

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»
Профиль подготовки: «Вычислительные машины, комплексы, системы и сети»

БАКАЛАВРСКАЯ РАБОТА

на тему

**ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ
ПРОВЕРКИ QR-КОДОВ ТОВАРОВ**

Студент Гросс 31.05.2022 Гросс Дмитрий Александрович
(подпись, дата) (ФИО полностью)

Руководитель Дубравин 31.05.2022 Дубравин А.В.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр Бычков 31.05.2022 Бычков А. С.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 02.06.2022 г. № 13)

Заведующий кафедрой

(подпись)

М.А. Митрохин

Работа защищена с отметкой хорошо (протокол заседания ГЭК от
20.06.2022 № 20)

Секретарь ГЭК

(подпись)

Е.И. Гурин

Пенза 2022

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«26» апреля 2022 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Гросс Дмитрий Александрович, гр.18BV1 факультета вычислительной техники.

2. Тема работы: «Программное обеспечение проверки QR-кодов товаров»

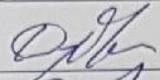
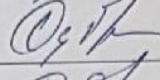
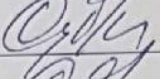
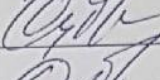
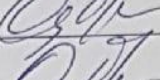
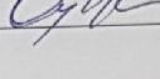
Тема утверждена приказом ПГУ №0357а/о от "26" апреля 2022 г.

3. Руководитель бакалаврской работы к.т.н., доцент каф. ВТ ПГУ Дубравин А.В.

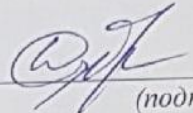
4. Задание на работу (назначение разработки, исходные данные и т.п.)
Разработать программное обеспечение проверки QR-кодов товаров. Оно должно сканировать QR и штрих коды товаров, и выводить информацию о нём, также должна выводиться информация об акциях, которые проводят магазины.

5. Перечень подлежащих разработке вопросов
Требуется произвести анализ предметной области, разработать архитектуру приложения, выбрать инструменты для разработки, спроектировать интерфейс приложения.

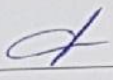
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области	15%	26.04.22-29.04.22	
2	Выбор инструментов для разработки	15%	30.04.22-6.05.22	
3	Разработка серверной части	20%	7.05.22-13.05.22	
4	Разработка клиентской части	25%	14.05.22-22.05.22	
5	Выполнение задания по БЖД	10%	22.05.22-26.05.22	
6	Оформление выпускной работы.	15%	26.05.22-31.05.22	

Дата выдачи задания "26" апреля 2022 г.

Руководитель бакалаврской работы  26.04.22, Дубравин А.В.
(подпись, дата)

Задание к исполнению принял студент  26.04.22, Гросс Д.А.
(подпись, дата)

Работу к защите допустить
Декан факультета  20.06.2022, Фионова Л.Р.
(подпись, дата)

Реферат

Работа содержит 90 листов, 20 рисунков, 1 таблицу, 6 источников, 3 приложения.

ANDROID, KOTLIN, RETROFIT, PYTHON, FLASK, КЛИЕНТ-СЕРВЕРНОЕ ПРИЛОЖЕНИЕ, GOOGLE API ML KIT.

Целью ВКР является разработка программного обеспечения для сканирования *QR* кодов товаров. Сканирование осуществляется с помощью *Google API ML Kit*, после чего данные выполняются запрос, по серверу, на котором выполняется поиск по наименованию товара, и эти данные отправляются пользователю. После выбора пользователем наименования товара, на сервере выполняется поиск по выбранному наименованию и отправляет данные пользователю и пользователю выдаются данные о нужном ему товаре. Также есть возможность просматривать акции, которые проводят магазины, добавления товаров в список.

В результате выполнения ВКР, было разработано клиент-серверное приложение, которое удовлетворяет поставленной задаче.

					ПГУ 1.090.01.10.001 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб		Гросс Д.А.					
Пров		Дубравин А.В.					
Н. Контр.		Бычков А.С.					
Утв		Митрохин М.А.					
Программное обеспечение проверки QR кодов товаров Пояснительная записка					Лит.	Лист	Листов
						2	82
					ФВТ, каф. ВТ гр. 18ВВ1		

Содержание

Введение.....	5
1 Программное обеспечение проверки QR- кодов товаров	6
1.1 Анализ предметной области для решения поставленной задачи.....	6
1.1.1 Анализ темы	6
1.1.2 Анализ аналогов.....	6
1.1.3 Постановка задачи	10
1.2 Архитектура приложения.....	10
1.2.1 Выбор архитектуры	10
1.2.2 Разработка UML-диаграмм	11
1.2.3 Описание используемых инструментов и библиотек.....	11
1.3 Выбор инструментов разработки	13
1.3.1 Выбор языка программирования	13
1.3.2 Выбор среды разработки.....	14
1.4 Разработка сервера.....	16
1.4.1 Разработка сервера	16
1.5 Разработка клиента	21
1.5.1 Сканер QR и штрих кода.....	21
1.5.2 Описание работы экрана MainActivity2	27
1.5.3 Описание работы сканирования QR кода	30
1.5.4 Описание работы сканирования штрих кода.....	32
2 Безопасность жизнедеятельности.....	41
2.1 Характеристика опасных факторов при работе с ПВЭМ	41
2.2 Классификация и категорирование помещения	42
2.3 Освещение.....	42
2.4 Расчет освещения рабочего места оператора ПЭВМ.....	43
Заключение	46
Список использованных источников	47

Приложение А – UML-диаграммы.....	48
Приложение Б – Листинг программ.....	54
Приложение В – Презентация.....	82

Введение

Люди каждый день ходят в магазины за продуктами, просматривают каждый товар, его цену, есть ли на него скидка, какие товары учувствуют в акциях и тратят на это большое количество времени. Практически никто уже не использует листовки с написанными на них акциями, скидками на товары в магазинах. Сейчас всю нужную информацию можно найти в приложении на телефоне или сайте, экономя при этом много личного времени, и денег.

					<i>ПГУ 1.090301.10.001 ПЗ</i>	Лист
						5
Изм.	Лист	№ докум	Подпись	Дата		

1 Программное обеспечение проверки QR- кодов товаров

1.1 Анализ предметной области для решения поставленной задачи

1.1.1 Анализ темы

Люди все больше предпочитают, быстры и мобильный доступ к информации, популярность интернет–магазинов растет с каждым днём. Интернет один из основных и эффективных способов взаимодействия с покупателями. В настоящее время цены в магазин выросли, и сейчас люди больше предпочитают следить за акциями в магазинах, просматривать цены на товары через приложения, чтобы сэкономить. Просматривать каждый товар в магазине очень долго и нецелесообразно, поэтому было принято автоматизировать этот процесс с помощью мобильного приложения, ведь практически каждый человек имеет во владениях мобильный телефон.

Пользователь сможет сканировать штрих–код и QR код товара и просматривать его стоимость в различных магазинах, сравнивать их между собой, и выбирать подходящий ему вариант по цене. Также будут отображаться проходящие акции. Пользователь сможет составлять список нужных ему товаров.

1.1.2 Анализ аналогов

Приложение «Qrooto» – каталог скидок и акций в магазинах. Приложение позволяет получать кэшбэк, смотреть информацию о скидках и акциях в магазинах. Также в приложении можно сканировать чек и получать за это баллы, производить поиск товара, присутствует чат с техподдержкой. Но, можно сканировать только определенное количество чеков в день, часто происходит отклонение при сканировании чека. В приложении редко обновляется информация об акциях, присутствует небольшое количество рекламы. Внешне приложение представлено на рисунке 1.1.



Рисунок 1.1 – Приложение Qrooto

Самым популярным является приложение «Едадил». В нем тоже можно посмотреть акции в магазинах популярных сетей, получить кэшбэк отсканировав QR с чека. Есть возможность отсканировать QR и штрих – код товара и узнать в каком магазине он стоит дешевле. Можно добавлять карты на скидку любых магазинов. Но в приложении присутствует большое количество рекламы, кэшбэк действуют только на определенные товары. Приложение показано на рисунке 1.2.

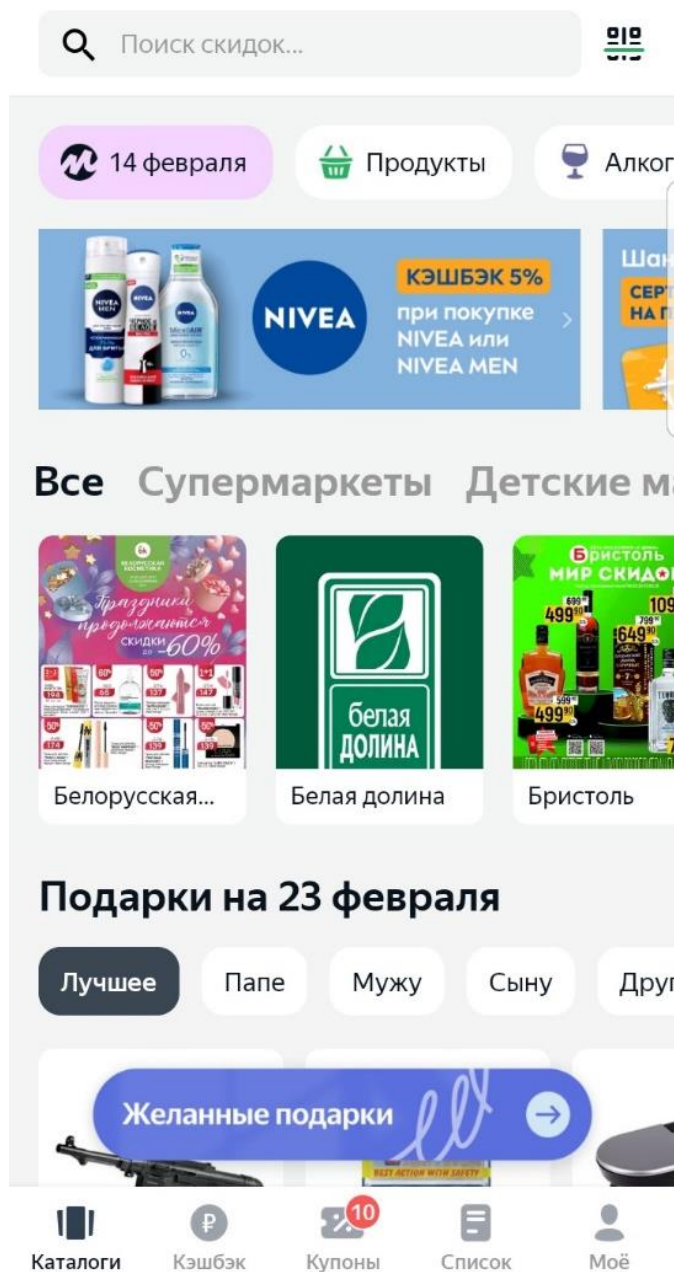


Рисунок 1. 2– Приложение Едадил

Приложение «Disconto». В этом приложении можно сканировать и проверить свой чек и участие в акциях, узнать о количестве начисляемого кэшбэка. Можно посмотреть историю отсканированных вами чеков. Есть большое количество акций магазинов сетей. Отсутствует поиск и фильтр, позволяющие найти вам акцию для нужного продукта. Внешне приложение представлено на рисунке 1.3.

Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

Лист

8

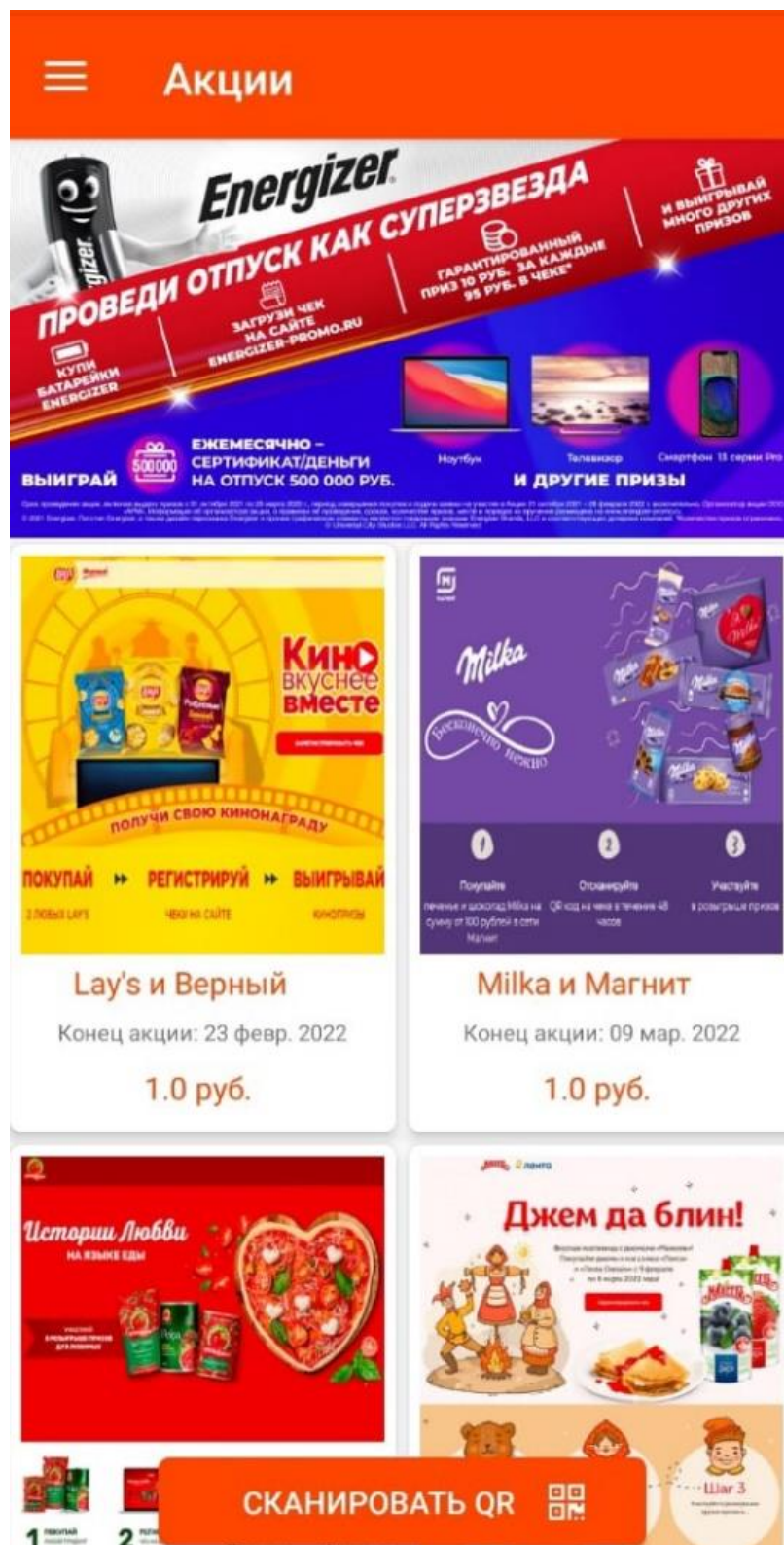


Рисунок 1. 3 – Приложение *Disconto*

Результаты сравнительного анализа приложений представлены в таблице 1.1.

Таблица 1.1 – Анализ приложений

Название	Функции				
	Сканирование QR и штрих-кода	Удобный интерфейс	Наличие виджета	Расширенный функционал	Список покупок
Едадил	+	+	-	+	+
Disconto	+	+	-	-	-
Qrooto	+	-	-	-	+

Проанализировав существующие аналоги, была составлена таблица с плюсами и минусами данных приложений, которые будут рассмотрены в дальнейшем при разработке приложения.

Было принято решение полностью исключить добавления рекламы в приложение. Также не добавлять в приложение считывание чеков и начисление баллов. Особое внимание нужно уделить разработке удобного интерфейса и подобрать идеальную цветовую гамму, чтобы пользователю было приятно пользоваться приложением.

1.1.3 Постановка задачи

Была поставлена задача разработать мобильное приложение со следующим функционалом:

- возможность сканирования QR и штрих – кода продукта;
- возможность выводить информацию о товаре с интернет-магазинов;
- раздел со списком покупок.

Также, программное обеспечение должно обладать приятным и интуитивно понятным для пользователя графическим интерфейсом.

1.2 Архитектура приложения

1.2.1 Выбор архитектуры

Клиент-серверная архитектур необходима для распределения обязанностей между клиентом и сервером. Сервер выполняет функции, запросы, операции. Операции могут выполняться длительное время, именно

поэтому необходимо использовать сервер. Также сервер хранит большое количество данных, чтобы не нагружать устройство пользователя. Именно поэтому было принято решение использовать клиент-серверную архитектуру.

1.2.2 Разработка *UML*-диаграмм

Все разработанные *UML*-диаграммы представлены в приложении Б.

В приложении реализована клиент-серверная архитектура. Была спроектирована диаграмма прецедентов для определения функционала клиента. Также была разработана диаграмма развертывания, которая показывает связь между клиентом и сервером.

Сканирование *Qr* и штрих кода является одной из главных функций приложения. После того, как пользователь отсканировал штрих код продукта, отсканированные данные отправляются на сервер, где происходит поиск наименований товаров по штрих коду. После нахождения названий товаров, по штрих коду. Из этого списка пользователю предлагается выбрать наиболее подходящее название его продукта. После выбора клиенту показывается список продуктов из различных магазинов (Лента, Спар, Перекресток) и цены этих продуктов. Анализирую этот список, пользователь может выбрать наиболее подходящий для него вариант, таким образом сэкономив деньги. Взаимодействие клиента и сервера показаны на диаграмме последовательности.

1.2.3 Описание используемых инструментов и библиотек

При разработки мобильного приложения использовался стандартный набор инструментов - *Android SDK*. Он содержит в себе все нужные классы для работы с графическим интерфейсом и утилиты для отладки и сборки приложения.

Для сканирования *QR* и штрих кодов использовалась библиотека от компании *Google API ML Kit*, которая позволяет распознавать и считывать различные форматы штрих-кодов.

Библиотека *CameraX* – библиотека поддержки *Android Jetpack*, которая служит для облегчения разработки приложения связанных с камерой. Она позволяет использовать предварительный просмотр изображения на дисплее, сохранять изображения и редактировать.

Для извлечения необходимых данных с сайтов магазинов, была использована библиотека *Bs4* написанная на языке *Python*, она поддерживает ряд сторонних парсеров, *html5lib*, *lxml*, *xml*, которые используются для разбора *XML* документов.

Фреймворк *Flask*[3] это веб-фреймворк написанный на *Python*, он не требует специальных инструментов, позволяет быстро создавать серверные приложения. С помощью данного фреймворка можно удобно и быстро создавать *RESTAPI*.

Для сравнения строк использовалась библиотека *FuzzyWuzzy*, эта библиотека эффективна для различения двух строк, которые относятся к одному объекту, но написаны они по-разному. В основу этой библиотеки лег алгоритм Левенштейна. Этот алгоритм позволяет сравнить две строки и узнать, на сколько они близки друг к другу.

Библиотека *Retrofit*[5] необходима для упрощения работы с *REST API*. Для запроса к веб-сервисам используются команды *Post*, *Get*, *Put*. Также библиотека поддерживает различные форматы данных, включая такие популярные форматы, как *XML* и *JSON*.

Для того, чтобы загружать изображения используя *url* адрес была использована библиотека *Picasso*.

Так как для приложения создавалось *REST API* необходимо было его протестировать, для этого использовалась программа *Postman*, которая позволяет быстро и удобно протестировать *API* своего приложения, отправлять различные запросы (*GET*, *POST* и другие), получать ответы в удобном формате, узнавать время, за которое приходит ответ, *HTTP* код ответа. Также эта программа позволяет писать скрипты для удобного и

автоматизированного тестирования. Интерфейс программы представлен на рисунке 1. 4.

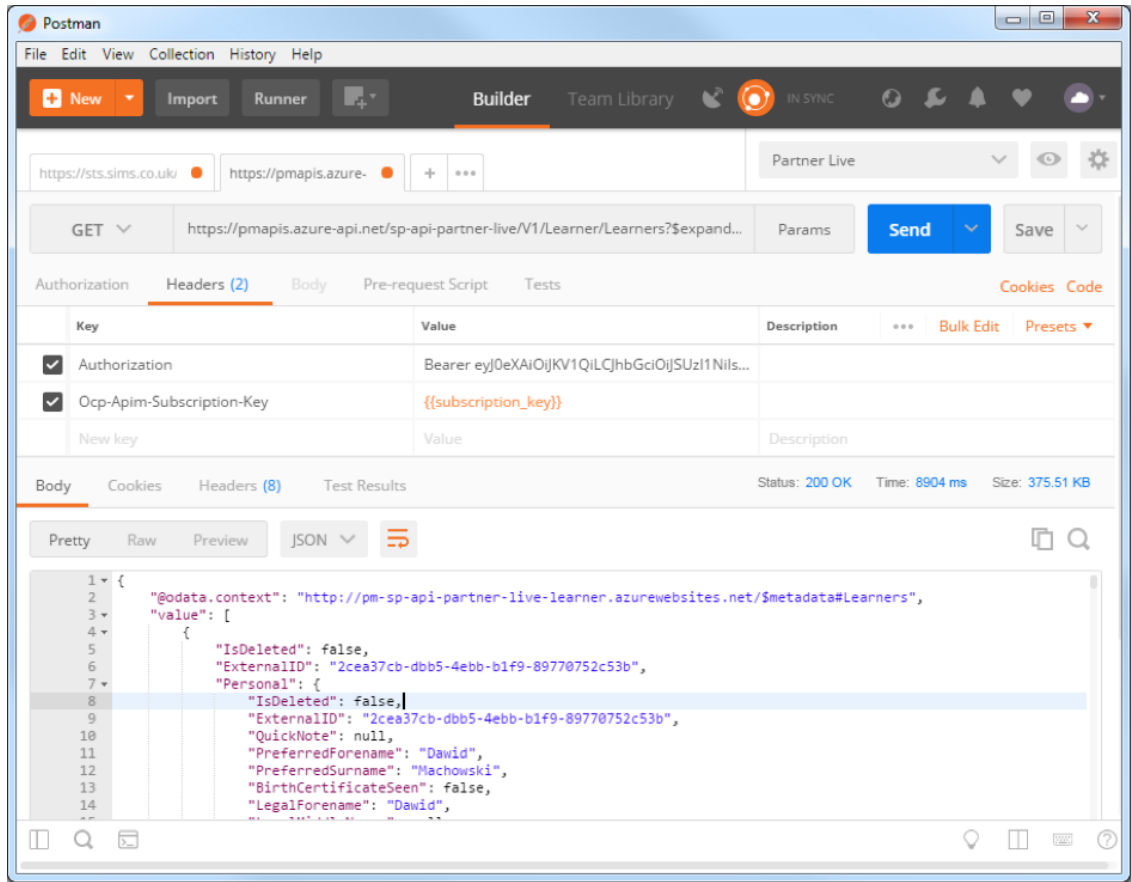


Рисунок 1.4 – Интерфейс программы

1.3 Выбор инструментов разработки

1.3.1 Выбор языка программирования

Прежде чем приступить к разработке приложения был выбор, какой язык программирования использовать. Было рассмотрено два языка *Java* и *Kotlin*.

Java появился намного раньше *Kotlin* и собрал рядом с собой большое количество пользователей, поэтому без проблем можно найти ответ на любой вопрос. Можно почитать документацию или же обратиться к другим разработчикам, которые вероятнее всего ответят на ваш вопрос. Имеется большое количество библиотек и фреймворков для помощи в написании программ и приложений.

Java является языком высокого уровня и такие языки преобразуются с использованием компиляторов и интерпретаторов, что упрощает разработку, и делая его более легким для написания, и чтения. Также язык *Java* строго типизирован, любое выражение или переменная имеют определенный тип на момент компиляции, компилятор сам подскажет, где была допущена ошибка и не даст её совершить.

Kotlin появился относительно недавно – в 2011 году, его разработала Российская компания *JetBrains*. Несмотря на то, что язык относительно молодой, в интернете можно найти большое количество примеров на нём. Также есть отлично написанная документация, которая покрывает все вопросы. Этот язык имеет интуитивный синтаксис, который улучшает скорость написания программ, также имеет хорошую производительность. В *Kotlin* есть возможность простого создания классов, необходимо добавить *data* в определение класса, и компилятор всё остальное сделает сам, так в *Java* классы необходимо прописывать самому, что приводит к большому количеству строк кода. *Kotlin* полностью совместим с *Java*, есть доступ к любым библиотекам, модули, написанные на *Kotlin*, будут работать с *Java*. Также компилятор дает возможность совмещать оба языка в одном проекте. *Kotlin* является официальным языком для создания android приложений, начиная с 2017 года. Компания *Google* разрабатывает библиотеку *Jetpack* с применением *Kotlin* и используют все современные конструкции и особенности этого языка.

Проанализировав представленные два языка, для разработки приложения был выбран *Kotlin*[1]. Большое количество библиотек для решения поставленных задач, краток в написании кода.

1.3.2 Выбор среды разработки

Для разработки приложения были рассмотрены такие средства разработки как: *Eclipse* и *AndroidStudio*.

Среда разработки Eclipse была создана компанией *EclipseFoundation* на основе *Java*. В ней представлены ряд функция и широкий набор инструментов, которые упрощают разработку приложения. Среда подходит для компьютеров с низко производительностью. Имеет функции для работы с сервером базой данных. Подходит для разработки мобильных и веб-приложений. Но после появления *AndroidStudio*, *Google* остановила поддержку среды для разработки приложений на *Android*[4].

Среда разработки *AndroidStudio* была анонсирована разработчиками *Google I/O*, основой для которой стал проект компании *JetBrains – IntelliJ* и созданная для разработки *Android*. *Android*, как операционная система, лидирует во всем мире. В настоящее время 2,8 млрд человек используют устройства на базе операционной системы *Android*, доля рынка составляет 75%.

AndroidStudio поддерживает работу с популярными языками программирования – *C/C++*, *Java*, *Kotlin*. Удобный редактор кода, большая библиотека с готовыми компонентами и шаблонами для разработки приложений. В *AndroidStudio* есть возможность разработки приложения для последней версии *AndroidN* и после создания приложения можно проверить на наличие ошибок благодаря большому набору инструментов для тестирования каждого элемента приложения. Есть инструменты для

просмотра эффективности рекламных объявлений, улучшения качества приложений и монетизации. В *Android Studio* есть поддержка системы «*Codeinjection*», которая позволяет редактировать фрагменты кода с привязкой к различным языкам. Также эта среда программирования позволяет разрабатывать приложения для смартфонов, приставок для телевизоров *AndroidTV* и для мобильных устройств с любым необычным разрешением экранов. В *AndroidStudio* есть встроенный эмулятор для проведения тестов производительности разрабатываемых приложений.

Можно отслеживать загруженность процессора и при необходимости провести оптимизацию.

Изучив представленные среды разработки, было принято решение использовать *Android Studio*.

1.4 Разработка сервера

1.4.1 Разработка сервера

Большинство сервисов, которые предоставляют информацию о наименованиях по штрих коду, не предоставляют возможность работать российским разработчикам, поэтому нужно было придумать другое решение. В интернете был найден сайт *Barcode-list.ru*. Данный сайт содержит большую базу данных штрих кодов и позволяет производить поиск наименованию товаров по коду. База данных постоянно расширяется благодаря анализу номенклатур, которые используются их партнерами и клиентами в сервисе *Microinvest Склад Pro*.

В продукт *Microinvest Склад Pro* эта база данных встроена, и после считывания штрих-кода «правильное» наименование продукта автоматически выводится системой. Пользователю остается сохранить его или ввести свой вариант.

Было принято решение использовать данный сервис для поиска наименования товаров по штрих коду. Скриншот результат поиска представлен на рисунке 1.5

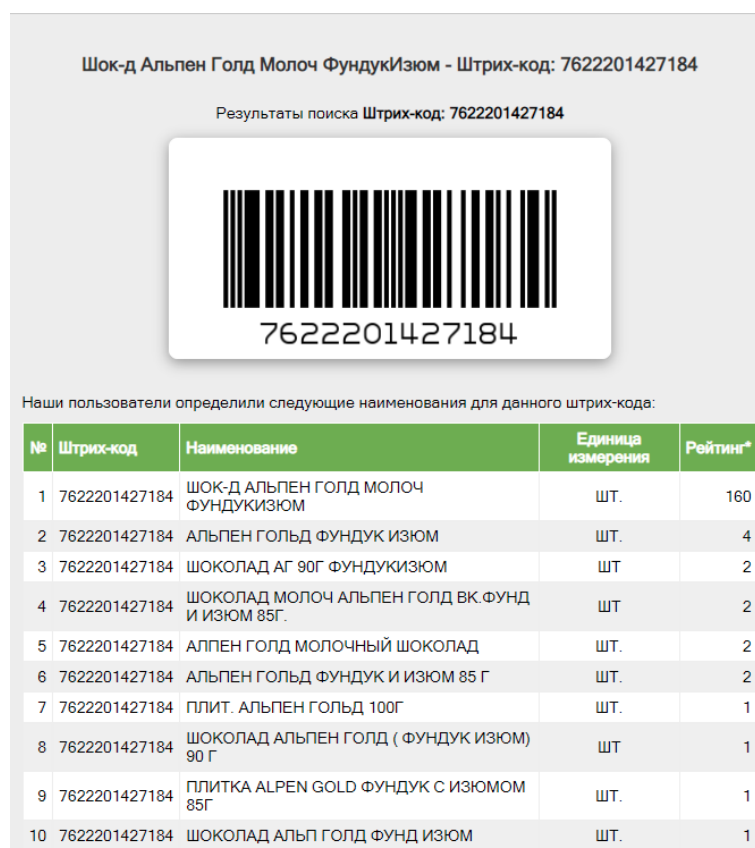


Рисунок 1.5 – Скриншот результат поиска

На данном скриншоте видны цифры штрих код и таблица с данными. Данная таблица включает в себя следующие колонки: номер товара, штрих код товара, наименование товара, измерение товара и рейтинг товара. Рейтинг показывает какое количество пользователей, посчитало это наименование более подходящим.

Была написана функция, которая переходит по этому сайту и выбирает необходимые данные и выше описанной таблицы. Данный процесс был осуществлен с помощью библиотеки *BeautifulSoup4*. На сайте была найдена таблица по её классу, а затем в цикле были выбраны необходимые данные, а именно наименование товара. Фрагмент кода представлен ниже.

```
defsearchBarcode (barcode):
list = []
result = requests.get('https://barcode-
list.ru/barcode/RU/Поиск.htm?barcode='+barcode)
```

```

src = result.content
soup = BeautifulSoup(src, "lxml")
table = soup.find(class_="randomBarcodes")
rows = table.find_all("tr")
for ind, row in enumerate(rows):
    if ind == 0: continue
    cols = row.find_all('td')
    cols = [ele.text.strip() for ele in cols][2]
    list.append(cols)
return list

```

При разработке сервера был использован фреймворк *flask*. С его помощью был разработан *RESTAPI*, которая отвечает за взаимодействия между серверными и клиентскими приложениями. В данной работе для сервера было создано три *GET* запроса для получения данных. Первый запрос: *productTitle* используется для получения всех наименований товара, связанных со штрих кодом, который передается в качестве параметра.

Фрагмент кода запроса представлен ниже.

```

@app.route('/productTitle')
def productTitleList():
    barcode = request.args.get('barcode')
    result = searchBarcode(barcode)
    return jsonify({'productList': result})

```

В программе *Postman* запрос *productTitle* представлен на рисунке 1.6.

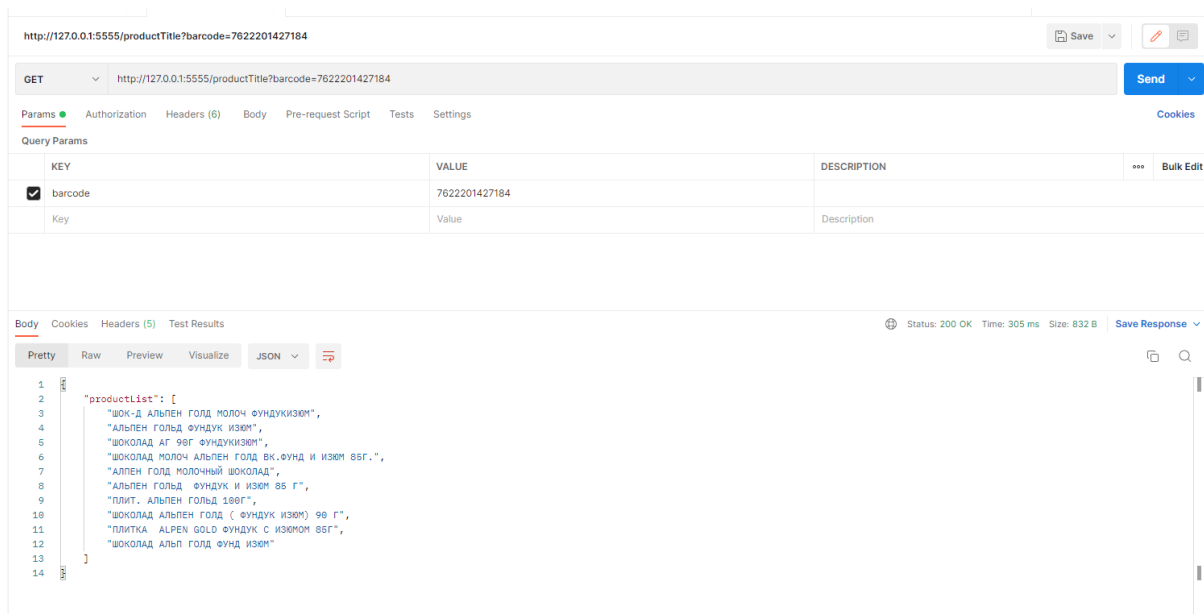


Рисунок 1. 6– Запрос *productTitle*

Второй запрос: *similarProducts* осуществляет поиск товаров из разных магазинов. На вход приходит один параметр – название товара, которое выбрал пользователь. Функция *searchSimilar* используется для поиска товаров из разных магазинов. После поиска результат отправляется клиенту. Фрагмент кода представлен ниже.

```
@app.route('/similarProducts')
defsimilarProducts():
    title = request.args.get('title')
    result = searchSimilar(productList, title)
    result = [item.serialize() for item in result]
    return jsonify({'similar' : result})
```

Запрос *similarProducts* представлен на рисунке 1.7.

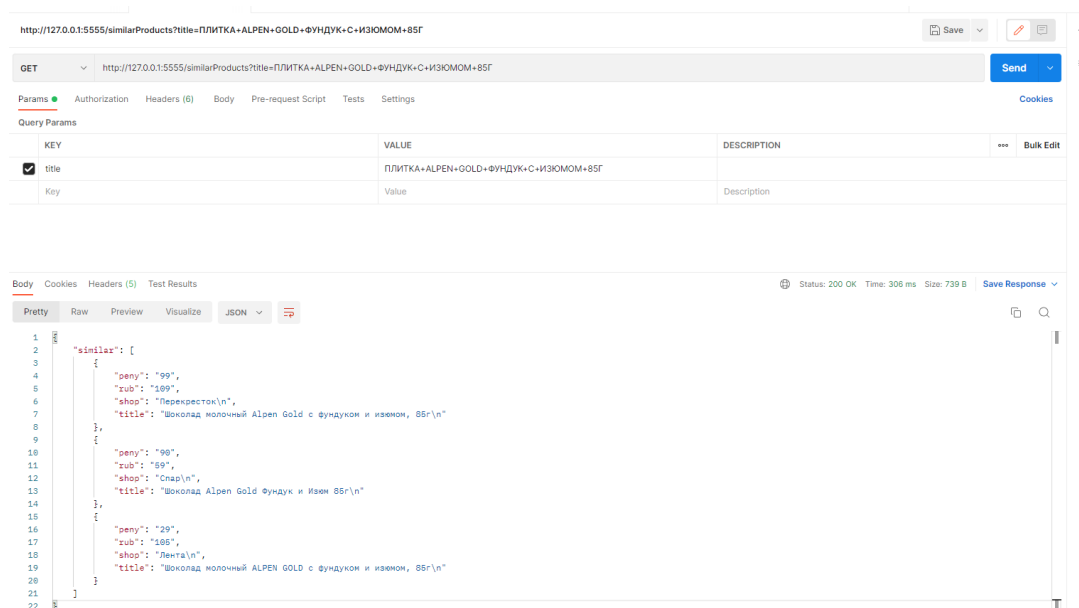


Рисунок 1. 7 – Запрос *similarProducts*

Третий запрос: *stockList* используется для получения списка текущих акция магазинов. Также фрагмент кода представлен ниже.

```
@app.route('/stockList')
defstockList():
    result = [item.serialize() for item in stocks]
    return jsonify({'stocks' : result})
```

Для получения данных о товарах были использованы следующие сайты (Лента, Спар, Перекрёсток). На этих сайтах установлена антипарсинговая система, которая защищает сайты от ботов. Поэтому при написании программ для парсинга данных и частых обращениях к нему, можно получить блокировку по IP. Было принято решение использовать прокси. Прокси является неким промежуточным фильтром между пользователем и сервером, откуда запрашивается информация о данных.

1.5 Разработка клиента

1.5.1 Сканер QR и штрих кода

Для сканирования *QR/Barcode* кода, была написана программа с использованием библиотек *CameraX*[6] и *MLKit*

Для отображения отсканированных данных в *activity_main.xml* добавили *PreviewView* и *TextView*. *PreviewView* позволяет отображать канал камеры для сценария использования *CameraX*. Код представлен ниже.

```
<?xmlversion="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.camera.view.PreviewView
        android:id="@+id/preview_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <TextView
        android:id="@+id/tvScannedData"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        android:layout_marginBottom="50dp"
        android:gravity="center"
        android:textColor="@android:color/white"
        android:textSize="20sp"

        app:layout_constraintBottom_toBottomOf="parent"
```

```

        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>

```

Теперь мы проверяем разрешение на использование камеры, чтобы проверить, было ли предоставлено разрешение камеры, и если нет, то запросить его.

Код представлен ниже.

```

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        if (isCameraPermissionGranted()) {
        } else {
            ActivityCompat.requestPermissions(
                this,
                arrayOf (Manifest.permission.CAMERA),
                PERMISSION_CAMERA_REQUEST
            )
        }
    }

    override fun onRequestPermissionsResult(
        requestCode: Int,
        permissions: Array<String>,
        grantResults: IntArray
    ) {
        if (requestCode == PERMISSION_CAMERA_REQUEST) {
            if (isCameraPermissionGranted()) {
            } else {

                Log.e(TAG, "no camera permission")
            }
        }
    }
}

```

```

        super.onRequestPermissionsResult(requestCode,
permissions, grantResults)
    }
    private fun isCameraPermissionGranted(): Boolean {
        return ContextCompat.checkSelfPermission(
            baseContext,
            Manifest.permission.CAMERA
        ) == PackageManager.PERMISSION_GRANTED
    }

    companion object {
        private val TAG = MainActivity::class.java.simpleName
        private const val PERMISSION_CAMERA_REQUEST = 1
    }
}

```

Был разработан класс *CameraXViewModel()*.
cameraProviderFuture.addListener() добавляет прослушивателя к
cameraProviderFuture. В качестве одного аргумента он принимает *Runnable*, а в
качестве второго *ContextCompat.getMainExecutor()*. Код представлен ниже.

```

class CameraXViewModel(application: Application) :
    AndroidViewModel(application) {

    private var cameraProviderLiveData:
    MutableLiveData<ProcessCameraProvider>? = null

    val processCameraProvider: LiveData<ProcessCameraProvider>
    get() {
        if (cameraProviderLiveData == null) {
            cameraProviderLiveData = MutableLiveData()
            val cameraProviderFuture =
            ProcessCameraProvider.getInstance(application())
            cameraProviderFuture.addListener(

```

```

        Runnable {
            try {

cameraProviderLiveData!!.setValue(cameraProviderFuture.get())
                } catch (e: ExecutionException) {
                    Log.e(TAG, "Unhandled exception", e)
                }
            },

ContextCompat.getMainExecutor(getApplication())
        )
    }
    return cameraProviderLiveData!!
}

companion object {
    private const val TAG = "CameraXViewModel"
}
}

```

Для предварительного просмотра варианта использования, создали новую функцию *bindPreviewUseCase()*. В *bindPreviewUseCase()* привязали объект *cameraSelector* и предварительный просмотр к экземпляру *ProcessCameraProvider*. Код представлен ниже.

```

private fun bindPreviewUseCase() {
    if (previewUseCase != null) {
        cameraProvider!!.unbind(previewUseCase)
    }
    previewUseCase = Preview.Builder()
        .setTargetRotation(previewView!!.display.rotation)
        .build()

    previewUseCase!!.setSurfaceProvider(previewView!!.surfaceProvide
r)

```

```

try {
    cameraProvider!!.bindToLifecycle(
        this,
        cameraSelector!!,
        previewUseCase
    )
} catch (illegalStateException: IllegalStateException) {
    Log.e(TAG, illegalStateException.message ?:
"IllegalStateException")
} catch (illegalArgumentException: IllegalArgumentException)
{
    Log.e(TAG, illegalArgumentException.message ?:
"IllegalArgumentException")
}
}

```

Функция *bindAnalyseUseCase()* для реализации варианта использования анализа. Внутри блока *bindAnalyseUseCase()* был создан экземпляр *BarcodeScanner* и экземпляр *ImageAnalysis*. Код представлен ниже.

```

private fun bindAnalyseUseCase() {
    val options = BarcodeScannerOptions.Builder()
        .setBarcodeFormats(Barcode.FORMAT_ALL_FORMATS).build()

    val barcodeScanner: BarcodeScanner =
BarcodeScanning.getClient(options)
    analysisUseCase = ImageAnalysis.Builder()
        .setTargetRotation(previewView!!.display.rotation)
        .build()

    val cameraExecutor = Executors.newSingleThreadExecutor()

    analysisUseCase?.setAnalyzer(
        cameraExecutor,
        ImageAnalysis.Analyzer { imageProxy ->
            processImageProxy(barcodeScanner, imageProxy)
        }
    )
}

```

```

        }
    )
    try {

        cameraProvider!!.bindToLifecycle(
            this,
            cameraSelector!!,
            analysisUseCase
        )
    } catch (illegalStateException: IllegalStateException) {
        Log.e(TAG, illegalStateException.message ?:
            "IllegalStateException")
    } catch (illegalArgumentException: IllegalArgumentException)
    {
        Log.e(TAG, illegalArgumentException.message ?:
            "IllegalArgumentException")
    }
}

```

В *processImageProxy()* был создан объект *InputImage* для распознавания штрих-кода в изображении. Далее передаем объект *InputImage* в метод процесса *BarcodeScanner*. Код представлен ниже.

```

private fun processImageProxy(
    barcodeScanner: BarcodeScanner,
    imageProxy: ImageProxy
) {
    val inputImage =
        InputImage.fromMediaImage(imageProxy.image!!,
            imageProxy.imageInfo.rotationDegrees)

    barcodeScanner.process(inputImage)
        .addOnSuccessListener { barcodes ->
            barcodes.forEach { barcode ->
                val bounds = barcode.boundingBox
            }
        }
}

```

```

        val corners = barcode.cornerPoints

        val rawValue = barcode.rawValue
        tvScannedData.text = barcode.rawValue

        val valueType = barcode.valueType
        when (valueType) {
            Barcode.TYPE_WIFI -> {
                val ssid = barcode.wifi!!.ssid
                val password = barcode.wifi!!.password
                val type = barcode.wifi!!.encryptionType
                tvScannedData.text =
                    "ssid: " + ssid + "\npassword: " +
password + "\ntype: " + type
            }
            Barcode.TYPE_URL -> {
                val title = barcode.url!!.title
                val url = barcode.url!!.url
                tvScannedData.text = "Title: " + title +
"\nURL: " + url
            }
        }
    }
    .addOnFailureListener {
        Log.e(TAG, it.message ?: it.toString())
    }
    .addOnCompleteListener {
        imageProxy.close()
    }
}

```

1.5.2 Описание работы экрана *MainActivity2*

Это главный экран, который открывает при запуске приложения. Тут происходит инициализация трёх фрагментов: *StocksFragment*,

ProductsFragment, *SetingsFragment*. Переключение между фрагментами осуществляется с помощью графического интерфейса *BottomNavigationView*.

StocksFragment содержит информацию об акциях, которые проводят магазины. Фрагмент *StocksFragment* представлен на рисунке 1.8.

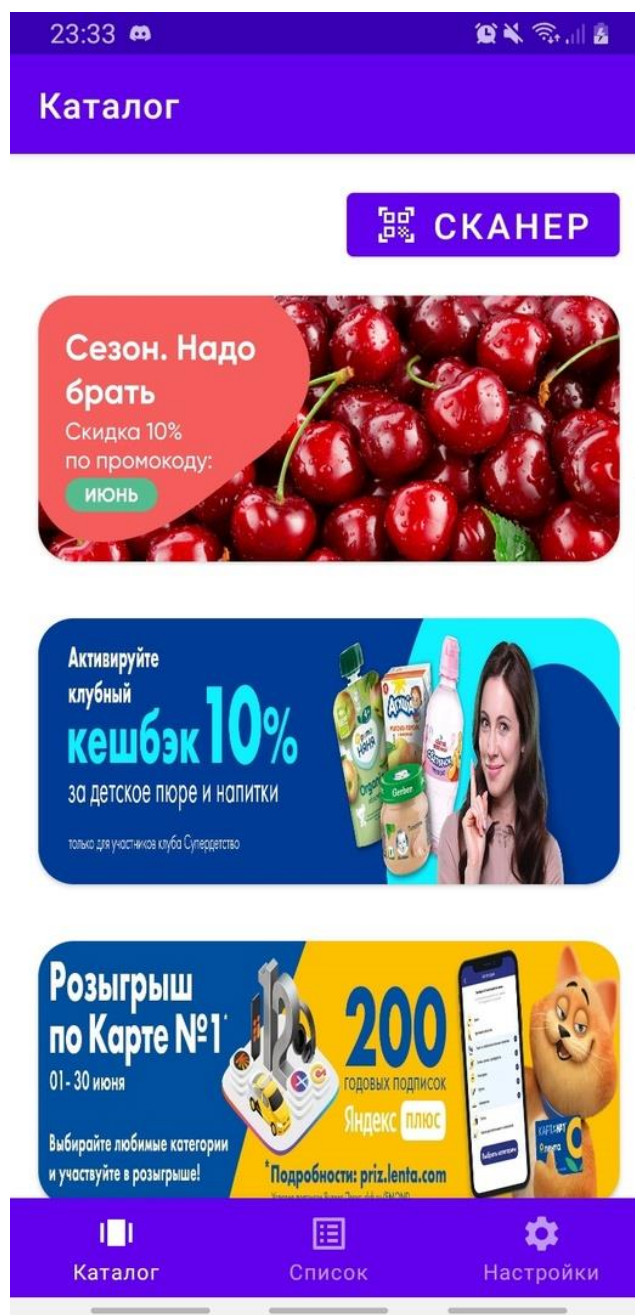
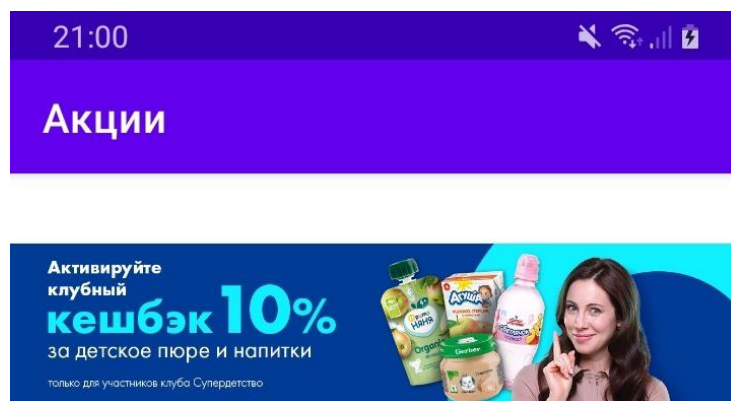


Рисунок 1. 8 – Фрагмент *StocksFragment*

Также при нажатии на акцию, пользователь переходит на другое окно, где может изучить условия проведения акции. Окно с детальной информацией об акции представлено на рисунке 1.9.



С 1 по 30 июня 2022 г. получайте кешбэк за покупки детского пюре и напитков.
Для получения кешбэка:

1. вступайте в клуб Супердетство
2. активируйте предложение в личном кабинете
3. совершайте покупки товаров из детских категорий:

- Фруктовое питание в паучах
- Овощное питание в банках
- Мясное готовое моно питание в банках
- Мясорастительное готовое детское питание в банках
- Рыбное детское питание в банках
- Готовые супы, супы-пюре
- Готовое детское питание Healthy Food
- Детский сок в тетрапаках и паучах
- Детский сок в стекле
- Детская вода до 1 л
- Детская вода более 1 л
- Детский чай
- Другие детские напитки
- Детские напитки Healthy Food

Максимальный лимит по начислению 500 баллов на одного участника. Использовать баллы можно с 01.06.2022 по 31.07.2022 включительно. Неиспользованные баллы сгорают.

Рисунок 1. 9 – Детальная информация об акции

Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

Лист

29

1.5.3 Описание работы сканирования QR кода

На фрагменте *StocksFragment* представлена *Button*, при нажатии на которую происходит открытие камеры для осуществления сканирования QR и штрих-кода. Процесс сканирования QR кода представлен на рисунке 1.10.



Рисунок 1. 10– Процесс сканирования

Изм.	Лист	№ докум	Подпись	Дата

После сканирования QR кода берется *Url* и происходит переход по нему. Переход по *Url* представлен на рисунке 1.11.



Рисунок 1. 11– Процесс перехода по *Url*

1.5.4 Описание работы сканирования штрих кода

Процесс сканирования штрих кода представлен на рисунке 1.12.



Рисунок 1. 12 – Процесс сканирования штрих кода

Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

Лист

32

После сканирования штрих кода, перед пользователем открывается окно для выбора более подходящего наименования товара. Скриншот с наименованием товара представлен на рисунке 1.13.

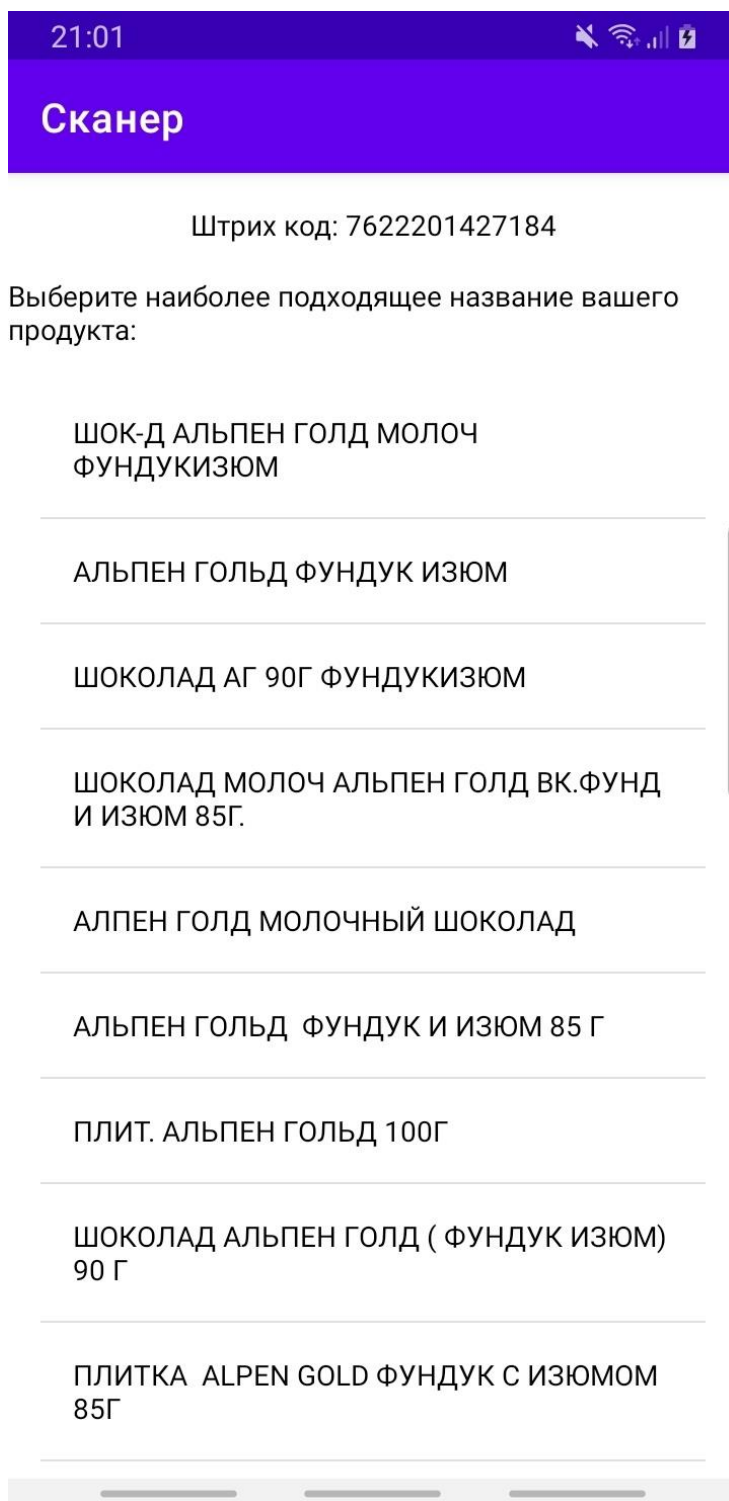


Рисунок 1. 13 – Наименование товара

После того, как пользователь выбрал наиболее подходящее наименование, откроется новый экран с товаром из разных магазинов. Проанализировав их, пользователь может выбрать наиболее подходящий ему товар по цене. Скриншот с товарами представлен на рисунке 1.14.



Рисунок 1. 14 – Список с товарами из магазинов

ProductsFragment отображает список товаров, которые пользователь может внести самостоятельно. Фрагмент *ProductsFragment* представлен на рисунке 1.15.

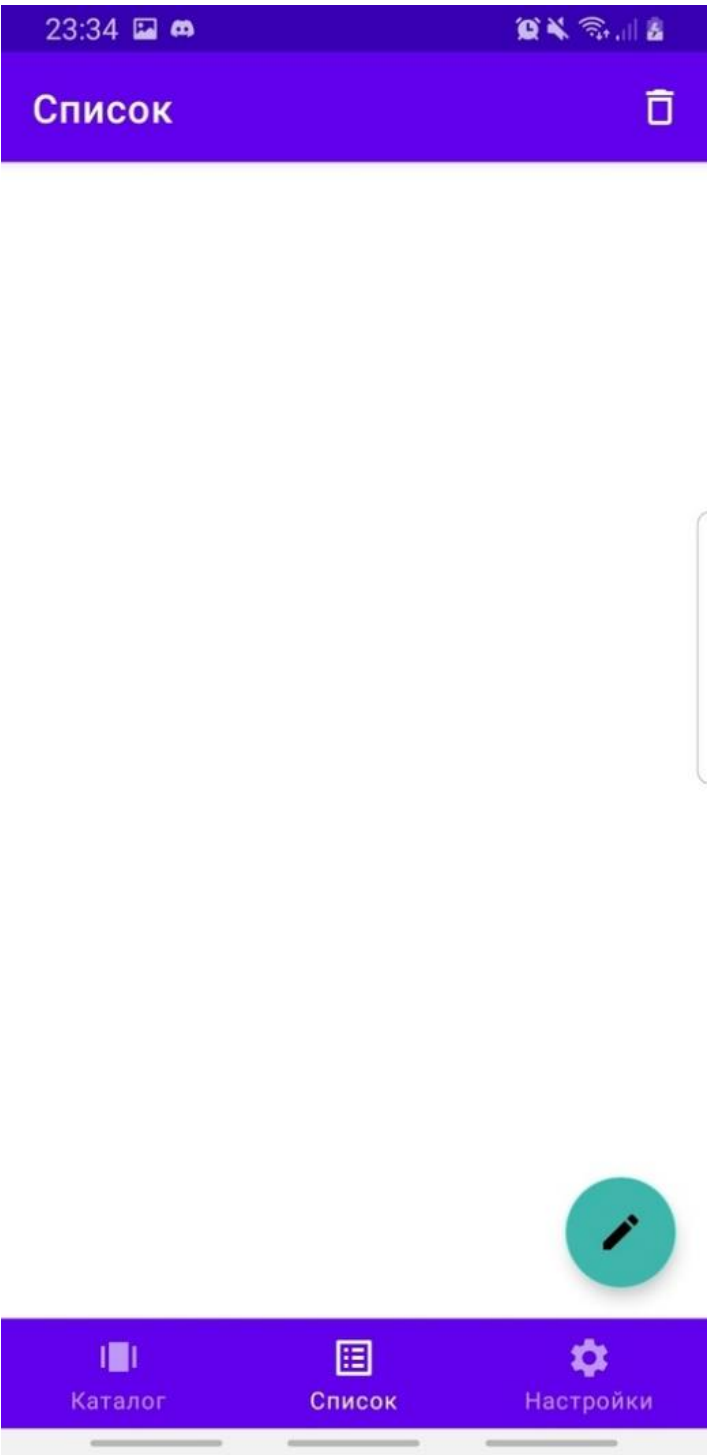


Рисунок 1. 15 – Фрагмент *ProductsFragment*

Для того, чтобы добавить товар в список необходимо нажать на элемент *FloatingActionButton* и перед пользователем появится диалоговое окно, в котором он может вводить любой продукт. Диалоговое окно представлено на рисунке 1.16.

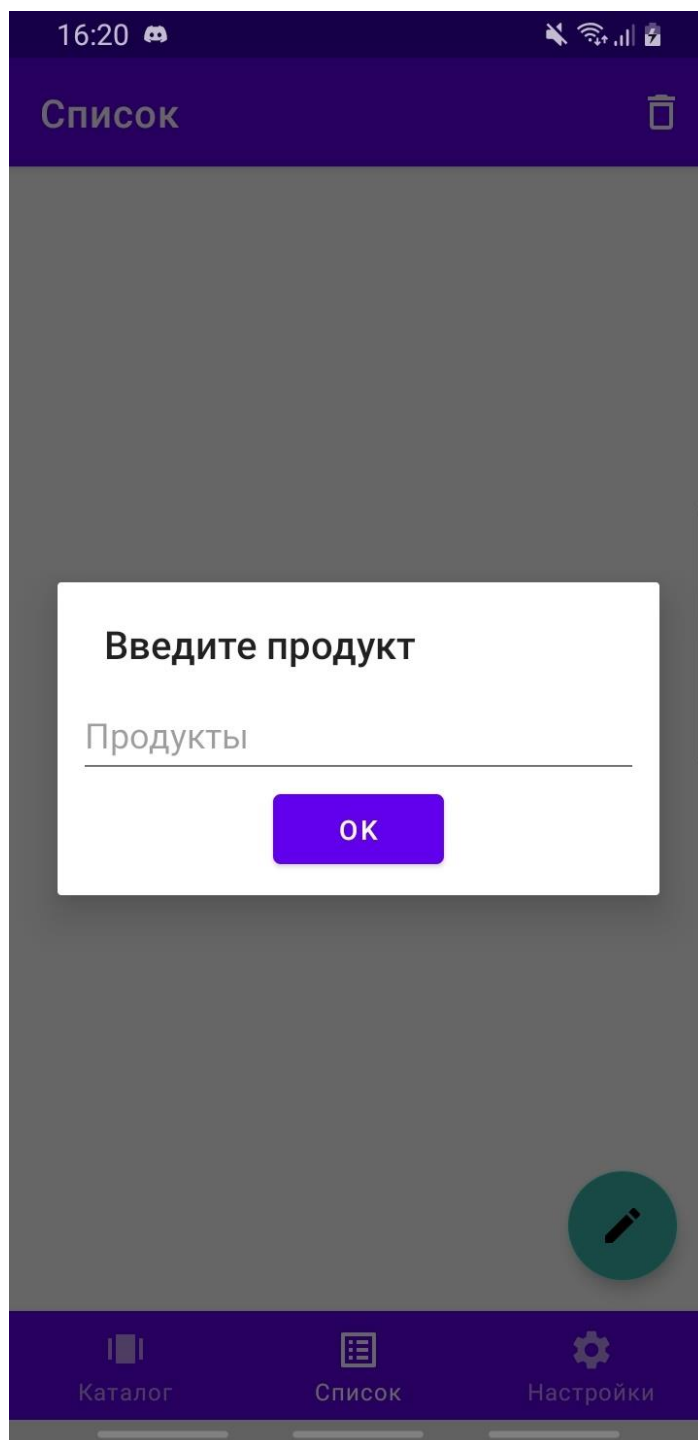


Рисунок 1. 16 – Диалоговое окно

После того, как пользователь ввел данные в диалоговое окно, эти данные появятся в виде списка в *Recyclerview*. Список с товаром представлен на рисунке 1.17.



Рисунок 1. 17 – Список с товаром

Также пользователю предоставляется возможность очистить список с помощью нажатия на элемент меню с иконкой корзины. Удаление списка представлено на рисунке 1.18.



Рисунок 1. 18 – Очистка списка

Фрагмент *SetingsFragment* содержит настройки приложения. Здесь присутствует возможность сменить язык и тему приложения при нажатии на *Switch*. На рисунке 1.19 представлен экран при активации темной темы.

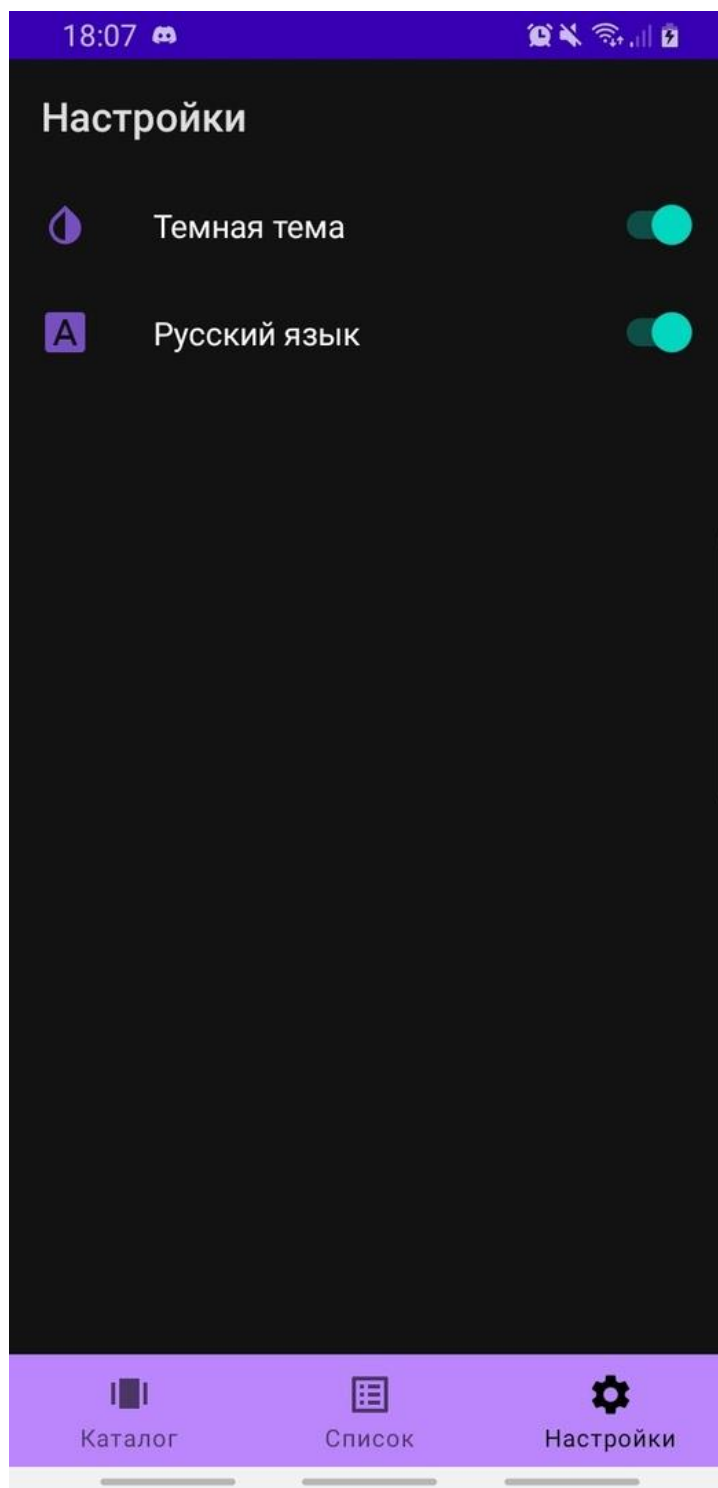


Рисунок 1. 19 – Фрагмент настройками

При выключении русского языка, интерфейс станет отображаться на английском. Скриншот интерфейса на английском языке представлен на рисунке 1.20.

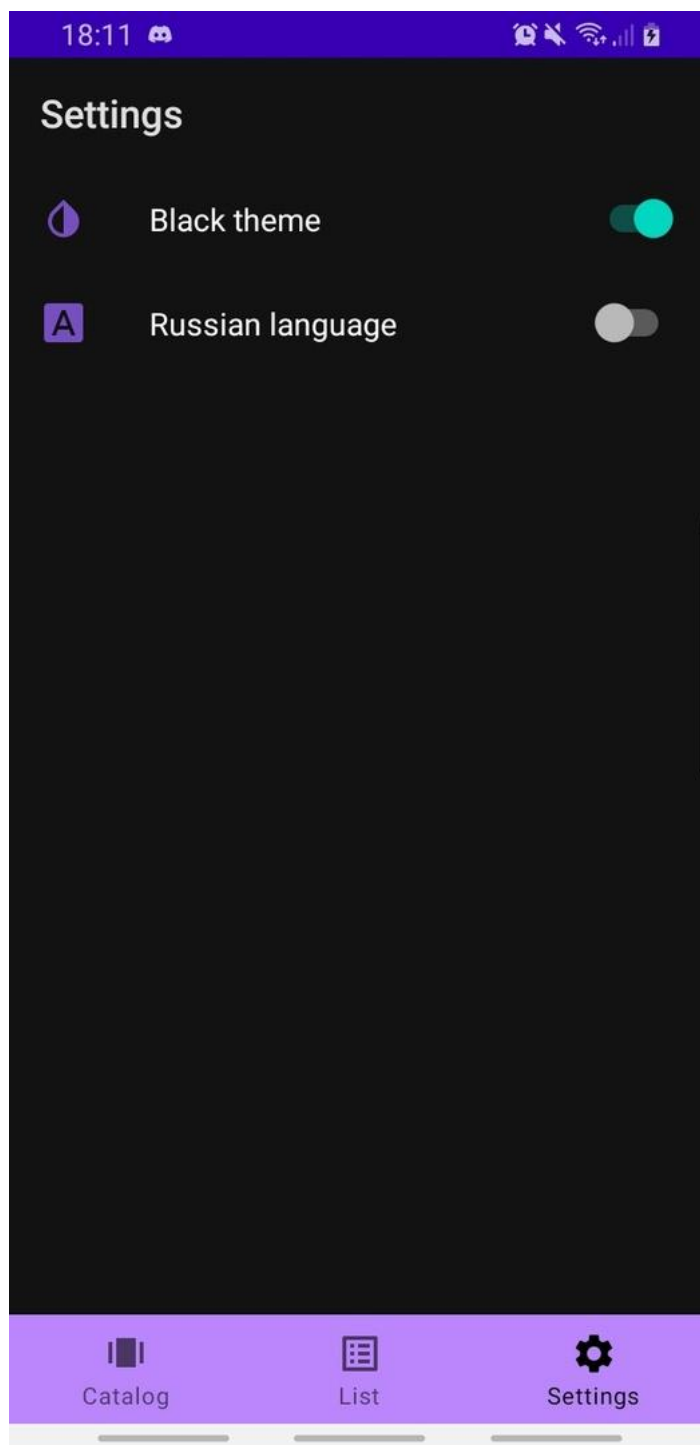


Рисунок 1. 20 – Интерфейс на английском языке

2 Безопасность жизнедеятельности

2.1 Характеристика опасных факторов при работе с ПВЭМ

При разработке приложения необходимо следить за безопасностью жизнедеятельности человека. Должны соблюдаться все комфортные условия при работе с ЭВМ.

Чтобы обеспечить комфортны условия для работы, нужно правильно оценить опасные факторы, которые могут повлиять на разработку. К этим факторам можно отнести шум, высока физическая нагрузка, зрительная нагрузка и другие сочетания вредных факторов.

При долгой работе за компьютером, могут возникать такие типы факторов:

- химические;
- физические;
- психологические.

Чаще всего на человека влияют физические и психологические факторы.

К физическим факторам можно отнести:

- повышенная температура воздуха, где работает человек;
- чрезмерная запыленность воздуха, где работает человек;
- пониженная температура воздуха, где работает человек;
- повышенная и пониженная влажность воздуха;
- превышающий уровень нормы шум;
- малая освещенность рабочего места;
- повышенный уровень ионизирующего излучения;
- опасность поражения электрическим током;
- повышенная уровень яркости света и монитора.

К психологическим факторам, которые могут воздействовать на человека при работе, можно отнести следующие:

- напряжение зрения по время долгой работы с компьютером;
- умственное напряжение;
- эмоциональные нагрузки человека;
- перенапряжение мышц;
- высокие интеллектуальные нагрузки.

2.2 Классификация и категорирование помещения

Кабинеты, в которых работает программист, относятся к пожароопасной категории, которые не предусматривают возможность возникновения пожара В1-В4.

Работа программиста относится к категории работ *IA* – легкие физические работы. Поэтому должны соблюдаться такие требования:

Температура на рабочем месте должна быть в пределах от 22 - 25 °С.

Оптимальная влажность воздуха должна быть 40-60%.

Для поддержания, на рабочем месте, оптимального уровня температуры и влажности воздуха, в холодное время года используется водяное отопление, а в теплое время года кондиционер.

Также работодатель обязан обеспечить следующее:

- соблюдение правил пожарной безопасности;
- осуществление мер по обеспечению пожарной безопасности;
- надежность работы электроустановок и безопасность их обслуживания;
- обеспечение безопасности и условий труда;
- учет и анализ нарушений в работе электроустановок, несчастных случаев и принятие мер по устранению причин их возникновения;
- поддержка систем и средств противопожарной безопасности.

2.3 Освещение

Работа с ПЭВМ зачастую происходит в помещениях с недостаточным освещением, что может, приводит к напряжению зрения, утомляемости, ослаблению внимания. Для освещения помещения должно применяться искусственное освещение совместно с естественным.

Естественное освещение помещений должно осуществляться через световые проемы, ориентированные преимущественно с северной стороны. По отношению к световым проемам должны располагаться так, чтобы естественный свет падал сбоку, преимущественно слева. Если солнечный свет падает с южных или западных окон, необходимо предусматривать средства регулирования интенсивности естественного освещения. Оконные проемы в помещениях, предназначенных для использования компьютеров, необходимо оборудовать устройствами регулирования интенсивности естественного освещения типа жалюзи, занавесей, внешних козырьков. Искусственное освещение в помещениях должно осуществляться системой общего равномерного освещения.

В производственных и административно-общественных помещениях, в случаях преимущественной работы с документами, следует применять системы комбинированного освещения. При искусственном освещении следует использовать специальные люминесцентные лампы, которые излучают свет различного «качества», имитируя, таким образом, полный спектр естественного солнечного света. Они меньше раздражают, чем любые другие лампы искусственного освещения.

Источники света необходимо равномерно распределять по комнате, komponуя в сплошные или прерывистые линии. Линии должны располагаться сбоку от рабочих мест параллельно линии зрения пользователя, если при этом компьютеры размещены в ряд и локализовано над рабочим столом, когда рабочие места размещены по периметру помещения.

2.4 Расчет освещения рабочего места оператора ПЭВМ

					<i>ПГУ 1.090301.10.001 ПЗ</i>	Лист
						43
Изм.	Лист	№ докум	Подпись	Дата		

Освещение важная часть при работе с ПЭВМ. Правильный выбор и расчет освещенности рабочего места обеспечивает создание нормальных условий для зрения обслуживающего персонала, способствует повышению производительности труда. Работа при недостаточном освещении или его отсутствии может негативно сказываться на зрении человека.

Чтобы рассчитать какое необходимое количество светильников должно быть в помещении, будет использована следующая формула

$$N = E_{\text{н}} * S * K_{\text{з}} * Z / (n * \Phi),$$

где n – является коэффициентом использования светового потока

$K_{\text{з}}$ – коэффициент запаса, значение берем равным 1.4 для сухих помещений;

Z – коэффициент неравномерности освещения, для освещения взяты люминесцентные коэффициент неравномерности, которых лампы, равен 1.1;

S – площадь помещения;

$E_{\text{н}}$ – минимальная освещенность, значение которой равное 300 лк;

Φ – световой поток.

Площадь помещения при длине $A = 6$ м, $B = 6$ м, рассчитаем по формуле:

$$S = A * B = 6 * 6 = 36 \text{ м}^2.$$

Расчёт индекса помещения найдем по формуле:

$$i = S / (h_{\text{р}} * (A + B)),$$

где $h_{\text{р}}$ – высота светильников над рабочей поверхностью, в данном случае это значение равно 2 м. A ширина помещения, а B , соответственно, длинна помещения. Найдем значение коэффициента светового потока:

$$i = 36 / (2 * (6 + 6)) = 1,5.$$

Значения взятые для коэффициента отражения потолка R_p и стен R_c равны 70% и 50%, соответственно, так как являются светлыми поверхностями. Исходя из этих данных находим коэффициент использования светового потока $n = 0,64$.

Для освещения выбираем лампы ЛБ40, световой поток которых равен $\Phi = 2800$ лм.

Имея все необходимые данные, можем подсчитать количество необходимых светильников для ламп ЛПО с двумя лампами:

$$N = 300 * 36 * 1.4 * 1.1 / (0,64 * 2800 * 2) = 5.$$

Таким образом, необходимое количество светильников равно 5.

Заключение

В результате выполнения работы было разработано клиент-серверное приложение, с помощью которого пользователи могут сканировать QR и штрих коды товаров и находят по ним информацию в магазинах, по отсканированному штрих коду пользователь может узнать цену, название и магазин этого товара. Также пользователи может узнать об акциях, которые проводят магазины. При необходимости может вписать товар в список покупок перед походом в магазин.

В ходе разработки приложения, был изучен принцип *REST API*. Вместе с этим, были улучшены навыки работы с языками программирования *Kotlin* [2] и *Python*, а также навыки работы со средой *Android Studio*. Были изучены принципы сканирования *Qr* и штрих кодов товаров.

В дальнейшем разработанное приложение можно улучшить, добавив больше магазинов для поиска по ним товаров, добавить авторизацию пользователя, для сохранения данных которые

Список использованных источников

1. Дмитрий Джемеров и Светлана Исакова: Kotlin в действии. – Февраль 2017 г. – 246 с.
2. Venkat Subramaniam. Programming Kotlin. – Сентябрь 2019. – 316 с.
3. Miguel Grinberg: разработка веб-приложений с использованием Flask на языке Python. - ДМК Пресс, 2018г. – 126 с.
4. Пол Дейтел, Харви Дейтел, Александер Уолд: Android для разработчиков. – Питер СПб. –2016 г. – 249 с.
5. Retrofit 2.0 официальная документация [Электронный ресурс]URL: <https://russianblogs.com/article/7286717114/>
6. CameraX [Электронный ресурс]
URL:<https://developer.android.com/training/cameras>

Приложение А

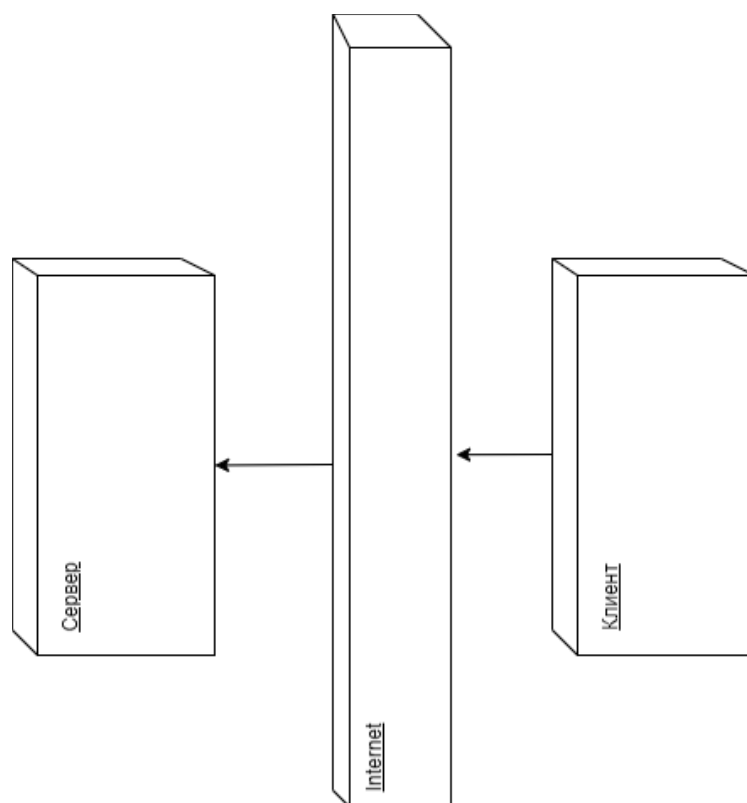
(Обязательное)

Программное обеспечение проверки QR кодов товаров

UML-диаграммы

					ПГУ 1.090301.10.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		48

Приложение А.1 – Диаграмма развертывания

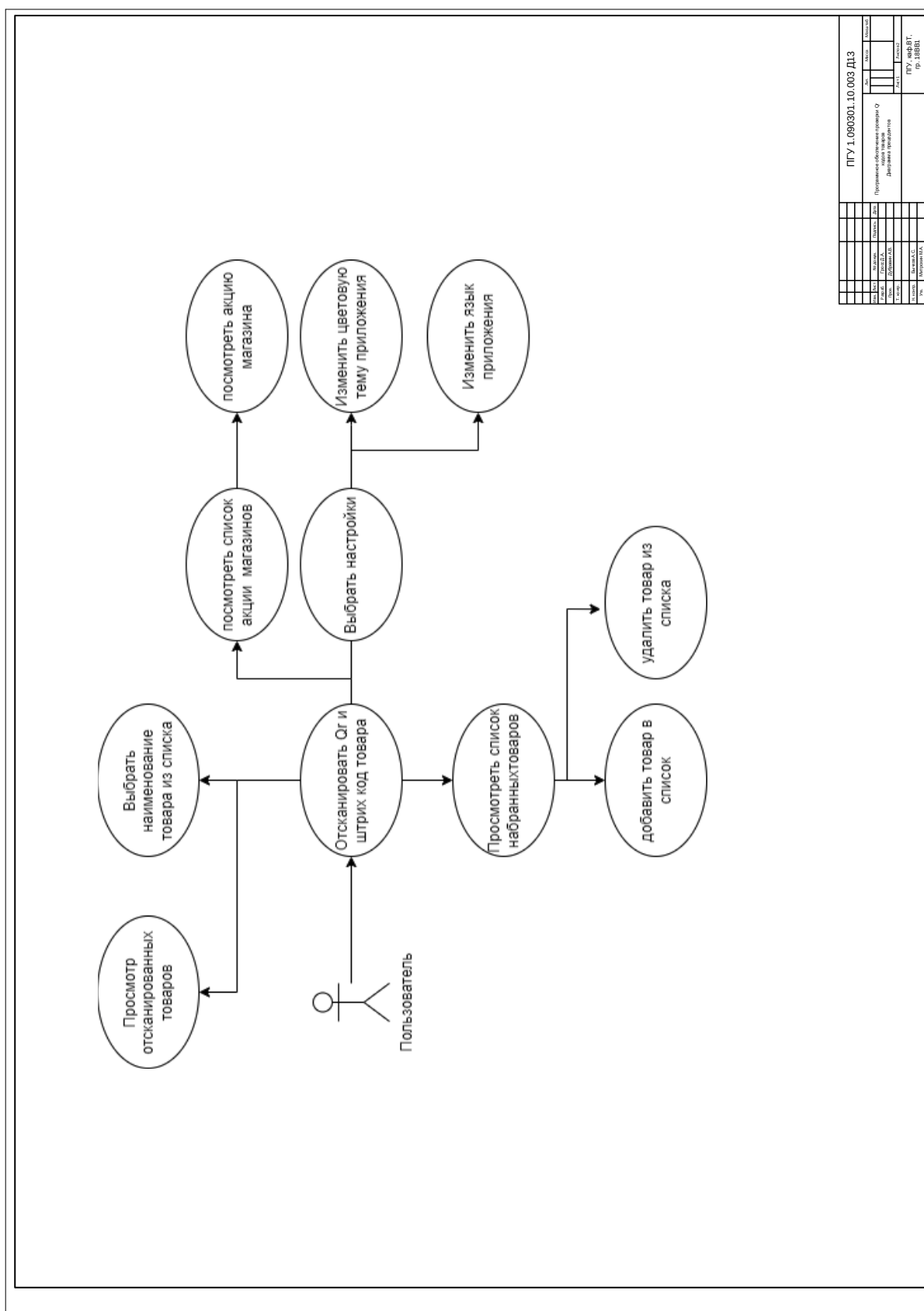
[illegible]

Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

Лист

Приложение А.2 – Диаграмма прецедентов



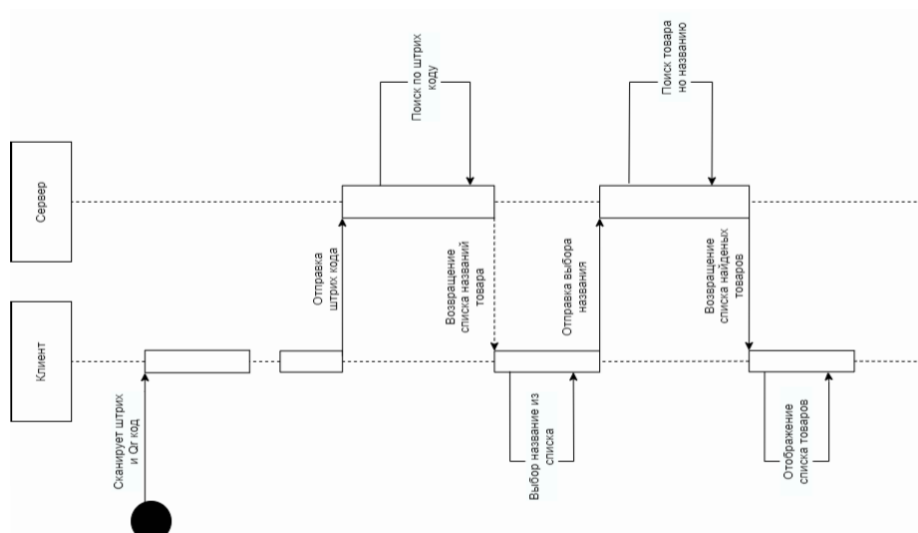
Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

Лист

50

Приложение А.3 – Диаграмма последовательностей



ПГУ 1.090301.10.004 ДЦ4									
Программное обеспечение					Получено				
Диаграмма последовательности					ПГУ, май 2017 г.				
					р. 188ВВ1				

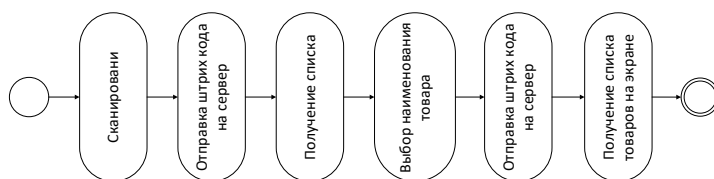
Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

Лист

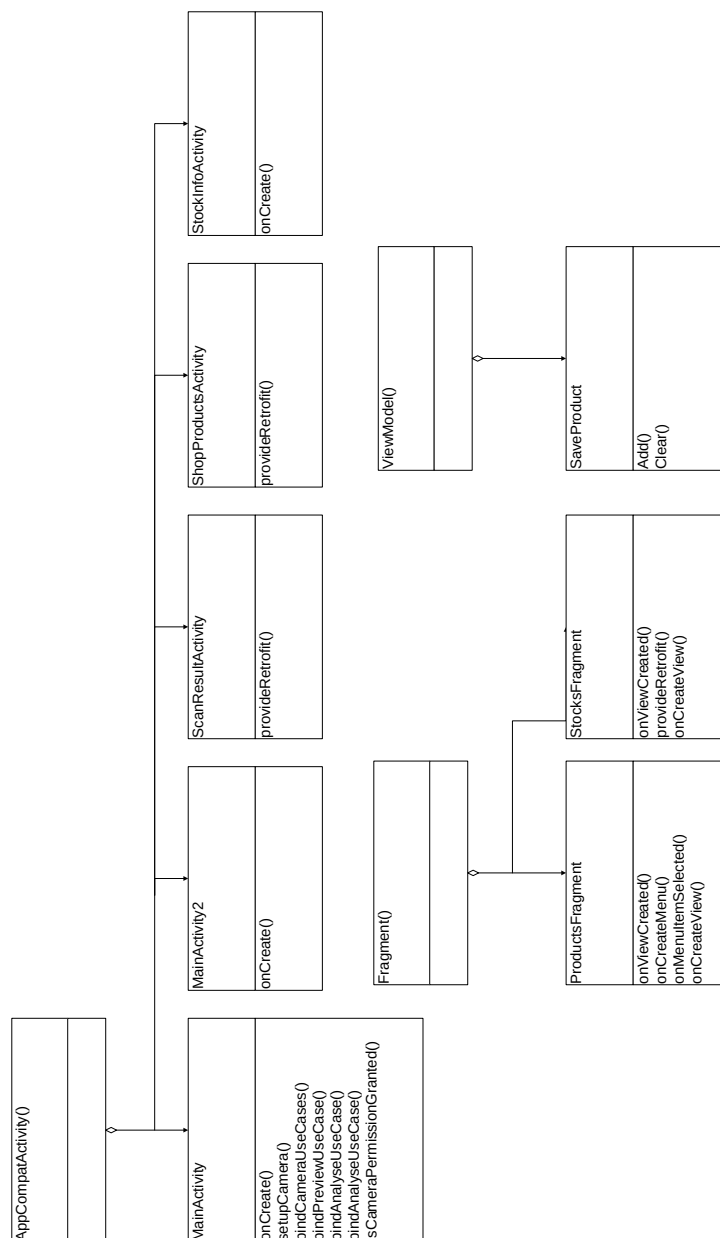
51

Приложение А.4 – Диаграмма деятельности



										ПГУ 1.090301.10.005 Д17				
				</										

Приложение А.5 – Диаграмма классов

[illegible]

Приложение Б

(Обязательное)

Программное обеспечение проверки QR кодов товаров

Листинг программы

					ПГУ 1.090301.10.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		54

MainActivity.kt

```
package com.clctech.barcodescannerexp

import android.Manifest

import android.annotation.SuppressLint

import android.content.Intent

import android.content.pm.PackageManager

import androidx.appcompat.app.AppCompatActivity

import kotlinx.android.synthetic.main.activity_main.*

import android.os.Bundle

import android.util.Log

import androidx.core.app.ActivityCompat

import androidx.core.content.ContextCompat

import

com.google.mlkit.vision.barcode.BarcodeScannerOptions

import androidx.camera.core.*

import androidx.camera.lifecycle.ProcessCameraProvider

import androidx.camera.view.PreviewView

import androidx.lifecycle.ViewModelProvider

import com.google.mlkit.vision.barcode.BarcodeScanner

import com.google.mlkit.vision.common.InputImage

import java.util.concurrent.Executors

import kotlin.IllegalStateException
```

```

import com.google.mlkit.vision.barcode.BarcodeScanning

import com.google.mlkit.vision.barcode.common.Barcode

class MainActivity : AppCompatActivity() {

    private var previewView: PreviewView? = null

    private var cameraProvider: ProcessCameraProvider? =
null

    private var cameraSelector: CameraSelector? = null

    private var lensFacing =
CameraSelector.LENS_FACING_BACK

    private var previewUseCase: Preview? = null

    private var analysisUseCase: ImageAnalysis? = null

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity_main)

        setupCamera()

    }

    private fun setupCamera() {

        previewView = findViewById(R.id.preview_view)

```

Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

Лист

56

```

        cameraSelector =
CameraSelector.Builder().requireLensFacing(lensFacing).build()

        ViewModelProvider(

            this,
ViewModelProvider.AndroidViewModelFactory.getInstance(application)

        ).get(CameraXViewModel::class.java)

            .processCameraProvider

            .observe(this) { provider:
ProcessCameraProvider? ->

                cameraProvider = provider

                if (isCameraPermissionGranted()) {

                    bindCameraUseCases()

                } else {

                    ActivityCompat.requestPermissions(

                        this,

arrayOf(Manifest.permission.CAMERA),

                            PERMISSION_CAMERA_REQUEST

                        )

                    }

                }

            }

```

```
private fun bindCameraUseCases() {

    bindPreviewUseCase()

    bindAnalyseUseCase()

}
```

```
private fun bindPreviewUseCase() {

    if (cameraProvider == null) {

        return

    }

    if (previewUseCase != null) {

        cameraProvider!!.unbind(previewUseCase)

    }


```

```
    previewUseCase = Preview.Builder()
```

```
    .setTargetRotation(previewView!!.display.rotation)
```

```
    .build()
```

```
    previewUseCase!!.setSurfaceProvider(previewView!!.surfacePr
    ovider)
```

```
    try {
```

```

        cameraProvider!!.bindToLifecycle(

            this,

            cameraSelector!!,

            previewUseCase

        )

    } catch (illegalStateException:
IllegalStateException) {

        Log.e(TAG, illegalStateException.message ?:
"IllegalStateException")

    } catch (illegalArgumentException:
IllegalArgumentException) {

        Log.e(TAG, illegalArgumentException.message ?:
"IllegalArgumentException")

    }

}

private fun bindAnalyseUseCase() {

    val options = BarcodeScannerOptions.Builder()

        .setBarcodeFormats(Barcode.FORMAT_ALL_FORMATS).build()

    val barcodeScanner: BarcodeScanner =
BarcodeScanning.getClient(options)

```

```

        if (cameraProvider == null) {

            return

        }

        if (analysisUseCase != null) {

            cameraProvider!!.unbind(analysisUseCase)

        }

        analysisUseCase = ImageAnalysis.Builder()

        .setTargetRotation(previewView!!.display.rotation)

        .build()

        // Initialize our background executor

        val cameraExecutor =
        Executors.newSingleThreadExecutor()

        analysisUseCase?.setAnalyzer(

            cameraExecutor,

            ImageAnalysis.Analyzer { imageProxy ->

                processImageProxy(barcodeScanner,
                imageProxy)

            }

```

```

        )

        try {

            cameraProvider!!.bindToLifecycle(

                /* lifecycleOwner= */this,

                cameraSelector!!,

                analysisUseCase

            )

        } catch (illegalStateException:
IllegalStateException) {

            Log.e(TAG, illegalStateException.message ?:
"IllegalStateException")

        } catch (illegalArgumentException:
IllegalArgumentException) {

            Log.e(TAG, illegalArgumentException.message ?:
"IllegalArgumentException")

        }

    }

}

@SuppressLint("UnsafeOptInUsageError")

private fun processImageProxy(

    barcodeScanner: BarcodeScanner,

    imageProxy: ImageProxy

) {

```

```

        val inputImage =

            InputImage.fromMediaImage(imageProxy.image!!,
            imageProxy.imageInfo.rotationDegrees)

        barcodeScanner.process(inputImage)

            .addOnSuccessListener { barcodes ->

                barcodes.forEach { barcode ->

                    val bounds = barcode.boundingBox

                    val corners = barcode.cornerPoints

                    val rawValue = barcode.rawValue

                    tvScannedData.text = barcode.rawValue

                    val valueType = barcode.valueType

                    Log.d(TAG, "тип данных: ${valueType}")

                    // See API reference for complete list
of supported types

                    when (valueType) {

                        Barcode.TYPE_PRODUCT ->{

                            tvScannedData.text = "штрих
код; ${barcode.rawValue}"

                            val intent = Intent(this,
                            ScanResultActivity::class.java)

```

Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

Лист

62

```

intent.putExtra(ScanResultActivity.EXTRA_BARCODE,
barcode.rawValue)

                startActivity(intent)

            }

Barcode.TYPE_WIFI -> {

                val ssid = barcode.wifi!!.ssid

                val password =
barcode.wifi!!.password

                val type =
barcode.wifi!!.encryptionType

                tvScannedData.text =

                    "ssid: " + ssid +

"\npassword: " + password + "\ntype: " + type

            }

Barcode.TYPE_URL -> {

                val title = barcode.url!!.title

                val url = barcode.url!!.url

                tvScannedData.text = "Title: "

+ title + "\nURL: " + url

            }

        }

    }

```

Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

Лист

63

```

    }

    .addOnFailureListener {

        Log.e(TAG, it.message ?: it.toString())

    }

    .addOnCompleteListener {

        // When the image is from CameraX analysis
        use case, must call image.close() on received

        // images when finished using them.
        Otherwise, new images may not be received or the camera

        // may stall.

        imageProxy.close()

    }

}

```

```

override fun onRequestPermissionsResult(

    requestCode: Int,

    permissions: Array<String>,

    grantResults: IntArray

) {

    if (requestCode == PERMISSION_CAMERA_REQUEST) {

        if (isCameraPermissionGranted()) {

            bindCameraUseCases()

        }

    }

}

```

```

        } else {

            Log.e(TAG, "no camera permission")

        }

    }

    super.onRequestPermissionsResult(requestCode,
permissions, grantResults)

    }

private fun isCameraPermissionGranted(): Boolean {

    return ContextCompat.checkSelfPermission(

        baseContext,

        Manifest.permission.CAMERA

    ) == PackageManager.PERMISSION_GRANTED

}

companion object {

    private val TAG =
MainActivity::class.java.simpleName

    private const val PERMISSION_CAMERA_REQUEST = 1

}

}

```

Файл MainActivity2.kt

```
package com.clctech.barcodescannerexp

import androidx.appcompat.app.AppCompatActivity

import android.os.Bundle

import android.view.View

import androidx.navigation.findNavController

import androidx.navigation.fragment.NavHostFragment

import androidx.navigation.ui.AppBarConfiguration

import
androidx.navigation.ui.setupActionBarWithNavController

import androidx.navigation.ui.setupWithNavController

import
com.google.android.material.bottomnavigation.BottomNavigationVi
onView

class MainActivity2 : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

        setContentView(R.layout.activity_main2)

        val bottomNavigationView =
findViewById<BottomNavigationView>(R.id.bottomNavigationVie
w)
```

```

        val navController =
            (supportFragmentManager.findFragmentById(R.id.Fragment) as
                NavHostFragment).navController

        val appBarConfiguration =
            AppBarConfiguration(setOf(

                R.id.firstFragment,

                R.id.secondFragment,

                R.id.thirdFragment))

            setupActionBarWithNavController(navController,
            appBarConfiguration)

        bottomNavigationView.setupWithNavController(navController)

    }

}

```

Файл Products.kt

```

package com.clctech.barcodescannerexp

import android.app.AlertDialog
import android.os.Bundle
import android.view.*
import android.widget.Toast
import androidx.core.view.MenuHost
import androidx.core.view.MenuProvider

```

```

import androidx.fragment.app.Fragment

import androidx.fragment.app.viewModels

import androidx.lifecycle.Lifecycle

import androidx.recyclerview.widget.DividerItemDecoration

import androidx.recyclerview.widget.LinearLayoutManager

import
com.clctech.barcodescannerexp.databinding.FragmentProductsB
inding

import kotlinx.android.synthetic.main.product_dialog.view.*

class ProductsFragment : Fragment() {

    private val adapter = TodoAdapter()

    private lateinit var binding: FragmentProductsBinding

    private val viewModel by viewModels<SaveProduct>()

    override fun onCreateView(view: View,
savedInstanceState: Bundle?) {

        super.onCreateView(view, savedInstanceState)

        binding.rvTodos.adapter = adapter

```

```

binding.rvTodos.addItemDecoration(DividerItemDecoration(requireContext(),
    DividerItemDecoration.VERTICAL))

binding.rvTodos.layoutManager =
    LinearLayoutManager(requireContext())

viewModel.list.observe(viewLifecycleOwner) {

    adapter.tipeDate(it)

}

binding.floatingActionButton.setOnClickListener {

    val mDialogView =

        LayoutInflater.from(requireContext()).inflate(R.layout.product_dialog, null)

        val mBuilder =
            AlertDialog.Builder(requireContext())

                .setView(mDialogView)

                .setTitle("Введите продукт")

        val mAlertDialog = mBuilder.show()

        mDialogView.dialogLogOk.setOnClickListener {

            mAlertDialog.dismiss()

            val name =
                mDialogView.dialog_ProductEm.text.toString()

```

```

        val todo = Todo(name, false)

        viewModel.add(todo)

    }

}

val menuHost: MenuHost = requireActivity()

menuHost.addMenuProvider(object : MenuProvider {

    override fun onCreateMenu(menu: Menu,
menuInflater: MenuInflater) {

        // Add menu items here

        menuInflater.inflate(R.menu.menu_products,
menu)

    }

    override fun onMenuItemSelected(menuItem:
MenuItem): Boolean {

        // Handle the menu selection

        return when (menuItem.itemId) {

            R.id.menu_delete -> {

                viewModel.clear()

                Toast.makeText(requireContext(),
"удалил", Toast.LENGTH_SHORT).show()

                true

```

```

        }

        else -> false

    }

}

}, viewLifecycleOwner, Lifecycle.State.RESUMED)

}

override fun onCreateView(

    inflater: LayoutInflater, container: ViewGroup?,

    savedInstanceState: Bundle?

): View? {

    binding = FragmentProductsBinding.inflate(inflater)

    return binding.root

}

}

```

StocksFragmen.kt

```

package com.clctech.barcodescannerexp

import android.content.Intent

import android.os.Bundle

import androidx.fragment.app.Fragment

import android.view.LayoutInflater

import android.view.View

```

Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

Лист

71

```

import android.view.ViewGroup

import androidx.navigation.findNavController

import androidx.recyclerview.widget.LinearLayoutManager

import com.google.gson.GsonBuilder

import kotlinx.android.synthetic.main.fragment_stocks.*

import kotlinx.coroutines.Dispatchers

import kotlinx.coroutines.GlobalScope

import kotlinx.coroutines.launch

import kotlinx.coroutines.withContext

import okhttp3.OkHttpClient

import okhttp3.Request

import retrofit2.Retrofit

import retrofit2.converter.gson.GsonConverterFactory

import retrofit2.converter.scalars.ScalarsConverterFactory

import java.util.concurrent.TimeUnit

class StocksFragment : Fragment() {

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)

```

```

    }

    override fun onCreateView(view: View,
savedInstanceState: Bundle?) {

        super.onCreateView(view, savedInstanceState)

        GlobalScope.launch(Dispatchers.IO) {

            val retrofit = provideRetrofit()

            var service: BarcodeAPI =
retrofit.create(BarcodeAPI::class.java)

            val result = service.getStockList()

            withContext(Dispatchers.Main) {

                stocks.layoutManager =

                    LinearLayoutManager(requireContext())

                stocks.adapter =
StockAdapter(result.stocks.toMutableList()).apply {

                    itemClick = {

                        val intent =
Intent(requireContext(), StockInfoActivity::class.java)

                        intent.putExtra(StockInfoActivity.EXTRA_STOCK, it)

                        startActivity(intent)

                    }

                }

            }

        }
    }

```

```

        }

    }

    button.setOnClickListener{view : View ->

view.findNavController().navigate(R.id.action_firstFragment
_to_mainActivity)

    }

}

fun provideRetrofit(): Retrofit {

    val gson = GsonBuilder().setLenient().create()

    val httpClient =
OkHttpClient.Builder().retryOnConnectionFailure(true)

    httpClient.connectTimeout(60, TimeUnit.SECONDS)

    httpClient.readTimeout(60, TimeUnit.SECONDS)

    httpClient.addInterceptor { chain ->

        val original: Request = chain.request()

        val request: Request =

            original.newBuilder().header("Connection",
"close").header("Accept", "image/png")

```

```

        .header("Content-Type",
"image/jpg").header("Content-Type",
"application/json").header("Accept-Encoding", "identity")

        .method(original.method(),
original.body()).build()

        chain.proceed(request)

    }

    return
Retrofit.Builder().baseUrl("http://192.168.0.105:5555/")

.addConverterFactory(GsonConverterFactory.create(gson))

.addConverterFactory(ScalarsConverterFactory.create()).client(
httpClient.build()).build()

}

override fun onCreateView(

    inflater: LayoutInflater, container: ViewGroup?,

    savedInstanceState: Bundle?

): View? {

    // Inflate the layout for this fragment

    return inflater.inflate(R.layout.fragment_stocks,

container, false)

}

}

```

StockInfoActivity.kt

```
package com.clctech.barcodescannerexp

import android.os.Bundle

import androidx.appcompat.app.AppCompatActivity

import com.clctech.barcodescannerexp.databinding.ActivityShopProductsBinding

import com.clctech.barcodescannerexp.databinding.ActivityStockInfoBinding

import com.squareup.picasso.Picasso

class StockInfoActivity: AppCompatActivity() {

    private lateinit var binding: ActivityStockInfoBinding

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)

        binding =
            ActivityStockInfoBinding.inflate(layoutInflater)

        setContentView(binding.root)

        val parcelableExtra =
            intent.getParcelableExtra<BarcodeAPI.StockResponse>(EXTRA_STOCK)!!

        binding.textView2.text =
            parcelableExtra.description
    }
}
```

```

Picasso.get().load(parcelableExtra.url).into(binding.imageView3)

    }

    companion object {

        const val EXTRA_STOCK = "EXTRA_STOCK"

    }

}

```

StockAdapter.kt

```

package com.clctech.barcodescannerexp

import android.view.LayoutInflater

import android.view.View

import android.view.ViewGroup

import androidx.recyclerview.widget.RecyclerView

import
com.clctech.barcodescannerexp.databinding.ItemBarcodeResult
Binding

import
com.clctech.barcodescannerexp.databinding.ItemStockBinding

import com.squareup.picasso.Picasso

class StockAdapter (private val stockList:
MutableList<BarcodeAPI.StockResponse>) :
RecyclerView.Adapter<StockAdapter.StockViewHolder>() {

```

```

        var itemClick: ((item: BarcodeAPI.StockResponse) ->
Unit) ? = null

        inner class StockViewHolder(private val binding:
ItemStockBinding) :

            RecyclerView.ViewHolder(binding.root) {

                fun bind(item: BarcodeAPI.StockResponse) {

                    binding.root.setOnClickListener {

                        itemClick?.invoke(item)

                    }

                    binding.textView.text = item.title

                    Picasso.get().load(item.url).into(binding.imageView2)

                }

            }

        override fun onCreateViewHolder(parent: ViewGroup,
viewType: Int): StockViewHolder {

            val binding =
ItemStockBinding.inflate(LayoutInflater.from(parent.context
), parent ,false)

            return StockViewHolder(binding)

        }

        override fun onBindViewHolder(holder: StockViewHolder,
position: Int) {

```

```

        holder.bind(stockList[position])

    }

    override fun getItemCount(): Int {

        return stockList.size

    }

}

```

BarcodeAPI

```

package com.clctech.barcodescannerexp

import android.os.Parcelable

import kotlinx.android.parcel.Parcelize

import retrofit2.http.GET

import retrofit2.http.Path

import retrofit2.http.Query

interface BarcodeAPI {

    @GET("productTitle")

    suspend fun productTitleList(@Query("barcode") barcode:
String): BarcodeResponse

    @Parcelize

```

```

        data class BarcodeResponse(val productList:
List<String>) : Parcelable

        @GET("similarProducts")

        suspend fun searchSimilar(@Query("title") title:
String): SimilarResponse

        @Parcelize

        data class ProductResponse(

            val title: String,

            val rub: String,

            val peny: String,

            val shop: String

        ) : Parcelable

        @Parcelize

        data class SimilarResponse(val similar:
List<ProductResponse>) : Parcelable

        @Parcelize

        data class StockResponse(

            val url: String,

            val title: String,

```

```

        val description: String

    ) : Parcelable

    @Parcelize

    data class StocksResponse(val stocks:
List<StockResponse>): Parcelable


    @GET("stockList")

    suspend fun getStockList(): StocksResponse

}

```

Приложение В

(Обязательное)

Программное обеспечение проверки QR кодов товаров

Презентация

					ПГУ 1.090301.10.001 ПЗ	Лист
Изм.	Лист	№ докум	Подпись	Дата		82

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРОВЕРКИ QR-КОДОВ ТОВАРОВ



Выполнил: студент группы 18ВВ1

Гросс Д.А.

Научный руководитель: Дубравин А.В.

Цели и задачи проекта

Цель дипломной работы - разработка приложения для сканирования QR и штрих кода товара.

Задачи

- Сканирование QR и штрих кода товара
- Вывод информации о товаре с интернет-магазинов
- Создание списка покупок

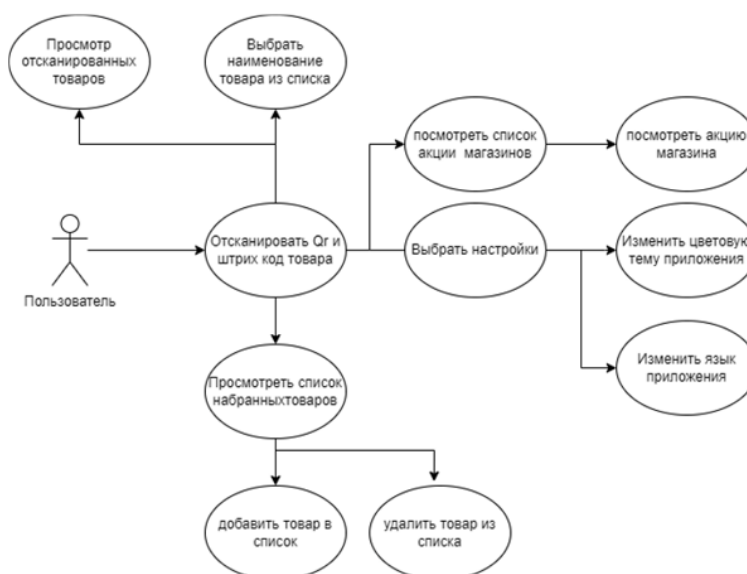


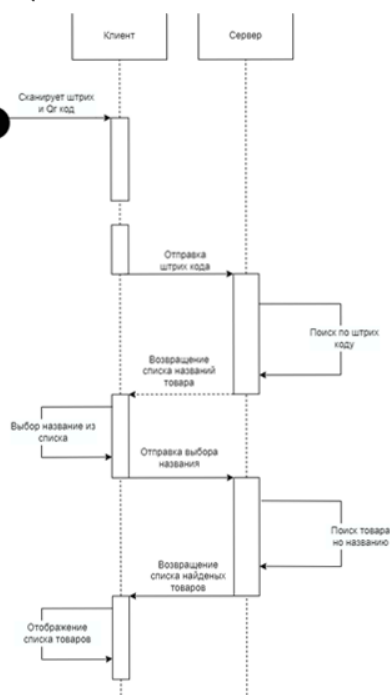
Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

Лист

83





Просмотр акций



Сканирование QR кода



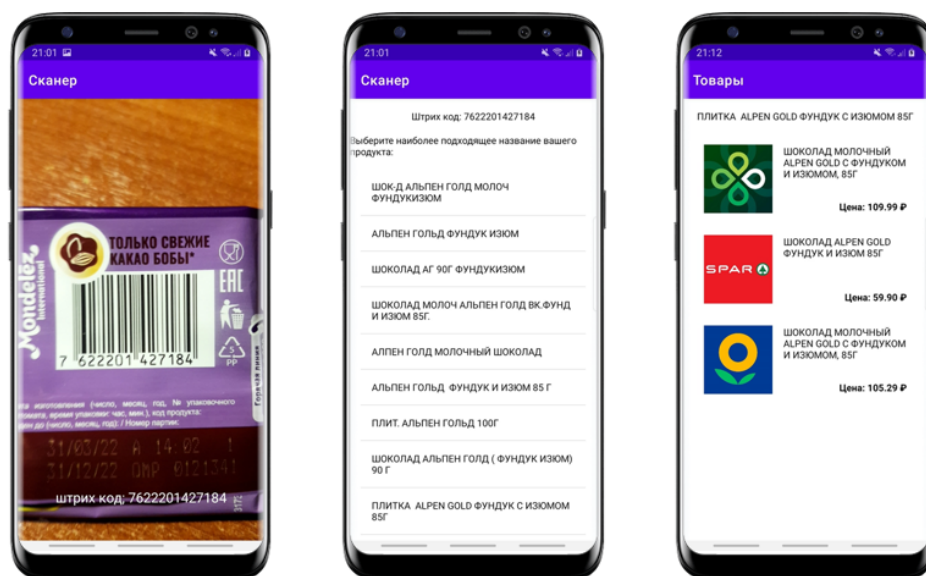
Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

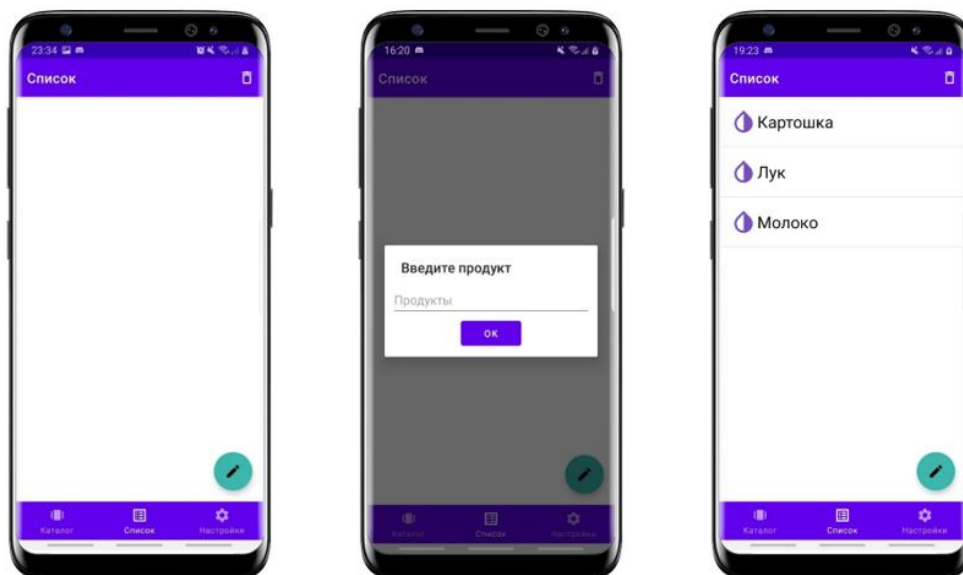
Лист

85

Сканирование штрих кода



Создание списка покупок



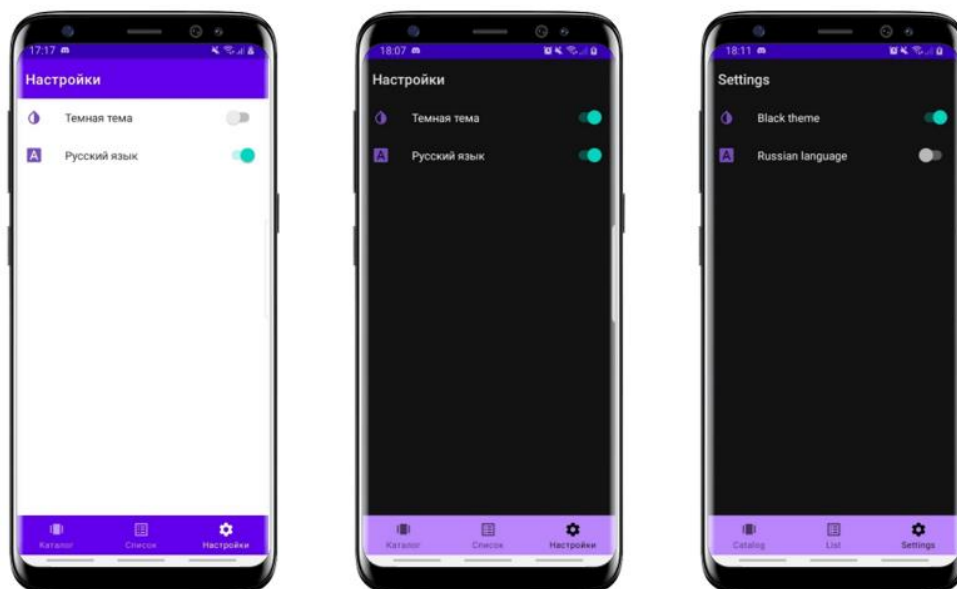
Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

Лист

86

Окно настроек приложения



Заключение

- Разработанное программное обеспечение обладает приятным и интуитивно понятным для пользователя графическим интерфейсом
- Приложение позволяет пользователю сканировать QR и штрих коды товаров, и выводить информацию о товаре из различных магазинов
- Создание списка покупок

Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

Лист

87

СПАСИБО ЗА ВНИМАНИЕ



Изм.	Лист	№ докум	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

Лист