователю будет отображена форма ввода новых данных для учета здоровья.

На рисунке 2.11 представлено окно переключения игровых локаций. Новые локации становятся доступными по мере игрового прогресса пользователя, а точнее по мере количества введенных данных. При переключении локации произойдет перенаправление на главный экран приложения. Отличие будет заключаться в том, что тема оформления главного экрана будет перерисована в соответствии с новой локацией.

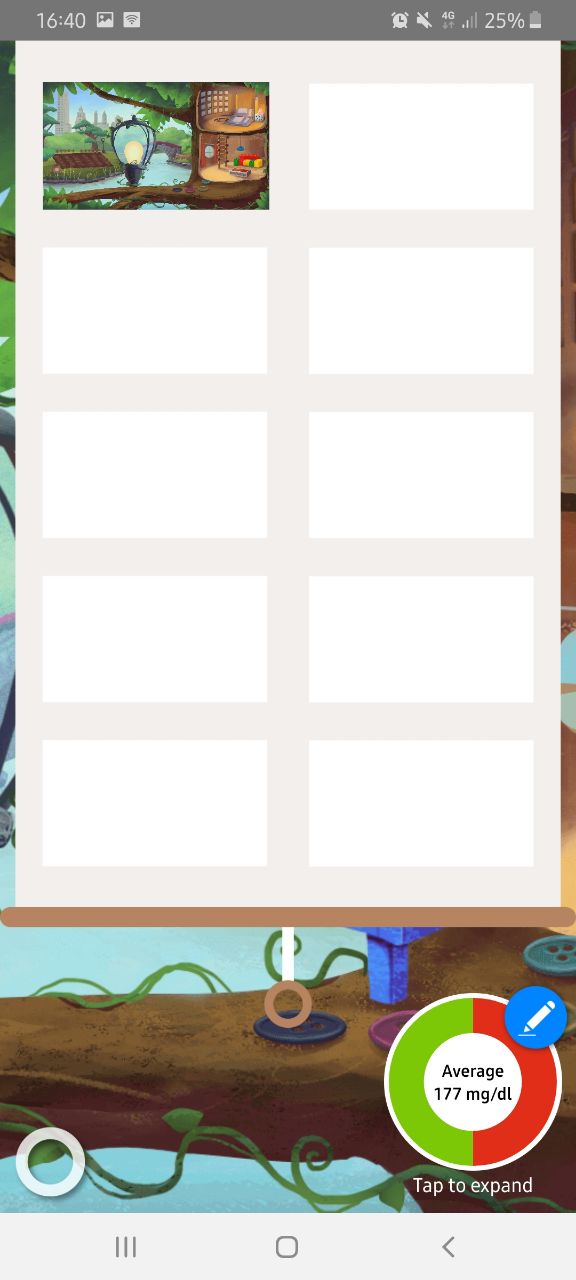


Рисунок 2.11 – Окно переключения игровой локации

Как уже было описано выше, при нажатии на круговой график пользователь перенаправляется на подробную статистику своего здоровья. На открывшемся экране отображаются все данные, которые пользователь когда-либо вносил. Данные отображаются в отсортированном по дням недели формате. Здесь пользователь может ознакомиться с информацией о его уровне глюкозы в крови, принятом количестве лекарственных средств, времени физической активности и количестве потребленных калорий.

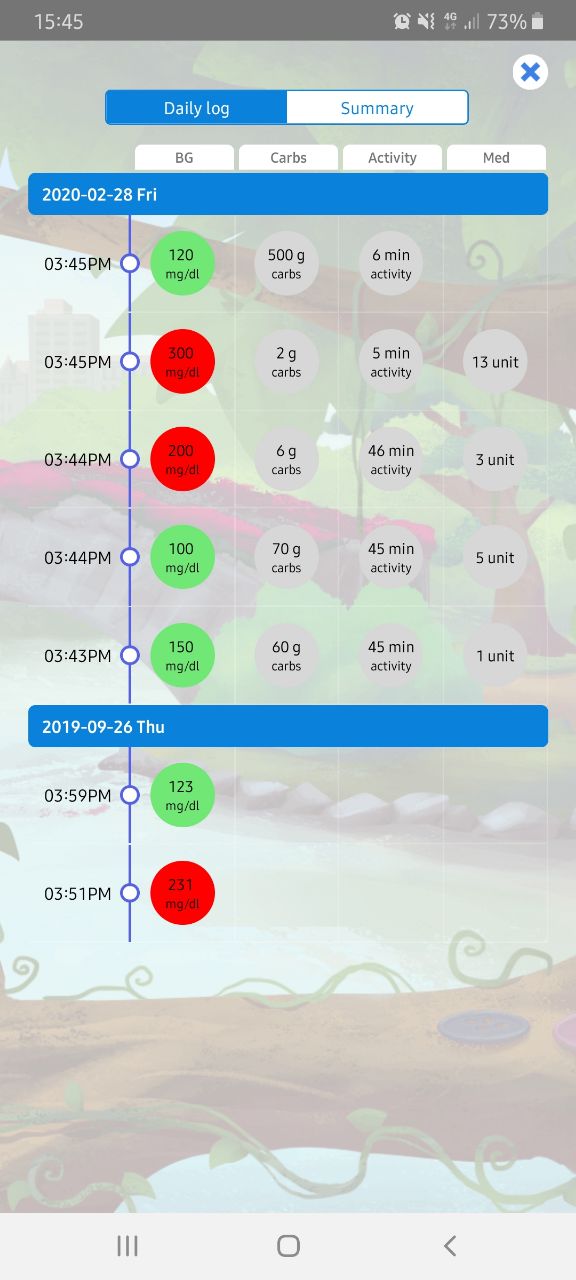


Рисунок 2.12 – Экран подробной статистики

На рисунке 2.13 изображен экран ввода новых данных о здоровье пользователя.



Рисунок 2.13 – Экран ввода данных

На экране ввода данных находится простая форма для внесения информации. Однако особенностью данного экрана является то, что на нем также присутствует цветок. Цветок является игровой наградой пользователя за ввод данных. Логика заключается в том, что чем больше полей формы заполнит пользователь, тем быстрее вырастет цветок. Полностью выросший цветок отправляется на «клумбу» главного экрана. Если же пользователь уже завершил первый уровень, то в качестве награды будет выращиваться уже другой тип предмета.

На рисунке 2.14 изображен экран чата. Он предназначен для имитации общения пользователя с игровым персонажем посредством онлайн-переписки.

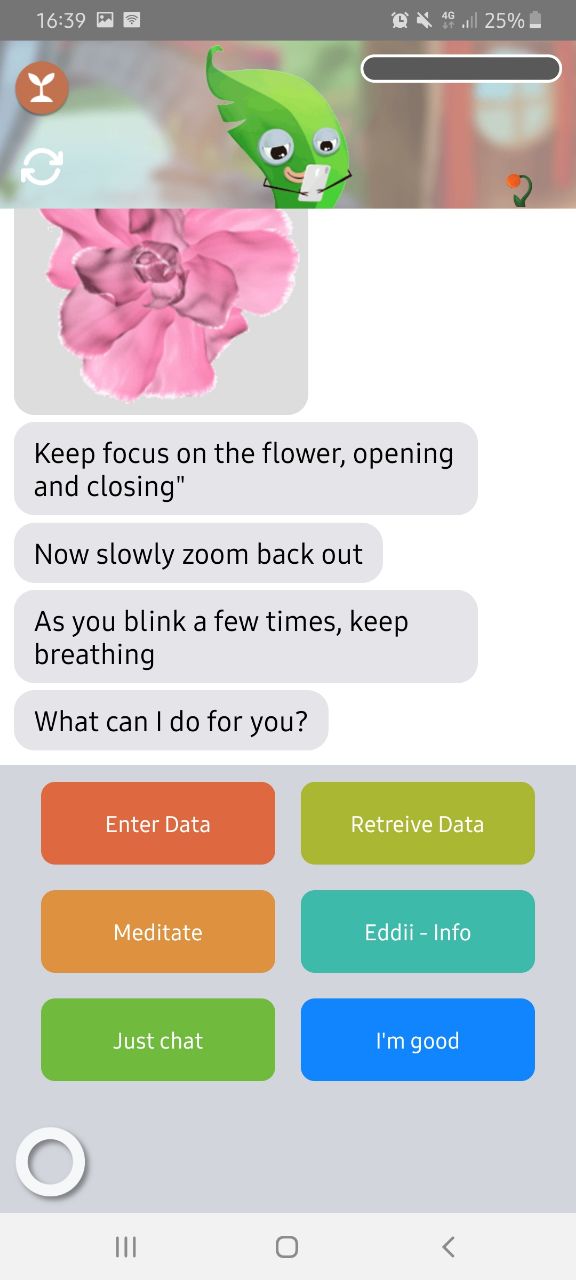


Рисунок 2.14 – Экран ввода данных

Вверху данного экрана также отображен текущий прогресс пользователя. Пользователю предоставляется выбрать ответное сообщение посредством нажатия кнопки. Игровой персонаж, в свою очередь, будет отвечать на сообщения пользователя.

Интерфейс проектировался таким образом, чтобы на экране устройства отображалась самая необходимая для пользователя информация и функции. Также получились интуитивно понятное меню и навигация в приложении. Реализация доступа к широким функциональным возможностям и обеспечение простоты работы противоречат друг другу. Разработка эффективного интерфейса призвана сбалансировать эти цели[25]. Таким образом был разработан пользовательский интерфейс, отвечающий требованиям, описанным в начале подраздела.

# **3 ИНЖЕНЕРНЫЕ РАСЧЕТЫ, ИСПОЛЬЗУЕМЫЕ**

# **В ПРОГРАММНОМ СРЕДСТВЕ**

## **3.1 Оценка файловой структуры игрового программного средства**

## **3.2 Анализ памяти, используемой программным средством**

# **4 ЭКСПЛУАТАЦИЯ ПРОГРАММНОГО СРЕДСТВА**

## **4.1 Ввод в эксплуатацию программного средства**

## **4.2 Руководство к пользованию разработанным программным**

## **средством**

# **5 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ**

# **РАЗРАБОТКИ ИГРОВОГО АРКАДНОГО**

# **ПРОГРАММНОГО СРЕДСТВА** **«RUN OUT» ПОД**

# **ОПЕРАЦИОННУЮ СИСТЕМУ ANDROID**

## **5.1 Краткая характеристика программного продукта**

Игровое аркадное программное средство будет предназначено для использования в развлекательных целях. Играть в мобильные игры - лучший вариант чтобы расслабиться в свободное время. Так же мобильная игра может способствовать развитию логики, внимания, рефлексов, а также помочь в преодолении депрессии и снижению уровня стресса у игрока.

В большей степени игровое приложение ориентировано на молодежь, которая относится к числу людей той возрастной категории, представители которой с наибольшей вероятностью имеют мобильный телефон. К тому же подростки обладают большим количеством свободного времени нежели люди старших возрастных категорий. Как показывают многие исследования, подростки любят играть в видеоигры, потому что они бросают вызов и этим заинтересовывают игрока. Игры предлагают умственные и визуальные стимулы, которые увеличивают вовлеченность. Многие подростки, играя в игры, любят соревноваться друг с другом и таким способом находят в играх способность к самовыражению.

Программное средство разработано в рамках дипломного проекта с целью дальнейшего размещения в сети Internet. Основной целевой аудиторией является молодежь. Ожидается экономический эффект в виде прироста прибыли от продажи копий игры. За счёт того, что пользователи могут рекомендовать игру своим друзьям, оставлять отзывы и выставлять рейтинги, игра всегда сможет привлечь новых пользователей.

## **5.2 Расчет затрат на разработку и отпускной цены программного**

## **продукта**

## **5.3 Расчет экономической эффективности реализации на рынке**

## **игрового аркадного программного средства «Run out»**

## **под операционную систему Android**

# **ЗАКЛЮЧЕНИЕ**

В рамках данной дипломной работы было разработано игровое аркадное программное средство «Run out» под операционную систему Аndroid. Была проанализирована информация о мобильных играх, жанрах, игровых движках. Было просмотрено большое количество литературы и интернет-источников. После разработки концепции игры, в ходе анализа четырех игровых движков, был выбран движок Unity3d, который удовлетворяют определённым требованиям:

– простота разработки;

– поддержка языков программирования С# и JavaScript;

– мультиплатформенность;

– бесплатная версия.

При работе со скриптами и проектной расстановке объектов на сцене был проделан тщательный анализ игр аналогов для поддержания жанра и выделении особенностей игры.

Получены большие знания в области разработке игровых приложений: работа с игровыми движками, работа с памятью мобильных устройств, оптимизация производительности. Также мной была изучена среда разработки Unity3D: возможности редактора движка, слабые и сильные стороны этого движка. Получен большой опыт разработки 3D сцен: изучены основные понятия, изучена и понята работа с текстурами как в 3D, так и в 2D. Приобретен опыт программирования на языке C#. Изучена и проанализирована библиотека Unity3D, составлены собственные классы, которые ее дополняют. Таким образом, все поставленные задачи были выполнены, цель работы достигнута.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Диабет [Электронный ресурс]. – Режим доступа: https://www.who.int/ru/news-room/fact-sheets/detail/diabetes
2. Мобильное приложение [Электронный ресурс]. – Режим доступа: https://indicator.ru/label/mobilnoe-prilozhenie
3. Мобильное приложение [Электронный ресурс]. – Режим доступа: https://www.mk.ru/social/health/2016/06/01/bolshinstvo-detey-s-sakharnym-diabetom-chuvstvuyut-sebya-neschastnymi.html
4. *Mobile OS market share worldwide* [Электронный ресурс]. – Режим доступа: https://gs.statcounter.com/os-market-share/mobile/worldwide
5. *React Native* [Электронный ресурс]. – Режим доступа: https://reactnative.dev
6. *React Native*: год спустя [Электронный ресурс]. – Режим доступа: https://engineering.fb.com/android/react-native-a-year-in-review/
7. Архитектура *React Native* приложений [Электронный ресурс]. – Режим доступа: https://2018.holyjs-piter.ru/talks/3wy7ejpjymg6cqu0igqywa/
8. *Flutter* для мобильных разработчиков [Электронный ресурс]. – Режим доступа: https://flutter.dev/docs/get-started/flutter-for/android-devs
9. *React Native, Flutter* [Электронный ресурс]. – Режим доступа: https://trends.google.ru/trends/explore?date=today%205-y&q=flutter,react%20native
10. Мобильная разработка: критерии выбора [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/simbirsoft/blog/460030/
11. *WebStorm* [Электронный ресурс]. – Режим доступа: https://www.jetbrains.com/webstorm/promo
12. *Meet Android Studio* [Электронный ресурс]. – Режим доступа: https://developer.android.com/studio/intro
13. Что такое *Activity* [Электронный ресурс]. – Режим доступа: *https://developer.android.com/guide/components/activities/?hl=ru*
14. Жизненный цикл на *Android* [Электронный ресурс]. Режим доступа: *http*://*developer*.*alexanderklimov*.*ru*/*android*/*theory*/*lifecycle*.*php*
15. *Material Design* для *Android* [Электронный ресурс].Режим доступа: *https*://*developer*.*android*.*com*/*design*/*material*?*hl*=*ru*
16. Система сборки *Gradle* [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Gradle
17. *PostgreSQL*: документация [Электронный ресурс]. – Режим доступа: https://postgrespro.ru/docs/postgresql/9.6/intro-whatis
18. Макеты [Электронный ресурс]. – Режим доступа: *https*://*webhamster*.*ru*/*mytetrashare*/*index*/*mtb*189/14906146443*cuohwb*6*fa*

# **ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Отчёт по анализу заимствования материала**

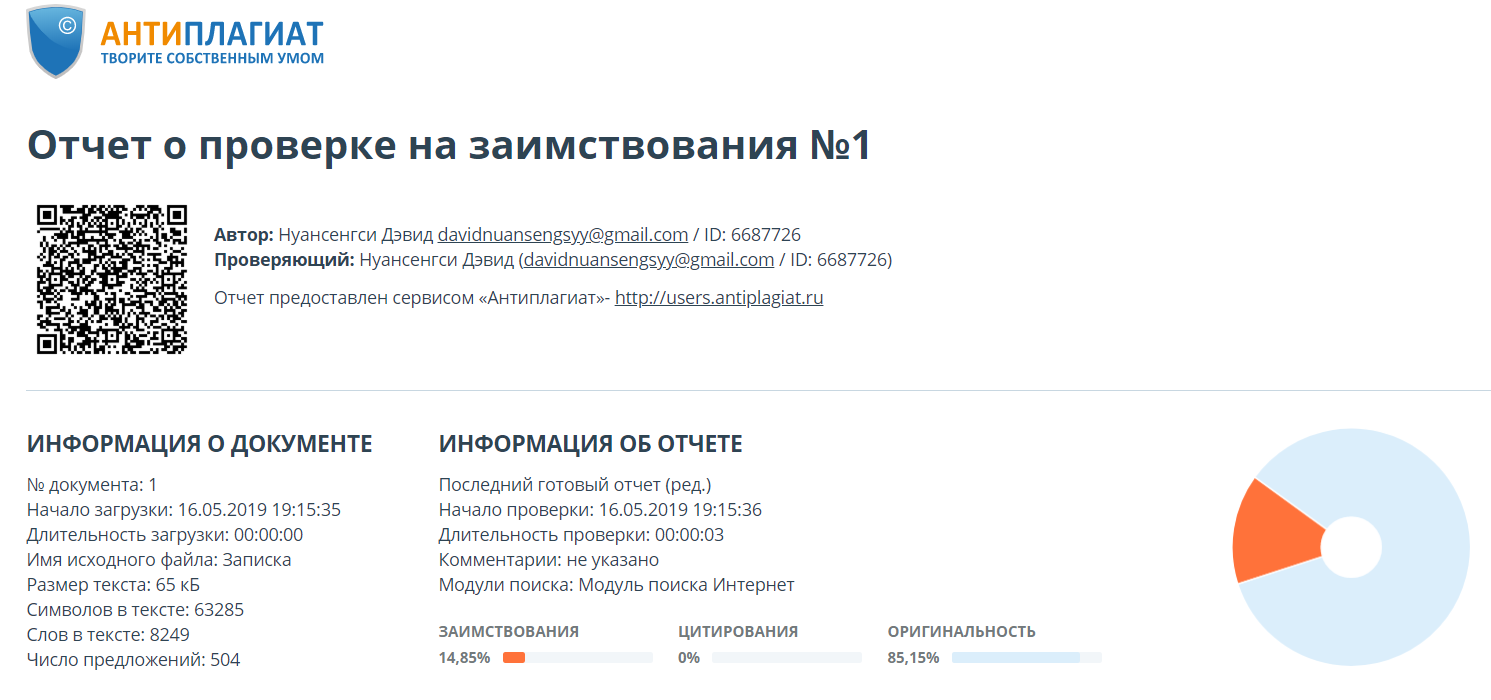


Рисунок А.1 – Скриншот отчета о проверке дипломного проекта в системе «Антиплагиат»

# **ПРИЛОЖЕНИЕ** **Б (обязательное)**

**Листинги программ**

# **ПРИЛОЖЕНИЕ В (обязательное)**

**Ведомость дипломного проекта**