# СТРУКТУРА ПРЕДПРИЯТИЯ. ХАРАКТЕРИСТИКА ОСНОВНЫХ ВИДОВ ДЕЯТЕЛЬНОСТИ

Прохождение технологической практики было осуществлено в ООО «Девкрафт». Предметом деятельности организации является разработка программного обеспечения web-технологий широкого круга agile-методологией.

Создания программного обеспечения в ООО «Девкрафт» включают в себя процессы постановки задачи, разработки и сопровождения программного обеспечения в течении всего жизненного цикла. Схема структуры организации изображена на рисунке 1.

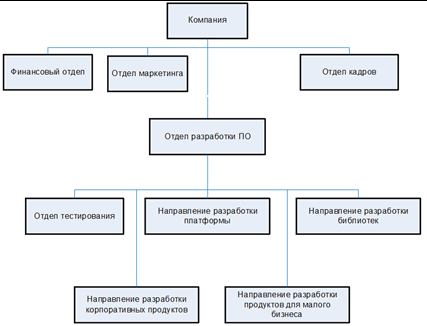


Рисунок 1 – Схема структуры организации ООО «Девкрафт»

Финансовый и маркетинговый отделы занимаются созданием определённой системы мероприятий, направленной на регулирование процессов купли-продажи, удовлетворение спроса и получения прибыли. Отдел разработки, включающий различные направления и тестирование, занимается разработкой и сопровождением программного обеспечения. Также входит HR-отдел.

Организация выполняет задачи, такие, как:

1. реализация информационно технологических проектов:
2. определение требований;
3. анализ и проектирование;
4. реализация;
5. тестирование;
6. развертывание.
7. обеспечение работоспособности информационных систем;
8. ведение бюджета информационных технологий, учет активов, обеспечение кадрового состава.

Организация постоянно развивается и расширяется, успешно справляясь с поставленными задачами заказчиков.

# ОПИСАНИЕ СТРУКТУРЫ ОТДЕЛА, ГДЕ ПРОХОДИЛА ПРАКТИКА

Технологическая практика проходила в отделе разработки программного обеспечения.

Отдел разработки осуществляет следующие задачи:

1. анализ требований к проекту;
2. проектирование;
3. разработка прототипов;
4. разработка программного обеспечения;
5. тестирование продукта;
6. внедрение и поддержка.

Схематичное выполнение отображено на рисунке 2.

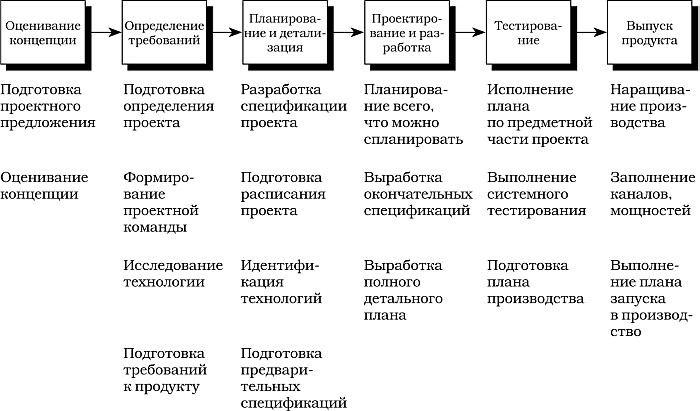


Рисунок 2 – Схема работы отдела разработки

В отделе разработки осуществляют работу множество компьютеров, связанных между собой одной сетью.

Схема подключения компьютеров по локальной сети, отображена на рисунке 3.

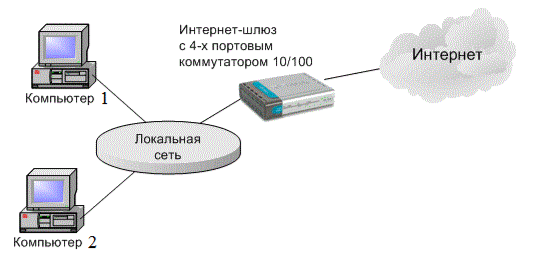


Рисунок 3 – Схема подключения компьютеров по локальной сети

На данных компьютерах установлена операционная система Linux семейства Ubuntu 18 LTS. Работа осуществляется в следующих интеграционных системах разработки:

1. WebStorm – мощная IDE для веб-разработки на JavaScript. Отличается удобным и умным редактором JavaSсript, HTML и CSS и поддержкой новых технологий и языков, таких как TypeScript, CoffeeScript, Dart, Less, Sass и Stylus.;
2. PhpStorm – это умная IDE для языка PHP и других веб-технологий, понимающая код и отличающаяся интеллектуальным редактором, автодополнением кода, рефакторингами, встроенным отладчиком и другими инструментами;
3. Rider ­– кроссплатформенная IDE для платформы .NET, построенная на базе IntelliJ IDEA и ReSharper. Поддерживает C#, VB.NET, ASP.NET, XAML, XML, JavaScript, TypeScript, JSON, HTML, CSS и SQL.;
4. Visual Studio Code – редактор исходного кода, разработанный Microsoft, для кроссплатформенной разработки веб- и облачных приложений.

Выше были перечислены основные программные средства. Также используется множество вспомогательного программного обеспечения, начиная web-браузерами и заканчивая плагинами для интеграционных систем разработок.

# ДОЛЖНОСТНЫЕ ОБЯЗАННОСТИ ТЕХНИЧЕСКОГО ОТДЕЛА

В ООО «Девкрафт», в техническом отделе, имеются следующие должности: ведущий- и инженер-программист, а также начальник отдела и системный администратор.

Ведущий программист – программист, возглавляющий один или несколько проектов по разработке программного обеспечения, либо программист, имеющий определенный уровень подготовки, трудового стажа и соответствующий требованиям к образованию для получения данного статуса. Несет ответственность за:

1. архитектуру, лежащую в основе разрабатываемой программы;
2. обучение новых или менее опытных разработчиков;
3. распределение работы и отслеживание выполнения задач другими участниками команды (редко, в основном этим занимается менеджер проекта).

Начальник отдела разработки отвечает за выполнение и результат следующих функций:

1. распределение работ между подчиненными и контроль выполнения подчиненными полученной работы;
2. оценивание объема поручаемой работы с целью правильного планирования и выполнения работы в срок;
3. уточнение постановленных перед сотрудниками отдела задач в рамках их привлечения в проектные группы, а также постановку задач вне проектных групп;
4. экспертиза артефактов, разработанных сотрудниками отдела;
5. поддержание в актуальном состоянии стандартов отдела, контроль соответствия деятельности сотрудников отдела стандартам;
6. предъявление требований к квалификации по вакансиям в отдел, оценка уровня квалификации кандидатов на вакансии в отдел;
7. контроль соблюдения сотрудниками отдела Правил внутреннего трудового распорядка.
8. реализацию алгоритмов;
9. разработку компонентов по подготовленным техническим требованиям;
10. разработку наиболее ответственных элементов системы;
11. формулирование и распределение заданий для стажера и младшего программиста;
12. контроль исполнения заданий и качества кода стажера и младшего программиста;
13. участие в обсуждении архитектуры и требований к системе;
14. выбор технологии, языка программирования, библиотек и т.п.;
15. отладка кода.

Системный администратор – сотрудник, должностные обязанности которого подразумевают обеспечение штатной работы парка компьютерной техники, сети и программного обеспечения.

В должностные обязанности инженера-программиста входят:

1. составление вычислительной схемы метода решения задач, перевод алгоритмов решения на формализованный машинный язык.
2. определение вводимой в машину информацию, ее объем, методы контроля производимых машиной операций, форму и содержание исходных документов и результатов вычислений.
3. разработка макетов и схем ввода, обработки, хранения и выдачи информации.
4. определение совокупности данных, обеспечивающих решение максимального числа включенных в данную программу условий.
5. проведение отладки разработанных программ, определение возможности использования готовых программ, разработанных другими организациями.

В течении прохождения технологической практики возлагались должностные обязанности инженера-программиста. В том числе были необходимы знания системного администратора.

# ПЕРЕЧЕНЬ СТАДИЙ, ЭТАПОВ И СРОКОВ РАЗРАБОТКИ ПРОГРАММЫ И САЙТА

Основные этапы разработки программного обеспечения:

1. разработка алгоритма;
2. кодирование;
3. отладка;
4. тестирование;
5. создание справочной системы.

На первом этапе, происходит подробное описание исходных данных, осуществляется формулировка требований к получаемому результату, рассматриваются всевозможные поведения программы при возникновении особых случаев, происходит разработка диалоговых окон, которые обеспечат взаимодействие пользователя и самой программы.

На втором этапе программист определяет последовательность необходимых действий, которые впоследствии нужно выполнить для получения желаемого результата.

На третьим этапе, после проведения спецификации и составления алгоритма решения, используемый алгоритм в итоге будет записан на необходимом языке программирования (Python, C#, Kotlin, R или Java). Результатом этапа кодирования является готовая программа.

На этапе «Отладка», программист занимается отладкой программы, то есть поиском и устранением ошибок. Последние делятся на две группы: алгоритмические и синтаксические (ошибки в тексте исходной программы). Из этих двух групп ошибок наиболее легко устранить синтаксические ошибки, тогда как алгоритмические ошибки определить достаточно трудно.

На последнем этапе, выполняется тестирование программы. Очень важный этап, поскольку в большинстве случаев программисты создают программы не для личного применения, а чтоб их программой пользовались другие. На этапе тестирования разработчик проверяет поведение программы при большом числе наборов входных данных.

Основные этапы создания программного обеспечения:

1. определение целей программного средства;
2. создание технического задания;
3. кодирование программного средства;
4. тестирование.

На первом этапе необходимо определить решение задач программного средства.

Следующим этапом, разрабатывается техническое задание. В техническом задании необходимо как можно более подробно описать:

1. цели создания программного обеспечения и его целевую аудиторию;
2. архитектуру приложения;
3. используемые технологии;
4. порядок предоставления, обработки или создания графической и текстовой информации;
5. технические требования.

Техническое задание является основным документом, на основе которого осуществляются все последующие этапы разработки.

Следующим этапом выполняется кодирование на основании пунктов, заданных в техническом задании, после чего осуществляется отладка.

На последнем этапе, происходит тестирование программного обеспечения. На данном этапе выявляются все ошибки и недочеты в программировании и написании текстов. Срок тестирования зависит от сложности проекта, но автоматизированные тесты позволяют ускорить текущий процесс.

# ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА РАЗРАБОТКУ ПРОГРАММНОГО ДОКУМЕНТА

## 5.1 Введение

Техническое задание составлено для разработки программы, составляющей собственную книжную библиотеку с помощью метода распознавания образов.

Анализ проблем управления большими и сложными системами осуществляется, как правило, с позиции системного подхода. Системный подход позволяет упорядочить исходную информацию о сложной системе: понизить уровень сложности, осуществить решение задач проектирования и управления сложными по отношению к интеллектуальным возможностям человека объектам.

Требования к разработке:

1. пользовательский интерфейс должен быть прост, удобен и доступен даже неподготовленному пользователю;
2. необходимо обеспечивать выполнение всех эргономических требований (комфортность, цветовую и звуковую гамму, соответствующие наилучшему восприятию, удобство расположения информации и доступность всех необходимых для работы средств, единый стиль выполнения операций и т.д.);
3. пользователь должен выполнять все действия, не выходя из системы, поэтому требуется оснащенность всеми необходимыми операциями внутри одного приложения.

Цель работы заключается в том, чтобы правильно составить подход решения поставленной задачи, разработать и отладить программу, реализующую требуемый функционал, исходя из данных технического задания технологической практики.

## 5.2 Основание для разработки

Основанием разработки «Программы для составления собственной книжной библиотеки с помощью метода распознавания образов», является индивидуальное задание по технологической практике.

## 5.3 Назначение разработки

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.

Первоначально мобильные приложения использовались для быстрой проверки электронной почты, но их высокий спрос привел к расширению их назначений и в других областях, таких как игры для мобильных телефонов и GPS, общение, просмотр видео и пользование интернетом.

Программа нацелена на широкий круг пользователей и позволяет уместить информацию о всех книгах в одном программном средстве. Таким образом возможно вести учет собственной книжной библиотеки вне зависимости от нахождения их физических оболочек.

Цель разработки – быстро и без особых усилий внести новый книжный экземпляр в личный каталог, использую камеру приложения. Также просматривать уже добавленную литературу и иметь возможность удалять из библиотеки.

# ТЕХНИКО-РАБОЧИЙ ПРОЕКТ

## 6.1 Постановка задачи

Создание любой программы начинается с постановки задачи. Изначально задача формулируется в терминах предметной области, и необходимо перевести ее на язык понятий, более близких к программированию. Поскольку программист редко досконально разбирается в предметной области, а заказчик – в программировании, постановка задачи может стать весьма непростым итерационным процессом.

По этой причине была разработана диаграмма IDEF0, представленная на изображении 4, на которой отображена функция «внести книгу». К ней относятся механизмы «пользователь» и «база данных», управление «существование книги во всемирной библиотеки», входная информация – «фотография» и выходная информация – «добавленная книга».

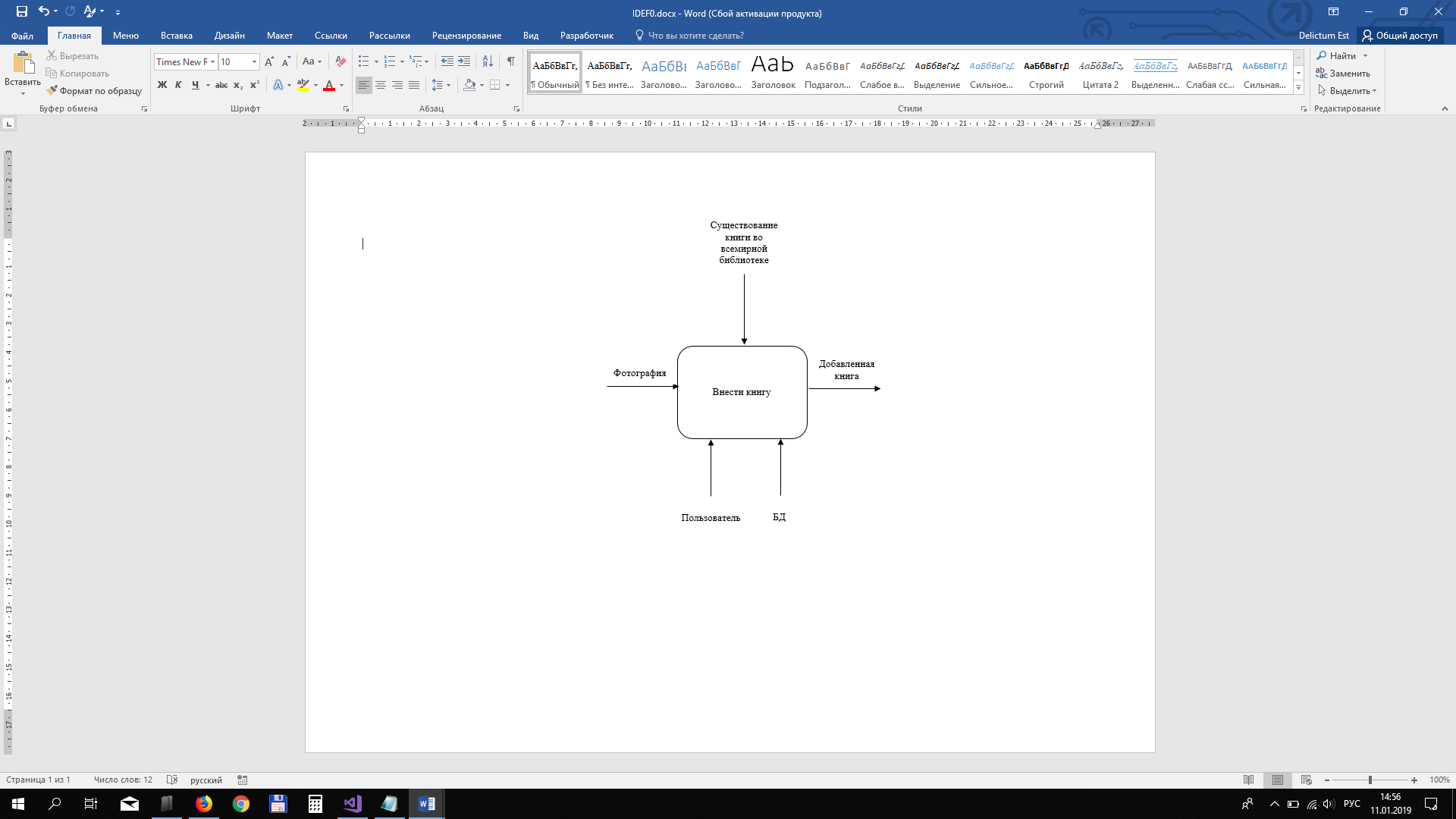


Рисунок 4 – Диаграмма IDEF0

Также была составлена декомпозиция данной методологии, отображенной на рисунке 5.

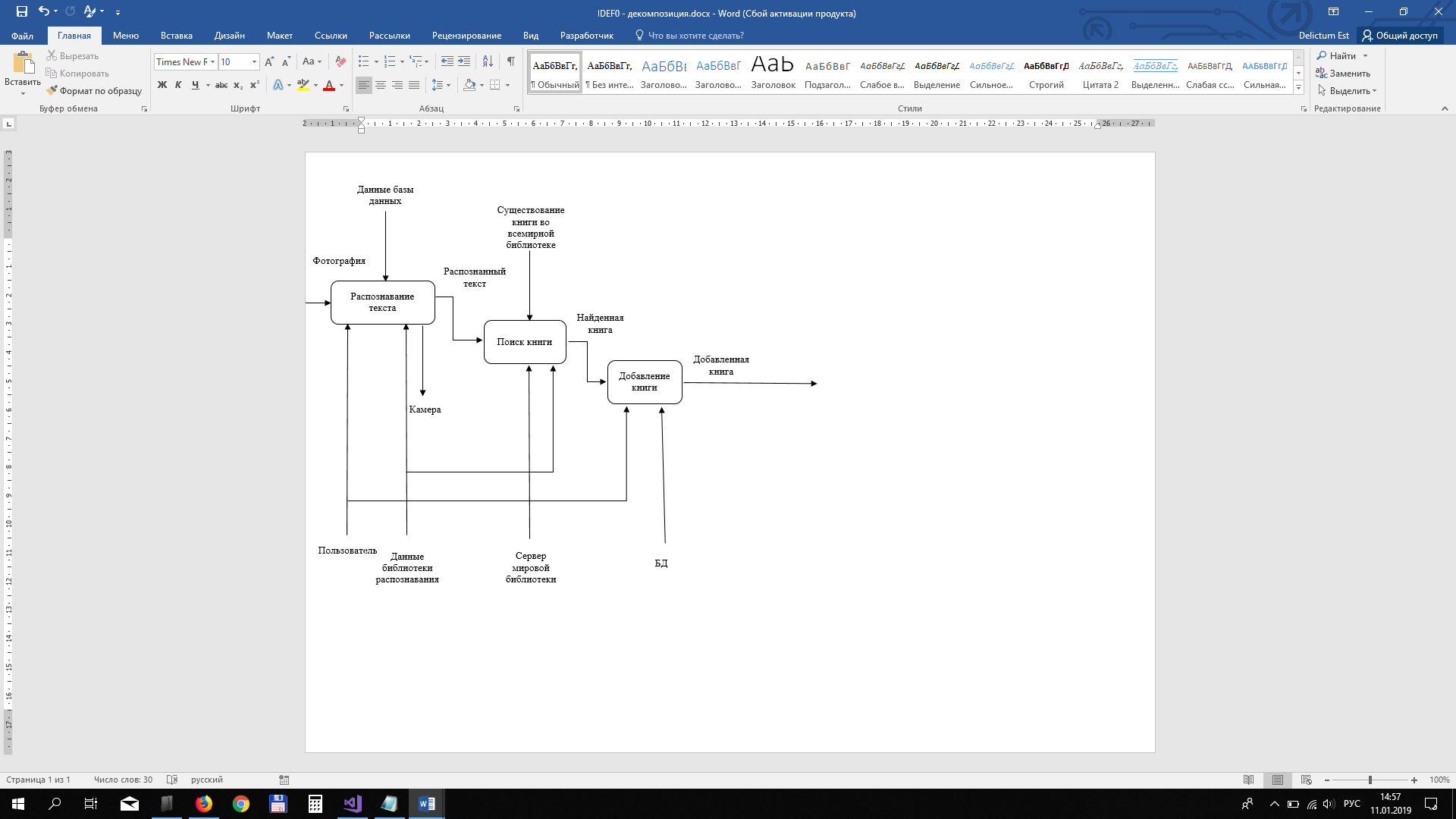


Рисунок 5 – Декомпозиция IDEF0

Диаграмма претендентов описывает взаимоотношения и зависимости между группами вариантов использования и действующих лиц, участвующими в процессе. Данная диаграмма представлена на рисунке 6.

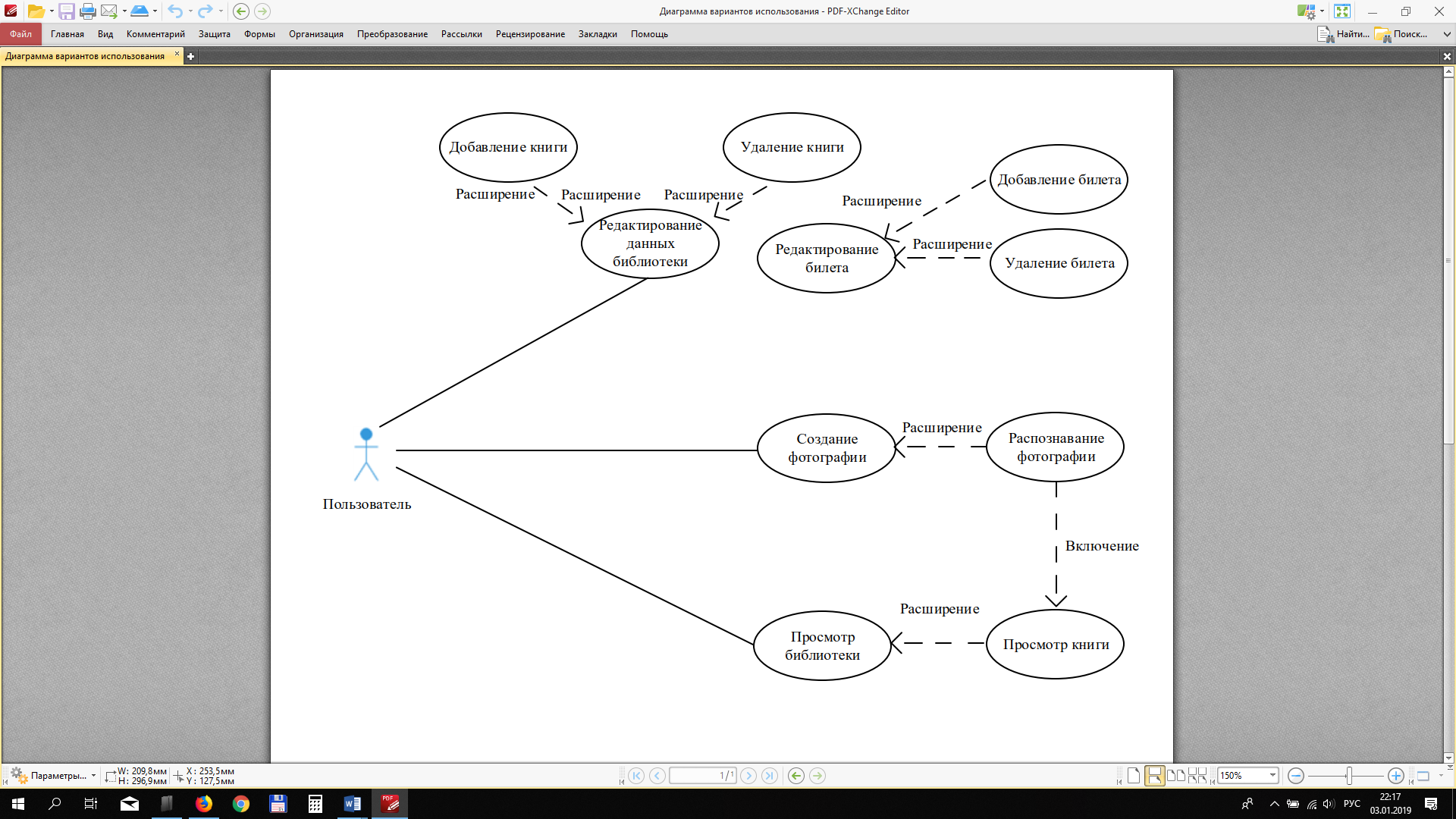


Рисунок 6 – Диаграмма вариантов использования

При разработке программы для составления собственной книжной библиотеки с помощью метода распознавания образов использовать следующие исходные данные:

1. уникальный книжный номер;
2. всемирный каталог книг.

Также необходимо иметь устройство ввода, а именно камеру, и обладать подключением к Интернету.

Исходя из этого была выстроена диаграмма последовательности действий, отображенная на рисунке 7.

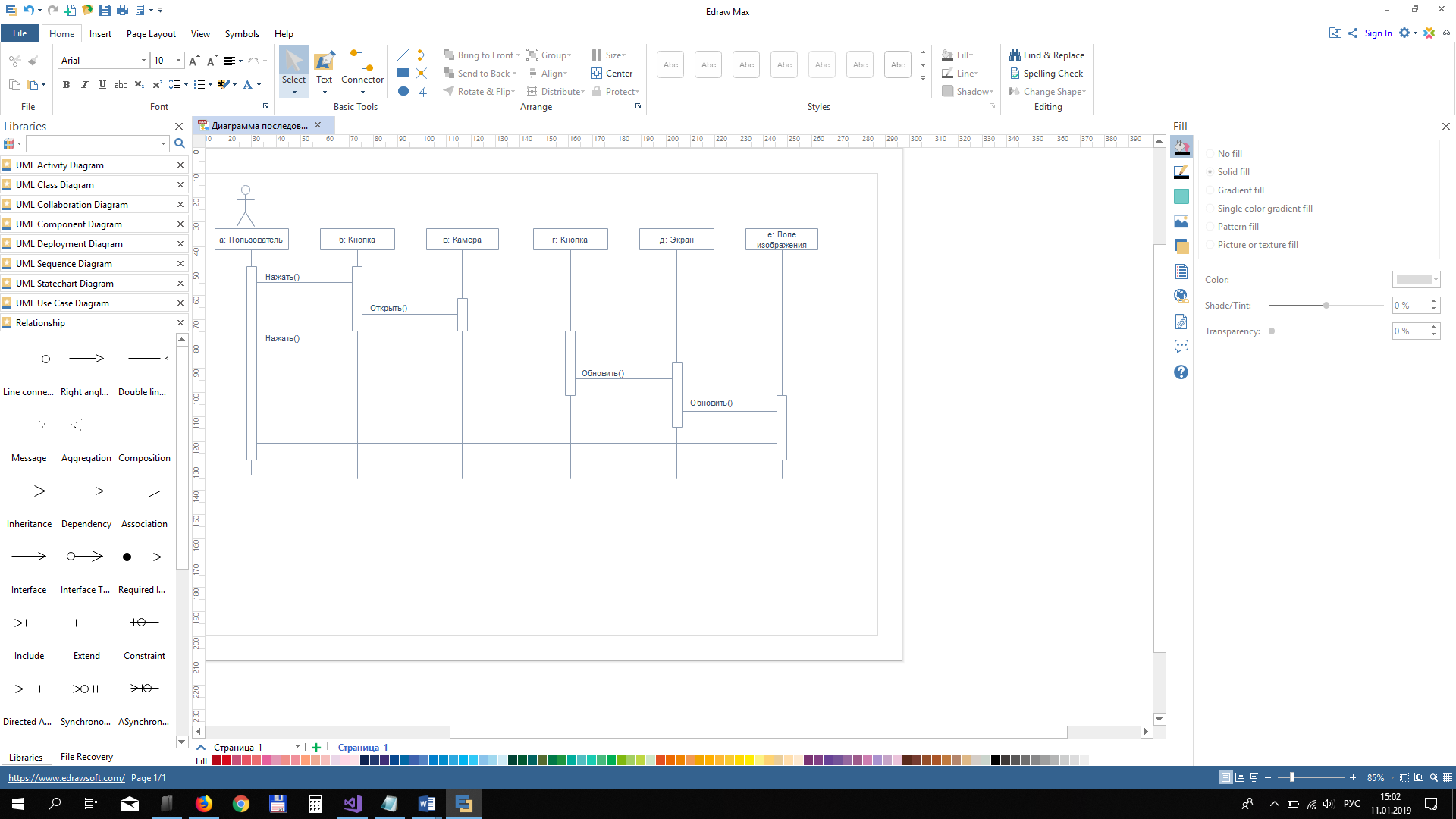


Рисунок 7 – Диаграмма последовательности действий

На основе этих данных выполнять анализ и выдавать информацию о книжном экземпляре.

## 6.2 Выбор среды разработки программного продукта

На этом этапе определяется среда, в которой будет выполняться программа: требования к аппаратуре, используемая операционная система и другое программное обеспечение.

Придерживаясь индивидуального задания – «программа для составления собственной книжной библиотеки с помощью метода распознавания образов» – реализация будет осуществлена для Android, на основании самой популярной операционной системы для смартфонов. Используемый язык программирования – Java, так как он является императивным языком высокого уровня с большим сообществом, доступным в бесплатном доступе.

Для хранения данных будет использоваться реляционная система управления базами данных «SQLite». Для реализации задуманного будет использована интеграционная система разработки – Android Studio.

## 6.3 Определение структуры данных

База данных – представленная в объективной форме совокупность самостоятельных материалов (статей, расчётов, нормативных актов, судебных решений и иных подобных материалов), систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины.

СУБД (Система управления базами данных) – это совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями. Система управления базами данных (СУБД) является посредником между базой данных и ее пользователями.

SQL – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

Встраиваемая СУБД – библиотека, которая позволяет унифицированным образом хранить большие объёмы данных на локальной машине. Доступ к данным может происходить через геоинформационные системы.

SQLite это программная библиотека, которая реализует автономный, безсерверный, нулевой конфигурации транзакционный механизм СУБД SQL. SQLite наиболее широко распространенный механизм СУБД SQL в мире. Исходный код находится в открытом доступе.

Для правильной разработки базы данных вначале требуется изучить предметную область. База данных должна быть сформирована таким образом, чтобы хранить в себе требуемую информацию и не быть нагруженной.

Таблицы предметной области:

* 1. BookName;
  2. Book;
  3. Author.

Используемые типы полей:

1. primaryKey (первичный ключ);
2. foreignKey (внешние ключи);
3. string;
4. integer.

Сущность – это реальный или представляемый тип объекта, информация о котором должна сохраняться и быть доступна.

Сущность базы данных «Book» является ключевой таблицей, которая связывает в себе «Author» и «BookName». У каждой сущности необходимо наличие атрибута «Id», являющегося первичным ключом. Также у «BookName» это «Name» и у «Author» такие атрибуты как «FirstName», «LastName».

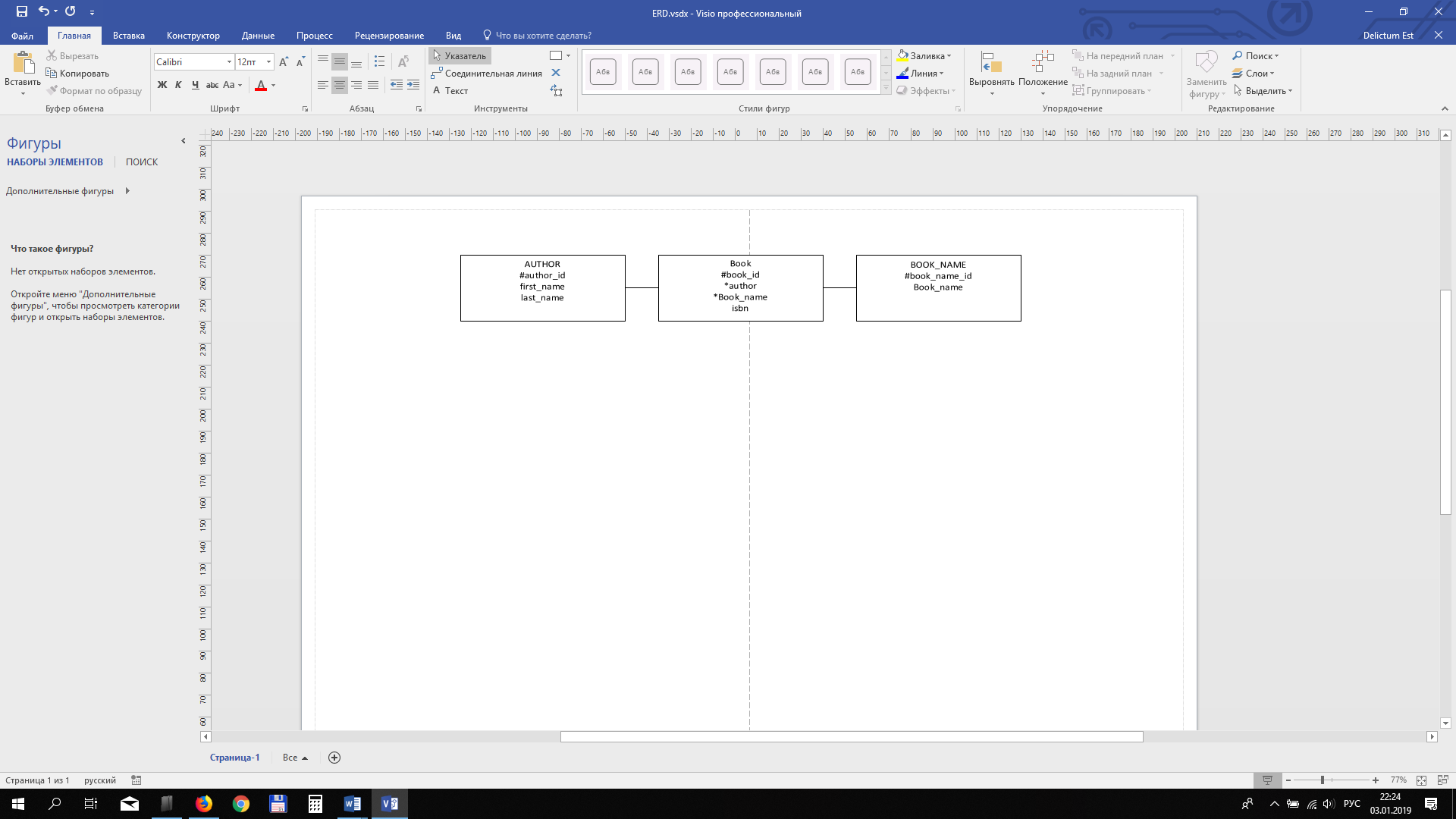
Организация данных исходила из построенной диаграммы «сущность-связь», которая представлена на рисунке 4. 

Рисунок 4 – Схема подключения компьютеров по локальной сети

Реляционная модель ориентирована на организацию данных в виде двумерных таблиц. Каждая реляционная таблица представляет собой двумерный массив и обладает следующими свойствами:

1. каждый элемент таблицы – один элемент данных;
2. все столбцы в таблице однородные, т.е. все элементы в столбце имеют одинаковый тип (числовой, символьный и т.д.) и длину;
3. каждый столбец имеет уникальное имя (заголовки столбцов являются названиями полей в записях);
4. одинаковые строки в таблице отсутствуют;
5. порядок следования строк и столбцов может быть произвольным.

Таким образом, с использованием реляционного языка запросов «SQL», сущности были полностью спроектированы и смоделированы.

## 6.4 Разработка схемы работы системы

Схема – графическое представление определения, анализа или метода решения задачи, в котором используются символы для отображения данных, потока, оборудования.

Блок-схема – распространенный тип схем (графических моделей), описывающих алгоритмы или процессы, в которых отдельные шаги изображаются в виде блоков различной формы, соединенных между собой линиями, указывающими направление последовательности.

Схема работы системы – распространенный тип схем, описывающий функции, изображая шаги в виде блоков различной формы, соединенных между собой стрелками.



Рисунок 5 – Схема работы программы

## 6.5 Результат реализации программного средства

Программа для составления собственной книжной библиотеки с помощью метода распознавания образов предназначена для широкого круга пользователей.

На рисунке 6 показан основной экран при запуске приложения.



Рисунок 6 – Стартовый экран

В данном окне при нажатии на кнопку с «плюсом» или при проведении пальцем справа-налево открывается камера устройства. При сделанной фотографии, она помещается в центр главного окна, как видно из рисунка 7.

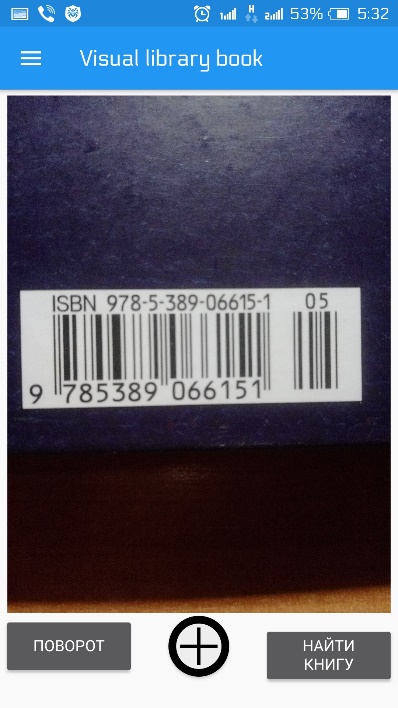


Рисунок 7 – Добавленная фотография

При нажатии на кнопку «Поворот» изображение изменит угол наклона на 90 градусов, как показано на рисунке 8. Угол наклона может изменяться на 360 градусов.

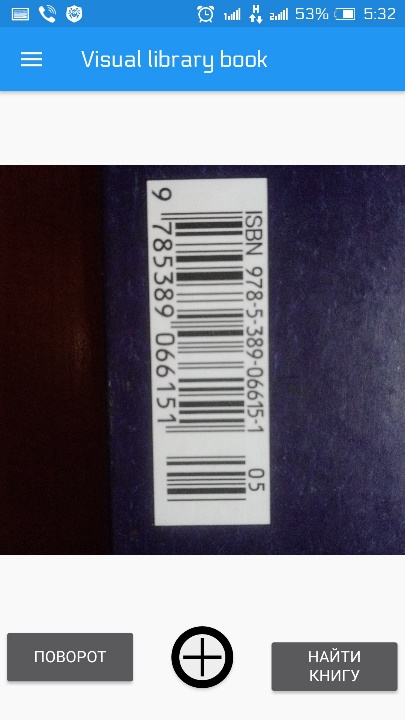


Рисунок 8 – Повернутая на 90 градусов фотография

После нажатия кнопки «Найти книгу» при активном подключении к Интернету и удачном распознавании открывается новый экран с данными книги, что продемонстрированно на рисунке 9.

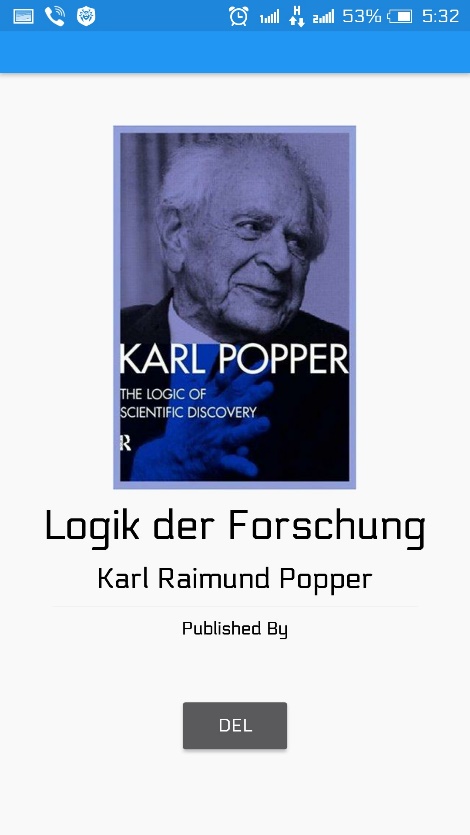


Рисунок 9 – Представление найденной книги

По нажатия кнопки «Add» / «Del» происходит добавление экземпляра в личный каталог, который отображен на рисунке 10.

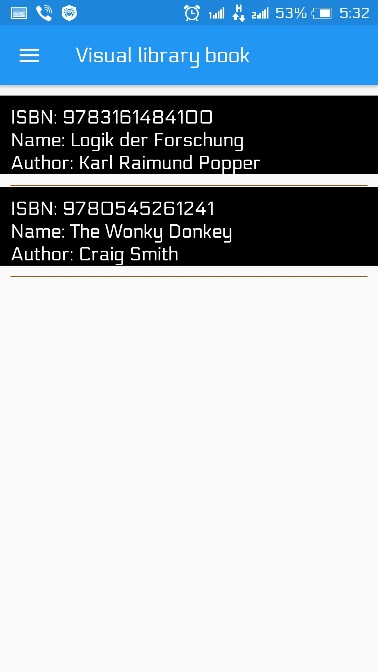


Рисунок 10 – Пользовательский каталог

Раздел настройки представлен на рисунке 11. Здесь можно выбрать язык: русский или английский, изменить начальный экран при запуске приложения: камера или личный каталог, просмотреть информацию о версии приложения и сведения о разработчике.

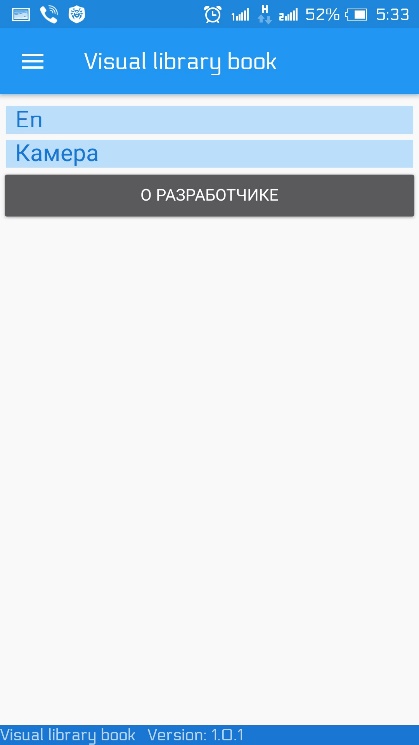


Рисунок 11 – Настройки приложения

Представление «Помощь» состоит из трех вкладок, которые изменяются при проведении пальцем по экрану в соответствующих направлениях, каждая из которых содержит вспомогательную информацию по использованию программного обеспечения. Данный раздел отображен на рисунке 12.



Рисунок 12 – Раздел «Помощь»

Также в приложении встроено меню, позволяющее перемещаться по остальным разделам, появляющееся при проведении пальцем слева-направо. Навигационное меню представлено на изображении 13.

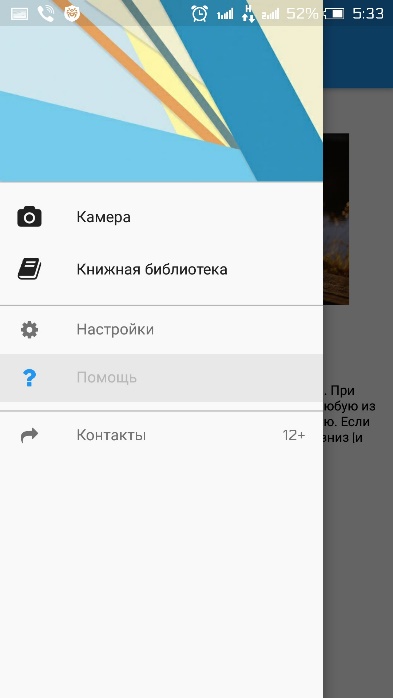


Рисунок 13 – Навигационное меню

В ходе данной демонстрации были представлены результаты реализации программного средства.

# ТЕСТИРОВАНИЕ

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

Жизненный цикл тестирования можно представить схемой, отображенной на рисунке 14.

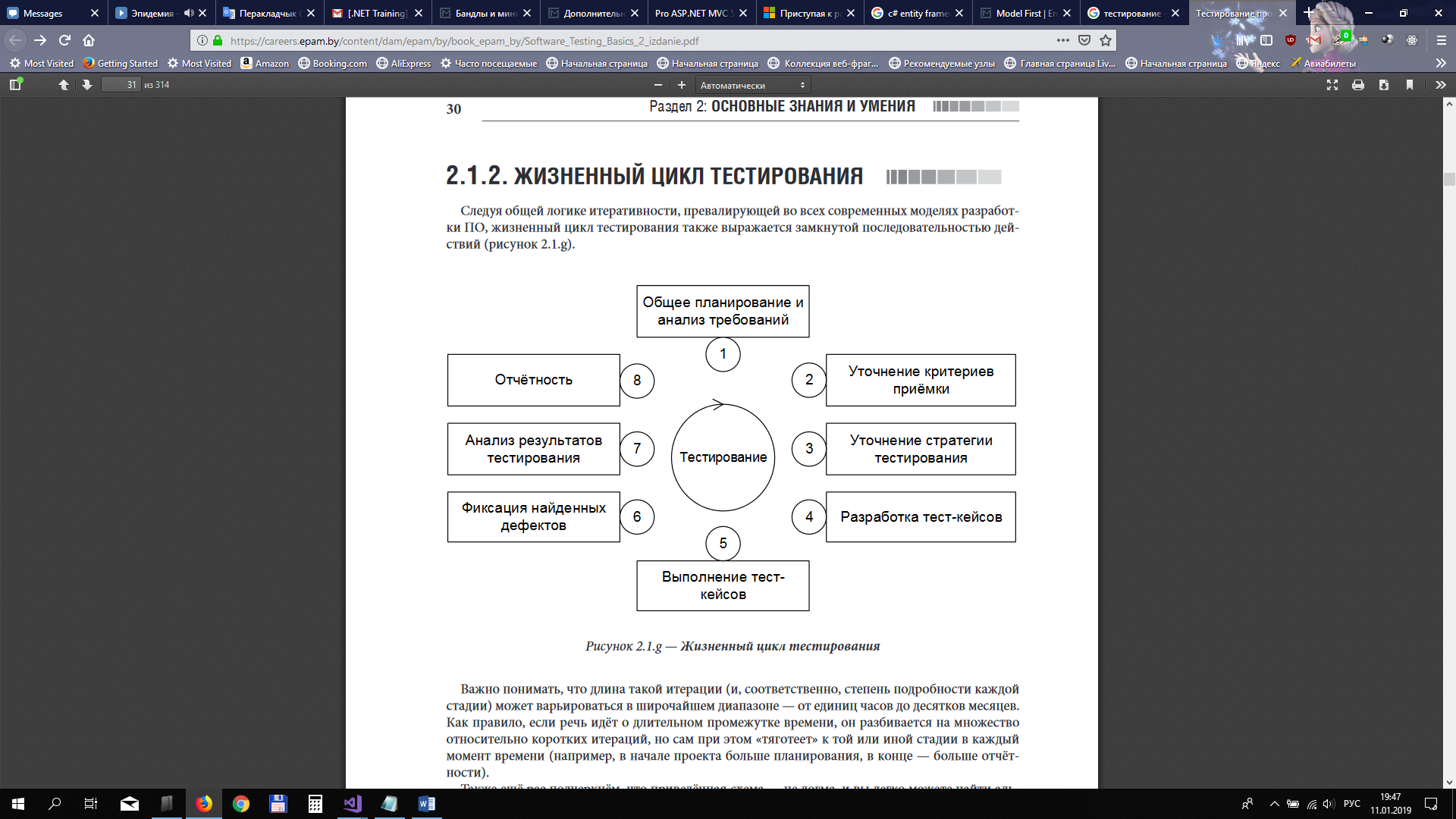


Рисунок 14 – Жизненный цикл тестирования

Для проверки и оценки программы, выбрали следующий тестовый вариант: стохастическое тестирование в рамках метода белого ящика.

Стохастическое тестирование – тестирование программ, при котором исходные тестовые данные берутся случайным образом (с использованием статистического распределения).

Метод белого ящика – у тестировщика есть доступ к внутренней структуре и коду приложения, а также есть достаточно знаний для понимания увиденного.

Существующие на сегодняшний день методы тестирования ПО не позволяют однозначно и полностью выявить все дефекты и установить корректность функционирования анализируемой программы, поэтому все существующие методы тестирования действуют в рамках формального процесса проверки исследуемого или разрабатываемого ПО.

Такой процесс формальной проверки или верификации может доказать, что дефекты отсутствуют с точки зрения используемого метода.

Сортировка базы данных – это упорядочение записей по значениям одного из полей.

Сортировка записей производится по какому-либо полю базы данных. Значения, содержащиеся в этом поле, располагаются в порядке возрастания или убывания. В процессе сортировки целостность записей сохраняется, т. е. строки таблицы перемещаются целиком.

Проведенное тестирование можно представить в виде диаграммы вариантов использования, представленной изображением 15.

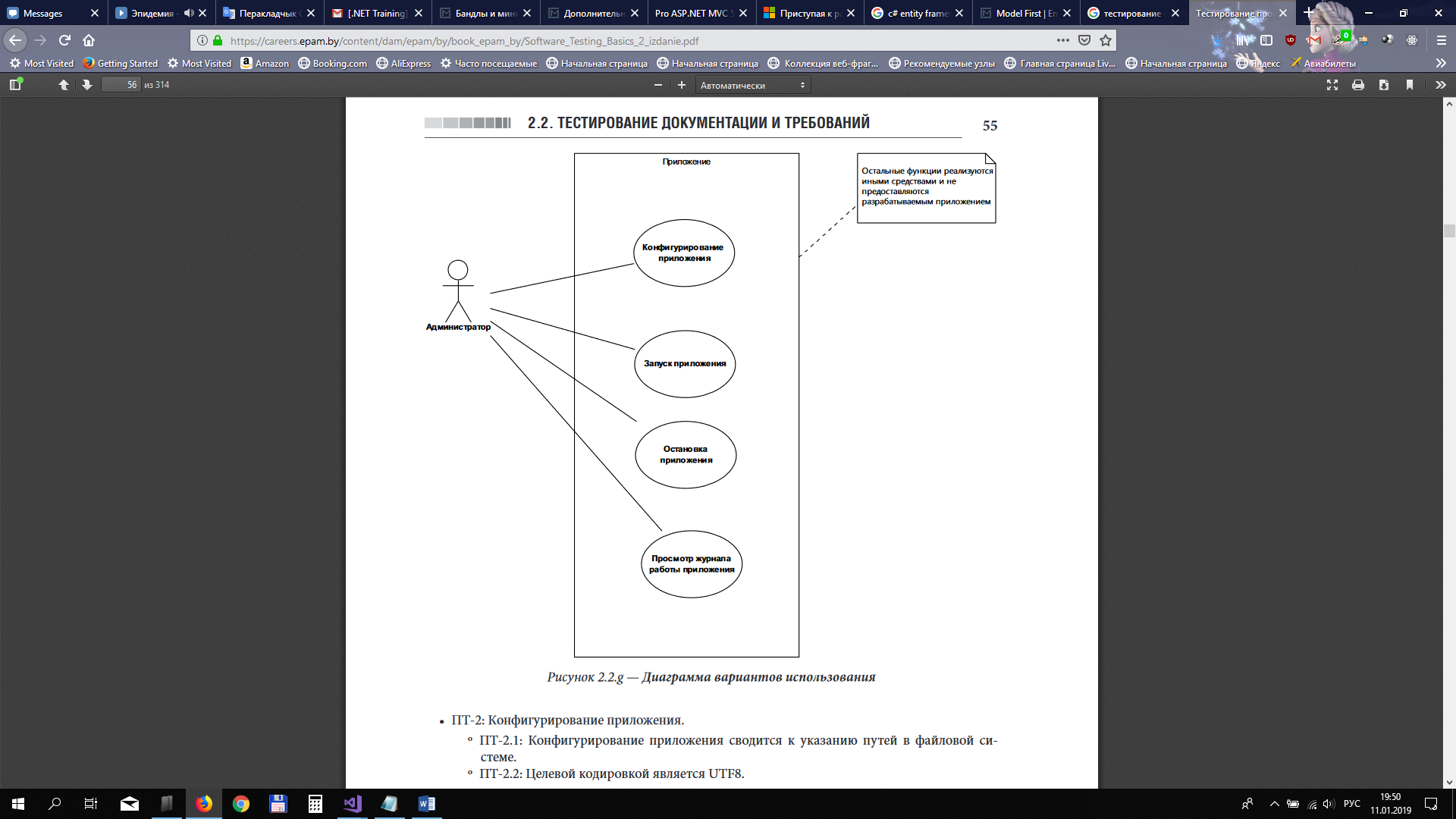


Рисунок 15 – Диаграмма вариантов использования тестирования

Программа для составления собственной книжной библиотеки с помощью метода распознавания образов было протестировано с использованием метода «Стохастическое тестирование» и не выявило активных ошибок. Выборка содержимого из базы данных происходит на высокой скорости. При добавлении записей, стоят многочисленные проверки наполненности полей, а также, маски ввода данных.

В результате тестирования, ошибок выявить не удалось. Программа готова к использованию.

# ВЫВОД

Задачей данного проекта по технологической практике являлась разработка android-приложения для составления собственной книжной библиотеки с помощью метода распознавания образов.

В ходе выполнения задания практики, было разработано программное средство с использованием языка программирования «Java», а также, с помощью языка структурированных запросов «SQL». Изучены возможности данного языка.

На проектирование методологии IDEF0 была создана функциональная модель в соответствии с данным программным продуктом, на котором отображены необходимые данные. В проектировании информационной базы проекта была смоделирована схема «Сущность-связь», на которой указаны требуемые таблицы и атрибуты для дальнейшей успешной реализации программы.

При проектирования следующих диаграмм: вариантов использования, последовательности действий, использовали язык моделирования UML и программное средство «EDRAW MAX», а также «Microsoft Office Word» и «Microsoft Office Visio». Результатом проектирования были созданы указанные диаграммы.

В разделе «Технико-рабочий проект» была реализована база данных, а также осуществлены все функции программы. Результатом данного раздела стал готовый программный продукт.

После вся программа неоднократно тестировалась на правильность получаемых результатов, все возникающие ошибки были устранены в ходе работы. Тестирование происходило с помощью метода «Чёрного ящика». После проведенного тестирования можно установить, что программа удовлетворяет всем поставленным перед ней требованиям и является готовым программным средством.

Основное внимание уделено изучению способов проектирования приложений, объектно-ориентированному и системному программированию.

Разработанное обеспечение быстро и безошибочно справляется с поставленной задачей хранения и обработки информации. В удобном интерфейсе воплощены все необходимые для данной работы возможности.

Основные модификации по дальнейшему улучшению заключаются в:

1. реинжиниринге;
2. интернационализации и локализации;
3. портировании и миграции программного обеспечения.

Достоинством проекта является простота и интуитивность программного средства.

Исходя из вышесказанного можно утверждать о выполненной цели технологической практики. Данный проект реализовал поставленные задачи в соответствии с заданным условием.

# ЛИТЕРАТУРА

1. Голощапов, A. Google Android программирование для мобильных устройств/ Голощапов А.Л. – Санкт-Петербург, 2011

2. Хашими, С. Разработка приложений для Android/ Хашими С., Коматинени С., Маклин Д. – Эксмо, 2011

3. Дейтел, П. Android для программистов/ Х. Дейтел, Э. Дейтел, М. Моргано – Питер, 2013

4. Майер, Р. Android 2. Программирование приложений для планшетных компьютеров и смартфонов/ Майер Р. – Эксмо, 2013

5. Бурнет, Э.Привет, Android! Разработка мобильных приложений/ Э. Бурнет – СПб Питер, 2012

6. Герберт, Ш. Java 8. Полное руководство; 9-е изд/ Ш. Герберт –Вильяме, 2015

7. Дэрси, Л. Android за 24 часа. Программирование приложений под операционную систему Google/ Л. Дэрси, К. Шейн – Рид Групп, 2012

8. Колиснеченко, Д. Программирование для Android. Самоучитель / Д. Колиснеченко – Санкт-Петербург, 2011

9. Блэйк, М. Программирование под Android / М. Блэйк – Санкт-Петербург, 2012

10. Жвалевский, А. Смартфоны Android без напряга. Руководство пользователя / А. Жвалевский – Санкт-Петербург, 2012

11. Бейзер, Б. Тестирование черного ящика: технологии функционального тестирования программного обеспечения и систем / Б. Бейзер – Питер, 2004

12. Хайкин, С. Нейронные сети. Полный курс / С. Хайкин – Вильямс, 2018

13. Каллан, Р. Основные концепции нейронных сетей / Р. Каллан – Вильямс, 2003

14. Куликов, С. Тестирование программного обеспечения. Базовый курс / С. Куликов – Четыре четверти, 2017

# ПРИЛОЖЕНИЕ А

**(обязательное)**

**Текст программы**

Листинг MainActivity.java:

package com.visualcheckbook.visualcheckbook.Activity;

import android.app.Activity;

import android.content.DialogInterface;

import android.graphics.drawable.BitmapDrawable;

import android.net.Uri;

import android.support.v4.app.Fragment;

import android.support.v7.app.AlertDialog;

import android.support.v7.app.AppCompatActivity;

import android.content.ActivityNotFoundException;

import android.content.Intent;

import android.graphics.Bitmap;

import android.os.Bundle;

import android.os.Environment;

import android.support.annotation.NonNull;

import android.support.v7.widget.Toolbar;

import android.util.Log;

import android.view.MotionEvent;

import android.view.View;

import android.widget.AdapterView;

import android.widget.Button;

import android.widget.ImageView;

import android.provider.MediaStore;

import android.widget.LinearLayout;

import com.google.android.gms.tasks.OnFailureListener;

import com.google.android.gms.tasks.OnSuccessListener;

import com.loopj.android.http.JsonHttpResponseHandler;

import com.mikepenz.iconics.typeface.FontAwesome;

import com.mikepenz.materialdrawer.model.PrimaryDrawerItem;

import com.visualcheckbook.visualcheckbook.BookAPI.Book;

import com.visualcheckbook.visualcheckbook.BookAPI.BookAdapter;

import com.visualcheckbook.visualcheckbook.BookAPI.BookClient;

import com.google.firebase.ml.vision.common.FirebaseVisionImage;

import com.google.firebase.ml.vision.text.FirebaseVisionText;

import com.google.firebase.ml.vision.text.FirebaseVisionTextRecognizer;

import com.google.firebase.ml.vision.FirebaseVision;

import com.mikepenz.materialdrawer.Drawer;

import com.mikepenz.materialdrawer.model.SecondaryDrawerItem;

import com.mikepenz.materialdrawer.model.interfaces.Badgeable;

import com.mikepenz.materialdrawer.model.interfaces.IDrawerItem;

import com.mikepenz.materialdrawer.model.interfaces.Nameable;

import com.visualcheckbook.visualcheckbook.BuildConfig;

import com.visualcheckbook.visualcheckbook.Fragments.BookLibraryFragment;

import com.visualcheckbook.visualcheckbook.Fragments.HelperTabFragment;

import com.visualcheckbook.visualcheckbook.Fragments.SettingsFragment;

import com.visualcheckbook.visualcheckbook.Helpers.ActivityHelper;

import com.visualcheckbook.visualcheckbook.Helpers.CustomSettingsHelper;

import com.visualcheckbook.visualcheckbook.Helpers.ImageHelper;

import com.visualcheckbook.visualcheckbook.IsbnParser;

import com.visualcheckbook.visualcheckbook.LockOrientation;

import com.visualcheckbook.visualcheckbook.OnSwipeTouchListener;

import com.visualcheckbook.visualcheckbook.R;

import org.json.JSONArray;

import org.json.JSONException;

import org.json.JSONObject;

import java.io.File;

import java.util.ArrayList;

import java.util.List;

import cz.msebera.android.httpclient.Header;

public class MainActivity extends AppCompatActivity {

private LinearLayout mainLinearLayout;

private ImageView mCameraImageView;

private Button mTextRecognitionButton;

private Button mCameraButton;

private Button mRotationButton;

private Bitmap mSelectedImage;

private AlertDialog.Builder mExitDialog;

public static Toolbar mToolbar;

public static Drawer.Result drawerResult = null;

private Uri outputFileUri;

private BookClient client;

private BookAdapter bookAdapter;

private Integer angleRotate = 0;

private final Integer ROTATE\_IMAGE = 90;

private Integer currentPositionDrawerMenu;

private Fragment currentFragment;

private String pictureImagePath = "";

private final String fileSaveImageName = "temp.jpg";

public static final String BOOK\_DETAIL\_KEY = "book";

private static final Integer REQUEST\_IMAGE\_CAPTURE = 1;

public static final String TAG = "VisualCheckBook";

public static final String VERSION = "1.0.2";

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

initCustomModel();

}

private void initListBooks() {

ArrayList<Book> aBooks = new ArrayList<Book>();

bookAdapter = new BookAdapter(this, aBooks);

}

private void queryBooks(final String searchString) {

client = new BookClient();

client.getBooks(searchString, new JsonHttpResponseHandler() {

@Override

public void onSuccess(int statusCode, Header[] headers, JSONObject response) {

try {

JSONArray docs = null;

if(response != null) {

docs = response.getJSONArray("docs");

// Parse json array into array of model objects

final ArrayList<Book> books = Book.fromJson(docs);

if (books == null) {

ActivityHelper.showToast(getString(R.string.does\_not\_contain\_book), getApplicationContext());

return;

}

if (bookAdapter != null)

bookAdapter.clear();

for (Book book : books) {

bookAdapter.add(book); // add book through the adapter

}

if (bookAdapter != null)

bookAdapter.notifyDataSetChanged();

BookDetailActivity.Isbn = searchString;

Intent intent = new Intent(MainActivity.this, BookDetailActivity.class);

intent.putExtra(BOOK\_DETAIL\_KEY, bookAdapter.getItem(0));

startActivity(intent);

}

} catch (JSONException e) {

if (BuildConfig.DEBUG) {

Log.e(TAG, "Invalid JSON format", e);

}

}

}

@Override

public void onFailure(int statusCode, Header[] headers, String responseString, Throwable throwable) {

ActivityHelper.showToast(getString(R.string.problem\_server\_connection), getApplicationContext());

}

});

}

private void runTextRecognition() {

FirebaseVisionImage image = FirebaseVisionImage.fromBitmap(mSelectedImage);

FirebaseVisionTextRecognizer recognizer = FirebaseVision.getInstance()

.getOnDeviceTextRecognizer();

mTextRecognitionButton.setEnabled(false);

recognizer.processImage(image)

.addOnSuccessListener(

new OnSuccessListener<FirebaseVisionText>() {

@Override

public void onSuccess(FirebaseVisionText texts) {

mTextRecognitionButton.setEnabled(true);

processTextRecognitionResult(texts);

}

})

.addOnFailureListener(

new OnFailureListener() {

@Override

public void onFailure(@NonNull Exception e) {

// Task failed with an exception

mTextRecognitionButton.setEnabled(true);

e.printStackTrace();

}

});

}

private void processTextRecognitionResult(FirebaseVisionText texts) {

List<FirebaseVisionText.TextBlock> blocks = texts.getTextBlocks();

if (blocks.size() == 0) {

ActivityHelper.showToast(getString(R.string.text\_not\_found), getApplicationContext());

return;

}

String allText = "";

for (int i = 0; i < blocks.size(); i++) {

List<FirebaseVisionText.Line> lines = blocks.get(i).getLines();

for (int j = 0; j < lines.size(); j++) {

List<FirebaseVisionText.Element> elements = lines.get(j).getElements();

for (int k = 0; k < elements.size(); k++) {

allText += elements.get(k).getText() + ";";

}

}

}

String ISBN = IsbnParser.ParserISBN(allText);

if (ISBN == null)

ActivityHelper.showToast(getString(R.string.incorrect\_recognition\_isbn), getApplicationContext());

else

queryBooks(ISBN);

}

private void RotateImage() {

angleRotate += ROTATE\_IMAGE;

mCameraImageView.animate().rotation(angleRotate).start();

mSelectedImage = ImageHelper.rotateImage(angleRotate, mSelectedImage);

}

private void initCustomModel() {

setContentView(R.layout.activity\_main);

new LockOrientation(this).lock();

ActivityHelper.initLocaleHelper(this);

initRecognition();

initCamera();

initRotate();

initToolbar();

initDrawerMenu();

initSliding();

initListBooks();

initQuestionExitDialog();

mainLinearLayout = findViewById(R.id.main\_liner\_layout);

currentPositionDrawerMenu = initStartScreen();

}

private int initStartScreen() {

int valueStartScreen = CustomSettingsHelper.getPositionStartScreen(this);

setEnabledDrawerItem(valueStartScreen == 0 ? 1 : valueStartScreen, false);

switch (valueStartScreen) {

case (0): {

break;

}

case (1): {

setVisibilityMainLayout(View.INVISIBLE);

setEnabledDrawerItem(1, true);

setEnabledDrawerItem(2, false);

currentFragment = new BookLibraryFragment();

break;

}

}

if (valueStartScreen != 0) {

getSupportFragmentManager().beginTransaction()

.replace(R.id.container, currentFragment)

.commit();

}

return ++valueStartScreen;

}

private void initRotate() {

mRotationButton = findViewById(R.id.rotate\_button);

mRotationButton.setEnabled(false);

mRotationButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

RotateImage();

}

});

}

private void initSliding() {

mCameraImageView.setOnTouchListener(new OnSwipeTouchListener(MainActivity.this) {

public void onSwipeRight() {

drawerResult.openDrawer();

}

public void onSwipeLeft() {

dispatchTakePictureIntent();

}

public boolean onTouch(View v, MotionEvent event) {

return gestureDetector.onTouchEvent(event);

}

});

}

private void initRecognition() {

mCameraImageView = findViewById(R.id.image\_view);

mTextRecognitionButton = findViewById(R.id.button\_text);

mTextRecognitionButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

runTextRecognition();

}

});

mTextRecognitionButton.setEnabled(false);

}

private void initCamera() {

mCameraButton = findViewById(R.id.camera\_button);

mCameraButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

dispatchTakePictureIntent();

}

});

}

private void initDrawerMenu() {

drawerResult = new Drawer()

.withActivity(this)

.withToolbar(mToolbar)

.withActionBarDrawerToggle(true)

.withHeader(R.layout.drawer\_header)

.addDrawerItems(ActivityHelper.initDrawerItems(0))

.withOnDrawerItemClickListener(new Drawer.OnDrawerItemClickListener() {

@Override

public void onItemClick(AdapterView<?> parent, View view, int position, long id, IDrawerItem drawerItem) {

if (drawerItem instanceof Nameable) {

setEnabledDrawerItem(position, false);

setEnabledDrawerItem(currentPositionDrawerMenu, true);

if (position == 1) {

setVisibilityMainLayout(View.VISIBLE);

getSupportFragmentManager().beginTransaction()

.remove(currentFragment)

.commit();

} else if (position == 2) {

setVisibilityMainLayout(View.INVISIBLE);

currentFragment = new BookLibraryFragment();

} else if (position == 4) {

setVisibilityMainLayout(View.INVISIBLE);

currentFragment = new SettingsFragment();

}

else if (position == 5) {

setVisibilityMainLayout(View.INVISIBLE);

currentFragment = new HelperTabFragment();

}

}

if (drawerItem instanceof Badgeable) {

Badgeable badgeable = (Badgeable) drawerItem;

if (badgeable.getBadge() != null) {

try {

int badge = Integer.valueOf(badgeable.getBadge());

if (badge > 0) {

drawerResult.updateBadge(String.valueOf(badge - 1), position);

}

} catch (Exception e) {

}

}

}

if (position != 0) {

if (position == 4) {

SettingsFragment settingsFragment = (SettingsFragment) currentFragment;

settingsFragment.first = true;

currentFragment = settingsFragment;

}

getSupportFragmentManager().beginTransaction()

.replace(R.id.container, currentFragment)

.commit();

currentPositionDrawerMenu = position;

}

}

})

.withOnDrawerItemLongClickListener(new Drawer.OnDrawerItemLongClickListener() {

@Override

public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id, IDrawerItem drawerItem) {

if (drawerItem instanceof SecondaryDrawerItem) {

}

return false;

}

})

.build();

}

private void initToolbar() {

mToolbar = (Toolbar) findViewById(R.id.toolbar);

setSupportActionBar(mToolbar);

getSupportActionBar().setDisplayHomeAsUpEnabled(true);

}

private void initQuestionExitDialog() {

mExitDialog = new AlertDialog.Builder(this, R.style.AlertDialogCustom)

.setIcon(R.drawable.ic\_book)

.setMessage(R.string.message\_question\_exit\_dialog);

mExitDialog.setPositiveButton(R.string.yes, new DialogInterface.OnClickListener() {

public void onClick (DialogInterface dialog,int arg1){

finish();

}

});

mExitDialog.setNegativeButton(R.string.no, null);

}

private void setVisibilityMainLayout(int condition) {

mainLinearLayout.setVisibility(condition);

mCameraImageView.setVisibility(condition);

}

private void setEnabledDrawerItem(int position, boolean condition) {

switch (position) {

case 1: {

drawerResult.updateItem(new PrimaryDrawerItem()

.withName(R.string.drawer\_item\_home)

.withIcon(FontAwesome.Icon.faw\_camera)

.setEnabled(condition), position);

break;

}

case 2: {

drawerResult.updateItem(new PrimaryDrawerItem()

.withName(R.string.drawer\_item\_library\_book)

.withIcon(FontAwesome.Icon.faw\_book)

.setEnabled(condition), position);

break;

}

case 4: {

drawerResult.updateItem(new SecondaryDrawerItem()

.withName(R.string.drawer\_item\_settings)

.withIcon(FontAwesome.Icon.faw\_cog)

.setEnabled(condition), position);

break;

}

case 5: {

drawerResult.updateItem(new SecondaryDrawerItem()

.withName(R.string.drawer\_item\_help)

.withIcon(FontAwesome.Icon.faw\_question)

.setEnabled(condition), position);

break;

}

}

}

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

try {

if (requestCode == REQUEST\_IMAGE\_CAPTURE && resultCode == RESULT\_OK) {

File imgFile = new File(pictureImagePath);

if (imgFile.exists()) {

mCameraImageView.setImageResource(0);

mCameraImageView.setImageURI(outputFileUri);

mSelectedImage = ((BitmapDrawable) mCameraImageView.getDrawable()).getBitmap();

if (!mTextRecognitionButton.isEnabled()) {

mTextRecognitionButton.setEnabled(true);

mRotationButton.setEnabled(true);

}

}

} else if (resultCode == Activity.RESULT\_CANCELED) {

ActivityHelper.showToast(getString(R.string.cancel\_operation), getApplicationContext());

} else {

throw new Exception();

}

} catch (Exception e) {

ActivityHelper.showToast(getString(R.string.error\_receiving\_photos), getApplicationContext());

if (BuildConfig.DEBUG) {

Log.w(TAG, getString(R.string.error\_receiving\_photos), e);

}

}

}

private void dispatchTakePictureIntent() {

try {

File storageDir = Environment.getExternalStoragePublicDirectory(

Environment.DIRECTORY\_PICTURES);

pictureImagePath = storageDir.getAbsolutePath() + "/" + fileSaveImageName;

File file = new File(pictureImagePath);

outputFileUri = Uri.fromFile(file);

Intent cameraIntent = new Intent(MediaStore.INTENT\_ACTION\_STILL\_IMAGE\_CAMERA);

cameraIntent.putExtra(MediaStore.EXTRA\_OUTPUT, outputFileUri);

startActivityForResult(cameraIntent, REQUEST\_IMAGE\_CAPTURE);

} catch (ActivityNotFoundException e) {

ActivityHelper.showToast(getString(R.string.device\_not\_support\_shooting), getApplicationContext());

if (BuildConfig.DEBUG) {

Log.e(TAG, "Device does not support shooting", e);

}

}

}

@Override

public void onBackPressed() {

if (drawerResult.isDrawerOpen()) {

drawerResult.closeDrawer();

} else {

mExitDialog.show();

}

}

}