МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет	*Кафедра*
вычислительной техники	«Вычислительная техника»

Специальность: 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения»

Специализация: «Эксплуатация вычислительных машин, комплексов, систем и сетей специального назначения»

ДИПЛОМНЫЙ ПРОЕКТ

на тему

Программное обеспечение проверки товаров

Студент _____ Волков Максим Алексеевич
(подпись, дата) (ФИО полностью)

Руководитель _____ Федюнин Р.Н.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ Кучин А.В.
(подпись, дата) (фамилия, инициалы)

Рецензент Доцент ПензГТУ _____ 19.06.20 Печерский С.В.
(должность, место работы) (подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 10.06.2020 г. № 13)

Заведующий кафедрой _____ Митрохин М.А.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой Хорошо (протокол заседания ГЭК от 19.06.2020 № 15)

Секретарь ГЭК _____ Гурин Е.И.
(подпись) (фамилия, инициалы)

Пенза 2020

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
Вычислительная техника

«Утверждаю»
Заведующий кафедрой
_____ Митрохин М.А.
«30» апреля 2020г.

ЗАДАНИЕ
НА ДИПЛОМНЫЙ ПРОЕКТ

1. Студент Волков Максим Алексеевич гр. 15ВО2 факультета вычислительной техники.

2.Тема работы: «Программное обеспечение проверки товаров».

Тема утверждена приказом ПГУ № 0335а/о от 30.04.2020 г.

3. Руководитель работы к.т.н., доцент каф. ВТ ПГУ Федюнин Р.Н.

4. Задание на работу (назначение разработки, исходные данные и т.п.).

Назначение разработки – Система предназначена для обеспечения для сканирования данных товара по маркировке и иным отличительным признакам на упаковке, в состав которого вошли приложение и библиотеки по работе с аппаратными и программными модулями ОС Andriod

5. Перечень подлежащих разработке вопросов.

Требуется произвести анализ предметной области, выбрать функциональный состав системы, осуществить выбор стека технологий для реализации системы, разработать алгоритм работы системы, разработать структурную схему системы, разработать диаграммы: прецедентов, активностей, последовательностей, объектов, классов, с последующим написанием программы.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	10	30.04.20	
2.	Выбор компонентов для реализации приложения	10	07.05.20	
3.	Разработка архитектуры приложения	20	08.05.20	
4.	Написание программы	10	14.05.20	
5.	Отладка системы	10	16.05.20	
6.	Моделирование и поиск уязвимостей, исправление ошибок в работе системы	10	22.05.20	
7.	Оформление выпускной работы	30	28.05.20	

Дата выдачи задания 30 «апреля» 2020 года.

Руководитель дипломного проекта _____ Федюнин Р.Н.
(подпись, дата)

Задание к исполнению принял студент _____ Волков М.А.
(подпись, дата)

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

РЕФЕРАТ

Дипломный проект содержит 109 лист, 39 рисунков, 7 источников, 2 приложения.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, NoSQL, СКАНИРОВАНИЕ ТОВАРА, СУБД, ANDROID.

Цель работы – разработать программное обеспечения для сканирования данных товара по маркировке и иным отличительным признакам на упаковке.

Объектом разработки является мобильное приложение, обеспечивающее сканирование штрихкода товара на упаковке и предоставление информации о товаре в соответствии с данными из СУБД. Данный проект это мобильное приложение на основе клиент серверного подхода, созданное на основе краудсорсинговых данных пользователей. Приложение интегрируется со сторонними службами для создания базы данных картографирования товаров. Приложение построено с использованием технологий, которые являются производительными, гибкими и масштабируемыми.

В результате проведенной работы разработан проект, удовлетворяющий поставленной задаче, а именно, создан программный комплекс, в состав которого вошли приложение и библиотеки по работе с аппаратными и программными модулями ОС Andriod.

Содержание

Реферат 4
Введение 6
1. ИССЛЕДОВАНИЕ 7
1.1. Анализ предметной области 7
1.2. Сравнение идеи с аналогичными программными продуктами 9
1.3. Требования к разрабатываемому программному обеспечению 14
1.4. Требования к системе 17
1.5. Сравнение и выбор средств разработки 19
2. Разработка приложения 30
3. Технология тестирования приложений на базе платформы amazon 65
4 Технико-экономическое обоснование разработки инвестиционного проекта 73
4.1 Описание программного продукта 73
4.2 Расчет затрат на разработку 74
4.3 Формирование денежного потока 78
4.3.1 Расчет поступлений денежных средств 78
4.3.2 Расчет выплат денежных средств 80
4.4 Анализ рисков 84
5 Оценка травмобезопасности на рабочем месте оператора пэвм 85
5.1 Общие положения 86
5.2 Обеспечение электробезопасности 87
5.3 Требования безопасности при работе на ПЭВМ 90
Заключение 93
ЗАКЛЮЧЕНИЕ 94
Список используемых ИСТОЧНИКОВ 95
ПРИЛОЖЕНИЕ А 96
ПРИЛОЖЕНИЕ Б 107
Приложение В 113

ВВЕДЕНИЕ

Роскачество провело исследование мобильных приложений по доставке продуктов. 24 приложения из российского сегмента Google Play и App Store проверили по 60 критериям, включая безопасность, удобство, производительность, функциональность и информативность.

В конечном итого выяснено, что меньше десяти из них имеют функции работы с товарами вызывающими аллергию.

Из-за ограниченного количества продуктов, не содержащих аллергены, потребители тратят много времени на чтение списка ингредиентов продуктов находясь в магазине, чтобы проверить их безопасность, не зная, есть ли в магазине продукты которые бы удовлетворяли их запросам.

В данной дипломной работе будет создана система, которая решает эту проблему, позволяя пользователям находить продукты без аллергенов через мобильное приложение, которое позволит конечным пользователям не тратить свое время на поиск безопасных продуктов. Данный проект это гибридное мобильное приложение на основе облачных вычислений, созданное на основе краудсорсинговых данных пользователей. Приложение интегрируется со сторонними службами для создания базы данных картографирования продуктов. Приложение построено с использованием технологий, которые являются производительными, гибкими и масштабируемыми.

1. ИССЛЕДОВАНИЕ

1.1. Анализ предметной области

Найти в магазине продукты, не содержащие аллергены, сложно, прежде всего потому, что количество и разнообразие гипоаллергенных продуктов на рынке ограничено. Кроме того, в магазинах, как правило, имеется четко ограниченный набор гипоаллергенных продуктов, а в разных точках одной и той же сети не обязательно есть один и тот же набор продуктов. Из-за этих несоответствий человеку с аллергией, трудно узнать обо всех доступных продуктах питания, которые он имеет на рынке. Другим фактором, который усложняет поиск продуктов, не содержащих аллергенов, является процесс определения того, является ли продукт безопасным для покупки. Люди с пищевой аллергией тратят значительное количество времени на покупку продуктов, читая списки ингредиентов продуктов, чтобы убедиться в их безопасности, что делает процесс покупки продуктов трудоемким и неприятным.

Пищевая аллергия является растущей проблемой. Исследование проведенных АНО Роскачество в 2019 году, показало, что в период между 2010 и 2019 г.г. на 50% увеличились случаи пищевой аллергии среди детей. Согласно оценкам АНО Роскачество 1,5 миллионов человек в России сегодня страдают пищевой аллергией, из них 10% в возрасте до 18 лет. До тех пор, пока лекарство не найдено, единственный вариант, которым могут пользоваться данная категория людей это полностью избегать продуктов, на которые у них аллергия. Это, однако, не так легко сделать, как может показаться, потому что наиболее распространенные аллергены, которые вызвали реакции, являются основными ингредиентами пищевых продуктов и

блюд: молоко, яйцо, пшеница, арахис, орехи, соя, рыба и моллюски; которые встречаются почти во всех обычных продуктах, продаваемых на рынке. Несмотря на то, что за последние несколько лет рынок альтернативных продуктов питания значительно вырос, и сейчас существует больше компаний, производящих новые продукты без аллергенов, количество таких продуктов все еще очень ограничено, и найти их сложно.

Исходя из выше сказанного, цель данного дипломного проекта - приложение, которое решает проблему поиска продуктов без аллергенов, предоставляя платформу, которая позволяет пользователям искать и находить безопасные продукты с помощью своих мобильных телефонов. Приложение помогает пользователям экономить время, позволяя им автоматически проверять безопасность продукта. Приложение построено с использованием данных, предоставленных пользователями. Когда пользователи используют приложение для проверки безопасности продукта, они выбирают пункт геолокации и создают записи в СУБД, отображающие актуальное соотношение продуктов и магазинов. Приложение интегрируется со сторонними сервисами API служб Google и API для сбора необходимой информации о местоположении и продукте. Версия продукта, поставляемая в рамках этого проекта, представляет собой минимальный жизнеспособный продукт (MVP), который содержит четыре основные функции.

Функционал разрабатываемого мобильного приложения:

- функция управления пользователями, которая позволяет пользователям регистрироваться и входить в приложение,
- функция проверки продукта, которая позволяет пользователю проверять безопасность продукта,
- функция поиска в магазине, которая позволяет пользователю для поиска всех доступных безопасных продуктов и отслеживания их местонахождения,
- функция поиска, позволяющей пользователю узнать, где продается конкретный продукт.

Приложение представляет собой гибридное приложение, построенное на платформе Ionic для Android.

1.2. Сравнение идеи с аналогичными программными продуктами

У большинства крупных торговых сетей есть собственные мобильные приложения, которые в последнее время набирают популярность (у наиболее популярных приложений – «Пятерочка» и «Красное&Белое» – более 5 миллионов установок из Google Play). АНО Роскачество провело сравнение функциональных возможностей 18 наиболее популярных мобильных приложений торговых сетей в App Store и Google Play.

Среди оцененных приложений «Ашан», «ВкусВилл», «Вкусомания Азбука Вкуса», «Дикси», «Красное&Белое», «Лента», «Магнит: акции и скидки», «Мой Перекресток», «Мой Спар», «Моя Виктория», «Моя Карусель», «О'Кей доставка продуктов», «Пятерочка», «Сеть Магазинов Семья», «Billa Россия», «Globus Гипермаркеты Глобус», Fix Price и Metro Cash & Carry.

Эксперты анализировали функциональность по 10 параметрам: список покупок, история покупок, адреса магазинов, карта лояльности или программа лояльности в приложении, перечень акционных товаров, наличие товаров в магазине, онлайн-заказ (с доставкой и с самовывозом из магазина), сканер штрихкодов для проверки цен и поддержка системы самостоятельного сканирования товаров.

Среди прочих параметров функциональности, по результатам анализа приложений в декабре 2019 года, не все так безоблачно (рисунок 1.1., 1.2).

Рисунок 1.1 Проверка функционала в мобильных приложениях

Рисунок 1.2 Проверка функционала в мобильных приложениях

Список покупок. Опция реализована в 12 из 18 исследованных приложений, а именно: «Ашан», «ВкусВилл», «Дикси», «Лента», «Магнит: акции и скидки», «Мой Перекресток», «Моя Виктория», «Моя Карусель»,

«Пятерочка», «Сеть Магазинов Семья», «Billa Россия» и «Globus – Гипермаркеты Глобус». С помощью данной функции пользователи могут планировать свои покупки перед походом в супермаркет. Возможность добавить товар в корзину в интернет-магазине или создать шаблоны в данном случае не рассматривалась, требовалось наличие отдельного списка с возможностью ввода произвольных наименований.

История покупок. Этой функцией могут похвастаться 14 приложений, за исключением «Дикси», «Красное&Белое», «Billa Россия» и Fix Price. Используя данную функцию, потребители могут отследить, какие товары покупали и когда.

Создание карты покупок. Возможность привязки или создания карты покупок (или использования иной программы покупок в приложении) реализована в 17 приложениях, за исключением «Billa Россия» (торговая сеть имеет свою карту, но добавить ее в приложение нельзя).

Товары по акции. Отдельный раздел с перечнем акционных товаров представлен во всех приложениях, кроме «Вкусомания – Азбука Вкуса».

Наличие товара в магазине. Что касается возможности проверить наличие товаров в магазине, то только у 5 из 18 исследованных приложений такая функция имеется: это «ВкусВилл», «Красное&Белое», «Лента», Fix Price и Metro Cash & Carry. Данная функция особенно удобна, если нужен конкретный товар, ради которого клиент идете в магазин.

Онлайн-заказ с доставкой на дом. Оформить онлайн-заказ с доставкой на дом, не выходя из основного приложения, позволяют шесть программ: «Ашан», «Мой Перекресток», «Мой Спар», «О’Кей», «Сеть Магазинов Семья» (заказ только в Перми и Березниках) и Metro Cash & Carry. Для «Азбуки Вкуса», «ВкусВилла», «Ленты» и «Пятерочки» для этой цели необходимо установить отдельное приложение. «Globus – Гипермаркеты Глобус» имеет ссылку на сайт интернет-магазина, который открывается в браузере. Подробнее о приложениях для заказа продуктов читайте здесь.

Онлайн-заказ с самовывозом из магазина. Немного отличается картина с онлайн-заказом с самовывозом из магазина. Оформить такой заказ полноценно позволяют шесть приложений: «Ашан», «ВкусВилл», «Красное&Белое», «Мой Спар», «О'Кей» и Metro Cash & Carry. Для «Азбуки Вкуса» и «Пятерочки» для этой цели необходимо установить отдельное приложение. В приложении «Globus – Гипермаркеты Глобус» имеется ссылка на сайт интернет-магазина, который открывается в браузере.

Сканер штрихкодов реализован в пяти приложениях: «ВкусВилл», «Дикси», «О'Кей», «Globus – Гипермаркеты Глобус» и Metro Cash & Carry. Соответствующая функциональность в приложении «Ашан» работает только в магазинах с системой самостоятельного сканирования товаров. По аналогичной схеме функция реализована в «Пятерочке», однако, для того чтобы ею воспользоваться, необходимо скачать отдельное приложение. В приложении «Мой Перекресток» сканер штрихкодов для проверки цен работает для категорий «Виски» и «Вино» во всех магазинах сети, а в магазинах с системой самостоятельного сканирования товаров – для всех товаров, через отдельное приложение. В приложении «Красное&Белое» при помощи сканера штрихкодов можно узнать информацию о товарах, но не их цену.

Сканирование через камеру смартфона. Система самостоятельного сканирования товаров при помощи камеры смартфона реализована в приложениях «ВкусВилл», «Ашан», «Перекресток. Сканер в телефоне» и «Пятерочка с собой». Система позволяет отсканировать камерой смартфона товары, которые вы берете с полки, а на кассе после сканирования QR-кода сразу оплатить покупку. Это существенно экономит покупателям время, а также позволяет оценить, сколько стоит набранная корзина. В некоторых торговых сетях перед началом покупок необходимо также отсканировать стартовый QR-код. При помощи ручных сканеров, которые можно взять на входе в супермаркет, данная система реализована в «Ленте», «О'Кей», «Globus – Гипермаркеты Глобус» и Metro Cash & Carry.

Стоит отметить, что на момент исследования система самостоятельного сканирования товаров реализована не во всех, а лишь в некоторых магазинах торговых сетей.

Из выше описанного анализа видно, что функционал приложений специализирующееся на анализе гипоаллергенных продуктов в супермаркетах является не развитым среди популярных специализированных мобильных приложений. Работа со штрихкодами как элемент анализа и верификации продуктов также присутствует не во всех выше перечисленных приложениям.

Отсюда можно сделать вывод, что разработка мобильного приложения которое позволит потребителю выбрать адекватный для него товар является актуальной темой.

1.3. Требования к разрабатываемому программному обеспечению

Реализуемое приложение удовлетворяет следующим функциональным требованиям:

Мобильный доступ. Пользователи могут получить доступ к приложению через мобильные устройства. Мобильный доступ является неотъемлемой частью системы, поскольку он поддерживает работу со штрих-кодом товара и предоставляет и данные о геолокации пользователя и магазина, которые необходимы для создания связи продукт-магазин.

Управление пользователями. Для работы приложения требуется, чтобы пользователи регистрировались в системе, т.к. приложение собирает данные пользователя, товаров и магазинов. Пользователи настраивают приложение со списком аллергенов, которые они хотят избежать. В настоящее время система поддерживает 6 аллергенов: молоко, яйцо, арахис, соя, миндаль и морепродукты. Пользователи должны выбрать аллергены из этого списка при регистрации.

Проверка продукта. Приложение позволяет пользователям проверять безопасность продукта на основе информации о продуктах взывающих

аллергию. Пользователь инициирует процесс проверки продукта путем сканирования штрих-кода продукта. Штрих-код предоставляет приложению UPC-код продукта, который используется для получения списка ингредиентов продукта из службы API. Приложение определяет безопасность продукта, сверяя список ингредиентов продукта с аллергенами, указанными в профиле пользователя. Информация о продукте, полученная из API, также сохраняется в базе данных приложения, размещенной на сервере, поэтому последующие проверки на тот же штрих-код не требуют вызовов API.

Сбор геолокационных данных. Приложению необходимо знать, где продаются продукты, чтобы оно могло информировать пользователей. Каждый раз, когда пользователь просматривает товар в магазине; приложение выбирает местоположение пользователя и создает запись, отображающую товар в магазине. Таким образом, он создает базу данных сопоставления, которая предоставляет ему информацию о том, какие продукты находятся в каких магазинах.

Данные, собранные от пользователей, содержат географическое местоположение магазина и штрих-код продукта. API Google (сервисы API карт Google) предоставляет приложению информацию о магазине, а API Google меток предоставляет данные, относящиеся к продукту.

Информация о магазине состоит из названия магазина, адреса, географического местоположения и идентификатора места на карте Google Maps. Каждый магазин уникально идентифицируется по названию и адресу. Информация о продукте состоит из названия продукта, торговой марки, описания, кода UPC и списка аллергенов, которых продукт содержит или не содержит. Каждый продукт уникально идентифицируется по коду UPC.

Найти продукт. Система позволяет пользователям находить продукты, используя название продукта. В результате получается поиска товара пользователь получает список магазинов, в которых находятся товары. Места расположения магазинов отсортированы по алфавиту. Пользователи могут видеть местоположение каждого магазина на карте Google.

Поиск магазина. Поиск магазина позволяет пользователям видеть магазины в базе данных приложегния, в которых хранятся продукты, безопасные для них. Результат сгруппирован по брендам магазинов. Каждый результат для бренда показывает общее количество уникальных продуктов и общее количество мест, в которых эти продукты находятся. Пользователи могут просмотреть результаты, чтобы увидеть адреса магазинов и сколько продуктов продается в каждом из них.

Дополнительные требования. Данные требования заключаются в том, чтобы создать производительное, масштабируемое и гибкое приложение, которое может адаптироваться к изменениям с минимальными доработками системы. В систему будут добавлены новые аллергены, а характер и объем данных будет меняться с ростом использования приложения. Набор функций будет изменяться в зависимости от потребностей пользователей. Система также может поддерживать новые интерфейсы и/или подвергать изменениям части backend системы. Эти изменения могут быть отражены при проектировании системы.

Следовательно, система должна быть построена как модульная, надежная, расширяемая, гибкая и масштабируемая система, в которой компоненты слабо связаны, а каждый модуль реализован по принципу открытого проектирования.

Рисунок 1.3 Диаграмма требования к функционалу разрабатываемого программного обеспечения.

1.4. Требования к системе

Система состоит из независимых клиентского и серверного модулей, которые взаимодействуют друг с другом через RESTful API. Backend система размещается в Amazon EC2 (Elastic Compute Cloud (EC2)), в то время как клиентская система представляет собой мобильное приложение, которое предоставляет пользовательский интерфейс приложения и зависит от серверной части для всех потребностей.

Данные хранятся в СУБД MongoDB, которая ориентированна на базы данных NoSql. Система зависит от сервисных API, чтобы найти продукт и сохранить информацию. API сервисов Google (веб-сервисы API Google Maps) предоставляет информацию о магазине и местоположении продуктов, а API

Label будет обрабатывать информацию о продукте. На рисунке 1.4 показана общая системная архитектура приложения.

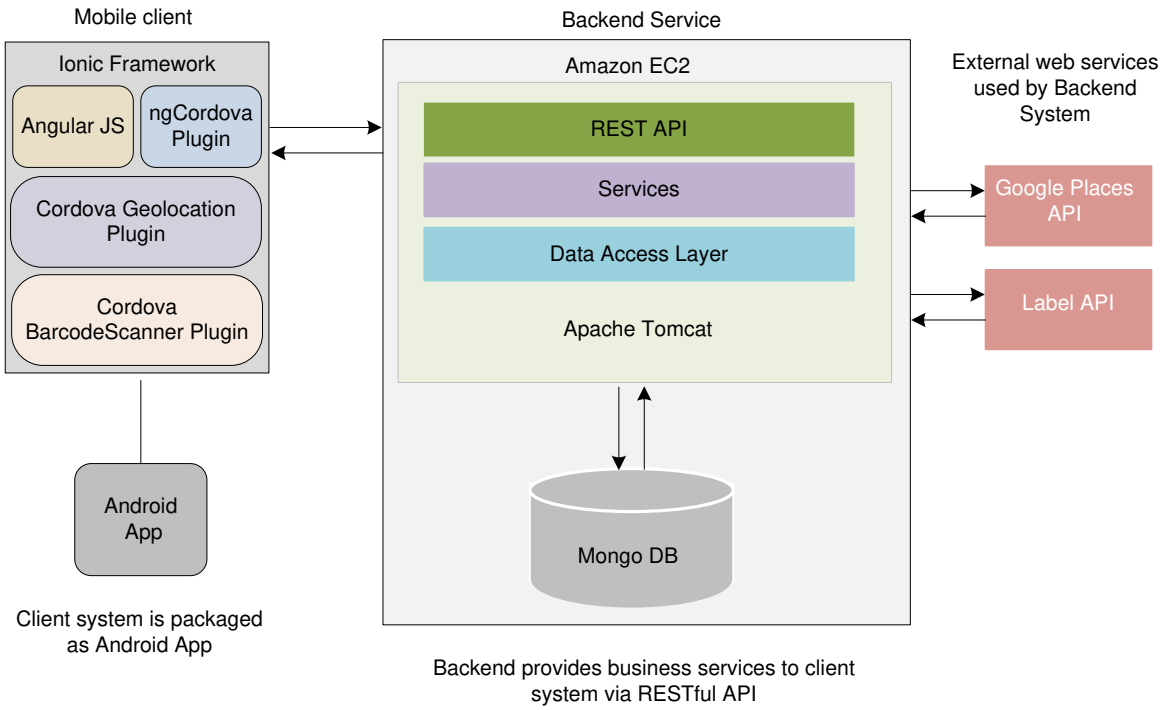


Рисунок 1.4 Архитектура реализуемой системы

Мобильное приложение построено на платформе Ionic и развернуто на платформе Android. Подключаемые модули мобильного приложения ngCordova, Cordova BarcodeScanner и Cordova GeoLocation используются для обеспечения необходимого доступа разрабатываемого приложения к аппаратному API мобильного телефона. Вызовы REST к backend системе облегчаются модулем RESTFullAPI Angular, который предоставляет механизм для отправки HTTP-запросов от клиента серверу.

Сервер (backend системы) - Java-приложение, которое развертывается на веб-сервере Apache Tomcat. Контейнер сервлетов Apache Tomcat поддерживает слой взаимодействия бизнес логики и данных приложения. Сервер состоит из модуля API RESTful, модуля бизнес логики и модуля службы доступа к данным. Модуль RESTful API предоставляет интерфейсы для взаимодействия севера с мобильным клиентом. Модуль бизнес логики предоставляет всю бизнес-логику, включая проверки данных и создание

записей сопоставления продуктов с магазинами. Модуль службы доступа к данным отвечает за всю логику работы с СУБД.

1.5. Сравнение и выбор средств разработки

Успех мобильного приложения во многом зависит от выбора технологий при его разработке. Именно они предопределяют жизнеспособность и конкурентоспособность программного обеспечения, его функционал, масштабируемость и сложность обслуживания. Также технический стек влияет на стоимость и скорость разработки. В этом разделе будут рассмотрены подходы к разработке мобильных приложений и технологии, которые разработчики смогут использовать для реализации проекта.

1.5.1 Сравнение платформ Android и iOS

Android составляет 86% мирового рынка. По прогнозам эта тенденция будет сохраняться в будущем. Если вас интересует максимально широкая аудитория, то это один из тех фактов, который имеет значение.

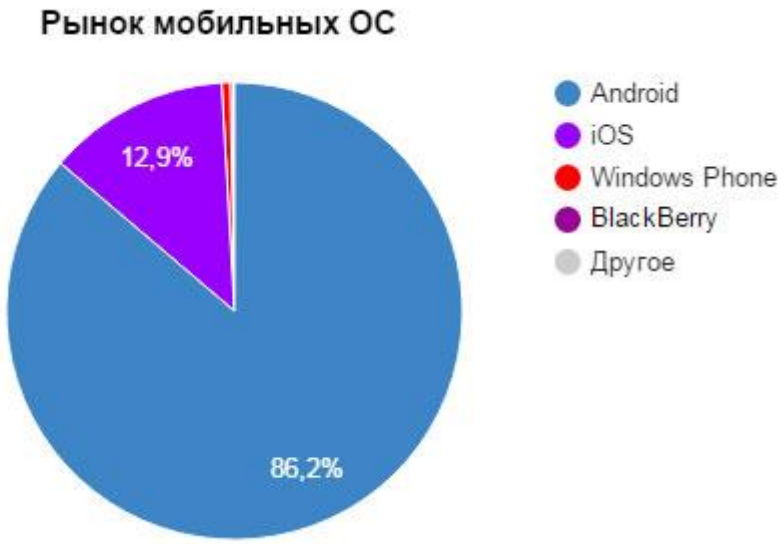


Рисунок 1.5 Процент использования Android и iOS.

Обновление платформ. Android содержит множество специализированных прошивок, регулярные исправления, возможность получать права суперпользователя (полная свобода действий). Вывод: на этапе разработки понадобится учесть все особенности версий, которые были до этого. А с каждым обновлением проверять мобильное приложение на работоспособность.

Приложение для IOS не содержат сторонних прошивок, обновления выходят редко, но это зрелые релизы без ошибок. Вывод: часто с обновлением iOS приложения просто перестают работать или появляются критические ошибки, в связи, с чем приходится их обновлять в экстренном режиме.

Интерфейс. В плане интерфейсов iOS и Android отличаются, в основном, из-за стиля описания элементов Apple и Google. В линейке iOS навигация в iPhone X отличается от других смартфонов компании Apple.

В целом, все производители и разработчики стараются приходить к общему интерфейсу для узнаваемости приложения, но всё же стиль описания элементов диктует свои правила, из-за которых отличий iOS и Android не избежать.

Язык программирования. Вариантов инструментов разработки достаточно для обоих вариантов. Есть кроссплатформенные решения React Native, Xamarin, PhoneGap и прочие, основанные на WebView. Наиболее изветсные средства разработки для iOS - это Swift и Objective-C, для Android - Java или Kotlin.

Из вышесказанного можно сделать следующий вывод в пользу платформы Android:

- Разнообразие устройств. На Android работают не только смартфоны и планшеты, но и телевизоры, холодильники, автомобильные аудиосистемы, причем от разных брендов. Соответственно, и спектр использования разработок может быть широкий.

- Для отладочной работы можно использовать компьютер. Неважно, на основе Windows, Linux или Mac. Android Studio и программы Android SDK доступны для всех платформ.
- Сообщество разработчиков. Сообщество разработчиков под Android шире чем у iOS. Для новичка это весомый плюс: проще найти помощь на форумах, получить информацию в чатах.
- Возможность адаптировать платформу под специализированную задачу без потери поддержки. Если разрабатывать приложение под какую-то специализированную задачу, то Android подходит для этого лучше. Например, недавно с введением ОФД появилась задача адаптации кассовых аппаратов терминалов для проведения оплаты. Компания «Эвотор» для этого взяла платформу Android и создала приложение для работы. Для этого они отказались от поддержки Google, убрали Market. То есть полностью кастомизировали платформу. И даже если Google уйдет из России, то эти кассовые аппараты будут работать. С iOS такого бы не получилось.
- Среда разработки. Среда разработки Android Studio основана на популярной в мире среде Intellij Idea. В этой среде есть умный анализ кода, удобная.

1.5.2 Сравнение и выбор архитектуры приложения

Разработчики могут выбрать один из трех вариантов мобильных приложений: нативные, веб или гибридные. Давайте проанализируем каждый тип, рассмотрим преимущества и недостатки, и определимся, какой из них будет лучшим выбором в той или иной ситуации и лучше всего удовлетворит потребности при создании мобильного приложения.

Под нативным подразумеваем мобильное приложение, которое создается для определенной платформы и непосредственно устанавливается на устройство пользователя (занимая определенный объем памяти). Такие

приложения пользователь загружает через магазин приложений той или иной платформы, такой как Play Store для Google и Apple App Store для iOS.

С нативными приложениями разработчики могут изготовить приложение согласно индивидуальным запросам, чтобы затем пользователю было удобно им пользоваться, в дополнение к веб-сайту. Эта целостность и является существенным преимуществом нативных приложений.

Другие важные преимущества нативных приложений:

- обозначение геолокации. Потребители могут получать уведомления, когда они находятся возле физических магазинов;
- данные действий (или бездействия) пользователя могут быть легко собраны и проанализированы, таким образом облегчая оценку эффективности всего приложения либо его отдельных функций;
- нативные приложения, как правило, работают лучше web-приложений, которые иногда создаются для имитации нативных.

Недостатки нативных приложений:

- нативные приложения дорогие в разработке, особенно при разработке приложения на кроссплатформенных ОС;
- нативные приложения должны быть одобрены каждым магазином приложений, а процесс привлечения внимания к нему пользователей может быть сложным.

Веб-приложения работают через веб-браузер на устройстве пользователя. Эти приложения по сути являются индивидуализированными веб сайтами, которые сделаны таким образом, чтобы выглядеть и использоваться как нативные приложения, но на самом деле они не находятся на устройстве пользователя. Их можно сравнить с облачным хранилищем в сравнении с данными, которые хранятся на сервере. При хорошей, качественной разработке, которая включает в себя подбор размеров и прокрутку, веб-приложения часто работают подобно нативным приложениям.

Ключевые преимущества веб-приложений:

- приложения на веб-основе легче поддерживаются и они могут функционировать на платформе с любой ОС;

- разработчики могут предлагать приложения без необходимости их утверждения какими-либо магазинами приложений;

- более быстрая разработка циклов с использованием CSS, HTML и JavaScript.

Основные минусы:

- у веб-приложений нет доступа к устройству пользователя. Несмотря на то, что иногда было бы удобно, это ограничивает многие функции, которые используются в нативных приложениях для более персонализированного использования;

- пользователи должны использовать их через сеть, что значительно снижает контроль безопасности;

- поиск приложения может быть затруднительным, так как не существует магазина приложений с каталогом и функцией поиска в нем.

Гибридные приложения являются чем-то средним между нативными и веб-приложениями. Фактически они создаются так, чтобы выглядеть и использоваться как нативные приложения. Их также устанавливают на телефон пользователя и их можно найти в магазинах приложений. Различие заключается в том, что они обязательно должны размещаться в рамках нативного приложения и созданы, чтобы работать через WebView, и таким образом они могут получать доступ к информации на устройстве пользователя для больших возможностей.

Дополнительные преимущества гибридных приложений:

- гибридные приложения обладают наибольшей функциональностью и персонализацией для пользователя;

- разработчики не ограничены одной платформой, вместо этого они могут создать гибридное приложение, которое будет работать с несколькими платформами;

- гибриды опция для разработчиков, которые создают визуально насыщенные приложения.

В данном проекте приложение будет реализовано в рамках гибридной архитектуры.

1.5.3 Сравнение языков программирования

Среди фреймворков-лидеров можно выделить два наиболее популярных фреймворка: Ionic 2 и React Native. Далее рассматривается каждый из них по отдельности, а также выполняется их сравнение.

Ionic Framework это фреймворк для создания гибридных мобильных приложений. В его состав входит набор JavaScript и CSS компонент, созданных на основе Angular 2, SASS и Apache Cordova. Ionic Framework содержит инфраструктуру для написания гибридных приложений, которая будет ориентирована на производительность и будет построена с использованием современных веб-стандартов.

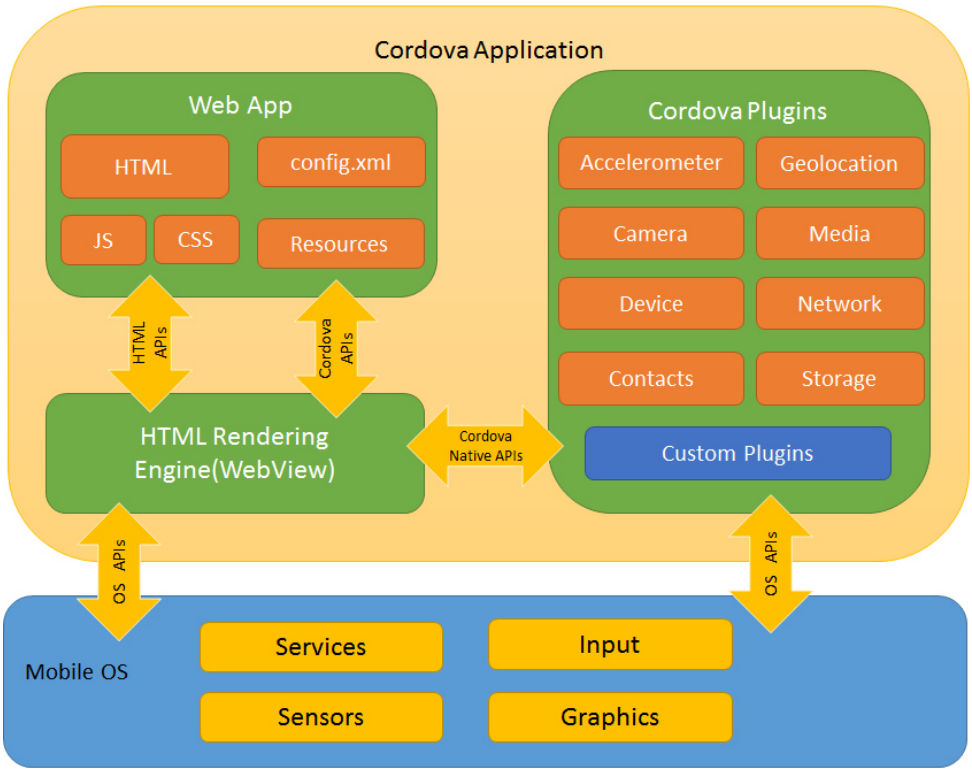


Рисунок 1.5 Архитектура приложения на Apache Cordova.

По своей сути, Ionic Framework это дополнение над очень популярным фреймворком Apache Cordova, но со своим мощным CLI (Command Line Interface) и объемной документацией. Следуя принципам Apache Cordova, приложения на Ionic Framework это гибридные HTML приложения. Такие приложения на телефоне выполняются в специальной оболочке (UIWebView для iOS и WebView для Android), которая позволяет показывать HTML и выполнять JavaScript. Соответственно при работе в приложении пользователь работает как бы в веб-браузере.

Apache Cordova является своеобразной прослойкой между пользовательским интерфейсом и ресурсами устройства. Какие-либо виджеты пользовательского интерфейса или MV* (Model-View) фреймворков не входят в его состав. Если необходимо использовать UI-виджеты и/или MV* фреймворк, то нужно выбрать и включить их в приложение самостоятельно, как ресурсы третьей стороны. Ionic 2 это один из фреймворков, который предоставляет UI виджеты. MVC модель для него предоставляет Angular 2.

Нативные функции телефона (например, камера, хранилище ключей, GPS координаты) недоступны из веб-браузера. Поэтому для работы с ними используются плагины Apache Cordova. Кроме официальных плагинов есть ряд сторонних открытых плагинов.

React Native это JavaScript фреймворк, разработанный компанией Facebook для создания нативных мобильных приложений с использованием JavaScript. Фреймворк построен на основе библиотеки ReactJS. React Native использует JavaScript API поверх нативных компонентов. При создании приложения, пишется JavaScript код, который работает с нативными компонентами операционной системы. То есть, React Native использует те же самые фундаментальные стандартные блоки UI что и обычные приложения iOS и Android, не используя при этом ни браузер, ни WebView/UIWebView.

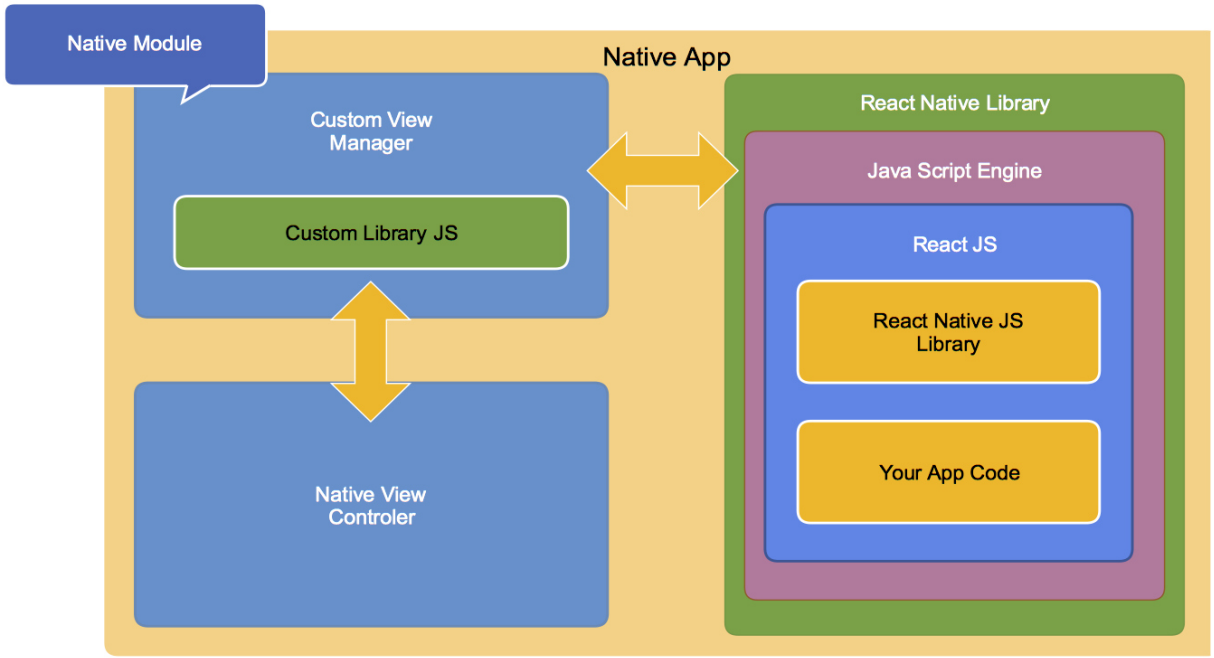


Рисунок 1.6 Архитектура приложения на React Native.

Приложение на React Native использует отдельный поток для выполнения JavaScript в устройстве. Этот поток взаимодействует с нативным кодом. Важно отметить, что нативные компоненты (включая элементы пользовательского интерфейса) отличаются для разных операционных систем. И, соответственно, для каждой операционной системы необходимо создавать свои «bridges» между JavaScript кодом и этими компонентами. Ряд таких компонент предоставляет Facebook, а также существует много компонент в свободном доступе, разработанных сообществом.

Ionic 2 и React Native позволяют создавать мобильные приложения. Принципиальная разница между этими фреймворками заключает в типе мобильного приложения, которое создается с использованием того или иного фреймворка.

Ionic 2 позволяет создавать гибридное мобильное приложение, которое осуществляет рендеринг в WebView/UIWebView, при необходимости использовать API платформы нужно использовать дополнительные плагины.

React Native позволяет создавать нативное мобильное приложение, используя фундаментальные стандартные блоки UI (как и обычные

приложения iOS и Android, написанные на родном языке — Objective-C/Swift и Java соответственно). В приложении на React Native используются определенные шаблоны заданные самой платформой.

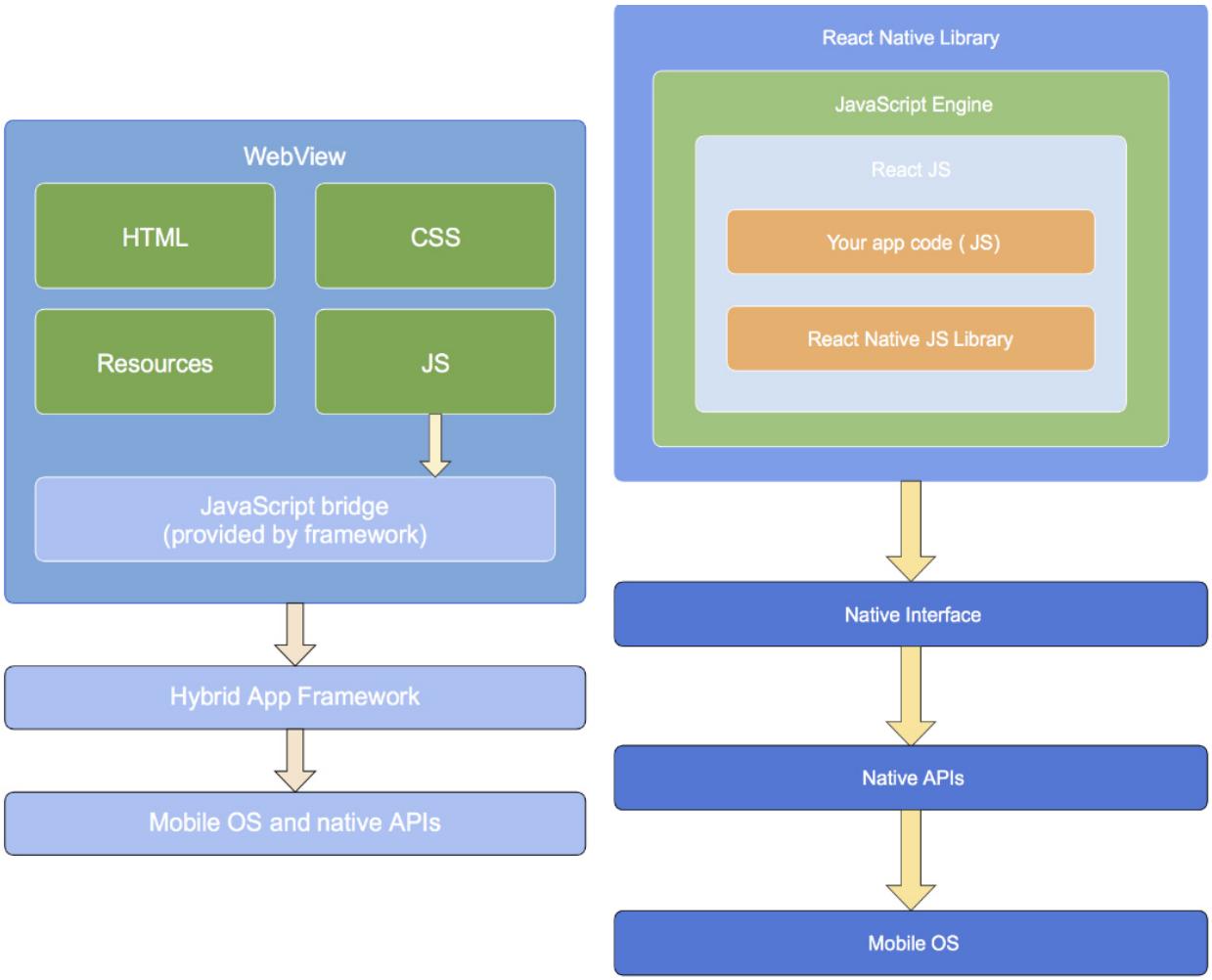


Рисунок 1.7 Выполнение кода и связь с нативным API в приложениях, написанных на Ionic 2 (слева) и React Native (справа)

Схема показывает составляющие кода приложения и вызов нативного API из приложений. В случае Ionic 2 отрисовку интерфейса пользователя выполняет браузер, для React Native это делает непосредственно операционная система.

В целом можно рекомендовать использовать Ionic 2 для быстрой разработки прототипов мобильных приложений, а React Native для полноценной разработки готовых решений. Ниже представлен список

преимуществ и недостатков каждого фреймворка, основанный на опыте использования Ionic 2 и React Native.

- Преимущества Ionic 2:
- известный набор инструментов для frontend разработчика. Набор используемых технологий является знакомым, это существенно сокращает время на выполнение поставленной задачи;
- используя шаблоны приложений предоставленные Ionic, можно за короткое время создать прототип для показа заказчику;
- написанное приложение работает на всех популярных платформах, это дает кросс-платформенность с незначительными изменениями кода.

Недостатки Ionic 2:

- изменение стилей по-умолчанию для их приведения в соответствии с макетом значительно увеличивает затраты времени на создание и тестирование приложений на разных платформах и устройствах;
- использование браузера приводит к более длительной загрузке приложения и плохой отзывчивости при нагрузке процессора из-за использования браузерной оболочки для имитации поведение компонентов каждой платформы;
- при возникновении нативных событий блокируется главный поток и передается управление на JavaScript-код, ожидая его инструкций, что может приводить к непредсказуемой работе нативных функций.

Преимущество React Native:

- стандартные нативные блоки UI. Использование фундаментальных стандартных блоков UI, что и обычные приложения iOS и Android, знакомые пользователю по другим нативным приложениям, облегчает взаимодействие пользователя с приложением;
- JavaScript выполняется в отдельном фоновом потоке, взаимодействуя с главным потоком асинхронно;

- приложение одинаково хорошо выглядит на разных платформах и устройствах;
- проведенные тесты использования приложением CPU и памяти показывают лучшие результаты по сравнению с Ionic 2;

Недостатки React Native:

- необходимость разбираться в компонентах, написанных на ObjectiveC/Swift или Java;
- сложность старта (без опыта работы с ReactJS) необходимо дополнительное время на знакомство React, JSX и отличающимся подходом написания (компоненты, как самостоятельные единицы интерфейса, которые сами отвечают за свое состояние и поведение).

2. Разработка приложения

Для реализации серверной части приложения были выбраны Java и webсервер Tomcat 8, поскольку эти технологии обеспечивают кроссплатформенную инфраструктуру для создания объектно-ориентированных веб-приложений.

Java является очень надежным языком, и Apache Tomcat является одним из самых популярных веб-серверов с открытым исходным кодом на сегодняшний день. MongoDB была выбрана в качестве системы управления базами данных вместо MySql, как было изначально предложено, поскольку MongoDB более гибок и масштабируем по своей природе, чем MySql.

Morphia (Morphia), структура отображения, которая отображает объекты Java на документы MongoDB, используется в качестве системы объектно-реляционного отображения (ORM) для сохранения моделей данных в MongoDB.

На рисунке 2.1 показана диаграмма структурированной модели данных приложения. Поскольку в настоящее время приложение предлагает лишь несколько функций эта модель проста, но эта структурированная модель может легко стать сложной, поскольку в систему добавляются новые функции.

На рисунке 2.2 показано, как три независимых внутренних модуля связаны друг с другом. Каждый модуль предоставляет свой сервис через интерфейс, т.к. детали реализации не предоставляются вызывающему модулю. Модули RESTful API и Business Service имеют компоненты отображения объектов данных, которые отображают объекты данных между модулями. В качестве средства отображения данных была выбрана библиотека Dozer.

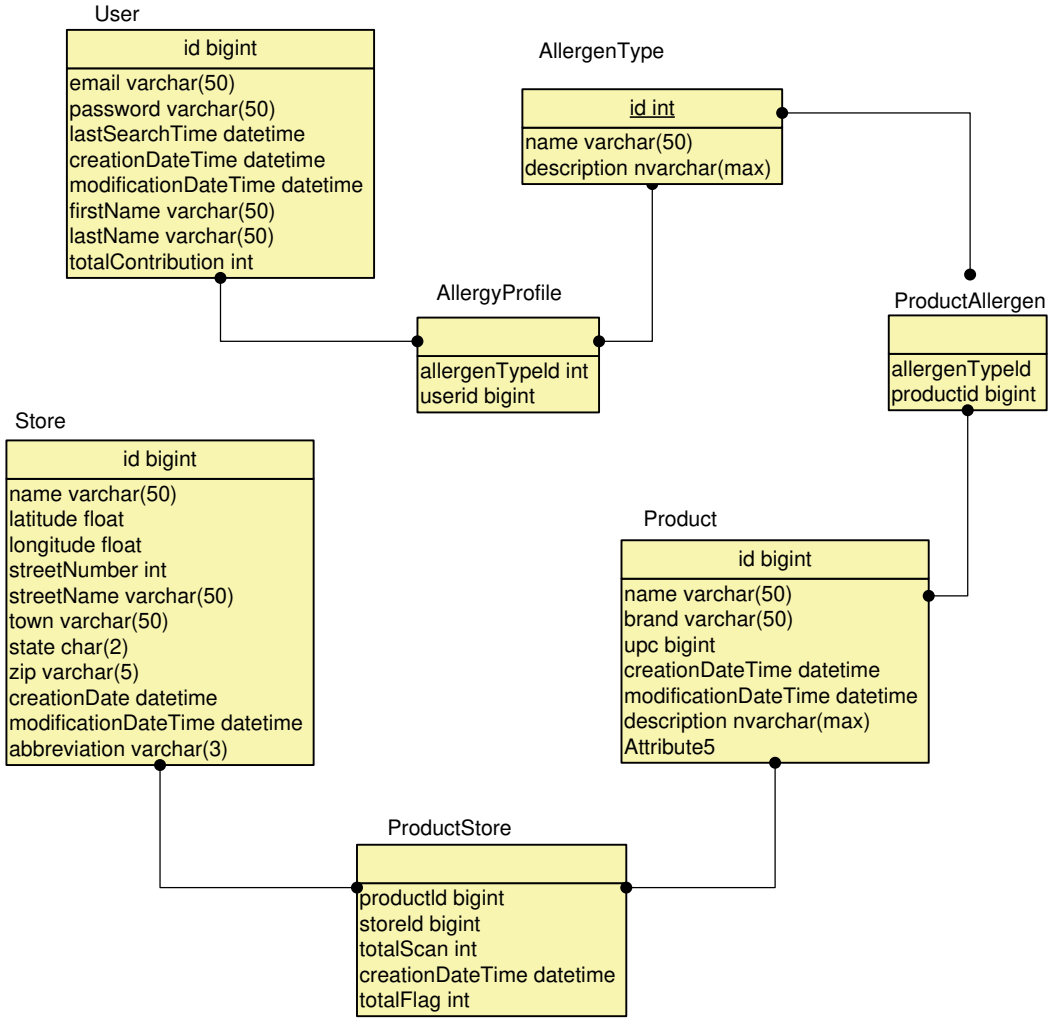


Рисунок 2.1 Модель данных приложения.

2.1 Реализация бизнес логики приложения

Модуль RESTful API реализован в библиотеке Jersey, которая является стабильной и популярной средой для реализации веб-сервисов RESTful на основе Java. Jersey обеспечивает основу для реализации веб-сервисов. Jersey поддерживает формат JSON, который является форматом обмена данными приложения. Библиотека Jersey поддерживает преобразования классов Java в веб-сервисы и имеет надежный клиентский API для работы с клиентами. Модуль Jersey подходит для реализации backend с реализацией интерфейсов RESTful.

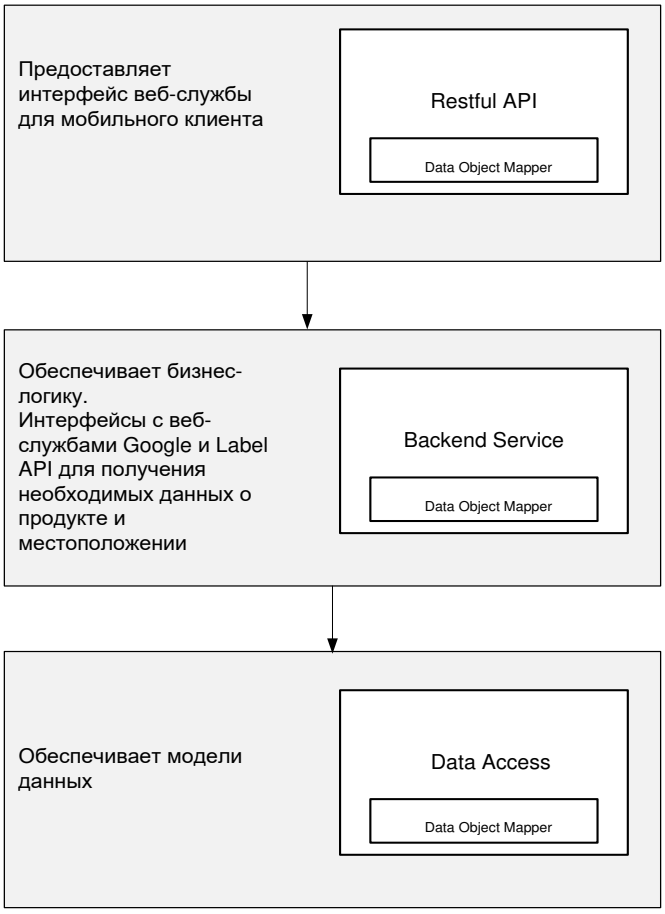


Рисунок 2.2 Реализация mapping механизма в системе.

API служб Google, используемый в backend сервисе, включает в себя API Google Places, API Matrix Distance Google Maps и API Google Maps. Google Places API предоставляет информацию о магазине в backend сервисе. API Matrix Distance Google Maps рассчитывает расстояние между текущим

местоположением пользователя и магазинами, а API Google Maps предоставляет логику для отображения местоположения магазина на карте Google.

API Label предоставляет информацию о продукте: список ингредиентов, название, торговая марка и описание, в backend сервисе.

Label API это база данных о продуктовых продуктах, заполненных пользователями системы, и предоставляет доступ к данным через интерфейс REST.

Диаграмма классов UML на рисунке 2.3 показывает, как компоненты в системе связаны друг с другом.

Основным функционалом, который отличает приложение от анаголов, является его способность находить продукты, чтобы информировать пользователей, где делать покупки. Разрабатываемое приложение предоставляет функциональность, создавая записи сопоставления продуктов для магазинов, которые сопоставляют продукты с магазинами, которые их продают, используя краудсорсинг.

Краудсорсинг использует входные данные от пользователей приложения для генерации данных. В приложениях с краудсорсингом пользователи становятся активными поставщиками данных в систему, а не просто ее потребителями.

Сканирование данных пользователем инициирует процесс создания записей СУБД, так как таким образом система может получить информацию, такую как код UPC продукта и текущее географическое местоположение пользователя, которые необходимы для создания записи СУБД. Данные сканирования передаются серверной части приложения, которая использует сервисы хранения информации. СУБД использует код UPC для поиска информации о сканируемом продукте и использует данные о географическом местоположении для поиска магазина, из которого был отсканирован продукт. Как только система находит эту информацию, она создает запись, отображающую товар в магазине.

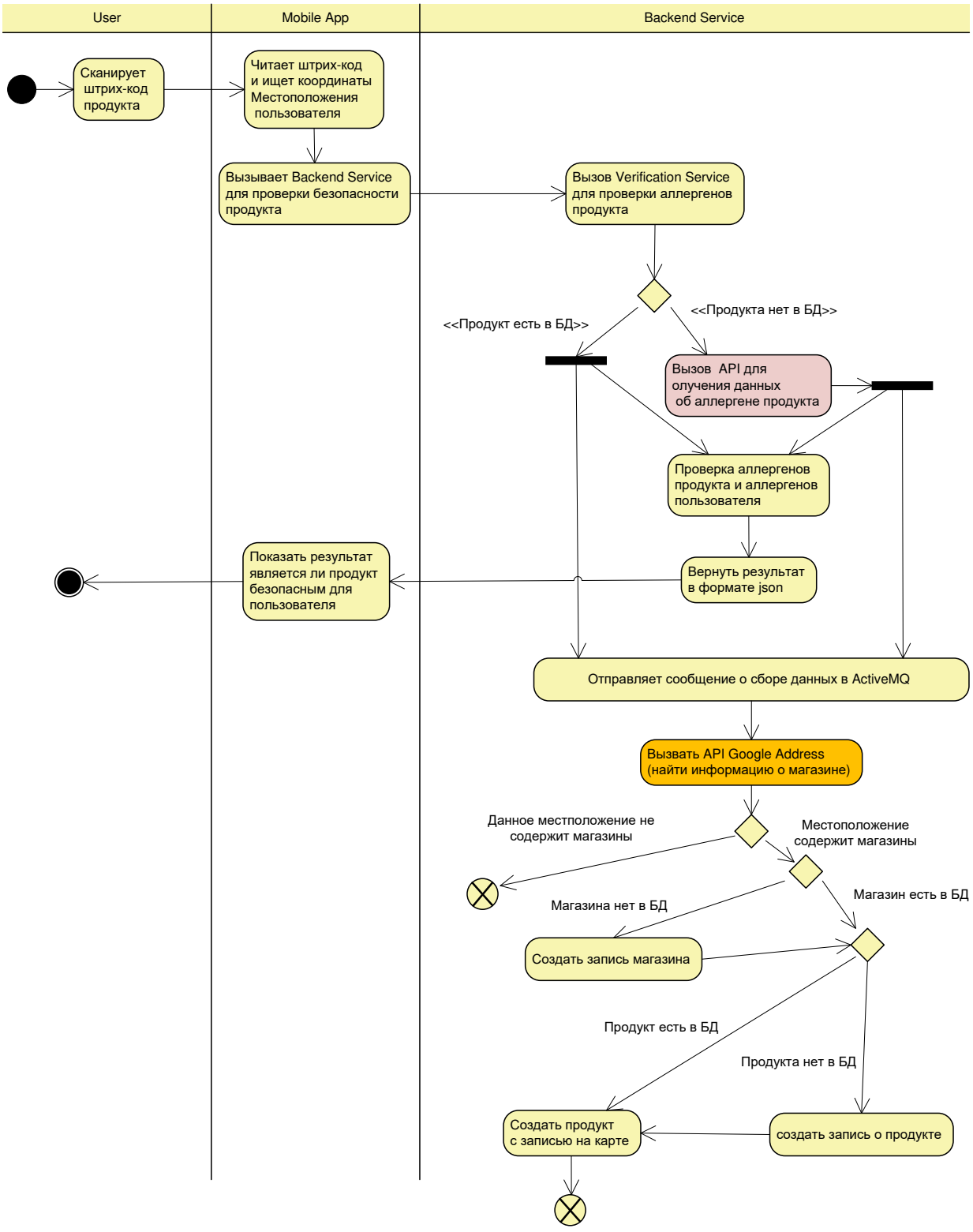


Рисунок 2.3 Связь между компонентами системы.

Модуль RESTful API предоставляет сервисный интерфейс для мобильного клиента. Он зависит от модуля BusinessServiceModule бизнес-логики и реализует свой собственный набор объектов данных, которые

совместно используются клиентом. Модуль BusinessServiceModule разделяет интерфейс на службы ProductController и UserController.

Интерфейсные классы определяются как веб-сервисы посредством аннотаций на Jersy. Аннотации Jersy @Path на классах идентифицируют пути URI, которые обслуживают классы.

Используя аннотации, ProductController сопоставляется с путем «/product», а UserController сопоставляется с путем «/user». Публичные методы в классах также снабжены аннотациями @Path для определения путей URI ресурсов. Методы также включают аннотации HTTP-заголовков для определения типа HTTP-запросов (GET, POST, PUT). Сервлет Jersey, работающий в контейнере сервера Tomcat, получает все запросы веб-службы и направляет эти запросы в соответствующие классы и методы на основе определенных путей URI.

Аннотации @Consumes и @Produces используются для определения формата обмена данными.

Интерфейс UserController предоставляет следующие методы для поддержки функций управления пользователями.

createUser: мобильный клиент использует этот метод для регистрации нового пользователя в системе. Метод принимает объект AppUser в качестве параметра, который содержит всю информацию, необходимую для регистрации нового пользователя. Ответ содержит вновь созданный объект User, если вызов завершается успешно.

validateUser: мобильный клиент использует этот метод для проверки пользователей при входе в систему. Метод принимает две строки: имя пользователя и пароль. Ответ содержит причину сбоя, если проверка не пройдена.

updateUser: мобильный клиент использует этот метод для обновления данных объекта User.

Интерфейс ProductController предоставляет следующие методы для проверки и определения местоположения товара.

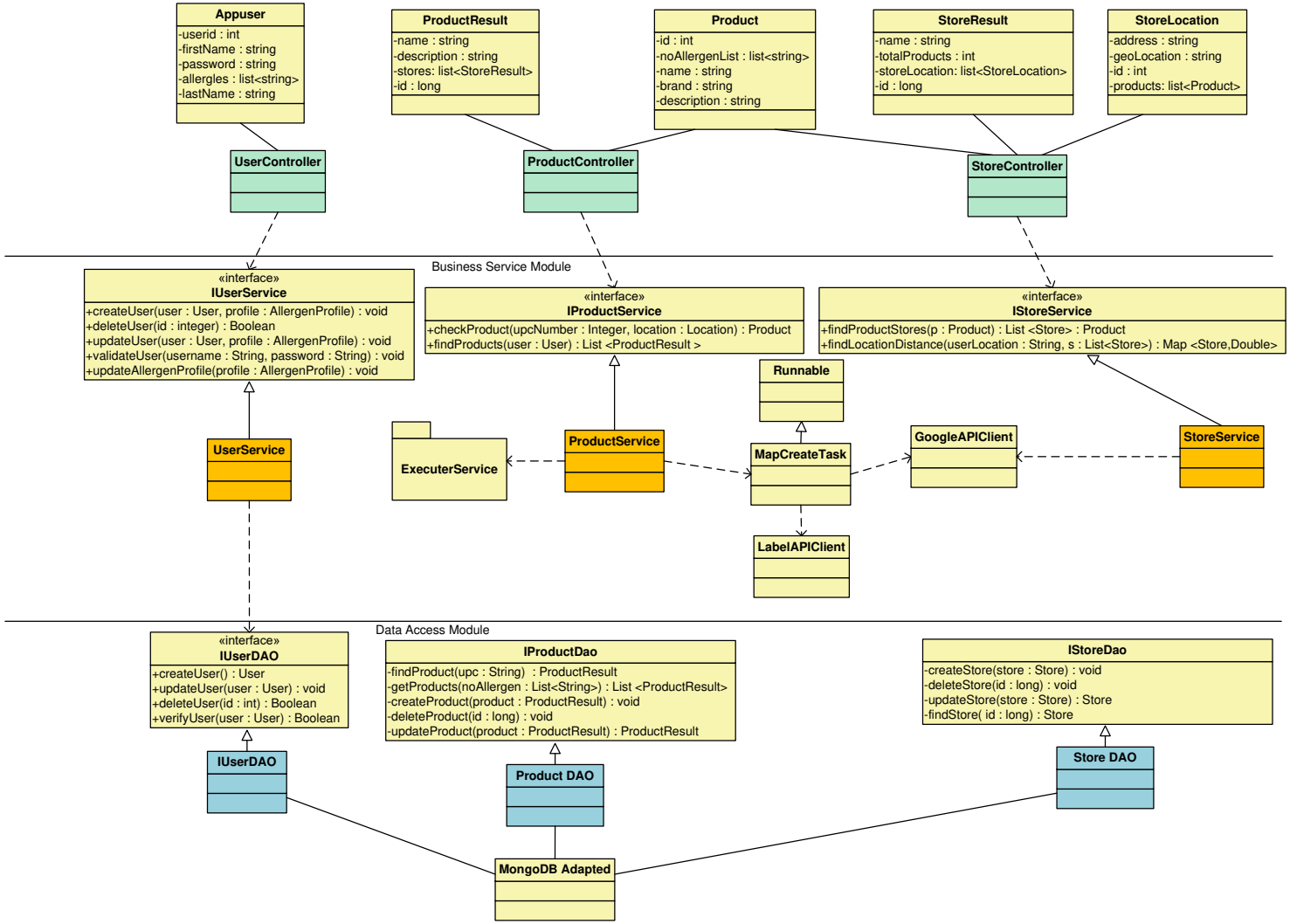


Рисунок 2.4 Модуль RestFullAPI серверной части приложения.

checkProduct: мобильный клиент использует этот метод для проверки безопасности продукта. Вызывающий объект класса предоставляет код UPC отсканированного продукта и список аллергенов пользователя. Контроллер вызывает модуль BusinessService для получения продукта по коду UPC и сравнивает его список аллергенов с аллергенами пользователя для проверки безопасности продукта. модуль BusinessService возвращает объект AllergenCheckResult, который содержит результат, а если результат отрицательный, то возвращается значение null.

findProducts: этот метод принимает noAllergen в качестве параметра, который содержит список аллергенов, и возвращает все продукты в базе данных приложения, которые не содержат аллергенов, перечисленных в параметре. Метод также извлекает записи сопоставления магазина для каждого продукта в результате, используя интерфейс StoreService, и объединяет результаты по бренду магазина. Мобильный клиент использует этот метод, чтобы показать все продукты, которые пользователь может безопасно использовать, а также магазины, в которых эти продукты находятся.

locateProduct: мобильный клиент использует этот метод для поиска определенного продукта по поисковому запросу. Метод принимает текущие данные о географическом местоположении пользователя в качестве параметра. Модуль возвращает местоположения магазинов, в которых имеются товары, чье название, описание или название соответствуют строке поиска. Результат также содержит расстояние от текущего местоположения пользователя до каждого местоположения магазина в результате. Контроллер использует интерфейсы StoreService и ProductService для получения необходимых данных.

Модуль BackendService реализует интерфейсы бизнес-сервисов для модуля RESTful API. Он предоставляет различные интерфейсы объектов Product, Store и User. Модуль BackendService зависит от модуля службы доступа к данным и реализует собственный набор моделей данных. На

рисунке 2.5 представлена диаграмма, которая показывает интерфейсы и классы, реализованные в модуле BusinessService.

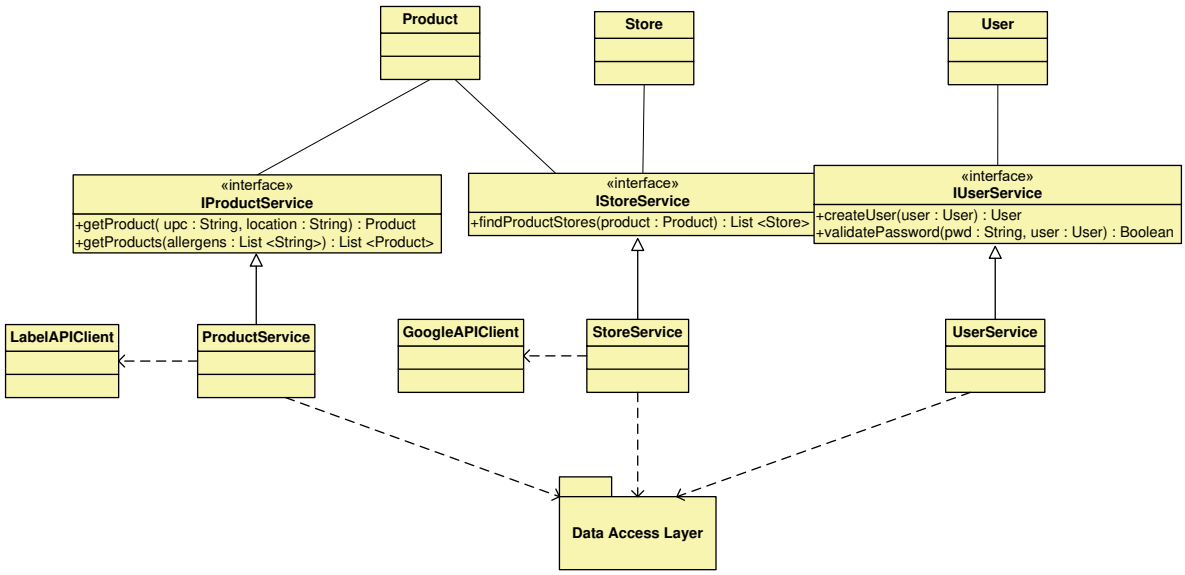


Рисунок 2.5 Модуль BusinessService серверной части приложения.

Модуль BackendService реализует классы LabelAPIClient и GoogleAPIClient в качестве клиентов веб-служб, использующих клиентскую библиотеку Jersey.

LabelAPIClient управляет вызовами службы к службе API Label, а GoogleAPIClient вызывает службы API Google. Вызовы API необходимы для использования этих служб, поэтому приложение должно сначала зарегистрироваться в этих службах и получить ключи API, прежде чем оно сможет выполнять вызовы служб.

LabelAPIClient использует ресурсы «label» и «searchprods» для получения информации о продукте из службы Label API. Оба метода принимают код UPC в качестве параметра и возвращают информацию другого типа, связанную с продуктом. Ресурс «label» возвращает список аллергенов вместе с их значениями аллергенов для соответствующего продукта. Значение аллергена представляет собой число, представляющее, какой вес имеет конкретный аллерген в продукте.

Например, если у аллергена allergen_value = 0, это означает, что продукт не содержит аллергена. LabelAPIClient использует этот результат для создания

списка «noAllergen» для продукта. Ресурс «searchprods» предоставляет подробную информацию о продукте, включая название, марку и описание системы.

GoogleAPIClient использует ресурс «place/nearsearch» для поиска информации о местоположении. Он принимает данные о географическом местоположении в качестве входных данных и возвращает информацию о месте, такую как идентификатор места. Метод также принимает параметры для ограничения поиска по определенному радиусу и фильтрации поиска по ключевым словам. Радиус поиска ограничен, а ключевое слово фильтра, используемое в вызове, - «продуктовый магазин». Эти параметры гарантируют, что GoogleAPIClient возвратит только продуктовые магазины. Однако метод «ближнего поиска» не возвращает адрес места, поэтому GoogleAPIClient необходимо сделать второй вызов, чтобы получить полный набор информации о нужном пользователю магазине. Он использует значение «идентификатор места», возвращенное предыдущим вызовом, чтобы присвоить ресурсу метку «место/детали» и чтобы получить адрес магазина. Расстояние между магазином и местоположением пользователя рассчитывается с помощью Google Distance Matrix API. Ресурс «distancematrix» принимает данные о географическом местоположении начальной и конечной точек в качестве параметров и возвращает рассчитанное расстояние для каждой конечной точки.

Ниже описаны модули интерфейсов, предоставляемые интерфейсами модуля BackendService:

Интерфейс IProductService предоставляет бизнес-логику для объекта Product. Он также реализует логику для создания записей сопоставления товаров и хранилищ.

Метод getProduct принимает UPC и географическое местоположение в качестве параметров, предоставляемых мобильным клиентом, и возвращает объект Product, который соответствует данному коду UPC. Метод вызывает метод IPProductDAO findProduct для извлечения записи продукта из базы

данных, если товар не найден возвращается null. Этот метод также создает новую задачу сопоставления продуктов и хранилищ и передает ее в ExecuterService для асинхронной обработки задачи в ThreadPool. Задача включает в себя вызов LabelAPIClient и GoogleAPIClient для получения продукта и хранения информации.

Метод findProducts принимает список аллергенов в качестве параметра и возвращает список объектов Product, которые не содержат ни одного из аллергенов в списке параметров. Метод вызывает метод IPProductDAO getProducts для поиска этих продуктов.

Метод findProductByName принимает productName в качестве параметра и возвращает список объектов Product, имя, торговая марка или описание которых содержат данное имя. Метод вызывает IProductDAO метод findProductByName для поиска товаров.

Интерфейс IStoreService предоставляет бизнес-логику для объекта Store. Он также реализует логику для расчета расстояния магазина.

Метод findProductStores принимает объект Product в качестве параметра и возвращает список объектов Store, содержащих данный продукт. Он вызывает IStoreDAO метод findProductStores для поиска.

Метод findLocationDistance принимает текущие данные о географическом местоположении пользователя и список магазинов в качестве параметров для расчета расстояния от местоположения пользователя до каждого из данного магазина. Метод возвращает карту, содержащую магазины и их рассчитанные расстояния, отсортированные по возрастанию. Он вызывает GoogleAPIClient, чтобы сделать расчет.

Интерфейс IUserService предоставляет бизнес-логику для объекта User. Он реализует логику для создания, выборки, обновления и проверки пользователей.

Метод createUser принимает объект User в качестве параметра и создает новый документ User в базе данных, используя метод createUser IUserDAO.

Пароли хешируются с помощью алгоритма MD5 перед сохранением в базе данных.

Метод validatePassword принимает простой текстовый пароль и объект User в качестве параметров для проверки пользователя. Метод вызывает метод getUser IUserDAO для выборки пользователя, который соответствует объекту User, и возвращает true, если данный пароль соответствует паролю пользователя.

Метод findUser принимает имя пользователя в качестве параметра и возвращает объект User, соответствующий имени пользователя. Он вызывает метод getUser IUserDAO для получения пользовательской записи.

Метод updateUser принимает объект User в качестве параметра, который содержит обновленную информацию о пользователе, и обновляет ее в базе данных. Он вызывает метод IUserDAO updateUser, чтобы выполнить обновление, и возвращает true, если обновление выполнено успешно.

Модуль DataAccessService – это службы доступа к данным доступные интерфейсам бизнес логики приложения. Модуль DataAccessService реализует отдельные интерфейсы для объектов Product, Store и User. Диаграмма классов UML на рисунке 2.6 дает представление о реализации модуля доступа к данным.

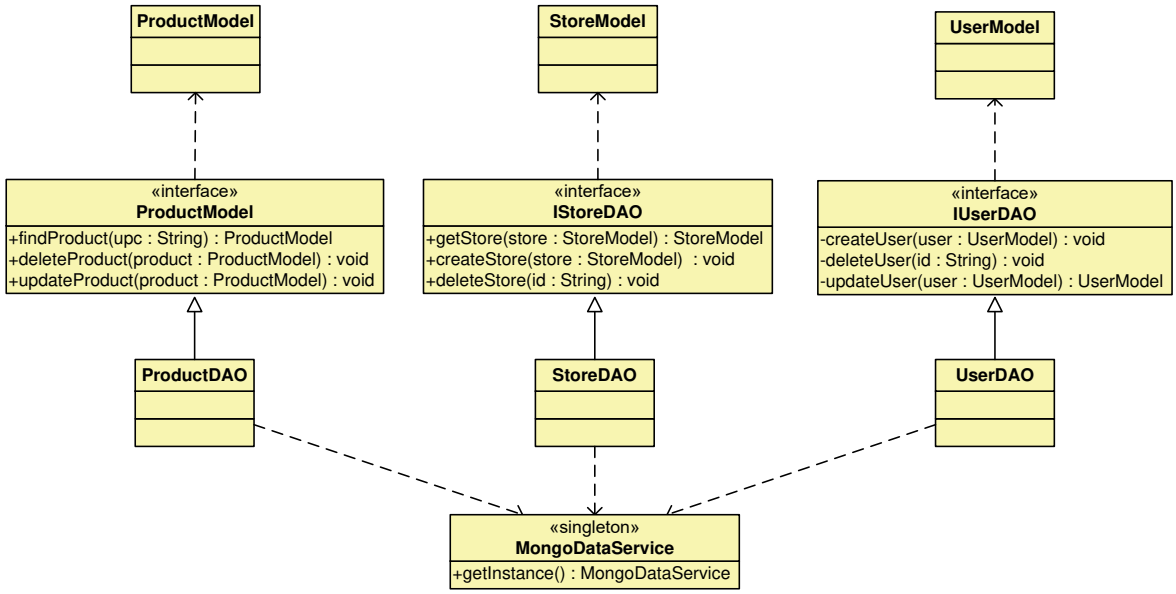


Рисунок 2.6 Модуль DataAccessService серверной части приложения.

Класс Singleo (Singleton Pattern) MongoDataService.java реализует фасад для Morphia (ORM для MongoDB). Реализация класса MongoDataService с помощью паттерна Singleton гарантирует, что в системе будет только один экземпляр связи между бизнес логикой приложения и СУБД (объект MongoClient). Плагин Morphia настроен для подключения к базе данныхMongoDB. Несколько одновременных подключений от одного пользователя приводят к излишней нагрузке на систему, именно для этого и была реализована защита от нескольких подключений. Все классы модели данных помещаются в пакет, классы которого сопоставляются с MongoDB. В данный момент в пакете реализованы следующие модели данных: ProductModel, StoreModel и UserModel.

Ниже представлен пример реализации модели ProductModel. Все остальные модели реализованы аналогичным образом.

```
@Entity("ProductModel")

class Product {

        @Id ObjectId id; // генерируется автоматически, если не задан вручную

        String iD, ProductName;

       Long salary = null; // Хранятся только не-null Значения

       Address address; //

       Key<Product> product;

       @Reference List<Product> underlings = new ArrayList<Product>(); //

       @Serialized ProductReviews; //

       @Property("started") Date startDate; //

       @Property("left") Date endDate;

       @Indexed boolean active = false; //

       @NotSaved String readButNotStored; //

        @Transient int notStored; //

       transient boolean stored = true;
```

```
        @PostLoad void postLoad(DBObject dbObj) { ... }
}
```

Объект ProductModel сопоставляется с коллекцией продуктов. ProductModel имеет свойства, показанные ниже.

Каждый продукт уникально идентифицируется по коду UPC. Уникальное ограничение, определенное в модели, обеспечивает его уникальность в базе данных. В свойстве noAllergen перечислены все аллергены, из которых продукт не должен состоять. Свойство «store» содержит ссылки на магазины, которые содержат продукт.

```
@db.Product.find({"upc" : “661799264327”}).pretty()
{
        «_id» ObjectIdId (“584ce3b89797fd822cc90”);
        «ClassName» (“com.app.mongo.data.model.ProductModel”);
        «upc» (“4600935000432”);
        «name» (“гречневая крупа Prosto”);
        «brandName» (“ООО Ангстрем Трейдинг”);
        «description» (“гречневая крупа”);
        «noAllergen»: [
                «egg»,
                «fish»,
                «gluten»,
                «coconut»,
        ],
        «stores»: [
                ObjectID («8432VCW3V+PG»),
```

```
        ObjectID («8431VCW3V+PG»),
        ObjectID («8444VCW3V+PG»),
        ObjectID («8433VCW3V+PG»),,
    ]
    «CreationDate» : ISODate (“2020-04-29”);
    «modificationDate» : ISODate (“2020-05-05”);
    }
```

Объект StoreModel связан с коллекцией СУБД Store и имеет свойства, как показано ниже. Магазины однозначно идентифицируются не только по имени, но и по адресу. Два магазина с одинаковым названием, но расположенные по разным адресам, считаются уникальными. Наряду с именем и адресом, коллекция Store также имеет свойства geoLocation и googlePlaceId. Свойство geoLocation определяет данные о географическом местоположении магазина. Свойство геолокации предоставляется мобильным клиентом, когда пользователь сканирует продукт, и используется для отображения магазина на карте Google. Свойство ‘GooglePlaceId’ предоставляется API Служб Google и однозначно определяет адрес.

```
@db.Store.findOne().pretty()
{
    «_id» ObjectIdId (“584fff656fdd822cc90”);
    «ClassName» (“com.app.mongo.data.model.StoreModel”);
    «streetNumber» (“567”);
    «streetName» (“Ленинградская”);
    «city» (“Пенза”);
    «province» (“Пенза”);
    «provinceAbbr » (“PNZ”);
    «storeName » (“Два гуся”);
```

```
        «country » (“RUS”);
        «zipcode » (“440000”);
        «geoLocation » (“53.188170, 44.980904”);
        «googlePlaceId» (“ChIJcdRHI05x44kR2MqPTo_ikkE”);
        «CreationDate» : ISODate (“2020-04-29”);
}
```

Объект UserModel сопоставляется с коллекцией User. Коллекция User состоит из полей показанных ниже. Каждый пользователь идентифицируется свойством username, которое соответствует электронной почте пользователя. Свойство «allergy» содержит список всех аллергенов, которые пользователь хочет избежать.

```
@db.User.findOne()
{
        «_id» ObjectIdId (“58465765765756dd822cc90”);
        «ClassName» (“com.app.mongo.data.model.UserModel”);
        «streetName» (“volkov.m@gmail.com”);
        «passord» (“e233244hh654654fdsfb654656456456”);
        «firstName» (“Max”);
        «lastName» (“Volkov”);
        «allergy»: [
              «egg»,
              «milk»,
              «soy»
        ]
}
```

В следующих разделах описаны методы интерфейса, реализованные модулем DataAccessService. Интерфейс IProductDAO обеспечивает операции CRUD для коллекции Product.

- findProduct: этот метод принимает код UPC в качестве параметра и возвращает объект ProductModel, который соответствует указанному коду UPC.

- findProductByName: этот метод принимает имя продукта в качестве параметра и возвращает список объектов ProductModel, имена которых соответствуют данному имени.

- getProducts: этот метод принимает список аллергенов в качестве параметра и возвращает список объектов ProductModel, которые не содержат аллергенов, перечисленных в параметре.

- createProduct: этот метод принимает объект ProductModel и объект StoreModel в качестве параметров и создает новый документ продукта, который сопоставляется с данной коллекцией магазина.

- addStoreToProduct: этот метод принимает объект ProductModel и объект StoreModel в качестве параметров и создает сопоставление между ними.

- deleteProduct: этот метод принимает идентификатор продукта в качестве параметра и удаляет документ продукта из базы данных, который соответствует указанному идентификатору.

- updateProduct: этот метод принимает объект ProductModel, который содержит обновленную информацию, в качестве параметра и обновляет существующий документ продукта новой информацией.

Интерфейс IStoreDAO обеспечивает операции CRUD для Коллекции Магазинов.

getStore: этот метод принимает объект StoreModel, который содержит адрес магазина в качестве параметра, и возвращает объект StoreModel, который соответствует адресу.

- createStore: этот метод принимает объект StoreModel в качестве параметра и создает

новый магазинный документ в коллекции магазина.

- findProductStores: этот метод принимает объект ProductModel в качестве параметра и возвращает список объектов StoreModel, которые содержат данные продукты.

Интерфейс IUserDAO обеспечивает операции CRUD для коллекции User.

- createUser: метод принимает объект UserModel в качестве параметра и создает новыю запись коллекции User.

- getUser: этот метод принимает имя пользователя в качестве параметра и возвращает объект UserModel, который соответствует данному имени пользователя.

- updateUser: этот метод принимает объект UserModel в качестве параметра и обновляет существующий документ пользователя обновленными данными, предоставленными в параметре UserModel.

2.2 Реализация интерфейса мобильного клиента

Мобильный клиент реализует интерфейс пользователя с использованием директив Ionic (IonicDocumentation) и компонентов AngularJS. Основной макет пользовательского интерфейса находится в index.html и определяется директивой <ion-nav-view>, которая работает с AngularUI Router (AngularUI Router) для визуализации страниц в зависимости от состояния приложения.

```
var app = angular.module('myApp', ['ionic']);
app.config(function($stateProvider) {
  $stateProvider
  .state('index', {
    url: '/',
    templateUrl: 'login.html'
```

```
  })
  .state('login', {
    url: '/login',
    templateUrl: 'login.html'
  });
});
```

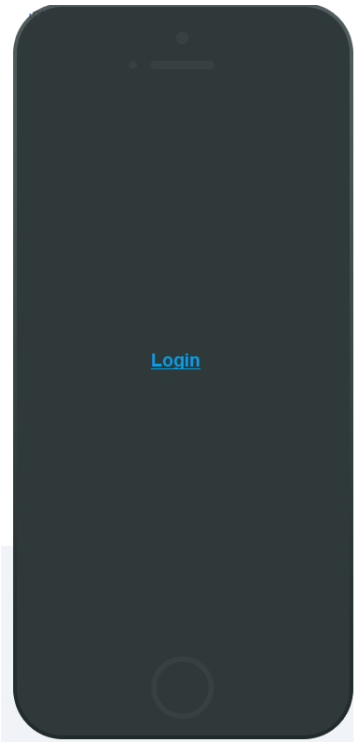


Рисунок 2.7 Первый результат работы с шаблонами мобильного клиента

Отдельные страницы определяются как шаблоны, которые вставляются в раздел <ion-nav-view> файла index.html, когда приложение меняет свое состояние. Например, когда пользователь обращается к приложению в первый раз, приложение переходит в состояние по умолчанию, которое определяется как состояние входа в систему, в результате чего маршрутизатор AngularUI внедряет шаблон страницы входа в <ion-nav-view > раздел, чтобы показать страницу входа в пользовательском интерфейсе.

Состояния приложения определяются в файле JavaScript AngularUI Router. Каждое состояние приложения определяется именем состояния (URL-адресом), именем шаблона и именем контроллера. Название State однозначно

идентифицирует состояние приложения. URL-адрес определяет путь к бизнес логике состояния.

Имя шаблона идентифицирует шаблон HTML, который нужно вставить, когда приложение переходит в это состояние. Имя контроллера идентифицирует контроллер AngularJS для инициализации, когда происходит переход для наложения данных бизнес логики на шаблон отображения. Ниже показаны некоторые состояния, определенные в приложении.

```
$stateProvider

  .state('allergies', {
  url: '/allergies',
  templateUrl: 'templates/allergies.html',
  controller: 'allergiesController'
})

.state('store', {
  url: '/store',
  templateUrl: 'templates/store.html',
  controller: 'storeController'
})

.state('result', {
  url: '/result/',
  templateUrl: 'templates/result.html',
  controller: 'resultController'
})
```

Контроллеры AngularJS в данном приложении - это компоненты, которые управляют данными и предоставляют их в обработанном виде для интерфейса приложения. Каждая страница шаблона связана со своим собственным контроллером. Эти контроллеры используют сервисы AngularJS для обеспечения функциональности страницы, которой они управляют. $ Scope, $ rootScope и $ http являются одними из основных сервисов AngularJS, используемых реализованными контроллерами.

$Scope сервис, который связывает представление с контроллером. Каждое представление имеет свой собственный экземпляр службы $ scope.

С другой стороны, $rootScope - это глобальный объект с одним экземпляром, который доступен всему приложению.

$RootScope используется в приложении для установки свойства системного уровня, такого как URL-адрес внутренней системы. Служба $http обеспечивает платформу для выполнения вызовов веб-службы к серверной системе. Эти сервисы обрабатывают объекты контроллеров (бизнес логики), когда они создаются.

В приложении реализован сервис «dataService», который позволяет обеспечить доступ к СУБД.

Далее опишем реализацию страниц интерфейса приложения.

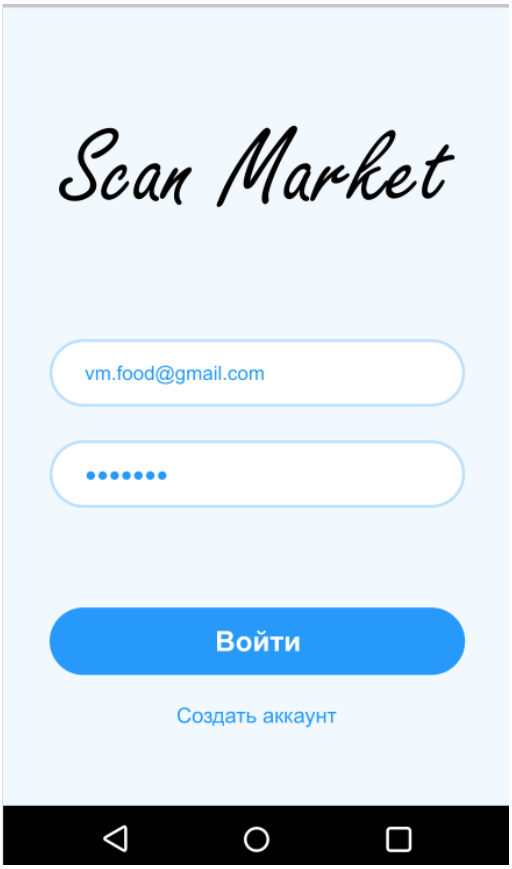


Рисунок 2.8 - Вход в систему

Страница регистрации предоставляет функцию регистрации пользователей, как указано на рисунке 2.9. Пользователь переходит на страницу регистрации, щелкнув ссылку «создать аккаунт» на странице входа в систему. Страница регистрации имеет две панели. Верхняя панель предоставляет пользователям текстовые поля для ввода их имени, фамилии,

имени пользователя и пароля. Все текстовые поля являются обязательными полями. Имя пользователя должно быть действительным адресом электронной почты. На нижней панели расположены ползунки переключателей для 6 поддерживаемых аллергенов, из которых пользователи могут выбирать аллергены, которых они хотят избежать. Кнопка «Войти» отправляет данные. Проверки формы обеспечивают правильность заполнения всех значений, а соответствующие сообщения об ошибках отображаются в верхней части представления в случае неудачной проверки. Если пользователь успешно регистрируется, приложение переходит на домашнюю страницу.

Страница регистрации связана с контроллером «registerController». Кнопка «Войти» при отправке вызывает метод «register» в контроллере, который отправляет HTTP-запрос POST в интерфейс UserController на уровень бизнес логики объекту класса BackendSystem для публикации данных регистрации. Если регистрация прошла успешно, возвращается вновь зарегистрированный объект User, с заполненными переменными loggedInUser, связанной со службой интерфейса $rootScope, чтобы пользовательский объект был доступен для других частей приложения, а пользователь перенаправлялся на домашнюю страницу.

Если регистрация не удалась, пользователь остается на странице регистрации с сообщением об ошибке, отображаемым в верхней части страницы.

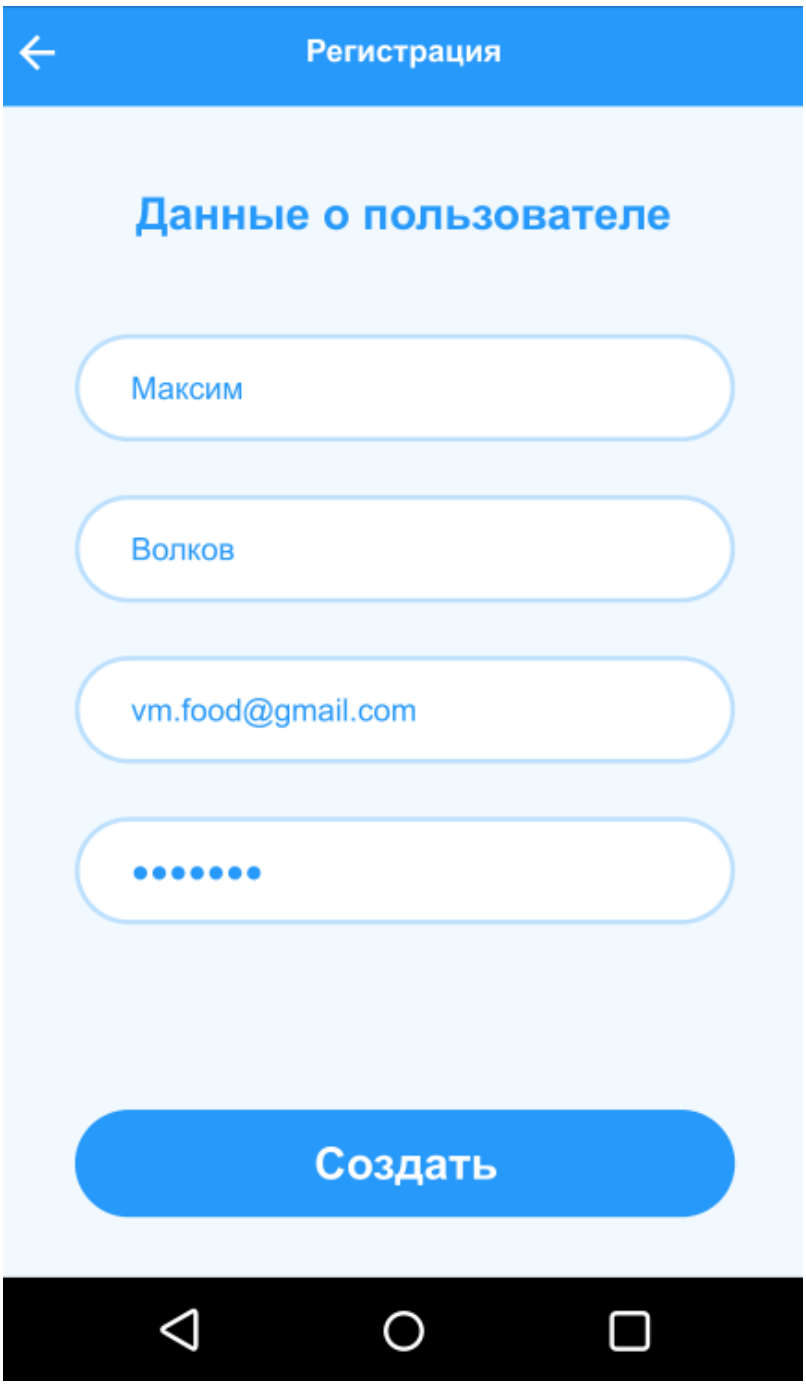


Рисунок 2.9 Интерфейс регистрации мобильного приложения

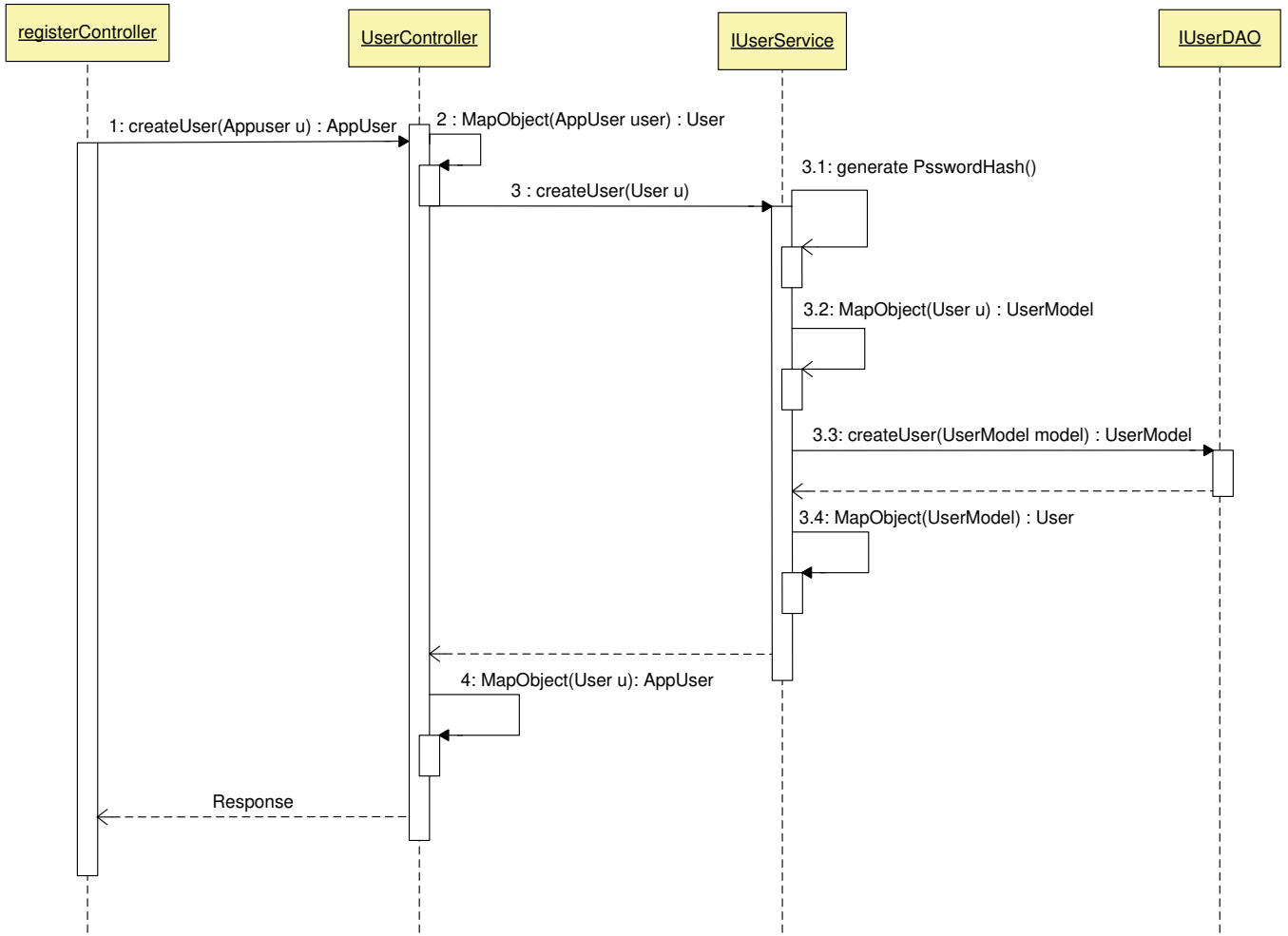


Рисунок 2.10 Процесс регистрации с точки зрения вызова функций

Домашняя страница имеет три кнопки на главной панели и одну кнопку в нижнем колонтитуле. Три кнопки на главной панели: «Сканирование продуктов», «Поиск продуктов» и «Поиск магазинов». Кнопка «Настройки» находится в нижнем колонтитуле. Кнопка «Сканирование продуктов» вызывает функцию проверки продукта. Кнопка «Поиск продуктов» позволяет пользователям перейти на страницу «Поиск», где они могут искать магазины, в которых продаются товары, которые им интересны. Кнопка «Поиск магазинов» позволяет пользователям перейти на страницу «Магазины», где они могут искать и фильтровать магазины, в которых хранятся товары, безопасные для них.

Кнопка «Настройки» позволяет пользователям перейти на страницу настроек, где они могут редактировать данные своего профиля.

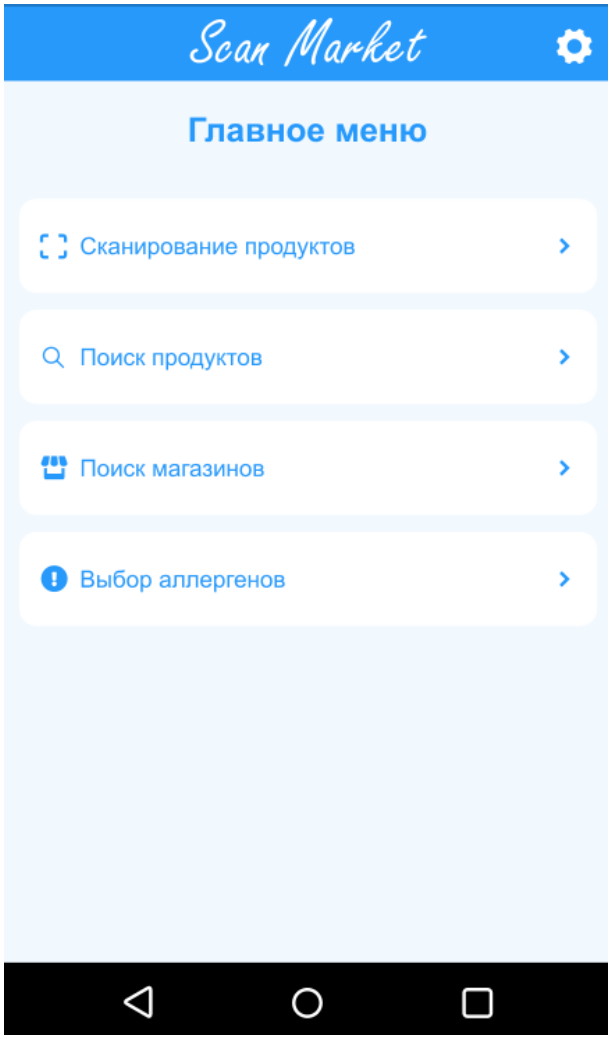


Рисунок 2.11 Главная страница приложения.

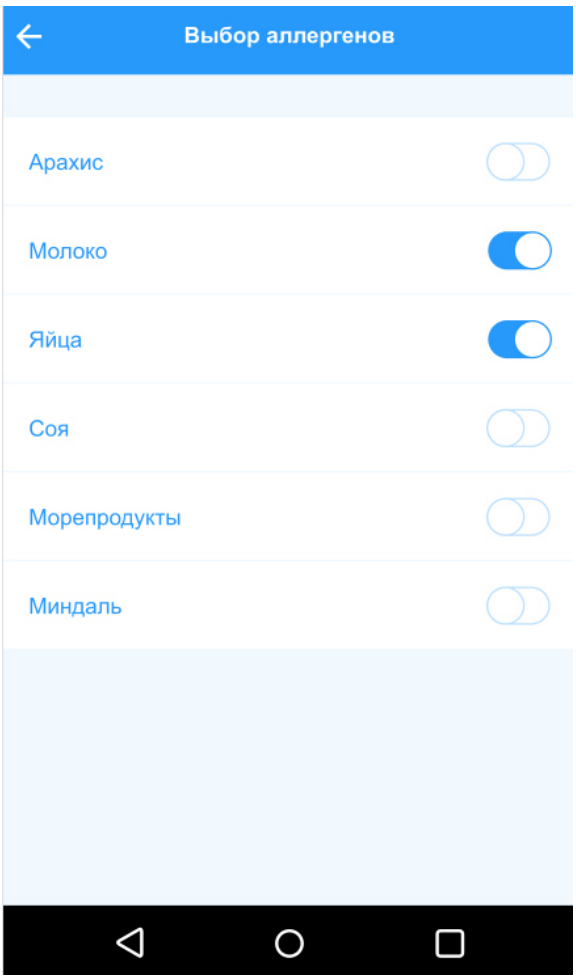


Рисунок 2.12 Выбор аллергенов

Домашняя страница привязана к «homeController». При нажатии на кнопку «Сканирование продуктов» запускается плагин BarcodeScanner, который запускает камеру телефона, позволяя пользователям делать снимки штрих-кода продукта. BarcodeScanner предоставляет UPC-код отсканированного продукта приложению. Контроллер также извлекает данные о географическом местоположении текущего местоположения пользователя с помощью плагина GeoLocation и вызывает интерфейс ProductController в системе BackendSystem для проверки продукта на аллергены. Система BackendSystem создает продукт для хранения картографических записей, как описано в классе ProductStoreMapping. UML Диаграмма показывает последовательность событий, которые происходят, когда пользователь нажимает кнопку «Сканирование продуктов».

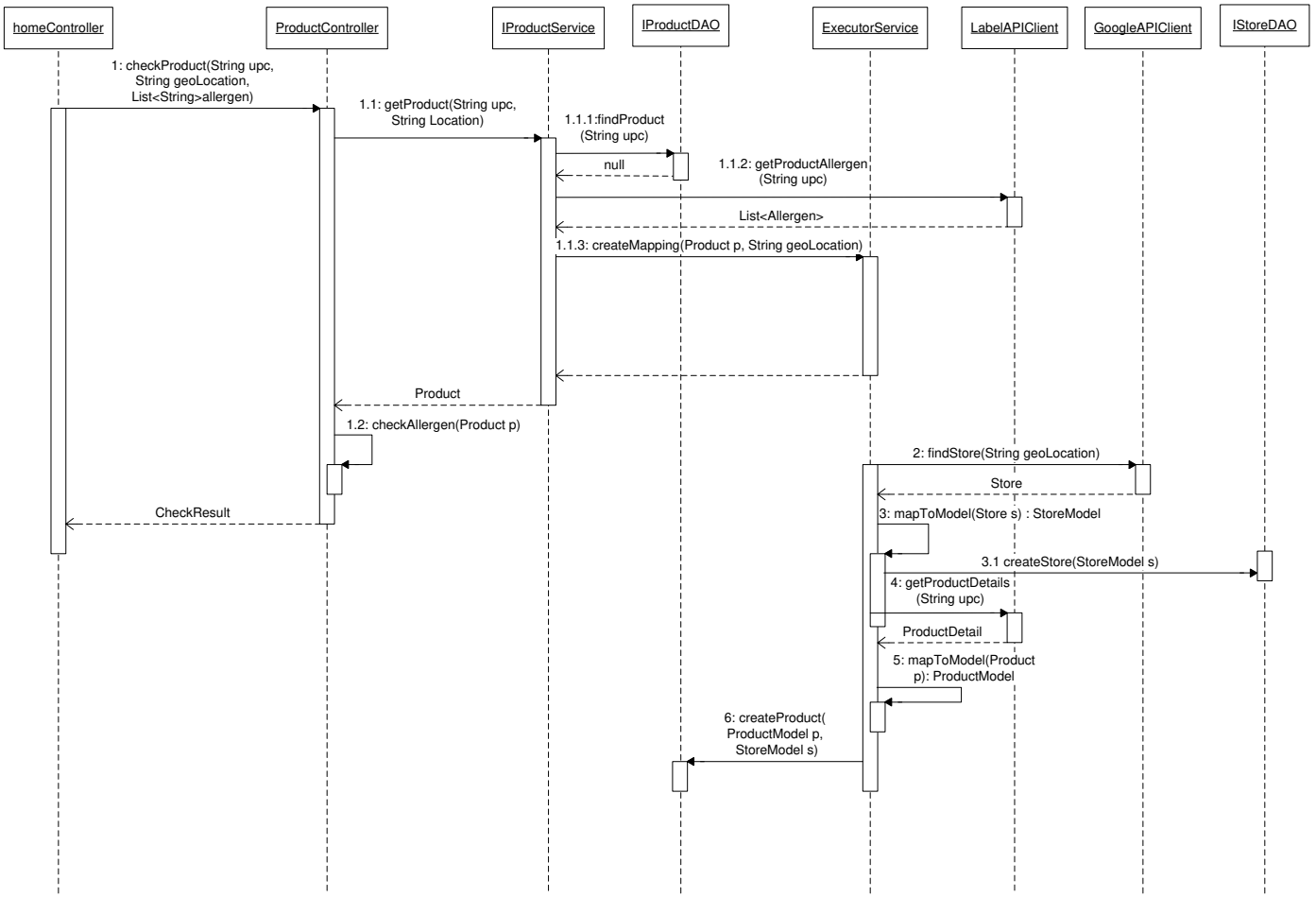


Рисунок 2.13 Процесс сканирования продукта

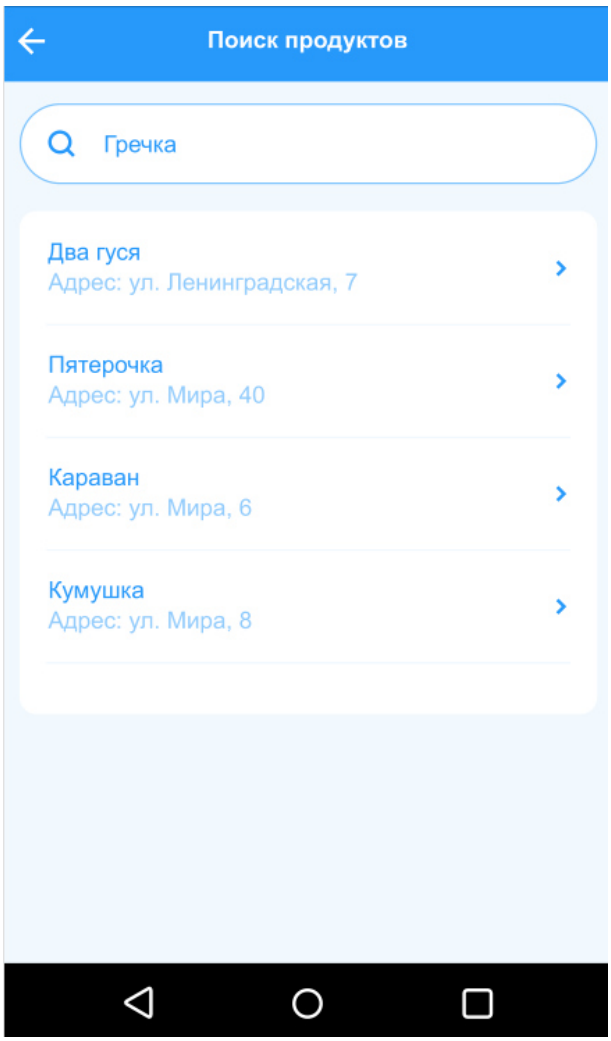


Рисунок 2.13 Процесс выбора магазина в текстовом виде

Страница поиска обеспечивает функциональность поиска продукта, на основании контроллеров и моделей указанных выше.

Доступ к странице осуществляется с помощью кнопки «Поиск магазина» на домашней странице и позволяет пользователям искать магазины, в которых представлены товары, в которых они заинтересованы. На странице есть одно текстовое поле, в котором находится строка поиска. Строка поиска должна содержать наименование товара. Поиск начинается, когда пользователь нажимает кнопку поиска или клавишу ввода. Поиск не чувствителен к регистру. Результат поиска отображается на нижней панели.

В результате отображается список местоположений магазина в табличном формате, упорядоченный по текущему местоположению

пользователя в алфавитном порядке. Расстояние от текущего местоположения пользователя до каждого местоположения сохраняется в СУБД.

Адрес магазина вместе с информацией о товаре отображается списком. Кнопка расстояния нажимается и переносит пользователя на страницу, которая показывает местоположение магазина на карте Google.

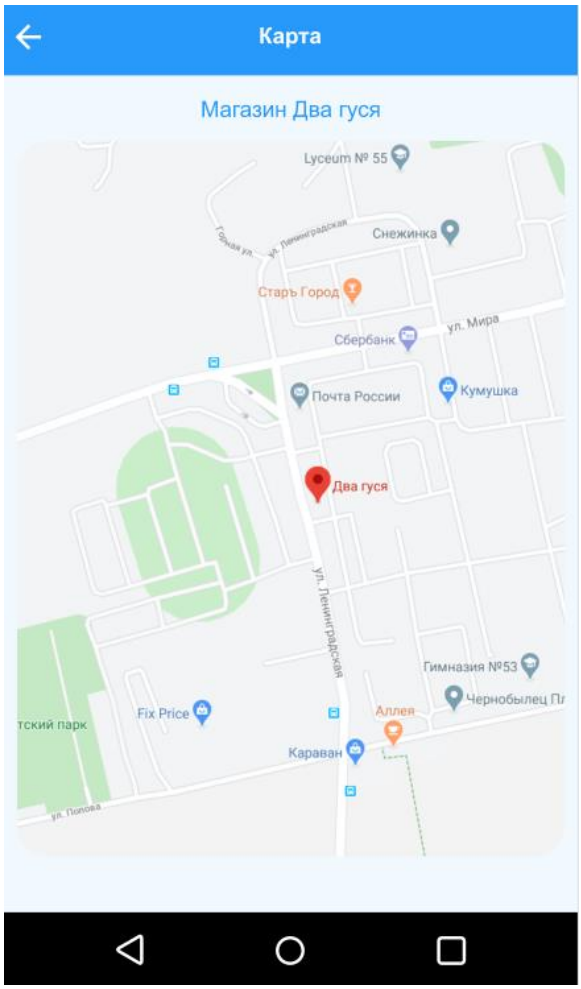


Рисунок 2.14 Процесс выбора магазина в графическом виде

Страница привязана к «locateController». При нажатии на кнопку поиска или клавишу ввода контроллер получает текущее местоположение пользователя с помощью плагина GeoLocation и отправляет HTTP-запрос GET к интерфейсу ProductController в модуле Backend System, чтобы начать поиск. Местоположение на карте Google отображается с помощью Google Map API JavaScript.

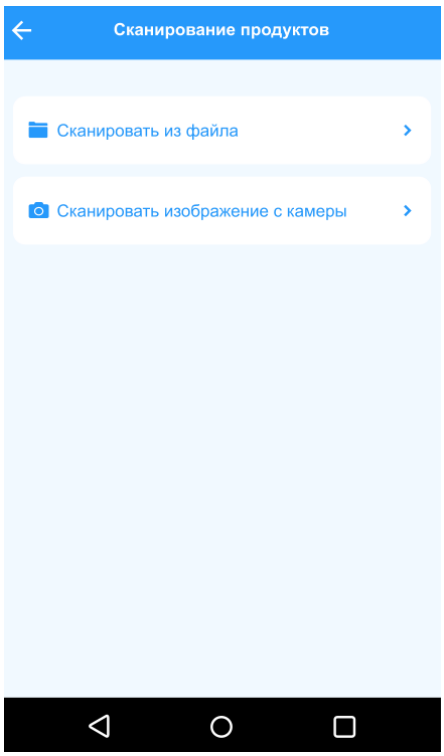


Рисунок 2.15 Процесс выбора типа сканирования

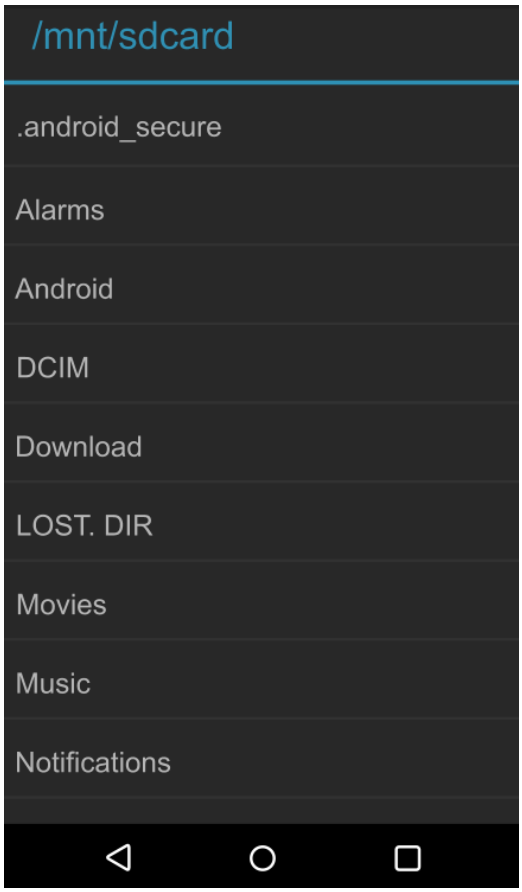


Рисунок 2.16 Выбор файла

Рисунок 2.17 Выбранный файла UPC

Доступ к странице Поиск товара можно получить с помощью соответствующей кнопки на домашней странице. На странице отображаются все магазины в базе данных, в которых хранятся продукты, безопасные для пользователя. На этой странице реализовано требование поиска и фильтрации, указанное в СУБД. На странице показаны местоположения магазинов, сгруппированные по алфавиту, как показано на рисунке 2.20, а также количество уникальных продуктов. Это дает пользователям сводное представление о магазине и о том, сколько уникальных продуктов содержит магазин.

Кнопка со стрелкой справа от строки результатов переводит пользователей на подстраницу, которая показывает расположение магазинов выбранной торговой марки. На этой подстранице показаны адреса магазинов в табличном формате, а также общее количество уникальных продуктов,

найденных в этом месте, в виде округлой кнопки, как показано на рисунке 2.20.

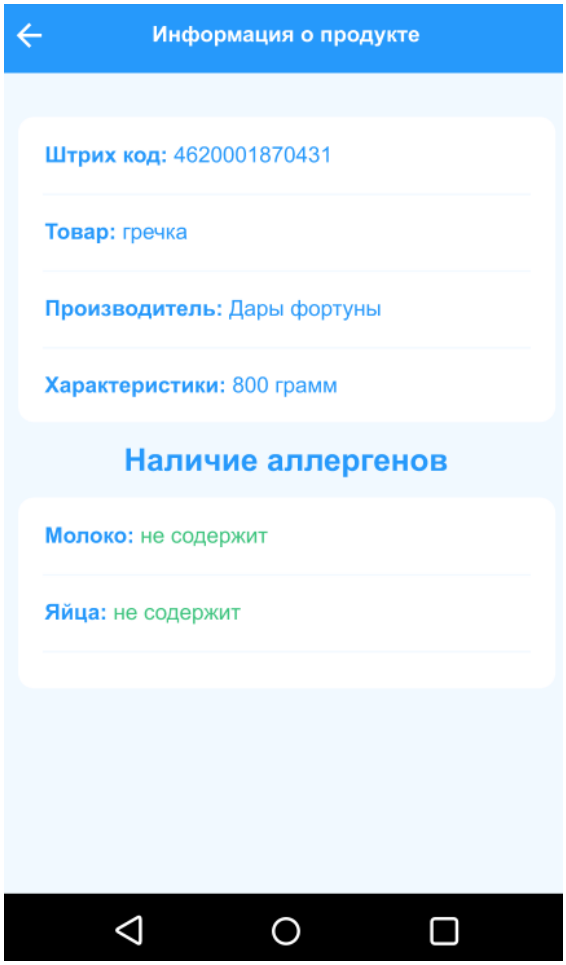


Рисунок 2.18 Результат сканирования файла

Кнопка номера продукта на подстранице «Местоположение магазина» переносит пользователей на другую подстраницу, на которой отображается список всех уникальных продуктов, найденных в этом конкретном месте. Список продуктов отображается в табличном формате и содержит название и описание продукта.

Страница Поиск продуктов использует контроллер «dataService» для предоставления данных о сопоставлении продуктов. Каждая подстраница данного блока связана с различными контроллерами, которые управляют данными для представления в нужном пользователю виде, но все они зависят от одной и той же службы данных. ProductController в BackendSystem предоставляет сервис «findProducts», который возвращает все необходимые

данные отображения в одном вызове метода в виде сборного набора данных, поэтому «dataService» выполняет только один HTTP-вызов GET для BackendSystem, чтобы показать все представления.

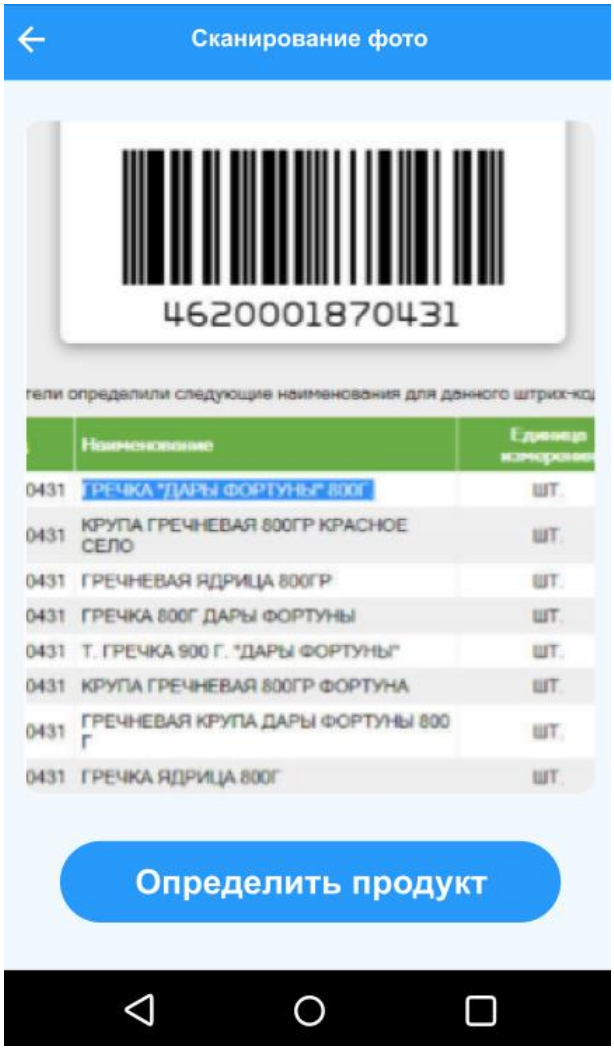


Рисунок 2.19 Привязка продуктов к UPC

Страница входа обеспечивает безопасность для пользователей, позволяя работать в системе только авторизированным пользователям. Страница содержит текстовые поля «Имя пользователя» и «Пароль» и ссылку для регистрации. Кнопка «Вход» запускает процесс аутентификации. Успешная аутентификация перенаправляет пользователей на домашнюю страницу, запускает приложение в индивидуальном сеансе пользователя. Если аутентификация не пройдена, пользователи остаются на странице входа, а в верхней части страницы отображается сообщение об ошибке.

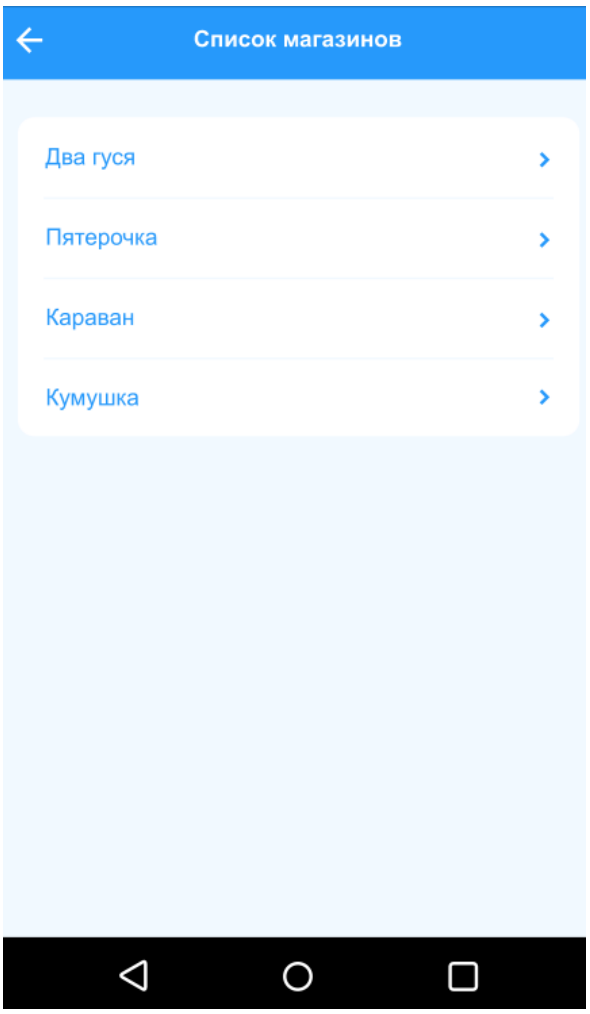


Рисунок 2.20 Поиск магазинов с выбранным товаром

Страница «Вход» привязана к контроллеру «loginController». Нажатие на кнопку «Login» вызывает функцию «login» в контроллере, которая вызывает интерфейс UserController в BackendSystem для проверки пользователя.

Страница настроек позволяет пользователям редактировать информацию своего профиля. Доступ к странице осуществляется с помощью кнопки «Настройки» в нижней части домашней страницы. Страница позволяет пользователям редактировать свое имя, фамилию и выбор аллергена. Панель вверху содержит предварительно заполненные редактируемые текстовые поля «Имя» и «Фамилия», а панель внизу содержит переключатели для изменения выбора аллергена. Кнопка «Обновить» отправляет изменения на сервер. Если обновление выполнено успешно, аутентифицированный пользователь с

помощью переменной $rootScope обновляет информацию о себе и возвращается на домашнюю страницу.

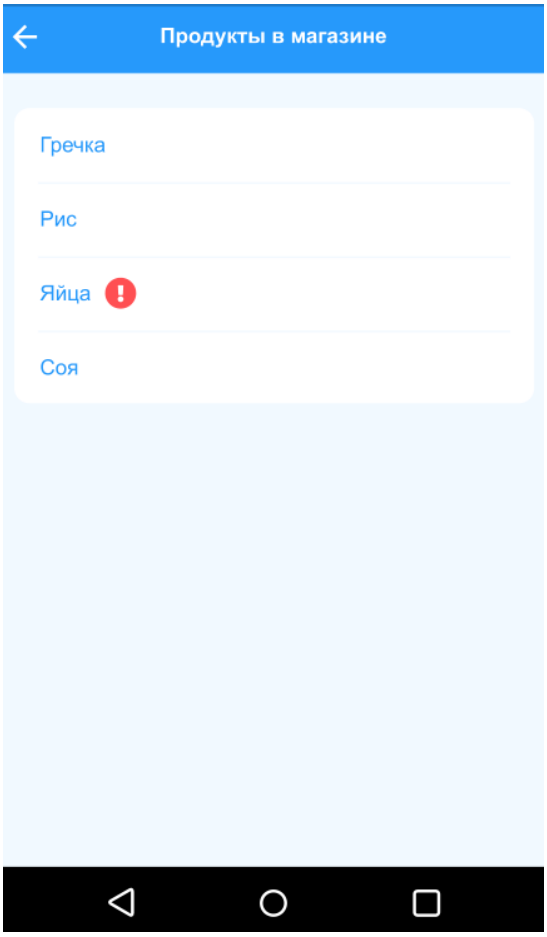


Рисунок 2.21 Выбор продукта с аллергеном

Если обновление не удалось, соответствующее сообщение об ошибке отображается в верхней части страницы. На странице также есть кнопка «Назад» в верхнем левом углу, которая позволяет пользователям выходить из данного режима системы.

Когда пользователь выходит из системы, для объекта пользователя в $ rootScope устанавливается значение NULL, и аутентифицируемый сеанс приложения заканчивается.

3. Технология тестирования приложений на базе платформы amazon

На сегодняшний момент Amazon самая мощная ферма для тестирования на виртуальных и реальных устройствах на сегодняшний день (более 2500 устройств). Для тестировщика это может быть приоритетным сервисом, так как сервисы backend приложения развёрнуты в VPS среде AWS, кроме того, для некоммерческого использования предоставляется бесплатный доступ. AWS позволяет работать как с фреймворками автоматизированного тестирования. Помимо автоматизированного тестирования, сервис предоставляет возможность ручной отладки.

При проведении тестов AWS (в бесплатном общедоступном облаке) использует выполнение на стороне сервера. Это значит, что тестовый код работает на инфраструктуре Amazon. В качестве альтернативы другие провайдеры, такие как Browserstack или Saucelabs, применяют выполнение на стороне клиента, когда тестовый код запускается на компьютере пользователя и взаимодействует с облачной службой через веб-сервис.

Выполнение на стороне клиента легче реализовать, процесс аналогичен выполнению тестов локально.

Подход AWS слегка сложнее, так как требует дополнительных шагов: упаковки и загрузки всего необходимого для запуска тестов на серверах. Это связано с дополнительными настройками.

Ниже представлен способ загрузки приложения для проведения тестов, через облачную консоль AWS.

3.1 Веб-консоль AWS Device Farm

Самый простой способ запустить тесты в AWS Device Farm сделать это через веб-консоль. Этот процесс эффективен если тесты запускаются редко. Для частого тестирования рационально использовать отладочные API AWS Device Farm.

Далее идет описание процесса работы с отладочной консолью AWS Device Farm. Войдите в AWS и откройте панель управления Device Farm (рисунок 3.1).

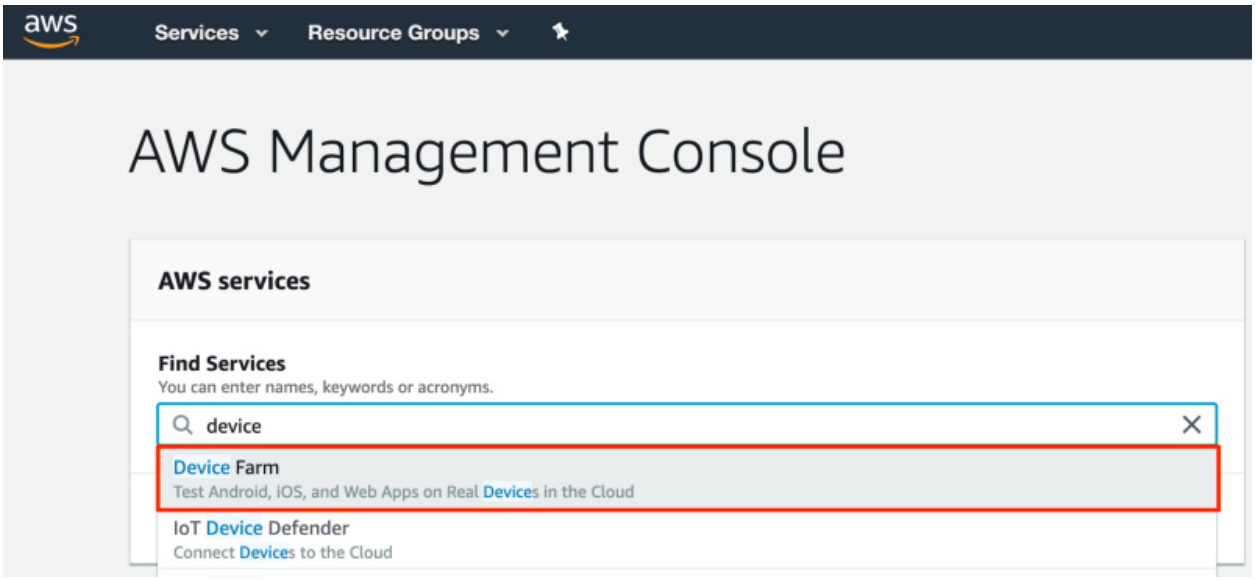


Рисунок 3.1 Консоль тестирования AWS Device Farm.

Далее создается новый проект для тестирования (рисунок 3.2).

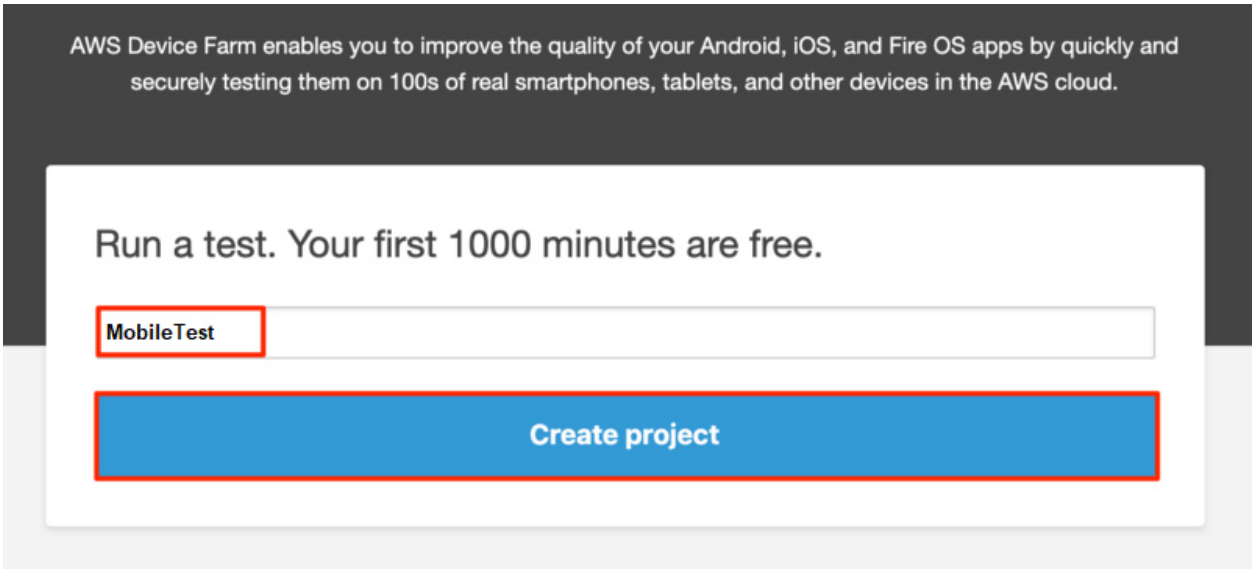


Рисунок 3.2 Наименование нового проекта

Нажмите кнопку «Create project» и после создания в облаке среды для тестирования нажмите кнопку «Create new run» для пробного запуска теста в облаке.

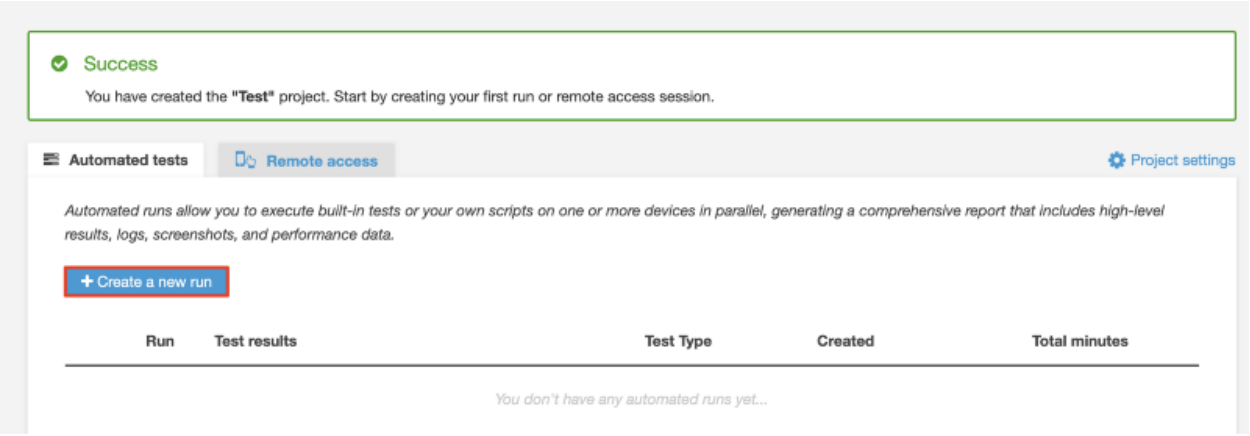


Рисунок 3.3 Подготовка проекта к запуску

Загружаем в облако прошивку приложения, которое необходимо протестировать.

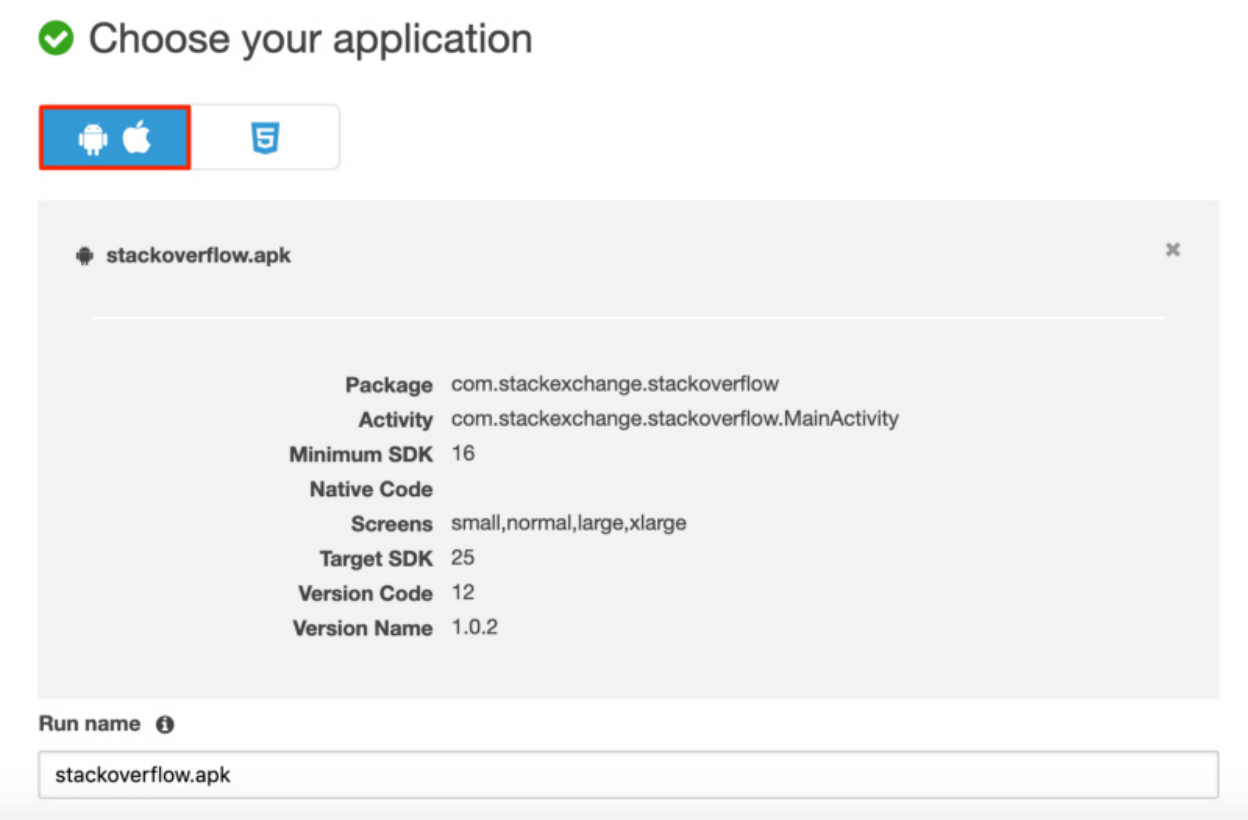


Рисунок 3.4 Подготовка проекта к запуску

Выбираем тип тестирования загруженного в облако приложения.

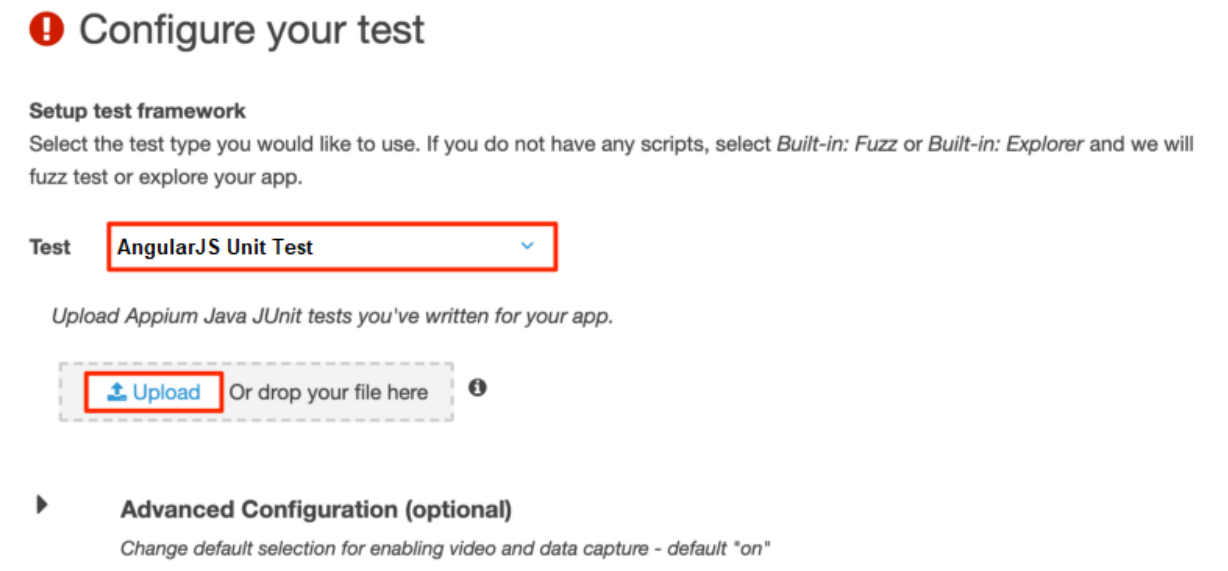


Рисунок 3.5 Выбираем тип тестирования

С использованием программных средств Ionic создаём тестовый пакет в виде zip-файла, следуя инструкциям в документации.

Спецификация теста – это YAML-файл, в котором содержится список команд оболочки, что запускаются на разных этапах выполнения теста. Это даёт некоторый контроль над способом выполнения тестов в AWS.

Спецификация теста представляет из себя единый файл в формате YAML, состоящий из трёх разделов:

заголовок, содержащий название, описание и версию API, а также дополнительную информацию;

описание всех ресурсов, включая их идентификаторы, методы, все входные параметры, а также коды и форматы тела ответов, со ссылками на определения;

все определения объектов в формате JSON Schema, которые могут использоваться как во входных параметрах, так и в ответах на тестовые задачи.

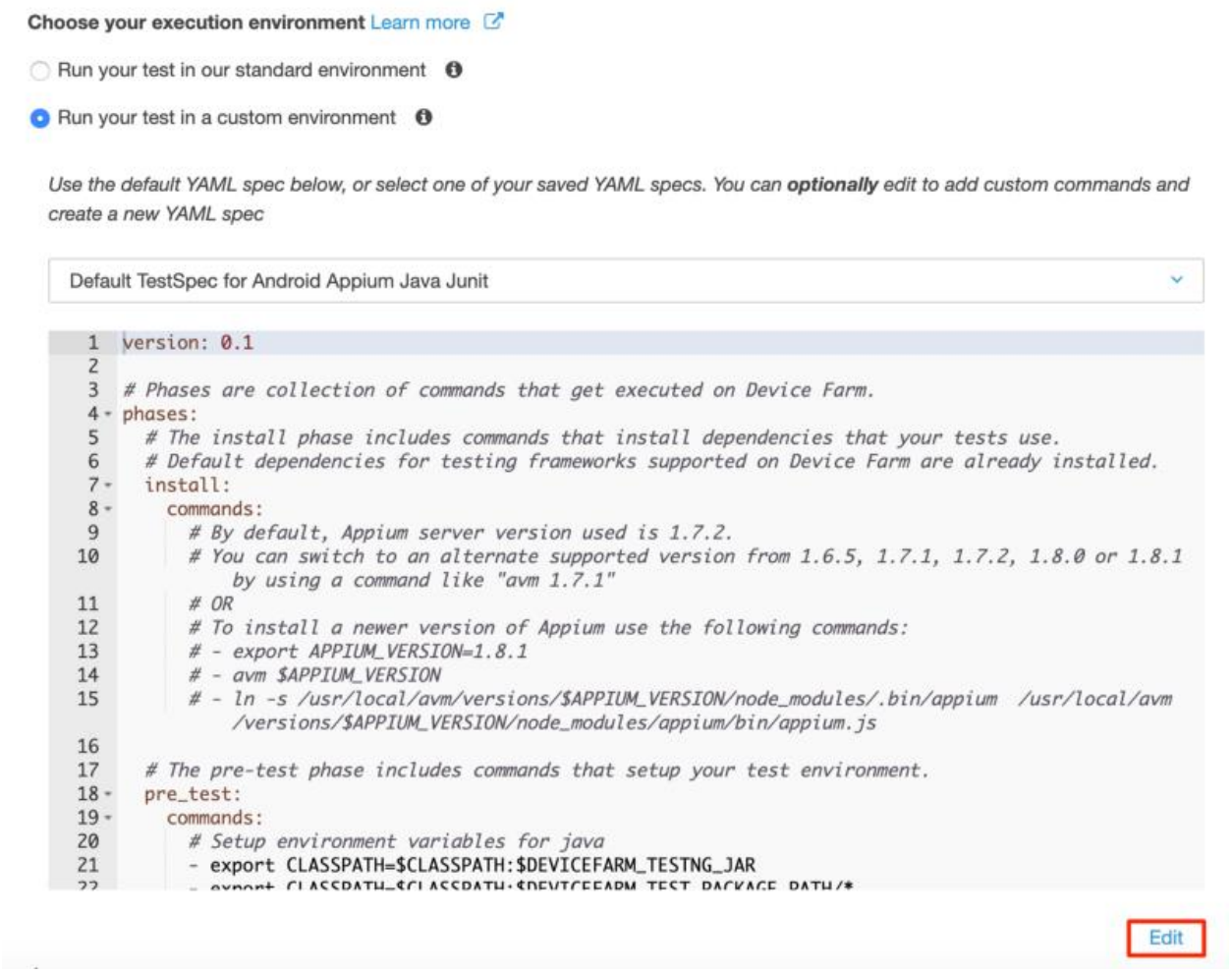


Рисунок 3.6 Файл теста

AWS использует концепцию под названием пулы устройств. Каждое выполнение теста будет запускать все тесты на всех устройствах в пуле. Для начала хватит одного устройства, поэтому создаём новый пул устройств с единственным Google Pixel.

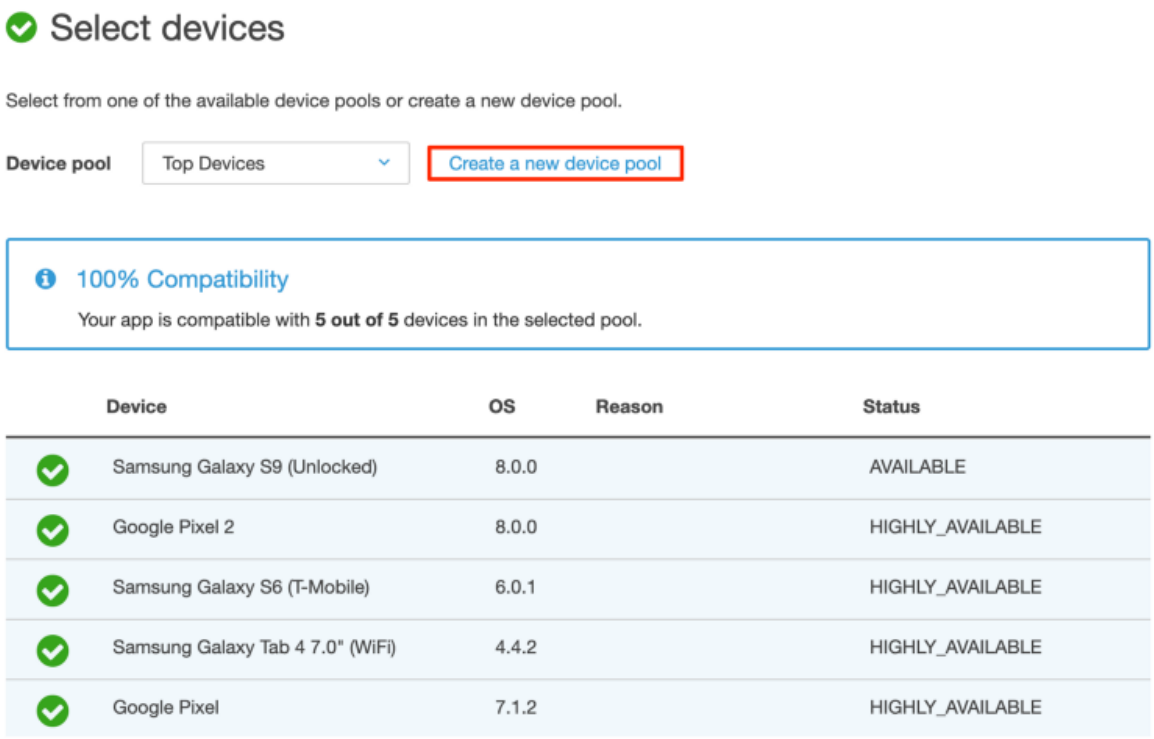


Рисунок 3.7 Выбираем устройство для тестирования

Далее следует шаг «Укажите состояние устройства» (Specify device state), который пока пропускаем (оставляем значения по умолчанию). На последней странице видим запуск тестов. Сократим таймаут, чтобы не тратить бесплатные минуты тестирования устройства.

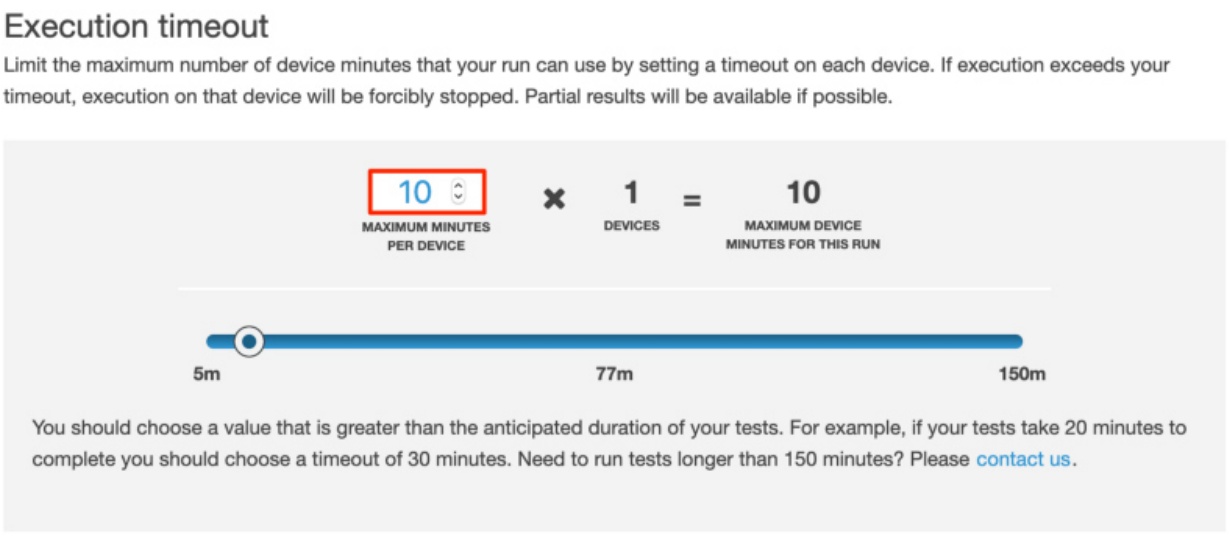


Рисунок 3.8 Установка таймера работы устройства

После мероприятий по настройке среды исполнения теста, переходим к запуску теста на исполнение.

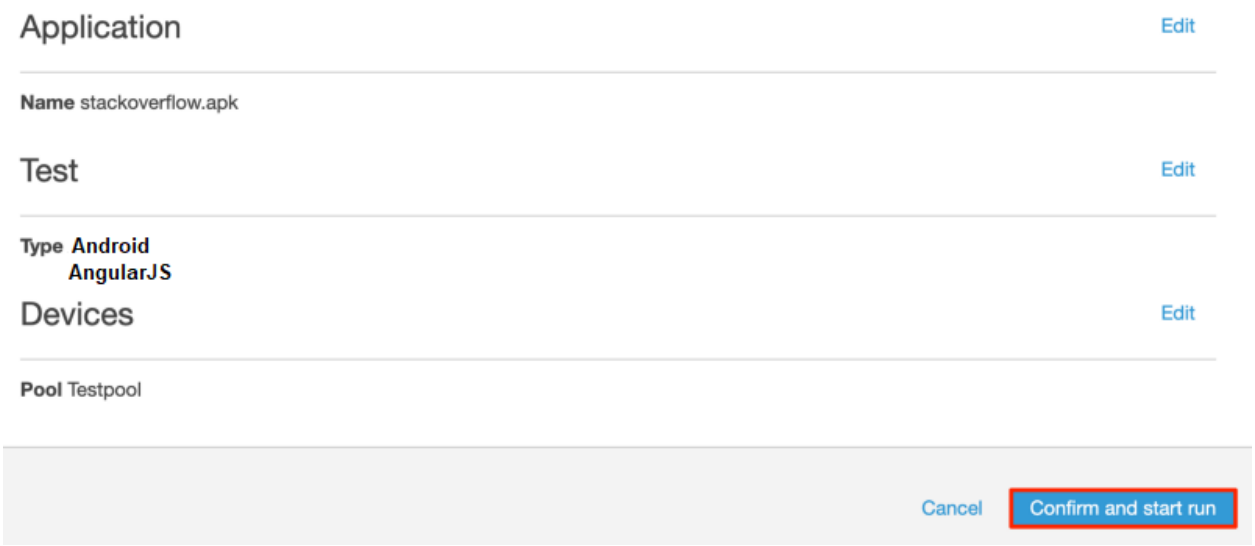


Рисунок 3.9 Запуск теста

Видим запуск на информационной панели. Для продолжения требуется нажать и выбирать тестовое устройство (Google Pixel), чтобы просмотреть тестовые отчеты в реальном времени и прямую трансляцию устройства (настройка займёт несколько минут).

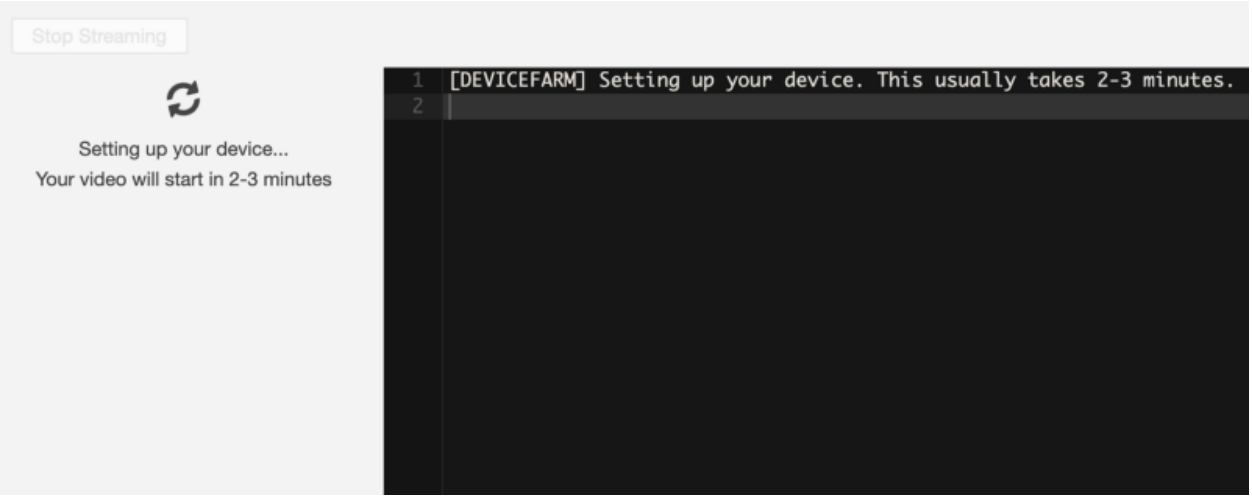


Рисунок 3.10 Система производит автоматическое тестирование.

Обратите внимание на продолжительность выполнения теста (рисунок 3.11). Видим немалые накладные расходы на запуск и завершение устройства (и на выполнение спецификации теста и инициализации WebDriver), которые стоит учитывать при планировании выполнения теста. Из-за этих издержек AWS, как правило, гораздо лучше подходит для запуска больших наборов тестов за один раз, чем меньших за пару прогонов. В некоторой степени это справедливо для всех облачных решений, но размер накладных расходов зависит от провайдера. И, следовательно, играет роль при выборе.

Рисунок 3.11 Результат тестирования

В результате проведенного тестирования был освоен способ реализации автотрестов для мобильного приложения и проведено пробное тестирование, которое показало, что приложение будет работать в штатном режиме.

4 Технико-экономическое обоснование разработки инвестиционного проекта

4.1 Описание программного продукта

В данной работе будут рассмотрены экономические аспекты разработки автоматизированной системы сбора информации об объектах капитального строительства в Пензенский филиал ФГУП «Ростехинвентаризация – Федеральное БТИ».

Главная цель проекта состоит в создании автоматизированной системы сбора информации, которая позволит сократить сроки поиска объектов капитального строения по владельцам и по реквизитам самого объекта, для формирования входящих запросов, поступающих от юридических лиц.

Автоматизированная система сбора информации выполняет следующие функции:

- ввод, редактирование и хранение заявок, поступающих от клиентов;
- расчет текущей инвентаризационной стоимости объекта;
- формирование файла в XML формате для предоставления данных в федеральную налоговую службу (ФНС);
- формирование отчета по жилым объектам.

В системе работают следующие пользователи:

а) техники по приему заявок осуществляют прием запросов, поступающих от юридических лиц, отправляют ответ на каждый входящий запрос.

б) техники-инвентаризаторы, отмечающие объекты, которые необходимо передать в ФНС;

в) работники отдела информационных технологий, обрабатывающие поступающие объекты для передачи в ФНС.

Созданием системы по заказу предприятия будут заниматься бизнес-аналитик и программист, выполняющего кроме своих основных обязанностей функции автора документации. Разработка будет вестись в помещениях и на оборудовании заказчика. Время, необходимое для разработки, равно 65 дней (3,1 месяца). Общая сумма затрат на разработку составляет 29094,68 руб.

4.2 Расчет затрат на разработку

Для создания автоматизированной системы сбора информации об объектах капитального строительства, необходимо привлечение следующих специалистов:

- бизнес-аналитик, знающий предметную область, формулирующий техническое задание на разработку;
- программист, непосредственно занимающийся разработкой клиентской и серверной частей системы и выполняющий, помимо этого, функции администратора и автора технической документации.

Дневная заработная плата специалистов приведена в таблице 10.1

Таблица 10.1 — Дневная заработная плата специалистов

Специалист	Дневная заработная плата
Бизнес-аналитик	350
Программист	250

Для работы этих специалистов были использованы 2 персональных компьютера стоимостью 25000 руб. каждый. Компьютеры принадлежат заказчику. Годовая норма амортизации компьютеров 20%. Расходы на обслуживание оборудования составляют 25% амортизационных отчислений. Энергопотребление каждого компьютера составляет 150 Вт в час. Стоимость одного кВт/ч электроэнергии составляет 3,44 руб. Компьютеры работают 8 часов в день.

Разработка ведется в помещениях и на оборудовании заказчика. Расходы на оплату машинного времени (амортизация), обслуживание оборудования, электроэнергию, зарплату разработчикам, накладные расходы несет заказчик.

Для определения состава работ и их последовательности был использован метод сетевого планирования, основанный на построении графического изображения определенного комплекса работ, отражающего их логическую последовательность, взаимосвязь и длительность. Для обеспечения построения графического изображения необходимо разбить процесс разработки на этапы (см. таблицу 10.2). Эти этапы в основном соответствуют обычному процессу разработки любого программного обеспечения: постановка задачи, анализ требований, проектирование, кодирование и т. п.

Этапы разработки программного обеспечения:

- анализ предметной области;
- постановка требований к системе;
- построена функциональная модель;
- построена инфологическая модель;
- разработана серверная часть системы;
- разработана программа для сбора информации об объектах капитального строительства;
- отладка программы;
- тестирование программы;
- оформление документации и обучение персонала;
- сдача в эксплуатацию.

На основе этапов разработки программного обеспечения составляется список работ, который отражает их взаимосвязь и логическую последовательность (см. таблицу 10.3).

Таблица 10.3 - Список работ этапов

Наименование события	Исполнители	Время выполнения дн.
Анализ предметной области	Бизнес-аналитик	4
Постановка требований к системе	Бизнес-аналитик	3
Изучение требований к системе	Программист	2

Продолжение таблицы 10.3

Разработка программы сбора информации об объектах капитального строительства	Программист	24
Отладка программы	Программист	5
Тестирование программы	Бизнес-аналитик Программист	4
Оформление документации и обучение персонала	Программист	5
Сдача в эксплуатацию	Бизнес-аналитик Программист	2
Разработка функциональной модели	Бизнес-аналитик	3
Разработка инфологической модели	Бизнес-аналитик	3
Разработка серверной части системы	Программист	10

Длина критического пути (время, необходимое для разработки) равна 65 рабочих дней. График распределения ресурсов приведен на рисунке 10.1. Расчет заработной платы участников разработки приведен в таблице 10.4.

Таблица 10.4 - Заработная плата участников разработки

Специалист	Время работы, раб. дн.	Сумма оплаты, руб.	Отчисления в соц. фонды, руб.	Зарплата и отчисления, руб.
Бизнес-аналитик	20	7000	1820	8820
Программист	58	14500	3770	18270
Всего		21500	5590	27090

Таким образом, расходы на оплату труда составляют 27090 руб.

Накладные расходы составляют 25% от затрат на оплату труда, не учитывая отчисления в социальные фонды. Они равны 5590 руб.

Стоимостная оценка затрат машинного времени определяется на основе амортизационных отчислений. Так как работы велись при пятидневной рабочей неделе, то в году получается 258 рабочих дней.

Годовая сумма амортизации одного компьютера равна

25000*0,2= 5000 руб.

Дневная сумма амортизации равна

5000/258=19,38 руб.

Данные об амортизации приведены в таблице 10.5.

Таблица 10.5 - Расходы на амортизацию

Специалист	Время работы, раб. дн.	Расходы на амортизацию, руб.
Бизнес-аналитик	20	387,6
Программист	58	1124,04
Всего		1511,64

Таким образом, сумма затрат на амортизацию равна 1511,64 руб.

Затраты на обслуживание оборудования составляют 25% от амортизационных отчислений и равны: 1511,64*0,25= 377,91 руб.

Затраты электроэнергии определяются на основе энергопотребления каждого компьютера. Эти затраты описаны в таблице 10.6. Они составляют 115,13 руб.

Таблица 10.6 - Затраты электроэнергии

Специалист	Время работы, час	Сумма оплаты, руб.
Бизнес-аналитик	160	82,56
Программист	464	239,42
Всего		321,92

Общая сумма затрат на разработку приведена в таблице 10.7.

Таблица 10.7 - Общая сумма затрат

Статья затрат	Сумма, руб.
Зарплата и отчисления	27090
Затраты на амортизацию	1511,64
Обслуживание оборудования	377,91
Электроэнергия	321,92
Всего	29301,47

Таким образом, на создание системы сбора информации об объектах капитального строительства требуется 29301,47 руб.

4.3 Формирование денежного потока

4.3.1 Расчет поступлений денежных средств

Разрабатываемая система предназначена для использования только в пределах одного предприятия. Разработка осуществлялась по заказу предприятия, поэтому все права на нее принадлежат этому предприятию. Из этого вытекает невозможность продажи системы кому-либо еще. Необходимо оценить эффективность внедрения такой системы для единственного заказчика.

Для оценки эффективности был использован метод внедрения информационной технологии как инвестиционного проекта. В соответствии с этим методом для оценки эффективности были проведены следующие мероприятия:

а) руководители подразделений подготовили техническое предложение по разработке системы;

б) специалисты управления информатизации проанализировали предложение и вынесли заключение о необходимости и возможности разработки данной системы;

в) специалистами управления информатизации совместно с менеджерами отделов были выявлены следующие исчисляемые эффекты от внедрения системы.

Исчисляемые эффекты:

а) отсутствие временных затрат на поиск как объектов капитального строительства, так и владельцев этих объектов. На поиск данных для одной заявки приходилось осуществлять поиск в 29 базах данных, потратив 1 минуту на поиск в каждой базе. Время на поиск в разработанной системе составляет в среднем 2 секунды (≈0,0333 минуты). Таким образом, время поиска в разработанной системе сократится в несколько раз. При заработной плате у техника, который

выдает ответы на запросы, в 5000 руб. и среднем количестве 22 рабочих дня в месяц экономия в день на ответ одного запроса составит 61,18 руб.

Дневная зарплата техника = 5000/22 = 227,27 руб.

Одна минута от з/п техника = 60 минут * 8 часов/227,27 = 2,112 руб.

Поиск данных на один запрос перед внедрением = 2,112*1*29=61,25 руб.

Поиск данных на один запрос после внедрения = 2,112*0,0333=0,07 руб.

Экономия на один запрос = 61,25 – 0,07 = 61,18 руб.

б) отсутствие затрат на телефонные переговоры. После внедрения системы техникам-инвентаризаторам будет необходимо только отметить необходимые данные в базе данных на местах, которые нужно передать в ФНС. В среднем стоимость одной минуты междугороднего телефонного разговора равна 2,40 руб. В день поступает 5 звонков, в среднем по 5 минут разговора. После внедрения системы экономия составит 67,5 руб. в день.

5*5*2,40 = 67,5 руб.

Данные об исчисляемых эффектах приведены в таблице 10.8.

Таблица 10.8 - Исчисляемые эффекты

Эффект	Сумма ежедневной экономии, руб.
Отсутствие временных затрат на поиск данных	61,18
Отсутствие затрат на телефонные переговоры	67,5
Всего экономии в день	128,68

Неисчисляемые эффекты от внедрения системы:

а) повышение уровня автоматизации и качества обработки информации. Кроме того, компьютер, производя все необходимые вычисления и операции, не ошибается. Следовательно, при выполнении расчетов машиной математические ошибки исключены;

б) целостность и актуальность имеющейся информации. Информация хранится в общей базе данных, а также существуют логические связи между ее отдельными элементами, что исключает хранение избыточной

информации. Кроме того, хранение информации в единой базе данных ведет к более прозрачному отражению деятельности предприятия в ежедневной и ежемесячной документации;

в) упрощение поиска и доступа к необходимой информации об объектах недвижимого имущества;

г) постоянная актуальность данных, т.к. происходит постоянное обновление загружаемых данных через Интернет (VPN-соединение).

После внедрения системы техники-инвентаризаторы не будут вести справочники адресов и коэффициентов инфляции, они будут заноситься автоматически.

На основе получаемых при внедрении информационной системы исчисляемых эффектов был выявлен срок окупаемости проекта. Он равен отношению затрат на разработку к сумме ежедневной прямой экономии:

29301,47 / 128,68,

что составляет 10,3 месяца. Дальнейшее использование информационной системы несет заказчику чистую прибыль.

При внедрении ПП может возникнуть риск, связанный с тем, что, пользователи воспримут новую систему неоднозначно, понадобится время на освоение. Однако, предполагается, что доверие к системе будет возрастать по мере того, как они будут ее осваивать.

Был утвержден срок разработки проекта - 65 рабочих дней.

4.3.2 Расчет выплат денежных средств

Целесообразно представить финансирование проекта по этапам разработки. Сумма средств, направленных на финансирование каждого этапа складывается из средств на оплату труда, затрат на амортизацию и обслуживание оборудования, накладных расходов, а также на электроэнергию (см. таблицу 10.9).

Таблица 10.9 - Финансирование проекта по этапам разработки

Наименование события	Исполнители	Время выполнения, дн.	Зарплата, руб	Машин, время и обслуж, руб	Электроэнергия, руб	Всего затрат, руб
Анализ предметной области	Бизнес-аналитик	4	1764	96,9	5,904	1866,80
Постановка требований к системе	Бизнес-аналитик	3	1323	72,675	4,428	1400,10
Изучение требований к системе	Программист	2	630	48,45	2,952	681,40
Разработка функциональной модели	Бизнес-аналитик Программист	3	2268	145,35	8,856	2422,21
Разработка инфологической модели	Бизнес-аналитик Программист	3	2268	145,35	8,856	2422,21
Разработка серверной части системы	Программист	10	3150	242,25	14,76	3407,01
Разработка программы сбора информации об	Программист	24	7560	581,4	35,424	8176,82
Отладка программы	Бизнес-аналитик Программист	5	3780	242,25	14,76	4037,01
Тестирование программы	Программист	4	1260	96,9	5,904	1362,80

Продолжение таблицы 1.9

Оформление документации и обучение персонала	Программист	5	1575	121,125	7,38	1703,51
Сдача в эксплуатацию	Бизнес-аналитик Программист	2	1512	96,9	5,904	1614,80

Итого			27090	1889,55	115,13	29094,68

График финансирования проекта по этапам разработки представлен на рисунке 10.2.

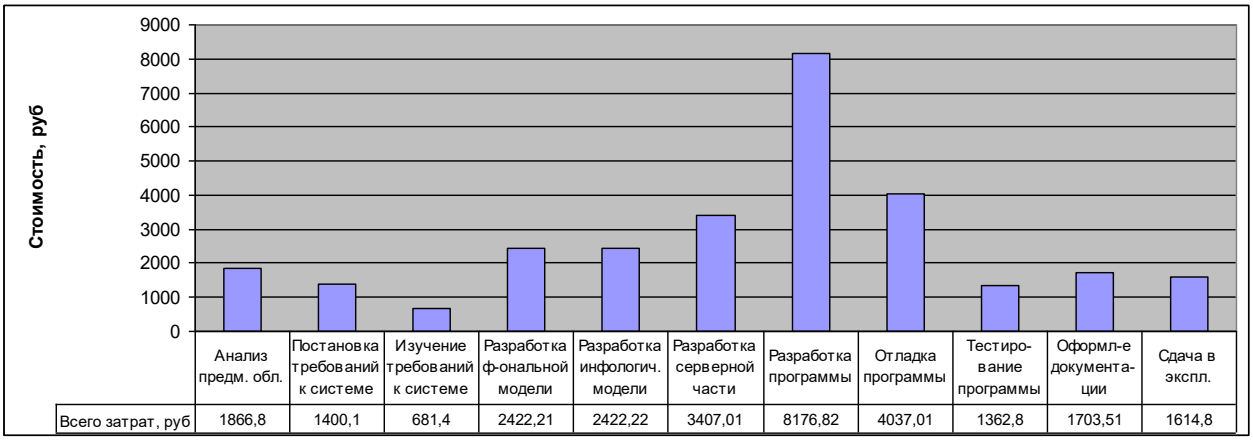


Рисунок 10.2 - Финансирование проекта по этапам разработки

График финансирования проекта по месяцам разработки представлен на рисунке 10.3.

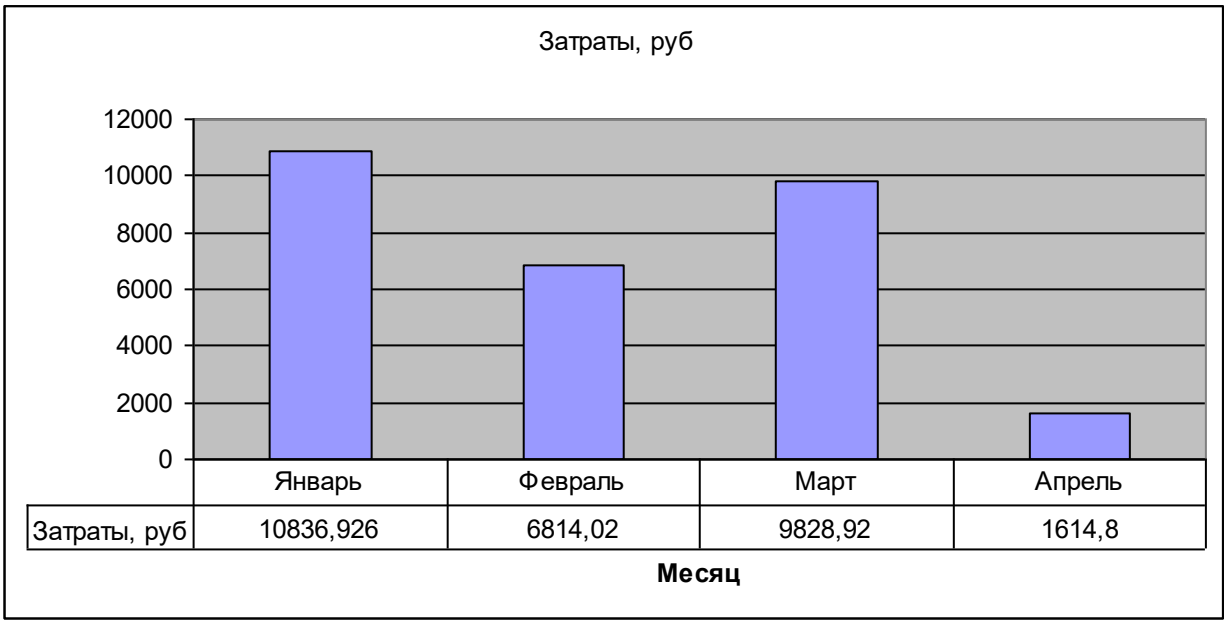


Рисунок 10.3 - Финансирование проекта по месяцам разработки

Выше приведенные расчеты не учитывают инфляционные процессы, поэтому не отражают в полной мере реальную стоимость разработки. Для полного учета используют приведение денежного потока к текущей

дисконтированной стоимости (ТДС) и чистой дисконтированной стоимости (ЧДС). Данные расчеты приведены в таблице 10.10.

Таблица 10.10 - ТДС и ЧДС

Номер месяца	Экономия	Денежный поток	Коэффициент дисконтирования	ТДС	ЧДС
1	0	-10836,926	0,992	-10750,51	-10750,51
2	0	-6814,02	0,984	-6705,35	-17455,86
3	0	-9828,92	0,976	-9593,79	-27049,65
4	2830,96	-1614,8	0,968	1052,79	-25996,86
5	2830,96	2830,96	0,960	5436,18	-20560,68
6	2830,96	2830,96	0,952	5391,03	-15169,61
7	2830,96	2830,96	0,944	5345,88	-9823,76
8	2830,96	2830,96	0,936	5300,73	-4523,03
9	2830,96	2830,96	0,928	5255,59	732,55
10	2830,96	2830,96	0,920	5210,44	5942,99
11	2830,96	2830,96	0,912	5165,29	11108,28
12	2830,96	2830,96	0,904	5120,14	16228,42
13	2830,96	2830,96	0,896	5074,99	21303,41
14	2830,96	2830,96	0,888	5029,85	26333,26
15	2830,96	2830,96	0,880	4984,70	31317,96

Продолжение таблицы 10.10

16	2830,96	2830,96	0,872	4939,55	36257,51
17	2830,96	2830,96	0,864	4894,40	41151,91
18	2830,96	2830,96	0,856	4849,25	46001,16

Текущая дисконтированная стоимость (ТДС) рассчитывается исходя из ставки дисконтирования (R), равной 10% в год. Месячная ставка определяется как:

$$(1+0{,}1)^{\frac{1}{12}} = 0{,}007974$$

ТДС рассчитывается путем произведения денежного потока на коэффициент дисконтирования K, который определяется так:

$$K = 1 - 0{,}007974 = 0{,}992$$

Исходя из расчетов ТДС и ЧДС видно, что на первых 3-х месяцах разработки денежный поток отрицательный, а, начиная с 4-го месяца, происходит экономия средств от использования системы. ЧДС принимает

положительное значение, начиная с 9 месяца использования системы с момента начала экономии. Дальнейшее ее применение несет заказчику прибыль.

4.4 Анализ рисков

Как и в любом инвестиционном проекте, при разработке программных продуктов возникают риски, причем для сферы информационных технологий эти риски специфические.

Финансовый риск возникает при минимальной норме рентабельности, заложенной в проект, при больших инвестиционных затратах, при длительном сроке окупаемости. Для снижения финансового риска в проект должна быть заложена достаточная норма рентабельности, срок окупаемости не должен быть слишком большим. Вообще, высокий финансовый риск допустим лишь в случае особо уникального проекта. В данном случае финансовый риск средний, так как проект разрабатывается не для продажи на рынке и финансируется заказчиком.

Технический риск связан с обеспечением эффективной технической поддержки для реализации проекта. Для уменьшения данного риска было использовано необходимое для разработки проекта высокопроизводительное аппаратное и программное обеспечение, кроме того, в штате заказчика имеются квалифицированные кадры, также участвующие в технической поддержке проекта.

Проектный риск возникает в том случае, когда масштаб сложности проекта не соответствует квалификации и опыту разработчиков. Чтобы снизить этот риск, для реализации проекта были привлечены специалисты с достаточным опытом и высокой квалификацией.

Функциональный риск связан с возможным несовпадением функционального наполнения продукта и требований пользователя. Для минимизации этого риска требования к продукту были подробно описаны заказчиком, в разработке участвовал бизнес-аналитик, все работы велись в

постоянном контакте с будущим пользователем, чтобы разработанный продукт максимально соответствовал требованиям пользователя.

Системный риск связан с нежеланием пользователя производить радикальные изменения в окружающей среде. Для снижения этого риска все изменения в работе пользователя, связанные с внедрением ИТ, проводились согласованно с руководством подразделения.

5 Оценка травмобезопасности на рабочем месте оператора пэвм

5.1 Общие положения

Травмобезопасность - свойство рабочих мест соответствовать требованиям безопасности труда, исключающим травмирование работающих в условиях установленных нормативно-правовыми актами.

Травмобезопасность рабочих мест обеспечивается исключением повреждений частей тела человека, которые могут быть получены в результате воздействия:

а) движущихся предметов механизмов или машин, а также неподвижными их элементами на рабочем месте (при механическом воздействии). Такими предметами являются зубчатые цепные клиноременные передачи, кривошипные механизмы, подвижные столы, вращающиеся детали, органы управления и т.п.;

б) электрического тока. Источником поражения могут быть незащищенные и неизолированные электропровода, поврежденные электродвигатели, открытые коммутаторы, не заземленное оборудование и др.;

в) агрессивных и ядовитых химических веществ. Например, химические ожоги сильными кислотами, едкими щелочами и ядовитыми химическими веществами (хлор аммиак и т.д. при попадании их на кожу или в легкие при вдыхании);

г) нагретых элементов оборудования перерабатываемого сырья, других теплоносителей (при термическом воздействии). Примерами таких элементов являются горячие трубопроводы, крышки котлов, танков, корпуса оборудования, детали холодильных установок и т.д.;

д) повреждения, полученные при падениях. Падения подразделяются на два вида: падения на человека различных предметов и падения человека в результате подскальзывания, запинания, падения с высоты или внезапного ухудшения здоровья.

При эксплуатации ЭВМ травмы могут быть получены в результате воздействия электрического тока и повреждений, полученных при падениях.

5.2 Обеспечение электробезопасности

Электрические установки, к которым относится практически все оборудование ЭВМ, представляют для человека большую потенциальную опасность, так как в процессе эксплуатации или проведении профилактических работ человек может коснуться частей, находящихся под напряжением. Специфическая опасность электроустановок заключается в том, что токоведущие проводники, корпуса стоек ЭВМ и прочего оборудования, оказавшегося под напряжением в результате повреждения (пробоя) изоляции, не подают каких-либо сигналов, которые предупреждают человека об опасности. Реакция человека на электрический ток возникает лишь при протекании последнего через тело человека. Исключительно важное значение для предотвращения электротравматизма имеет правильная организация обслуживания действующих электроустановок ВЦ, проведения ремонтных, монтажных и профилактических работ. При этом под правильной организацией понимается строгое выполнение ряда организационных и технических мероприятий и средств, установленных действующими "Правилами технической эксплуатации электроустановок потребителей и правилами техники безопасности при эксплуатации электроустановок потребителей" (ПТЭ и ПТБ потребителей) и "Правилами установки электроустановок" (ПУЭ). В зависимости от категории помещения необходимо принять определенные меры, обеспечивающие достаточную электробезопасность при эксплуатации и ремонте электрооборудования. Помещение ВЦ по опасности поражения электрическим током можно отнести к 1 классу, т.е. это помещение без повышенной опасности (сухое, беспыльное, с нормальной температурой воздуха, изолированными полами и малым числом заземленных приборов).

На рабочем месте оператора ПЭВМ из всего оборудования металлическим является лишь корпус системного блока компьютера, но здесь используются системные блоки, в которых кроме рабочей изоляции предусмотрен элемент для заземления и провод с заземляющей жилой для присоединения к источнику питания.

Системный блок компьютера имеет напряжения сигналов ТТЛ уровней (-1,+4 В), цифровые и аналоговые микросхемы запитываются постоянными напряжениями ±5 и ±12 В, которые получаются путем преобразования переменного напряжения 220В в блоке питания. Блок питания содержит в себе схемы преобразования напряжения, схемы стабилизации и схему защитного отключения при коротком замыкании. Так как корпус компьютера выполнен из металла, то существует опасность пробоя фазы на корпус. Мониторы современных компьютеров практически всегда изготовляются из пластика, поэтому несмотря на большое напряжение, присутствующее в мониторе, поражение током человека практически исключено. Однако в организации есть еще мониторы старого образца. Может возникнуть опасность по уровням напряженности электромагнитного поля. На расстоянии 5-10 см от экрана и корпуса монитора уровни напряженности могут достигать 140 В/м по электрической составляющей, что значительно превышает допустимые значения СанПиН 2.2.4.3359-16 (20 В/м). Для предупреждения внедрения опасной техники все дисплеи должны проходить испытания на соответствие требованиям безопасности (например международные стандарты MRP 2, TCO 99). Рекомендуется также применение защитных экранов, особенно для мониторов старого образца.

Электробезопасность помещения обеспечивается в соответствии с ПУЭ. Опасное и вредное воздействие на людей электрического тока, электрической дуги и электромагнитных полей проявляется в виде электротравм и профессиональных заболеваний.

Степень опасного и вредного воздействия на человека электрического тока, электрической дуги и электромагнитных полей зависит от:

а) рода и величины напряжения и тока;

б) частоты электрического тока;

в) пути тока через тело человека;

г) продолжительности воздействия на организм человека.

Рассмотрим основные причины поражения человека электрическим током на рабочем месте:

а) прикосновение к металлическим нетоковедущим частям (корпусу, периферии компьютера), которые могут оказаться под напряжением в результате повреждения изоляции.

б) нерегламентированное использование электрических приборов.

в) отсутствие инструктажа сотрудников по правилам электробезопасности.

Электробезопасность на рабочем месте оператора ПЭВМ обеспечивается техническими способами и средствами защиты, а также организационными и техническими мероприятиями.

Для обеспечения защиты от поражения электрическим током при прикосновении к металлическим нетоковедущим частям, которые могут оказаться под напряжением в результате повреждения изоляции, необходимо применять защитное заземление.

Заземление корпуса ЭВМ обеспечено подведением заземляющей жилы к питающим розеткам. Сопротивление заземления 4 Ом, согласно (ПУЭ) для электроустановок с напряжением до 1000 В.

Основным организационным мероприятием является инструктаж и обучение безопасным методам труда, а так же проверка знаний правил безопасности и инструкций в соответствии с занимаемой должностью применительно к выполняемой работе.

Также большую потенциальную опасность представляют собой работы без снятия напряжения на токоведущих частях и вблизи них, работы проводимые непосредственно на этих частях или при приближении к ним на расстояние менее установленного ПЭУ. К этим работам можно отнести

работы по наладке отдельных узлов, блоков. При выполнении такого рода работ в электроустановках до 1000 В необходимо применение определенных технических и организационных мер, таких как ограждения расположенные вблизи рабочего места и других токоведущих частей, к которым возможно случайное прикосновение; работа в диэлектрических перчатках или стоя на диэлектрическом коврике; применение инструмента с изолирующими рукоятками, при отсутствии такого инструмента следует пользоваться диэлектрическими перчатками. Работы этого вида должны выполнятся не менее чем двумя работниками.

В соответствии с ПТЭ и ПТВ потребителям и обслуживающему персоналу электроустановок предъявляются следующие требования :

а) лица, не достигшие 18-летнего возраста, не могут быть допущены к работам в электроустановках;

б) лица не должны иметь увечий и болезней, мешающих производственной работе;

в) лица должны после соответствующей теоретической и практической подготовки пройти проверку знаний и иметь удостоверение на доступ к работам в электроустановках.

В процессе работы на корпусе компьютера накапливается статическое электричество. На расстоянии 5-10 см от экрана напряженность электростатического поля составляет 60-280 кВ/м, то есть в 10 раз превышает норму 20 кВ/м. Для уменьшения напряжённости применять увлажнители и нейтрализаторы, антистатическое покрытия пола.

5.3 Требования безопасности при работе на ПЭВМ

Требования безопасности перед началом работ:

а) необходимо проверить внешним осмотром и убедиться в исправности соеденительных кабелей и шнуров, блока бесперебойного питания (UPS), системного блока, монитора, клавиатуры, внешних устройств ЭВМ;

б) важно проверить состояние общего освещения рабочего места;

в) нельзя производить каких-либо работ по ремонту блока бесперебойного питания, системного блока, монитора, внешних устройств ЭВМ;

г) запрещается после включения ЭВМ перемещать блок бесперебойного питания, системный блок, монитор, внешние устройства ЭВМ;

д) обо всех недостатках и неисправностях, обнаруженных при осмотре на рабочем месте, доложить старшему смены (инженеру) для принятия мер к их полному устранению;

е) расположить клавиатуру, манипулятор "мышь" на рабочем месте с максимальным удобством для пользования, не допуская наличия в зоне работы лишних предметов;

ж) запрещается курить в помещении, в котором расположены ЭВМ и ее внешние устройства;

з) запрещается прием пищи вблизи ЭВМ и ее внешних устройств.

Требования безопасности во время работы;

а) при работе оборудования все крышки и кожухи должны быть закрыты;

б) при работе на оборудовании нельзя касаться токоведущих частей штепселей и вилок, соеденительных шнуров и кабелей, защитного кожуха монитора;

в) шнуры должны быть в полной иправности: изоляционные втулки штепселей не должны иметь трещин, а шнуры - оголенных мест;

г) включать шнуры и провода можно только при выключенном оборудовании, держа их за изоляционные втулки;

д) при работе с печатающими устройствами и другими внешними устройствами ЭВМ нельзя открывать защитные крышки и касаться движущихся узлов, верхние крышки печатающих устройств должны быть закрыты;

е) проверка на отсутствие замыкания на корпус и состояние изоляции производится специально назначенным лицом не реже 1 раза в 6 месяцев;

ж) двери и окна помещения необходимо держать всегда закрытыми;

з) руки, одежда и обувь персонала должны быть всегда сухими;

Требования безопасности в аварийных ситуациях:

а) каждый работник, обнаруживший нарушения требований настоящей инструкции и правил охраны труда или заметивший неисправность оборудования, представляющую опасность для людей, обязан сообщить об этом непосредственному руководителю. В тех случаях, когда неисправность оборудования представляет угрожающую опасность для людей или самого оборудования, работник обязан принять меры по прекращению действия оборудования, а затем известить об этом непосредственного руководителя. Устранение неисправности производится при соблюдении требований безопасности;

б) если во время работы произошёл несчастный случай, необходимо немедленно оказать первую медицинскую помощь пострадавшему, доложить о случившемся своему непосредственному начальнику и принять меры для сохранения обстановки несчастного случая, если это сопряжено с опасностью для жизни и здоровья людей;

в) при поражении электрическим током необходимо как можно скорее освободить пострадавшего от действия тока. Отключение оборудования произвести с помощью выключателей, разъёма штепсельного соединения, перерубить питающий провод инструментом с изолированными ручками. Если отключить оборудование достаточно быстро нельзя, необходимо принять другие меры к освобождению пострадавшего от действия тока. Для отделения пострадавшего от токоведущих частей или провода следует воспользоваться палкой, доской или каким-либо сухим предметом, не проводящим электроток, при этом оказывающий помощь должен встать на сухое, непроводящее ток место, или надеть диэлектрические перчатки;

г) при возникновении пожара в техническом помещении следует немедленно приступить к его тушению имеющимися средствами

(углекислотные огнетушители, асбестовые покрывало, песок) и вызвать пожарную часть;

д) при обнаружении постороннего напряжения на рабочем месте необходимо немедленно прекратить работу и доложить старшему смены.

Требования безопасности по окончании работы:

а) необходимо привести в порядок рабочее место;

б) сообщить сменщику (старшему смены) обо всех неисправностях, замеченных во время работы, и мерах, принятых к их устранению.

Заключение

В данном разделе дипломной проекта были изложены требования электробезопасности и требования безопасности при работе на ПЭВМ в течение всего рабочего дня. Соблюдение этих требований позволит избежать аварийных ситуаций, а также травмирования людей, работающих на ПЭВМ.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта было разработано мобильное приложение, обеспечивающее сканирование штрихкода товара на упаковке и предоставление информации о товаре в соответствии с данными из СУБД. Данный проект это мобильное приложение на основе клиент серверного подхода, созданное на основе краудсорсинговых данных пользователей. Приложение интегрируется со сторонними службами для создания базы данных картографирования товаров. Приложение построено с использованием технологий, которые являются производительными, гибкими и масштабируемыми.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Barry, Burd Android Application Development/ Barry Burd. - Москва: Машиностроение, 2017. - 816 с.
2. Биллиг, В. А. Основы объектного программирования на Ionic () / В.А. Биллиг. - М.: Интернет-университет информационных технологий, Бином., 2015. - 584 с.
3. Гарнаев, А. М. программирование Android на Java и JavaScript / А.М. Гарнаев , С.И. Гарнаев. - М.: БХВ-Петербург, 2016. - 179 с.
4. Голощапов, А. С. Google Android. Программирование для мобильных устройств (+ CD-ROM) / А.С. Голощапов. - М.: БХВ-Петербург, 2015. - 438 с.
5. Голощапов, А.С. Google Android. Программирование для мобильных устройств / А.С. Голощапов. - М.: БХВ-Петербург, 2018. - 448 с.
6. Андреев С.И. Создание приложения на Ionic с использованием API- М.: "Издательство "Эксмо", 2018. - 472 с.
7. Игнатов В.Ф. Основные приемы работы в Ionic API- М.: "Издательство "Известия", 2015. - 472 с.

ПРИЛОЖЕНИЕ А

(обязательное)

Исходный код приложения

MainActivity.java

```
package me.dm7.barcodescanner.zbar.sample;
import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
    private static final int ZBAR_CAMERA_PERMISSION = 1;
    private Class<?> mClss;
    @Override
    public void onCreate(Bundle state) {
        super.onCreate(state);
        setContentView(R.layout.activity_main);
        setupToolbar();
    }
    public void setupToolbar() {
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
    }
    public void launchSimpleActivity(View v) {
        launchActivity(SimpleScannerActivity.class);
    }
    public void launchSimpleFragmentActivity(View v) {
        launchActivity(SimpleScannerFragmentActivity.class);
    }
    public void launchFullActivity(View v) {
        launchActivity(FullScannerActivity.class);
    }
    public void launchFullFragmentActivity(View v) {
        launchActivity(FullScannerFragmentActivity.class);
    }
    public void launchActivity(Class<?> clss) {
        if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA)
                != PackageManager.PERMISSION_GRANTED) {
            mClss = clss;
```

```
            ActivityCompat.requestPermissions(this,
                    new String[]{Manifest.permission.CAMERA}, ZBAR_CAMERA_PERMISSION);
        } else {
            Intent intent = new Intent(this, clss);
            startActivity(intent);
        }
    }
    @Override
    public void onRequestPermissionsResult(int requestCode,  String permissions[], int[]
grantResults) {
        switch (requestCode) {
            case ZBAR_CAMERA_PERMISSION:
                if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                    if(mClss != null) {
                        Intent intent = new Intent(this, mClss);
                        startActivity(intent);
                    }
                } else {
                    Toast.makeText(this, "Please grant camera permission to use the QR
Scanner", Toast.LENGTH_SHORT).show();
                }
                return;
        }
    }
}
```

DialogProduct.java

```
        import android.app.AlertDialog;
        import android.app.Dialog;
        import android.content.DialogInterface;
        import android.os.Bundle;
        import android.support.v4.app.DialogFragment;
        public class Dialogproduct extends Dialog {
            public interface ProductDialogListener {
                public void onDialogPositiveClick(DialogFragment dialog);
            }
            private String mTitle;
            private String mMessage;
            private MessageDialogListener mListener;
            public void onCreate(Bundle state) {
                super.onCreate(state);
                setRetainInstance(true);
            }
            public static ProductDialogFragment newInstance(String title, String message,
        MessageDialogListener listener) {
```

```
            MessageDialogFragment fragment = new MessageDialogFragment();
            fragment.mTitle = title;
            fragment.mMessage = message;
            fragment.mListener = listener;
            return fragment;
        }
        @Override
        public Dialog onCreateDialog(Bundle savedInstanceState) {
            AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
            builder.setMessage(mMessage)
                    .setTitle(mTitle);
            builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                    if(mListener != null) {
                        mListener.onDialogPositiveClick(MessageDialogFragment.this);
                    }
                }
            });
            return builder.create();
        }
    }
```

BaseScannerActivity.java

```
package me.dm7.barcodescanner.zbar.sample;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
public class BaseScannerActivity extends AppCompatActivity {
    public void setupToolbar() {
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        final ActionBar ab = getSupportActionBar();
        if(ab != null) {
            ab.setDisplayHomeAsUpEnabled(true);
        }
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            // Respond to the action bar's Up/Home button
            case android.R.id.home:
                finish();
                return true;
```

```
        }
        return super.onOptionsItemSelected(item);
    }
}
    public class CameraSelectorDialogFragment extends DialogFragment {
    public interface CameraSelectorDialogListener {
        public void onCameraSelected(int cameraId);
    }
    private int mCameraId;
    private CameraSelectorDialogListener mListener;
    public void onCreate(Bundle state) {
        super.onCreate(state);
        setRetainInstance(true);
    }
    public static CameraSelectorDialogFragment newInstance(CameraSelectorDialogListener
listener, int cameraId) {
        CameraSelectorDialogFragment fragment = new CameraSelectorDialogFragment();
        fragment.mCameraId = cameraId;
        fragment.mListener = listener;
        return fragment;
    }
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        if(mListener == null) {
            dismiss();
            return null;
        }
        int numberOfCameras = Camera.getNumberOfCameras();
        String[] cameraNames = new String[numberOfCameras];
        int checkedIndex = 0;
        for (int i = 0; i < numberOfCameras; i++) {
            Camera.CameraInfo info = new Camera.CameraInfo();
            Camera.getCameraInfo(i, info);
            if(info.facing == Camera.CameraInfo.CAMERA_FACING_FRONT) {
                cameraNames[i] = "Front Facing";
            } else if(info.facing == Camera.CameraInfo.CAMERA_FACING_BACK) {
                cameraNames[i] = "Rear Facing";
            } else {
                cameraNames[i] = "Camera ID: " + i;
            }
            if(i == mCameraId) {
                checkedIndex = i;
            }
        }
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
        // Set the dialog title
        builder.setTitle(R.string.select_camera)
```

```
                // Specify the list array, the items to be selected by default (null for
none),
                // and the listener through which to receive callbacks when items are
selected
                .setSingleChoiceItems(cameraNames, checkedIndex,
                        new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog, int which) {
                                mCameraId = which;
                            }
                        })
                        // Set the action buttons
                .setPositiveButton(R.string.ok_button, new DialogInterface.OnClickListener()
{
                    @Override
                    public void onClick(DialogInterface dialog, int id) {
                        // User clicked OK, so save the mSelectedIndices results somewhere
                        // or return them to the component that opened the dialog
                        if (mListener != null) {
                            mListener.onCameraSelected(mCameraId);
                        }
                    }
                })
                .setNegativeButton(R.string.cancel_button, new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int id) {
                    }
                });
        return builder.create();
    }
}
```

ViewFinderView.java

```
import android.content.Context;
import android.content.res.Configuration;
import android.graphics.Canvas;
import android.graphics.CornerPathEffect;
import android.graphics.Paint;
import android.graphics.Path;
import android.graphics.Point;
import android.graphics.Rect;
import android.util.AttributeSet;
import android.view.View;
public class ViewFinderView extends View implements IViewFinder {
    private static final String TAG = "ViewFinderView";
```

```
    private Rect mFramingRect;
    private static final float PORTRAIT_WIDTH_RATIO = 6f/8;
    private static final float PORTRAIT_WIDTH_HEIGHT_RATIO = 0.75f;
    private static final float LANDSCAPE_HEIGHT_RATIO = 5f/8;
    private static final float LANDSCAPE_WIDTH_HEIGHT_RATIO = 1.4f;
    private static final int MIN_DIMENSION_DIFF = 50;
    private static final float DEFAULT_SQUARE_DIMENSION_RATIO = 5f / 8;
    private static final int[] SCANNER_ALPHA = {0, 64, 128, 192, 255, 192, 128, 64};
    private int scannerAlpha;
    private static final int POINT_SIZE = 10;
    private static final long ANIMATION_DELAY = 80l;
    private final int mDefaultLaserColor =
getResources().getColor(R.color.viewfinder_laser);
    private final int mDefaultMaskColor =
getResources().getColor(R.color.viewfinder_mask);
    private final int mDefaultBorderColor =
getResources().getColor(R.color.viewfinder_border);
    private final int mDefaultBorderStrokeWidth =
getResources().getInteger(R.integer.viewfinder_border_width);
    private final int mDefaultBorderLineLength =
getResources().getInteger(R.integer.viewfinder_border_length);
    protected Paint mLaserPaint;
    protected Paint mFinderMaskPaint;
    protected Paint mBorderPaint;
    protected int mBorderLineLength;
    protected boolean mSquareViewFinder;
    private boolean mIsLaserEnabled;
    private float mBordersAlpha;
    private int mViewFinderOffset = 0;
    public ViewFinderView(Context context) {
        super(context);
        init();
    }
    public ViewFinderView(Context context, AttributeSet attributeSet) {
        super(context, attributeSet);
        init();
    }
    private void init() {
        //set up laser paint
        mLaserPaint = new Paint();
        mLaserPaint.setColor(mDefaultLaserColor);
        mLaserPaint.setStyle(Paint.Style.FILL);
        //finder mask paint
        mFinderMaskPaint = new Paint();
        mFinderMaskPaint.setColor(mDefaultMaskColor);
        //border paint
        mBorderPaint = new Paint();
        mBorderPaint.setColor(mDefaultBorderColor);
```

```
            mBorderPaint.setStyle(Paint.Style.STROKE);
            mBorderPaint.setStrokeWidth(mDefaultBorderStrokeWidth);
            mBorderPaint.setAntiAlias(true);
            mBorderLineLength = mDefaultBorderLineLength;
        }
        @Override
        public void setLaserColor(int laserColor) {
            mLaserPaint.setColor(laserColor);
        }
        @Override
        public void setMaskColor(int maskColor) {
            mFinderMaskPaint.setColor(maskColor);
        }
        @Override
        public void setBorderColor(int borderColor) {
            mBorderPaint.setColor(borderColor);
        }
        @Override
        public void setBorderStrokeWidth(int borderStrokeWidth) {
            mBorderPaint.setStrokeWidth(borderStrokeWidth);
        }
        @Override
        public void setBorderLineLength(int borderLineLength) {
            mBorderLineLength = borderLineLength;
        }
        @Override
        public void setLaserEnabled(boolean isLaserEnabled) { mIsLaserEnabled =
    isLaserEnabled; }
        @Override
        public void setBorderCornerRounded(boolean isBorderCornersRounded) {
            if (isBorderCornersRounded) {
                mBorderPaint.setStrokeJoin(Paint.Join.ROUND);
            } else {
                mBorderPaint.setStrokeJoin(Paint.Join.BEVEL);
            }
        }
        @Override
        public void setBorderAlpha(float alpha) {
            int colorAlpha = (int) (255 * alpha);
            mBordersAlpha = alpha;
            mBorderPaint.setAlpha(colorAlpha);
        }
        @Override
        public void setBorderCornerRadius(int borderCornersRadius) {
            mBorderPaint.setPathEffect(new CornerPathEffect(borderCornersRadius));
        }
        @Override
        public void setViewFinderOffset(int offset) {
```

```
            mViewFinderOffset = offset;
        }
        // TODO: Need a better way to configure this. Revisit when working on 2.0
        @Override
        public void setSquareViewFinder(boolean set) {
            mSquareViewFinder = set;
        }
        public void setupViewFinder() {
            updateFramingRect();
            invalidate();
        }
        public Rect getFramingRect() {
            return mFramingRect;
        }
        @Override
        public void onDraw(Canvas canvas) {
            if(getFramingRect() == null) {
                return;
            }
            drawViewFinderMask(canvas);
            drawViewFinderBorder(canvas);
            if (mIsLaserEnabled) {
                drawLaser(canvas);
            }
        }
        public void drawViewFinderMask(Canvas canvas) {
            int width = canvas.getWidth();
            int height = canvas.getHeight();
            Rect framingRect = getFramingRect();

            canvas.drawRect(0, 0, width, framingRect.top, mFinderMaskPaint);
            canvas.drawRect(0, framingRect.top, framingRect.left, framingRect.bottom +
1, mFinderMaskPaint);
            canvas.drawRect(framingRect.right + 1, framingRect.top, width,
framingRect.bottom + 1, mFinderMaskPaint);
            canvas.drawRect(0, framingRect.bottom + 1, width, height, mFinderMaskPaint);
        }
        public void drawViewFinderBorder(Canvas canvas) {
            Rect framingRect = getFramingRect();
            // Top-left corner
            Path path = new Path();
            path.moveTo(framingRect.left, framingRect.top + mBorderLineLength);
            path.lineTo(framingRect.left, framingRect.top);
            path.lineTo(framingRect.left + mBorderLineLength, framingRect.top);
            canvas.drawPath(path, mBorderPaint);
            // Top-right corner
            path.moveTo(framingRect.right, framingRect.top + mBorderLineLength);
            path.lineTo(framingRect.right, framingRect.top);
```

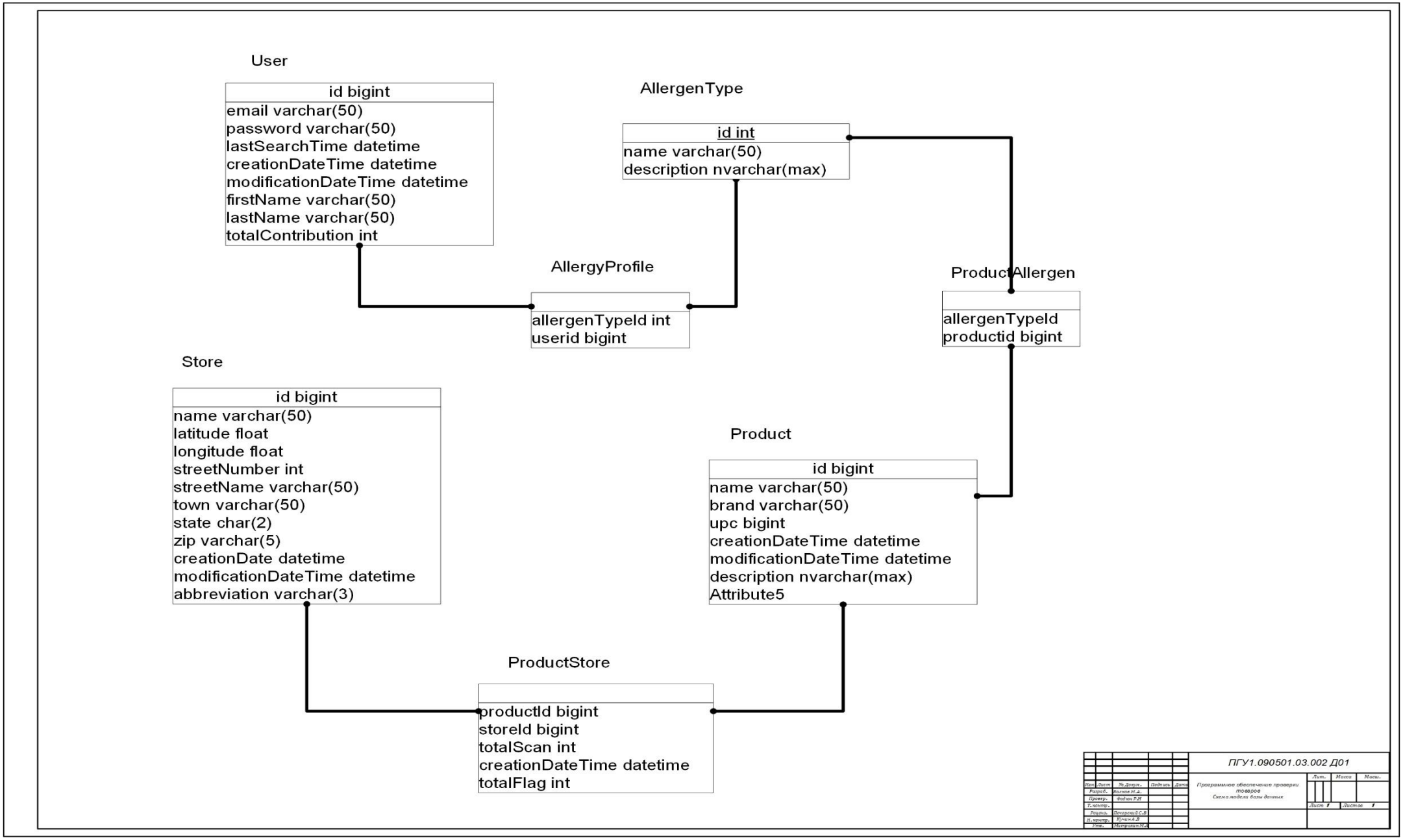
```
            path.lineTo(framingRect.right - mBorderLineLength, framingRect.top);
            canvas.drawPath(path, mBorderPaint);
            // Bottom-right corner
            path.moveTo(framingRect.right, framingRect.bottom - mBorderLineLength);
            path.lineTo(framingRect.right, framingRect.bottom);
            path.lineTo(framingRect.right - mBorderLineLength, framingRect.bottom);
            canvas.drawPath(path, mBorderPaint);
            // Bottom-left corner
            path.moveTo(framingRect.left, framingRect.bottom - mBorderLineLength);
            path.lineTo(framingRect.left, framingRect.bottom);
            path.lineTo(framingRect.left + mBorderLineLength, framingRect.bottom);
            canvas.drawPath(path, mBorderPaint);
        }
        public void drawLaser(Canvas canvas) {
            Rect framingRect = getFramingRect();

            // Draw a red "laser scanner" line through the middle to show decoding is
active
            mLaserPaint.setAlpha(SCANNER_ALPHA[scannerAlpha]);
            scannerAlpha = (scannerAlpha + 1) % SCANNER_ALPHA.length;
            int middle = framingRect.height() / 2 + framingRect.top;
            canvas.drawRect(framingRect.left + 2, middle - 1, framingRect.right - 1,
middle + 2, mLaserPaint);
            postInvalidateDelayed(ANIMATION_DELAY,
                    framingRect.left - POINT_SIZE,
                    framingRect.top - POINT_SIZE,
                    framingRect.right + POINT_SIZE,
                    framingRect.bottom + POINT_SIZE);
        }
        @Override
        protected void onSizeChanged(int xNew, int yNew, int xOld, int yOld) {
            updateFramingRect();
        }
        public synchronized void updateFramingRect() {
            Point viewResolution = new Point(getWidth(), getHeight());
            int width;
            int height;
            int orientation = DisplayUtils.getScreenOrientation(getContext());
            if(mSquareViewFinder) {
                if(orientation != Configuration.ORIENTATION_PORTRAIT) {
                    height = (int) (getHeight() * DEFAULT_SQUARE_DIMENSION_RATIO);
                    width = height;
                } else {
                    width = (int) (getWidth() * DEFAULT_SQUARE_DIMENSION_RATIO);
                    height = width;
                }
            } else {
                if(orientation != Configuration.ORIENTATION_PORTRAIT) {
```

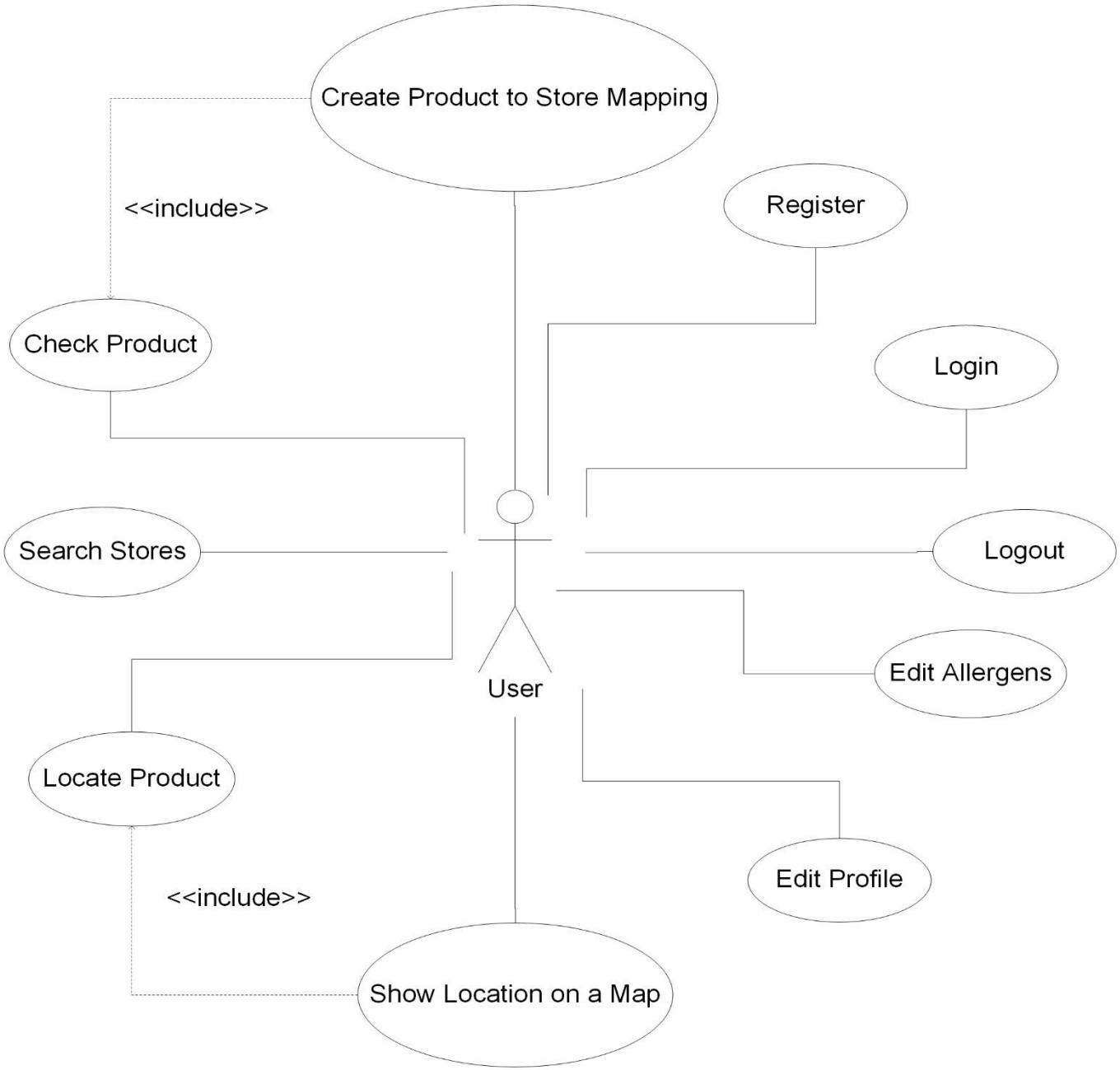
```
                height = (int) (getHeight() * LANDSCAPE_HEIGHT_RATIO);
                width = (int) (LANDSCAPE_WIDTH_HEIGHT_RATIO * height);
            } else {
                width = (int) (getWidth() * PORTRAIT_WIDTH_RATIO);
                height = (int) (PORTRAIT_WIDTH_HEIGHT_RATIO * width);
            }
        }
        if(width > getWidth()) {
            width = getWidth() - MIN_DIMENSION_DIFF;
        }
        if(height > getHeight()) {
            height = getHeight() - MIN_DIMENSION_DIFF;
        }
        int leftOffset = (viewResolution.x - width) / 2;
        int topOffset = (viewResolution.y - height) / 2;
        mFramingRect = new Rect(leftOffset + mViewFinderOffset, topOffset +
mViewFinderOffset, leftOffset + width - mViewFinderOffset, topOffset + height -
mViewFinderOffset);
    }
}
```

ПРИЛОЖЕНИЕ Б

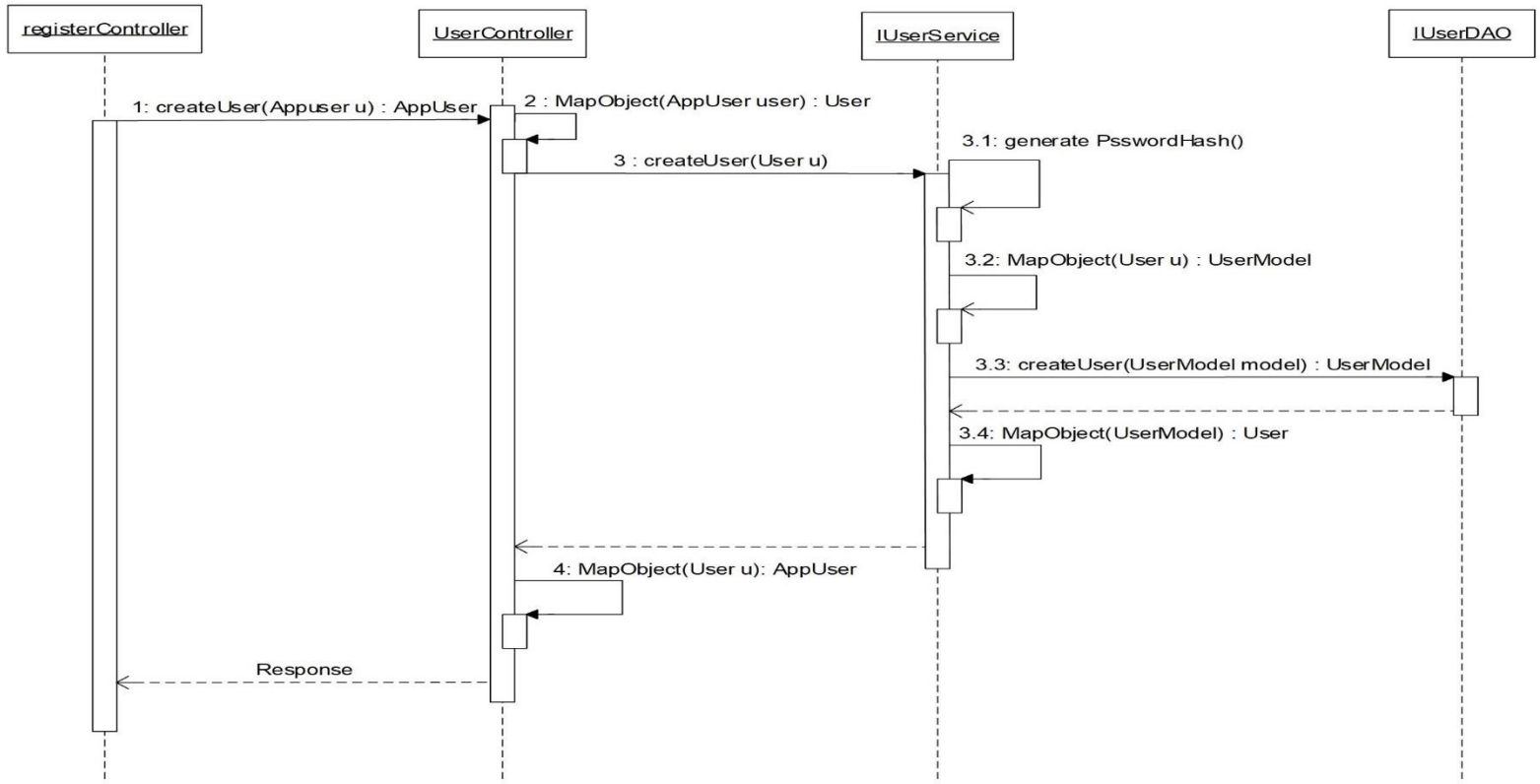
(обязательное)

СХЕМЫ

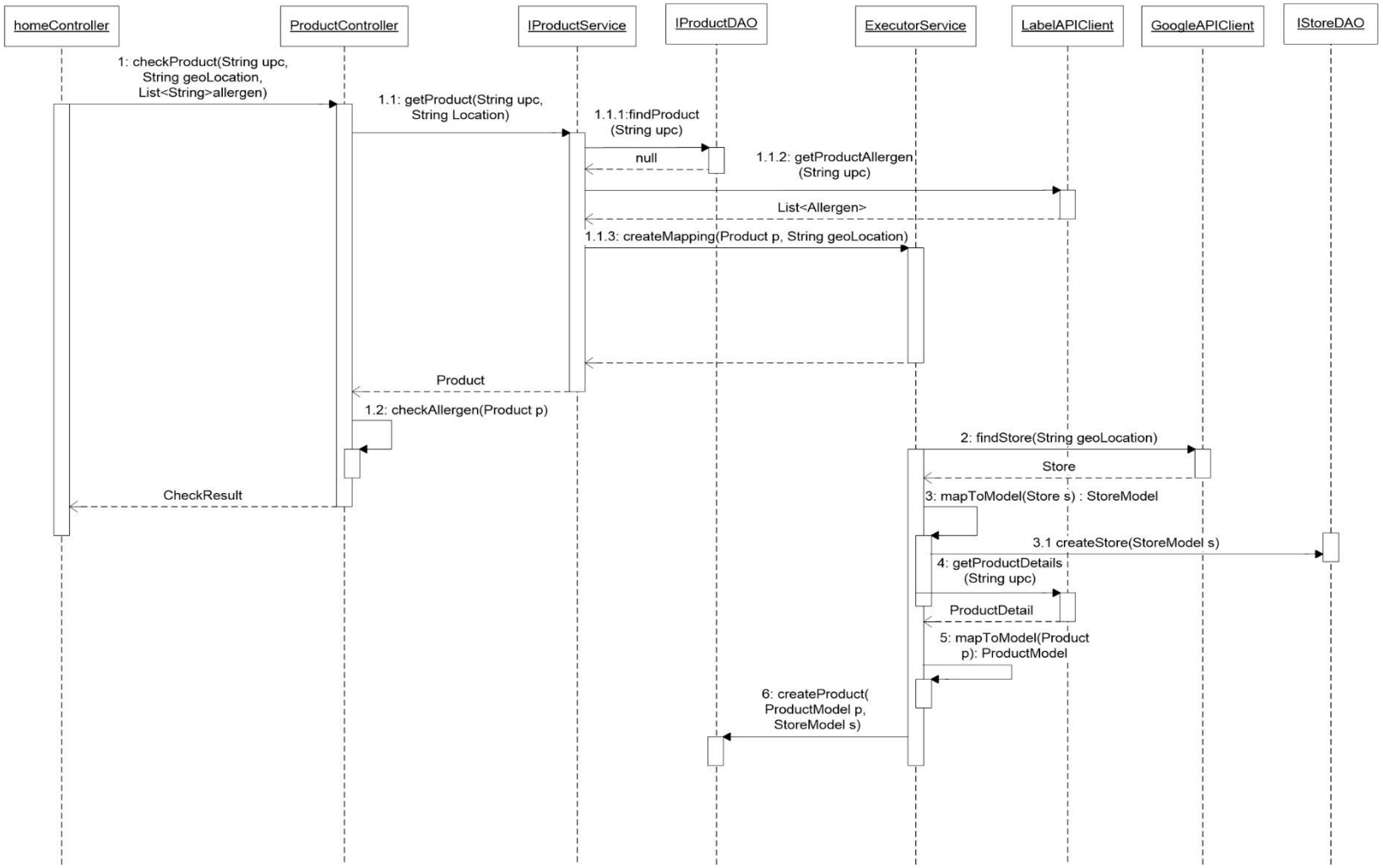

User
id bigint
email varchar(50)
password varchar(50)
lastSearchTime datetime
creationDateTime datetime
modificationDateTime datetime
firstName varchar(50)
lastName varchar(50)
totalContribution int
AllergenType
id int
name varchar(50)
description nvarchar(max)
AllergyProfile
allergenTypeId int
userid bigint
ProductAllergen
allergenTypeId
productid bigint
Store
id bigint
name varchar(50)
latitude float
longitude float
streetNumber int
streetName varchar(50)
town varchar(50)
state char(2)
zip varchar(5)
creationDate datetime
modificationDateTime datetime
abbreviation varchar(3)
Product
id bigint
name varchar(50)
brand varchar(50)
upc bigint
creationDateTime datetime
modificationDateTime datetime
description nvarchar(max)
Attribute5
ProductStore
productId bigint
storeId bigint
totalScan int
creationDateTime datetime
totalFlag int
ПГУ1.090501.03.002 Д01
Программное обеспечение проверки товаров
Схема модели базы данных



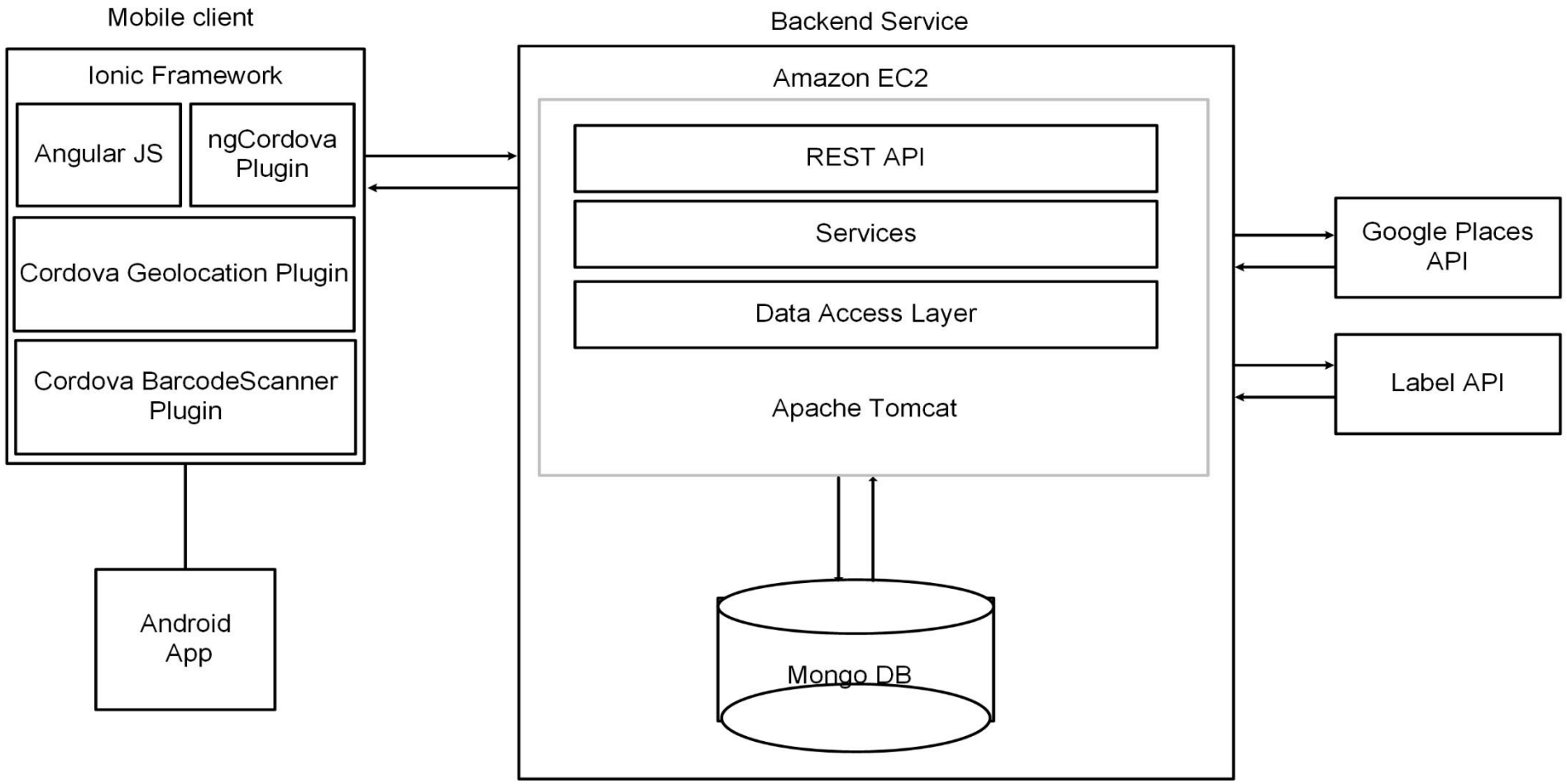
						ПГУ1.090501.03.003 Д02
Изм.	Лист	№ докум.	Подпись	Дата		
Разраб.		Волков М.А.			Программное обеспечение проверки товаров Диаграмма использования	Лит. / Масса / Масш.
Провер.		Федонин Р.Ю.				
Т.контр.						Лист 1 / Листов 1
Реценз.		Вечерский С.В.				
Н.контр.		Кучин А.В.				
Утв.		Мутрохин М.А.				



					ПГУ1.090501.03.004 Д03	
Изм.	Лист	№ докум.	Подпись	Дата		
Разраб.		Волков М.А.			Программное обеспечение проверки товаров Диаграмма последовательности	Лит. / Масса / Масш.
Провер.		Федюкин Р.Н				
Т.контр.						Лист 1 / Листов 1
Реценз.		Начаркий С.В.				
Н. контр.		Кучин А.Ю.				
Утв.		Митрофанов М.А.				



					ПГУ1.090501.03.005 Д03	
Изм.	Лист	№ докум.	Подпись	Дата		
Разраб.		Волков М.А.			Программное обеспечение проверки товаров Диаграмма последовательности	Лит. / Масса / Масш.
Провер.		Федюшин Р.Н.				
Т. контр.						Лист 1 / Листов 1
Реценз.						
Н. контр.		Кунцик А.В.				
Утв.		Ибатуллин М.Д.				

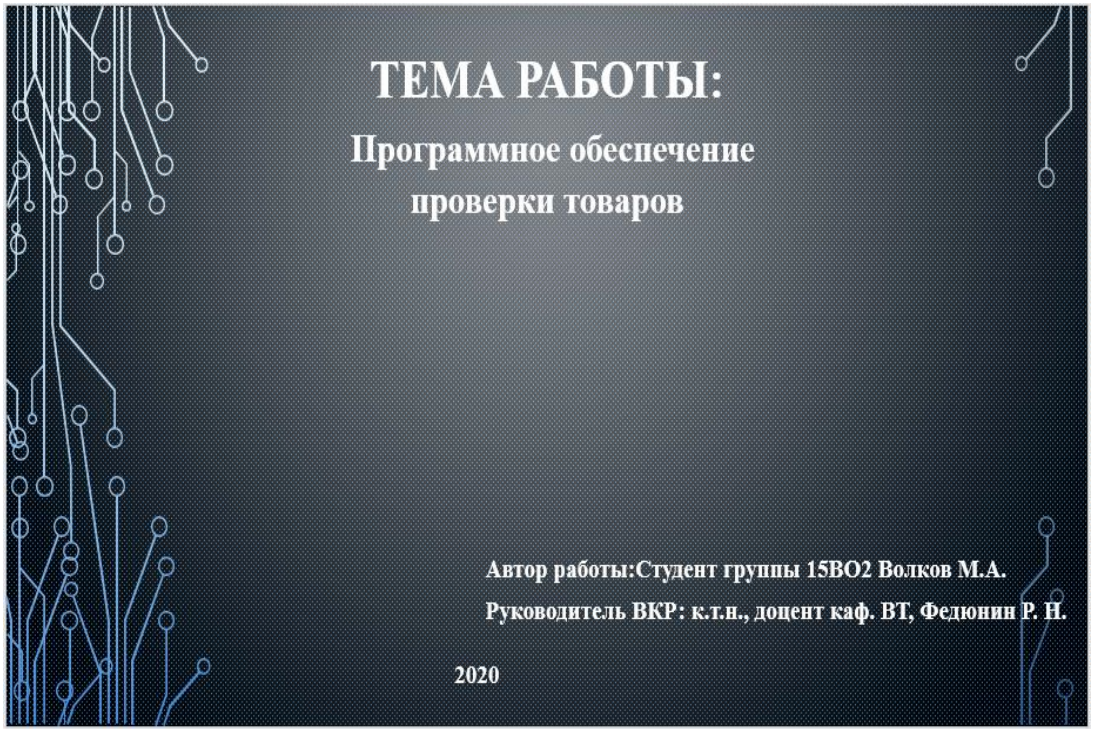


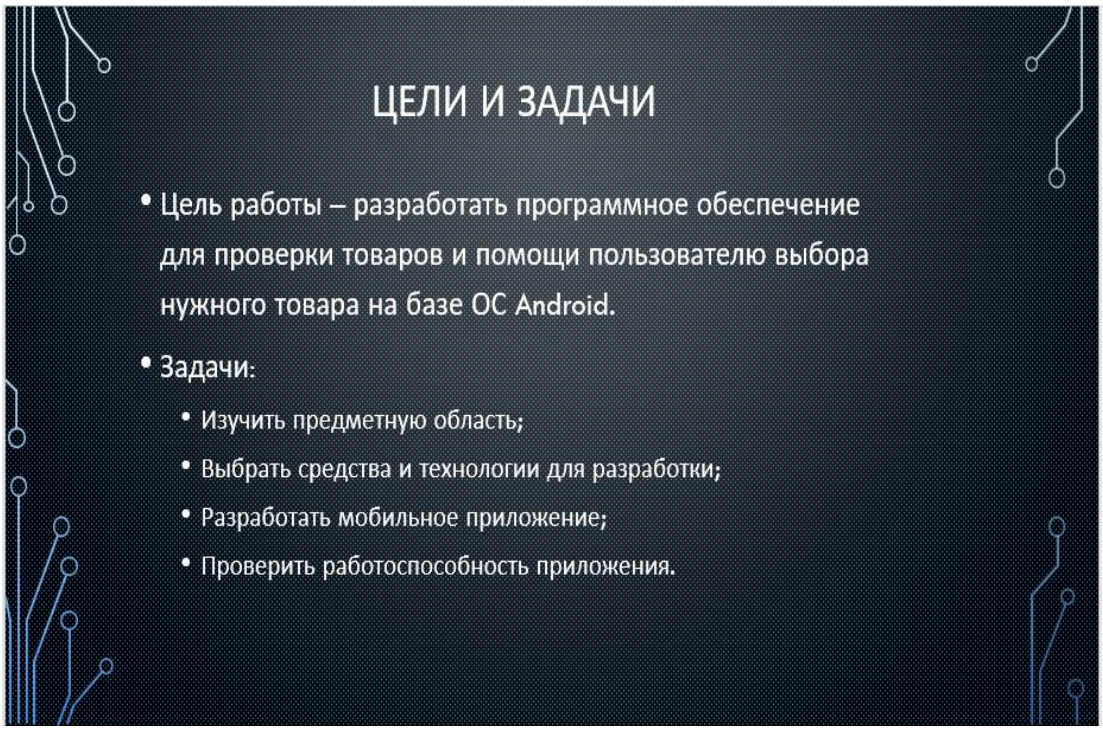
						ПГУ1.090501.03.006 Д04	
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Волкова М.А			Программное обеспечение проверки товаров Структурная схема	Лит.	Масса	Масш.
Провер.		Федонин Р.Н						
Т. контр.						Лист 1	Листов 1	
Реценз.		Печерников В.						
Н. контр.		Кучин А.Д.						
Утв.		Катрошин М.А						

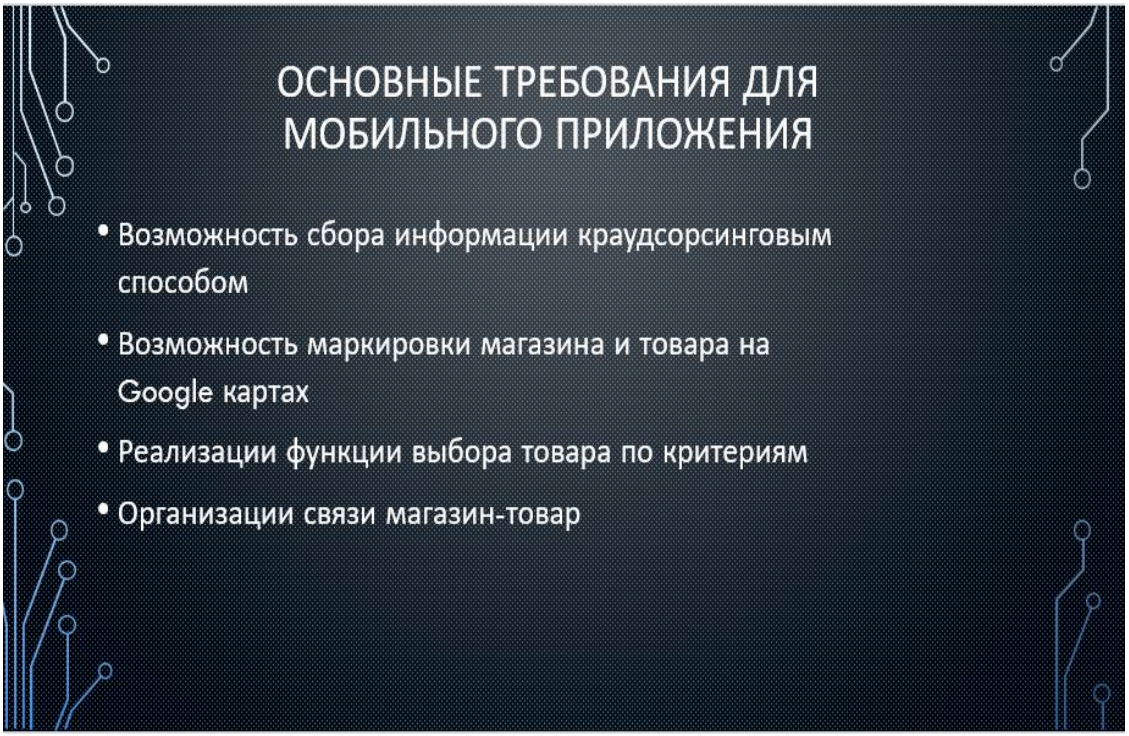
ПРИЛОЖЕНИЕ В

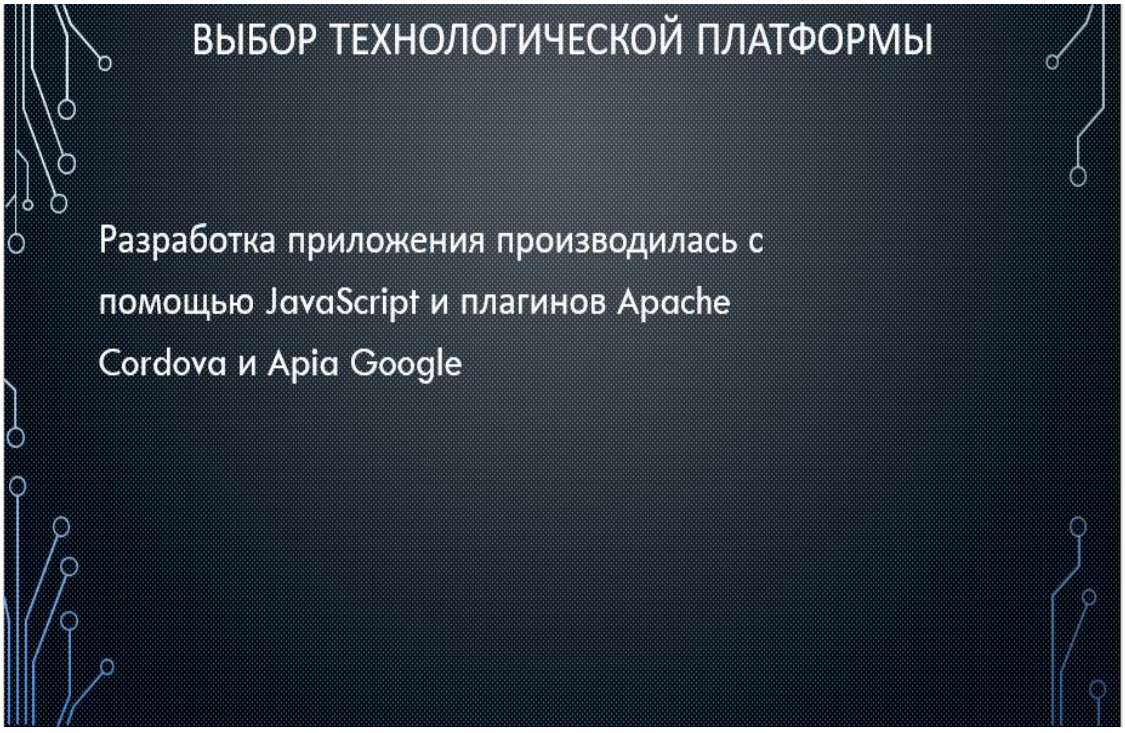
(обязательное)

Презентация


ТЕМА РАБОТЫ:
Программное обеспечение
проверки товаров
Автор работы:Студент группы 15ВО2 Волков М.А.
Руководитель ВКР: к.т.н., доцент каф. ВТ, Федюнин Р. Н.
2020


ЦЕЛИ И ЗАДАЧИ
• Цель работы – разработать программное обеспечение для проверки товаров и помощи пользователю выбора нужного товара на базе ОС Android.
• Задачи:
• Изучить предметную область;
• Выбрать средства и технологии для разработки;
• Разработать мобильное приложение;
• Проверить работоспособность приложения.


ОСНОВНЫЕ ТРЕБОВАНИЯ ДЛЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ
• Возможность сбора информации краудсорсинговым способом
• Возможность маркировки магазина и товара на Google картах
• Реализации функции выбора товара по критериям
• Организации связи магазин-товар


ВЫБОР ТЕХНОЛОГИЧЕСКОЙ ПЛАТФОРМЫ
Разработка приложения производилась с помощью JavaScript и плагинов Apache Cordova и Apia Google

IOS ИЛИ ANDROID?
Choose me
symbian

АРХИТЕКТУРА РЕАЛИЗУЕМОЙ СИСТЕМЫ
Mobile client
Ionic Framework
Angular JS
ngCordova Plugin
Cordova Geolocation Plugin
Cordova BarcodeScanner Plugin
Android App
Backend Service
Amazon EC2
REST API
Services
Data Access Layer
Apache Tomcat
Mongo DB
External web services used by Backend System
Google Places API
Label API

ДИАГРАММА ДИОГРАММА ТРЕБОВАНИЯ К ФУНКЦИОНАЛУ РАЗРАБАТЫВАЕМОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ КЛАССОВ

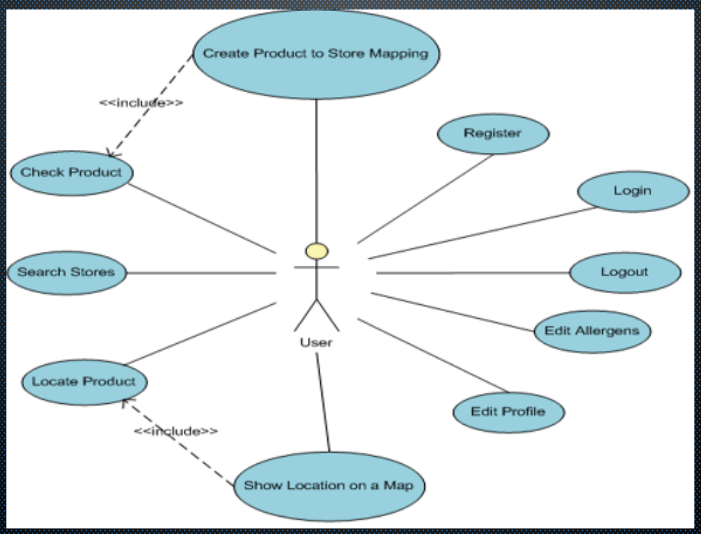
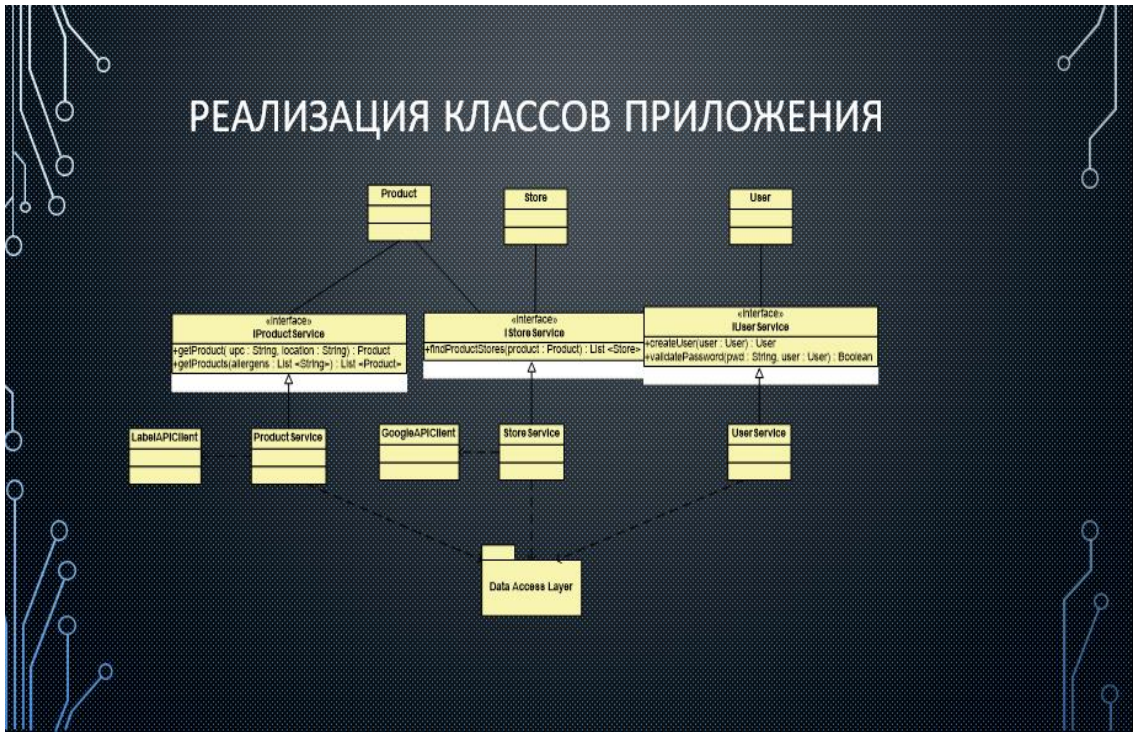
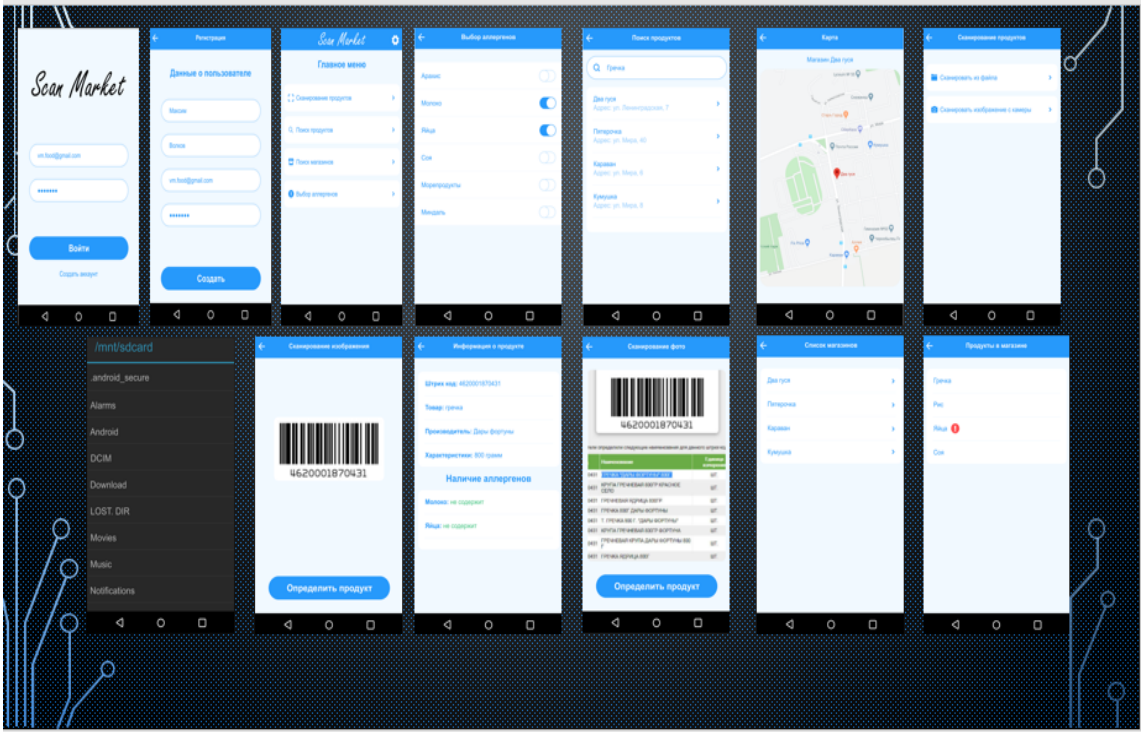


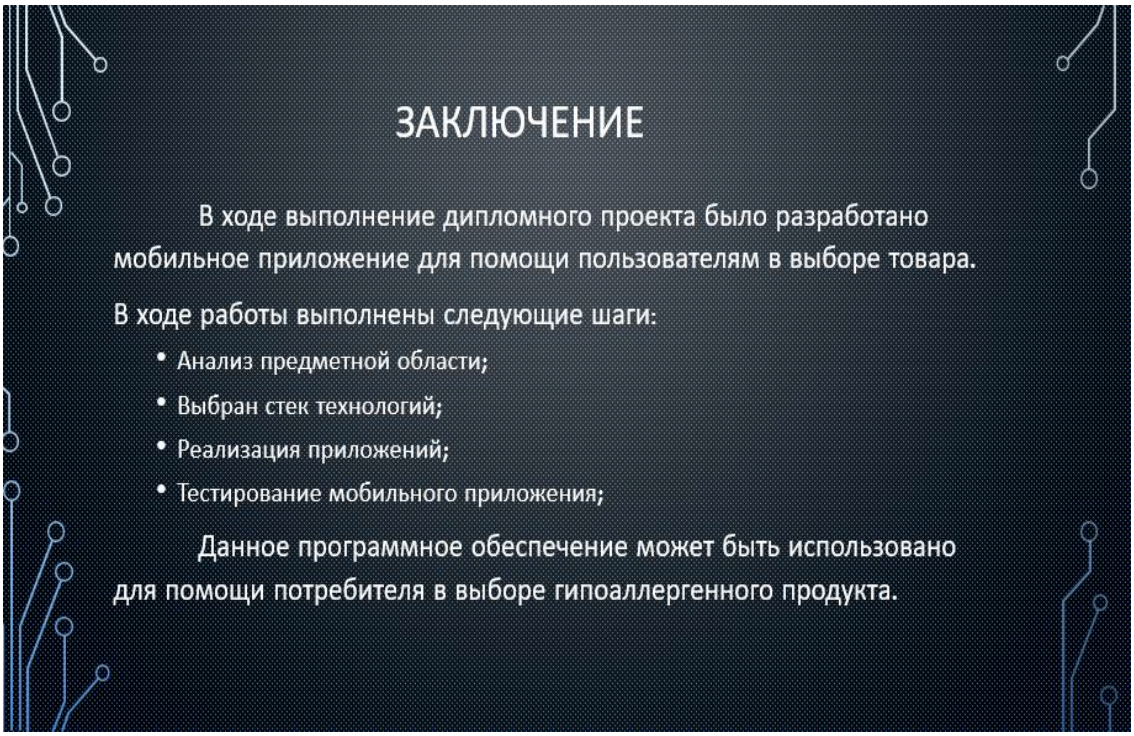
Схема передачи данных между слоями приложения

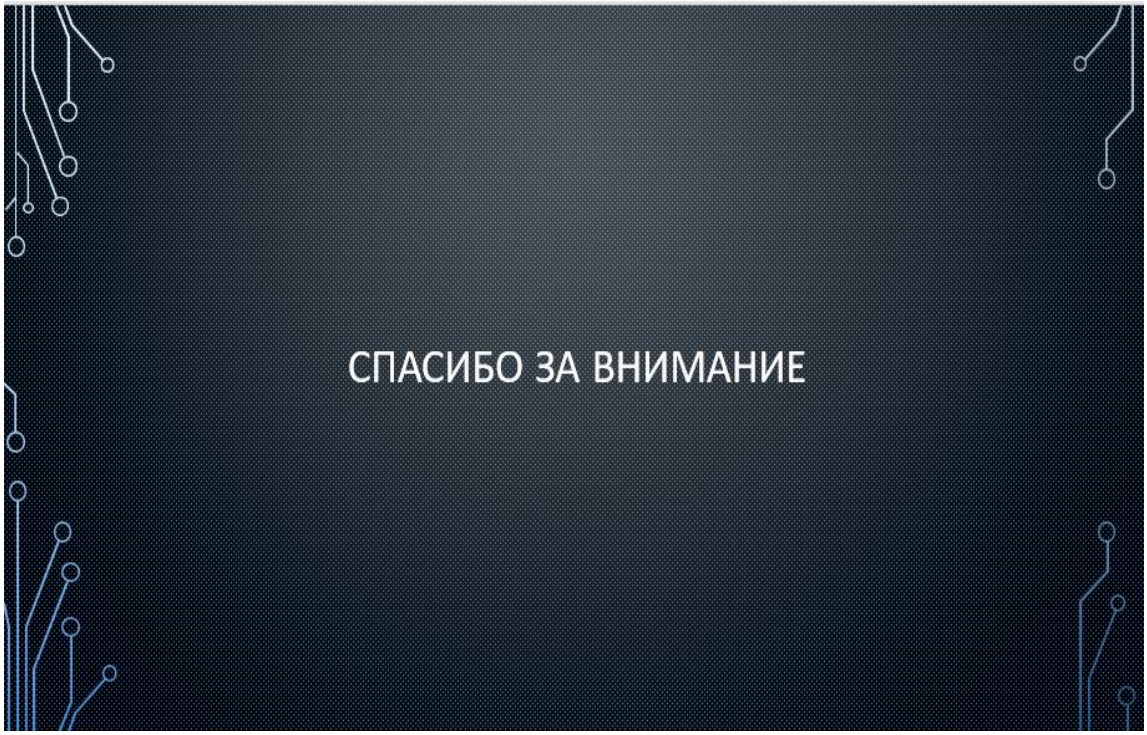
СВЯЗЬ МЕЖДУ КОМПОНЕНТАМИ СИСТЕМЫ
User
Mobile App
Backend Service
Сканирует штрих-код продукта
Читает штрих-код и ищет координаты Местоположения пользователя
Вызывает Backend Service для проверки безопасности продукта
Вызов Verification Service для проверки аллергенов продукта
Вызов API для слушания данных об аллергене продукта
Проверка аллергенов продукта и аллергенов пользователя
Показать результат: является ли продукт безопасным для пользователя
Вернуть результат в формате json
Отправляет сообщение о сборе данных в ActiveMQ
Вызвать API Google Address (найти информацию о магазине)
Создать запись магазина
Создать продукт с записью на карте
создать запись о продукте

ПРОЦЕСС СКАНИРОВАНИЯ ПРОДУКТА


РЕАЛИЗАЦИЯ КЛАССОВ ПРИЛОЖЕНИЯ
Product
Store
User
«interface»
IProductService
+getProduct(upc : String, location : String) : Product
+getProducts(allergens : List «String») : List «Product»
«interface»
IStoreService
+findProductStores(product : Product) : List «Store»
«interface»
IUserService
+createUser(user : User) : User
+validatePassword(pwd : String, user : User) : Boolean
LabelAPIClient
Product Service
GoogleAPIClient
Store Service
User Service
Data Access Layer


Scan Market
Войти
Создать
Главное меню
/mnt/sdcard
Определить продукт
4620001870431


ЗАКЛЮЧЕНИЕ
В ходе выполнение дипломного проекта было разработано мобильное приложение для помощи пользователям в выборе товара.
В ходе работы выполнены следующие шаги:
• Анализ предметной области;
• Выбран стек технологий;
• Реализация приложений;
• Тестирование мобильного приложения;
Данное программное обеспечение может быть использовано для помощи потребителя в выборе гипоаллергенного продукта.


СПАСИБО ЗА ВНИМАНИЕ