МИНОБРНАУКИ РОССИИ

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

**Институт компьютерных наук и технологий**

**Высшая инженерная школа**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**Реализация прототипа мобильного приложения «Cписок покупок»**

**на базе операционной системы Android**

по программе профессиональной переподготовки:

«Разработчик прикладного программного обеспечения (Язык Java)»

Выполнил:

Ленский Илья Ильич

Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель:

ст. преподаватель ВШПИ

Маслаков Алексей Павлович

Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2022

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc96190420)

[1. АНАЛИТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ПРОЕКТИРОВАНИЯ 4](#_Toc96190421)

[1.1 Понятие о мобильном приложении 4](#_Toc96190422)

[1.2 Преимущества использования 6](#_Toc96190423)

[1.3 Цель и задачи 8](#_Toc96190424)

[1.4 Новизна и практическая значимость 8](#_Toc96190425)

[1.5 Выводы по разделу 9](#_Toc96190426)

[2. ТРЕБОВАНИЯ К ПРИЛОЖЕНИЮ 10](#_Toc96190427)

[2.1 Нефункциональные требования 10](#_Toc96190428)

[2.2 Функциональные требования 12](#_Toc96190429)

[2.3 Выводы по разделу 14](#_Toc96190430)

[3. ПРОЕКТИРВОВАНИЕ ПРИЛОЖЕНИЯ 15](#_Toc96190431)

[3.1 Определение пользователи 15](#_Toc96190432)

[3.2 Архитектура 17](#_Toc96190433)

[3.3 Диаграмма обмена данными и классов 21](#_Toc96190434)

[3.4 Проектирование базы данных 23](#_Toc96190435)

[3.5 Выводы по разделу 25](#_Toc96190436)

[4. РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ 26](#_Toc96190437)

[4.1 Выбор технологий 26](#_Toc96190438)

[4.2 Выбор системы управления баз данных 28](#_Toc96190439)

[4.3 Реализация приложения 29](#_Toc96190440)

[4.4 Выводы по разделу 33](#_Toc96190441)

[5. ТЕСТИРОВАНИЕ 34](#_Toc96190442)

[5.1 Функциональное тестирование 35](#_Toc96190443)

[5.2 Выводы по разделу 37](#_Toc96190444)

[ЗАКЛЮЧЕНИЕ 38](#_Toc96190445)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 39](#_Toc96190446)

[ПРИЛОЖЕНИЕ А. Фрагменты исходного кода 41](#_Toc96190447)

# **ВВЕДЕНИЕ**

В настоящее время мобильные устройства играют большую роль в жизни человека. На текущий момент рынок мобильных устройств увеличился в несколько раз по сравнению с началом 21 века [1].

Смартфоны и планшеты стали выполнять большой спектр задач с помощью специально разработанных мобильных приложений, спрос на которые значительно вырос [2]. Один из косвенных критериев, подтверждающих данный факт, это рост мобильного интернет-трафика в разы по сравнению с прошлыми годами [3], тем самым пользователи все больше и больше используют мобильные устройства для работы и отдыха.

Все приложения условно можно разделить на те, которые используются для приятного времяпрепровождения, и те, которые предоставляют удобство использования в работе или учебе. Первая категория в основном включает игры, средства для обработки фото и видеоматериалов, приложения для общения. Вторая направлена на решение конкретных задач, предоставляющих необходимый функционал пользователю в интересующих его сферах деятельности.

Целью данной работы является разработка прототипа мобильного приложения «ShoppingList» (список покупок), на языке Java под управлением операционной системой (ОС) Android.

Для достижения цели необходимо решить следующие задачи:

1. изучить предметную область;
2. разработать требования;
3. спроектировать архитектуру;
4. реализовать;
5. протестировать.

# **1. АНАЛИТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ПРОЕКТИРОВАНИЯ**

## **1.1 Понятие о мобильном приложении**

Буквально 15-20 лет назад на вопрос, что такое мобильное приложение, владелец сотового телефона не нашел бы ответа. Возможности устройств ограничивались функциями звонков, отправки СМС и простейшими играми. Впрочем, незатейливую игру на сотовом можно было уже тогда называть приложением.

Предварительный национальный стандарт Российской Федерации дает вот такое определение мобильному приложению – это программное обеспечение, предназначенное для работы на смартфонах, планшетах и других мобильных устройствах [4]. Также можно встретить другое определение – это программное обеспечение, применяемое физическими лицами с использованием технического устройства (мобильного телефона, смартфона или компьютера, включая планшетный компьютер), подключенного к информационно-телекоммуникационной сети Интернет, для доступа к инвестиционной платформе [5].

Сегодня, мобильное приложение – это специально разработанное под функциональные возможности гаджетов программное обеспечение. Назначение ПО может быть самым разнообразным: сервисы, магазины, развлечения, онлайн-помощники и другое. Эти приложения скачиваются и устанавливаются самим пользователем через мобильные маркетплейсы. Технически все приложения создаются под конкретную платформу мобильного гаджета. Наиболее популярные операционные системы – Android, iOS.

Мобильное приложение – это программный пакет, функционал и дизайн которого адаптирован под возможности мобильных платформ. Перечислим несколько основных плюсов приложения:

* Интерфейс программы изначально создан под работу на мобильном устройстве через сенсорный экран или кнопки;
* Удобная и понятная для пользователей гаджетов навигация, мобильное меню;
* Лучшее взаимодействие с пользователем через сообщения, пуш-уведомления, напоминания. Приложение может выполнять функции даже в фоновом режиме, чего нельзя сказать о сайте. Для работы с программой не нужно открывать браузер, а многие приложения поддерживают ряд функций и при отключенном интернете;
* Хранение персональных данных пользователя. Эта функция расширяет возможности персонализации приложений. Например, вызывает такси на дом (прописка), записывает на прием к врачу по медицинскому полису и другие преимущества;
* Более гибкая обратная связь с компанией, сервисом;
* Можно задействовать больше ресурсов. Например, подключить геолокацию и вызывать машину в любую точку города;
* Приложения могут учитывать биологические ритмы человека и оповещать его о необходимости следовать режиму.

Мобильные приложения можно разделить на три типа:

1. Мобильные веб-приложения и сайты. Как уже писали выше у таких решений есть несколько плюсов – это кроссплатформенность, простота создания и обновления. Минус в низкой функциональности. Это неплохой вариант для старта, чтобы проанализировать мобильный трафик в бизнес-нише, однако данное решение имеет ограниченный функционал.
2. Гибридные приложения – это уже более современный вариант, который работает на API. В программах уже есть push-уведомления, приложение может размещаться в плейсмаркетах для свободного или платного скачивания. Такие программные решения имеют возможность независимого обновления, что снимает необходимость выпуска новых версий.
3. Собственные (нативные) приложения – максимально оптимизированные, которые дают максимальную функциональность и скорость взаимодействия. Однако для их реализации и поддержания стабильной работы требуются серьезные ресурсы.

Мобильное приложение можно разделить на два больших блока – это front- и back-end. Соответственно в часть front-end входят компоненты и опции программы, с которой взаимодействует пользователь. Например, панель выбора, дашборд, настройки опций и прочее. Back-end – это скрытая часть, «задник». С этими компонентами взаимодействует разработчик посредством серверного софта.

Иными словами, мобильное приложение напоминает сплит-систему, в которой одна часть находится на стороне пользователя – это Front-end, а другая на стороне разработчика – это Back-end.

Пользователи мобильной сети загружают необходимые приложения из магазинов Google Play и App Store. Компании на стороне Back-end получают массу преимуществ от такого формата взаимодействия. Например, анализируют информацию о целевой аудитории, быстрее доводят политику лояльности пользователям, повышают уровень продаж с мобильного направления [6].

## **1.2 Преимущества использования**

Успешный бизнес должен идти в ногу с современными технологиями иначе он останется в прошлом и придет в упадок. Какие перспективы открывают перед бизнесом мобильные приложения:

1. Рост продаж. Получать прибыль с новых каналов привлечения клиентов – это основная коммерческая цель любого бизнеса. Ранее мобильный трафик считался просто одним из дополнительных каналов генерации потенциальных клиентов. Основной платформой были сайты на ПК. В 2018 году эксперты посчитали, что 47-50% продаж генерируют приложения. Конверсия с такого ПО в 3-4 раза выше, чем с сайтов. Это говорит о том, что продажи неизбежно переходят в мобильную сферу, а значит, и бизнес должен активнее развивать этот канал.
2. Высокая лояльность клиентов. Смартфон гораздо чаще находится в зоне контакта с клиентом, чем тот же сайт. Пользователю проще заказать товар через приложение, найдя нужную иконку на своем телефоне, чем искать сайт на компьютере. Компания может стимулировать интерес клиента пуш-уведомлениями – это дешевый и достаточно эффективный вариант повышения лояльности целевой аудитории (ЦА).
3. Автоматизировать часть бизнес-процессов. Например, вызов такси через приложения позволяет разгрузить диспетчерские службы. Также в ресторанах и кафе можно заказывать еду, бронировать билеты на рейсы и многое другое. Для малого и среднего бизнеса выпускают приложения, которые могут принимать и обрабатывать заказы, перенаправлять вызовы на специалистов.
4. Принимать платежи и работать с онлайн-транзакциями. Сегодня многие пользуются мобильными приложения для интернет-банкинга, электронными деньгами и сервисами кэш-бек. Компании могут легко настроить прием платежей на своем приложении, подключившись к одному из финансовый агрегаторов.
5. Анализировать ЦА. С помощью приложений можно получить ряд дополнительных поведенческих метрик о целевой аудитории сайта, компании, продукта. Сегодня аналитические системы могут разделять трафик по устройствам входа. Такая информация поможет создать более точный портрет ЦА, а значит, разработать эффективные маркетинговые программы.
6. Сократить расходы на содержания штата сотрудников. Например, ряд приложения для бизнеса могут закрыть для компании проблему поиска операторов колл-центра. Кроме того, используя пуш-уведомления можно в разы сократить бюджет на контекстную рекламу и email-маркетинг.
7. Поддержка пользователей, сервисная служба. Мобильные приложения могут выступать в качестве центра поддержки пользователей. Основная задача таких программ – эффективная коммуникация с клиентами. Например, приложение может помочь настроить платежи, автооплату, заявку на вызов специалиста, перенаправить на чат с сотрудником и прочее.

## **1.3 Цель и задачи**

Целью данной выпускной квалификационной работы является разработка прототипа мобильного приложения «ShoppingList» (список покупок), на языке Java под управлением операционной системой Android.

Для достижения цели необходимо решить следующие задачи:

1. изучить предметную область;
2. разработать требования;
3. спроектировать архитектуру;
4. реализовать;
5. протестировать.

## **1.4 Новизна и практическая значимость**

Как показал анализ площадки Google Play, предназначенной для публикаций мобильных приложений, существуют десятки уже реализованных приложений, представляющих похожий функции. Также есть приложения, которые позволяют вести заметки, которые можно представить в виде списка покупок.

Все вышесказанное показывает отсутствие новизны и низкую практическую значимости приложения, тем не менее, для автора остается высокая степень субъективной значимости по рядку критериев:

* познакомиться и освоить инструментарий для разработки ОС Android;
* получить практические навыки по разработки ПО на языке Java под ОС Android;
* реализовать мобильное приложение, которое можно было бы приложить к своему портфолио.

## **1.5 Выводы по разделу**

В данном разделе было рассмотрено аналитическое обеспечение проектирования, что позволило сделать ряд выводов:

* Мобильное приложение – это программное обеспечение, предназначенное для работы на смартфонах, планшетах и других мобильных устройствах;
* Мобильное приложение имеет сложную структуру и разрабатывается под разные ОС;
* Для повышения конкурентоспособности и привлечения инвестиций бизнесу желательно использовать мобильные приложения;
* Мобильные приложения – это удобный инструмент для удовлетворения потребностей пользователя;
* Хотя новизна отсутствует, и практическая значимость реализуемого приложения имеет низкую значимость, но это не снимает вопрос субъективной значимости данного приложения для ее автора, что позитивно скажется в других проектах.

Таким образом, можно приступить к следующему разделу данной работы, а именно к формированию требований.

# **2. ТРЕБОВАНИЯ К ПРИЛОЖЕНИЮ**

Требования могут быть функциональными и нефункциональными. Функциональные требования указывают на то, что должна делать система, процесс или приложение, то есть относятся к поведению, а нефункциональные классифицируются как все остальные [7].

На этапе анализа требований предлагается использовать язык графического описания – UML, который предназначен для объектного моделирования. Он используется в разработке ПО и системном проектировании, а также для моделирования бизнес-процессов и организационная структура предприятий. Указанные структуры и модели поведения различных систем в UML создаются в виде специализированных диаграмм. Типология диаграмм довольно обширна, приведем несколько примеров: диаграммы деятельности, взаимодействия, развертывания и другие [8].

В работе были использованы диаграмма развертывания и вариантов использования (прецедентов). Диаграмма вариантов использования отражает отношения, существующие между актерами (actor) и вариантами использования. Она позволяет лучше понять физическую архитектуру (или архитектуру развертывания), что позволяет сформулировать нефункциональные требования.

Основная задача на данном этапе – предоставить разработчику и пользователю одно универсальное средства для обсуждения требований и поведения разрабатываемого программного обеспечения.

## **2.1 Нефункциональные требования**

Согласно своему жизненному циклу и предполагаемыми отношениями с другими приложениями и самой операционной системе, приложение должно учитывать и реализовывать нефункциональные требования, диктуемые общей средой разработки и развертывания, а также те, что дополнительно могут предложить участники проекта. Ниже приведен общий список нефункциональных требований:

1. приложение должно быть реализовано под платформу Android;
2. приложение должно поддерживаться устройствами с установленной операционной системой Android, начиная с версии 8.0 и более;
3. приложение должно работать в режиме альбомной/портретной ориентации экрана;
4. приложение должно быть разработано на языке программирования Java;
5. приложение должно быть устойчиво к ошибкам и вести их протоколирование;
6. приложение должно уметь синхронизировать свои данные по клиент-серверной архитектуре;
7. на этапе прототипа, серверная часть может быть статична и реализована на другом языке программирования, отличного от Java.

Подробнее можно посмотреть на рисунке 1.

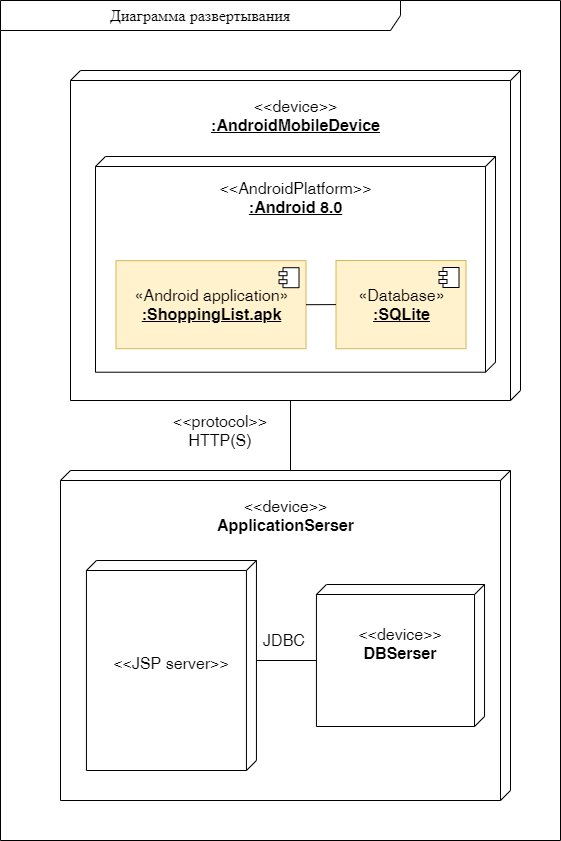


Рисунок 1 - Диаграмма развертывания

## **2.2 Функциональные требования**

На этапе анализа были выявлены следующие функциональные требования:

1. приложение должно иметь механизм аутентификации по номеру телефона;
2. приложение должно уметь создавать/удалять/редактировать список покупок;
3. приложение должно уметь создавать множество списков покупок;
4. приложение должно уметь взаимодействовать с фотокамерой телефона, то есть получать ссылку на захваченное изображение;
5. приложение должно уметь распознавать QR-коды на наличие HTTP-ссылок, на захваченных изображениях;
6. приложение может иметь функционал для создания сводных отчетов и другой аналитики на базе списков покупок.

Данные требования были перенесены на диаграмму вариантов использования, см. рисунок ниже. При этом стоит отметить некоторые детали характерные данной диаграмме, а именно связи между элементами, где обобщение подчеркивается при помощи стрелки с не закрашенным треугольником, пунктирная стрелка со стереотипом include это включение, а расширение является пунктирной стрелкой со стереотипом extend [9].

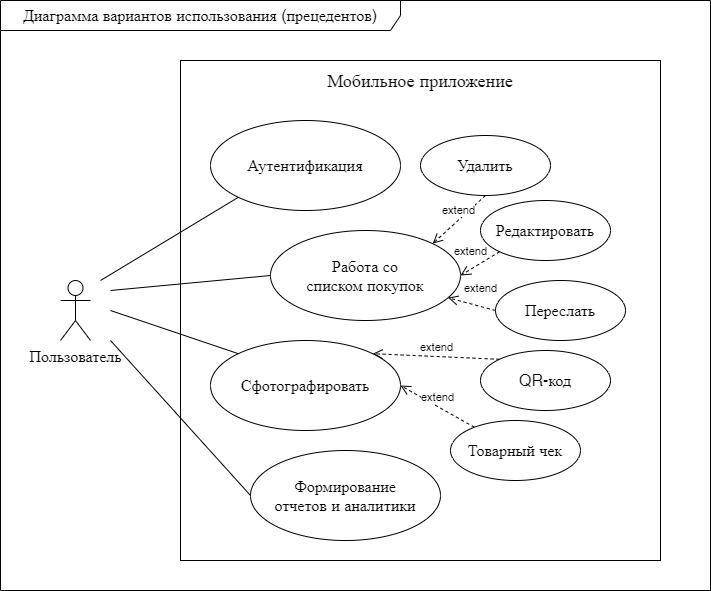


Рисунок 2 - Диаграмма вариантов использования

## **2.3 Выводы по разделу**

В данном разделе были сформулированы требования к приложению, что позволило выделить, описать и проанализировать:

• Нефункциональные требования, характеризующие свойства, которые должно демонстрировать приложение.

• Функциональные требования, то есть то, что должно делать приложение с точки зрения пользователя, тем самым закрывать его потребности.

Поскольку требования были сформулированы весьма обобщенно и не проходили этап верификации, то приступать к этапу проектирования нежелательно, но поскольку речь идет о разработке прототипа приложения, то это не критично.

Можно сделать вывод, что сформулированных требований будет достаточно для этапа проектирования прототипа приложения.

# **3. ПРОЕКТИРВОВАНИЕ ПРИЛОЖЕНИЯ**

На этапе проектирования основное внимание уделяется концептуальному решению, обеспечивающему выполнение основных требований, но не вопросам его реализации. Например, на этапе проектирования описываются программные объекты или схема базы данных. Идеи проектирования зачастую исключают низкоуровневые или «очевидные» детали – очевидные с точки зрения потенциального пользователя. Конечно же, проектные решения могут быть реализованы в программном коде, и эта реализация должна быть точной и полной. [7].

Попробуем определить кто является целевой аудиторией приложения для формирования списка покупок.

## **3.1 Определение пользователи**

Память человека так устроена, что ей свойственно помнить важное, но забывать неважное. При этом критерий что важно, а что нет, может измениться со временем, когда нужно срочно вспомнить важное. И чтобы как-то это нивелировать, можно записывать потенциально актуальную информацию на тот или иной носитель.

Наверное, очень трудно найти человека, который бы ни разу не забыл ничего купить в магазине – на праздник или просто так, покупая что-то на ужин. Реализуемое мобильное приложение дает возможность «помнить», то есть записать все, сверяясь со списком. По идее оно может помогать при:

* Не забыть купить продукты, которые нужны постоянно.
* Быть готовыми к приготовлению именно тех блюд, которые хотите.
* Экономить время при совершении покупок в магазине.

Поэтому, данное приложение может подойти:

* Семьи (чаще всего женщины), которые совершают покупки на несколько дней вперед. В основном предпочитают гипермаркеты.
* Компании людей, которые планируют проведение совместных мероприятий.

Все вышесказанное попробуем проиллюстрировать на диаграмме видов деятельности, которая отображает последовательные и параллельные процессы. Они полезны для моделирования бизнес-процессов, последовательностей выполнения задач, потоков данных и сложных алгоритмов [7].

Ниже представлена общая диаграмма видов деятельности для всех пользователей мобильного приложения, см. рисунок ниже.

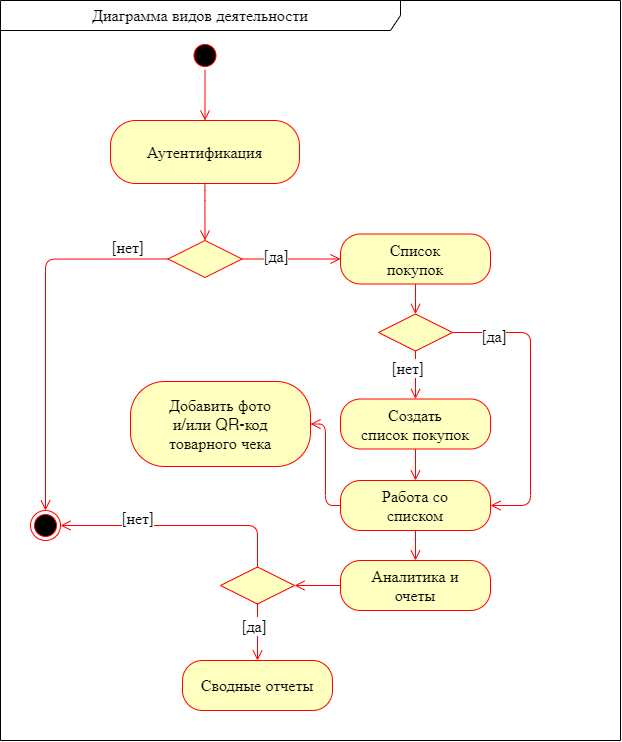


Рисунок 3 - Сценарии использования

## **3.2 Архитектура**

Основными характеристиками любого программного обеспечения, важными для предсказуемой работы с ним, являются надежность, эффективность использования различных функций, качество сопровождения и другие. Все они находятся в прямой зависимости от уровня качества проектирования архитектуры. В свою очередь уровень качества проектирования архитектуры зависит от того, какая была выбрана архитектура для программных систем.

Типология основных программных архитектур на сегодняшний день включает в себя монолитную архитектуру, клиент-серверную, микросервисную и сервис-ориентированную [10]. Каждая из них обладает как определенными преимуществами для разработчика, так и некоторыми недостатками.

Поскольку одно из нефункциональных требований сформулировано – «приложение должно уметь синхронизировать свои данные по клиент-серверной архитектуре», поэтому была выбрана клиент-сервер архитектура, при которой само мобильное приложение будет иметь монолитную архитектуру, включающая компоненты среды и слой.

Архитектура приложения включает в себя компоненты и слои в зависимости от выбранного типа разработки [11]. Она может быть одноуровневой, двухуровневой, трехуровневой, иметь разные типы подключений. В ее основе находится единый интерфейс, благодаря которому между собой взаимодействуют все остальные части программы. Общая структура выглядит следующим образом:

* Ядро приложения. Включает слои и компоненты, с которыми пользователь не может взаимодействовать, но при этом все процессы происходят именно тут.
* Графический интерфейс. Это визуальная часть приложения, с которой пользователь взаимодействует. Выполняя какие-либо действия, он визуально наблюдает изменения, при этом запрос идет в серверную часть, в ядро, где он обрабатывается и выдается результат.
* Компоненты, которые можно использовать повторно. Например, различные библиотеки, визуальные составляющие и другие элементы.
* Дополнительные ресурсы, которые использует приложение. Это могут быть графические элементы, звуки и другие компоненты.

Android – это программный стек на базе Linux с открытым исходным кодом, созданный для широкого спектра устройств и форм-факторов. На следующем изображении показаны основные компоненты платформы Android, как это представлено в официальной документации [12].



Рисунок 4 - Программный стек Android

Программный код нашего приложения разворачивается и исполняется в желтом блоке – Android Runtime.

Ниже представлена диаграмма компонентов, отражающая общую клиент-серверную архитектуру, см. рисунок ниже.

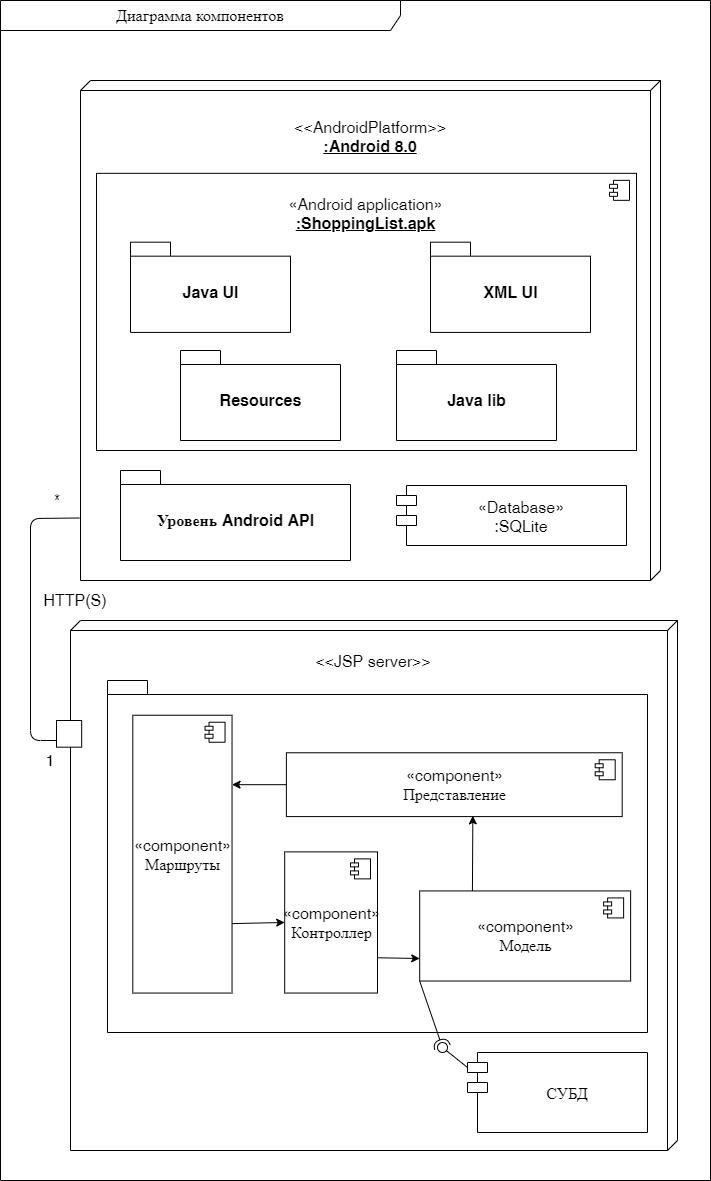


Рисунок 5 - Общая архитектура приложения и его расположение в среде исполнения

## **3.3 Диаграмма обмена данными и классов**

Для представления обмена данными, одного из успешных сценариев, была использована диаграмма последовательности. Данная диаграмма имеет три дорожки, которые являются конкретными экземплярами классов, см рисунок. ниже.

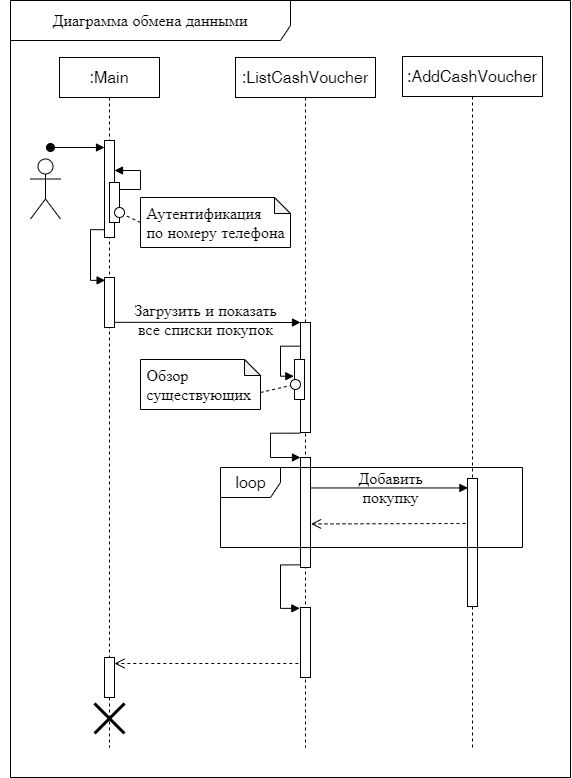


Рисунок 6 - Диаграмма обмена данными на примере диаграммы последовательности

Далее представлена диаграмма классов. См. рисунок ниже.

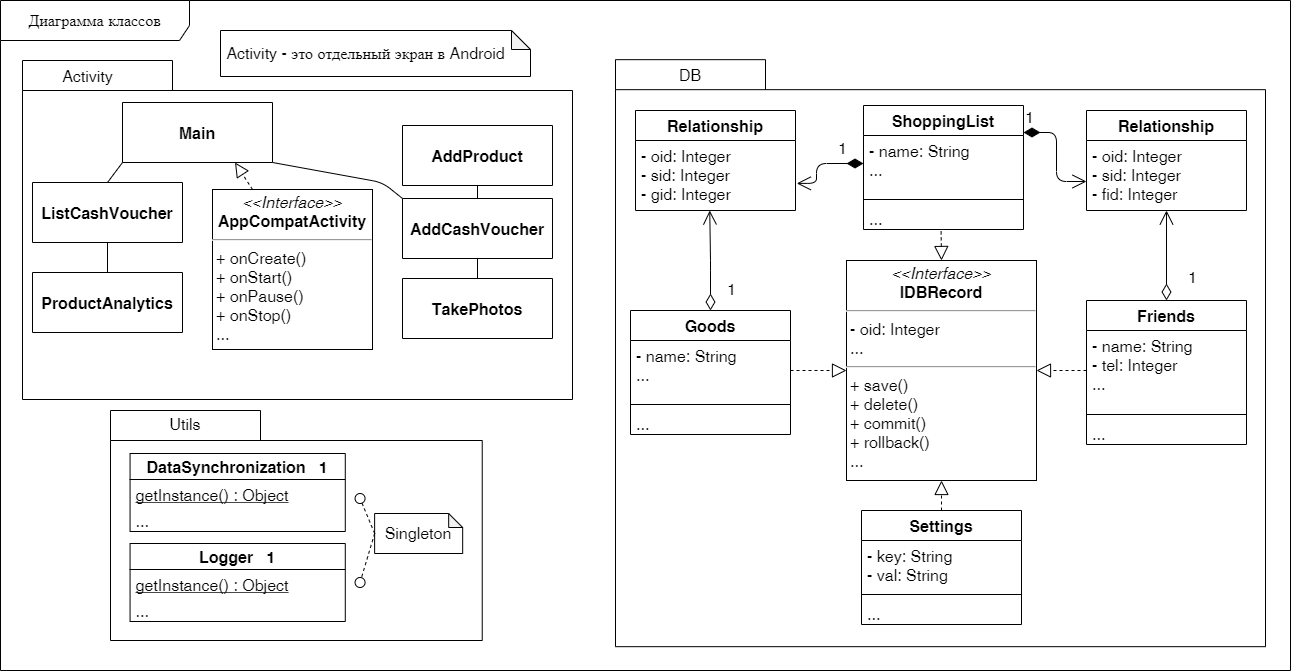


Рисунок 7 - Диаграмма классов мобильного приложения

## **3.4 Проектирование базы данных**

В качестве подходящей нам технологии проектирования выступает ORM (Object-Relational Mapping, рус. объектно-реляционное отображение, или преобразование). Технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных» [13].

Шаблон Representing Objects as Tables (Представление объектов в виде таблиц) предлагает для каждого класса объектов, подлежащих постоянному хранению, определить отдельную таблицу (при использовании реляционной базы данных), а атрибуты объекта, содержащие данные простых типов (числа, строки, логические переменные и т.д.), хранить в отдельных столбцах.

Если все атрибуты объекта являются данными простых типов, то установка такого соответствия не представляет сложностей. Однако реальность не столь проста, поскольку одни объекты могут содержать ссылки на другие сложные объекты, а реляционная модель базы данных требует, чтобы эти значения были простыми [7].

Существует много разных подходов к семантическому моделированию баз данных. В последние десятилетия, одним из наиболее популярных языков семантического моделирования является UML. Проектирование реляционных БД – только одна и не слишком большая область применения этого языка, его возможности гораздо шире, однако подмножество UML (диаграммы классов) успешно применяется именно для таких целей [14].

Иногда в диаграмме требуется отразить тот факт, что ассоциация между двумя блоками имеет специальный вид «часть-целое». В этом случае блок «целое» имеет более высокий концептуальный уровень, чем блок «часть». Ассоциация такого рода называется агрегатной. Графически агрегатные ассоциации изображаются в виде простой ассоциации с не закрашенным ромбом на стороне класса-«целого».

Бывают случаи, когда связь «части» и «целого» настолько сильна, что уничтожение «целого» приводит к уничтожению всех его «частей». Агрегатные ассоциации, обладающие таким свойством, называются композитными, или просто композициями. При наличии композиции объект-часть может быть частью только одного объекта-целого (композита). При обычной агрегатной ассоциации «часть» может одновременно принадлежать нескольким «целым». Графически композиция изображается в виде простой ассоциации, дополненной закрашенным ромбом со стороны «целого».

Пример агрегации и композиции показан на рисунке ниже.

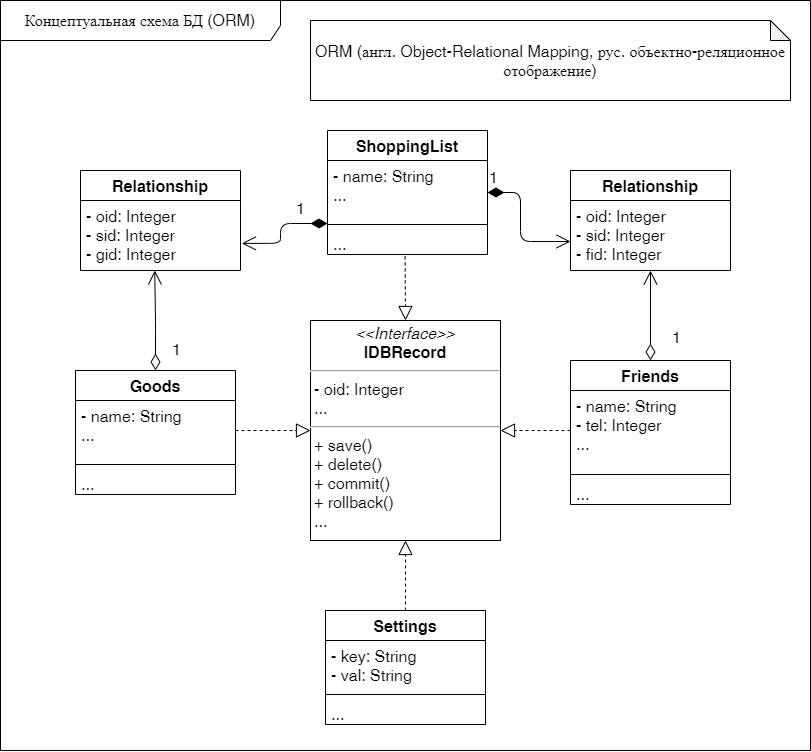


Рисунок 8 - Таблицы БД и их связи согласно ORM

Заметим, что в контексте проектирования реляционных БД агрегатные и в особенности композитные ассоциации влияют только на способ поддержки ссылочной целостности. В частности, композитная связь является явным указанием того, что ссылочная целостность между «целым» и «частями» должна поддерживаться путем каскадного удаления частей при удалении целого [14].

## **3.5 Выводы по разделу**

В данном разделе было проведено проектирование приложения, что позволило выделить, описать и проанализировать следующие положения.

Уточнена целевая аудитория и примерный список доступных им функций с последующим построением диаграммы видов деятельности.

Архитектура приложения является монолитной с множеством программных слоев, при этом среда исполнения является ОС Android. Была построена диаграмма компонентов приложения, отражающая некоторые слои в виде пакетов.

Была построена диаграмма обмена данными, отражающая входящие и выходящие данные. На диаграмме классов была уточнена общая структура иерархии классов, их атрибуты и зависимости.

Были спроектированы некоторые сущности (модель, таблица) БД для хранения состояний.

В результате, можно сделать общий вывод, что теперь у нас есть все необходимые данные и можно приступать к реализации приложения.

# **4. РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ**

## **4.1 Выбор технологий**

Выбор технологии реализации системы коренным образом влияет на все этапы жизненного цикла приложения, от проектирования до сопровождения. От выбранного языка программирования требуются следующие свойства:

* высокоуровневый и строго типизированный;
* наличие бесплатных инструментов для разработки;
* широкая распространённость и популярность;
* поддержка платформы OC Android;
* относительно невысокая стоимость рабочей силы.

Ниже перечислены самые популярные языки программирования под ОС Android:

* С++;
* Си (англ. C);
* Kotlin;
* Java.

С++ – компилируемый, статически типизированный язык программирования общего назначения. На сегодняшний день именно этот язык наиболее распространен по всему миру. Он уделяет значительное внимание поддержке объектно-ориентированного и обобщённого типов программирования. Содержательно в C++ сочетаются свойства как высокоуровневых, так и низкоуровневых языков. Скорость разработки на нем существенно более низкая в сравнении с другими известными языками, однако производительность готового ПО компенсирует данный недостаток.

Си (англ. C) – компилируемый статически типизированный язык программирования общего назначения. Согласно дизайну языка, его конструкции близко сопоставляются типичным машинным инструкциям, благодаря чему он нашёл применение в проектах, для которых был свойственен язык ассемблера, в том числе как в операционных системах, так и в различном прикладном программном обеспечении для множества устройств — от суперкомпьютеров до встраиваемых систем.

Kotlin (Котлин) — статически типизированный, объектно-ориентированный язык программирования, работающий поверх Java Virtual Machine и разрабатываемый компанией JetBrains. Авторы ставили целью создать язык более лаконичный и типобезопасный, чем Java, и более простой, чем Scala. Язык полностью совместим с Java, что позволяет Java -разработчикам постепенно перейти к его использованию; в частности, язык также встраивается Android, что позволяет для существующего android-приложения внедрять новые функции на Kotlin без переписывания приложения целиком. На Google I/O 2019 было объявлено, что язык программирования Kotlin стал приоритетным в разработке под Android.

Java – жестко типизированный объектно-ориентированный язык программирования. Одним из плюсов языка является платформонезависимость. В отличие от многих других языков, включая C и C++, программы на языке Java компилируются в байт-код, исполняемый на виртуальной Java-машине. Однако синтаксис языка является многословным, что делает код объёмным и сложно читаемым, по сравнению с Kotlin.

Java является и остается одним из самых популярных языков программирования в мире, подбор разработчиков будет сравнительно менее проблематичным [15].

Среди рассмотренных языков была выбрана Java, так как она удовлетворяет всем требованиям, а также обязательна согласно нефункциональным требованиям.

## **4.2 Выбор системы управления баз данных**

Выбор технологии хранения данных играет очень важную роль, поэтому желательно, приводить минимальный сравнительный обзор СУБД и выбирать подходящий вариант согласно требованиям, задачам и цели. Но поскольку ОС Android изначально поставляется с отличной СУБД SQLite и предоставляет удобный инструментарий, то остановится на SQLite [16]. Стоит сказать пару слов о данной СУБД.

SQLite – база данных, которая довольно просто встраивается в различные приложения. В отличие от сетевых СУБД она предоставляет разработчику обширный инструментарий, поскольку базируется на файлах с какими-либо данными, обращение к которым производятся напрямую, а не через сокетов и порты. Высокая скорость и мощность работы обеспечивается технологиями обслуживающих SQLite библиотек. Однако при всех обозначенных преимуществах данная СУБД имеет ряд недостатков, а именно: отсутствие системы пользователей; отсутствие возможности увеличения производительности; отсутствие возможности удалённого подключения. Перечисленные недостатки не являются критическими, в нашем случае их можно проигнорировать.

Явные преимущества, про которые не стоит забывать и учитывать в следующие проектах, а именно: простота освоения, отсутствия требований к транзакциям, наличие механизма представлений, низкое потребление ресурсов, разные режимы работы, ориентированные на быстроту, надежность, стабильность, ведение файл-журнала.

## **4.3 Реализация приложения**

При запуске приложения открывается экран аутентификации по номеру телефона см. рис. ниже.



Рисунок 9 - Экран аутентификации

После экрана аутентификации, отрывается экран всех списков покупок см. рис. ниже. При первом запуске приложения, данный список пустой, и чтобы его наполнить надо нажать на элемент управления «Добавить».

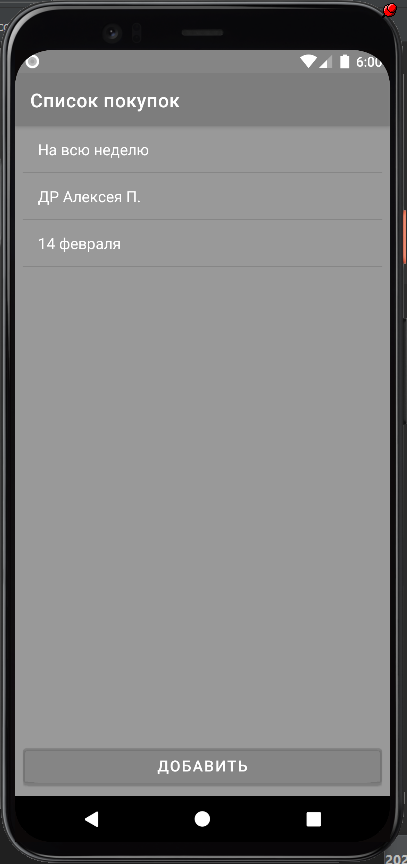


Рисунок 10 - Экран списков покупок

Чтобы создать список покупок и его наполнить, необходимо нажать на кнопку «Добавить», далее будет открыт экран, на котором можно ввести имя списка покупок и приступить непосредственно к наполнению. См. рисунок ниже.

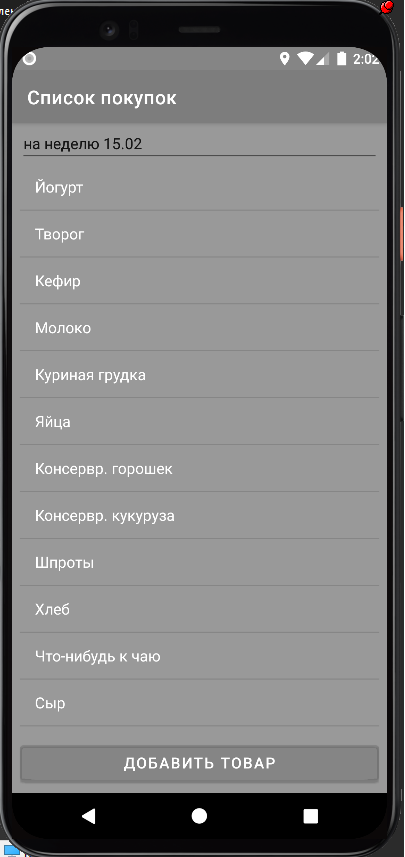


Рисунок 11 - Экран списка товаров

Чтобы добавить новый товар в список покупок, необходимо ввести его название и заполнить необязательное поле, что хранит его количество. См. рисунок ниже.

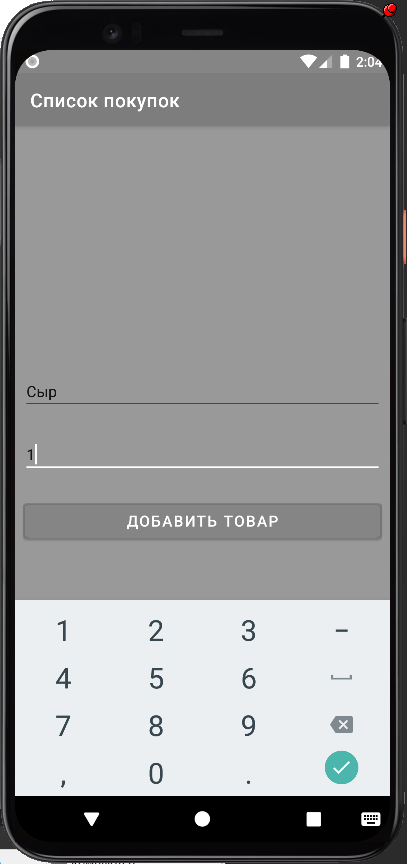


Рисунок 12 - Экран добавления товара

## **4.4 Выводы по разделу**

В данном разделе была проведена реализация прототипа мобильного приложения, что позволило выделить, описать и проанализировать следующие положения.

Были рассмотрены наиболее популярные языки программирования, среди которых был выбран наиболее подходящий – Java, также была выбрана СУБД SQLite. Были продемонстрированы некоторые экраны приложения.

В дальнейшем планируется доработать логику и экраны приложения, что должно позволить выпустить первую полноценную версии.

Можно сделать общий вывод, что прототип мобильного приложения был реализован и можно приступить к этапу тестирования.

# **5. ТЕСТИРОВАНИЕ**

Тестирование программного обеспечения – проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом. В более широком смысле, тестирование – это одна из техник контроля качества, включающая в себя активности по планированию работ, проектированию тестов, выполнению тестирования и анализу полученных результатов.

При тестировании приложения, в первую очередь, используется функциональный подход к тестированию.

Функциональное тестирование – это тестирование ПО для проверки реализуемости функциональных требований, то есть способность ПО в определенных условиях решать задачи, нужные пользователям. Данный тип тестирования является одним из ключевых видов тестирования, задача которого – установить соответствие разработанного программного обеспечения исходным функциональным требованиям заказчика.

Желательно провести модульное (блочное, unit) тестирование на этапе сборки проекта, поскольку это позволит получить информацию о корректности взаимодействия отдельных модулей программного кода, соответствие управления данными, процедурами использования и обработки.

Модульные тесты работают на очень низком уровне, близко к исходному коду приложения. Они заключаются в тестировании отдельных методов и функций классов, компонентов или модулей, используемых в ПО. Модульные тесты, как правило, не требуют больших расходов на автоматизацию и могут выполняться сервером непрерывной интеграции очень быстро.

## **5.1 Функциональное тестирование**

Функциональное тестирование (тестовый сценарий)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Входные**  **данные** | **Ожидаемый**  **результат** | **Полученный**  **результат** | **Пройден** |
| 1 | Запуск приложения | Пользователь нажимает на иконку приложения | Экран аутентификации | Экран аутентификации | Да |
| 2 | Аутентификация по номеру телефона | Пользователь находиться на экране аутентификации | Пользователь кликает по полю ввода номера телефона, появляется клавиатура, он вводит свой номер телефона и нажимает Далее | Получен номер телефона после нажатия на кнопку Далее | Да |
| 3 | Экран списка покупок | Пользователь находится на экране списка покупок первый раз | Список покупок пуст | Пустой список покупок | Да |
| 4 | Создание списка покупок | Пользователь находиться на экране всех списков покупок | Пользователь нажимает на кнопку Добавить | Отрылся экран Добавить список покупок | Да |
| 5 | Добавить новый товар | Пользователь находиться на экране Добавить список покупок | Пользователь нажимает на кнопку Добавить товар | Открылся экран Добавить новый товар | Да |
| 6 | Заполнение атрибутов товара | Открыт экран Добавить новый товар | Пользователь нажимает на обязательное для заполнения поле Наименование, при помощи клавиатуры заполняет и нажимает на кнопку Добавить | Экран обновился на Добавить список покупок и пользователь видит товар который только что добавил | Да |
| 7 | Удалить товар | Пользователь находиться на экране Добавить список покупок | Пользователь нажимает на элемент Удалить | Экран обновился на Добавить список покупок и на экране нет удаленного товара | Да |
| 8 | Свернуть приложение нажатием на управляющую кнопку Android Домой | Приложение открыто и пользователь находится на одном из экранов (любое) | Пользователь нажимает управляющую кнопку Android Домой | Приложение сворачивается, все введенные данные сохраняются во временное хранилище | Да |
| 9 | Показать все запущенные приложения нажатием на управляющую кнопку Android Последнее приложение | Приложение открыто и пользователь находится на одном из экранов (любое) | Пользователь нажимает управляющую кнопку Android Последнее приложение | Приложение отображается в уменьшенном виде рядом с другими приложениями, все введенные данные сохраняются во временное хранилище | Да |
| 10 | Изменение ориентации экрана с альбомной на книжную и/или обратно | Приложение открыто и пользователь находится на одном из экранов (любое) | Пользователь повернул телефон, изменив ориентацию телефона с книжной на портретную или на оборот | Ориентация изменилась, содержимое экрана сохранилось | Да |

Мобильное приложение было протестировано на эмуляторе Pixel 5 API 26.

## **5.2 Выводы по разделу**

В данном разделе было проведено минимальное функциональное тестирование приложения, которое затронуло один из сценариев использования, а именно открыть приложение, пройти аутентификацию, создать список покупок и наполнить его. Весь сценарий, его этапы и результаты были вынесены в отдельную сводную таблицу.

В результате, можно сделать общий вывод, что прототип приложения работает согласно тестовому сценарию и можно перейти к следующему разделу, то есть к заключительной части данной работы.

# **ЗАКЛЮЧЕНИЕ**

В выпускной квалификационной работе проведено аналитическое обеспечение проектирования, которое показало, что такое мобильное приложение, его сложную структуру и сложность разработки под ОС Android.

В этом же разделе, была поставлена цель – разработка прототипа мобильного приложения «ShoppingList» (список покупок), на языке Java под управлением операционной системой Android, и сформулированы задачи. Новизна поставленной цели отсутствует, а практическая значимость низкая, тем не менее, для автора субъективная значимость весьма высока.

В разделе о требованиях к приложению сформулированы нефункциональные и функциональные требования. Они были сформулированы весьма обобщенно, что не сильно повлияло на разработку прототипа, в дальнейшем, данный этап стоит прорабатывать гораздо тщательнее.

В разделе о проектировании приложения описаны и графически представлены в виде UML-диаграмм пользовательские функции; архитектура в виде диаграммы компонентов; диаграмма обмена данных и классов; таблица и модель для хранения результатов работы модуля.

В разделе о реализации приложения выбраны технологии, где основной язык Java, а СУБД SQLite. Показаны основные экраны приложения.

В разделе о тестировании, было проведено минимальное функциональное тестирование, которое затронуло один из сценариев использования.

На основании всего вышесказанного можно сделать вывод, все поставленные задачи решены и цель достигнута.

В дальнейшем планируется доработать приложение и перевести его из состояния прототипа в релиз-кандидат.

# **СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

1. Mobile marketing statistics compilation 2021 [Электронный ресурс]. URL: http://www.smartinsights.com/mobile-marketing/mobile-marketinganalytics/mobile-marketing-statistics/ (дата обращения: 26.01.2022).

2. Смартфоны (мировой рынок) [Электронный ресурс]. URL: https://www.tadviser.ru/index.php/Статья:Смартфоны\_(мировой\_рынок) (дата обращения: 26.01.2022).

3. Мобильный трафик (российский рынок)) [Электронный ресурс]. URL: https://www.tadviser.ru/index.php/Статья:Мобильный\_трафик\_(российский\_рынок)) (дата обращения: 26.01.2022).

4. ПНСТ 277-2018 Российская система качества. Сравнительные испытания мобильных приложений для смартфонов, ПНСТ от 26 июня 2018 года №277-2018 [Электронный ресурс] URL: https://docs.cntd.ru/document/1200159701 (дата обращения: 31.01.2022).

5. Федеральный закон от 02.08.2019 г. N 259-ФЗ "О привлечении инвестиций с использованием инвестиционных платформ и о внесении изменений в отдельные законодательные акты Российской Федерации".

6. Мобильное приложение [Электронный ресурс]. URL: https://www.calltouch.ru/glossary/chto-takoe-mobilnoe-prilozhenie-i-zachem-ono-mozhet-potrebovatsya/ (дата обращения: 31.01.2022).

7. Ларман, Крэг. Применение UML 2.0 и шаблонов проектирования. Практическое руководство. 3-е издание.: Пер. с англ. — М.: ООО «И.Д. Вильямс», 2013. — 736 с.: ил. — Парал. тит. англ.

8. Г. Буч, Д. Рамбо, И. Якобсон. Краткая история UML // Язык UML. Руководство пользователя = The Unified Modeling Language User Guide. — 2-е. — М.: ДМК Пресс, 2006. — С. 14. — 496 с. — ISBN 5-94074-334-X.

9. Использование диаграммы вариантов использования UML при проектировании программного обеспечения [Электронный ресурс]. URL: https://habr.com/ru/post/566218/ (дата обращения: 26.01.2022)

10. Фаулер М., Райс Д. Архитектура корпоративных программных приложений.: Пер. с англ. — М.: Издательский дом "Вильямс", 2007. — 544 с.: ил. — Парал. тит. англ.

11. Архитектура мобильного приложения [Электронный ресурс]. URL: https://wezom.com.ua/blog/arhitektura-mobilnogo-prilozheniya (дата обращения: 02.02.2022).

12. Platform Architecture [Электронный ресурс]. URL: https://developer.android.com/guide/platform (дата обращения: 03.02.2022).

13. ORM [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/ORM (дата обращения: 03.02.2022).

14. Концептуальное проектирование реляционных баз данных с использованием языка UML [Электронный ресурс]. URL: http://citforum.ru/database/articles/umlbases.shtml (дата обращения: 03.02.2022).

15. TIOBE Index for January 2022 [Электронный ресурс]. URL: https://www.tiobe.com/tiobe-index/ (дата обращения: 07.02.2022).

16. Save data in a local database using Room [Электронный ресурс]. URL: https://developer.android.com/training/data-storage/room/ (дата обращения: 08.02.2022).

# **ПРИЛОЖЕНИЕ А. Фрагменты исходного кода**

Весь исходный код хранится в распределенной системе управления версиями (GitHub) в глобальной сети Интернет в свободном доступе по адресу: https://github.com/IliaLenskii/graduate-work/tree/master/3