Содержание

Введение 7

1 Анализ требований к программному средству и постановка задач 8

1.1 Анализ исходных данных и описание функциональных

возможностей программного средства 8

1.2 Обоснование выбора языка программирования и средств разработки 10

1.3 Обзор существующих программных средств по теме

дипломного проекта 13

1.4 Постановка задач по разработке программного средства

для чтения электронных книг и ведения словаря 17

2 Проектирование и разработка программного средства

для чтения электронных книг и ведения словаря (переводчика) 23

2.1 Разработка объектной модели программного средства 23

2.2 Разработка архитектуры программного средства 25

2.3 Разработка алгоритмов функционирования программного средства 28

2.4 Разработка и обоснование пользовательского интерфейса

программного средства 31

2.5 Проектирование базы данных и интерфейсов взаимодействия с ней 35

2.6 Описание взаимодействия программного средства с *Google*

*Translation API* 38

3 Инженерные расчёты, используемые в дипломном проекте 43

3.1 Оценка объёма передаваемого трафика между программным

средством и *Google Translation API* 43

3.2 Оценка среднего времени отклика *Google Translation API* 46

4 Эксплуатация программного средства для чтения электронных

книг и ведения словаря (переводчика) 52

4.1 Ввод в эксплуатацию программного средства 52

4.2 Руководство к использованию разработанного

программного средства 55

5 Технико-экономическое обоснование разработки программного

средства под операционную систему *Android* для чтения

электронных книг и ведения словаря (переводчика) 58

5.1 Описание функций, назначения и потенциальных пользователей ПО 58

5.2 Расчёт затрат на основную заработную плату

команды разработчиков 59

5.2 Расчёт затрат на дополнительную заработную плату

команды разработчиков 60

5.2 Расчёт отчислений на социальные нужды 60

5.2 Расчёт прочих затрат 60

5.2 Расчёт экономической эффективности от реализации

программного модуля на рынке 61

5.2 Расчёт показателей эффективности инвестиций

в разработку ПО 62

Заключение 63

Список использованных источников 64

Приложение А (обязательное) Отчёт по анализу заимствования

материала 66

Приложение Б (обязательное) Листинги программ 67

Приложение В (обязательное) Ведомость дипломного проекта 71

Введение

Ученые давно выяснили как сохранить ясность ума на протяжении всей жизни: нужно постоянно развивать свой мозг. Один из лучших способов делать это – регулярно и вдумчиво читать. Сегодня люди слишком заняты и все реже находят время для чтения. Однако у людей читающих выше шанс построить успешную карьеру, лучшие отношения в семье, они выглядят моложе и дольше живут.

Книга – отличный помощник для повышения квалификации в разных сферах деятельности. Для обучения нужны не только учебники, которыми мы пользуемся в учебном заведении, но и огромное количество литературы, используемой в свободное время, даже спустя многие годы после окончания учебы. Мозг читающего человека с легкостью впитывает новую информацию. Это позволяет получать новые знания не только на родном языке, но и формирует отличную базу для изучения иностранных языков[1].

Сегодня знание иностранных языков необходимо. Нужно это и для того, чтобы найти престижную высокооплачиваемую работу, и для общения в повседневной жизни.

Поможет в изучении иностранного языка чтение книг на этом языке. Оно способствует овладению всеми тонкостями изучаемого языка. При чтении расширяется словарный запас, усваиваются новые грамматические конструкции, и кроме того, это просто увлекательное занятие. Здесь нужно отметить и то, что любое переводное произведение несет на себе отпечаток личности переводчика, поэтому оно в любом случае отличается от произведения оригинального. И во многих случаях ощутить всю красоту его можно только в том случае, если читать на том языке, на котором писал его автор[2].

В рамках дипломного проектирования требуется разработать мобильное *Android*-приложение для чтения книг. Приложение также должно давать пользователю возможность вести словарь с переводом слов к каждой книги. На основе анализа предметной области составляется объектная модель приложения. Далее происходит разработка архитектуры приложения. Это необходимо для того, чтобы программный код можно было удобно писать, а также поддерживать.

Дипломный проект выполнен самостоятельно, проверен в системе «Антиплагиат». Процент оригинальности соcтавляет 75%. Цитирования обозначены ссылками на публикации, указанными в «Списке использованных источников».

Скриншот приведен в приложении А.

1. Анализ требований к программному средству и постановка задач

**1.1 Анализ исходных данных и описание функциональных возможностей программного средства**

В качестве технического задания было необходимо разработать программное средство для чтения электронных книг и ведения словаря (переводчика).

Основные требования к функциональности – добавление книг из локального хранилища в библиотеку и их удаление посредством *SQL*-запросов; ведение словаря (переводчика), закрепленного за каждой книгой; возможность выбора языка перевода.

Где бы вы не находились, вся ваша библиотека будет под рукой. Художественная, техническая и учебная, словари и справочники – включите устройство и воспользуйтесь тем, что необходимо! Электронная книга сама всегда помнит, на какой странице вы остановились. Вам нет необходимости загибать уголок странички, класть туда закладку[3].

Разработанное приложение представляет собой мобильное *Android* приложение для чтения электронных книг с возможностью перевода и ведения словаря.

Мобильное приложение позволяет:

– добавлять в библиотеку книги разных форматов;

– при чтении иностранных книг осуществлять перевод непонятных слов;

– сохранять перевод в словаре;

– вести отдельный словарь к каждой книге.

Разработка приложения осуществлялась в *Android Studio*.

Приложение может использоваться всеми, кто любит читать или хочет повысить уровень иностранного языка посредством чтения зарубежной литературы в оригинале.

Чтение художественной литературы поможет быстро усвоить иностранный, даже тем, кто хочет говорить на простые бытовые темы. Каждый изучающий английский хотел бы звучать, как носитель. Чтение книг на английском языке поможет вам и в этом. Во время чтения книги вы будете запоминать новые слова и фразы, обороты и идиомы. Что-то будет забываться, а что-то останется в памяти, и эти знания можно будет использовать в собственной речи. Чем больше вы читаете, тем больше сможете сказать. Ваша речь станет более сложной и красивой благодаря чтению книг[4].

* 1. Обоснование выбора языка программирования и средств разработки

Сейчас разработка программ для мобильных устройств находится на пике популярности. *Android*-устройства заняли практически доминирующую позицию на рынке. Во все годы, лидером в разработке *Android*-приложений являлся язык программирования *Java*[5].

*Java* считается одним из лучших языков для разработки приложений. Одной из основных особенностей этого языка является объектно-ориентированность. С помощью виртуальной машины *Java* программы на этом языке могут запускаться практически в любой системе. В результате *Java* получает лидерство среди *Android*-приложений. На *Java* ссылается большинство официальной документации *Google*, а найти платные и бесплатные библиотеки и руководства не составит труда – их великое множество.

При разработке на *Java* под *Android* используются не только *Java*-классы, содержащие код, но также файлы манифеста на языке *XML*, предоставляющие системе основную информацию о программе, и система автоматической сборки *Gradle*. Сейчас активно развивается язык *Kotlin*, на котором можно писать под *Android*, также для этих целей подходит и *C*++. Но всё-таки главным языком *Android* остаётся *Java*[6].

Сделать первый шаг к освоению *Java* – это установить *Android* *Studio*. Это тип программного обеспечения под названием *IDE* – *Integrated Development Environment*, или встроенная среда разработки. В комплекте с *Android Studio* идёт *Android SDK* – набор инструментов для разработки под *Android* и всё, что нужно для надёжного старта.

К сожалению, сложность *Java* мешает программировать на нём всем желающим. Как у объектно-ориентированного языка программирования у него куча особенностей в виде конструкторов классов, исключений, приводящих к падению приложений во время работы и других моментов, которые всегда необходимо учитывать при разработке. Впрочем, код на *Java* легко читается и структурируется, особенно при соблюдении принятых стандартов его оформления.

В современных условиях разработка ПО в большинстве случаев ведется с использованием интегрированных сред разработки. *IDE* имеют несомненные достоинства: процесс компиляции, сборки и запуска приложения обычно автоматизирован, в связи с чем для начинающего разработчика создать свое первое приложение труда не составляет.

* 1. Обзор существующих программных средств по теме дипломного проекта

В данной области существует достаточно много решений. Самыми приближенными аналогами разработанного программного средства являются *English Books Reader, Soft Prompt, Cool Reader* и *Moon+ Reader*.

Каждый, кто когда-либо более или менее серьезно занимался изучением иностранного языка, знает, что это очень трудоемкий процесс, требующий огромных усилий и времени, и чем ниже уровень знаний у обучаемого, тем больше усилий приходится прилагать.

*English Books Reader* позволяет значительно облегчить этот процесс. Используя приложение, Вы можете читать свои любимые книги на английском языке, превращая изучение Английского в довольно приятное занятие. Достаточно лишь иметь смартфон под рукой.

Приложение работает с форматами *TXT, EPUB* и *FB2*. Для особо ленивых в приложении есть несколько предустановленных книг. Поскольку приложение не бесплатное, а сделать возврат можно только в течение 15 минут, это дает возможность быстро “пощупать” его на своем устройстве. Форматы файлов с затрудненным доступом к тексту, либо хранящие текст в виде изображений (напр. *pdf, djvu*), по очевидным причинам не поддерживаются[8].

На рисунке 1.3 можно увидеть окно чтения в *English Books Reader*.

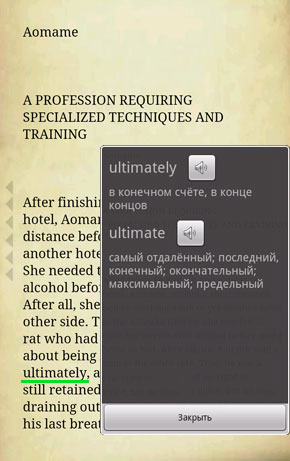


Рисунок 1.3 – *English Books Reader*

* 1. Постановка задач по разработке программного средства для чтения электронных книг и ведения словаря

В процессе разработки программного средства можно выделить несколько подзадач: анализ требований, проектирование, кодирование, тестирование и установка.

В результате анализа требований были выявлены основные требования к функциональности – добавление книг из локального хранилища в библиотеку и их удаление посредством *SQL*-запросов; ведение словаря (переводчика), закрепленного за каждой книгой; возможность выбора языка перевода.

Проектирование включало в себя продумывание структуры приложения и его графического дизайна. В приложении классы распределены по пакетам, что отражает их основное назначение. Общую структуру пакетов можно увидеть на рисунке 1.7. Так в пакете *activity* расположены три класса: *BookListActivity* – главная *activity* приложения, хранит в себе список всех добавленных книг; при нажатии на какой-либо из книг будет запущена *ReaderActivity* – получает *id* книги, загружает и отображает её для чтения; из *ReaderActivity* можно перейти в словарь текущей книги, где расположены все слова и есть возможность переводить и добавлять новые – *DictionaryActivity*.

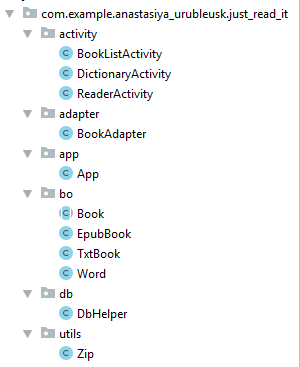


Рисунок 1.7 – Структура приложения

1. Проектирование и разработка программного средства для чтения электронных книг и ведения словаря (переводчика)
   1. Разработка объектной модели программного средства

Объектно-ориентированный анализ и проектирование принципиально отличаются от традиционных подходов структурного проектирования: здесь нужно по-другому представлять себе процесс декомпозиции, а архитектура получающегося программного продукта в значительной степени выходит за рамки представлений, традиционных для структурного программирования. Отличия обусловлены тем, что структурное проектирование основано на структурном программировании, тогда как в основе объектно-ориентированного проектирования лежит методология объектно-ориентированного программирования.

Методы структурного проектирования помогают упростить процесс разработки сложных систем за счет использования алгоритмов как готовых строительных блоков. Аналогично, методы объектно-ориентированного проектирования созданы для того, чтобы помочь разработчикам применять мощные выразительные средства объектного и объектно-ориентированного программирования, использующего в качестве блоков классы и объекты.

Объектно-ориентированная технология основывается на так называемой объектной модели. К основным элементам объектной модели относятся – объект, класс, атрибут, операция, интерфейс, компонент, связи.

Объектопределяется как предмет или явление, имеющие четко определяемое пове­дение. Объект может представлять собой абстракцию некоторой сущности предметной области или прог­раммной системы. Любой объект обла­дает состоянием, поведениеми индивидуальностью.

Состояние объекта – одно из возможных условий, в которых он может существовать, оно изменяется со временем. Состояние объекта определяется значениями его свойств и связями с другими объек­тами.

Поведениеопределяет действия объекта и его реакцию на зап­росы от других объектов. Поведение объекта полностью определяется его действиями. Поведение представляется с помощью набора сообщений*,*воспринимаемых объектом (операций*,*которые может выполнять объект).

Класс***–*** это множество объектов, связанных общностью свойств, поведения, связей и семантики. Любой объект является экземпляром класса. Определение классов и объектов – одна из самых сложных задач объектно-ориентированного проектирования.

* 1. Разработка архитектуры программного средства

Архитектура программного средства – концепция, которую просто понять, и которую большинство инженеров, особенно с небольшим опытом, интуитивно чувствуют, но которой трудно дать точное определение. В частности, трудно провести четкую линию между дизайном и архитектурой: архитектура является одним из аспектов дизайна, сконцентрированным на некоторых конкретных особенностях.

Но архитектура – это более чем просто структура; рабочая группа *IEEE* по архитектуре определяет её как «концепция системы на самом высоком уровне в её окружении». Также она включает в себя соответствие требованиям целостности системы, экономическим ограничениям, эстетическим требованиям, и общему стилю. Архитектура не ограничивается взглядом «внутрь», но также рассматривает систему в целом в среде её пользователей и в среде её разработки – то есть, учитывается и окружение.

Архитектура ПС – это его строение как оно видно, т.е. представление ПС как системы, состоящей из некоторой совокупности взаимодействующих подсистем. В качестве таких подсистем выступают обычно отдельные программы. Разработка архитектуры является первым этапом борьбы со сложностью ПС, на котором реализуется принцип выделения относительно независимых компонент.

Основные задачи разработки архитектуры ПС:

– выделение программных подсистем и отображение на них внешних функций (заданных во внешнем описании) ПС;

– определение способов взаимодействия между выделенными програм-мными подсистемами.

Различают следующие основные классы архитектур программных средств:

– цельная программа;

– комплекс автономно выполняемых программ;

– слоистая программная система;

– коллектив параллельно выполняемых программ.

Цельная программа представляет вырожденный случай архитектуры ПС: в состав ПС входит только одна программа. Такую архитектуру выбирают обычно в том случае, когда ПС должно выполнять одну какую-либо ярко выраженную функцию и ее реализация не представляется слишком сложной. Естественно, что такая архитектура не требует какого-либо описания, так как определять способ взаимодействия не требуется.

Разрабатываемое приложение является цельной программой. В нём классы распределены по пакетам, что отражает их основное назначение. Так в пакете *activity* расположены три класса: *BookListActivity* – главная *activity* приложения, хранит в себе список всех добавленных книг; при нажатии на какой-либо из книг будет запущена *ReaderActivity* – получает *id* книги, загружает и отображает её для чтения; из *ReaderActivity* можно перейти в словарь текущей книги, где расположены все слова и есть возможность переводить и добавлять новые – *DictionaryActivity*.

Вообще *activity* представляет собой внешний пользовательский интерфейс для одной операции, которую может совершить пользователь. Если упростить, то это просто один текущий экран как некоторая единица активности, своего рода кадр с одним пользовательским действием. Здесь и далее будет использоваться термин *activity* без перевода, как некоторое имя собственное. Хотя дословный перевод, «активность» или «действие», хорошо передает общий смысл компонента. На рисунке 2.2 можно увидеть жизненный цикл *activity*.



Рисунок 2.2 – Жизненный цикл *activity*

* 1. Разработка алгоритмов функционирования программного средства

Алгоритм – это точно установленное предписание (инструкция) о выполнении в определённом порядке некоторой последовательности операций, однозначно ведущих к решению той или иной конкретной задачи.

Предписание алгоритма представляет собой конечный набор правил, который задаёт потенциально осуществимый вычислительный процесс (процесс исполнения алгоритма), ведущий от варьирующих в определённых пределах исходных данных к получению результата, однозначно определяемого допустимыми исходными данными. Последнее подразумевает, что результат выполнения алгоритма напрямую зависит от исходных данных: то есть один и тот же алгоритм при разных исходных данных даст разные результаты; с другой стороны, если одному и тому же алгоритму передать несколько раз одни и те же данные, он должен столько же раз выдать один и тот же результат. Простейшими примерами алгоритмов являются арифметические правила сложения, вычитания, умножения, деления и тому подобные.

Предписание алгоритма, как правило, фиксируется в виде текста на некотором формализованном языке, называемого программой. Понятие программы формулируется в чисто структурных терминах синтаксиса этого языка, без какого-либо обращения к смысловым категориям. Точно такой же характер носит и описание процедуры выполнения программы. Поэтому в роли исполнителя алгоритмов, записанных на формализованных языках, может выступать не только человек, но и наделённое соответствующими возможностями автоматическое устройство, машина.

Универсальным исполнителем алгоритмов является компьютер. С его помощью возможно выполнять все основные виды алгоритмов. Возможность машинного осуществления алгоритмических процедур, и, соответственно, машинного решения задач, стимулировала развитие вычислительной техники и создание математической теории алгоритмов.

В современной науке теория алгоритмов является основой конструктивного направления в математике и логике, а также выступает одной из базисных дисциплин в области вычислительной техники и программирования, машинного решения разнообразных задач, моделирования различных процессов и других областях.

Различают следующие виды алгоритмов:

1 Линейный – список команд (указаний), выполняемых последовательно друг за другом.

2 Разветвляющийся – алгоритм, содержащий хотя бы одну проверку условия, в результате которой обеспечивается переход на один из возможных вариантов решения.

3 Циклический – алгоритм, предусматривающий многократное повторение одной и той же последовательности действий. Количество повторений обусловливается исходными данными или условием задачи.

В приложении чаще всего используется разветвляющийся вид алгоритмов, для наглядности его общая схема приведена на рисунке 2.3.

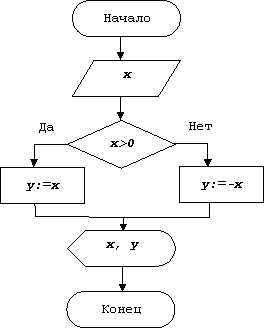


Рисунок 2.3 – Схема разветвляющегося алгоритма

На схеме наглядно проявляется важное свойство ветвящихся алгоритмов: их исполнение всегда проходит только по одному из возможных путей, который определяется конкретными текущими условиями, причем в каждом случае от начала алгоритма (входа) до его конца (выхода). Это свойство присуще всякому логически правильно составленному алгоритму и является признаком правильной организации ветвлений.

Существует широкий круг задач, при решении которых необходимо сделать определенный выбор в зависимости от выполнения некоторых условий. В разветвляющихся алгоритмах принцип линейного автоматического перехода от команды к команде не является всеобщим. Ветвящиеся алгоритмы допускают два способа представления – графический и словесный.

* 1. Разработка и обоснование пользовательского интерфейса программного средства

Пользовательский интерфейс часто понимают только как внешний вид программы. Однако, на деле пользователь воспринимает через него всю программу в целом, а значит, такое понимание является слишком узким.

Одним из важных показателей качества программного обеспечения является удобство его использования. Оно описывается с помощью таких характеристик, как понятность пользовательского интерфейса, легкость обучения работе с ним, трудоемкость решения определенных задач с его помощью, производительность работы пользователя с ПО. Для построения действительно удобных программ нужен учет контекста их использования, психологии пользователей, необходимости помогать начинающим пользователям и предоставлять все нужное для работы опытных. Однако самым значимым фактором является то, помогает ли данная программа решать действительно значимые для пользователей задачи[15].

Цель создания эргономичного интерфейса состоит в том, чтобы отобразить информацию настолько эффективно насколько это возможно для человеческого восприятия и структурировать отображение на экране монитора таким образом, чтобы привлечь внимание к наиболее важным единицам информации. Основная же цель состоит в том, чтобы минимизировать общую информацию на экране и представить только то, что является необходимым для пользователя.

Пользовательский интерфейс для *Android*-приложений может быть разработан с помощью одного из двух методов: процедурного и декларативного. Процедурная разработка заключается в использовании программного кода. Например, если разрабатывается приложение *Swing*, для создания и манипулирования всеми объектами пользовательского интерфейса необходимо писать код.

Декларативный дизайн, с другой стороны, не включает в себя никакого кода, а предполагает объявление интерфейса в *XML*-файлах. Затем эти *XML*-определения визуального отображения загружаются в приложение пользовательского интерфейса в виде окон. Это очень похоже на веб-страницы на основе *HTML*. Язык разметки *XML* описывает то, что необходимо отобразить на странице, но не то, как эта страница должна себя вести. *XML* – это декларативный язык.

*Google* советует использовать декларативный *XML* всегда, когда это возможно. *XML*-код короче и проще для понимания, чем аналогичный *Java*-код, и это вряд ли изменится в будущих версиях[16].

При разработке интерфейса рассматриваемого программного средства использовался декларативный подход. В папке *layout* были созданы пять *XML*-файлов для описания главного окна и отдельных элементов списка библиотеки, окна режима чтения, окна словаря и каждого элемента его списка.

Для описания главного экрана программного средства используется *ReiativeLayout* разметка, которая позволяет дочерним компонентам определять свою позицию относительно родительского компонента или относительно соседних дочерних элементов. Как ни странно разметка главного окна содержит только один дочерний элемент *RecyclerView,* который помечается идентификатором *book\_list\_holder* и определяет список книг в библиотеке.

Описание внешнего вида отдельного элемента каждой книги в списке задаётся в файле *book\_list\_item.xml.* Он использует *LinearLayout* контейнер для размещения элементов, который выравнивает все дочерние объекты в одном направлении – вертикально или горизонтально. Направление задается при помощи атрибута ориентации *android:orientation.* Для размещения элементов книг применяется вертикальная ориентация. Книга в библиотеке определяется с помощью элемента *TextView,* который отображает её название и задаётся идентификатором *book\_title.*

Содержимое файла для описания главного окна программного средства можно увидеть на рисунке 2.4.



Рисунок 2.4 – Содержимое файла *activity\_book\_list.xml*

Кроме списка книг главное окно содержит пункты меню, которые описываются в специальном файле *options.xml* папки *menu*. Этот файл содержит два элемента *item,* которые определяют кнопку добавления новой книги на панели меню и кнопку для просмотра информации о приложении.

Все элементы интерфейса пользователя в приложении *Android* создаются с помощью объектов *View* и *ViewGroup*. Объект *View* формирует на экране элемент, с которым пользователь может взаимодействовать. Объект *ViewGroup* содержит другие объекты *View* для определения макета интерфейса[17].

Наибольшее количество *View* элементов содержится в файле *activity\_reader.xml,* который определяет внешний вид окна режима чтения. Так элемент  *WebView* служит для размещения содержимого книги и определяется в дочерней разметке *FrameLayout,* которая является самым простым типом разметки. Это пустое пространство на экране, которое можно заполнить только дочерними объектами *View* или *ViewGroup*. Все дочерние элементы *FrameLayout* прикрепляются к верхнему левому углу экрана.

В другой дочерней разметке *LinearLayout* содержатся кнопки нижней панели. Они задаются идентификаторами *prev\_button, next\_button* и *dictionary\_button* и используются для пролистывания книги и открытия словаря (переводчика).

Файл для описания внешнего вида окна словаря также содержит большое количество элементов. Среди них – *Spinner* для задания выпадающего списка поддерживаемых языков, *EditText* для ввода и вывода перевода и *Button* для определения кнопок добавления и перевода слов.

С описанием кнопки перевода можно ознакомиться на рисунке 2.5.



Рисунок 2.5 – Описание кнопки перевода

Разработанное программное средство имеет простой интерфейс, состоящий из главного окна библиотеки, окна режима чтения и окна словаря. Для построения графического интерфейса использовались такие элементы как *Button, Menu, EditText, TextView, Spinner, RecyclerView, WebView*.

Окно библиотеки содержит список добавленных книг, представленный *RecyclerView*, панель меню с названием приложения и кнопкой для добавления новой книги. Чтобы удалить книгу, необходимо осуществить длительный клик по ней, для вызова соответствующего контекстного меню с возможностью выбора удаления. При коротком клике по книге откроется окно режима чтения. Это окно построено на основе *WebView* элемента. Скроллинг книги возможен двумя способами: проведением пальцев по экрану вниз-вверх или нажатием кнопок вперед-назад, расположенных в нижней части текущего окна. Там же находится кнопка для открытия словаря.

На рисунке 2.6 изображено окно библиотеки приложения.

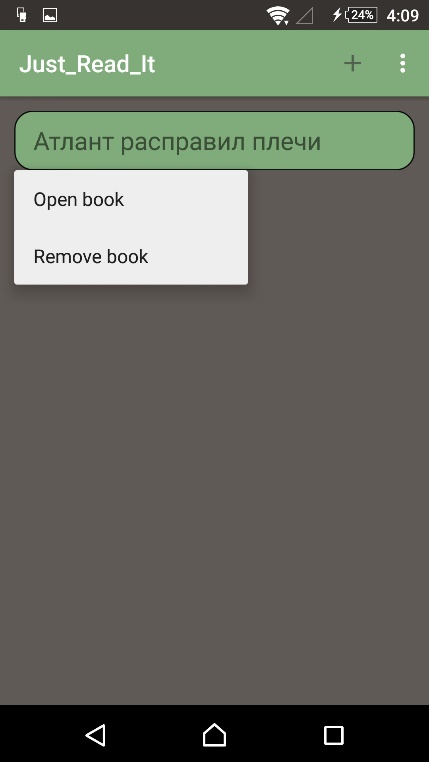


Рисунок 2.6 – Окно библиотеки

Окно словаря содержит два поля ввода для написания слов, список языков для конфигурации перевода, кнопку для вызова метода перевода, кнопку для добавления пары слово-перевод в словарь и список всех ранее добавленных слов.

* 1. Проектирование базы данных и интерфейсов взаимодействия с ней

Данные всегда были самой важной частью любого мобильного приложения. Разработчикам нужно хранить значительные объемы данных в организованном порядке для того, чтобы их приложения, по-настоящему были ценными для своих пользователей.

В *Android*, можно хранить данные, используя довольно много различных методов, как в активной памяти так и в файле. Многие приложения также имеют удаленные веб-службы, которые предоставляют данные о приложении. *Android* также поддерживает хранение данных в локальной базе данных, и операционная система обеспечивает хорошую инфраструктуру для хранения и извлечения данных. В большинстве случаев, самым простым способом приобретения и сохранения пользовательских данных является использование баз данных *SQLite*.

*SQLite* – компактная встраиваемая реляционная база данных. Исходный код библиотеки передан в общественное достояние. В 2005 году проект получил награду *Google-O’Reilly Open Source Awards*.

Слово «встраиваемый» (*embedded*) означает, что *SQLite* не использует парадигму клиент-сервер, то есть движок *SQLite* не является отдельно работающим процессом, с которым взаимодействует программа, а представляет собой библиотеку, с которой программа компонуется, и движок становится составной частью программы. Таким образом, в качестве протокола обмена используются вызовы функций (*API*) библиотеки *SQLite*. Такой подход уменьшает накладные расходы, время отклика и упрощает программу.

*SQLite* хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том компьютере, на котором исполняется программа. Простота реализации достигается за счёт того, что перед началом исполнения транзакции записи весь файл, хранящий базу данных, блокируется; *ACID*-функции достигаются в том числе за счёт создания файла журнала[18].

В *SQLite* применяется следующая система типов данных:

– *NULL* представляет нулевое значение;

– *INTEGER* представляет целое число, аналог типу *int* в *java*;

– *REAL* представляет число с плавающей точкой, аналог *float* и *double*;

– *TEXT* представляет набор символов, аналог *String* и *char* в *java*;

– *BLOB* представляет массив бинарных данных, изображение, аналог типу *int* в *java*.

* 1. Описание взаимодействия программного средства *с Google Translation API*

*API* (сокращение от английского *Application Programming Interface*) – это интерфейс программирования приложений. Интернет-сервисы, библиотеки, операционная система и др. предоставляют разработчикам интерфейсы для взаимодействия со своими наборами готовых классов, методов и данных. В результате этого можно в своих приложениях использовать функции других интернет-сервисов.

*API* предоставляют многие сервисы, в частности это социальные сети ВКонтакте, *Instagram* и *Twitter*; также это могут быть всевозможные переводчики, конвертеры валют, информационные и новостные сервисы. Кроме того, можно самостоятельно сделать *API* на собственном *web*-ресурсе и использовать его при разработке своих приложений, либо сделать его публичным – и тогда другие люди смогут им воспользоваться.

*API* для перевода предоставляют два крупнейших сервиса – *Google* и Яндекс. «Яндекс.Переводчик» – интернет-сервис, предоставляющий онлайно-вые инструментальные средства для автоматического перевода слов, фраз и предложений с одного языка на другой.

С помощью *API* можно получить доступ к онлайн-сервису машинного перевода Яндекса. Он поддерживает более 90 языков и умеет переводить отдельные слова и целые тексты. Этот *API* позволяет встроить Яндекс.Переводчик в мобильное приложение или веб-сервис для конечных пользователей. Или же переводить большие объёмы текста – например, техническую документацию[21].

В разработанном программном средстве используется *Google Translate API*. Он входит в большой набор других *Google API* сервисов. *Google APIs* – это набор интерфейсов прикладного программирования (*API*), разработанный *Google*, которые позволяют взаимодействовать со службами *Google* и интегрировать их в другие службы. Примеры таких интерфейсов – Поиск, *Gmail, Translate* и *Google Maps*. Сторонние приложения могут использовать эти *API* для использования или расширения функциональности существующих служб.

*API* предоставляют такие функции, как аналитика, машинное обучение или доступ к пользовательским данным (когда предоставляется разрешение на чтение данных). Другим важным примером является встроенная карта *Google* на веб-сайте, чего можно добиться с помощью *Static maps API, Places API* или *Google Earth API*.

1. Инженерные расчёты, используемые в дипломном проекте
   1. Оценка объёма передаваемого трафика между программным средством и *Google Translation API*

Интернет-трафик – объем данных, которыми обмениваются между собой компьютеры и прочие устройства, подключенные к сети Интернет. Количество трафика измеряется в гигабайтах, мегабайтах, килобайтах.

Когда речь идет об обычном интернет-трафике, его можно делить на входящий и исходящий. Входящий трафик принимается устройством из интернета. Исходящий трафик передает устройство в сеть[25].

В соответствии с природой трафика измерение в течение ограниченного промежутка времени соответствует регистрации конкретной реализации процесса переноса трафика. Таким образом, измерение - это выборка одной и нескольких случайных величин.

Любое измерение в рамках процесса передачи трафика может быть реализовано с помощью объединения двух фундаментальных операций:

1 Число событий – например, количество звонков, количество ошибок (ложных звонков), количество прерванных звонков и т.д.

2 Временные интервалы – например, времена разговоров, времена ожидания соединения и т.д.

С помощью этих двух операций можно получить любую характеристику трафика.

Наиболее важными характеристиками являются объем (передаваемого) трафика, то есть сумма всех времен пребывания на линии в рамках заданного периода измерения.

С функциональной точки зрения методы измерения трафика могут быть разделены на следующие два класса:

– непрерывные методы измерения;

– дискретные методы измерения[26].

Для оценки передаваемого трафика было проведено тестирование, при котором приложение многократно обратилось к *API* для перевода с английского на русский в течение небольшого промежутка времени. Количество запросов за 5 минут – 49. Средняя длина слова – 7 символов. Если посмотреть на график в *console.cloud.google.com*, изображенный на рисунке 3.1, то передаваемый трафик со стороны *Google Translation API* колеблется от 0,033 до 0,05 кб/с.



Рисунок 3.1 – График передаваемого трафика

Аналогичное тестирование проводилось для разных комбинаций языков: перевод с белорусского на русский, с польского на русский, с испанского на русский, с итальянского на русский, с французского на русский, с немецкого на русский. В ходе тестирования были выбраны небольшие тексты на соответствующих языках. Среднее количество слов в тексте – 40.

При этом наибольший расход трафика отмечен при переводе с белорусского на русский – 14,32 кб. Стоит отметить, что и по времени данный перевод осуществлялся дольше остальных. Расход трафика при переводе с белорусского на русский приведён на рисунке 3.2.

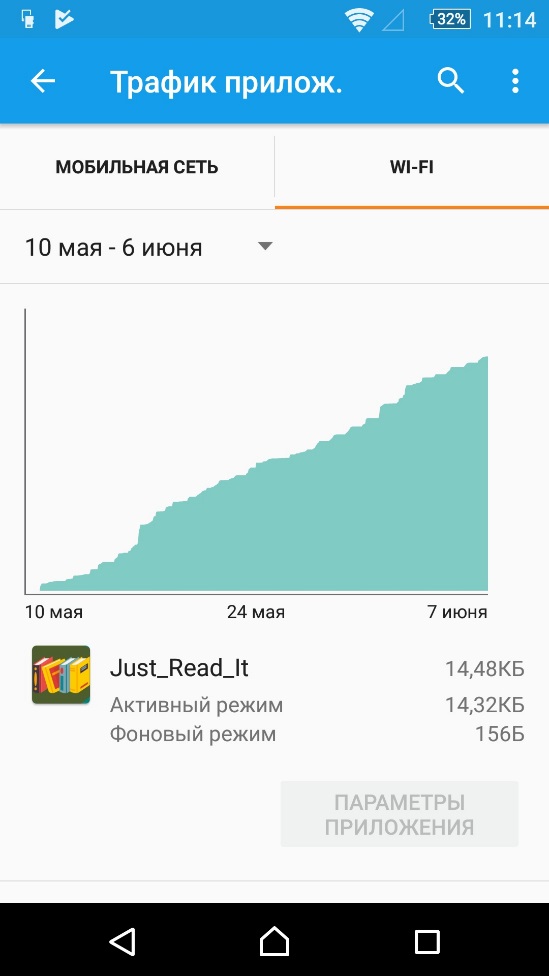


Рисунок 3.2 – Расход трафика при переводе с белорусского на русский

1. Эксплуатация программного средства для чтения электронных книг и ведения словаря (переводчика)
   1. Ввод в эксплуатацию программного средства

Программное средство уже можно считать полностью готовым к эксплуатации. Однако в течение первого месяца приложение будет использоваться ограниченным кругом пользователей для проведения альфа-тестирования в качестве внутреннего приёмочного тестирования. Затем будет организованно бета-тестирование с привлечением добровольцев из числа обычных будущих пользователей продукта, которым будет доступна предварительная версия продукта (так называемая бета-версия).

После окончания упомянутых стадий тестирования и исправления ошибок планируется загрузка приложения в *Google Play*. Любое приложение, выкладываемое в магазин, должно иметь подписанный сертификат. Сертификат позволяет идентифицировать вас как автора программы. И если кто-то попытается выложить программу с таким же именем как у вас, то ему будет отказано из-за конфликта имён. Под именем приложения имеется в виду полное название пакета.

Чтобы подписать приложение уникальным сертификатом необходимо в *Android Studio* выбрать меню *Build – Generate Signed APK*. Появится диалоговое окно мастера, как на рисунке 4.1, которое необходимо заполнить данными.

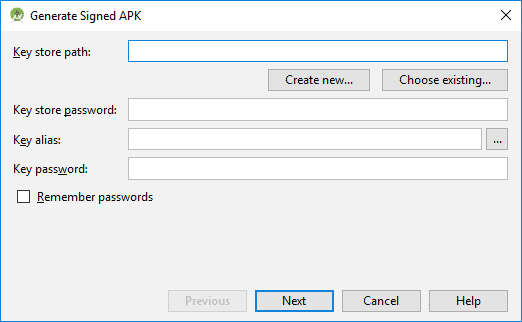


Рисунок 4.1 – Создание уникального сертификата

После заполнения всех полей должно появиться диалоговое окно с подтверждением создания сертификата, как на рисунке 4.2.



Рисунок 4.2 – Уведомление о создании сертификата

В *Android Studio* предусмотрен режим автоматического создания подписанного приложения. Щёлкните правой кнопкой мыши на папке *app* и в контекстом меню выберите пункт *Open Module Settings*. Выберите раздел *app* в секции *Modules*. После заполнения полей во вкладке *Signing*, переходите на вкладку *Build Types* и выбирайте сборку *release*. В выпадающем списке *Signing* *Config* выбирайте только что созданную конфигурацию. По умолчанию она имеет имя *config*.

Окно создания подписанного проекта изображено на рисунке 4.3.

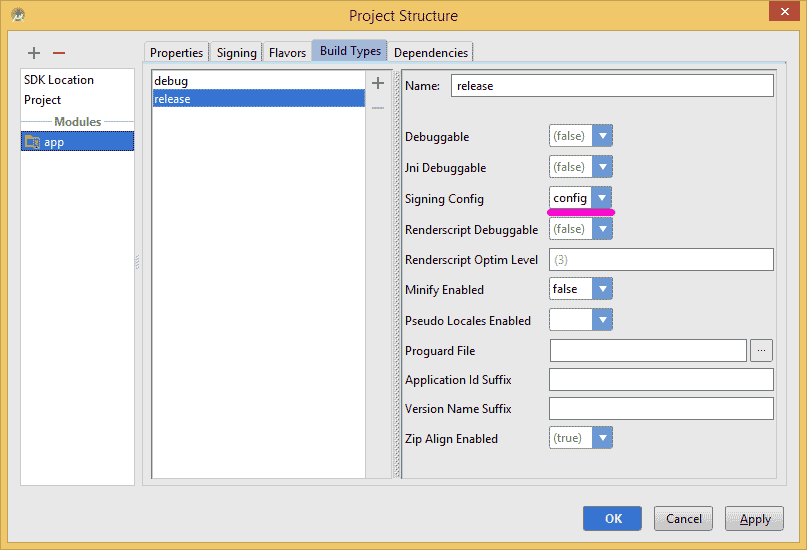


Рисунок 4.3 – Создание подписанного приложения

Для того, чтобы опубликовать приложение в *Google Play*, нужен аккаунт разработчика. Это особый тип аккаунта, который позволяет выкладывать приложение в *Google Play*. Создать его можно с помощью стандартного аккаунта *Google*. Необходимая часть процесса – внесение разовой оплаты за соглашение разработчика в размере 25 долларов. Уведомление о взносе изображено на рисунке 4.4.

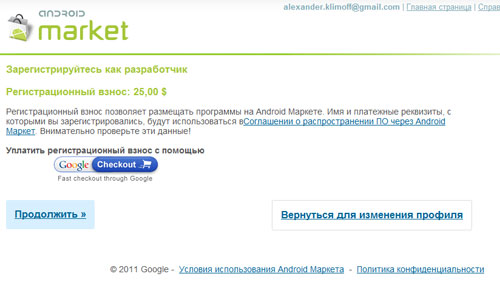


Рисунок 4.4 – Регистрационный взнос в *Google Play*

Если платёж прошел успешно, то ссылка на следующий шаг будет доступна и вы попадаете в специальный личный кабинет, где можете добавлять свои приложения. Загружаем подготовленный *APK*-файл, а также необходимые картинки-скриншоты и значок[30]. Часть страницы загрузки приложения в личном кабинете можно увидеть на рисунке 4.5.

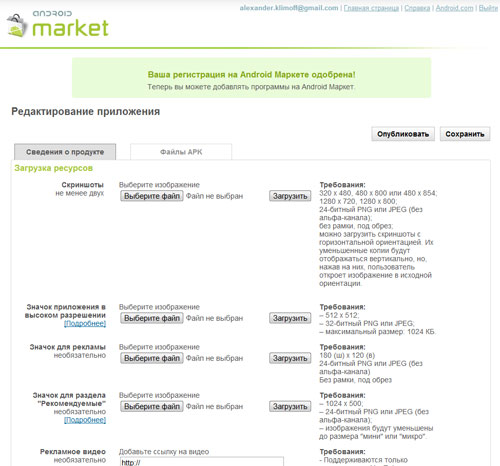


Рисунок 4.5 – Личный кабинет в *Google Play*

* 1. Руководство к использованию разработанного программного средства

После запуска приложения перед пользователем откроется главное окно, так называемая библиотека, со всеми добавленными книгами. Книги размещаются в вертикальном списке с использованием элемента *RecyclerView*. Здесь можно выбрать книгу для перехода в режим чтения или для её удаления из библиотеки. При длительном клике по книге открывается контекстное меню с возможностью её открытия или удаления. Чтобы добавить новую книгу необходимо нажать значок плюса на панели меню, при этом откроется окно локального хранилища.

Главное окно приложения изображено на рисунке 4.6.

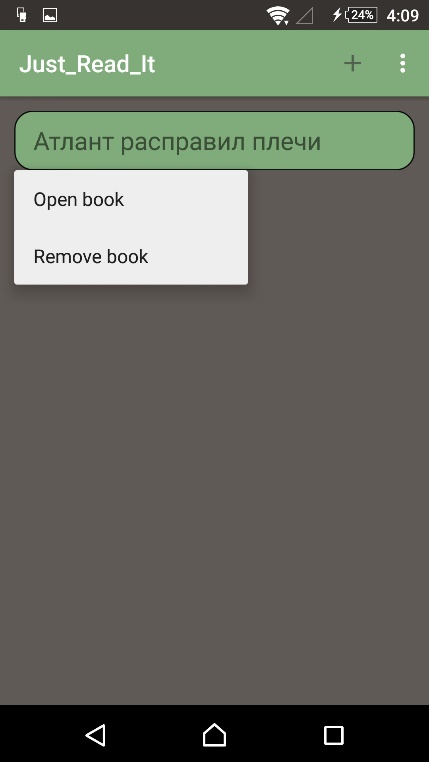


Рисунок 4.6 – Главное окно приложения

**5 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНОГО СРЕДСТВА ДЛЯ ЧТЕНИЯ ЭЛЕКТРОННЫХ КНИГ И ВЕДЕНИЯ СЛОВАРЯ (ПЕРЕВОДЧИКА)**

**5.1** **Описание функций, назначения и потенциальных пользователей ПО**

Одной из важнейших стадий оценки проекта является разработка ТЭО. Оно представляет собой официальный документ, содержащий все необходимые исследования, дающие возможность сделать обоснованный предварительный вывод о целесообразности инвестирования в конкретный проект.

Главной задачей при составлении ТЭО является оценка затрат на инвестиционный проект и его результатов, анализ срока окупаемости проекта. Поэтому разрабатываемое ПО должно быть не только совершенным в техническом плане, но и выгодными с точки зрения экономики.

Целью экономического расчёта является определение с экономической точки зрения целесообразности разработки, внедрения и использования данного программного средства, относительно его аналогов.

Разработанное приложение представляет собой мобильное *Android*- приложение для чтения электронных книг с возможностью перевода и ведения словаря.

Мобильное приложение позволяет:

– добавлять в библиотеку книги разных форматов;

– при чтении иностранных книг осуществлять перевод непонятных слов;

– сохранять перевод в словаре;

– вести отдельный словарь к каждой книге.

Разработка приложения осуществлялась в *Android Studio*.

Данное приложение разрабатывается *EPAM* и предназначено для использования широким кругом пользователей и свободного распространения на рынке *IT*. Главная цель проекта – увеличить количество читающих пользователей, сделать более удобным процесс чтения книг в оригинале.

Приложение может использоваться всеми, кто любит читать или хочет повысить уровень иностранного языка.

Реализация приложения планируется через *Google Play Market* бесплатно с добавлением платной рекламы.

**5.2 Расчёт затрат на основную заработную плату команды разработчиков**

Основная заработная плата исполнителей, занятых разработкой программного модуля, рассчитывается по формуле (5.1):

где *n* – количество исполнителей, занятых разработкой ПО;

Кпр – коэффициент премий (1,5-2,0);

Зч*i* – часовая заработная плата *i*-го исполнителя (руб.);

*ti* – трудоемкость работ, выполняемых *i*-м исполнителем (ч).

Расчет основной заработной платы команды разработчиков представлен в таблице 5.1.

Таблица 5.1 – Расчет затрат на основную заработную плату команды разработчиков

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Участник команды | Вид выполняемой работы | Месячная заработная плата, р. | Часовая заработная плата, р. | Трудоемкость работ, ч. | Зарплата по тарифу, р. |
| Программист | Разработка приложения | 1278 | 7,9 | 70 | 553 |
| Тестировщик | Тестирование приложения | 1065 | 6,7 | 20 | 134 |
| Руководитель | Организация процесса | 1000 | 6,25 | 10 | 62,5 |
| Всего | | | | | 749,5 |
| Премия (50%) | | | | | 374,75 |
| Итого затраты на основную заработную плату разработчиков | | | | | 1124,25 |

**5.3 Расчёт затрат на дополнительную заработную плату команды разработчиков**

Дополнительная заработная плата исполнителей, занятых разработкой программного модуля, рассчитывается по формуле (5.2):

где Зо – затраты на основную заработную плату, (руб.);

Нд – норматив дополнительной заработной платы (10-20%).

Дополнительная заработная плата исполнителей составит:

**5.4 Расчёт отчислений на социальные нужды**

Отчисления в фонд социальной защиты населения и на обязательное страхование определяются по формуле (5.3):

где Нсоц – норматив отчислений на социальные нужды (34+0,6%).

Отчисления в ФСЗН и обязательное страхование составят:

**5.5 Расчёт прочих затрат**

Прочие затраты включаются в себестоимость разработки ПО в процентах от затрат на основную заработную плату по формуле (5.4):

где Нпз – норматив прочих затрат (50%).

Полная сумма затрат на разработку программного обеспечения представлена в таблице 5.2.

Таблица 5.2 – Затраты на разработку программного обеспечения

|  |  |
| --- | --- |
| Статья затрат | Сумма, руб. |
| Основная заработная плата команды | 1124,25 |
| Дополнительная заработная плата команды | 112,425 |
| Отчисления на социальные нужды | 427,9 |
| Прочие затраты | 562,125 |
| Общая сумма затрат на разработку | 2226,7 |

**5.6 Расчет экономической эффективности от реализации программного модуля на рынке**

Как показывает практика многолетней работы *Google Play*, наибольший доход приходится на долю бесплатных приложений. Цена клика в среднестатистическом *Android*-приложении – 5 центов (согласно [*https://vipidei.com/internet/mobilnye-prilozheniya/prodvizhenie-v-google-play/*](https://vipidei.com/internet/mobilnye-prilozheniya/prodvizhenie-v-google-play/)).

Тогда при просмотре баннера рекламы 100 раз в день, прогнозный доход от размещения платной рекламы на мобильном приложении, составит 320 руб. в месяц и 3840 руб. в год (на момент расчёта 1 *USD* = 2,13 *BYN*).

Прибыль, полученная от реализации ПО на рынке (учитывая, что *EPAM*-резидент ПВТ освобожден от уплаты НДС и налога на прибыль), рассчитывается по формуле (5.5):

где Др – доход от размещения рекламы на мобильном приложении, реализуемом бесплатно, руб.

Тогда прибыль за первый год составит:

**5.7 Расчет показателей эффективности инвестиций в разработку ПО**

Для оценки эффективности затрат в разработку ПО необходимо рассчитать уровень рентабельности затрат по формуле (5.6):

Поскольку приложение станет приносить прибыль уже на седьмой месяц после его выхода на рынок и рентабельность затрат на его разработку составляет 72%, можно сделать вывод об экономической целесообразности разработки программного средства для чтения электронных книг и ведения словаря (переводчика).

Заключение

Результатом дипломного проектирования является разработанное программное средство для чтения электронных книг и ведения словаря (переводчика). При разработке были учтены основные требования к его функциональности: добавление книг из локального хранилища в библиотеку и их удаление посредством *SQL*-запросов; ведение словаря (переводчика), закрепленного за каждой книгой; возможность выбора языка перевода.

Перед началом разработки был проведен анализ аналогичных приложений и выполнено технико-экономическое обоснование проекта с целью определения экономической целесообразности разработки, внедрения и использования данного программного средства, относительно его аналогов.

Для разработки программного средства использовалась интегрированная среда разработки *Android Studio*. Программное средство предназначено для смартфонов с операционной системой *Android* с версией от 4.4 до 9.

В процессе выполнения дипломной работы использовались научно-техническая литература и материалы с сайтов, посвящённых программированию для мобильных телефонов с операционной системой *Android*.

Приложение было протестировано и показало свою работоспособность как на стандартных эмуляторах, взятых из *AVD Manager*, так и на реальном устройстве на платформе *Android.*

Подводя итоги дипломного проектирования, можно сделать вывод о выполнении поставленных задач и достижении главной цели работы – разработки полноценного программного средства для чтения электронных книг и ведения словаря (переводчика).

Список использованных источников

**ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Отчёт по анализу заимствования материала**

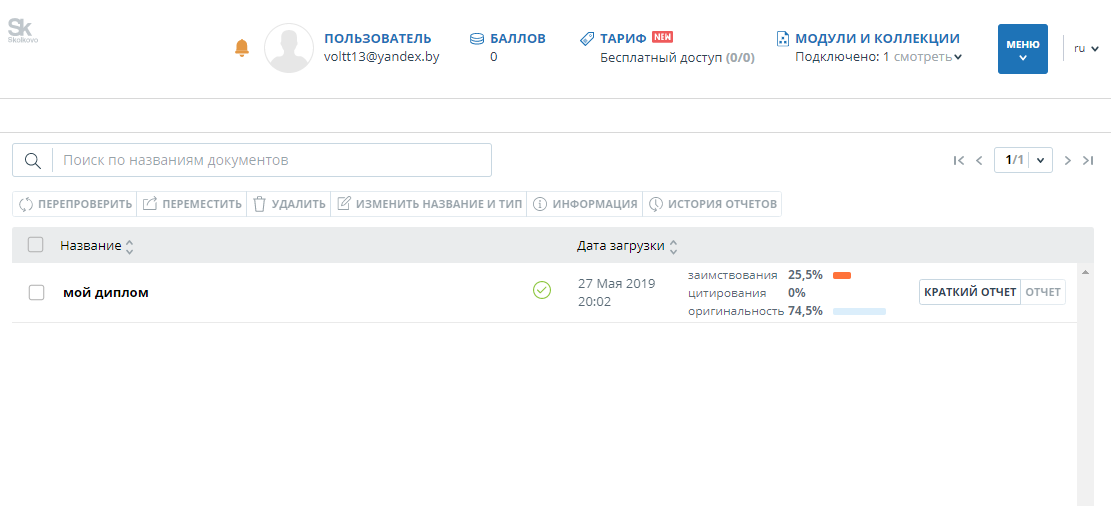
****

Рисунок А.1 – Скриншот проверки дипломного проекта

в системе «Антиплагиат»

**ПРИЛОЖЕНИЕ Б**

**(обязательное)**

**Листинги программ**

**ПРИЛОЖЕНИЕ В**

**(обязательное)**

**Ведомость дипломного проекта**