МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра Прикладная математика и информатика
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки / специальности)

Разработка программного обеспечения
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка мобильного приложения для онлайн-магазина»

Обучающийся К.Ю. Сухарков _____
(Инициалы Фамилия) (личная подпись)

Руководитель Кандидат технических наук, Д.Г. Токарев
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

Аннотация

Бакалаврская работа выполнена на тему «Разработка мобильного приложения для онлайн-магазина».

Цель данной работы – разработать мобильного приложения для онлайн-магазина.

Во введении раскрываются актуальность исследования, объект и предмет работы, а также ее цель и поставленные задачи.

В первой главе рассмотрены вопросы, которые изучают предмет исследования, а именно описание деятельности онлайн магазина компании ООО "Прима-Сервис" и приводится описание основных бизнес-процессов онлайн магазина.

Во второй главе рассмотрены вопросы проектирования и разработки мобильного приложения.

В третьей главе рассчитана экономическая эффективность проекта и описан тестовый пример реализации мобильного приложения онлайн магазина компании ООО "Прима-Сервис".

В заключении содержатся выводы о работе в целом и по главам.

Бакалаврская работа состоит из 63 страниц и включает 32 рисунка, 9 таблиц, 20 источников литературы и 1 приложение.

ABSTRACT

The title of the graduation work is "Development of a Mobile Application for an Online Store".

The graduation work consists of an introduction, three parts, 32 figures, 9 tables, a conclusion, and a list of 20 references including foreign sources.

The aim of this graduation work is to create a mobile application for an online store.

The object of the graduation work is the company «Prima-Service» LLC.

The subject of the graduation work is the process of purchasing canned products of the company through a mobile application.

The key issue of the graduation work is to develop a mobile application for an online store, including its design, implementation and description of the main business processes of the online store.

The graduation work may be divided into 3 parts: domain analysis, mobile application design, and mobile application development.

The first part of the graduation work is devoted to the analysis of the subject area. The description of the activities of the company OOO Prima-Service and the main business processes are described.

The second part outlines the results designing a mobile application, which includes logical modeling and database design.

The third part consists of the implementation and testing methodology of the developed mobile application.

In conclusion, the developed mobile application for the online store successfully addresses the set objectives. Nevertheless, further optimization and implementation of new features are required.

The work is of interest for a wide range of people interested in the development of mobile applications for e-commerce.

Содержание

Введение 5
1 Анализ предметной области 6
1.1 Характеристика объекта автоматизации 6
1.2 Описание бизнес-процессов компании 8
1.3 Требования к разработке 12
1.4 Анализ возможных проектных решений 13
2 Проектирование мобильного приложения 16
2.1 Логическое моделирование информационной системы 16
2.2 Логическое моделирование 18
2.3 Проектирование базы данных 21
3 Разработка мобильного приложения 30
3.1 Разработка пользовательского интерфейса 30
3.2 Тестирование мобильного приложения 41
Заключение 44
Список используемой литературы и используемых источников 45
Приложение А 47

Введение

Тема выпускной квалификационной работы «Разработка мобильного приложения для онлайн-магазина».

Актуальность работы заключается в необходимости разработки мобильного приложения для организации ООО "Прима-Сервис".

Объект исследования – компания ООО "Прима-Сервис".

Предмет исследования - процесс покупки консервированной продукции компании ООО "Прима-Сервис" через мобильное приложение.

Цель – разработать мобильное приложение для он-лайн магазина мясных консервов.

Для достижения цели необходимо решить следующие задачи:

- изучить предметную область деятельности организации ООО "Прима-Сервис";
- описать бизнес-процессы, которые описывают логику работы он-лайн магазина;
- разработать прототип мобильного приложения;
- провести тестирование разработанного прототипа приложения.

Структурно работа состоит из введения, 3 глав, заключения, списка использованной литературы и приложения.

Практическая значимость состоит в применение полученных теоретических знаний к проектированию и разработки мобильного приложения для организации.

Результаты работы могут быть успешно внедрены в любую компания, где есть актуальны вопросы использования, внедрения и разработки мобильных приложений.

1 Анализ предметной области

1.1 Характеристика объекта автоматизации

Предприятие на базе, которого пройдена практика – общество с ограниченной ответственностью "Прима-Сервис". ООО "Прима-Сервис" (ИНН 2901292338) работает на рынке с 2018 года.

Основным видом экономической деятельности компании является 47.11 - Торговля розничная преимущественно пищевыми продуктами, включая напитки, и табачными изделиями в неспециализированных магазинах. Также ООО "Прима-Сервис" дополнительно работает еще по нескольким направлениям. Дополнительные виды деятельности связаны напрямую с продажей товаров:

- «10.11 Переработка и консервирование мяса;
- 10.12 Производство и консервирование мяса птицы;
- 10.13 Производство продукции из мяса убойных животных и мяса птицы;
- 10.20 Переработка и консервирование рыбы, ракообразных и моллюсков;
- 10.31 Переработка и консервирование картофеля;
- 10.32 Производство соковой продукции из фруктов и овощей;
- 10.39 Прочие виды переработки и консервирования фруктов и овощей;
- 10.51 Производство молока (кроме сырого) и молочной продукции;
- 10.71 Производство хлеба и мучных кондитерских изделий, тортов и пирожных недлительного хранения;
- 10.72 Производство сухарей, печенья и прочих сухарных хлебобулочных изделий, производство мучных кондитерских изделий, тортов, пирожных, пирогов и бисквитов, предназначенных для длительного хранения;
- 10.85 Производство готовых пищевых продуктов и блюд;

– 10.89 Производство прочих пищевых продуктов, не включенных в другие группировки» [4].

Организационная структура – это система взаимосвязей между различными уровнями управления и функциональными областями [8]. Организационная структура компании представлена на рисунке 1. Численность персонала компании – 18 человек.

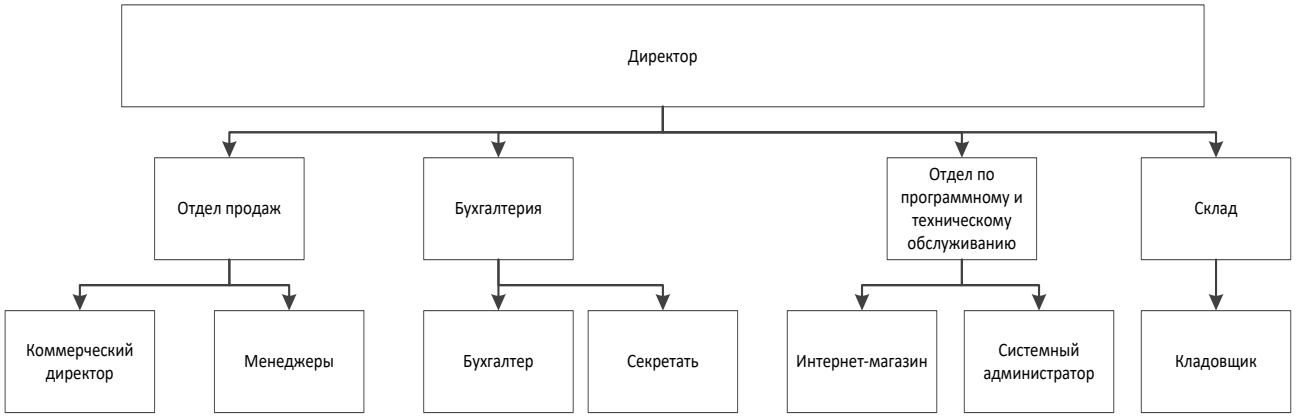


Рисунок 1 - Организационная структура ООО "Прима-Сервис"

ООО "Прима-Сервис" небольшое предприятие, которое в последнее время стало активно развивать продажу мясных консервов. Мясные консервы – это готовый к употреблению продукт, полученный из мяса, субпродуктов, жира, пряностей и специй.

Мясные консервы имеют длительный срок хранения. Это обусловленно технологией консервирования.

Продажа мясных консервов на территории Российской федерации регламентируется «Техническим регламентом Таможенного союза "О безопасности мяса и мясной продукции"(ТР ТС 034/2013)»

Основные позиции для онлайн продаж магазина консервов компании "Прима-Сервис" представленные на сегодняшний день состоят из 6 позиций:

– Тушеная говядина высший сорт 338г,
– Филе индейки тушеное Йошкар-Олинская тушенка 325г,

– Говядина тушеная 325г,
– Говядина тушеная Люкс 320г,
– Оленина тушеная 338г,
– Ветчина 338г.

Анализ деятельности компании, позволяет сделать следующий вывод, что его развитие с помощью автоматизации бизнес-процессов можно считать обоснованным.

1.2 Описание бизнес-процессов компании

Рассмотрим контекстную диаграмму процесса, который необходимо автоматизировать в компании согласно темы ВКР «Разработка мобильного приложения для онлайн-магазина», диаграмма нулевого уровня показана на рисунке 2.

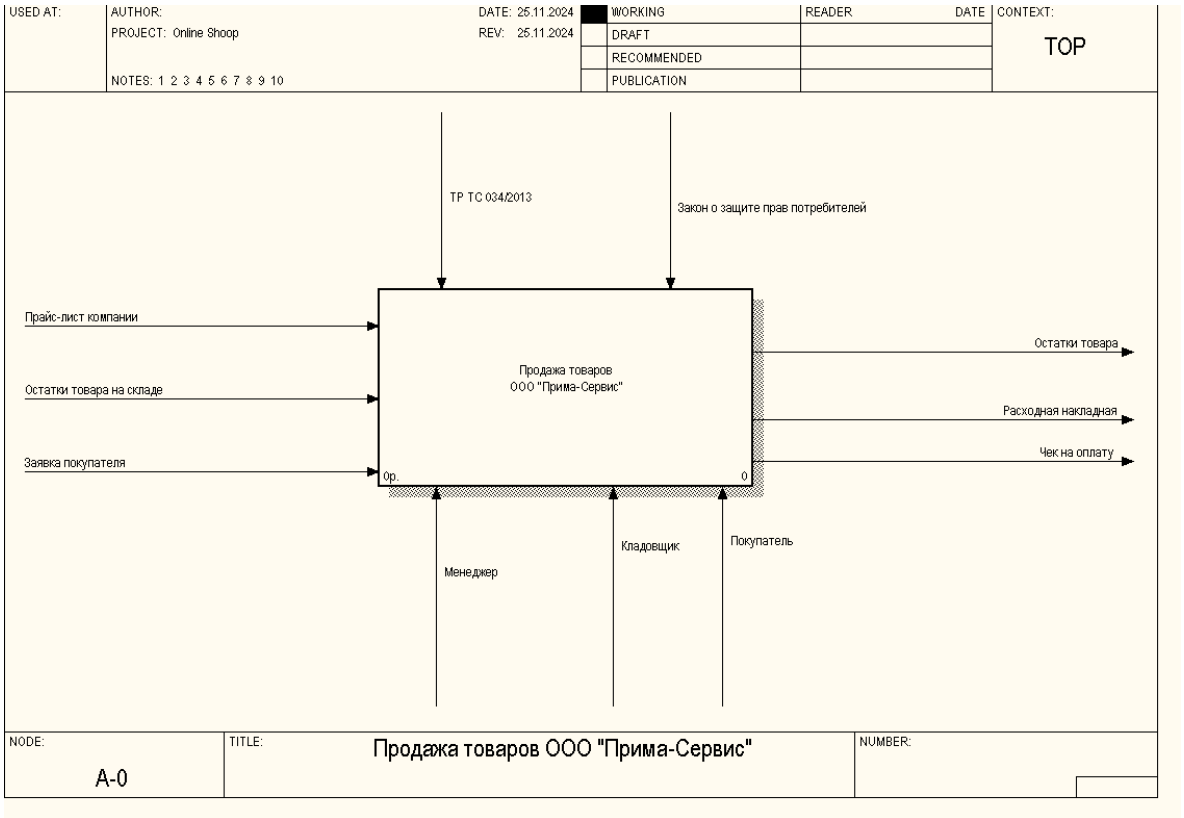


Рисунок 2 – Продажа товаров ООО "Прима-Сервис". Как есть

Входной информацией бизнес-процесса «Продажа товаров ООО "Прима-Сервис» служат следующие информационные потоки:

– Прайс-лист компании,
– Остатки товара на складе,
– Заявка покупателя.

«Выходной информацией бизнес-процесса «Продажа товаров ООО "Прима-Сервис» служат следующие информационные потоки:

– Остатки товара,
– Расходная накладная,
– Чек на оплату.

Управляющей информацией бизнес-процесса «Продажа товаров ООО "Прима-Сервис» служат следующие информационные потоки:

– ТР ТС 034/2013,
– Закон о защите прав потребителей» [11].

Механизмами бизнес-процесса «Продажа товаров ООО "Прима-Сервис» служат следующие информационные потоки:

– Менеджер,
– Покупатель,
– Кладовщик.

Далее рассмотрим процесс «Продажа товаров ООО "Прима-Сервис» в текущем состояние более подробно, то есть без автоматизации и использования информационных систем. Основная продажа товара в компании ООО "Прима-Сервис» осуществляется Менеджером, который контролирует все этапы работы с покупателями, которые могут осуществлять выбор товара как при личном участии, так и удаленно, т.е. по телефону или использую электронную почту. Основными операциями верхнего уровня бизнес-процесса «Продажа товаров ООО "Прима-Сервис» являются (рисунок 3) [18]:

– Выбор товара,
– Проверка наличия товара на складе,
– Оформление покупки,
– Отгрузка товара.

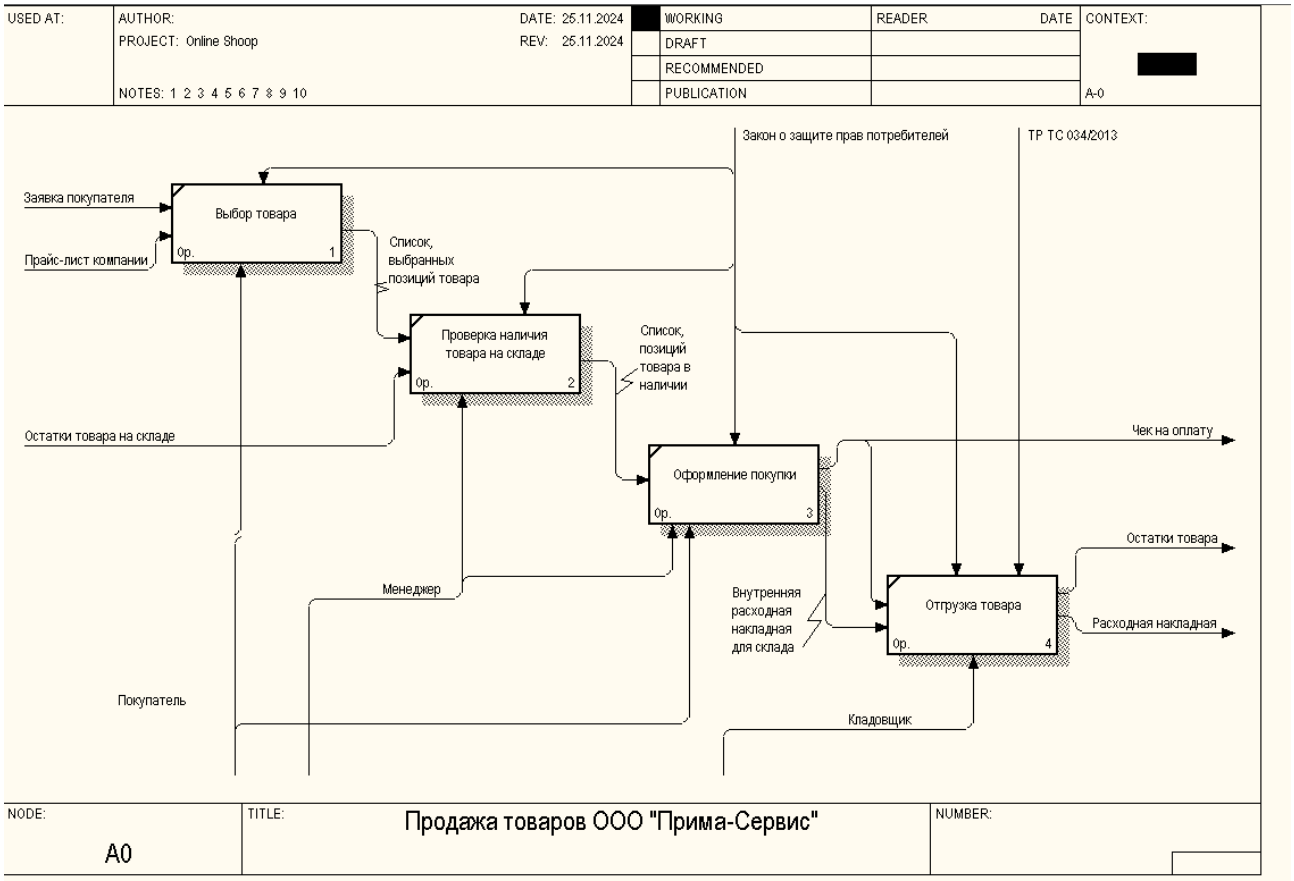


Рисунок 3 – Уровень 1. «Продажа товаров ООО "Прима-Сервис"» Как есть

Особое внимание в ходе преддипломной практике было уделено процессу автоматизации выбора товара и оформление покупки с помощью специально разработанного мобильного приложения, для того чтобы в дальнейшем снизить нагрузку на менеджеров компании и автоматизировать работу склада [3].

Рассмотрим процесс покупки товаров, а именно мясных консервов, с помощью специально разработанного мобильного приложения. Процесс покупки консервов от компании ООО «Прима-Сервис» будет выглядеть

следующим образом, показанном на рисунке 4 [10].

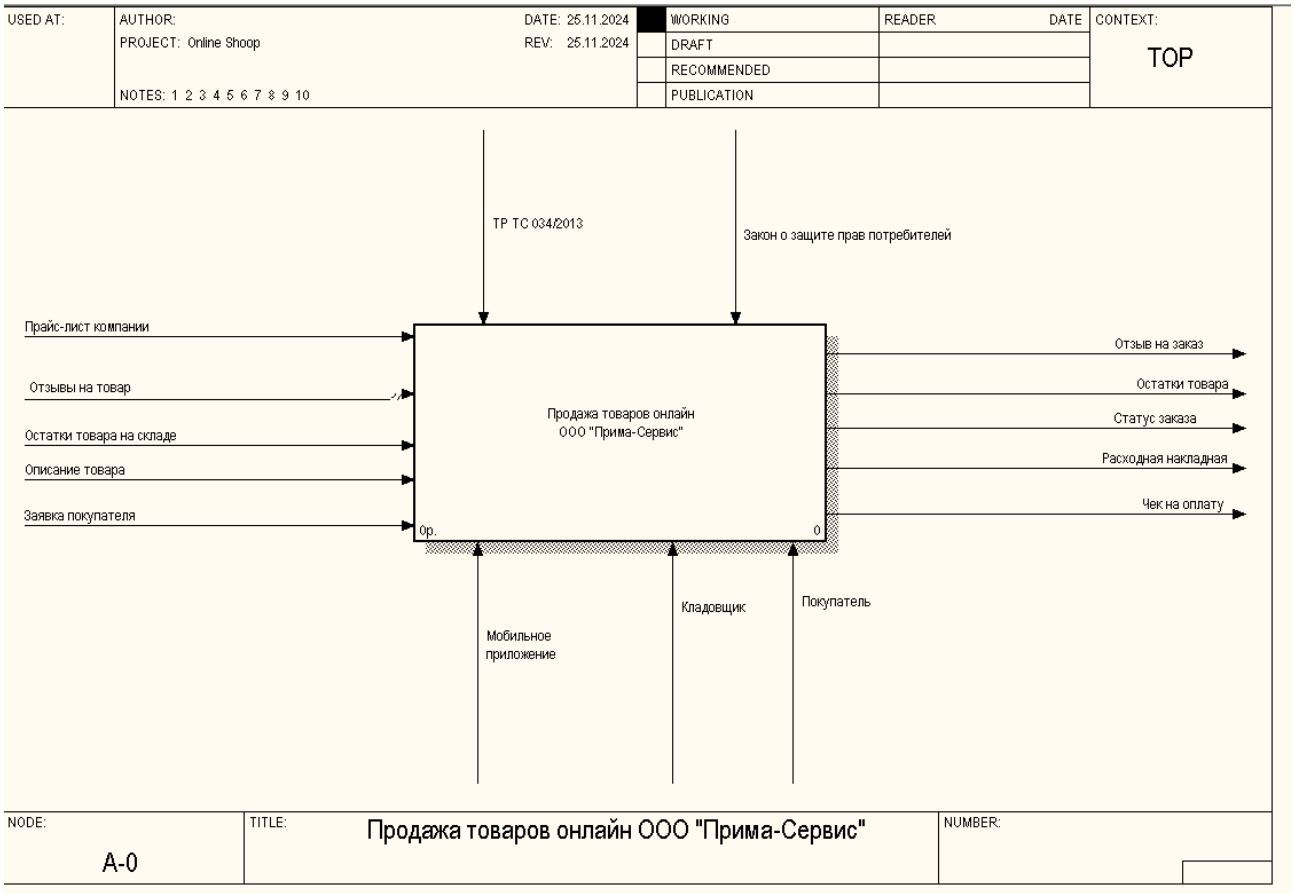


Рисунок 4 - Продажа товаров он-лайн ООО «Прима-Сервис «Как должно быть». Уровень 0

После автоматизации всего процесса продажи товаров он-лайн ООО «Прима-Сервис» через мобильное приложение, во-первых увеличилось количество входной информации, которые раньше могла рассказать менеджер, а теперь необходимо покупателю самостоятельно ознакомиться как с описанием товара, так и с отзывами других покупателей на товар. Также изменения видны и на блоки механизмов, Менеджер сейчас совсем не принимает участия в покупки товаров, все делается автоматизировано с помощью мобильного приложения. А также увеличилось количество выходной информации, появилась информация о Статусе заказа и возможность у покупателей оставлять отзывы, как на товар так и на работу приложения [1]. Рассмотрим изменения, которые произошли после

декомпозиции с блоками они показаны на рисунке 5.

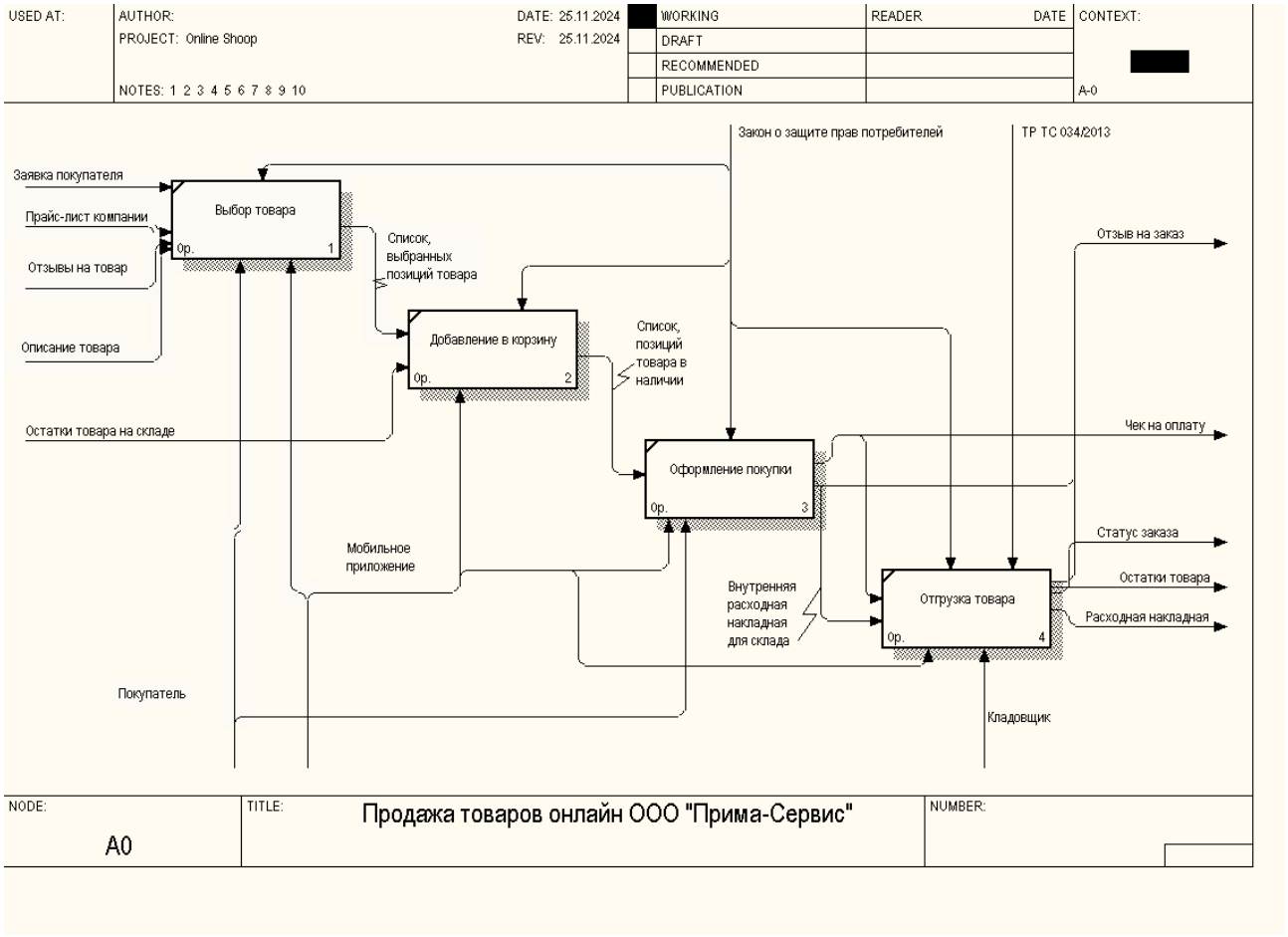


Рисунок 5 - Продажа товаров онлайн ООО «Прима-Сервис, «Как должно быть». Уровень 1

Процесс проверки остатков товаров нужного количества и возможность доставки до покупателя осуществляется на блоке «Добавить товар в корзину».

1.3 Требования к разработке

Определим требования к мобильной разработки онлайн магазина компании ООО «Прима-Сервис» в методологии FURPS+ [1] и отобразим из в таблице 1.

Таблица 1 – Требования к мобильной разработке онлайн магазина компании ООО «Прима-Сервис» в методологии FURPS+

Вид требований	Содержание требований
F. Функциональные требования к мобильной разработки онлайн магазина компании ООО «Прима-Сервис»	− выбор мясных консервов покупателями, − выбор количество товара каждой позиции в корзине, − Наличие в интерфейсе мобильного приложения корзины, − Регистрация пользователя в мобильном приложении.
U. Удобство использования мобильной разработки онлайн магазина компании ООО «Прима-Сервис»	− интуитивно понятный интерфейс, − просмотри информации о товаре.
R. Надежность мобильной разработки онлайн магазина компании ООО «Прима-Сервис»	− проведение оплаты через мобильное приложение, − хранение покупок − отображение статуса покупки .
P. Производительность мобильной разработки онлайн магазина компании ООО «Прима-Сервис»	− подстройка приложения под любой интерфейс.
S. Поддерживаемость мобильной разработки онлайн магазина компании ООО «Прима-Сервис»	− возможность добавление новых позиций для продажи, без изменения интерфейса
+. Ограничения мобильной разработки онлайн магазина компании ООО «Прима-Сервис»	− поддержка продажи товаров , разрешенных для продажи через онлайн

После определение требований к разрабатываемому мобильному приложению переходим к выбору языка и среды разработки. Так как рассмотрение аналогов онлайн магазинов и мобильных приложений не имеет смысла их огромное количество и основной минимальный функционал определен [13].

1.4 Анализ возможных проектных решений

На данный момент в мире информационных технологий существует несколько различных средств разработки мобильных приложений на iOS и Android [7].

На сегодняшний день самыми востребованными средами являются:

– VS Code,
– Xcode,
– Android Studio.

1.4.1 Visual Studio Code

Visual Studio Code обеспечивает возможность разработки приложений для различных операционных систем, включая Windows, macOS и Linux, что позволяет разработчикам работать на платформе по своему выбору.

Visual Studio Code поддерживает множество языков программирования благодаря расширениям, интегрированный терминал, рефакторинг, навигация по коду, отладку.

1.4.2 Xcode

Xcode – это среда разработки от компании Apple. Она предназначенна для создания приложений для iOS, macOS, watchOS и tvOS.

Из положительных сторон у Xcode глубокая интеграция с экосистемой Apple и предоставление всех необходимых инструментов для разработки приложений для данной экосистемы. Xcode включает в себя инструмент для визуализации проектирования интерфейса, эмуляторы для тестирования приложения на устройствах, отладку, инструменты профилирования.

Из недостатков Xcode требует больших системных ресурсов. Из-за работы с большими проектами и данными могут возникнуть сложности на слабых станциях.

1.4.3 Android Studio

Android Studio – это среда разработки от компании Google для создания приложений под систему Android.

«Ключевыми особенностями является хороший редактор кода, эмулятор Android, визуальный редактор, инструменты для отладки, интеграция с GooglePlay.

Недостатками данной среды можно назвать высокую требовательность к ресурсам системы, зависимость от экосистемы Google» [16].

Проанализировав достоинства и недостатки данных сред разработки, было принято решение разрабатывать мобильное приложение с помощью Visual Studio Code в связи с более удобным и интуитивно понятным интерфейсом, большим выбором подключаемых библиотек и возможностью писать приложение в одной IDE для двух платформ.

В первом разделе рассмотрены вопросы анализа предметной области, а именно: рассмотрена деятельность ООО "Прима-Сервис, на базе которой выполнялась выпускная квалификационная работа. Так же описаны бизнес-процессы компании (до и после автоматизации), составлены требования на разработку мобильного приложения, рассмотрены средства разработки мобильных приложений и выбрана Visual Studio Code.

2 Проектирование мобильного приложения

2.1 Логическое моделирование информационной системы

Архитекрута Flutter BloC представляет собой State Manager, средство по управлению состояниями. В архитектуре Flutter BLoC данные хранятся не в моделях, в отличии от MVVM, а в сущностях – Entities. Все методы по запросу данных с сервера должны находиться в репозиториях – Repository. Все возможные состояния бизнес-логики хранятся в классе State, события изменения состояний – в Event (рисунок 6).

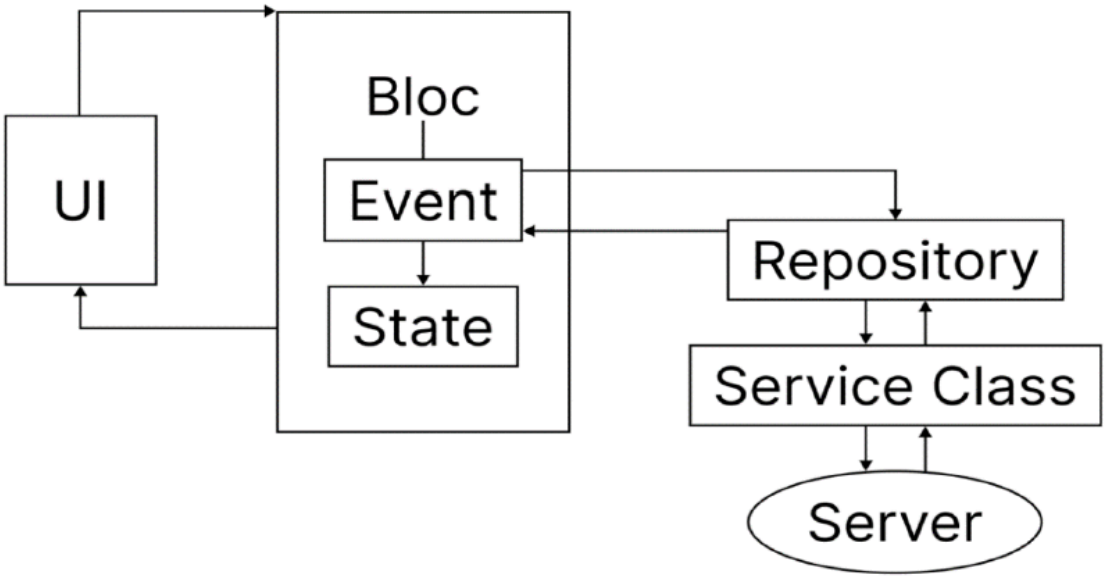


Рисунок 6 – Архитектура Flutter BloC

Класс Bloc выступает в роли ViewModel: вызывает методы из репозитория, и в зависимости от их ответа меняет State. Flutter BloC максимально разделяет бизнес-логику и UI, всё по принципам SOLID [17].

Всё взаимодействие пользователя будет осуществляться через мобильное приложение. Приложение состоит из двух ключевых компонентов:

– клиентская часть, которая представляет собой графический пользовательский интерфейс;
– база данных, место в котором информация сохраняется в соответствии с утсновленными.

В качестве базы данных выбрана облачная платформа Firebase. Firebase – это набор инструментов и сервисов для разработки мобильных и web-приложений.

Кроссплатформенность упрощает интеграцию Firebase в приложения для Android, iOS и web.

Firebase платформа с набором облачных инструментов, которые помогают создавать, управлять и масштабировать приложения. Это изображено на рисунке 7.

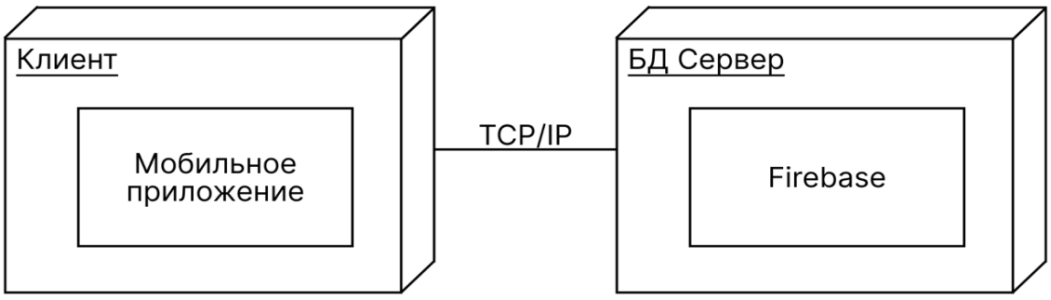


Рисунок 7 – Диаграмма развёртывания приложения

Firebase предоставляет возможности для организации, контроля и администрирования данных в режиме реального времени, обеспечивая синхронизацию данных между клиентами и сервером без необходимости создания собственного сервера или управления инфраструктурой.

2.2 Логическое моделирование

Диаграмма вариантов использования приложения приведена на рисунке 8.

Рисунок 8 – Диаграмма вариантов использования мобильного приложения

«На рисунке 9 представлено взаимодействие компонентов мобильного приложения в виде диаграммы последовательности.

Используя данное приложение, пользователь сможет:

- регистрироваться,
- авторизоваться,
- просматривать товары,
- детально изучать состав и описание консервов,
- добавлять понравившиеся товары в корзину» [6].

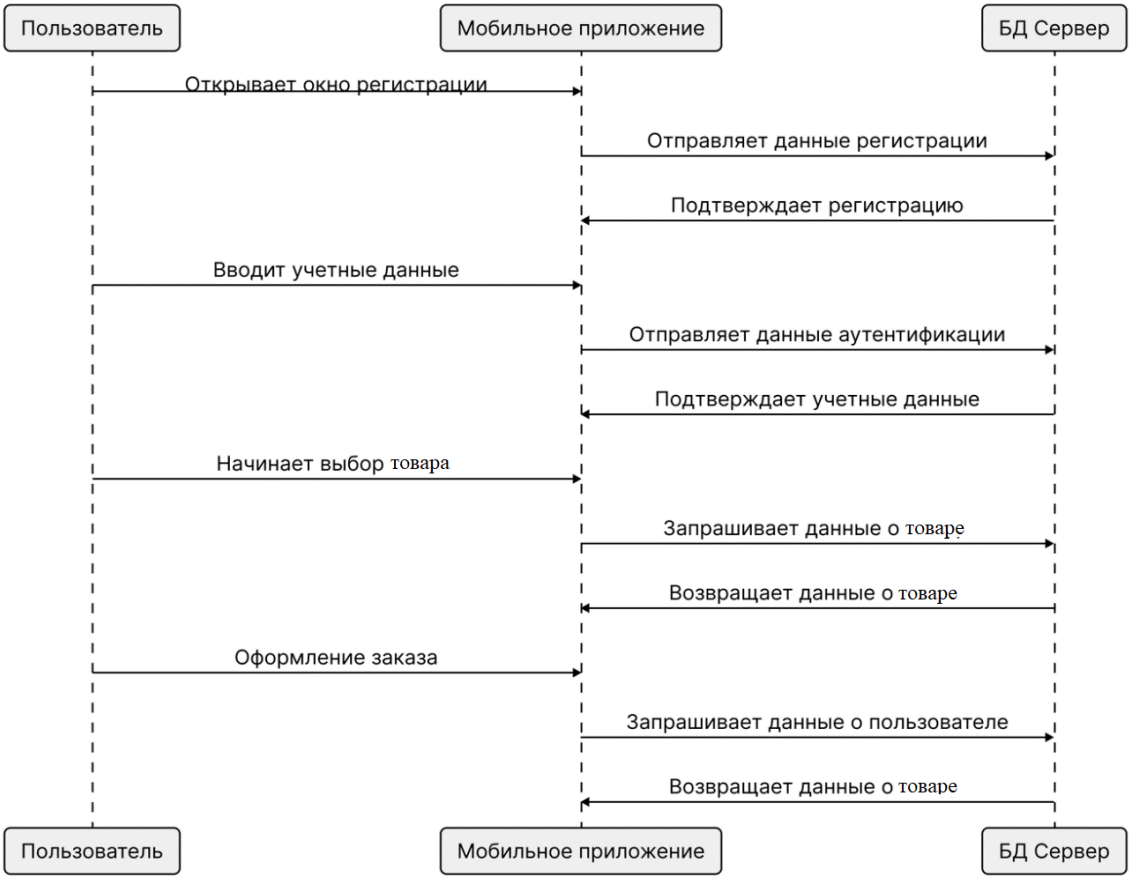


Рисунок 9 - Диаграмма последовательности

На UML-диаграмме классов можно увидеть основные файлы проекта (рисунок 10).

Каждый из них отвечает за свою задачу, что соответствует принципу разделения ответственности и модульности.

В файле main.dart вызывается основная глобальная функция main(), которая инициализирует ключевые компоненты приложения — Firebase и Bloc, а также запускает главный экран приложения.

В классе MyAppView задается начальный экран приложения: если пользователь зарегистрирован, происходит переход к каталогу товаров.

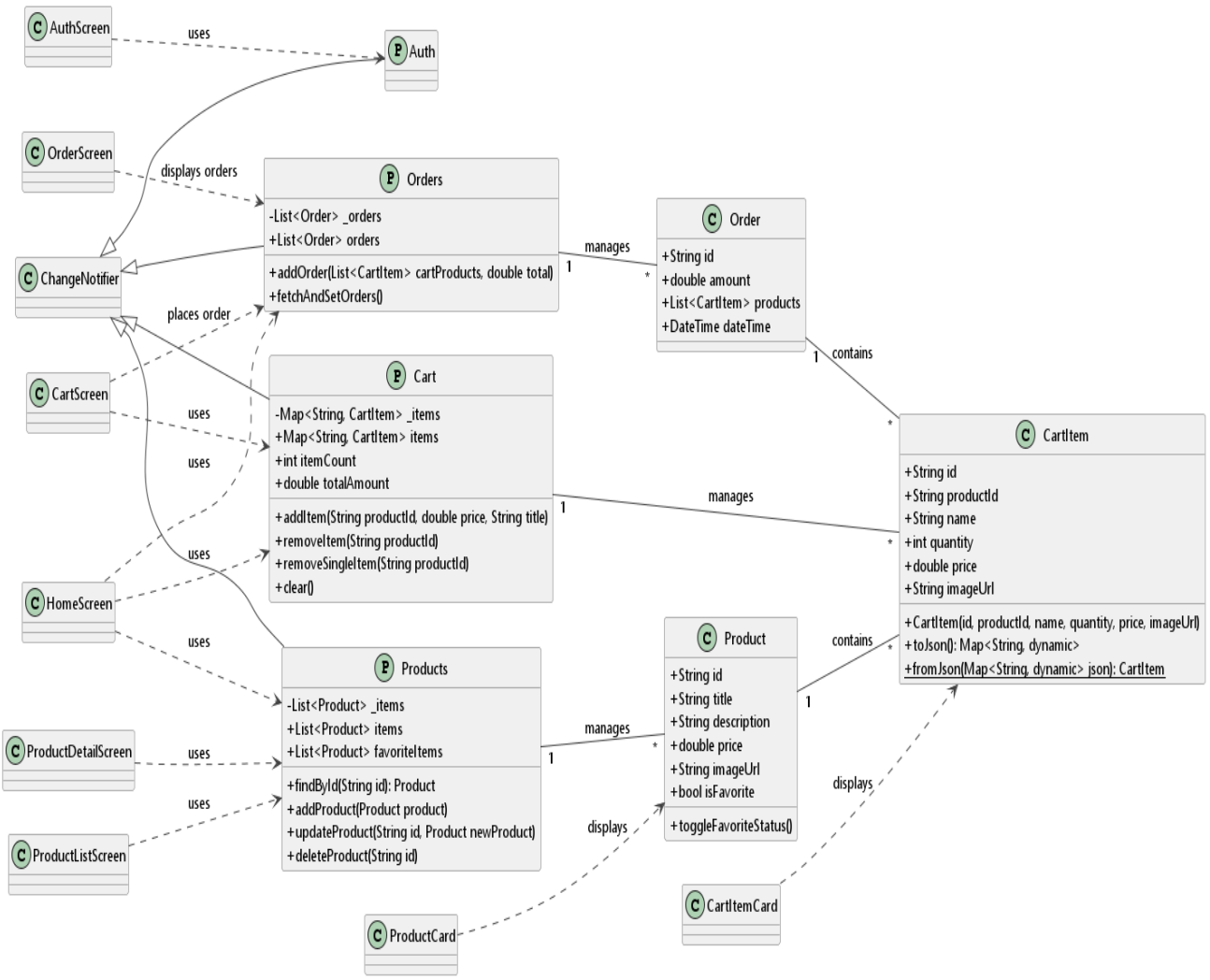


Рисунок 10 – Диаграмма классов

В классе Auth осуществляется проверка состояний пользователя и подписка на их изменения. В нем реализована логика вызова методов регистрации и авторизации. В зависимости от успешности ответа происходит изменение состояний, которые, в свою очередь, влияют на итоговое состояние Auth.

Класс AuthScreen отвечает за отображение формы регистрации, формы входа и демонстрацию экрана приветствия.

Класс ProductListScreen отвечает за отображение списка продуктов, ProductDetailsScreen – за подробную инфорацию о продукте, а HomeScreen демонстрирует основной экран приложения.

Класс Cart отвечает за бизнес-логику связанную с корзиной. Он управляет добавлением, удалением и количеством товара. В CartScreen отображается содержимое корзины, а OrderScreen – список всех заказов.

Таким образом, обеспечивается четкое разделение между компонентами интерфейса, бизнес-логики и доступа к данным.

2.3 Проектирование базы данных

При проектировании модели базы данных используем программу MS Visio (рисунок 11) [19].

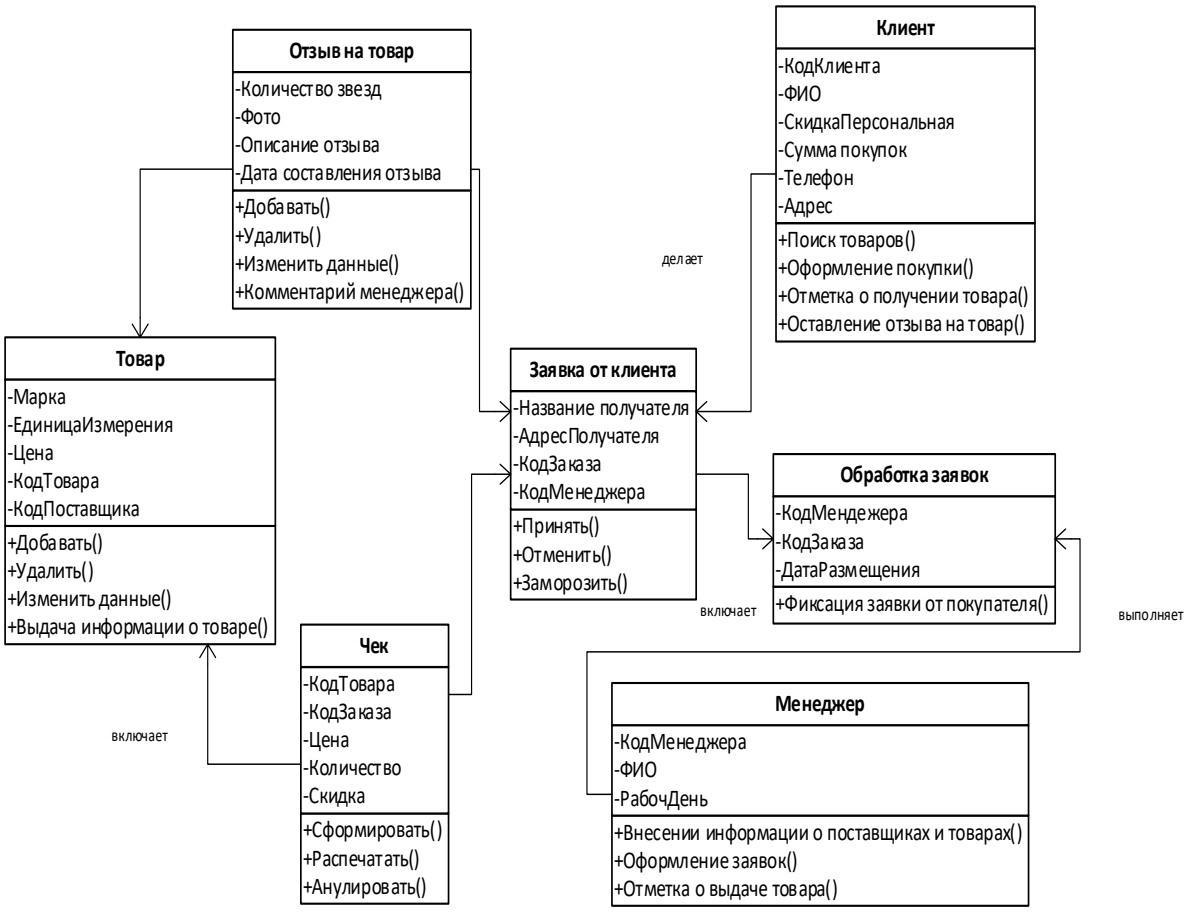


Рисунок 11 – Логическое проектирование базы данных «Интернет-магазина»

«Рассмотрим подробно схему, которая содержит описание следующих объектов автоматизации:

– Отзыв на товар,
– Товар,
– Чек,
– Заявка покупателя,
– Обработка заявок,
– Менеджер,
– Клиент» [15].

Класс «Клиент» используется для хранения информации о клиентах Интернет-магазина и истории покупок клиента. Структура класса представлена в таблице 2.

Таблица 2 - Структура «Клиент»

Атрибут	Операция
КодКлиента	Поиск товара
Фио	Оформить покупки
Сумма покупок	Отметка о получении товара
Телефон	Оставление отзыва на товар
Адрес	

Класс «Товар» используется для хранения информации о поставщиках [3] с которыми работает интернет-магазин. Структура класса представлена в таблице 3.

Таблица 3 - Структура класса «Товар»

Атрибут	Операция
Марка	Добавить
ЕдиницаИзмерения	Удалить
Цена	Изменить данные
КодПоставщика	Выдача информации о товаре
КодТовара	

Класс «Отзыв на товар» используется для хранения информации о полученном товаре клиентом и его отзыве, отзыва на товар в дальнейшем могут помогать магазину продавать более популярные товары, работать с качеством товара от поставщика, отслеживать доставку товара. Так же менеджеры магазина могут комментировать отзывы на товар и модерировать отзывы, которые не отражают суть работы магазина или качество товара [12]. Структура класса представлена в таблице 4.

Таблица 4 - Структура класса «Отзыв на товар»

Атрибут	Операция
Количество звезд	Добавить
Фото	Удалить
Описание отзывы	Изменить данные
Дата составления отзыва	Комментарий менеджера

Класс «Заявка от покупателя» используется для хранения информации о выбранном товаре Клиента и является началом оформления покупки. Структура класса представлена в таблице 5.

Таблица 5 - Структура класса «Заявка от покупателя»

Атрибут	Операция
Название получателя	Добавить заявку
АдресПолучателя	Удалить заявку
КодЗаказа	Заморозить
КодМенеджера	

Класс «Чек» используется для хранения информации [4] об оплате заказа клиентов. Структура класса представлена в таблице 6.

Таблица 6 - Структура класса «Чек»

Атрибут	Операция
КодТовара	Сформировать
КодЗаказа	Распечатать
Цена	Аннулировать
Количество	
Скидка	

Класс «Обработка заявок» используется для хранения информации об принятых заявках и статусе их отгрузки. Структура класса представлена в таблице 7.

Таблица 7 - Структура класса «Обработка заявок»

Атрибут	Операция
КодМенеджера	Добавить
КодЗаказа	Удалить
ДатаРазмещения	

Класс «Менеджер» используется для хранения информации менеджерах [4], которые работают над заказами клиентов. Структура класса представлена в таблице 8.

Таблица 8 - Структура класса «Менеджер»

Атрибут	Операция
КодМенеджера	Внесение информации о поставщиках товара
Фио	Оформление заявок
Рабочий день	Отметка о выдаче товара

Выделенные классы преобразуем в сущности базы данных, в результате для оптимизации приложения разделим часть классов в разные базы данных, т.е. сделаем распределенную базу данных [19]. Деление проведем на

информацию ДО покупки и оформления товара и информацию ПОСЛЕ покупки. В первой базе данных будет информация для поиска и выбора, во второй базе данных будет уже храниться и анализироваться информация по проданным позициям магазина, а также будет осуществляется действия по проданным позициям – отзывы, доставка и т.д.

Подробно рассмотрим разработку базы данных предназначенной для хранения информации, т.е. ДО покупки (рисунок 12) [2].

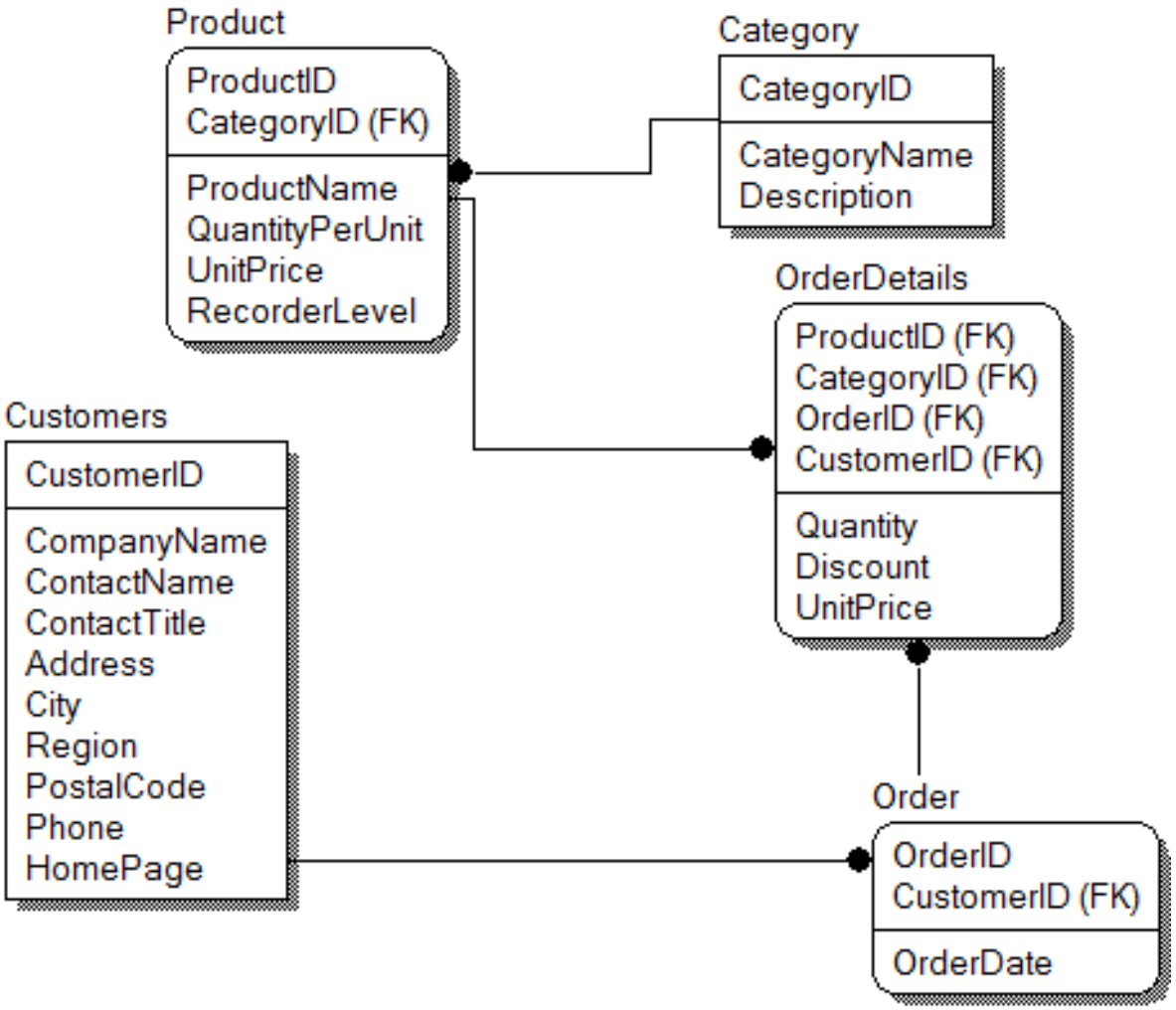


Рисунок 12 – Схема база данных

Согласно разработанной модели создадем таблицы. Проект создание таблицы клиенты показан на рисунке 13.

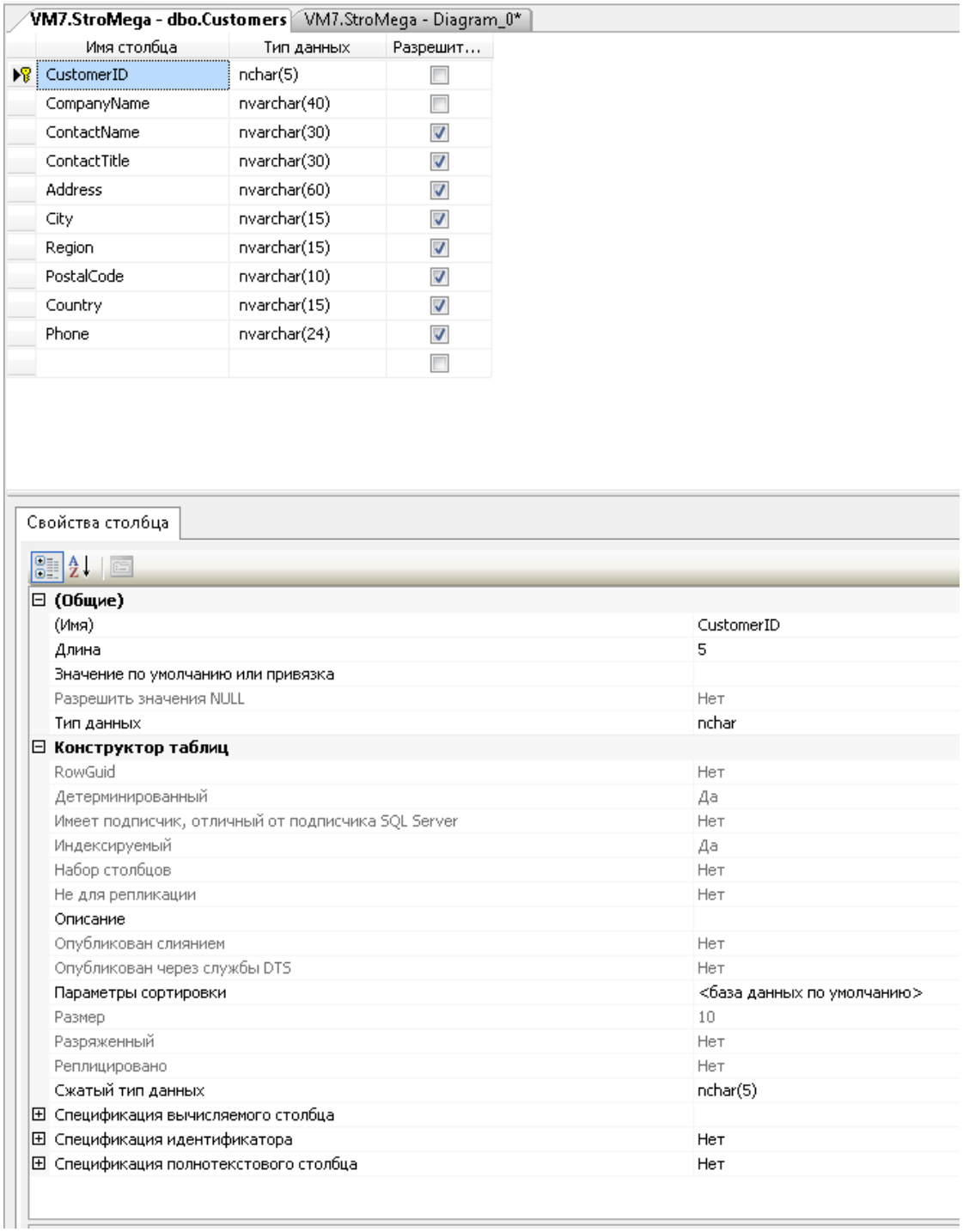


Рисунок 13 – Проект таблицы клиенты

Проект создание таблицы товары показан на рисунке 14.

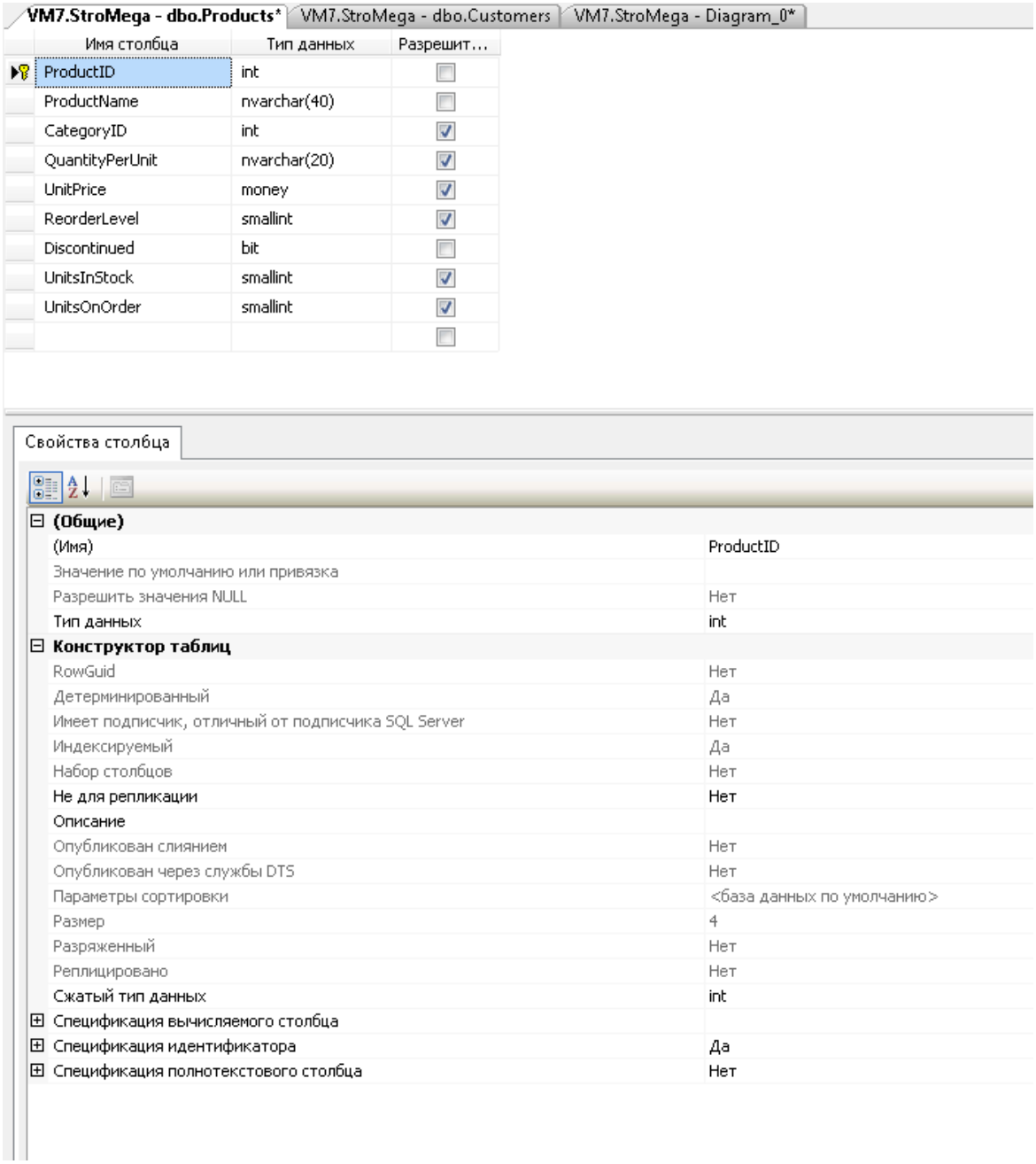


Рисунок 14 – Проект таблицы Товары

Для остальных таблиц базы данных покажем только ключевые поля и имена столбцов.

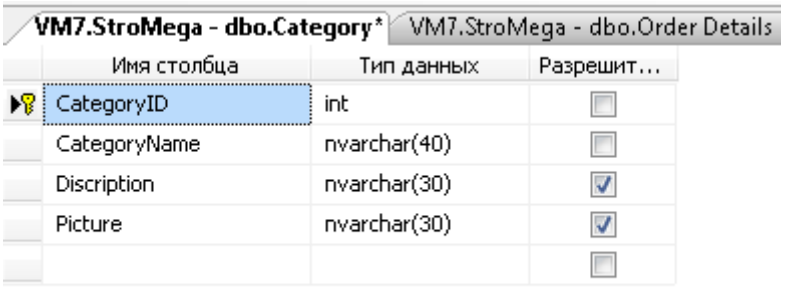


Рисунок 15 – Таблица Категории, с описанием имен столбцов и типов данных

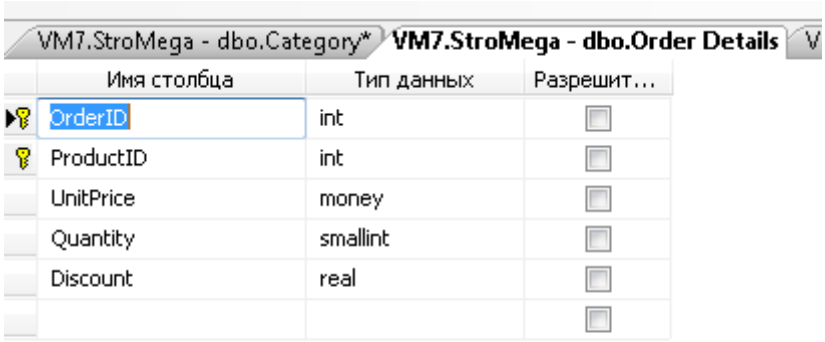


Рисунок 16 – Таблица Детали заказа, с описанием имен столбцов и типов данных

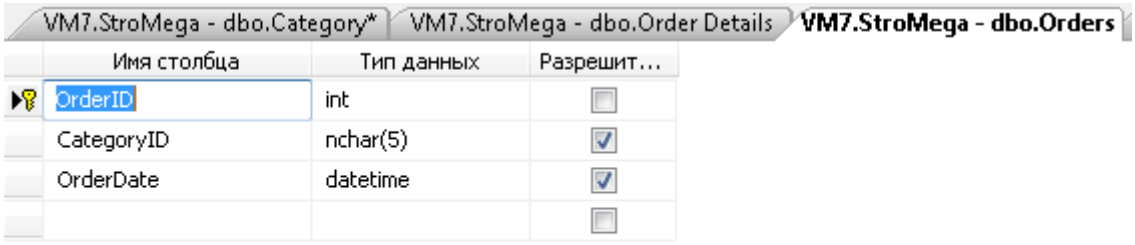


Рисунок 17 – Таблица Заказ, с описанием имен столбцов и типов данных

Все таблицы соединим в схему данных, которая показана на рисунке 18.

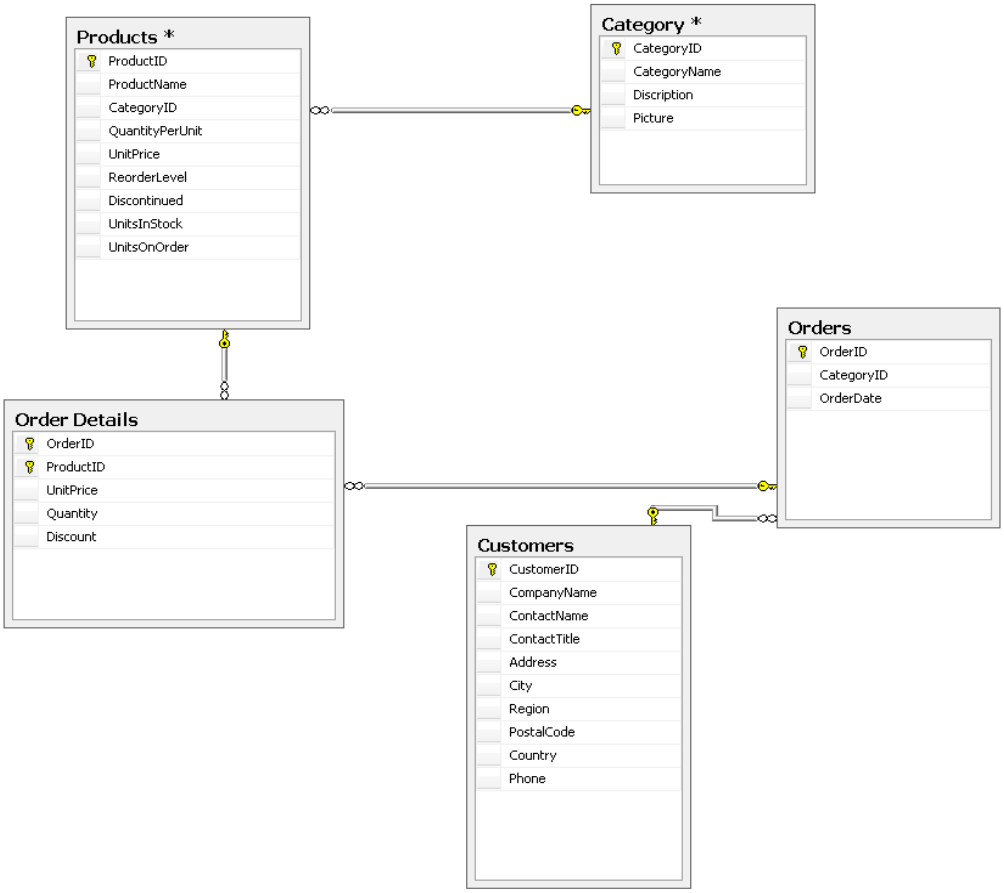


Рисунок 18 – Физическая модель базы данных

После всех этапов проектирования, далее приступаем к разработке приложения.

Во второй главе были рассмотрены вопросы проектирования приложения для онлайн магазина. Определены функциональные требования к системе, пользователи системы. Проведено логическое и физическое проектирование базы данных, которая будет являться ядром, разрабатываемого приложения.

3 Разработка мобильного приложения

3.1 Разработка пользовательского интерфейса

В процессе выполнения выпускной работы разработана система для авторизации пользователей (рисунок 19). Для авторизации используется класс WelcomeScreen. Пользователю необходимо ввести электронную почту и пароль в соответствующие поля.

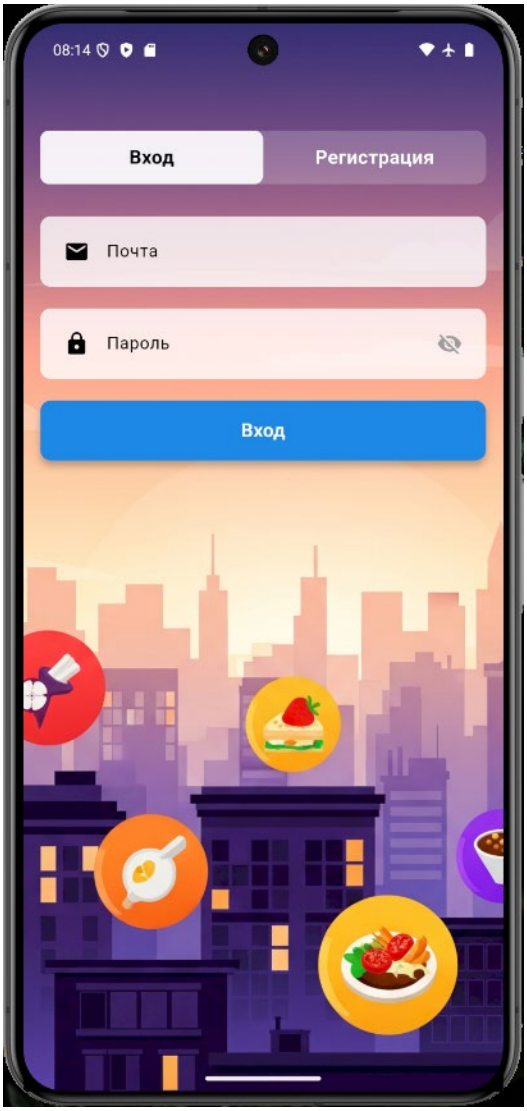


Рисунок 19 – Экран входа в приложение

Код блока авторизации пользователя представлен на рисунке 20.

```dart
class HomeScreen extends StatelessWidget {
  final String userIdentifier;
  const HomeScreen({Key? key, required this.userIdentifier}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Добро пожаловать!'),
        automaticallyImplyLeading: false,
        backgroundColor: Colors.transparent,
        elevation: 0,
      ), // AppBar
      extendBodyBehindAppBar: true,
      body: Stack(
        children: [
          Positioned.fill(
            child: Image.asset(
              'assets/images/background_image.jpg',
              fit: BoxFit.cover,
            ), // Image.asset
          ), // Positioned.fill
          Center(
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Container(
                  padding: const EdgeInsets.all(15),
                  decoration: BoxDecoration(
                    color: Colors.black.withOpacity(0.6),
                    borderRadius: BorderRadius.circular(15),
                  ), // BoxDecoration
                  child: Text(
                    // Используем userIdentifier
                    'Вы успешно вошли как: $userIdentifier',
                    textAlign: TextAlign.center,
                    style: const TextStyle(fontSize: 20, color: Colors.white),
                  ), // Text
                ), // Container
                const SizedBox(height: 30),
                ElevatedButton(
                  onPressed: () {
                    Navigator.pushReplacementNamed(context, '/');
                  },
                  child: const Text('Выйти'),
                ), // ElevatedButton
              ],
            ), // Column
          ), // Center
```

Рисунок 20 – Блок кода авторизации пользователя

«Если пользователь ещё не зарегистрирован в системе, то ему необходимо зарегистрироваться (рисунок 21). Для этого необходимо ввести следующие данные:

– электронная почта,

– пароль,

– имя» [14].

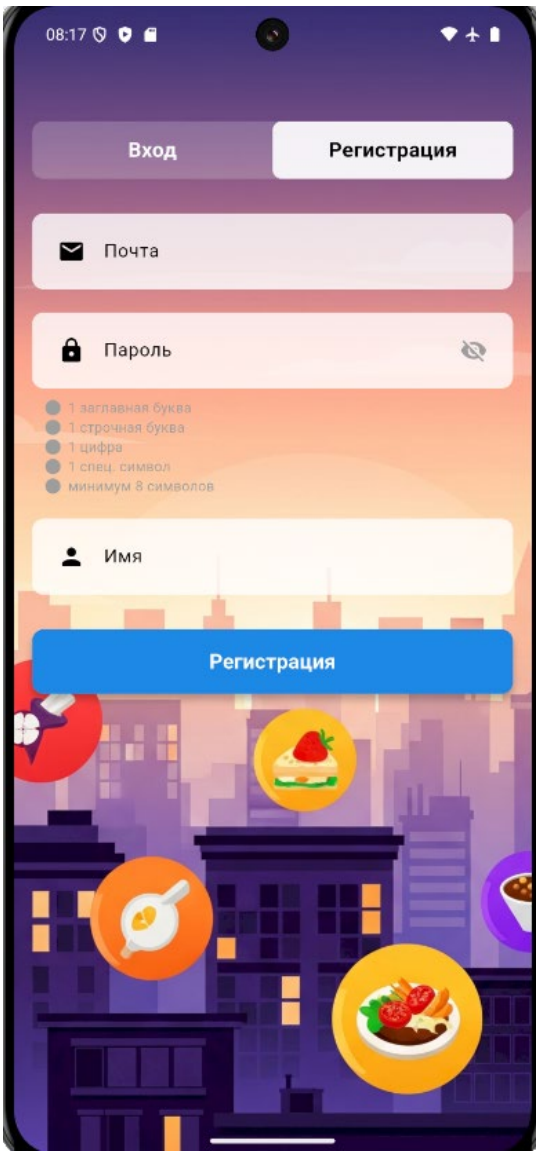
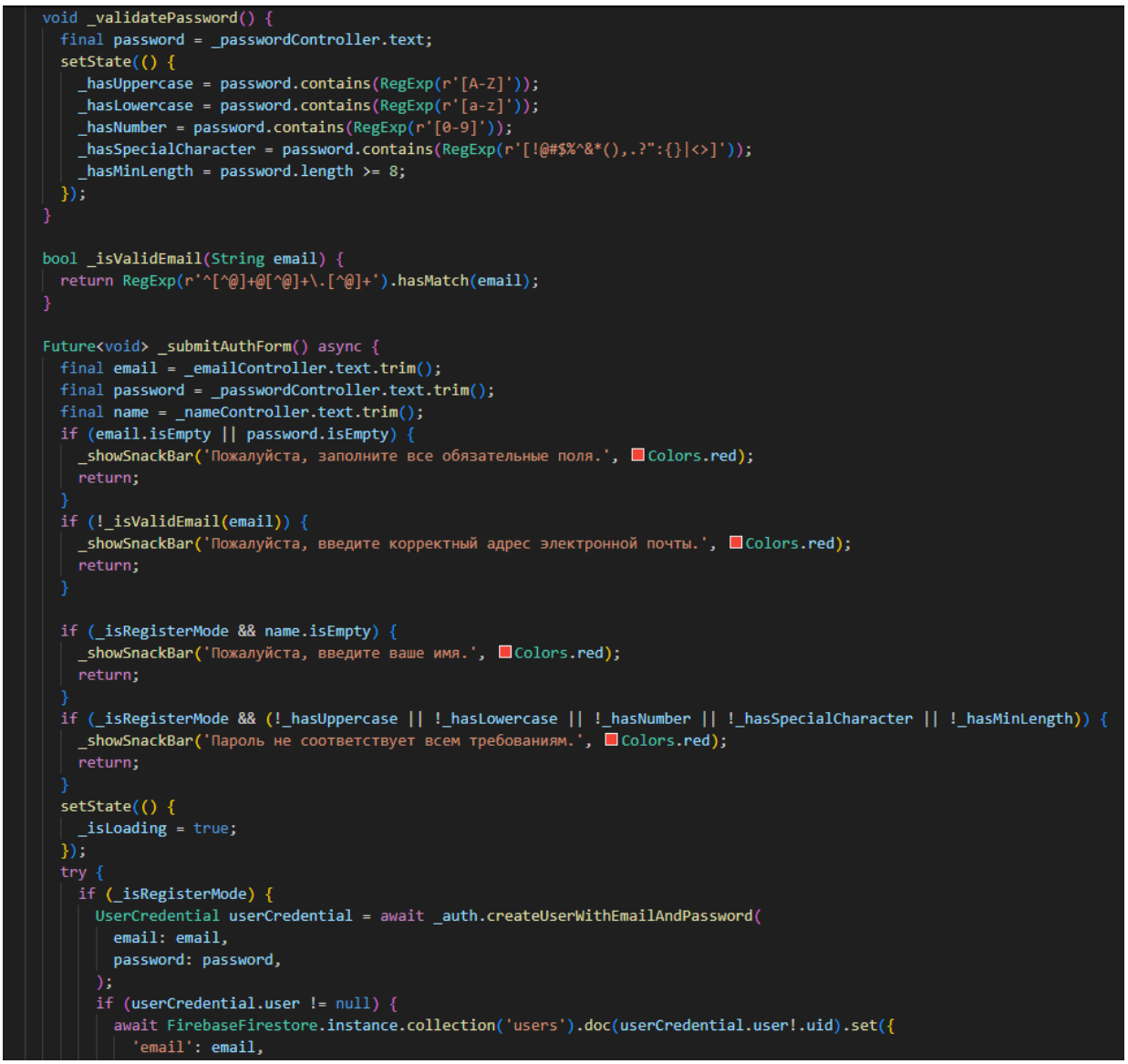


Рисунок 21 – Экран регистрации в приложение

Код данного блока представлен на рисунке 22 и полностью представлен в Приложении А.



```
void _validatePassword() {
  final password = _passwordController.text;
  setState(() {
    _hasUppercase = password.contains(RegExp(r'[A-Z]'));
    _hasLowercase = password.contains(RegExp(r'[a-z]'));
    _hasNumber = password.contains(RegExp(r'[0-9]'));
    _hasSpecialCharacter = password.contains(RegExp(r'[!@#$%^&*(),.?":{}|<>]'));
    _hasMinLength = password.length >= 8;
  });
}

bool _isValidEmail(String email) {
  return RegExp(r'^[^@]+@[^@]+\.[^@]+').hasMatch(email);
}

Future<void> _submitAuthForm() async {
  final email = _emailController.text.trim();
  final password = _passwordController.text.trim();
  final name = _nameController.text.trim();
  if (email.isEmpty || password.isEmpty) {
    _showSnackBar('Пожалуйста, заполните все обязательные поля.', Colors.red);
    return;
  }
  if (!_isValidEmail(email)) {
    _showSnackBar('Пожалуйста, введите корректный адрес электронной почты.', Colors.red);
    return;
  }

  if (_isRegisterMode && name.isEmpty) {
    _showSnackBar('Пожалуйста, введите ваше имя.', Colors.red);
    return;
  }
  if (_isRegisterMode && (!_hasUppercase || !_hasLowercase || !_hasNumber || !_hasSpecialCharacter || !_hasMinLength)) {
    _showSnackBar('Пароль не соответствует всем требованиям.', Colors.red);
    return;
  }
  setState(() {
    _isLoading = true;
  });
  try {
    if (_isRegisterMode) {
      UserCredential userCredential = await _auth.createUserWithEmailAndPassword(
        email: email,
        password: password,
      );
      if (userCredential.user != null) {
        await FirebaseFirestore.instance.collection('users').doc(userCredential.user!.uid).set({
          'email': email,
```

Рисунок 22 – Блок кода регистрации в приложении

Важное условие, чтобы пароль удовлетворял ниже представленным требованиям:

– 1 символ верхнего регистра,

– 1 символ нижнего регистра,

– 1 цифра,

– 1 специальный символ,

– минимум 8 символов.

Код проверки условий пароля представлен на рисунке 23.

```dart
Widget _buildPasswordField() {
  return Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      Container(
        padding: const EdgeInsets.symmetric(horizontal: 10, vertical: 5),
        decoration: BoxDecoration(
          color: Colors.white.withOpacity(0.8),
          borderRadius: BorderRadius.circular(10),
        ), // BoxDecoration
        child: TextField(
          controller: _passwordController,
          obscureText: !_isPasswordVisible,
          decoration: InputDecoration(
            labelText: 'Пароль',
            border: InputBorder.none,
            prefixIcon: const Icon(Icons.lock),
            suffixIcon: IconButton(
              icon: Icon(
                _isPasswordVisible ? Icons.visibility : Icons.visibility_off,
                color: Colors.grey,
              ), // Icon
              onPressed: () {
                setState(() {
                  _isPasswordVisible = !_isPasswordVisible;
                });
              },
            ), // IconButton
          ), // InputDecoration
        ), // TextField
      ), // Container
      if (_isRegisterMode)
        Padding(
          padding: const EdgeInsets.only(left: 10.0, top: 8.0),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              _buildPasswordRequirement('1 заглавная буква', _hasUppercase),
              _buildPasswordRequirement('1 строчная буква', _hasLowercase),
              _buildPasswordRequirement('1 цифра', _hasNumber),
              _buildPasswordRequirement('1 спец. символ', _hasSpecialCharacter),
              _buildPasswordRequirement('минимум 8 символов', _hasMinLength),
            ],
          ), // Column
        ), // Padding
    ],
  ); // Column
```

Рисунок 23 – Проверка сложности пароля

Данные электронной почты и имя пользователя отправляются в облачную базу данных Firebase. Это приведено на рисунке 24.

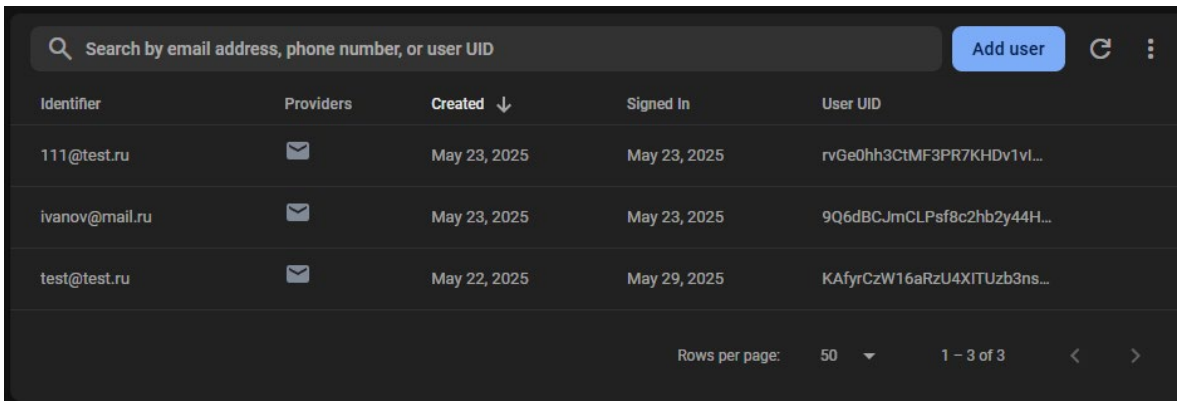


Рисунок 24 – Таблица с пользователям в Firebase

Далее в ходе выпускной работы будет реализован механизм выбора товара из меню приложения.

После успешного входа в приложение, открывает экран меню. Меню состоит из набора карточек товара, где указаны следующие данные:

– название;

– описание;

– цена;

– цена со скидкой.

Экран реализации меню приведен на рисунке 25.

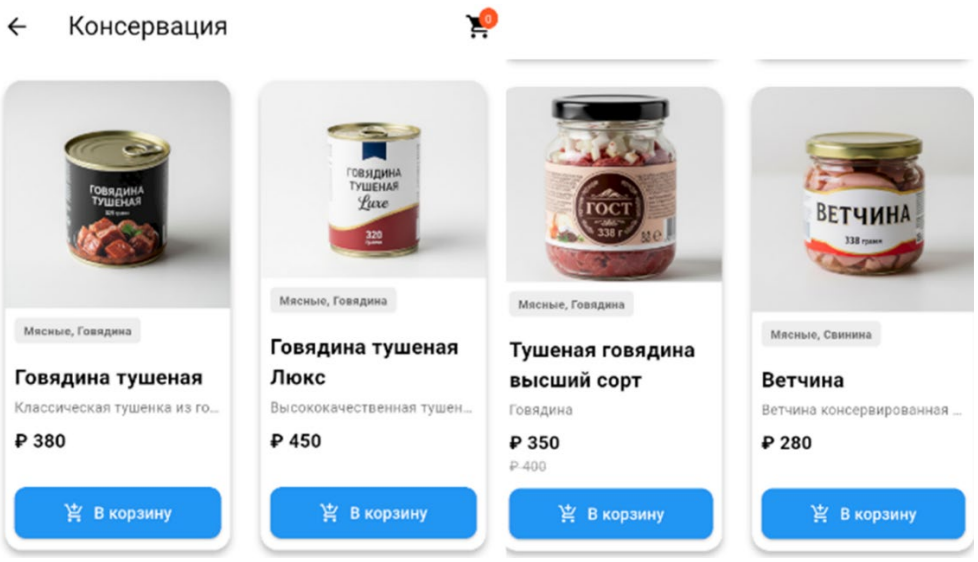


Рисунок 25 – Экран реализации меню

Указанные данные хранятся в Firebase, откуда они и загружаются в приложение. Это отображено на рисунке 26.

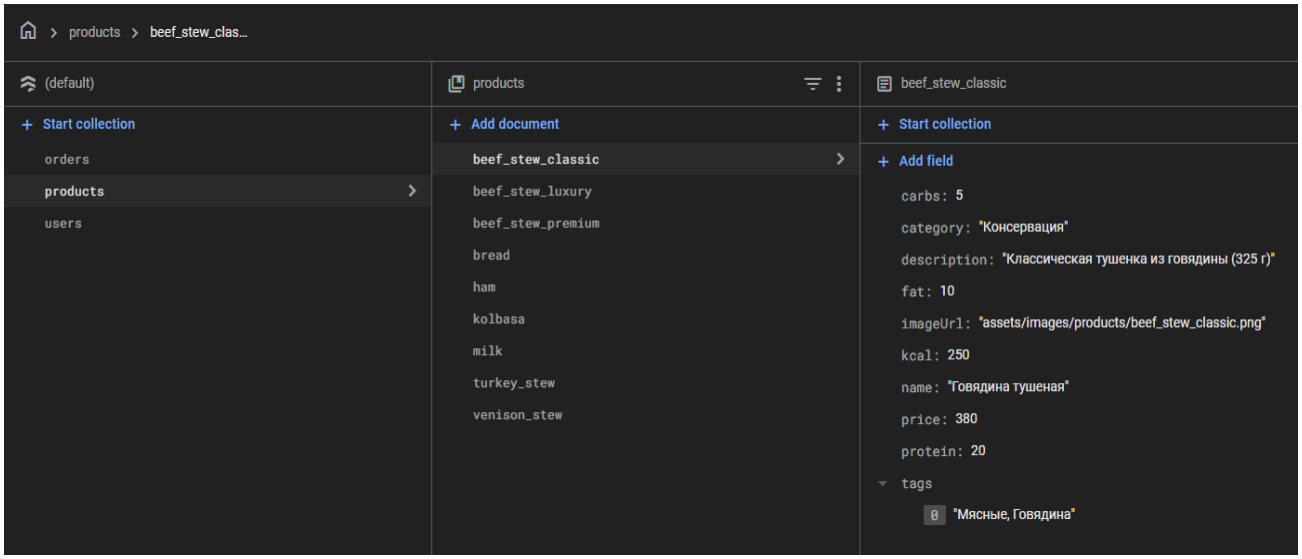


Рисунок 26 – Данные о блюде в БД Firebase

Все изображения также хранятся в Firebase, в разделе «Storage». В базе данных указана url-ссылка на каждое изображение в хранилище (рисунок 27).

Рисунок 27 – «Storage» в БД Firebase

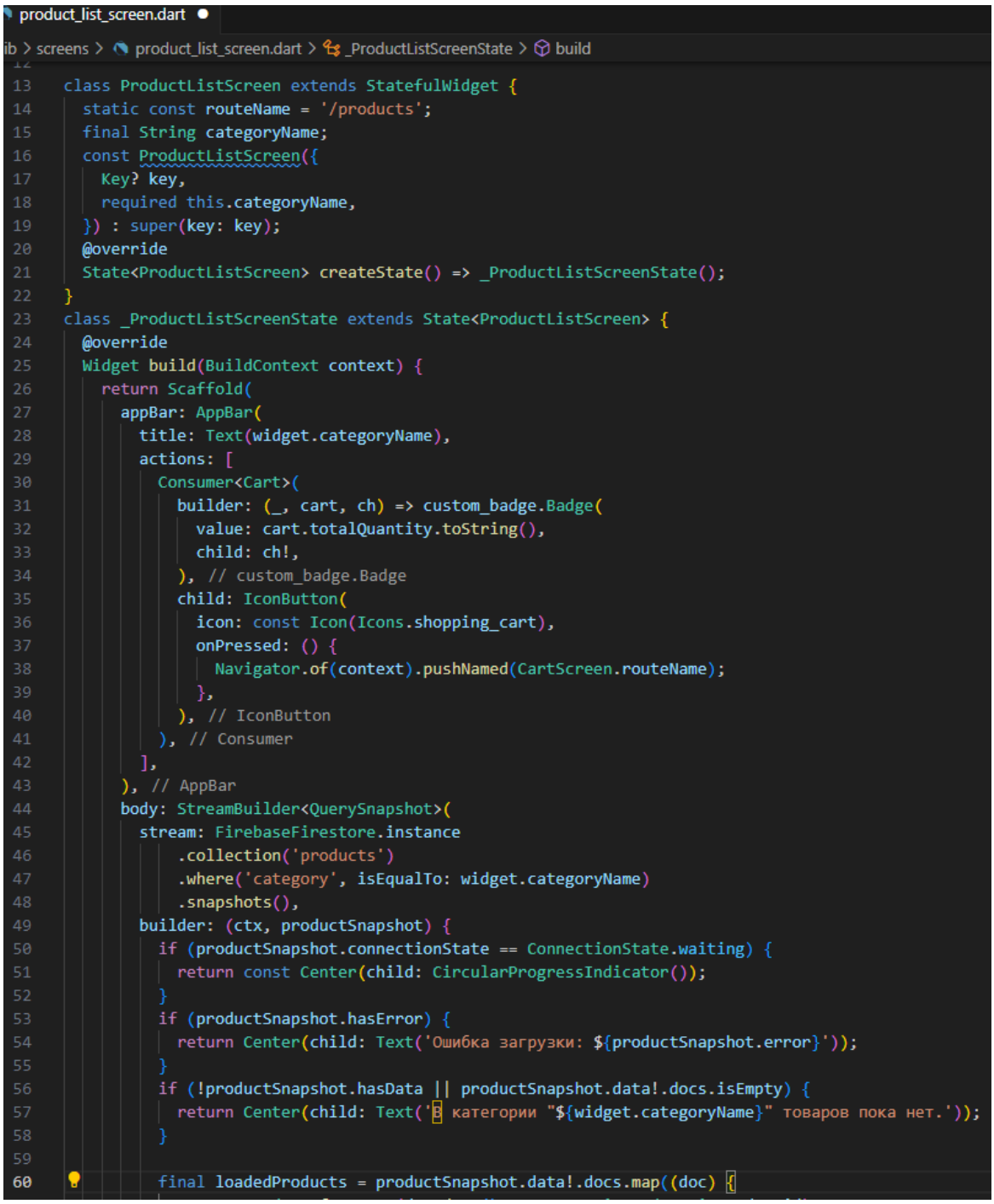


Рисунок 28 – Код реализации меню

При нажатии на карточку товара, открывается отдельный экран, в котором представлена подробная информация о еде, а именно:

– название,

– цена,

– цена со скидкой,

– описание,
– калораж,
– белки,
– жиры,
– углеводы.

Это отображено на рисунке 29.

Рисунок 29 – Подробная карточка товара

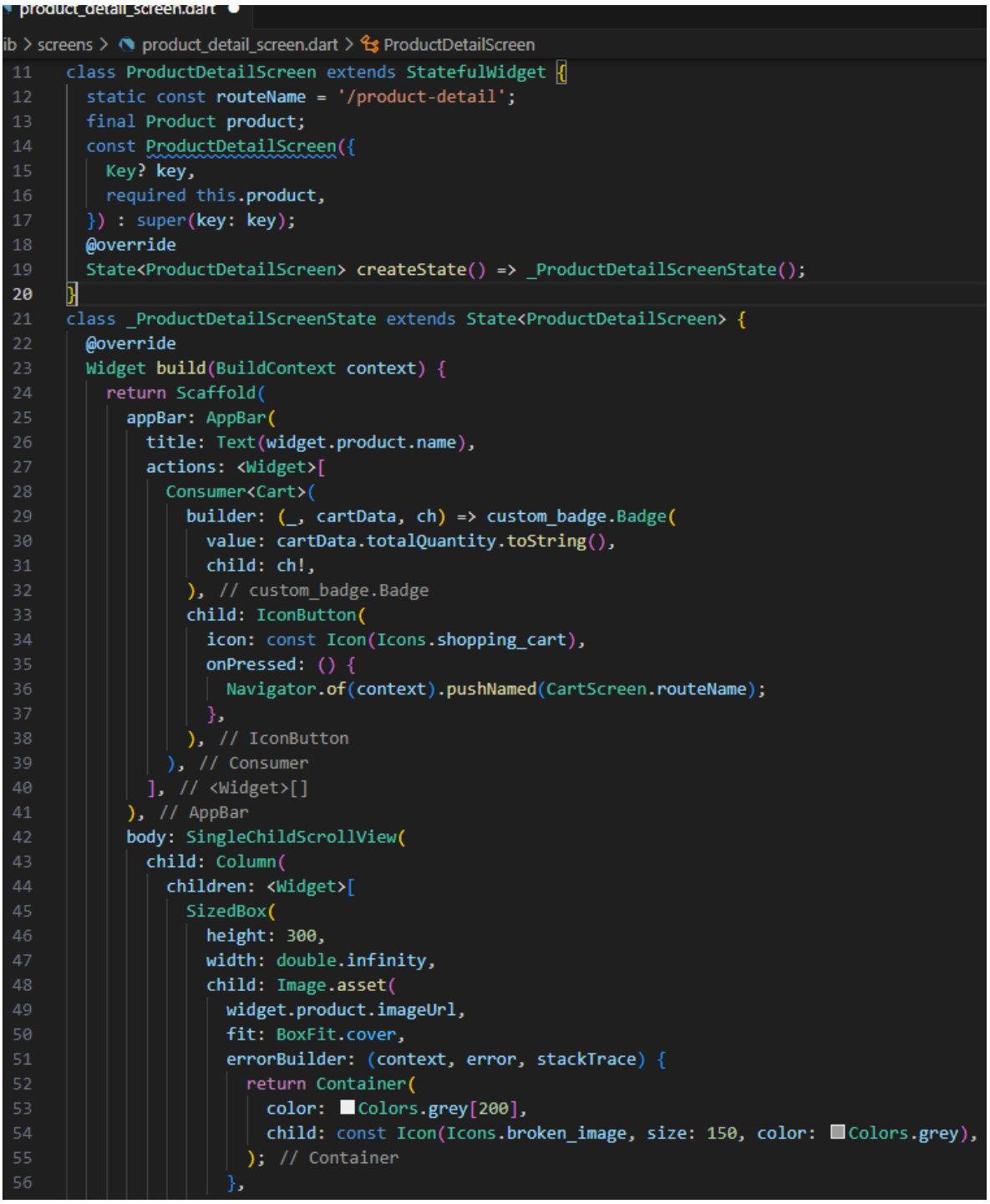


Рисунок 30 – Код реализации карточки товара

Экран корзины представляет собой главное место для управления покупками пользователя. Он содержит:

– список товаров: перечень всех добавленных в корзину товаров с их изображениями, названиями и ценами;

– общая сумма: в нижней части экрана отображена общая сумма всех товаров в корзине;
– кнопка оформления заказа: под списком товаров расположена крупная и заметная кнопка, которая позволяет пользователю перейти к процессу оплаты.

Это отображено на рисунке 29.

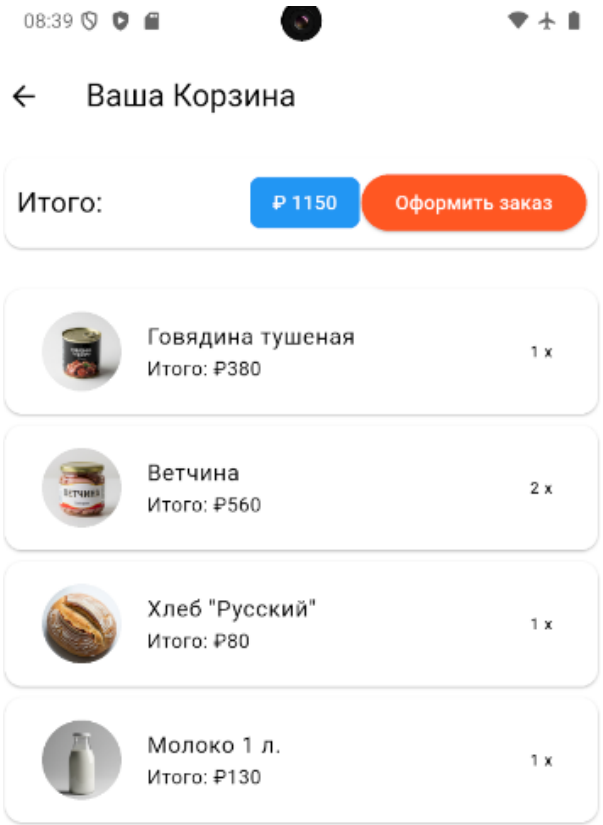


Рисунок 31 – Реализация экрана корзины

Код программы представлен на рисунке 32.

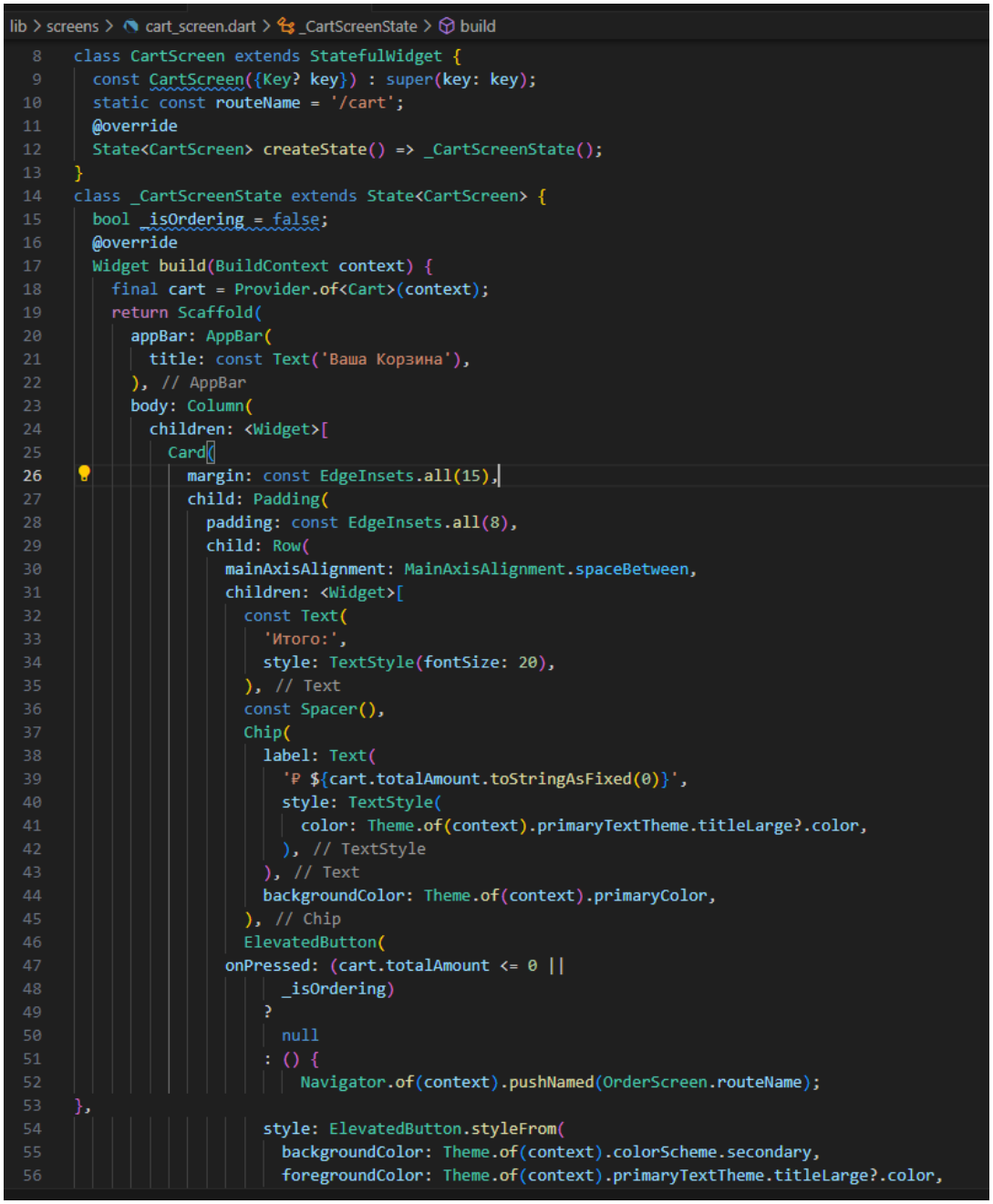


Рисунок 32 – Код реализации корзины

3.2 Тестирование мобильного приложения

Тестирование мобильного приложения проводилось с целью проверки соответсвия разработанного функционала установленным требованиям и

обеспечения его стабильной работы. В процессе тестирования были использованы следующие инструменты для анализа и выявления ошибок:

- Screen Mirroring Tools, иструмент, позволяющий отопражать экран тестируемого устройства на компьютере и управлять им, что существенно упрощает процесс наблюдения за поведением приложения и документирование найденных дефектов;
- Android Debug Bridge применялся для установки APK-файлов приложения на тестовые устройства, а так же для получения системных логов. Совместно с ADB использовались Logcat Readers/Viewers для удобного просмотра и фильтрации этих логов непостредственно на устройстве, что позволило оперативно выявлять ошибки и искючения.

Результаты тестирования представлены в таблице 9.

Таблица 9 – Результаты тестирования приложения

Аспект	Функция	Результат
Функиональность	Регистрация и авторизация	Протестированна регистрация и авторизация пользователей
	Карточка товара	Проверено правильное отображение карточек товаров
	Добавление товара в корзину	Проверено добавление товара из карточки в корзину
	Оформление заказа	Протестированно оформление заказа, ввод данных адреса доставки и оплата
	Обработка заказа	Проверен механизм получение данных о заказе для дальнейшей обработки
Удобство использования	Интерфейс приложения	Оценена интуитивность и навигация между экранами приложения
	Отзывчивость	Приложение протестировано на быстрые реации пользователя
	Визуализация	Проверена читабельность информации в приложении
Производительность	Стабильность	Приложение протестированно на работу при долгосрочном использовании
	Работа на различных система	Протестирована работа на различных устройствах
	Потребление ресурсов	Проверено потребление ресурсов на различных устройствах

Продолжение таблицы 9

Надежность	Персональные данные	Проверены правила механизма облачный базы данных, для предотвращения утечки данных
	Хранение паролей	Проверено хранение паролей в зашифрованном виде

Результаты тестирования приложения показали, что основные возможности приложения соответствуют требованиям и обеспечивают стабильную работу.

В третьей главе выпускной работы решены практические вопросы реализации мобильного приложения. Описаны модули системы (их задачи и функции), приведен код приложения, описан контрольный пример и проведено тестирование работы системы.

Заключение

В процессе выполнения выпускной работы автоматизированы процессы по покупки товаров через мобильное приложение.

В ходе выполнения работы были получены следующие результаты:

- рассмотрена деятельность компании ООО "Прима-Сервис";
- в методологии IDEF0 рассмотрены бизнес процессы, связанные с регистрацией, вводом данных и совершением покупки в мобильном приложении;
- построены схемы, отражающие процесс создания мобильного приложения;
- рассмотрены средства разработки мобильного приложения;
- создан прототип мобильного приложения;
- описана особенности создания пароля и его требования в мобильном приложении;
- проведено тестирование мобильного приложения.

Проделанная работа позволила не только успешно решить поставленную задачу по автоматизации процессов покупки товаров, но и создать основу для дальнейшего развития данного направления в ООО «Прима-Сервис».

Разработанное мобильное приложение является современным инструментом, который может существенно повысить конкурентоспособность компании на рынке.

Список используемой литературы и используемых источников

1. Балдин, К.В. Информационные системы в экономике / К.В. Балдин, В.Б. Уткин. – Москва : Издательско-торговая корпорация «Дашков и К, 2022. – 394 с. – ISBN 978-5-394-04783-1. – Текст : непосредственный.
2. Белайчук, А. А. Свод знаний по управлению бизнес-процессами: BPM CBOK 3.0 / А. А. Белайчук, В. Г. Елифёров. – Москва : ООО «Альпина Паблишер», 2016. – 680 с. – ISBN 978-5-9614-4208-3. – Текст : непосредственный.
3. Волк В. К. Введение в программную инженерию : учебное пособие / К.В. Волк. – Курган : Курганский государственный университет, 2018. – 155 с. – ISBN 978-5-4217-0452-2. – Текст : непосредственный.
4. Граничин, О.Н. Информационные технологии в управлении / О.Н. Граничин. – Москва : Бином, 2018. – 483 с. – ISBN 978-5-4497-2400-7. – Текст : непосредственный.
5. Дадян, Э. Г. Проектирование современных баз данных. Практикум: учебно-методическое пособие / Э. Г. Дадян. – Москва : НИЦ ИНФРА-М, 2017. – 84 с. – Текст : непосредственный.
6. Марка, Д. М. Методология структурного анализа и проектирования SADT Structured Analysis & Design Technique / Д.М. Марка, М.Г. Кремент. - Москва : Мета технология, 2008. – 243 с. – Текст : непосредственный.
7. Матяш, С.А. Корпоративные информационные системы : учебное пособие / С.А. Матяш. – М.-Берлин : Директ-Медиа, 2015. – 471 с. – ISBN: 978-5-4475-6085-0. – Текст : непосредственный.
8. Мкртычев, С.В. Прикладная информатика. Бакалаврская работа : электронное учебно-методическое пособие / С.В. Мкртычев, О.М. Гущина, А.В Очеповский. – Тольятти : ТГУ, 2019. – Текст : электронный.
9. Никитаева, А. Ю. Корпоративные информационные системы : учебное пособие / А.Ю. Никитаева, О.А. Чернова, М.Н Федосова. – Таганрог :

Южный федеральный университет, 2017. – 149 с. – ISBN 978-5-9275-2236-1.

10. Похилько, А.Ф. CASE-технология моделирования процессов с использованием средств BPWin и ERWin / А.Ф. Похилько, И.В. Горбачев. – Ульяновск : УлГТУ, 2008. – 120 с. – ISBN 978-5-9795-0398-1. – Текст : непосредственный.

11. Симдянов, И. В. Программирование. Ступени успешной карьеры / И.В. Симдянов. – СПб. : БХВ-Петербург, 2016. – 320 с. ISBN 5-94157-802-4. – Текст : непосредственный.

12. Ротер, М. Учитесь видеть бизнес-процессы. Практика построения карт потоков создания ценности / М. Ротер. – Москва : Альпина Паблишер, 2015. – 144 с. – ISBN 978-5-9614-3074-5. – Текст : непосредственный.

13. Хабр : [сайт]. – Москва, 2006 – . – URL: http://habrahabr.ru/ (дата обращения: 05.04.2025). – Текст. Изображение : электронные.

14. Bruce Silver. Bpmn Method and Style. A Levels-Based Methodology for Bpm Process Modeling and Improvement Using Bpmn 2.0 / B. Silver. – San Francisco : Cody-Cassidy Press, 2017. – 236 с. – Текст : непосредственный.

15. Hardware Inspector : [сайт]. – Улан-Удэ, 2003 – . – URL: https://www.hwinspector.com/ (дата обращения: 01.05.2025). – Текст : электронный. – Текст : непосредственный.

16. Jakob Freund. Real-Life BPMN: Using BPMN 2.0 to Analyze, Improve, and Automate Processes in Your Company / F. Jakob. – California : CreateSpace, 2016. – 232 с. – ISBN 978-1480034983. – Текст : непосредственный.

17. Joseph M. Hellerstein. Architecture of a Database System / H.M. Joseph, S. Michael, H. James. – California : Now Publisher Inc., 2020. – 119 с. – Текст : непосредственный.

18. Korotkevitch D. Pro SQL Server Internals / D. Korotkevitch. – New York : APress, 2019. – 804 с. – Текст : непосредственный.

19. Microsoft : [сайт]. – Redmond, 1991 – . – URL: https://office.com (дата обращения: 05.02.2025). – Текст. Изображение : электронные.

Приложение А

Файл sign_up_bloc.dart

```
import 'package:bloc/bloc.dart';
import 'package:equatable/equatable.dart';
import 'package:user_repository/user_repository.dart';

part
'sign_up_event.dar
t'; part
'sign_up_state.dar
t';

class SignUpBloc extends Bloc<SignUpEvent,
  SignUpState> { final UserRepository
  _userRepository;

  SignUpBloc(this._userRepository) :
    super(SignUpInitial()) {
    on<SignUpRequired>((event, emit) async {
      emit(SignUpPro
      cess()); try {
        MyUser myUser =
            await
_userRepository.signUp(event.user,
event.password);
        await
        _userRepository.setUserData(myUser
        ); emit(SignUpSuccess());
} catch (e) { emit(SignUpFailure());

      });
    }
}
```

Файл sign_in_screen.dart

```
import
'package:flutter/cupertino.dart'
; import
'package:flutter/material.dart';
import 'package:flutter bloc/flutter bloc.dart';

import
'../../../components/my_text_field.dart';
import
'../blocs/sign_in_bloc/sign_in_bloc.dart'
;

class SignInScreen extends
  StatefulWidget { const
  SignInScreen({super.key});

  @override
  State<SignInScreen> createState() =>  SignInScreenState();
}

class _SignInScreenState extends
  State<SignInScreen> { final
  passwordController =
  TextEditingController(); final
  emailController = TextEditingController();
  final _formKey = GlobalKey<FormState>();
  bool signInRequired = false;
  IconData iconPassword =
  CupertinoIcons.eye_fill; bool
  obscurePassword = true;
tring? _errorMsg;


  @override
  Widget build(BuildContext context) {
    return BlocListener<SignInBloc,
      SignInState>( listener: (context,
      state) {
        if (state is
          SignInSuccess) {
          setState(() {
            signInRequired = false;
          });
        } else if (state is
          SignInProcess) {
          setState(() {
            signInRequired = true;
          });
        } else if (state is
          SignInFailure) {
```

```
            setState(() {
                signInRequired = false;
                 errorMsg = 'Неправильная почта или пароль';
              });
            }
          },
      child: Form(
              key:
              _formK
              ey,
              child:
              Column
              (
                children: [
                  const
                  SizedBox(height:
                  20), SizedBox(
                      width: MediaQuery.of(context).size.width *
0.9,
                      child: MyTextField(
                          controller:
                          emailController,
                          hintText: 'Почта',
                          obscureText: false,
                          keyboardType:
                          TextInputType.emailAddress,
                          prefixIcon: const
Icon(CupertinoIcons.mail_solid),
                          errorMsg:
                          _errorMsg,
                          validator:
                          (val) {
                        if (val!.isEmpty) {
                              return 'Пожалуйста заполните поле';
                            } else if (!RegExp(r'^[\w-\.]+@([\w-
]+.)+[\w-]{2,4}$')
                                .hasMatch(val)) {
                              return 'Пожалуйста введите
                              правильную
почту';
                        }
                        return null;
                      })),
                  const
                  SizedBox(height:
                  10), SizedBox(
                    width:
                    MediaQuery.of(context).size.width *
                    0.9, child: MyTextField(
controller: passwordController, hintText: 'Пароль',


                      obscureText: obscurePassword,
```

```
                    keyboardType:
                    TextInputType.visiblePassword,
                    prefixIcon: const
Icon(CupertinoIcons.lock_fill),
                    errorMsg:
                    _errorMsg,
                    validator:
                    (val) {
                      if (val!.isEmpty) {
                        return 'Пожалуйста заполните поле';
                      } else if (!RegExp(
                            r'^(?=.*?[A-Z])(?=.*?[a-
                            z])(?=.*?[0-
9])(?=.*?[!@#\$&*~`)\%\-( +=;:,.<>/?"[{\]}\|^]).{8,}$')
                       .hasMatch(val)) {
                        return 'Пожалуйста введите пароль';
                      }
                      return null;
                    },
                    suffixIcon:
                      IconButton(
                      onPressed: ()
                      {
                        setState(() {
                          obscurePassword =
                          !obscurePassword; if
                          (obscurePassword) {
                         iconPassword =
CupertinoIcons.eye_fill;
                          } else {
                            iconPa
                            ssword
                            =
CupertinoIcons.eye_slash_fill;
                        }
                      });
                        },
                      icon: Icon(iconPassword),
                    ),
                  ),
                ),
            const SizedBox(height: 20),
                !signInRequired
                    ? SizedBox(
                      width:
                      MediaQuery.of(context).size.width *
0.5,
                      child:
                          TextBut
                          ton(
                          onPress
                          ed: ()
```

```
                            {
                                i
    f
    (_formKey.currentState!.valid
    ate()) {

    context.read<SignInBloc>().add(SignInRequired(
                                    emailController.te
                                    xt,
                                    passwordController
                                    .text));
                                }
                                },
style: TextButton.styleFrom( elevation: 3.0, backgroundColor:

    Theme.of(context).colorScheme.primary,
                                foregroundColor:
                                Colors.white, shape:
                                RoundedRectangleBorder
                                (
                                    borderRadius:
    BorderRadius.circular(60))),
                            child: const Padding(
                              padding:
                                  EdgeInsets.symmetric(
                                  horizontal: 25,
                                  vertical: 5),
                              ,
                                textAlign:
                                TextAlign.center,
                                style: TextStyle(
                                    color:
                                    Colors.white
                                    , fontSize:
                                    16,
                                    fontWeight:
                                    FontWeight.w600),
                                ),
                            )),
                        )
                    : const CircularProgressIndicator(),
                ],
            )),
        );
      }
    }
```

Файл sign_up_screen.dart

```
import
'package:flutter_bloc/flutter_bloc.dart';
import
'package:user_repository/user_repository.dart'
; import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import
'../../../components/my_text_field.dart';
import
'../blocs/sign_up_bloc/sign_up_bloc.dart'
;

class SignUpScreen extends
  StatefulWidget { const
  SignUpScreen({super.key});

  @override
  State<SignUpScreen> createState() =>  SignUpScreenState();
}

class _SignUpScreenState extends
  State<SignUpScreen> { final
  passwordController =
  TextEditingController(); final
  emailController = TextEditingController();
  final nameController =
  TextEditingController(); final _formKey =
  GlobalKey<FormState>();
  IconData iconPassword =
  CupertinoIcons.eye_fill; bool
  obscurePassword = true;
bool signUpRequired = false; bool containsUpperCase = false;


  bool containsLowerCase =
  false; bool
  containsNumber =  false;
  bool containsSpecialChar
  = false; bool
  contains8Length = false;

  @override
  Widget build(BuildContext context) {
    return BlocListener<SignUpBloc,
      SignUpState>( listener: (context,
      state) {
        if (state is
          SignUpSuccess) {
          setState(() {
            signUpRequired = false;
```

```
            });
          } else if (state is
            SignUpProcess) {
            setState(() {
              signUpRequired = true;
            });
          } else if (state is
            SignUpFailure) { return;
          }
        },
        child:
          Form(
          key:
          _formK
          ey,
          child:
          Center
          (
            child:
              Colu
              mn(
              chil
              dren
              : [
                const
                SizedBox(height:
                20), SizedBox(
                  width:
                  MediaQuery.of(context).size.width *
                  0.9, child: MyTextField(
                      controller:
                      emailController,
                      hintText: 'Почта',
                      obscureText: false,
                      keyboardType:
                      TextInputType.emailAddress,
                      prefixIcon: const
Icon(CupertinoIcons.mail solid),
                      validator: (val) {
                    if (val!.isEmpty) {
                        return 'Пожалуйста введите поле';
                      } else if (!RegExp(r'^[\w-\.]+@([\w-
]+.)+[\w-]{2,4}$')
                       .hasMatch(val)) {
                        return 'Пожалуйста введите правильный
email';
                    }
                    return null;
                      }),
                ),
                const
                SizedBox(height:
                10), SizedBox(
```

```
width: MediaQuery.of(context).size.width * 0.9, child:
MyTextField(

                        controller:
                        passwordController,
                        hintText: 'Пароль',
                        obscureText:
                        obscurePassword,
                        keyboardType:
                        TextInputType.visiblePassword,
                        prefixIcon: const
    Icon(CupertinoIcons.lock_fill),
                        onChanged: (val) {
                          if (val!.contains(RegExp(r'[A-
                            Z]'))) { setState(() {
                              containsUpperCase = true;
                        });
                          } else
                            {
                            setSt
                            ate((
                            ) {
                              containsUpperCase = false;
                        });
                          }
                          if (val.contains(RegExp(r'[a-
                            z]'))) { setState(() {
                              containsLowerCase = true;
                        });
                          } else
                            {
                            setSt
                            ate((
                            ) {
                              containsLowerCase = false;
                        });
                          }
                          if (val.contains(RegExp(r'[0-
                            9]'))) { setState(() {
                              containsNumber = true;
                        });
                          } else
                            {
                            setSt
                            ate((
                            ) {
                              containsNumber = false;
                        });
                          }
                          if
                              (val.contains(R
                              egExp(
```

```
                                r'^(?=.*?[!@#$&
                                *~`)\%\-
    (_+=;:,.<>/?"[{\]}\|^])'))) {
                                  setState(() {
                                    containsSpecialChar
                                    = true;
                                });
                                  } else
                                    {
                                    setSt
                                    ate((
                                    ) {
                                      containsSpecialChar = false;
                                });
                                  }
                                  if (val.length
                                    >= 8) {
                                    setState(()
                                    {
                                      contains8Length = true;
                                });
                                  } else
                                    {
                                    setSt
                                    ate((
                                    ) {
contains8Length = false;


                                  });
                                }
                                return null;
                                  },
                                  suffixIcon:
                                    IconButton(
                                    onPressed: ()
                                    {
                                      setState(() {
                                        obscurePassword =
                                        !obscurePassword; if
                                        (obscurePassword) {
                                       iconPassword =
    CupertinoIcons.eye fill;
                                        } else {
                                          iconPa
                                          ssword
                                          =
    CupertinoIcons.eye slash fill;
                                       }
                                    });
                                },
                                    icon: Icon(iconPassword),
                                  ),
```

```
                        validator: (val) {
                      if (val!.isEmpty) {
                            return 'Пожалуйста введите поле';
                      } else if (!RegExp(
                              r'^(?=.*?[A-Z])(?=.*?[a-
    z])(?=.*?[0-9])(?=.*?[!@#\$&*~`)\%\-
    (_+=;:,.<>/?"[{\]}\|^]).{8,}$')
                          .hasMatch(val)) {
                            return 'Пожалуйста введите правильный
    пароль';
                      }
                      return null;
                        }),
              ),
                  const
                  SizedBox(height:
                  10), Row(
                    mainAxisAl
    ignment:
    MainAxisAlignment.spaceEve
    nly,
                    crossAxisAlignment:
                    CrossAxisAlignment.start, children: [
                      Column(
                        crossAxisAl
                        ignment:
    CrossAxisAlignment.start,
                          (
                              "● 1
                            uppercase"
                             , style:
                            TextStyle(
                            color: containsUpperCase
                                    ? Colors.green
                                :
    Theme.of(context).colorScheme.onBackground),
                      ),
Text(


                              "● 1
                            lowercase"
                             , style:
                            TextStyle(
                                color: containsLowerCase
                                    ? Colors.green
                                :
    Theme.of(context).colorScheme.onBackground),
                      ),
                      Text(
                            "● 1
                            number",
```

```
                            style:
                            TextStyle
                            (
                             color: containsNumber
                                    ? Colors.green
                                    :
    Theme.of(context).colorScheme.onBackground),
                        ),
                          ],
                        ),
                        Column(
                          crossAxisAl
                          ignment:
    CrossAxisAlignment.start,
                           (
                            "●  1 special
                            character", style:
                            TextStyle(
                                color: containsSpecialChar
                                    ? Colors.green
                                    :
    Theme.of(context).colorScheme.onBackground),
                        ),
                        Text(
                              "●  8 minimum
                              character", style:
                              TextStyle(
                               color: contains8Length
                                      ? Colors.green
                                      :
    Theme.of(context).colorScheme.onBackground),
                        ),
                          ],
                        ),
                      ],
                    ),
                    const
                    SizedBox(height:
                    10), SizedBox(
                      width:
                      MediaQuery.of(context).size.width *
                      0.9, child: MyTextField(
                          controller:
                          nameController,
                          hintText: 'Имя',
                          obscureText: false,
                          keyboardType:
                          TextInputType.name,
                          prefixIcon: const
Icon(CupertinoIcons.person_fill),
```

```
                    validator: (val) {
                      if (val!.isEmpty) {
                        return 'Пожалуйста заполните поле';
                      } else if (val.length >
                        30) { return 'Имя
                        слишком длинное';
                    }
                    return null;
                      }),
                ),
                SizedBox(height:
MediaQuery.of(context).size.height
* 0.02),
                !signUpRequired
                ? SizedBox(
                        width:
                        MediaQuery.of(context).size.width *
0.5,
                        child:
                            TextBut
                            ton(
                            onPress
                            ed: ()
                            {
                              i
f
(_formKey.currentState!.valid
ate()) {
                                  MyUser myUser =
                                  MyUser.empty;
                                  myUser.email =
emailController.text;
                                  myUser.name =

                                  nameController.text;

                                  setState(() {

context.read<SignUpBloc>().add(SignUpRequired(
                                     myUser,
passwordController.text));
                                });
                              }
                            },
                            style:
                                TextButton.style
                                From( elevation:
                                3.0,
                                backgroundColor:

Theme.of(context).colorScheme.primary,
                                foregroundColor:
                                Colors.white, shape:
```

```
                                    RoundedRectangleBorder
                                    (
                                        borderRadius:
    BorderRadius.circular(60))),
                                child: const Padding(
                                  padding:
                                      EdgeInsets.symmetric(
                                      horizontal: 25,
                                      vertical: 5),
                                  child:
                                    Text(
                                    'Регис
                                    трация
                                    ',
                                    textAlign:
                                    TextAlign.center,
                                    style: TextStyle(
                                        color:
                                        Colors.white
                                        , fontSize:
                                        16,
                                        fontWeight:
                                        FontWeight.w600),
),

                                ),
                            )
                              : const CircularProgressIndicator()
                    ],
                      ),
                    ),
                  ),
                );
              }
            }
```

Файл sign_in_bloc.dart

```
import 'package:bloc/bloc.dart';
import 'package:equatable/equatable.dart';
import 'package:user repository/user repository.dart';

part
'sign_in_event.dar
t'; part
'sign_in_state.dar
t';

class SignInBloc extends Bloc<SignInEvent,
  SignInState> { final UserRepository
  _userRepository;

  SignInBloc(this._userRepository) :
    super(SignInInitial()) {
    on<SignInRequired>((event, emit) async {
      emit(SignInPro
      cess()); try {
        await
_userRepository.signIn(event.email,
event.password);
      } catch (e) {
        emit(SignInFai
        lure());
      }
    });

    on<SignOutRequired>((event, emit) async => await
_userRepository.logOut());
  }
}
```

Файл sign_in_event.dart

```
part of 'sign_in_bloc.dart';

sealed class SignInEvent extends
  Equatable { const SignInEvent();

  @override
  List<Object> get props => [];
}

class SignInRequired extends
  SignInEvent { final String
  email;
final String password;



  const SignInRequired(this.email, this.password);

  @override
  List<Object> get props => [email, password];
}

class SignOutRequired extends SignInEvent {}
```

Файл sign_in_state.dart

```
part of 'sign_in_bloc.dart';

sealed class SignInState extends
  Equatable { const SignInState();

  @override
  List<Object> get props => [];
}

final    class    SignInInitial    extends

SignInState {} final class SignInFailure

extends   SignInState   {}   final   class

SignInProcess   extends   SignInState   {}

final    class    SignInSuccess    extends

SignInState {}
```

Файл sign_up_event.dart

```
part of 'sign_up_bloc.dart';

sealed class SignUpEvent extends
  Equatable { const SignUpEvent();

  @override
  List<Object> get props => [];
}

class SignUpRequired extends
  SignUpEvent { final MyUser user;
  final String password;

  const SignUpRequired(this.user, this.password);

  @override
  List<Object> get props => [user, password];
}
```