

**МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

Факультет компьютерных наук
Кафедра программирования и информационных технологий

Продовольственный онлайн-супермаркет с доставкой на дом
Курсовой проект

09.03.04 Программная инженерия
Информационные системы и сетевые технологии

Зав. кафедрой _____ С.Д. Махортов, д-р физ.-мат. наук,
доцент __.__.20__
Обучающийся _____ Н.В. Морозов, 3 курс, д/о
Обучающийся _____ С.А. Черникова, 3 курс, д/о
Обучающийся _____ Р.Т. Халилов, 3 курс, д/о
Обучающийся _____ Ж. Флорексил, 3 курс, д/о
Руководитель _____ В.С. Тарасов, ст. преподаватель
Руководитель _____ К.В. Зенин, преподаватель

Воронеж 2023

Содержание

Введение	4
1 Постановка задачи	6
1.1 Цели создания приложения	6
1.2 Задачи приложения	6
1.3 Требования к разрабатываемой системе	6
1.3.1 Функциональные требования	6
1.4 Требования к интерфейсу	8
1.5 Задачи, решаемые в процессе разработки	8
2 Анализ предметной области	10
2.1 Терминология (гlossарий) предметной области	10
2.2 Анализ рынка продуктовых онлайн-магазинов	10
2.3 Обзор аналогов	12
2.3.1 СберМегаМаркет	12
2.3.2 Самокат	13
2.3.3 Магнит	14
2.3.4 Пятерочка	15
3 Реализация	17
3.1 Средства реализации	17
3.2 Диаграмма IDEF0	18
3.3 Диаграмма вариантов использования	19
3.3.1 Диаграмма пользователя	19
Рисунок 6 - Диаграмма пользователя	19
3.3.2 Диаграмма сотрудника	20
Рисунок 7 - Диаграмма сотрудника	20
3.4 Диаграмма состояний	20
3.5 Диаграмма классов сущностей	20
3.6 Диаграмма объектов	21
3.7 Диаграмма развертываний	21
3.8 Реализация серверной части приложения	22
3.8.1 Слой доступа к данным	23
3.8.2 Слой контроллеров	24
3.8.3 Слой моделей	25
3.8.4 Слой бизнес-логики	26
3.8.5 Механика работы приложения	27
3.9 Реализация клиентской части приложения	29
3.9.1 Слой доступа к данным	29
Рисунок 12 - Реализация слоя	29
3.9.2 Слой моделей	29
Рисунок 13 - Пример реализации модели	30

3.9.3 Слой Model.....	30
3.9.4 Слой WidgetModel.....	30
3.9.5 Слой Widget.....	30
Заключение	31
Список используемых источников	32

Введение

Последние несколько лет онлайн-ритейл в России стремительно развивается.

Исследование потребительского поведения в ритейл-индустрии проводилось в августе 2021 года в формате панельного онлайн-опроса. Всего было опрошено 300 человек в возрасте от 18 до 55 лет из разных городов России с населением от 500 тысяч жителей.

Обратимся к факторам, обеспечивающим привлекательность приобретения продуктов в онлайн и офлайн. В числе бесспорных преимуществ онлайн-покупок участники опроса отметили безопасность совершения покупок на фоне пандемии. Отсутствие рисков заболеть стало особенно важным для покупателей, поэтому многие сервисы онлайн-покупки продуктов начали практиковать бесконтактную доставку до двери. Так, возможные последствия для здоровья покупателей снижаются практически до нуля (что положительно оценили 38% участников опроса), чем не могут похвастаться физические магазины.

Другая причина выбора e-grocery среди опрошенных людей — онлайн-покупки существенно экономят время покупателей. Это выражается хотя бы в отсутствии необходимости ожидания своей очереди на кассе. Помимо временных затрат, сокращаются и физические усилия, которые нужно приложить во время похода в магазин: не нужно складывать продукты и тащить пакет до дома. Достаточно сделать несколько движений рукой по экрану телефона и встретить курьера.

Так же участники опроса отметили простоту сравнения цен на продукты в разных сервисах доставки: для этого не нужно перемещаться между разными магазинами, а достаточно скачать

несколько приложений, чтобы выбрать самый выгодный товар. В них же можно ознакомиться с актуальными акциями. Удобство такого устройства отметило 73% опрошенных.

Интернет диктует законы среде информационного обмена. В эру цифровизации и тотального распространения смартфонов знакомство с ассортиментом магазина и даже заказ товаров через мобильное приложение становится все более актуальным: об этом свидетельствует 41% анкетированных участников.

Почему люди положительно оценивают влияние мобильных приложений магазинов на свой привычный процесс шопинга? Клиентов привлекает система лояльности многих торговых сетей, доставка и удобство выбора продуктов. Некоторые из приложений даже предугадывают желания клиента и запоминают типичный набор его покупок. К тому же, в приложение с продуманным интерфейсом может быть просто интересно заходить время от времени, обновляя каталог в поисках новинок.

Целью данной работы является разработка мобильного приложения, а именно продовольственного онлайн-магазина под Android с возможностью как самовывоза из ближайших точек, так и доставки курьером. Особенностью данного приложения будет наличие внутренней карты магазина и наличие шагомера, обе доступны при выборе самовывоза. Первая функция поможет покупателю быстро найти в супермаркете товары, выбранные им для покупки. Вторая функция посчитает количество шагов, сделанных пользователем от его местоположения до магазина, и, исходя из этого количества, покупатель получит скидку при оплате продуктов на кассе.

1 Постановка задачи

1.1 Цели создания приложения

Целями создания мобильного приложения «The Shop» являются:

- продажа продуктов питания;
- привлечение новых покупателей.

1.2 Задачи приложения

Разрабатываемый проект должен решать следующие задачи:

- просмотр каталога товаров с возможностью сортировки по цене и по названию товара. В нем представлены продукты, продаваемые заказчиком;
- выбор способа оплаты;
- выбор способа доставки.

1.3 Требования к разрабатываемой системе

1.3.1 Функциональные требования

К разрабатываемому приложению выдвигаются следующие функциональные требования:

зарегистрированный покупатель обладает следующими возможностями:

- определение магазина, из которого будет совершаться самовывоз заказа;
- редактирование персональных данных (например, изменение ближайшего магазина для самовывоза);
- выбор способа получения товаров (самовывоз или доставка курьером);
- фильтрация и сортировка каталога товаров;
- изменение содержимого корзины (добавление/удаление продуктов);
- добавление товаров в избранное;

- создание чек-листа из списка продуктов, находящихся в корзине (необходимо для использования внутренней карты магазина);

- онлайн-оплата покупок;

- авторизация;

- использование шагомера;

- использование внутренней карты магазина (карта расположения стеллажей с продукцией в магазине).

Незарегистрированный покупатель обладает следующими возможностями:

- определение магазина, из которого будет совершаться самовывоз заказа;

- способ получения товаров (самовывоз или доставка курьером);

- фильтрация и сортировка каталога товаров;

- изменение содержимого корзины (добавление/удаление продуктов);

- онлайн-оплата покупки. При выборе самовывоза можно оплатить заказ только онлайн (для оплаты наличными на кассе нужно зарегистрироваться в приложении);

- регистрация;

- использование внутренней карты магазина (карта расположения стеллажей с продукцией в магазине).

Сотрудник магазина (администратор) обладает следующими возможностями:

- редактирование информации об акциях и выгодных предложениях магазина (например, добавить новую акцию);

- получение API ключа;

- обновление feed.

1.4 Требования к интерфейсу

Оформление и верстка экранов приложения должны соответствовать следующим требованиям:

- все экраны приложения должны быть оформлены в едином стиле;
- все экраны приложения должны быть оформлены в соответствии с принципами “Material Design”;
- дизайн приложения должен быть адаптирован для корректного отображения при различных размерах экрана;
- дизайн приложения должен поддерживать портретную ориентацию экрана.

1.5 Задачи, решаемые в процессе разработки

Были поставлены следующие задачи:

- анализ предметной области;
- выбор языка программирования для написания back-enda и front-enda;
- написание технического задания в соответствии с ГОСТ 34.201-20;
- написание курсовой работы по проекту;
- анализ аналогов;
- разработка дизайна приложения;
- создание макетов дизайна приложения;
- разработка основных функций приложения;
- разработка особенностей приложения, которые бы выделяли его на рынке среди аналогов;
- создание диаграмм: прецедентов, классов, активностей, последовательностей, развертывания, сотрудничества, объектов, состояний;
- создание доски заданий и репозитория на GitHub;

- реализация основных функций приложения с использованием REST API;
- реализация интерфейса приложения с использованием Flutter;
- описание процесса разработки и результата.

2 Анализ предметной области

2.1 Терминология (гlossарий) предметной области

Фреймворк - готовый набор инструментов, который помогает разработчику быстро создать продукт: сайт, приложение, интернет-магазин;

Ключ API – уникальный идентификатор, который используется для аутентификации запросов, связанных с вашим проектом.

Front-end - клиентская часть приложения. Отвечает за получение информации с программно-аппаратной части и отображение ее на устройстве пользователя. В нашем проекте, это само android приложение;

Back-end - программно-аппаратная часть приложения. Отвечает за функционирование внутренней части приложения;

Rest API - стиль архитектуры программного обеспечения для построения масштабируемых веб-приложений.

Material Design - дизайн-система для создания интерфейсов программного обеспечения и приложений, разработанная компанией Google.

Верстка экрана - корректное отображение интерфейса приложения на любых устройствах путем задания размеров и разрешения экрана для каждого макета.

Json - текстовый формат обмена данными, основанный на JavaScript.

Feed - список всех товаров магазина, данные о расположении магазинов, карта магазина и её разметка, представленные в виде json.

Неактуальная информация - информация, которая на текущий момент времени уже устарела.

2.2 Анализ рынка продуктовых онлайн-магазинов

Анализ рынка продуктовых онлайн-магазинов становится все более актуальным на фоне растущей конкуренции и

увеличивающегося количества игроков в этом сегменте. Продуктовые онлайн-магазины имеют некоторые преимущества перед традиционными магазинами, но также сталкиваются с ограничениями и некоторыми минусами.

Одним из главных преимуществ продуктовых онлайн-магазинов является возможность заказа и доставки продуктов питания без необходимости выхода из дома. Это особенно актуально в условиях пандемии, когда многие люди избегают посещения магазинов и предпочитают оставаться дома. Кроме того, такие магазины обычно предлагают более широкий ассортимент товаров, чем традиционные магазины, и часто предлагают товары из региональных производств.

Однако у продуктовых онлайн-магазинов есть и ряд минусов. Одним из них является возможность получения продуктов низкого качества или с истекшим сроком годности. Это может произойти из-за проблем с хранением и транспортировкой продуктов. Кроме того, у некоторых продуктовых онлайн-магазинов могут быть ограничения по срокам доставки в том числе из-за отдаленности региона.

Продуктовые онлайн-магазины также сталкиваются с ограничениями в связи с тем, что рынок достаточно конкурентный. Они вынуждены проявлять творческий подход и усердно работать над предоставлением уникальных услуг своим клиентам. Конкурентной способностью для таких магазинов становится не только сумма цен, но и качество предоставления услуг, удобство работы с сайтом и организация работы курьеров.

Таким образом, продуктовые онлайн-магазины имеют свои преимущества и ограничения, но благодаря необходимости усложнения работы для организации продаж на рынке, такие

магазины могут сохранять свою конкурентную способность. В будущем, с улучшением технологий и развитием сетей доставки, продуктовые онлайн-магазины, возможно, станут еще более популярными, для тех кто выбирает домашний комфорт и минимум рисков, связанных с пандемией.

2.3 Обзор аналогов

Этап обзора аналогов является важной частью процесса разработки мобильного приложения. Этот шаг включает в себя изучение приложений, аналогичных разрабатываемому, чтобы лучше понять ожидания пользователей, отраслевые тенденции и передовые методы разработки мобильных приложений. Этот обзор позволит собрать информацию и идеи, полезные для разработки высококачественного приложения, которое будет соответствовать потребностям целевой аудитории и будет выделяться на рынке.

2.3.1 СберМегаМаркет

Это онлайн-платформа для супермаркетов, запущенная крупнейшим банком России Сбербанком. СберМегаМаркет предлагает широкий ассортимент продуктов питания, товаров для дома, косметических товаров и товаров для домашних животных. Пользователи могут выбрать удобное время доставки, а также отслеживать их доставку в режиме реального времени.

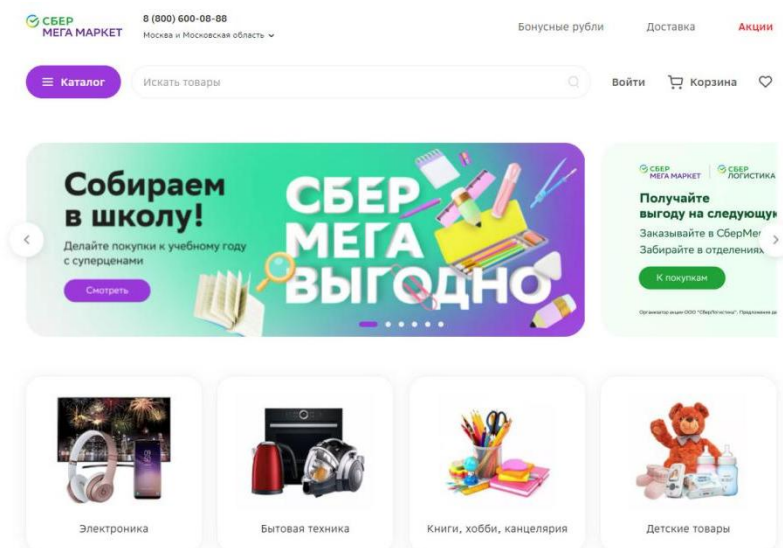


Рисунок 1 - интерфейс СберМегаМаркета

Плюсы:

- Автоматический импорт корзины из предыдущих заказов;
- Возможность отслеживания статуса заказа;
- Различные способы оплаты;
- Возможность заказа через голосовой ассистент;
- Можно использовать бонусы «Спасибо» и вход по «Сбер ID»;
- Бесплатная доставка для подписчиков «СберПрайм».

Минусы:

- Бывают проблемы с доставкой: пользователи или получают заказ частично, или вовсе не получают его;
- Иногда приложение тормозит.

2.3.2 Самокат

Это российская компания по онлайн-доставке продуктов, предлагающая продукты питания, аптечные товары, Товары для дома и туалетные принадлежности. Компания самокат гарантирует доставку за 15 минут и менее, что делает ее одним из самых быстрых вариантов доставки в России.

Быстрая доставка продуктов

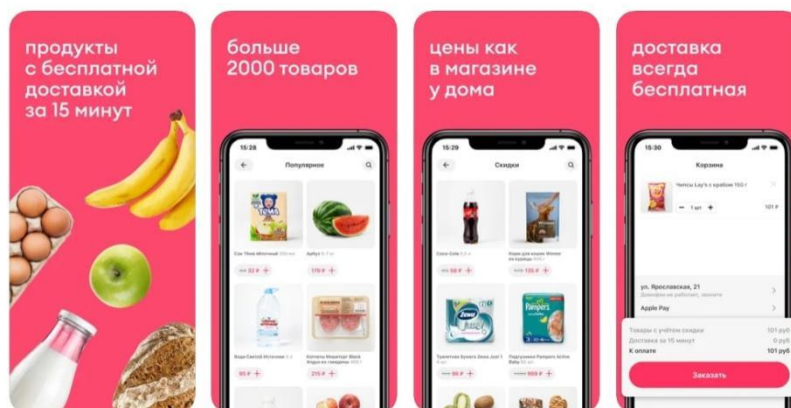


Рисунок 2 - интерфейс Самоката

Плюсы:

- Пользователь может выбрать время доставки;
- Возможность отслеживания статуса заказа в приложении.

Минусы:

- Высокая стоимость доставки;
- Иногда доставляют просроченный товар.

2.3.3 Магнит

Магнит - это сеть супермаркетов в России, предлагающая широкий ассортимент продуктов питания, а также товары для дома и косметики. Помимо своих обычных магазинов, Магнит также предлагает онлайн-платформу супермаркетов с доставкой на дом.

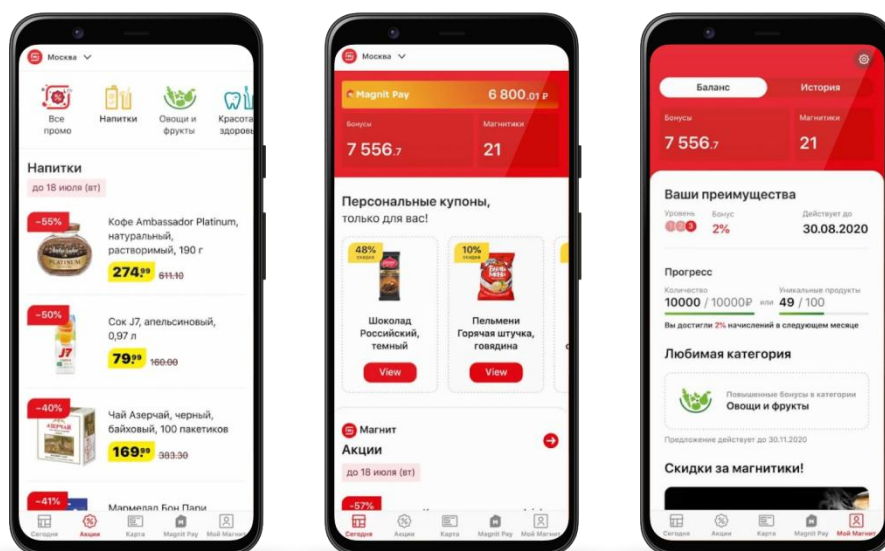


Рисунок 3 - интерфейс Магнита

Плюсы:

- Различные способы оплаты и доставки;
- Автоматический импорт корзины из предыдущих заказов.

Минусы:

- Приложение работает медленно и часто вылетает;
- Неактуальная информация о скидках и бонусах.

2.3.4 Пятёрочка

Это еще одна сеть супермаркетов в России, предлагающая широкий ассортимент продуктов питания, а также товары для дома и косметики. Пятёрочка, онлайн приложение которой представлено на рисунке 4) также предлагает онлайн-платформу супермаркетов с доставкой на дом.

Плюсы:

- Различные способы оплаты;
- Возможность отслеживания статуса заказа.

Минусы:

- Доставка продуктов занимает больше двух часов;
- Приложение часто зависает.

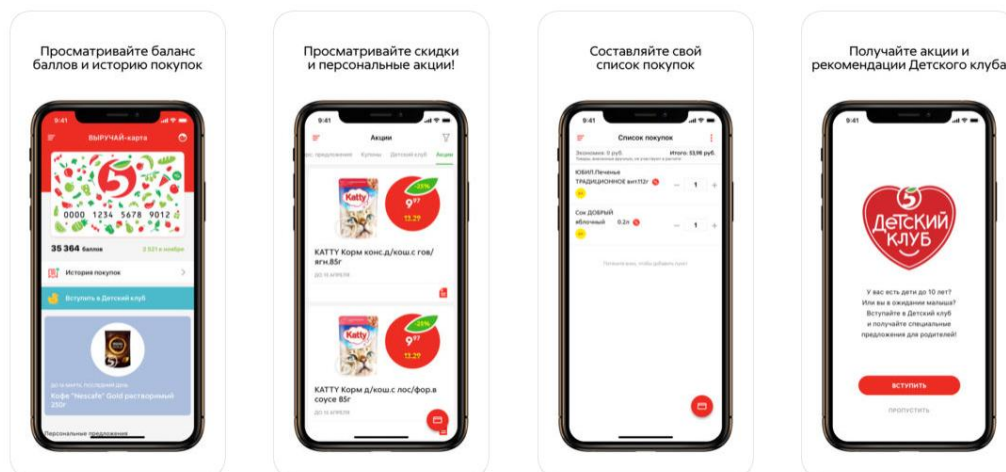


Рисунок 4 - интерфейс Пятерочки

3 Реализация

3.1 Средства реализации

Система должна состоять из сервера приложения, реляционной базы данных, клиентской части.

Основной используемый стек технологий:

Back-end (серверная часть):

- Java 17;
- Spring Framework;
- PostgreSQL, Liquibase;
- Elasticsearch;
- Система сборки Maven.

Язык Java был выбран, так как он не зависит от платформы. Можно создать Java-приложение на Windows, скомпилировать его в байт-код и запустить на любой другой платформе, поддерживающей JVM – виртуальную машину Java.

Основным преимуществом Spring Framework является большое количество реализованных внутренних библиотек, позволяющих быстро и качественно писать код.

В качестве базы данных была выбрана Postgres, т.к. поддерживает пользовательские объекты и их поведение, включая типы данных, функции, операции и другое. Это делает Postgres невероятно гибким и надежным. Среди прочего, он умеет создавать, хранить и извлекать сложные структуры данных.

Для управления базой данных и внесения изменений в нее будет использоваться библиотека Liquibase. Основным преимуществом является поддержка написания миграционных файлов в виде xml файлов.

Elasticsearch - это масштабируемый полнотекстовый поисковый движок с открытым исходным кодом, использующий

библиотеку Lucene и написанный на Java. Он предназначен для сложных поисков по базе документов/файлов. Его преимуществом является скорость поиска и быстрая выдача результатов.

Front-end (клиентская часть):

- Flutter;
- Firebase (Crashlytics, Analytics, Remote Config, A/B Testing, Authentication, Realtime Database);
- Google Maps;
- Elementary (фреймворк).

Главное преимущество Flutter - скорость разработки. Готовые решения данного фреймворка позволяют писать меньше кода, что значительно упрощает процесс создания приложений и существенно экономит время разработчиков.

Firebase разрабатывается Google. Предоставляет мощные базы данных для разработки веб- и мобильных приложений.

Elementary соответствует ряду требований: является комплексным решением и обеспечивает чётко разделённые по ответственностям слои. Они, в свою очередь, максимально изолированы и независимы друг от друга. Решение легко тестируемо. Реализация максимально близка к работе Flutter.

3.2 Диаграмма IDEF0

На рисунке 5 представлена IDEF0 диаграмма, иллюстрирующая процесс оформления и получения заказа.

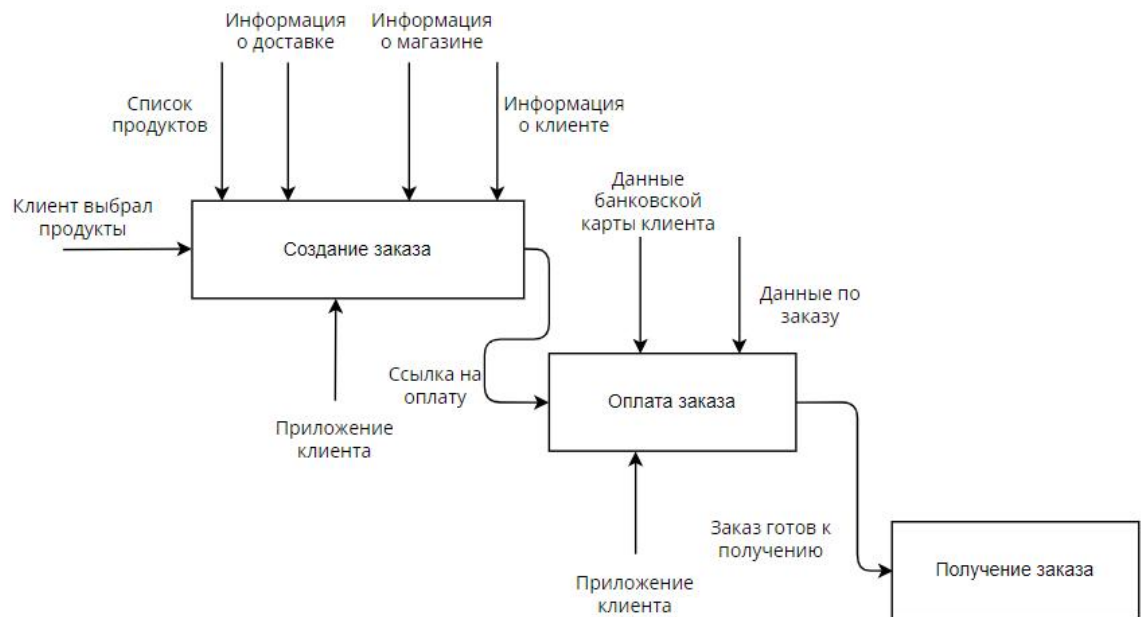


Рисунок 5 - Диаграмма IDEF0

3.3 Диаграмма вариантов использования

3.3.1 Диаграмма пользователя

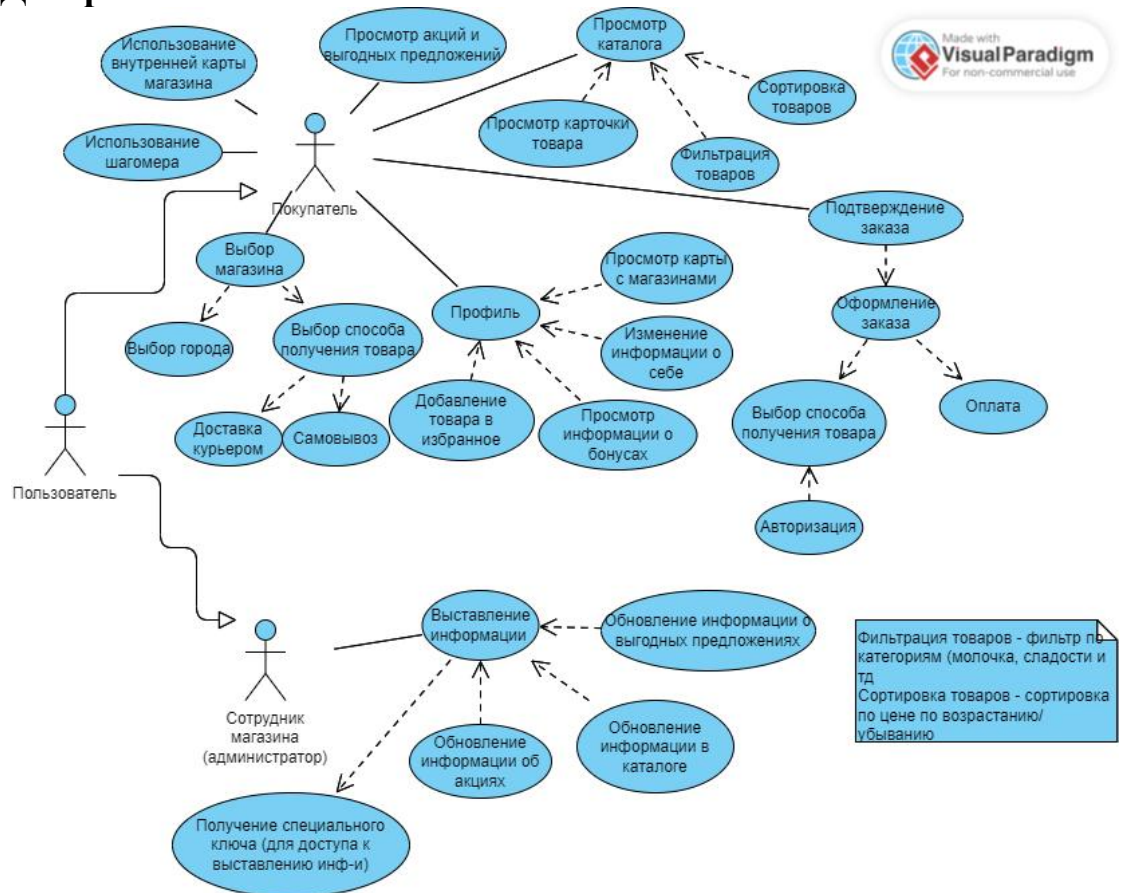


Рисунок 6 - Диаграмма пользователя

3.3.2 Диаграмма сотрудника

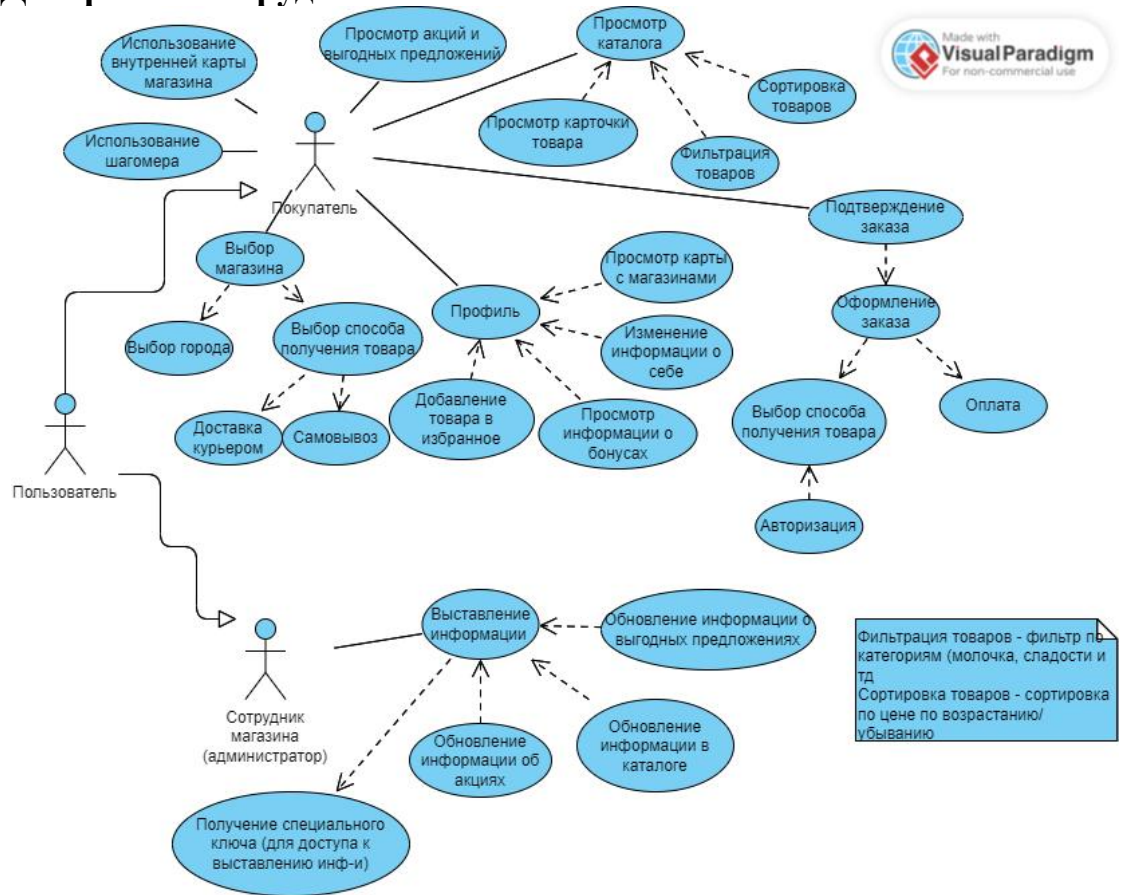


Рисунок 7 - Диаграмма сотрудника

3.4 Диаграмма состояний

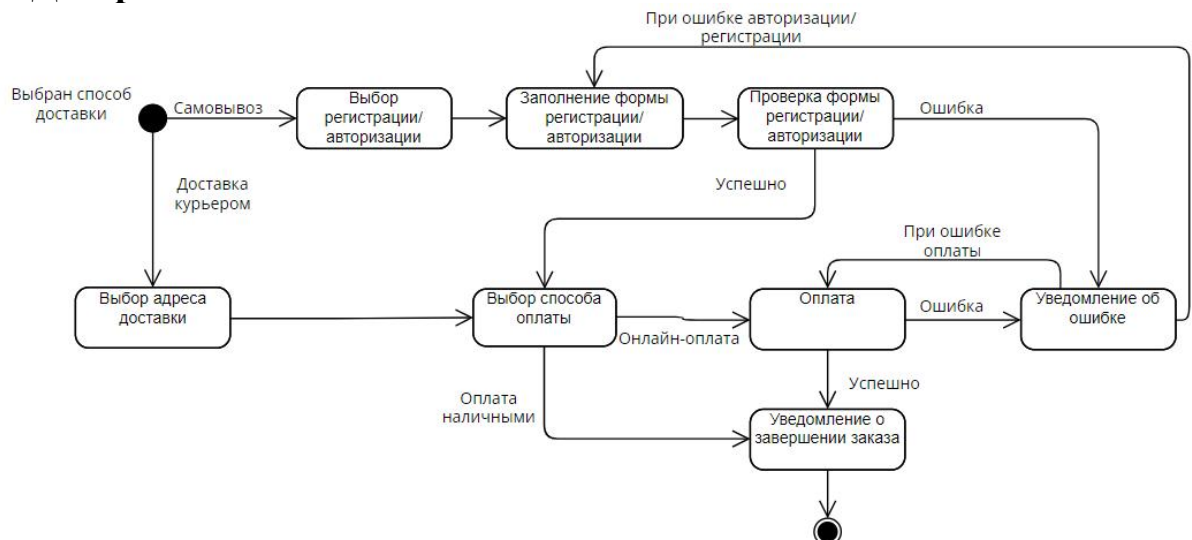


Рисунок 8 - Диаграмма состояний

3.5 Диаграмма классов сущностей

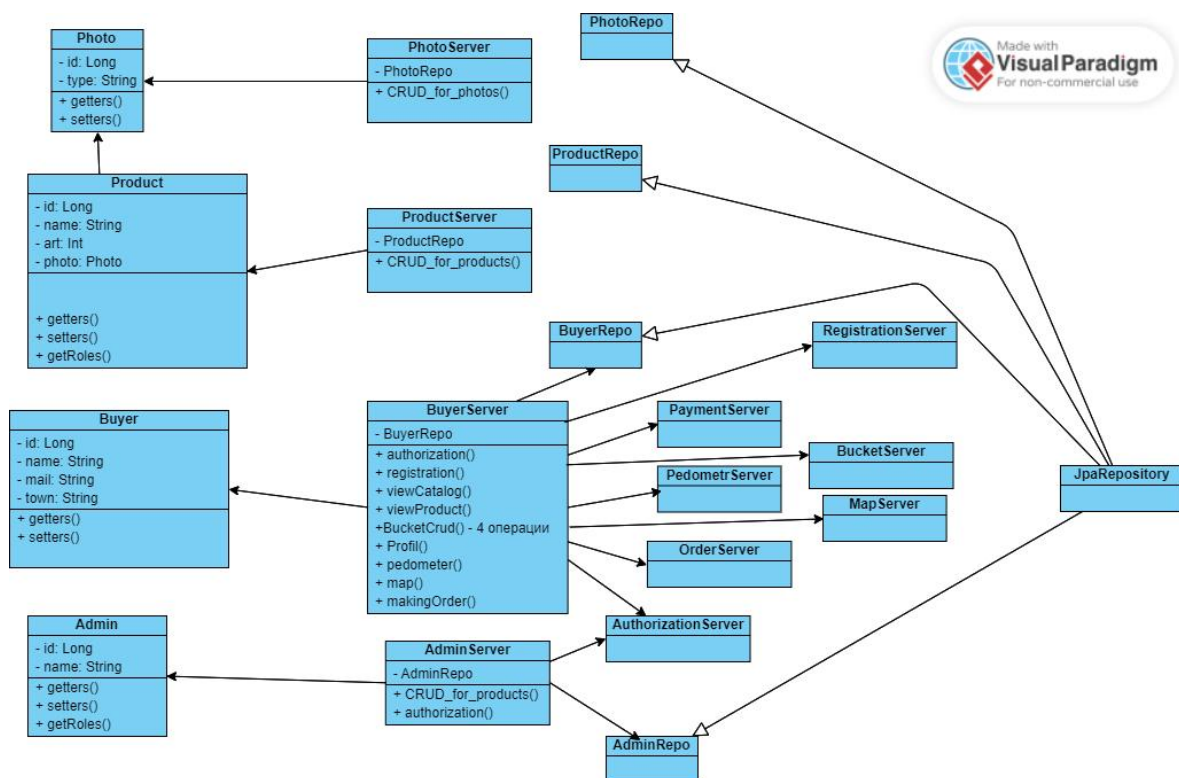


Рисунок 9 - Диаграмма классов сущностей

3.6 Диаграмма объектов

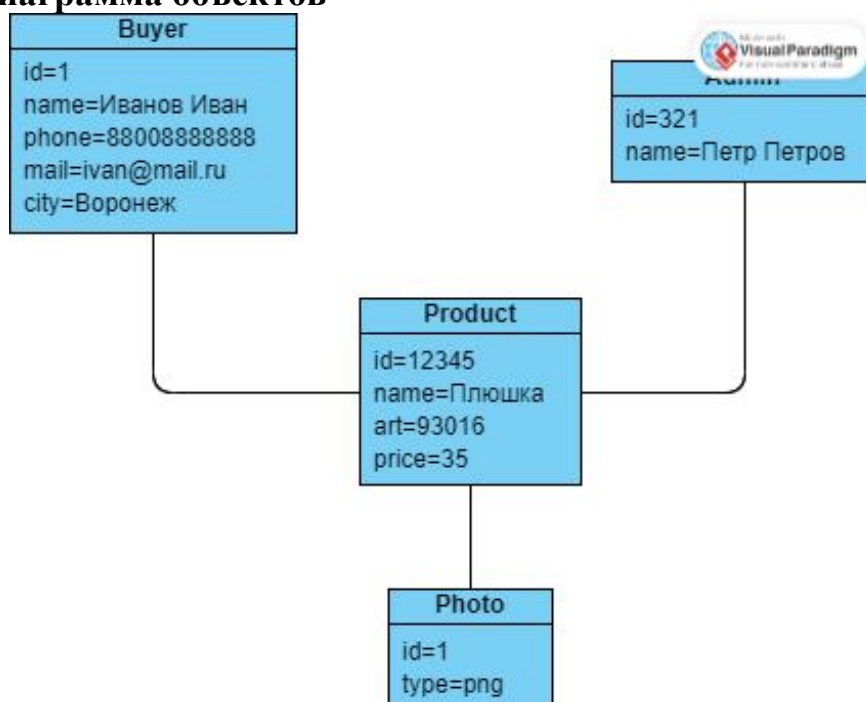


Рисунок 10 - Диаграмма объектов

3.7 Диаграмма развертываний

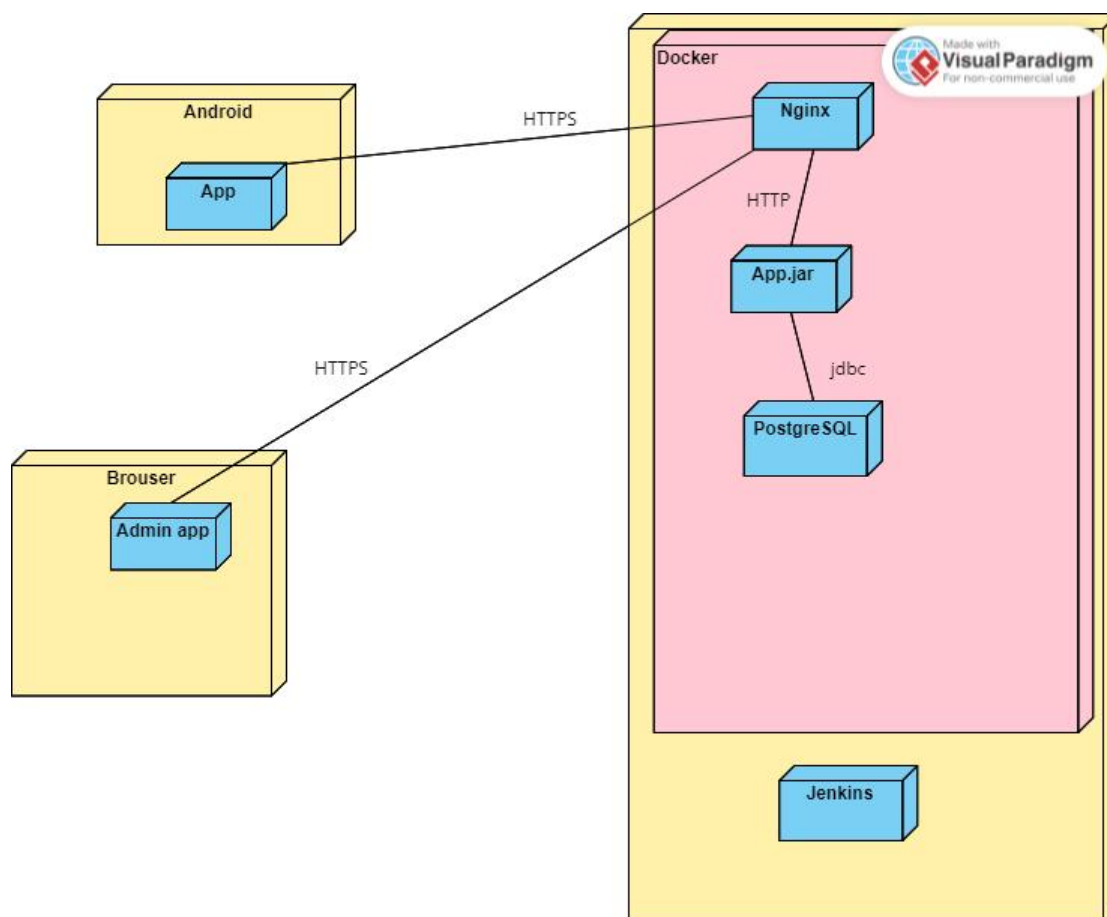


Рисунок 11 - Диаграмма развертываний

3.8 Реализация серверной части приложения

Для серверной части используется Spring Boot фреймворк для разработки веб-приложений, который облегчает создание приложений на основе архитектуры MVC (Model-View-Controller).

MVC (Model-View-Controller) - это шаблон проектирования, используемый для разработки приложений, которые разделяют данные приложения, взаимодействие пользователя и управление взаимодействием между ними. Он разделяет приложение на три абстрактные составляющие: Model (Модель), View (Представление) и Controller (Контроллер).

Достоинства паттерна MVC следующие:

— Разделение ответственности: Модель содержит данные и логику бизнес-процессов. Представление отображает данные, а Контроллер обрабатывает входные данные и управляет

взаимодействием между Моделью и Представлением. Обычно каждый компонент отвечает за свою функциональность, что делает код более читабельным и понятным;

— Увеличение возможности переиспользования кода: Благодаря разделению различных компонентов приложения, каждый из них может быть переиспользован или заменен без влияния на другие. Обновление компонентов может происходить независимо, что сокращает время разработки и риски ошибок.

— Улучшение тестируемости: Возможность тестирования каждого компонента независимо облегчает процесс тестирования и повышает качество приложения в целом;

— Повышение производительности: Благодаря эффективному управлению потоками данных, производительность пользователя улучшается.

В целом, паттерн MVC позволяет разрабатывать более стабильные, гибкие и отлично структурированные приложения. Он также помогает упрощать процесс разработки, повышать качество приложения и улучшает его производительность.

3.8.1 Слой доступа к данным

Слой доступа к данным (Data Access Layer, DAL) - это слой абстракции, ответственный за управление доступом к данным в приложении. Он позволяет разработчикам концентрироваться на бизнес-логике, а не на том, как данные хранятся и как к ним получать доступ. С помощью DAL можно развязать зависимости между кодом, работающим с данными, и представлением, что делает код более общим и переиспользуемым.

JPA Repository - это интерфейс, который предоставляет различные методы CRUD (Create, Read, Update, Delete) для работы с базой данных. Он представляет уровень доступа к данным, который

упрощает выполнение типовых операций с базой данных без необходимости написания множества SQL-запросов.

Использование JPA Repository позволяет упростить слой доступа к данным в приложении и сократить время разработки. Он автоматически генерирует SQL-запросы на основе методов, которые определены в репозитории, что уменьшает количество кода, который необходимо написать. Вместо этого можно сосредоточиться на бизнес-логике и взаимодействии с пользователем в представлении.

3.8.2 Слой контроллеров

Слой контроллеров (Controller) - это один из трех компонентов архитектуры MVC в Spring Boot Framework. Он отвечает за обработку запросов от клиента и выдачу ответа. Контроллер получает запросы от пользовательского интерфейса (View) или через API, взаимодействует с бизнес-логикой (Model) и формирует ответ клиенту.

Задачи слоя контроллеров в Spring Boot включают в себя:

- Обработка входящих запросов: Контроллеры отлавливают запросы от клиентов и передают их в соответствующий метод обработки;

- Извлечение данных запроса: Контроллеры извлекают данные, переданные в запросе, обрабатывают их и передают внутрь бизнес-логики приложения для дальнейшей обработки;

- Взаимодействие с бизнес-логикой: Контроллеры вызывают нужные методы бизнес-логики, чтобы обработать полученные запросы и передать данные обратно в слой представления для отображения;

- Формирование и отправка ответа: Контроллеры формируют ответ на запрос, который может быть представлен как

HTML-страница, JSON-объект или любой другой тип данных, и отправляют его обратно клиенту.

Слой контроллеров позволяет разработчикам создавать высокопроизводительные и масштабируемые веб-приложения на основе архитектуры MVC. В целом, слой контроллеров является важным компонентом приложения, который помогает обеспечить гибкость, масштабируемость и управляемость проекта.

3.8.3 Слой моделей

Слой моделей (Model) - это один из трех компонентов архитектуры MVC в Spring Boot Framework. Он представляет бизнес-логику приложения и обычно содержит классы, которые представляют различные сущности в приложении.

Задачи слоя моделей в Spring Boot включают в себя:

- Определение структуры данных: Модели предоставляют структуру данных для приложения, указывая, какие поля будут содержать каждый объект;

- Валидация данных: Модели обеспечивают проверку данных перед сохранением или отправкой пользователю, чтобы гарантировать, что они соответствуют требованиям для данной сущности.;

- Взаимодействие с базой данных: Модели содержат бизнес-логику и обеспечивают взаимодействие с базой данных через слой доступа к данным (DAL);

- Создание бизнес-логики: Модели обеспечивают реализацию бизнес-логики приложения в отношении конкретных сущностей.

В слое моделей определяется структура и логика работа с бизнес-объектами, которые используются приложением. Это место, где пишется код для интерактивности и функциональности

приложения, где описываются бизнес-правила и проверяется корректность данных перед сохранением их в базу данных. Правильно построенный слой моделей обеспечит целостность и надежность всей структуры приложения.

3.8.4 Слой бизнес-логики

Слой сервисов в бизнес-логике - это промежуточный слой, который находится между слоем контроллеров и слоем моделей в MVC-архитектуре приложения. Слои контроллеров и моделей занимаются обработкой запросов и предоставлением данных, а слой сервисов обеспечивает бизнес-логику, которая определяет, как эти данные будут обработаны.

Задачи слоя сервисов в бизнес-логике включают в себя:

- Определение бизнес-правил: Слой сервисов задает бизнес-правила, определяя, что именно должно происходить при выполнении различных операций;

- Обработка данных: Слой сервисов получает данные из слоя моделей, обрабатывает их и возвращает результат в слой контроллеров;

- Управление транзакциями: Слой сервисов отвечает за управление транзакциями между слоем моделей и слоем контроллеров, обеспечивая целостность данных в пределах приложения;

- Управление бизнес-логикой приложения: Слой сервисов занимается управлением всей бизнес-логикой приложения, обеспечивая ее правильную работу.

Слой сервисов в бизнес-логике необходим для того, чтобы разделить логику приложения и бизнес-правила от контроллеров и моделей. Он позволяет управлять всем приложением, управлять бизнес-логикой и создавать взаимодействие между различными

слоями приложения. Кроме того, слой сервисов также позволяет более легко расширять и изменять приложение: изменения могут быть внесены без необходимости изменения других компонентов приложения.

3.8.5 Механика работы приложения

Для работы приложения были реализованы следующие сервисы:

- UserService;
- UpdateFeedService;
- ShopService;
- ProductService;
- PaymentService;
- OrderService;
- JwtService;
- GeoService;
- DeliveryService;
- CategoryService;
- BannerService.

UserService:

Имеет методы для формирования корзины пользователя по его id: добавлению и удалению товаров из нее, для формирования списка избранного пользователя по его id с теми же функциями, что и для корзины, получение списка заказов пользователя по его id. Также предоставляет возможность удаления профиля и регистрации.

UpdateFeedService:

Имеет метод, который позволяет сотруднику магазина обновить ассортимент товаров в онлайн магазине, отправив файл в формате json.

ShopService:

Имеет методы для добавления нового магазина и демонстрации всех магазинов, откуда можно заказать доставку или забрать заказ самовывозом.

ProductService:

Имеет методы для демонстрации карточки товара по его id, а так же методы для страничного вывода продуктов с возможностью сортировки по имени и по цене в порядке возрастания или убывания, а так же страничного вывода продуктов, относящихся к одной категории с той же сортировкой.

OrderService:

Имеет методы для создания нового заказа и демонстрации информации о конкретном заказе по его id.

PaymentService:

Здесь находится метод по оплате заказа.

JwtService:

Имеет методы для работы с security и firebase.

GeoService:

Здесь реализованы методы по выводу списка городов России, в которых может проживать пользователь, а также метод для добавления нового города.

DeliveryService:

Имеет метод для создания новой доставки, а также дополнительные функции для расчета стоимости доставки в зависимости от расположения магазина.

CategoryService:

Здесь реализованы методы для демонстрации всех категорий товаров, их добавления и удаления.

BannerService:

Имеет методы для демонстрации баннеров: списки акций, выгодные предложения.

3.9 Реализация клиентской части приложения

3.9.1 Слой доступа к данным

Слой доступа к данным отвечает за взаимодействие с сервером. Этот слой делает все запросы на сервер, будь то: добавление или удаление из избранного, добавление или удаление из корзины, получение информации пользователя и т.п.

```
@RestApi()
abstract class OrderClient{
    factory OrderClient(Dio dio, {String baseUrl}) = _OrderClient;

    @GET(OrderUrls.baseOrder)
    Future<Order> getOrder(@Path() id);

    @POST(OrderUrls.createOrder)
    Future<Order> createOrder(@Body() Order order);
}
```

Рисунок 12 - Реализация дата слоя

Классы данного слоя имеют постфикс «Client». Каждая модель имеет свой клиент, через которую делаются все запросы, описанные в сваггере.

3.9.2 Слой моделей

Слой моделей отвечает за хранение данных. Модели созданы для каждого возможно json-файла, который приходит с сервера, или отправляется на сервер.

```

@Freezed(
  toJson: true,
  fromJson: true,
)
class ShortProduct with _$ShortProduct {
  factory ShortProduct({
    required int id,
    required String productName,
    required Decimal productPrice,
    Decimal? productOldPrice,
    required int basketQuantity,
    required List<String> pictures,
  }) = _ShortProduct;

  factory ShortProduct.fromJson(Map<String, dynamic> json) =>
    _ShortProductFromJson(json);
}

```

Рисунок 13 - Пример реализации модели

3.9.3 Слой Model

Каждый экран состоит из 3 составных частей - Model, WidgetModel, Widget и строится по архитектура MVVM. Model - это слой, который обращается к дата слою.

3.9.4 Слой WidgetModel

WidgetModel - слой бизнес-логики. ViewModel слой в архитектуре MVVM. Данный слой имеет доступ к слою Model и к слою Widget. Являясь прослойкой между 2 слоями, он соединяет данные с их отображением

3.9.5 Слой Widget

Слой отображения, который видит пользователь приложения. View слой в архитектуре MVVM. Та часть, которая не имеет в себе никакой бизнес логики. Существует для вывода данных и передачи команд пользователя в бизнес-логику.

Заключение

В ходе выполнения данного курсового проекта были выполнены все поставленные задачи.

Все поставленные цели были выполнены:

- Продажа продуктов питания;
- Привлечение новых покупателей.

Список используемых источников

1. Документация Flutter [Электронный ресурс]. – Режим доступа: URL:<https://docs.flutter.dev/>– Заглавие с экрана. – (Дата обращения 04.05.2023).

2. Документация SpringBoot [Электронный ресурс]. – Режим доступа: URL: <https://docs.spring.io/spring-boot/docs/current/reference/html/> – Заглавие с экрана. – (Дата обращения 22.04.2023).

3. Документация Swager [Электронный ресурс]. – Режим доступа: URL: <https://swagger.io/docs/> – (Дата обращения 10.05.2023).

4. Документация Elementary [Электронный ресурс]. – Режим доступа: URL: <https://pub.dev/documentation/elementary/latest/> – (Дата обращения 10.05.2023).