

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

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

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

Направленность Программная инженерия

БАКАЛАВРСКАЯ РАБОТА

на тему

**Мобильное программное приложение для поддержки
процессов функционирования компьютерного салона**

Студент _____ Самсонов Никита Денисович
(подпись, дата) (ФИО полностью)

Руководитель _____ Князев В.Н.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой _____ Козлов А.Ю.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2026

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

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ А.Ю. Козлов «26»
декабря 2025 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент _____ Самсонов Никита Денисович _____ гр. 22ВП1 _____ факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы _____ Мобильное программное приложение для поддержки процессов
функционирования компьютерного салона _____

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20 _____

3. Руководитель работы к.т.н., доцент кафедры «МОиПЭВМ» Князев В.Н. _____

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

- 1) *Разработать программное обеспечение для поддержки процессов функционирования компьютерного салона в форме мобильного приложения.*
- 2) *Среда выполнения приложения: операционная система Android 8.0 и выше*
Среда выполнения сервера: операционная система Windows 10 и выше
Языковые средства разработки: C#, Kotlin, SQL, UML, LINQ
Инструментальные средства разработки: Visual Studio Code, Android Studio, pgAdmin4, Visual Paradigm, Microsoft Project 2007
Технологии разработки: Agile, Kanban, ООП, MVC, MVVM, REST
- 3) *Основные функции мобильного приложения:*
 - а) *визуализация товаров каталога компьютерного салона и его фильтрация;*
 - б) *осуществление возможности совершения покупок компьютеров;*
 - в) *возможность отправки вопросов в техническую поддержку приложения;*
 - г) *написание отзывов на компьютеры и их оценивание;*
 - д) *создание новых товаров, их изменения и удаление;*
 - е) *получение сводки о продажах за указанный период времени.*

5. Перечень подлежащих разработке вопросов

1) Анализ предметной области и требований к системе

2) Проектирование

3) Реализация

4) Контроль качества программного обеспечения

5) Оформление пояснительной записки

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	07.04.2026 – 13.04.2026	
2	Проектирование программных средств	20%	14.04.2026 – 25.04.2026	
3	Реализация программных средств	30%	27.04.2026 – 11.05.2026	
4	Тестирование программных средств	20%	12.05.2026 – 21.05.2026	
5	Оформление пояснительной записки	20%	22.05.2026 – 02.06.2026	

Дата выдачи задания «26» декабря 2025г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

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

Реферат

Пояснительная записка содержит 144 листа, 59 рисунков, 26 таблиц, 47 использованных источников, 5 приложений.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, КЛИЕНТ-СЕРВЕРНАЯ АРХИТЕКТУРА, КОМПЬЮТЕРНЫЙ САЛОН, ANDROID, KOTLIN, C#, ASP.NET, MVC, MVVM, POSTGRESQL

Объектом разработки является мобильное приложение для пользования и управления салоном по продаже компьютеров.

Цель работы – разработка мобильного программного приложения с функциональными возможностями управления и просмотра каталога, оформления заказов, написания отзывов, задавать вопросы в поддержку, анализа статистики и разработка серверной составляющей салона по продаже персональных компьютеров.

Разработка серверной часть проводилась на языке программирования C# с использованием фреймворка ASP.NET в редакторе кода Visual Studio Code. В роли СУБД использовалась PostgreSQL. Серверная часть предназначена для работы под управлением операционной системы Windows 10 и выше.

Разработка мобильного приложения осуществлялась на языке программирования Kotlin при помощи набора инструментов Jetpack Compose под мобильную операционную систему Android минимальной версии 8.0 и выше в среде Android Studio.

					ПГУ 09.03.04 – 08БР221.18 ПЗ			
Изм.	Лист	№ докум.	Подп.	Дата	Мобильное программное приложение для поддержки процессов функционирования компьютерного салона. Пояснительная записка	Лит.	Лист	Листов
Разраб.		Самсонов Н.Д.					4	144
Пров.		Князев В.Н.						
Н. контр.		Такташкин Д.В.				Группа 22ВП1		
Утверд.		Козлов А.Ю.						

Содержание

Введение.....	7
1 Анализ предметной области	9
1.1 Анализ предметной области и постановка задачи	9
1.2 Анализ программных продуктов	10
1.2.1 Интернет-магазин «HYPERPC»	10
1.2.2 Интернет-магазин «Delta Game»	12
1.2.3 Интернет-магазин «COMPSHOP»	13
1.2.4 Сравнение аналогов	14
1.3 Анализ целевой аудитории.....	14
1.4 Анализ функциональных требований	17
1.5 Анализ нефункциональных требований	24
1.6 Планирование разработки и оценка бюджета	25
2 Проектирование мобильного приложения	47
2.1 Выбор технологий и инструментальных средств	47
2.2 Проектирование архитектуры программного обеспечения.....	48
2.3 Проектирование базы данных.....	50
2.3.1 Концептуальное проектирование базы данных	50
2.3.2 Логическое проектирование базы данных	52
2.4 Проектирование серверной части приложения	53
2.5 Проектирование пользовательского интерфейса.....	54
3 Реализация программного обеспечения	72
3.1 Реализация базы данных	72
3.2 Реализация серверной части приложения	77
3.3 Реализация клиентской части приложения	82
4 Тестирование программного продукта	87
4.1 Функциональное тестирование черным ящиком.....	87
4.2 Модульное тестирование.....	88
4.3 Метрики кода.....	90

4.3.1	Метрика исходных строк кода.....	91
4.3.2	Метрики Холстеда	91
	Заключение	95
	Список использованных источников	97
	Приложение А (справочное) Глоссарий	102
	Приложение Б (обязательное) Исходный код серверной части.....	106
	Приложение В (обязательное) Исходный код мобильного приложения	125
	Приложение Г (рекомендуемое) Результаты тестирования серверной части ..	133
	Приложение Д (рекомендуемое) Исходный код класса тестирования	137

Введение

На текущий момент времени информационные технологии глубоко внедрились в большинство сфер деятельности человека и тесно интегрированы в его повседневную жизнь. Современному человеку крайне трудно представить свою жизнь без пользования какими-либо информационными или коммуникационными интернет-ресурсами. Доступ к информационным технологиям обеспечивает глобализацию благодаря свободному распространению информации и общению людей вне зависимости от расстояния [1].

В настоящее время развитие цифровых технологий в Российской Федерации напрямую связано с расширением возможностей мобильной связи. Данные направления закреплены в программе «Цифровая экономика Российской Федерации», утвержденной распоряжением Правительства РФ от 28 июля 2017 года № 1632-р [2]. В документе подчеркивается, что развитие цифровой экономики требует создания условий, при которых «данные в цифровой форме являются ключевым фактором производства во всех сферах социально-экономической деятельности».

Особое внимание также уделяется развитию информационной инфраструктуры. В частности, одним из ключевых направлений обозначено создание таких условий, при которых «сети связи удовлетворяют потребности экономики по сбору и передаче данных граждан, бизнеса и власти с учетом технических требований, предъявляемых цифровыми технологиями». Данное положение отражает необходимость дальнейшего развития мобильных сетей передачи данных и увеличения скорости доступа к интернету.

Кроме того, программа предусматривает внедрение современных цифровых технологий и развитие перспективных средств коммуникации. Среди сквозных технологий, определяющих развитие цифровой экономики, отдельно выделяются «технологии беспроводной связи», являющиеся основой для

совершенствования мобильных устройств, мобильного интернета и сетей нового поколения.

Самым популярным средством выхода в сеть Интернет является мобильное устройство. Благодаря его портативности у пользователя есть возможность практически моментально выполнить поисковый запрос или выполнить подключение к интересующему его ресурсу. По состоянию на 2024 год в России в сеть выходило 92,5% населения, причем 88,2% из них осуществляли подключение с мобильных устройств [3]. Это говорит о развитости инфраструктуры мобильной связи и предпочтительности мобильного устройства.

Таким образом, при помощи настольного компьютера пользователь может работать с громоздкими приложениями, требующими полной отдачи, а с помощью мобильного телефона можно выполнять операции управления из любого места при наличии мобильной связи. Например, клиенту салона по продаже компьютеров удобно просматривать каталог, оставлять отзыв с мобильного устройства, администратор может добавить новый товар или изменить существующий вне зависимости от его местоположения, модератор имеет способ быстро отреагировать на неподходящий отзыв и удалить его.

Ссылаясь на вышеперечисленные преимущества мобильных устройств связи как способ быстрого реагирования на ситуацию и удобного пользования ресурсами, а персональных компьютеров как полноценных рабочих станций, можно сделать вывод о высокой степени актуальности проекта разработки мобильного приложения для салона по продаже персональных компьютеров.

1 Анализ предметной области

1.1 Анализ предметной области и постановка задачи

Разрабатываемое мобильное приложение компьютерного салона должно обеспечивать всем категориям пользователей комфортный пользовательский опыт, в который входят быстрое действие приложения, четкие границы интерфейса с понятным описанием функциональных элементов пользовательского интерфейса и весь базовый набор необходимых функций приложения.

В набор функций гостя (незарегистрированного пользователя) необходимо включить возможность продвинутых возможностей использования каталога: фильтрация по возможным видам комплектующих, таких как процессор, видеокарта, оперативная память, сортировка по популярности и цене, вывод соответствий по данным, введенным в поле поиска. Также гость должен иметь возможность просмотреть карточку товара с описанием его характеристик, цены и отзывов и обращаться в техническую поддержку.

Таким образом, приложение должно предоставлять следующие основные функции группам пользователей:

- регистрация пользователя;
- авторизация пользователя;
- просмотр каталога;
- фильтрация каталога;
- сортировка каталога;
- использование поля поиска;
- просмотр карточки товаров;
- просмотр отзывов;
- обращение в техническую поддержку;
- оформление заказа;
- написание отзывов;

- просмотр истории покупок;
- добавление, изменение, удаление товара.

Вышеописанные функции также включают остальные группы пользователей. Помимо гостя в приложении должна быть предусмотрена регистрация и авторизация пользователей, что позволит открыть доступ к оформлению заказов на персональные компьютеры (ПК), написанию отзывов и истории покупок.

Персонал компьютерного салона должен быть поделен на две роли: менеджер по работе с клиентами и администратор. Менеджер занимается закрытием заявок в техническую поддержку и фильтрацией пользовательского контента в приложении, а администратор имеет возможности добавления, изменения и удаления карточек товара и получения информации о совершенных продажах.

1.2 Анализ программных продуктов

На текущий год существует множество магазинов, занимающихся сборкой и продажей в основном игровых персональных компьютеров. Для сравнительного анализа аналогов были выбраны три наиболее популярных из них.

1.2.1 Интернет-магазин «HYPERPC»

Основанный в 2010-м году, на данный момент являются самым популярным производителем готовых сборок персональных компьютеров с большим количеством коллабораций контент-блоггеров по странам СНГ. На главной странице магазина представлены промо-материалы компании и несколько карточек товара с компьютерами. Ниже на рисунке 1 представлен фрагмент главной страницы магазина [4].

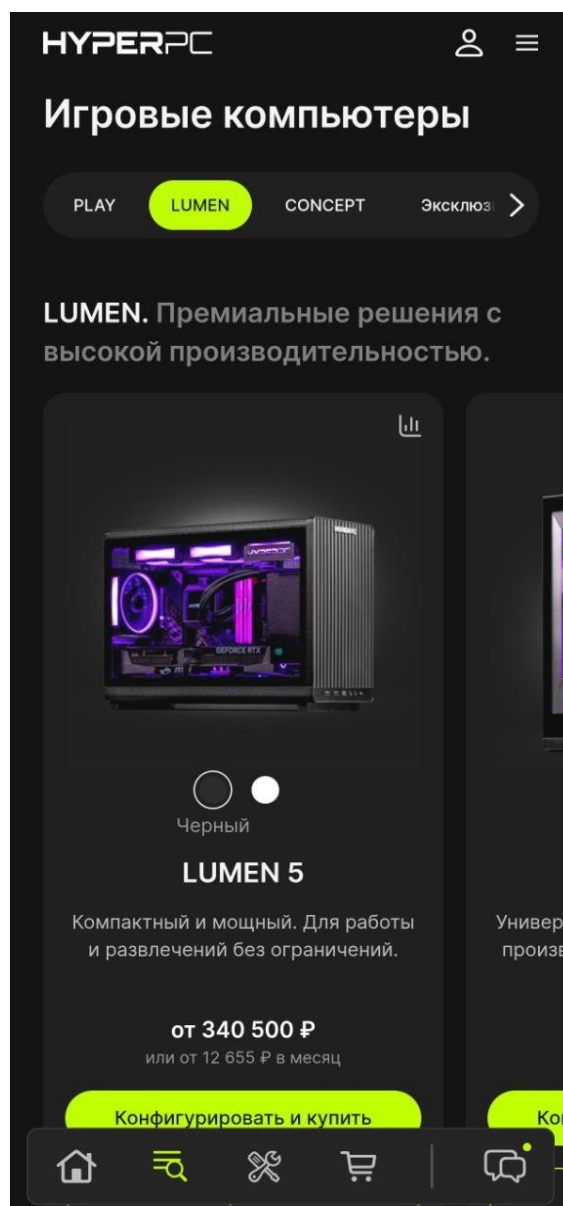


Рисунок 1 – Главная страница интернет-магазина HYPERPC

Среди достоинств сайта магазина можно выделить наличие технической поддержки, большой выбор ПК, продвинутые конфигураторы по комплектующим, рабочим нагрузкам, играм и программам.

Среди недостатков следует выделить перегруженный интерфейс, который создает негативный пользовательский опыт и затрудняет навигацию, а также ограниченная фильтрация и поиск в связи с продажей ПК премиум сегмента. Помимо этого, у магазина нет собственного мобильного приложения.

1.2.2 Интернет-магазин «Delta Game»

Компания была основана в 2002-м году и занимается сборкой ПК начиная средним и заканчивая премиум сегментом. Также на сайте компании присутствует раздел статей, в которых описываются конфигурации ПК специально под игровые новинки. На главной странице представлен перечень категорий товаров магазина, а фрагмент карточки товара предоставлен ниже на рисунке 2 [5].

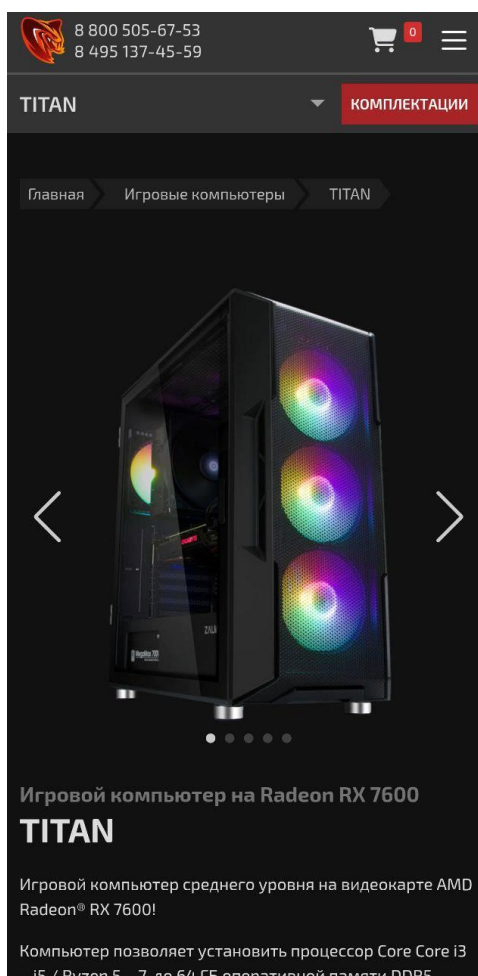


Рисунок 2 – Карточка товара магазина «Delta Game»

Достоинством магазина можно счесть удобно просматриваемый каталог и возможности фильтрации по цене, разрешению, видеокарте и процессору.

Недостатками можно счесть слабо разделенный на функциональные блоки интерфейс, отсутствие мобильного приложения и неудобную навигацию по сайту.

1.2.3 Интернет-магазин «COMPSHOP»

Компания «COMPSHOP» работает с 2016 года и предоставляет свои услуги на всей территории Российской Федерации. Переходя на сайт, пользователь попадает на страницу с фирменными ПК, формой для вопросов и разбором готовых проектов. Ниже на рисунке 3 представлена главная страница магазина [6].

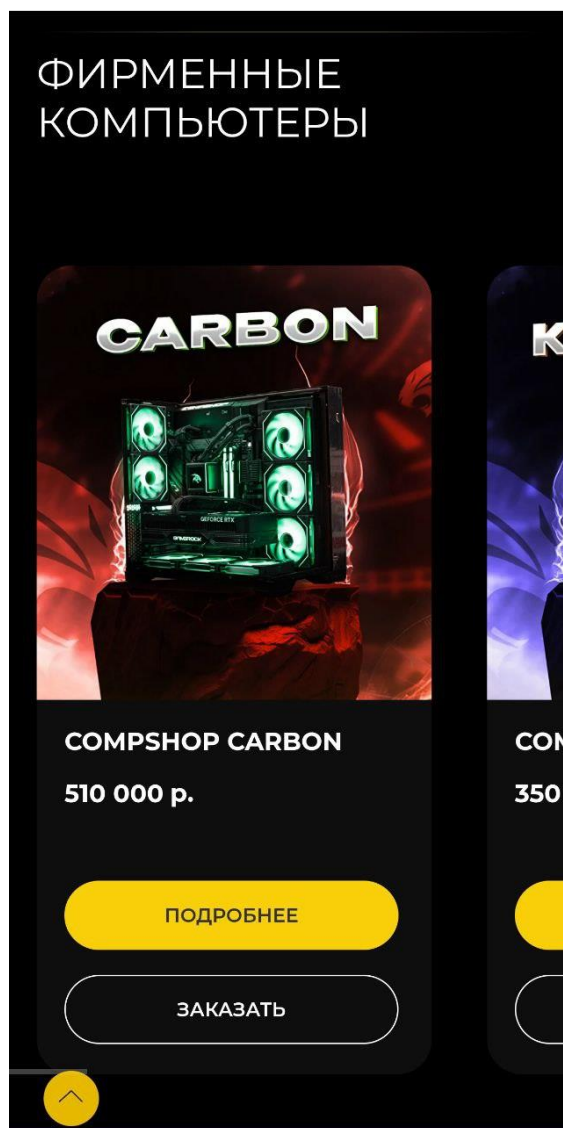


Рисунок 3 – Фрагмент главной страницы магазина «COMPSHOP»

К достоинствам сайта относятся понятный интерфейс и удобное отображение товаров с показом уже готовых сборок компьютеров.

К недостаткам относятся отсутствие фильтрации и сортировки каталога, отсутствие мобильного приложения и перегруженные информацией карточки товара.

1.2.4 Сравнение аналогов

При рассмотрении аналогов был проведен их сравнительный анализ с целью категоризации их главных преимуществ и недостатков. Результаты данного анализа представлены в таблице 1.

Таблица 1 – Сравнительный анализ аналогов

Название магазина	Наличие мобильного приложения	Удобство навигации	Удобство фильтрации	Удобство просмотра сайта/приложения	Наличие техподдержки
HYPERPC	Нет	Нет	Нет	Нет	Да
Delta Game	Нет	Нет	Да	Нет	Нет
COMPSHOP	Нет	Да	Нет	Да	Да
Разрабатываемое приложение	Да	Да	Да	Да	Да

Исходя из результатов проведенного анализа, можно сделать вывод о том, что главным преимуществом разрабатываемого продукта будет являться наличие мобильного приложения и отсутствие недостатков аналогов.

1.3 Анализ целевой аудитории

Для магазина по продаже компьютеров целевой аудиторией являются сотрудники магазина и заинтересованные в покупке ПК лица – компании, компьютерные клубы, люди, работающие с профессиональным программным обеспечением, увлекающиеся видеоиграми.

Согласно статистике tadviser.ru, за 2024 год рынок настольных компьютеров вырос на 22% и продолжил расти в первом квартале 2025 года, а

объем продаж составил 71 миллиард рублей, что на 10% выше, чем в предыдущем году [7].

Помимо этого, в 2025-м году за последние пять лет текущий оборот в компьютерных клубах вырос почти в 40 раз, сообщает ТАСС [8]. Это прямое следствие популярности компьютерных клубов, что означает большой спрос на закупки от магазинов готовых сборок ПК.

На основе вышеприведенных данных можно сделать заключение о том, что спрос на готовые компьютеры в России продолжает увеличиваться и имеет значительный спрос у заинтересованных лиц.

Далее рассмотрим рынок мобильных операционных систем по состоянию на январь 2026 года. Ниже представлена статистика мирового рынка, которую можно увидеть на рисунке 4 [9].



Рисунок 4 – Мировой рынок мобильных операционных систем в январе 2026 года

Исходя из данных по мировому рынку мобильных ОС, с большим отрывом лидирует ОС Android, занимая почти 70% рынка. Ниже на рисунке 5 представлена статистика долей операционных систем по России [10].



Рисунок 5 – Российский рынок мобильных операционных систем на январь 2026 года

Помимо данных операционных систем, на российском рынке появляются собственные мобильные операционные системы, нацеленные на импортозамещение в государственном сегменте [11]. Рассмотрим некоторые из них ниже:

- «ALT Mobile» – нацеленная на корпоративный сектор, имеет в качестве ядра Linux и репозиторий Sisyphus, что предоставляет пакет программ для работы «из коробки», а для устройств компаний предусмотрено шифрование данных и возможность удаленного управления;
- «Astra Linux Mobile» – представленная в 2022 году, на данный момент совместимо с подавляющим большинством планшетов Минпромторга России, имеет возможность переключения мобильного и настольного интерфейсов «на лету», эмуляции Android приложений и оснащенность необходимым уровнем защиты для рабочих данных;
- «Аврора» – разрабатываемая с 2016 года, операционная система имеет поддержку криптографической защиты информации и нативный менеджер пакетов для удобного управления приложениями и возможностями интеграции корпоративных политик безопасности.

Из рисунка видно, что на российском рынке мобильная ОС Android занимает две трети рынка, что говорит о необходимости ориентироваться на данную платформу в первую очередь. Для этого определим наиболее популярные версии операционной системы, чтобы определиться с минимальной

версией Android. Распределение версий Android на 1 декабря 2025 года представлено ниже на рисунке 6 [12].

Platform Version	API	Distribution
Android 5 (Lollipop)	21	0.1%
Android 5.1 (Lollipop)	22	0.3%
Android 6 (Marshmallow)	23	0.4%
Android 7 (Nougat)	24	0.4%
Android 7.1 (Nougat)	25	0.4%
Android 8 (Oreo)	26	0.8%
Android 8.1 (Oreo)	27	2.3%
Android 9 (Pie)	28	4.5%
Android 10 (Q)	29	7.8%
Android 11 (R)	30	13.7%
Android 12 (S)	31	11.4%
Android 13 (T)	33	13.9%
Android 14 (U)	34	17.2%
Android 15 (V)	35	19.3%
Android 16 (B)	36	7.4%

Рисунок 6 – Распределение версий ОС Android

На рисунке видно, что на данный момент на рынке преобладают смартфоны с Android версии 15, а общее число устройств до версии 8.0 составляет менее двух процентов. Согласно документации Google, новые приложения должны ориентироваться на Android 15 для публикации в магазине Google Play [13]. На основе этой информации остановим свой выбор на минимальной версии Android 8.0 или Software Development Kit (SDK) версии 26, а ориентацию на Android 15.

1.4 Анализ функциональных требований

В качестве основных ролей в приложении выделяются гость, зарегистрированный пользователь, менеджер по работе с клиентами и администратор магазина. Гостю доступны базовые функции приложения, такие как просмотр каталога и применение различных видов фильтрации и сортировки по отношению к нему, а также использование поля поиска для нахождения

точных соответствий. Помимо этого, гость может задать интересующий его вопрос через форму связи и просмотреть карточку товара с его отзывами.

Если гость будет заинтересован в товаре и захочет его приобрести, то с помощью опции регистрации он станет зарегистрированным пользователем и получит возможность оформления заказа на имеющийся в наличии персональный компьютер. После покупки ему также становится доступна опция написания отзыва на купленный ПК и просмотр историю покупок.

Контроль площадки выполняется ролью менеджера – он имеет возможность удаления отзывов, несоответствующих правилам магазина, а также отвечает на вопросы, заданные через форму. При необходимости он также может просмотреть историю заданных вопросов.

Администратор магазина наследует функционал всех ролей приложения и дополнительно имеет возможность управления товарами магазина: добавление, изменение, удаление компьютеров, а также при необходимости получить сводку о совершенных продажах за какой-либо период времени.

Таким образом, приложение реализует систему доступа к функционалу на основе ролей, что позволяет сразу настраивать возможности пользователя, исходя из его прав доступа на те или иные функции. Например, у зарегистрированного пользователя в приложении будет отсутствовать вкладка с администрированием, а вкладка поддержки будет отличаться от той, которую видит менеджер по работе с клиентами, несмотря на то что все пользователи используют идентичное мобильное приложение.

Диаграмма вариантов использования, отражающая вышеперечисленные возможности функционала, представлена ниже на рисунке 7.



Рисунок 7 – Диаграмма вариантов использования мобильного приложения

На диаграмме выделены основные функции приложения, каждая из которых относится к тому или иному типу пользователя. Стоит отметить, что администратор салона по продаже ПК наследует функционал всех остальных ролей, а актор СУБД связан со всеми вариантами использования, так как каждый запрос будет направлять и запрашивать данные из базы данных.

На основе выделенных вариантов использования составим их спецификации, отражающие основные свойства каждого варианта использования. Ниже на рисунке 8 представлена спецификация варианта использования «Задать вопрос» гостя.

Наименование: <u>Задать</u> вопрос
ID: 9
Краткое описание: Гость заполняет форму вопроса и отправляет его менеджерам по работе с клиентами
Действующие лица: Гость, СУБД
Предусловие: СУБД и сервер программного обеспечение в рабочем состоянии, клиентское мобильное приложение запущено
Основной поток: <ol style="list-style-type: none"> 1. Гость переходит на экран «Поддержка» 2. Гость нажимает кнопку «Задать вопрос» 3. Гость заполняет форму вопроса: заголовок и тело вопроса 4. Нажимает на кнопку «Отправить» 5. Если форма заполнена корректно <ol style="list-style-type: none"> 5.1. СУБД добавляет вопрос в список открытых вопросов 5.2. Переход на экран «Поддержка» 6. Иначе: вывод сообщения о некорректно заполненной форме
Постусловие: вопрос пользователя добавлен в базу данных и доступен для менеджеров

Рисунок 8 – Спецификация варианта использования «Задать вопрос»

Ниже на рисунке 9 представлена спецификация варианта использования «Добавить компьютер в каталог» администратора.

Наименование: <u>Добавить</u> компьютер в каталог
ID: 16
Краткое описание: Администратор заполняет форму добавления ПК в каталог
Действующие лица: Администратор, СУБД
Предусловие: СУБД и сервер программного обеспечение в рабочем состоянии, клиентское мобильное приложение запущено
Основной поток: <ol style="list-style-type: none"> 1. Администратор переходит в панель администрирования 2. Выбирает «Добавить новый ПК» 3. Заполняет характеристики компьютера 4. Добавляет фото компьютера 5. Если форма заполнена корректно <ol style="list-style-type: none"> 5.1. СУБД добавляет в список товаров новый ПК 5.2. Переход на экран «Профиль» <p>Иначе: вывод сообщения о некорректно заполненной форме</p>
Постусловие: новый компьютер доступен к покупке в каталоге пользователей

Рисунок 9 – Спецификация варианта использования «Добавить компьютер в каталог»

Ниже на рисунке 10 представлена спецификация варианта использования «Оформить заказ на компьютер» зарегистрированного пользователя.

Наименование: <u>Написать</u> отзыв на компьютер
ID: 11
Краткое описание: Зарегистрированный пользователь заполняет форму отзыва на ПК
Действующие лица: Зарегистрированный пользователь, СУБД
Предусловие: СУБД и сервер программного обеспечение в рабочем состоянии, клиентское мобильное приложение запущено
Основной поток: <ol style="list-style-type: none"> 1. Пользователь переходит на экран «Каталог» 2. Пользователь выбирает карточку товара 3. Пользователь нажимает кнопку «Оставить отзыв» 4. Заполняет форму отзыва 5. Нажимает на кнопку «Отправить» 6. Если форма заполнена корректно <ol style="list-style-type: none"> 6.1. СУБД добавляет отзыв к компьютеру 6.2. Переход на экран «Каталог» 7. Иначе: вывод сообщения о некорректно заполненной форме
Постусловие: отзыв пользователя добавлен к товару, виден остальным покупателям

Рисунок 10 – Спецификация варианта использования «Написать отзыв на компьютер»

Ниже на рисунке 11 представлена спецификация варианта использования «Ответить на вопрос» менеджера по работе с клиентами.

Наименование: <u>Ответить</u> на вопрос
ID: 9
Краткое описание: Менеджер по работе с клиентами отвечает на открытый вопрос пользователя
Действующие лица: Менеджер по работе с клиентами, СУБД
Предусловие: СУБД и сервер программного обеспечение в рабочем состоянии, клиентское мобильное приложение запущено
Основной поток: <ol style="list-style-type: none"> 1. Менеджер переходит на экран «Поддержка» 2. Менеджер нажимает кнопку «Открытые вопросы» 3. Менеджер выбирает вопрос из списка открытых 4. Заполняет форму ответа 5. Нажимает на кнопку «Отправить» 6. Если форма заполнена корректно <ol style="list-style-type: none"> 6.1. СУБД добавляет ответ на вопрос в список сообщений 6.2. Переход на экран «Поддержка» 7. Иначе: вывод сообщения о некорректно заполненной форме
Постусловие: ответ менеджера доступен к просмотру определенным пользователем

Рисунок 11 – Спецификация варианта использования «Ответить на вопрос»

Помимо спецификаций для вышеперечисленных вариантов использования, смоделируем их поведение. Для вариантов использования «Задать вопрос» и «Добавить компьютер в каталог» разработаем диаграммы деятельности, показывающие рабочий поток их обработки. Они представлены ниже на рисунках 12 и 13.

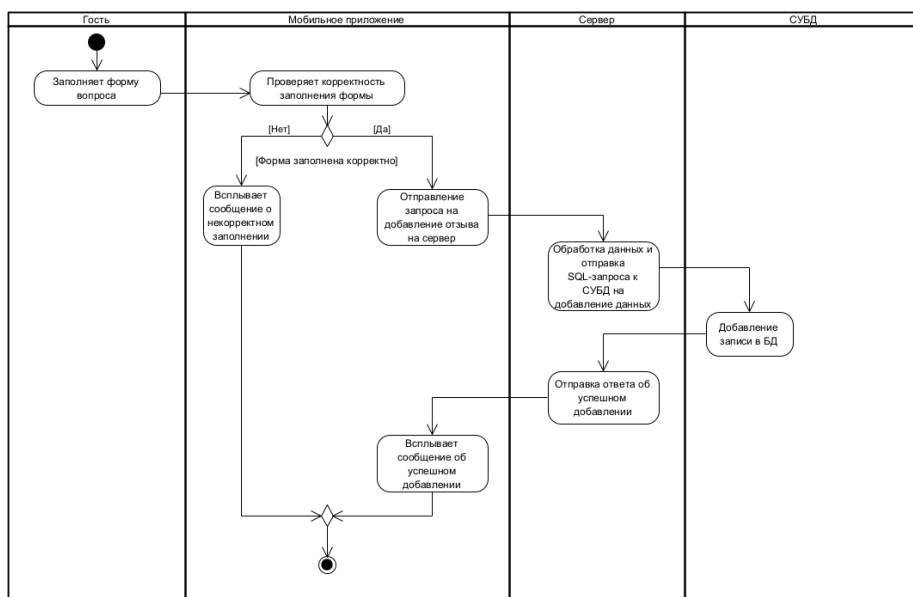


Рисунок 12 – Диаграмма деятельности варианта использования «Задать вопрос»

Данная диаграмма показывает последовательность обработки запроса вопроса пользователя, которая делит его на этап клиентской обработки корректности и непосредственной отправки на сервер.

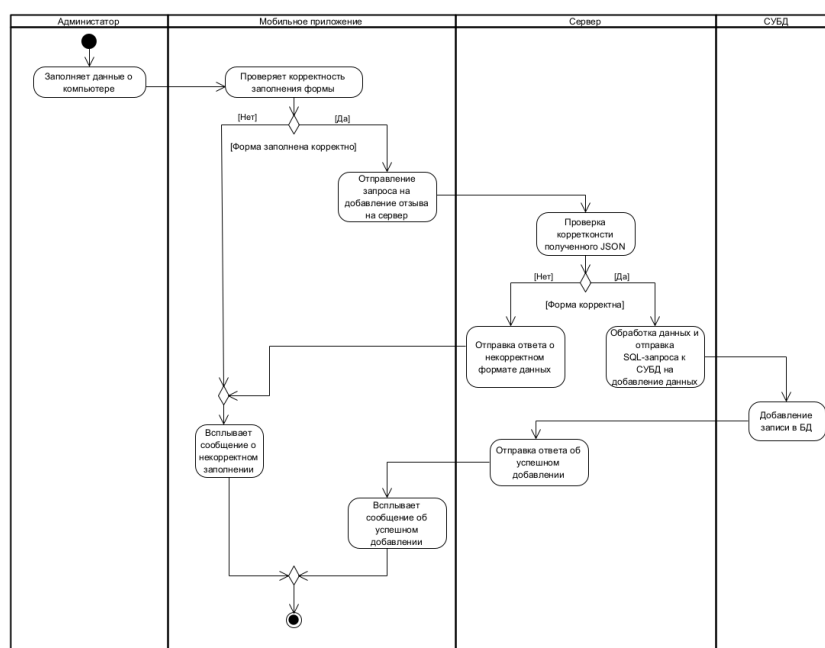


Рисунок 13 – Диаграмма деятельности «Добавить компьютер в каталог»

Для вариантов использования «Написать отзыв на компьютер» и «Ответить на вопрос» были разработаны диаграммы последовательности. Они представлены ниже на рисунках 14-15.

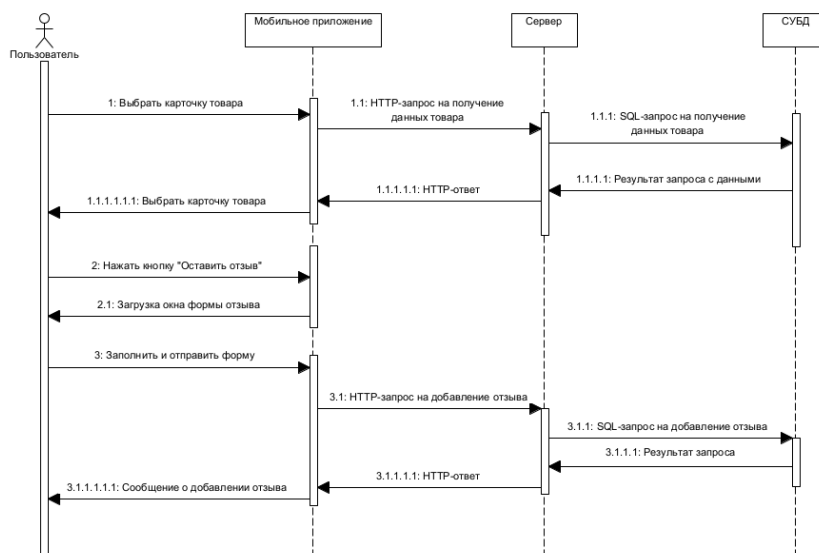


Рисунок 14 – Диаграмма последовательности варианта использования
«Написать отзыв на ПК»

Данная диаграмма показывает последовательную логику обработку запроса пользователя на написание отзыва, где мобильное приложение, сервер и СУБД последовательно получают и отправляют обработанные данные и действия.

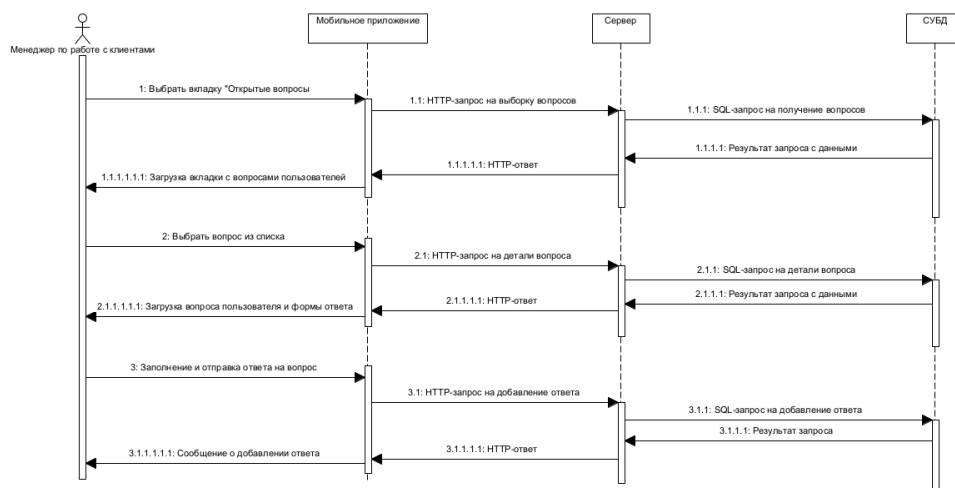


Рисунок 15 – Диаграмма последовательности варианта использования
«Ответить на вопрос»

Таким образом, был проанализирован необходимый функционал разрабатываемого программного обеспечения и смоделирован поток работы его определенных вариантов использования.

1.5 Анализ нефункциональных требований

На этапе анализа нефункциональных требований были определены основная методология разработки программного обеспечения, а также выделены архитектурные шаблоны проектирования серверной и клиентской частей.

В качестве методологии был выбран Agile, отвечающий принципам коммуникации между заказчиком и исполнителем, уменьшением необходимой документации в пользу списка задач с приоритетом заказчика, быстрым внесением правок в проект. В рамках самого Agile был выбран метод Kanban, предполагающая визуализацию процесса реализации задач через электронную доску с отображением их текущего состояния и ограничением количества задач, находящихся в работе.

Серверная часть приложения будет разрабатываться с использованием эффективного языка программирования C# и платформы ASP.NET, предназначенной для разработки серверной части веб-приложений. В качестве архитектурного шаблона серверной части была выбрана REST архитектура, что обеспечит простоту интеграции и обмена данными через JSON. Разработка будет проводиться в редакторе кода Microsoft Visual Studio Code и должна разворачиваться под управлением ОС Windows не старше 10-й версии.

Роль СУБД будет исполнять современная PostgreSQL версии 18.2, а интеграция с сервером будет выполняться с помощью инструмента Entity Framework Core.

Шаблоном проектирования клиентской части будет выступать MVVM (Model-View-ViewModel), что позволит разделить логику и интерфейс и улучшить тестируемость, а также хорошо работает в связке с мобильным приложением. Само мобильное приложение будет разрабатываться с использованием языка Kotlin при помощи набора инструментов Jetpack Compose в среде Android Studio.

1.6 Планирование разработки и оценка бюджета

Для управления задачами проекта, распределением доступных ресурсов и реакцией на изменения в изначальном плане было принято решение составить систематизированный перечень задач проекта, позволяющий заранее оценить примерную длительность задач и их зависимости друг от друга, что будет способствовать улучшению организации работы и лучшей оценке возможных рисков проекта [14]. Сам перечень составленных задач представлен ниже в таблице 2.

Таблица 2 – Перечень задач проекта

№ задачи	Название задачи	Длительность	Предшествующие
1	Начало проекта	—	—
2	Анализ требований	—	—
3	Анализ целевой аудитории	1 день	1
4	Анализ бизнес-процессов компании по продаже ПК	2 дня	3
5	Анализ рынка интернет-магазинов компьютерной техники	2 дня	3
6	Анализ функциональных требований	3 дня	3, 4
7	Анализ нефункциональных требований	2 дня	4, 5
8	Анализ UI/UX требований мобильного приложения	3 дня	3, 5
9	Анализ методологий разработки ПО	1 день	6, 7, 8
10	Выбор методологии проектирования и разработки	1 день	9
11	Анализ требований завершен	—	10
12	Проектирование программных средств	—	—

Продолжение таблицы 2

№ задачи	Название задачи	Длительность	Предшествующие
13	Выбор стека используемых технологий	2 дня	11
14	Проектирование структуры базы данных	6 дней	13
15	Проектирование серверной архитектуры	4 дня	13
16	Проектирование клиентской архитектуры	4 дня	13
17	Прототипирование дизайна мобильного приложения	4 дня	16
18	Проектирование программного обеспечения завершено	—	14, 15, 17
19	Реализация программных средств	—	—
20	Реализация базы данных	10 дней	18
21	Реализация серверной части	7 дней	18
22	Реализация мобильного приложения	7 дней	18
23	Реализация завершена	—	20, 21, 22
24	Тестирование программных средств	—	—
25	Функциональное тестирование	6 дней	23
26	Модульное тестирование	6 дней	23
27	Тестирование завершено	—	25, 26
28	Оформление технической документации	5 дней	28
29	Окончание проекта	—	29

Как видно из таблицы, в проекте насчитывается 29 задач, а также 4 основные стадии: анализ требований, проектирование программных средств, реализация программных средств, тестирование программных средств.

Помимо самих задач в проекте необходимо выделить список ресурсов, которыми будет необходимо оперировать в рамках этого проекта. Для этого составим список ресурсов проекта, расположенный ниже в таблице 3.

Таблица 3 – Список ресурсов проекта

Название ресурса	Тип ресурса	Стандартная ставка	Ставка сверхурочных
Backend-разработчик	Трудовой	350,00 Р/ч	200,00 Р/ч
Mobile-разработчик	Трудовой	300,00 Р/ч	200,00 Р/ч
Архитектор баз данных	Трудовой	350,00 Р/ч	200,00 Р/ч
Аналитик	Трудовой	250,00 Р/ч	100,00 Р/ч
UI/UX дизайнер	Трудовой	250,00 Р/ч	100,00 Р/ч
Тестировщик	Трудовой	200,00 Р/ч	100,00 Р/ч
Руководитель проекта	Трудовой	400,00 Р/ч	200,00 Р/ч
Персональный компьютер	Материальный	65 000,00 Р	–
Мобильное устройство	Материальный	30 000,00 Р	–
Бумага	Материальный	1 000,00 Р	–
Интернет	Затраты	3 500,00 Р	–

В проекте насчитывается 11 ресурсов, 7 из которых являются трудовыми и представляют собой главный источник расходов проекта.

Далее выполним назначение соответствующих ресурсов на задачи и оценим уровень их загрузки, чтобы возможные переработки участников реализации проекта. Результаты назначения представлены в таблице 4 ниже.

Таблица 4 – Назначение ресурсов на задачи проекта

№ задачи	Название задачи	Используемые ресурсы	Единицы использования ресурса
1	Начало проекта	–	–

Продолжение таблицы 4

№ задачи	Название задачи	Используемые ресурсы	Единицы использования ресурса
2	Анализ требований	—	—
3	Анализ целевой аудитории	Аналитик Руководитель проекта Персональный компьютер	65% 70% 1 шт.
4	Анализ бизнес-процессов компании по продаже ПК	Аналитик Руководитель проекта Персональный компьютер	50% 50% 1 шт.
5	Анализ рынка интернет-магазинов компьютерной техники	Аналитик Руководитель проекта Персональный компьютер Мобильное устройство	50% 50% 1 шт. 1 шт.
6	Анализ функциональных требований	Аналитик Backend-разработчик Персональный компьютер	33% 60% 2 шт.
7	Анализ нефункциональных требований	Аналитик Персональный компьютер	33% 1 шт.
8	Анализ UI/UX требований мобильного приложения	Аналитик UI/UX-дизайнер Персональный компьютер Мобильное устройство	33% 70% 1 шт. 1 шт.
9	Анализ методологий разработки ПО	Аналитик Персональный компьютер	60% 1 шт.
10	Выбор методологии проектирования и разработки	Аналитик Персональный компьютер	60% 1 шт.
11	Анализ требований завершен	—	—
12	Проектирование программных средств	—	—
13	Выбор стека используемых технологий	Backend-разработчик Mobile-разработчик UI/UX-дизайнер	30% 30% 30%

Продолжение таблицы 4

№ задачи	Название задачи	Используемые ресурсы	Единицы использования ресурса
		Архитектор баз данных	30%
		Аналитик	50%
		Персональный компьютер	5 шт.
14	Проектирование структуры базы данных	Архитектор баз данных	75%
		Персональный компьютер	1 шт.
15	Проектирование серверной архитектуры	Backend-разработчик	75%
		Персональный компьютер	1 шт.
16	Проектирование клиентской архитектуры	Mobile-разработчик	75%
		Персональный компьютер	1 шт.
17	Прототипирование дизайна мобильного приложения	UI/UX-дизайнер	80%
		Персональный компьютер	1 шт.
		Мобильное устройство	1 шт.
18	Проектирование программного обеспечения завершено	—	—
19	Реализация программных средств	—	—
20	Реализация базы данных	Архитектор баз данных	100%
		Персональный компьютер	1 шт.
21	Реализация серверной части	Backend-разработчик	100%
		Персональный компьютер	1 шт.
22	Реализация мобильного приложения	Mobile-разработчик	100%
		Персональный компьютер	1 шт.
23	Реализация завершена	—	—
24	Тестирование программных средств	—	—
25	Функциональное тестирование	Тестирующий	100%
		Персональный компьютер	1 шт.
26	Модульное тестирование	Тестирующий	50%
		Персональный компьютер	1 шт.
27	Нагрузочное тестирование	Тестирующий	50%
		Персональный компьютер	1 шт.

Окончание таблицы 4

№ задачи	Название задачи	Используемые ресурсы	Единицы использования ресурса
28	Тестирование завершено	–	–
29	Оформление технической документации	Backend-разработчик Mobile-разработчик Архитектор баз данных Руководитель проекта Персональный компьютер Бумага Интернет	50% 50% 50% 50% 4 шт. 1 шт.
30	Окончание проекта	–	–

Таким образом, на соответствующие задачи были выделены: аналитик, архитектор баз данных, разработчики серверной и клиентской логики, а также тестировщик и дизайнер пользовательского интерфейса с координацией руководителя проекта.

Чтобы вычислить на основе раннего и позднего начал работ длительность проекта, а также его критического пути – наиболее длинной последовательности взаимосвязанных задач, был спроектирован и описан сетевой график проекта, представленный ниже на рисунке 16 [15]. Ниже приведем математические модели для необходимых расчетов. Расчет раннего времени начала работ производится по формуле:

$$T_D(i) = \min_{j \in G} (T_D(j) + t_j),$$

где $T_D(i)$ – раннее время начала работы;

t_j – время выполнения работы j ;

G – множество работ, непосредственно предшествующих работе i .

Расчет позднего времени начала работ производится по формуле:

$$T_I(i) = \min_{j \in H} (T_I(j) - t_j),$$

где $T_I(i)$ – позднее время начала работы;

t_j – время выполнения работы j ;

H – множество работ, непосредственно следующих за работой i .

Критический путь проекта проходит через задачи, где резерв времени равен нулю, то есть вычитание позднего начала времени работ из раннего дает ноль.

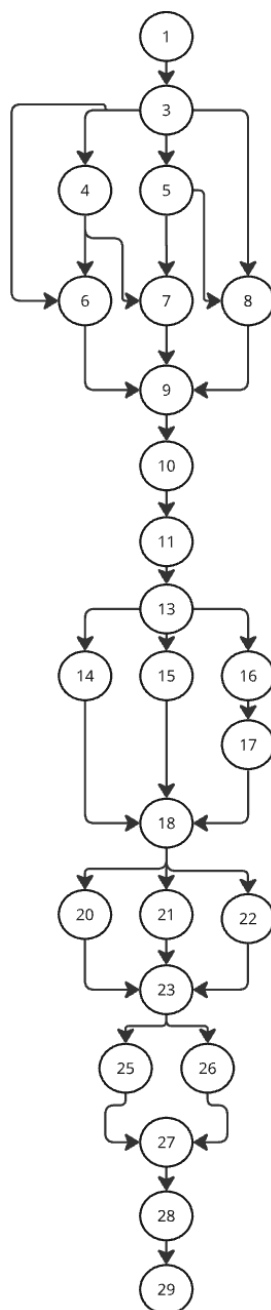


Рисунок 16 – Сетевой график проекта

Для данного графика приведем расчеты раннего времени начала работ для приведенных задач проекта в таблице 5.

Таблица 5 – Расчет раннего времени начала работ

Работа	Предшественники			Вычисление	Раннее время начала
	№	Раннее время начала	Длительность		
1	-	-	-	-	0
3	1	0	0	$0 + 0$	0
4	3	0	1	$0 + 1$	1
5	3	0	1	$0 + 1$	1
6	3	0	1	$\text{Max}(0+1; 1+2)$	3
	4	1	2		
7	4	1	2	$\text{Max}(1+2; 1+2)$	3
	5	1	2		
8	3	0	1	$\text{Max}(0+1; 1+2)$	3
	5	1	2		
9	6	3	3	$\text{Max}(3+3; 3+2; 3+3)$	6
	7	3	2		
	8	3	3		
10	9	6	1	$6 + 1$	7
11	10	7	1	$7 + 1$	8
13	11	8	0	$8 + 0$	8
14	13	8	2	$8 + 2$	10
15	13	8	2	$8 + 2$	10
16	13	8	2	$8 + 2$	10
17	16	10	4	$10 + 4$	14
18	14	10	6	$\text{Max}(10+6; 10+4; 14+4)$	18
	15	10	4		
	17	14	4		
20	18	18	0	$18 + 0$	18
21	18	18	0	$18 + 0$	18
22	18	18	0	$18 + 0$	18
23	20	18	10	$\text{Max}(18+10; 18+7; 18+7)$	28
	21	18	7		
	22	18	7		
25	23	28	0	$28 + 0$	28
26	23	28	0	$28 + 0$	28
27	25	28	6	$\text{Max}(28+6; 28+6)$	34
	26	28	6		
28	27	34	0	$34 + 0$	34
29	28	34	5	$34 + 5$	39

Также в таблице 6 было рассчитано позднее время начало работ. Его расчеты представлены ниже.

Таблица 6 – Расчет позднего времени начала работ

Работа	Последователи			Вычисление	Позднее время начала
	№	Позднее время начала	Длительность		
28	29	39	0	39 - 0	39
27	28	39	5	39 - 5	34
26	27	34	0	34 - 0	34
25	27	34	0	34 - 0	34
23	25	34	6	Min(34 - 6; 34 - 6)	28
	26	34	6		
22	23	28	0	28 - 0	28
21	23	28	0	28 - 0	28
20	23	28	0	28 - 0	28
18	20	28	10	Min(28 - 10; 28 - 7; 28 - 7)	18
	21	28	7		
	22	28	7		
17	18	18	0	18 - 0	18
16	17	18	4	18 - 4	14
15	18	18	0	18 - 0	18
14	18	18	0	18 - 0	18
13	14	18	6	Min(18 - 6; 18 - 4; 14 - 4)	10
	15	18	4		
	16	14	4		
11	13	10	2	10 - 2	8
10	11	8	0	8 - 0	8
9	10	8	1	8 - 1	7
8	9	7	1	7 - 1	6
7	9	7	1	7 - 1	6
6	9	7	1	7 - 1	6
5	7	6	2	Min(6 - 2; 6 - 3)	3
	8	6	3		
4	6	6	3	Min(6 - 3; 6 - 2)	3
	7	6	2		
3	4	3	2	Min(3 - 2; 3 - 2; 6 - 3; 6 - 3)	1
	5	3	2		
	6	6	3		
	8	6	3		
1	3	1	1	1 - 1	0

На основе рассчитанных раннего и позднего времен начала работ вычислим критический путь проекта. Расчеты приведены ниже в таблице 7.

Таблица 7 – Вычисление критического пути проекта

Номер работы	Раннее время начала	Позднее время начала	Резерв времени
1	0	0	0
3	0	1	1
4	1	3	2
5	1	3	2
6	3	6	3
7	3	6	3
8	3	6	3
9	6	7	1
10	7	8	1
11	8	8	0
13	8	10	2
14	10	18	8
15	10	18	8
16	10	14	4
17	14	18	4
18	18	18	0
20	18	28	10
21	18	28	10
22	18	28	10
23	28	28	0
25	28	34	6
26	28	34	6
27	34	34	0
28	34	39	5
29	39	39	0

На основе проведенных вычислений можно заключить, что критический путь проекта проходит через следующие задачи: 1, 3, 4, 6, 9, 10, 11, 13, 16, 17, 18, 20, 23, 25, 27, 28, 29, то есть путь, не имеющий резерва по времени выполнения и имеющий длительность 39 дней.

Ниже на рисунке 17 представлен сетевой граф проекта с выделенным пунктирной линией критическим путем.

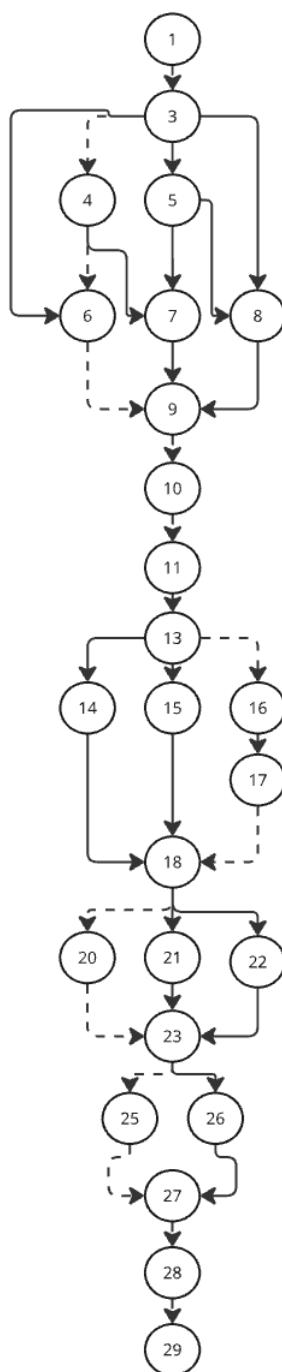


Рисунок 17 – Критический путь проекта на сетевом графе

Чтобы упростить организацию задач в проекте и управление ресурсами, было принято решение перенести проект в специализированную среду Microsoft Office Project 2007 для автоматизированного расчета длительности, дат, вычисления требуемого бюджета и генерации необходимых диаграмм. Перенесенный список задач проекта представлен ниже на рисунке 18.

Название задачи	Длительность	Начало	Окончание	Предшественники
Начало проекта	0 дней	Чт 26.02.26	Чт 26.02.26	
Анализ требований	8 дней	Чт 26.02.26	Пн 09.03.26	
Анализ целевой аудитории	1 день	Чт 26.02.26	Чт 26.02.26	1
Анализ бизнес-процессов компании по продаже ПК	2 дней	Чт 26.02.26	Пн 02.03.26	3
Анализ рынка интернет-магазинов компьютерной техники	2 дней	Чт 26.02.26	Пн 02.03.26	3
Анализ функциональных требований	3 дней	Пн 02.03.26	Чт 05.03.26	4;3
Анализ нефункциональных требований	2 дней	Пн 02.03.26	Ср 04.03.26	5;4
Анализ UI/UX требований мобильного приложения	3 дней	Пн 02.03.26	Чт 05.03.26	5;3
Анализ методологий разработки ПО	1 день	Чт 05.03.26	Пт 06.03.26	8;6;7
Выбор методологии проектирования и разработки	1 день	Пт 06.03.26	Пн 09.03.26	9
Анализ требований завершен	0 дней	Пн 09.03.26	Пн 09.03.26	10
Проектирование программных средств	10 дней	Пн 09.03.26	Пн 23.03.26	
Выбор стека используемых технологий	2 дней	Пн 09.03.26	Ср 11.03.26	11
Проектирование структуры базы данных	6 дней	Ср 11.03.26	Чт 19.03.26	13
Проектирование серверной архитектуры	4 дней	Ср 11.03.26	Вт 17.03.26	13
Проектирование клиентской архитектуры	4 дней	Ср 11.03.26	Вт 17.03.26	13
Прототипирование дизайна мобильного приложения	4 дней	Вт 17.03.26	Пн 23.03.26	16
Проектирование программного обеспечения завершенно	0 дней	Пн 23.03.26	Пн 23.03.26	17;14;15
Реализация программных средств	10 дней	Пн 23.03.26	Пн 06.04.26	
Реализация базы данных	10 дней	Пн 23.03.26	Пн 06.04.26	18
Реализация серверной части	7 дней	Пн 23.03.26	Ср 01.04.26	18
Реализация мобильного приложения	7 дней	Пн 23.03.26	Ср 01.04.26	18
Реализация завершенна	0 дней	Пн 06.04.26	Пн 06.04.26	20;21;22
Тестирование программных средств	6 дней	Пн 06.04.26	Вт 14.04.26	
Функциональное тестирование	6 дней	Пн 06.04.26	Вт 14.04.26	23
Модульное тестирование	6 дней	Пн 06.04.26	Вт 14.04.26	23
Тестирование завершенно	0 дней	Вт 14.04.26	Вт 14.04.26	26;25
Оформление технической документации	5 дней	Вт 14.04.26	Вт 21.04.26	27
Окончание проекта	0 дней	Вт 21.04.26	Вт 21.04.26	28

Рисунок 18 – Список задач в среде Microsoft Project

После присваивания задачам соответствующих ресурсов, на основе данных задач была сгенерирована диаграмма Ганта, отражающая последовательность задач проекта, она расположена ниже на рисунке 19.

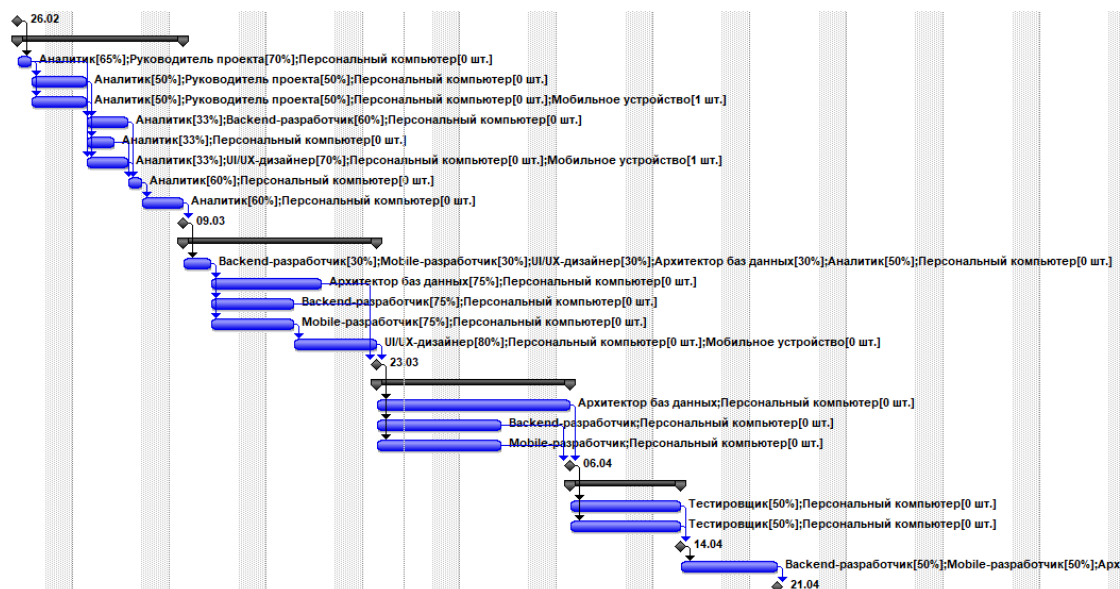


Рисунок 19 – Диаграмма Ганта проекта

На диаграмме видны соответствующие задачи проекта, его этапы, вехи, а также ресурсы, соответствующие задачам.

Помимо этого, среда предоставляет возможности анализа загрузки ресурсов для возможной переоценки сотрудника или реорганизации задач в соответствии с необходимой нагрузкой. Ниже на рисунках представлены графики загрузки ресурсов в виде столбчатых диаграмм.

На рисунке 20 ниже представлен график загрузки ресурса «Backend-разработчик».

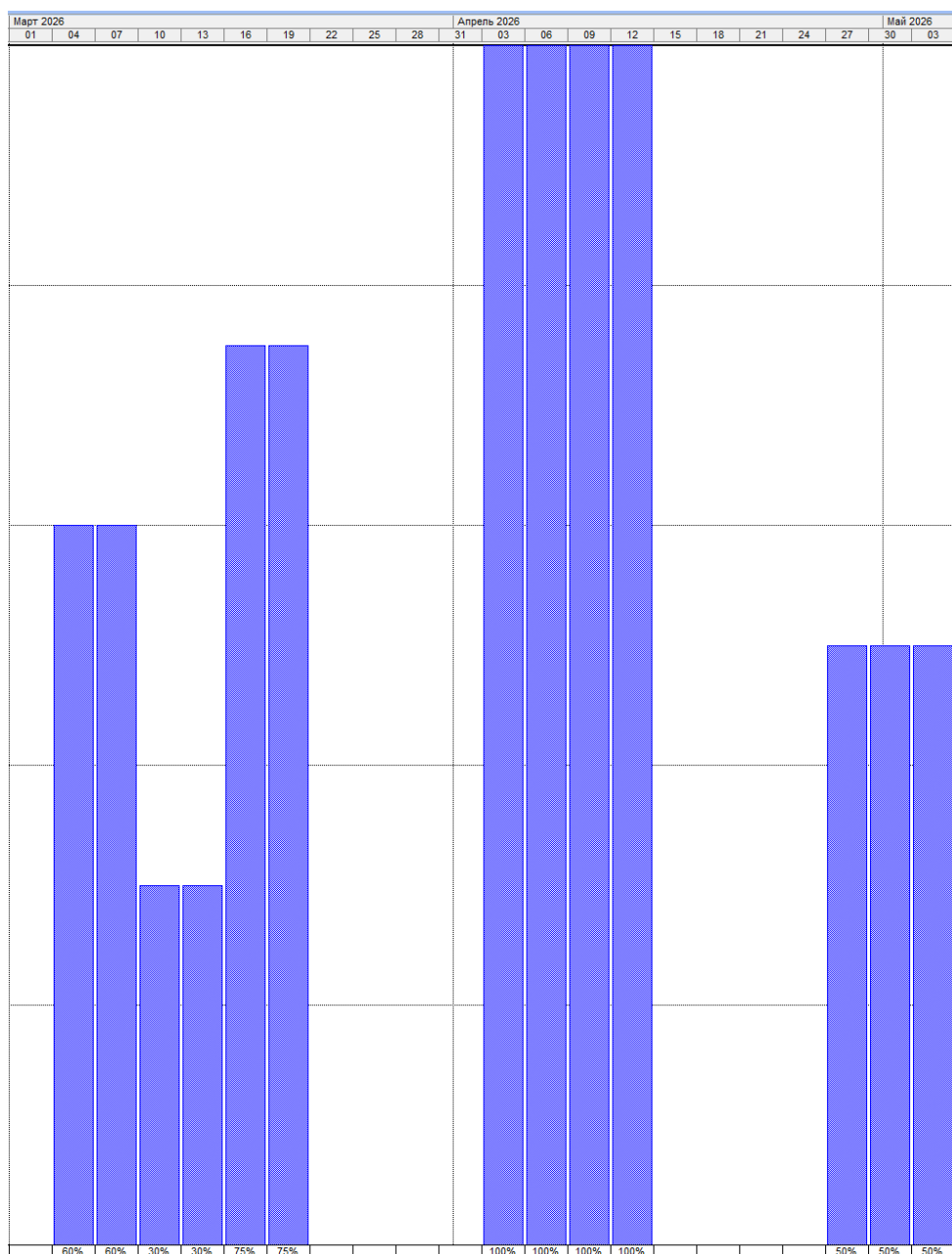


Рисунок 20 – График загрузки backend-разработчика

На рисунке 21 представлен график загрузки ресурса «mobile-разработчик».

На рисунке 22 представлен график загрузки ресурса «Архитектор баз данных».

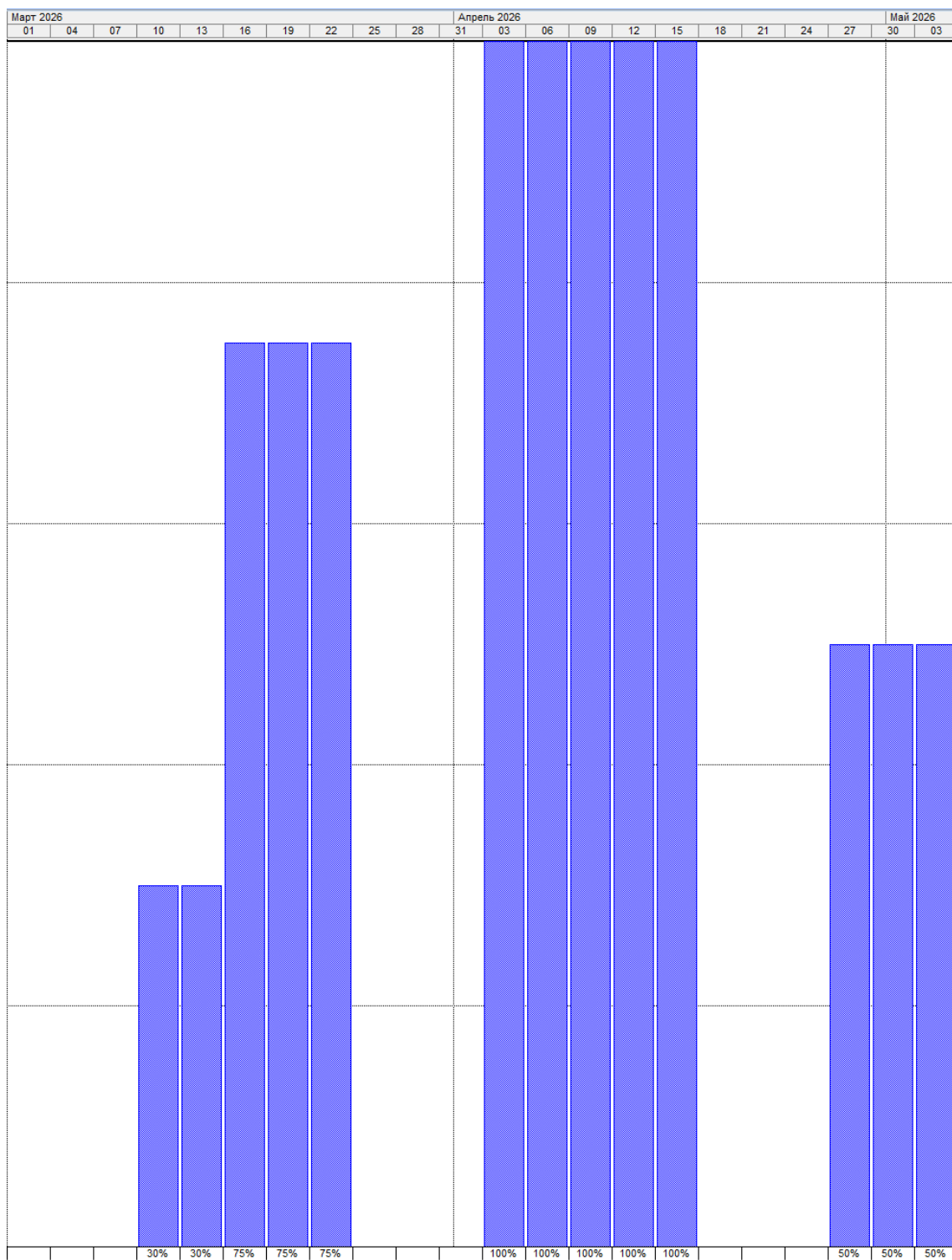


Рисунок 22 – График загрузки архитектора баз данных

На рисунке 23 представлен график загрузки ресурса «Аналитик».

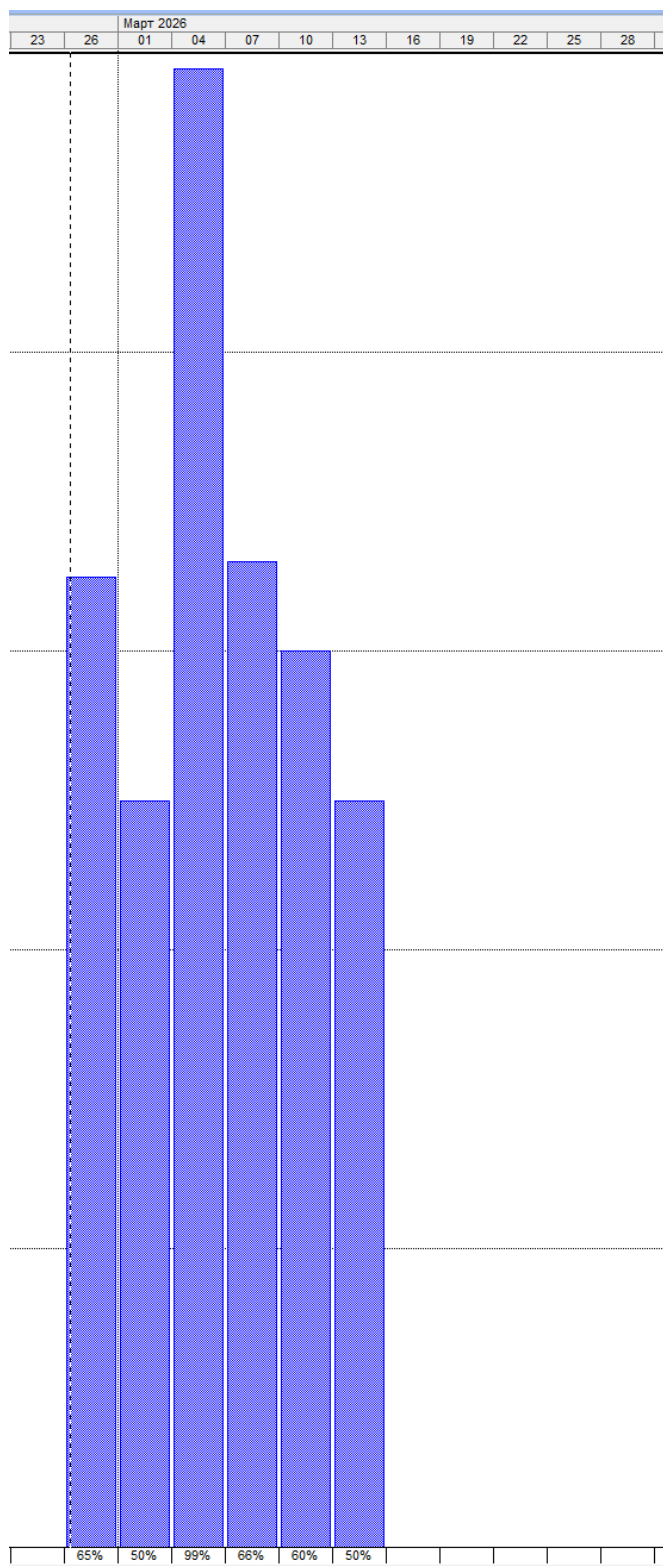


Рисунок 23 – График загрузки аналитика

На рисунке 24 представлен график загрузки ресурса «UI/UX дизайнер».

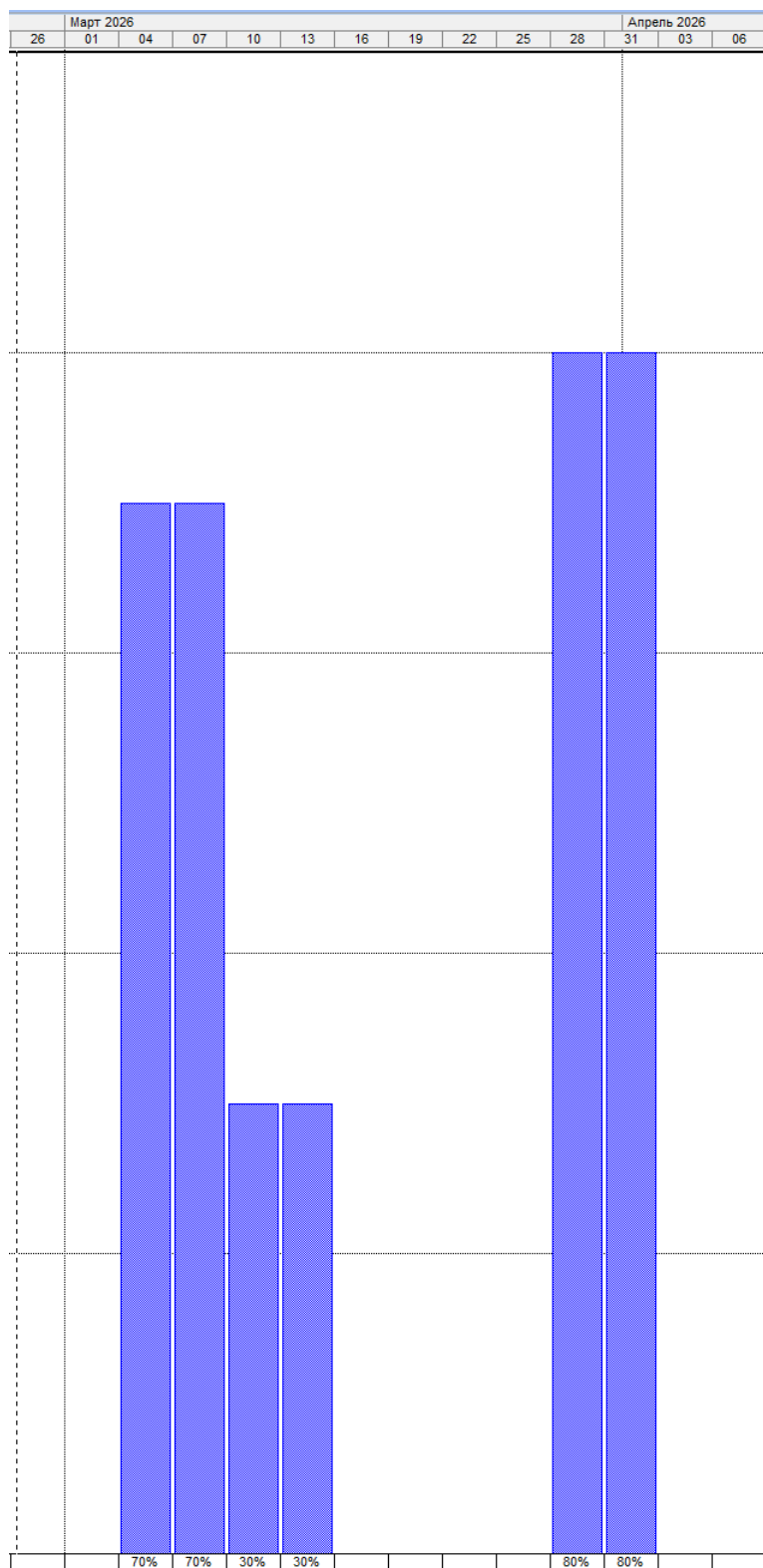


Рисунок 24 – График загрузки UI/UX-дизайнера

На рисунке 25 представлен график загрузки ресурса «Тестировщик».

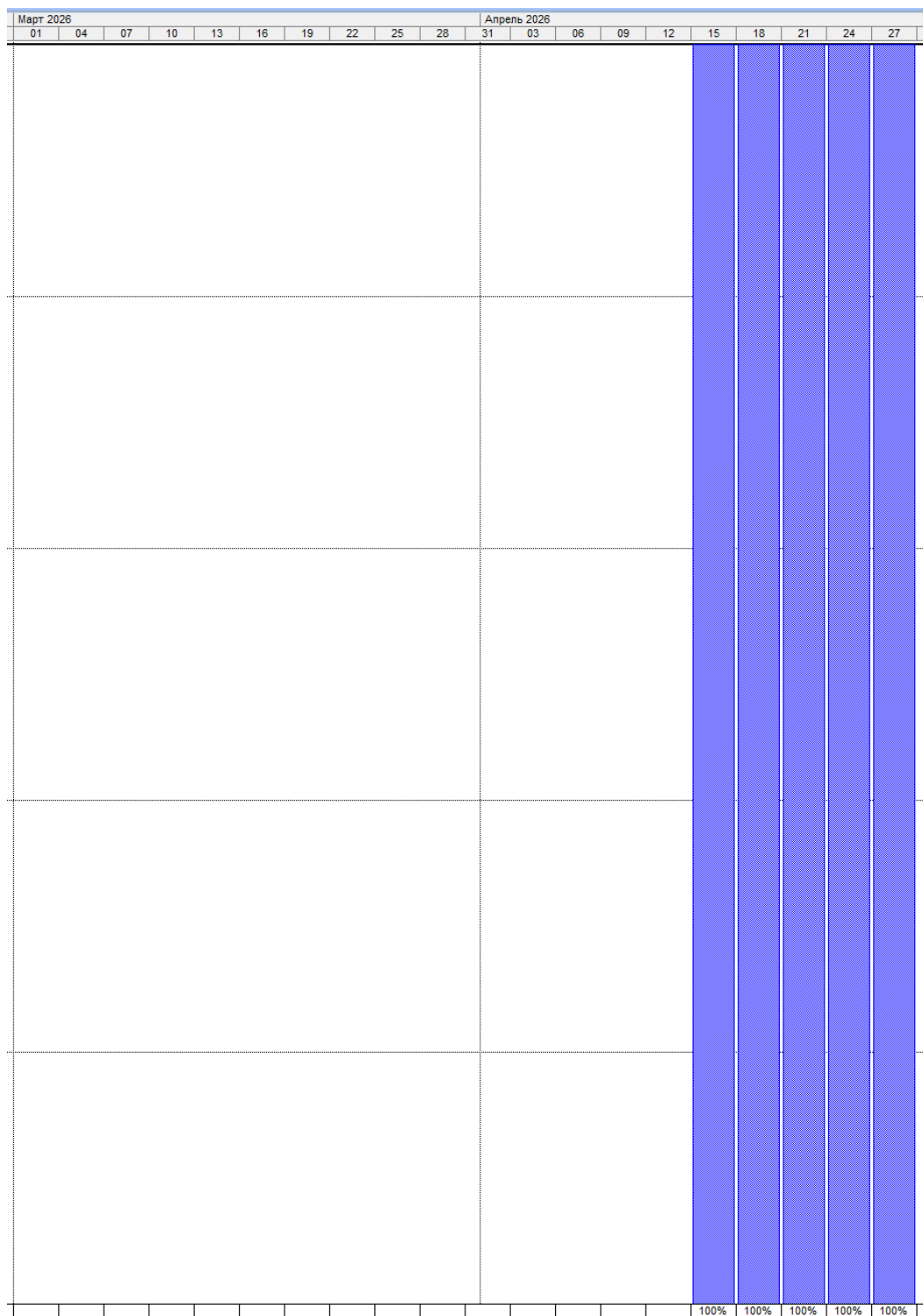


Рисунок 25 – График загрузки тестировщика

На рисунке 26 представлен график загрузки ресурса «Руководитель проекта».

На рисунке 27 представлен график загрузки ресурса «Персональный компьютер».

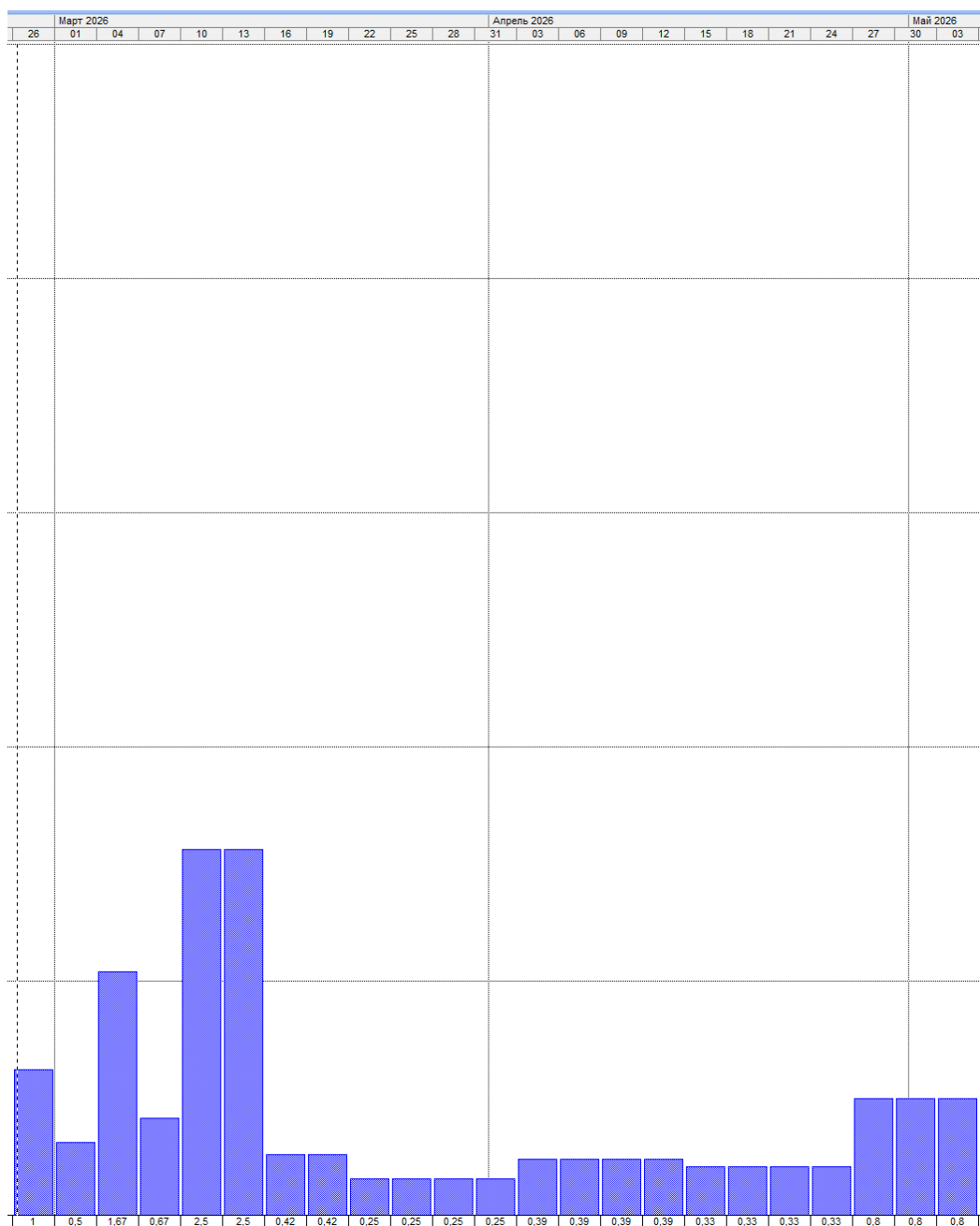


Рисунок 27 – График загрузки ресурса «Персональный компьютер»

На рисунке 28 представлен график загрузки ресурса «Мобильное устройство».

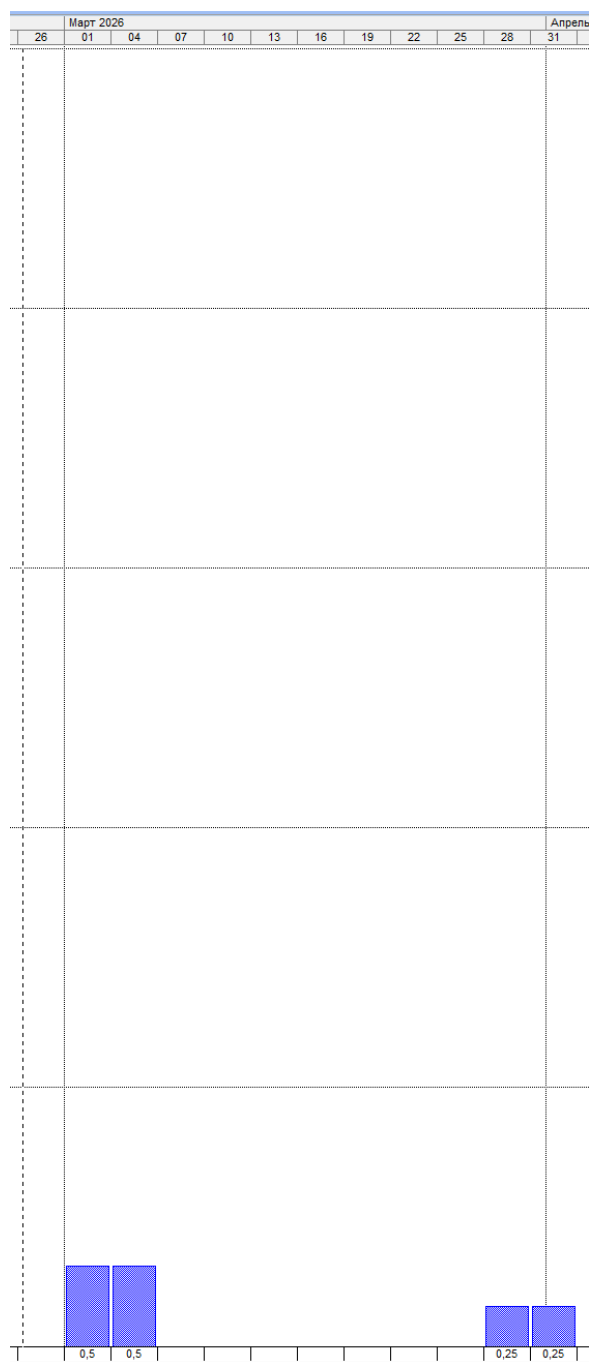


Рисунок 28 – График загрузки ресурса «Мобильное устройство»

После переноса проекта в Microsoft Project также становится доступна генерация отчетов о проекте на основе содержащихся в нем данных, что является удобным способом быстро получить необходимую пользователю информацию. Воспользуемся инструментом и получим сводку о проекте, она находится ниже на рисунке 29.

Проект по разработке и интеграции мобильного приложения интернет-магазина по продаже ПК

на дату Чт 26.03.26

Даты			
Начало:	Чт 26.02.26	Окончание:	Вт 21.04.26
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней
Длительность			
Текущий план:	39 дней	Оставшаяся:	39 дней
Базовый план:	0 дней?	Фактическая:	0 дней
Отклонение:	39 дней	Процент завершения:	0%
Трудозатраты			
Текущий план:	580,72 ч	Оставшиеся:	580,72 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	580,72 ч	Процент завершения:	0%
Затраты			
Текущий план:	261 260,00 Р	Оставшиеся:	261 260,00 Р
Базовый план:	0,00 Р	Фактические:	0,00 Р
Отклонение:	261 260,00 Р		
Состояние задачи		Состояние ресурса	
Еще не начатые задачи	29	Трудовые ресурсы:	8
Выполняющиеся задачи	0	Трудовые ресурсы с прев. дост	0
Завершенные задачи:	0	Материальные ресурсы:	3
Итого задач:	29	Итого ресурсов:	11

Рисунок 29 – Сводка проекта по разработке мобильного приложения салона по продаже ПК

Из рисунка видно, что проект рассчитан 39 рабочих дней или почти 600 рабочих часов, а общие затраты составляют 261 260,00 рублей, на основе чего можно сделать вывод о экономической целесообразности проекта.

2 Проектирование мобильного приложения

2.1 Выбор технологий и инструментальных средств

В ходе анализа, проектирования и разработки в рамках выпускной квалификационной работы будет использоваться язык моделирования UML, с помощью которого будут визуализированы необходимые диаграммы, показывающие необходимую информацию с использованием утвержденного стандарта. В качестве программного обеспечения для моделирования диаграмм UML использовалась Visual Paradigm Community Edition [16].

В ходе реализации приложения будет использоваться метод kanban в рамках методологии Agile, совмещающая в себе гибкость и эффективность разработки. При использовании kanban все задачи имеют состояния: «В работе», «Завершена», «Планируется», что комбинируется в ограничением одновременно выполняемых задач, что ограничивает контекст разработки взаимосвязанными компонентами без частого его переключения [17]. Помимо этого, выполняемым задачам можно менять приоритет выполнения, что позволяет сконцентрировать усилия на наиболее важным аспектах.

Задачи также анализируются с помощью таких метрик, как время выполнения и общее время выполнения. Таким образом, появляется возможность оценить скорость разработки и сравнить прогресс серверной и клиентской частей, что особо хорошо интегрируется с разработанным проектом в Microsoft Project, позволяя быстро реагировать на изменения в рабочем потоке.

В самой методологии отсутствуют типичные итерации разработки, вместо этого этапы можно представить как буфер задач, проходящих через состояние разработки.

На первом этапе фиксируются задачи и помечаются как «Планируется», затем организуются по порядку выполнения.

Затем в рамках ограничений на одновременное количество задач выбираются наиболее приоритетные задачи, после чего над ними ведется работа.

Такие задачи помечаются как «В работе». При этом задачи могут быть помечены как «Зависимые», если требуют реализацию задач сервера.

На третьем этапе ведется работа над задачами из текущего списка «В работе», после выполнения задач они проходят этап интеграции и тестирования. Если в ходе тестирования были обнаружены ошибки, этап откатывается до реализации.

После успешного тестирования задачи помечаются как «Завершена», а разработчик сразу берет следующую задачу, вместо ожидания окончания спринта. Вместе с этим происходит анализ метрик по времени выполнения задачи и полного цикла от планирования до реализации.

На этапе анализа требований для описания задач, ресурсов и временных рамок проекта использовалось программное обеспечение Microsoft Project, предоставляющее возможность визуализировать распределение ресурсов и задач по проекту, предоставить сводку и диаграмму Ганта.

В качестве средств разработки клиентского мобильного приложения будет использоваться среда Android Studio с поддержкой языка Kotlin и использованием фреймворка Jetpack Compose. Серверная часть приложения будет разрабатываться с использованием современного языка C# в редакторе кода Visual Studio Code с использованием платформы ASP.NET и шаблоном и архитектурой модульного монолита с REST API. При выборе СУБД было принято решение использовать PostgreSQL, поэтому для взаимодействия с базой данных используются средства EF Core.

2.2 Проектирование архитектуры программного обеспечения

В ходе проектирования архитектуры программного обеспечения было принято решения использовать трехуровневую клиент-серверную архитектуру. В данном случае клиентская сторона отвечает за предоставление пользовательского интерфейса и отправку данных на сервер с малым количеством обрабатываемой логики, в то же время серверная часть отвечает за

обработку данных и обращения к базе данных для сохранения и актуализации информации [18].

При использовании клиент-серверной архитектуры учитываются следующие ее особенности:

- разделение функционала и ответственности между клиентским приложением и сервером, что дает возможность распределять нагрузку и повысить масштабирование;
- для конечного пользователя не важно, как именно работает сервер и его внутреннее устройство для пользования клиентом;
- возможна балансировка нагрузки и повышение отказоустойчивости при репликации серверов и установки балансировщиком нагрузки посредниками.

Ниже на рисунке 30 представлена визуализация трехуровневой архитектуры клиент-сервер.

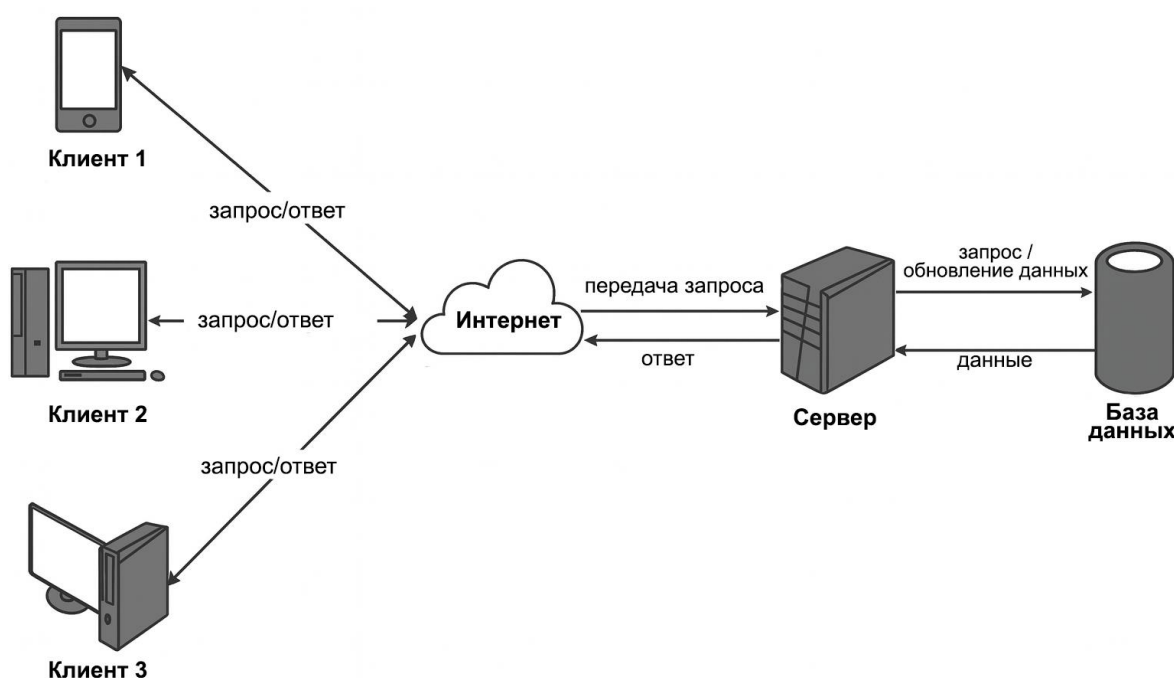


Рисунок 30 – Трехуровневая модель клиент-сервер

Таким образом, выбранная архитектура позволяет независимо разрабатывать сервер и мобильный клиент, разбивая зоны ответственности между ними.

2.3 Проектирование базы данных

2.3.1 Концептуальное проектирование базы данных

При проектировании базы данных были приняты во внимание результаты анализа предметной области, на основе чего была построена концептуальная модель базы данных в соответствии со стандартом UML 2.0. Она представлена ниже на рисунке 31.

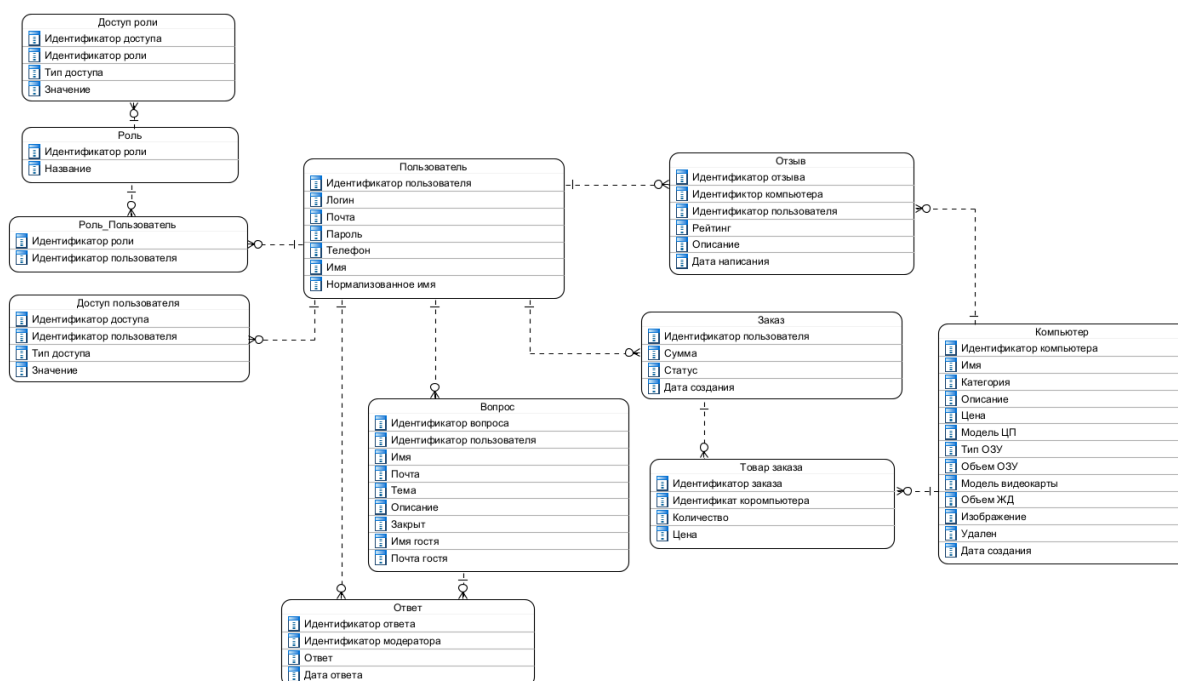


Рисунок 31 – Визуализация концептуальной модели базы данных

Описание сущностей, представленных на рисунке диаграммы сущностей и зависимостей, представлено ниже в таблице 8.

Таблица 8 – Описание сущностей концептуальной модели БД

№	Название	Описание	Количество экземпляров	Режим обновления
1	Пользователь	Все созданные аккаунты пользователей всех ролей	25000	При регистрации нового пользователя
2	Роль	Отдельные роли пользователя	3	При расширении функционала магазина
3	Роль_Пользователь	Каждая запись показывает, что определенный пользователь обладает определенной ролью	25000	При обновлении ролей пользователя

Продолжение таблицы 8

№	Название	Описание	Количество экземпляров	Режим обновления
4	Доступ роли	Определенные возможности для роли пользователя	30	При добавлении новых возможностей ролям
5	Доступ пользователя	Определенные возможности для конкретного пользователя	30	При обновлении функционала пользователя
6	Вопрос	Все заданные вопросы пользователями	50000	При задании пользователем вопроса
7	Ответ	Все ответы на вопросы менеджерами	50000	При написании менеджером ответа
8	Отзыв	Список оставленных отзывов на компьютер	50000	При написании пользователем отзыва
9	Заказ	Список оформленных пользователями заказов	100000	При оформлении покупки из корзины
10	Компьютер	Список добавленных компьютеров администраторами	25	При добавлении, изменении, удалении ПК администратором
11	Товар заказа	Список компьютеров в заказе	100000	При обновлении товаров заказа

Также рассмотрим указанные связи между сущностями. Они представлены ниже в таблице 9.

Таблица 9 – Связи сущностей концептуальной модели БД

№	Связь	Описание	Мощность	Обязательность
1	Пользователь – Роль_Пользователь	Один пользователей может иметь несколько ролей	1:M	Да
2	Роль – Роль_Пользователь	Одна роль присвоена множеству пользователей	1:M	Да
2	Роль – Доступ роли	Все роли имеют список доступных возможностей	1:M	Да
3	Пользователь – Доступ пользователя	Все пользователи имеют уникальные права доступа	1:M	Да
4	Пользователь - Вопрос	Все пользователи могут задать вопрос	1:M	Да
5	Пользователь – Ответ	Некоторые пользователи отвечают на вопрос	1:M	Да
6	Вопрос – Ответ	Все вопросы могут получить ответ	1:M	Да
7	Пользователь – Отзыв	Все пользователи могут оставить отзыв	1:M	Да
8	Пользователь - Заказ	Все пользователи могут оформить заказ	1:M	Да
9	Компьютер - Отзыв	Каждый компьютер может иметь отзывы	1:M	Да

Продолжение таблицы 9

№	Связь	Описание	Мощность	Обязательность
10	Заказ – Товар заказа	Все заказы имеют список товаров	1:M	Да
11	Компьютер – Товар заказа	Компьютер может быть в списке заказов	1:M	Да

На основе выделенной концептуальной модели были выявлены необходимые сущности и построены связи между ними.

2.3.2 Логическое проектирование базы данных

С использованием концептуальной модели базы данных была построена логическая модель базы данных, определяющая первичные и внешние ключи, атрибуты и их ограничения. Она представлена ниже на рисунке 32.

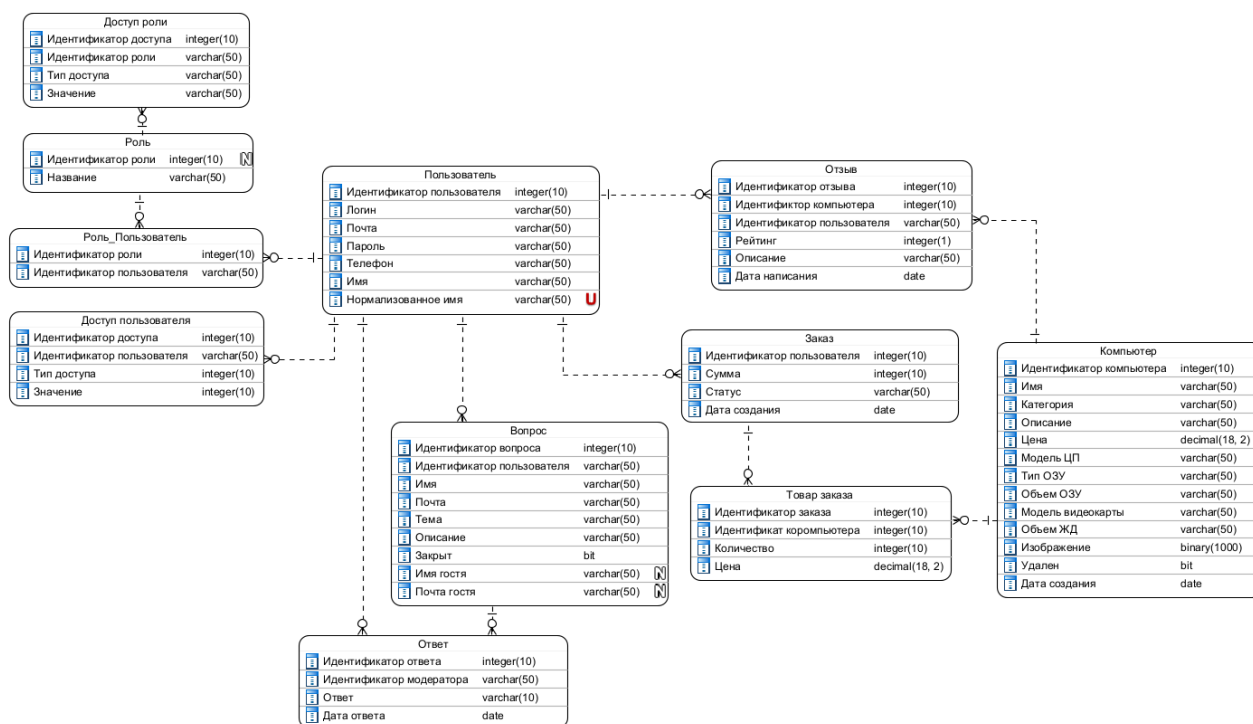


Рисунок 32 – Логическая модель схемы базы данных

На данном этапе были явно определены связи между сущностями в виде ключей, устранены неточности концептуальной модели и выделены конкретные атрибуты каждой сущности.

2.4 Проектирование серверной части приложения

В ходе проектирование серверной части приложения был выбран архитектурный стиль REST, позволяющий реализовать простое взаимодействие мобильного клиента и сервера. Он позволяет предоставлять определенный ресурс на объявленных конечных точках, не имеет данных о состоянии клиента, использует стандартные HTTP методы. Также в качестве обмена будет использоваться формат JSON и авторизация на основе JWT. Доступы различных пользователей будут определены специальными ролями – полями, определяющими доступ к функционалу приложения. Для этого на сервере используется платформа Identity как система аутентификации и управления пользователями.

Сама система представляет собой модульный монолит – приложение сервера разворачивается как один процесс с одной базой данных, но с четко определенными границами модулей и способами их взаимодействия [19]. Ее визуализация представлена ниже на рисунке 33.

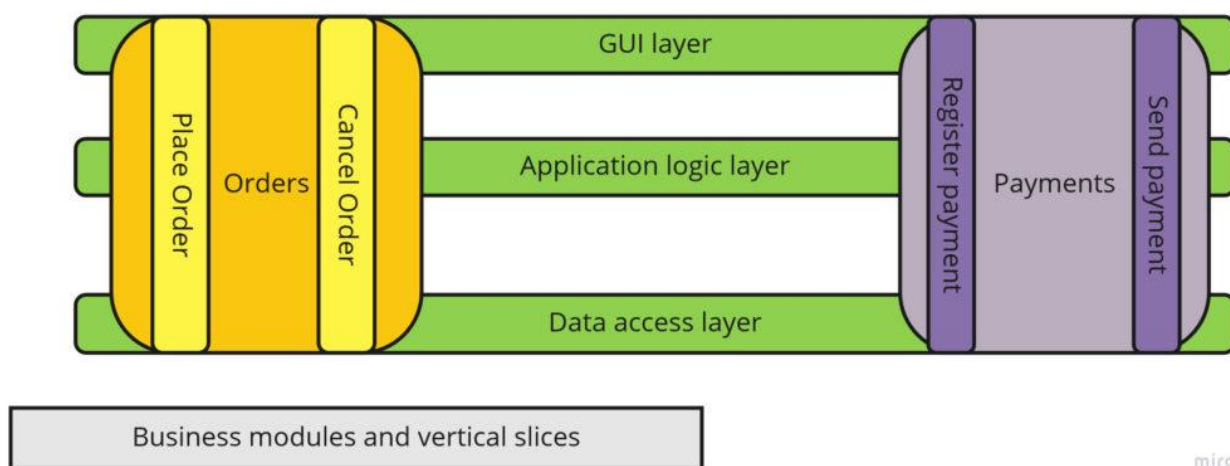


Рисунок 33 – Модульный монолит в визуальном представлении

Таким образом, на этапе проектирования серверной части были определены общие архитектурные стили для обмена данными и организации модулей в проекте.

2.5 Проектирование пользовательского интерфейса

На основе разработанной диаграммы вариантов использования и спроектированных возможностей серверной части приложения были разработаны макеты интерфейса мобильного приложения, позволяющие визуализировать важные элементы пользовательского интерфейса. Среди них можно выделить каталог, окно заказов, поддержки и личного кабинета с окнами регистрации и авторизации. На рисунках ниже представлены разработанные макеты интерфейса с пояснительными текстами для каждого макета. На рисунке 34 представлен макет каталога магазина.

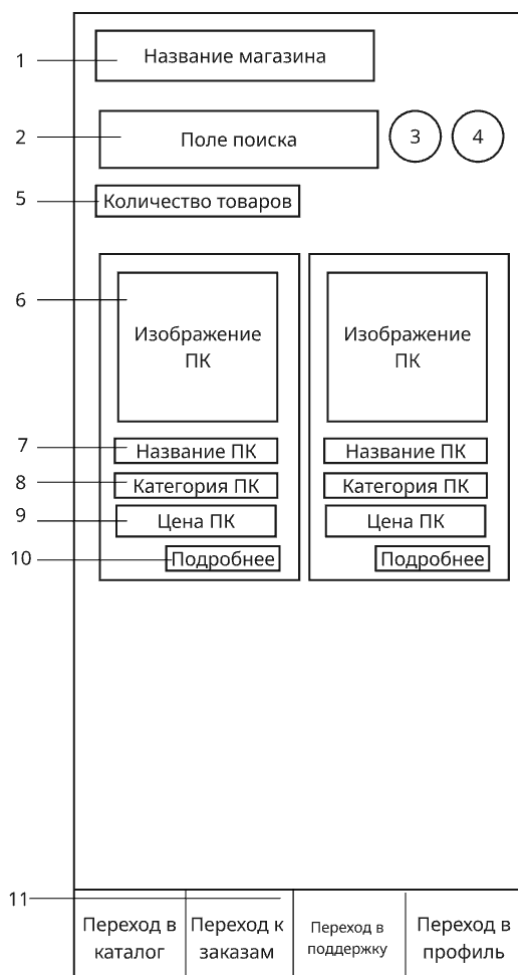


Рисунок 34 – Макет каталога магазина

В макете каталога присутствуют следующие элементы, перечисленные ниже:

1 – текст с названием магазина;

- 2 – поле поиска конкретного товара;
- 3 – кнопка фильтрации каталога;
- 4 – кнопка сортировки каталога;
- 5 – текст с полученным количеством товаров;
- 6 – изображение товара ПК;
- 7 – текст с названием ПК;
- 8 – текст с категорией ПК;
- 9 – текст с ценой ПК;
- 10 – кнопка перехода на карточку товара;
- 11 – меню навигации по приложению, включает каталог, заказы, поддержку и профиль.

Ниже на рисунке 35 представлен макет страницы карточки товара.



Рисунок 35 – Макет карточки товара

В макете карточки товара присутствуют следующие элементы, перечисленные ниже:

- 1 – кнопка перехода на предыдущую окно;

- 2 – текст названия окна;
- 3 – изображение ПК;
- 4 – текст с названием ПК;
- 5 – текст с ценой ПК;
- 6 – текст с описанием ПК;
- 7 – кнопка добавление ПК в корзину;
- 8 – кнопка написания отзыва (для зарегистрированного пользователя);
- 9 – таблица с текстом характеристик ПК;
- 10 – отзывы, написанные на ПК.

Ниже на рисунке 36 представлен макет формы написания отзыва.

The diagram shows a rectangular form with the following elements and labels:

- 1 points to a box labeled "Название формы" (Form name).
- 2 points to a box labeled "Текущая оценка" (Current rating).
- 3 points to a box labeled "Слайдер выбора оценки" (Rating selection slider).
- 4 points to a large box labeled "Поле ввода отзыва" (Review input field).
- 5 points to a button labeled "Отмена" (Cancel).
- 6 points to a button labeled "Кнопка отправки" (Send button).

Рисунок 36 – Макет формы отзыва

В макете формы отзыва присутствуют следующие элементы, перечисленные ниже:

- 1 – текст с названием формы;
- 2 – текст выбранной оценки;
- 3 – интерактивный слайдер для выбора оценки;
- 4 – поле ввода отзыва;
- 5 – кнопка отмены написания отзыва;
- 6 – кнопка отправки отзыва.

Ниже на рисунке 37 представлен макет окна корзины.



Рисунок 37 – Макет интерфейса корзины

В макете формы отзыва присутствуют следующие элементы, перечисленные ниже:

- 1 – кнопка перехода на предыдущую окно;
- 2 – текст названия окна;
- 3 – суммарная стоимость заказа;
- 4 – текст с названием ПК;
- 5 – текст с количеством заказанных ПК и их ценой;
- 6 – кнопка оформления заказа.

Ниже на рисунке 38 представлен макет окна заказов пользователя.



Рисунок 38 – Макет интерфейса окна заказов

В макете интерфейса заказов присутствуют следующие элементы, перечисленные ниже:

- 1 – текст названия окна;
- 2 – кнопка перехода в корзину;
- 3 – номер созданного заказа;
- 4 – итоговая сумма созданного заказа;
- 5 – статус созданного заказа;
- 6 – навигационное меню приложения.

Ниже на рисунке 39 представлен макет окна поддержки.

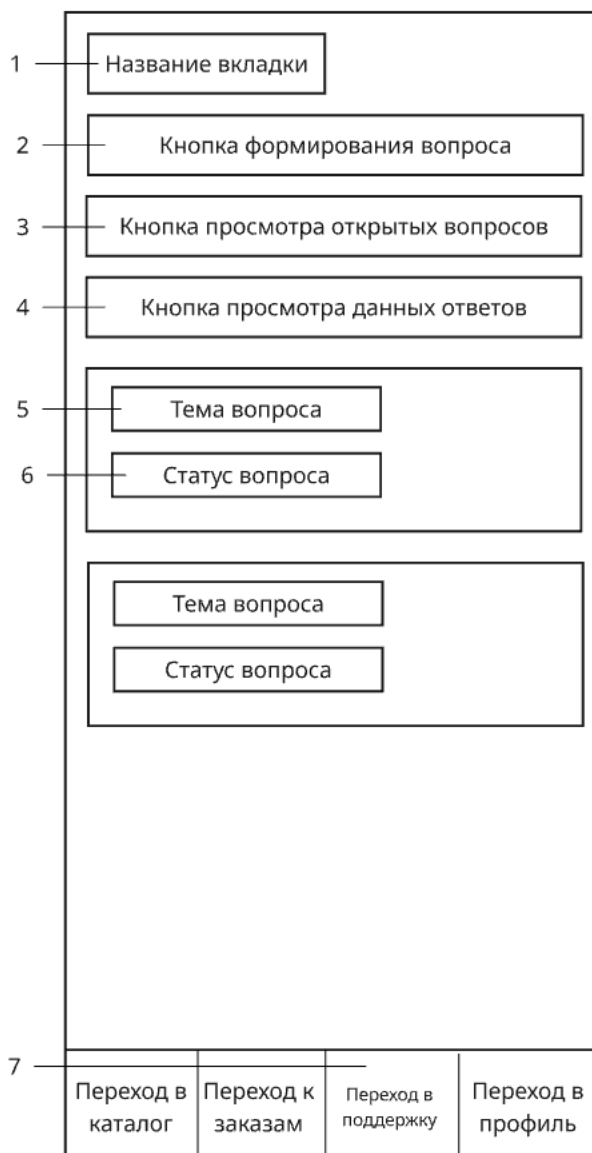


Рисунок 39 – Макет интерфейса окна поддержки

В макете интерфейса поддержки присутствуют следующие элементы, перечисленные ниже:

- 1 – текст названия окна;
- 2 – кнопка перехода на форму заполнения вопроса;
- 3 – кнопка перехода к списку открытых вопросов (для менеджера по работе с клиентами и администратора);
- 4 – кнопка перехода к списку ответов (для менеджера по работе с клиентами и администратора);
- 5 – тема заданного вопроса;

6 – статус вопроса (открыт или закрыт);

7 – навигационное меню приложения.

Ниже на рисунке 40 представлен макет окна заполнения вопроса.

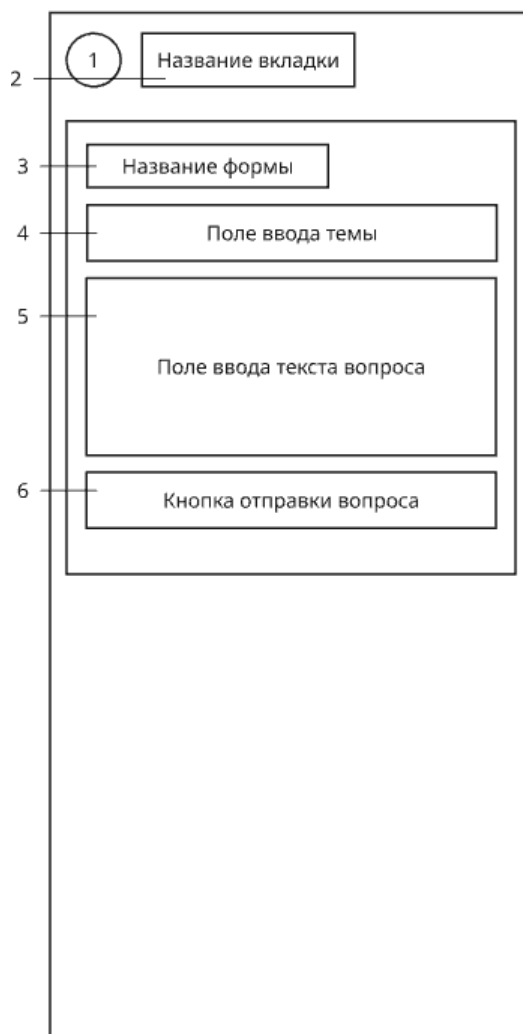


Рисунок 40 – Макет интерфейса отправки вопроса

В макете интерфейса отправки вопроса присутствуют следующие элементы, перечисленные ниже:

1 – кнопка возврата на предыдущую окно;

2 – текст с названием окна;

3 – текст с названием формы;

4 – поле ввода названия темы вопроса;

5 – поле ввода текста вопроса;

6 – кнопка отправки вопроса.

Ниже на рисунке 41 представлен макет окна просмотра списка вопросов.

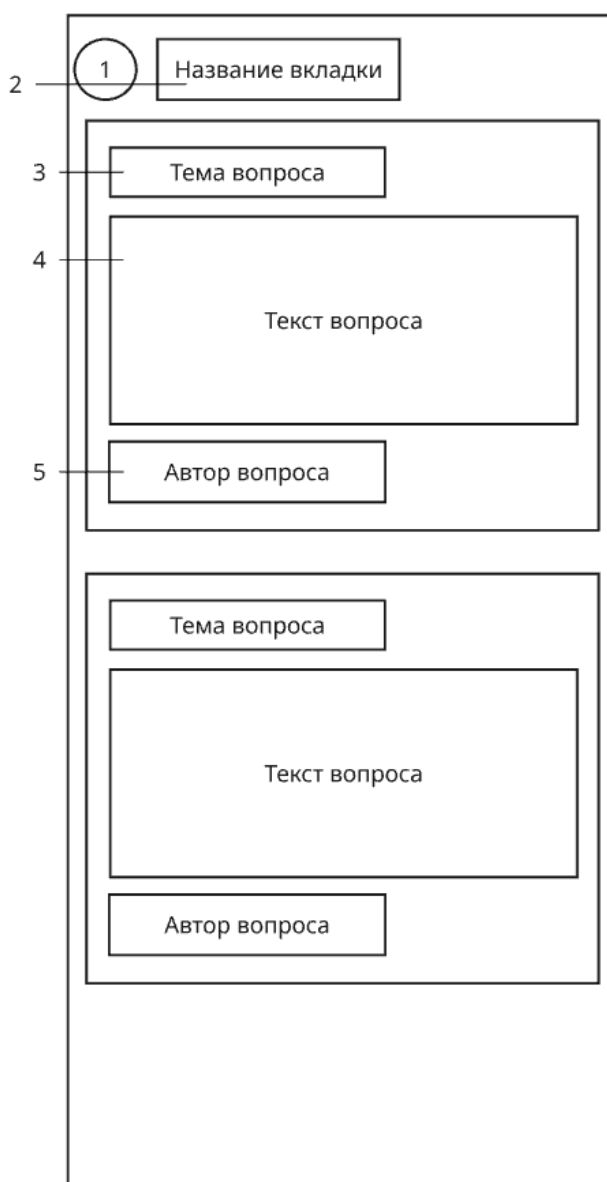


Рисунок 41 – Макет интерфейса просмотра вопросов

При нажатии на любой вопрос будет открыта форма ответа на него. В макете интерфейса просмотра вопросов присутствуют следующие элементы, перечисленные ниже:

- 1 – кнопка возврата на предыдущую окно;
- 2 – текст с названием окна;
- 3 – текст с темой вопроса;
- 4 – текст вопроса;
- 5 – текст с указанием автора вопроса.

Ниже на рисунке 42 представлен макет окна ответа на вопрос.

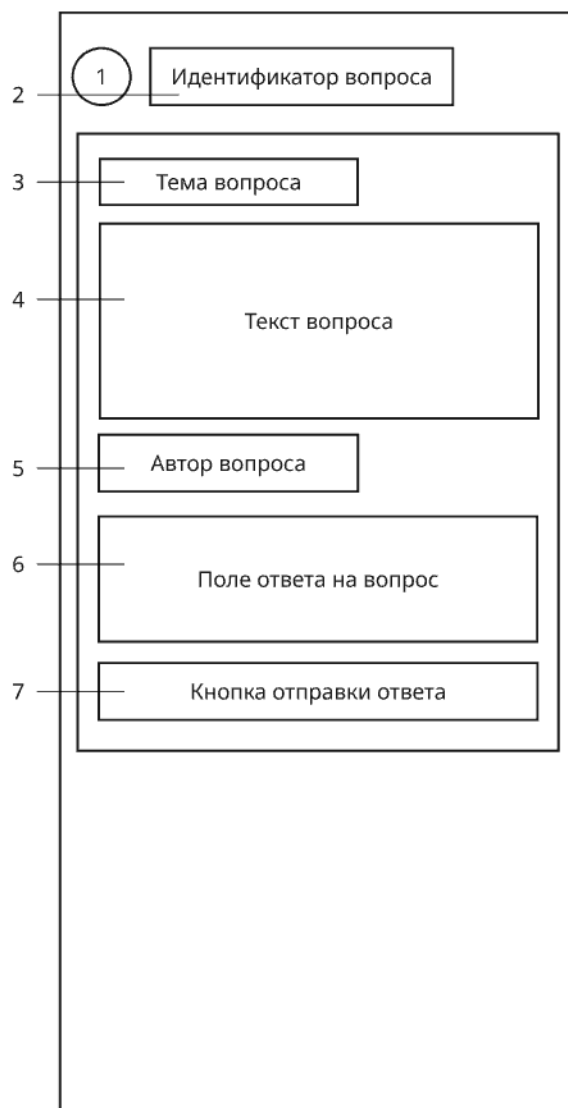


Рисунок 42 – Макет интерфейса ответа на вопрос

В макете интерфейса ответа на вопрос присутствуют следующие элементы, перечисленные ниже:

- 1 – кнопка возврата на предыдущую окно;
- 2 – текст с номером вопроса;
- 3 – текст с темой вопроса;
- 4 – текст вопроса;
- 5 – текст с указанием автора вопроса;
- 6 – поле ввода ответа на вопрос;
- 7 – кнопка отправки ответа.

Ниже на рисунке 43 представлен макет окна просмотра ответов на вопрос пользователем.

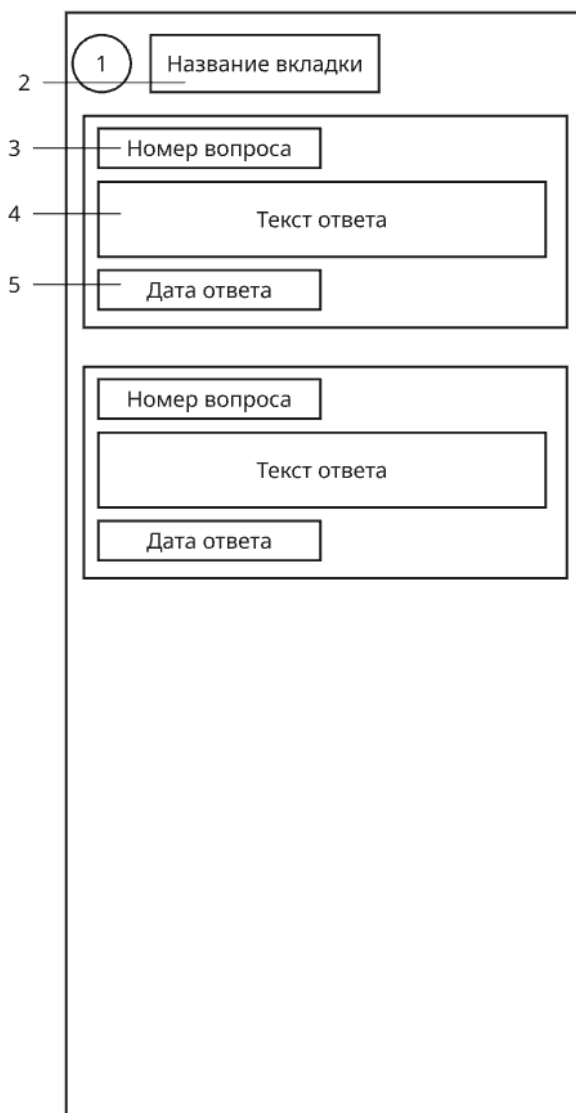


Рисунок 43 – Макет интерфейса просмотра ответов на вопрос пользователем

В макете интерфейса просмотра ответов на вопрос пользователем присутствуют следующие элементы, перечисленные ниже:

- 1 – кнопка возврата на предыдущую окно;
- 2 – текст с названием окна;
- 3 – текст с номером вопроса;
- 4 – текст ответа;
- 5 – дата ответа.

Ниже на рисунке 44 представлен макет окна личного кабинета пользователя.

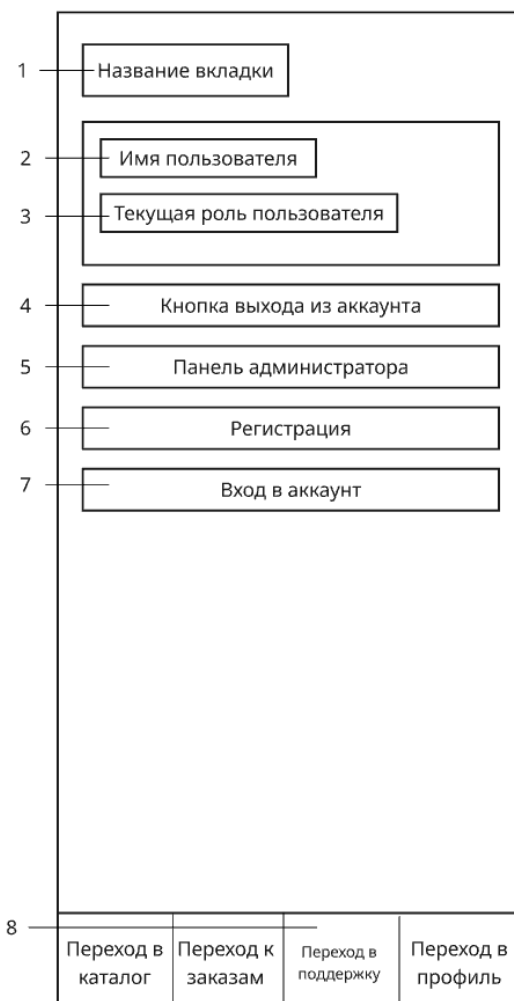


Рисунок 44 – Макет интерфейса личного кабинета пользователя

В макете интерфейса просмотра ответов на вопрос пользователем присутствуют следующие элементы, перечисленные ниже:

- 1 – текст с названием окна;
- 2 – текст с именем пользователя;
- 3 – текст с ролью пользователя;
- 4 – кнопка выхода из аккаунта;
- 5 – кнопка перехода в административную панель (только для администратора);
- 6 – переход к форме регистрации (только для гостя);
- 7 – переход к форме входа в аккаунт (только для гостя).

Ниже на рисунке 45 представлен макет окна регистрации.

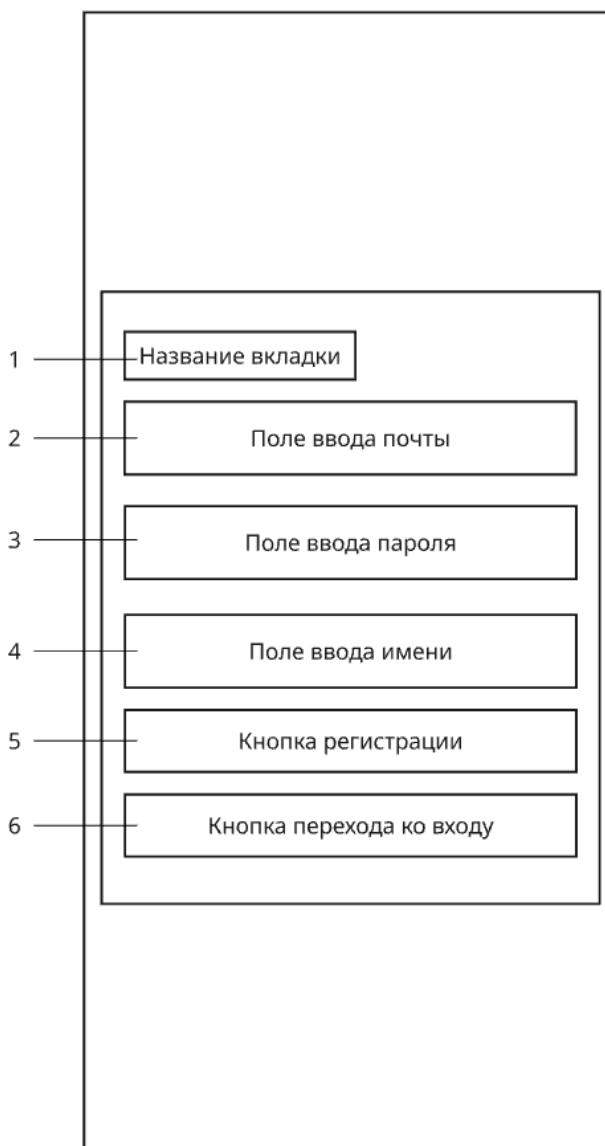


Рисунок 45 – Макет интерфейса окна регистрации

В макете интерфейса окна регистрации присутствуют следующие элементы, перечисленные ниже:

- 1 – текст с названием окна;
- 2 – поле ввода электронной почты;
- 3 – поле ввода пароля;
- 4 – поле ввода отображаемого имени;
- 5 – кнопка регистрации;
- 6 – переход к форме входа в аккаунт.

Ниже на рисунке 46 представлен макет окна входа в аккаунт.

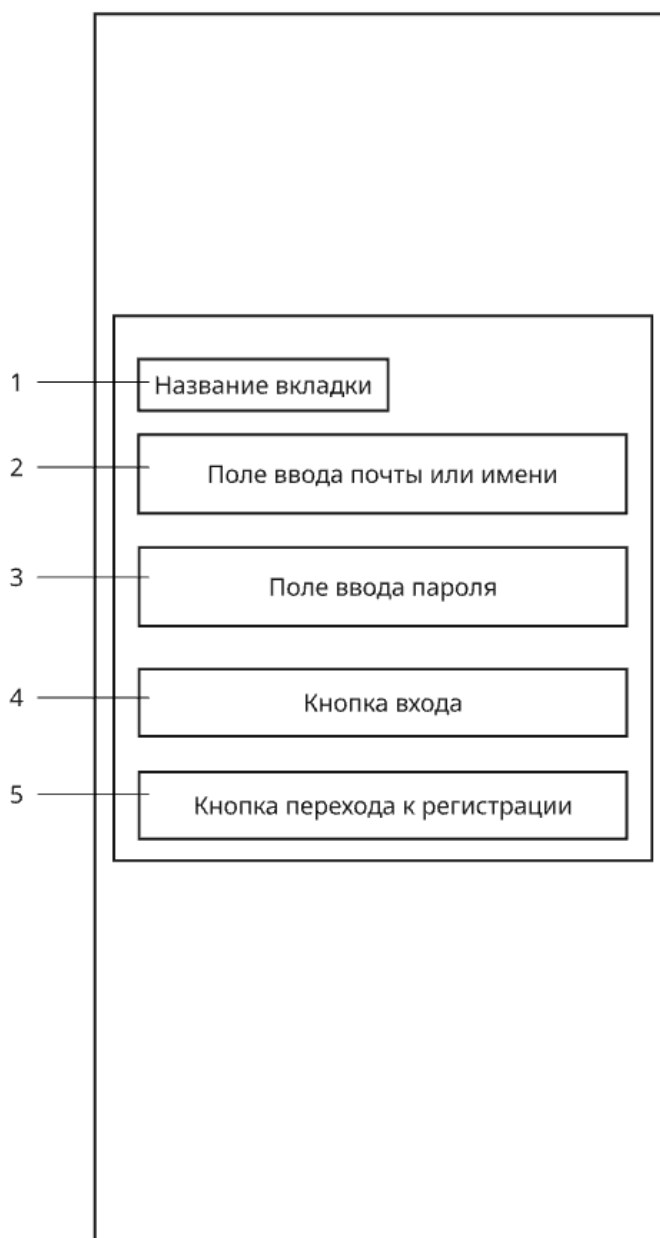


Рисунок 46 – Макет интерфейса окна входа в аккаунт

В макете интерфейса окна входа в аккаунт присутствуют следующие элементы, перечисленные ниже:

- 1 – текст с названием окна;
- 2 – поле ввода электронной почты или имени;
- 3 – поле ввода пароля;
- 4 – кнопка входа;
- 5 – переход к форме регистрации.

Ниже на рисунке 47 представлен макет окна административной панели.

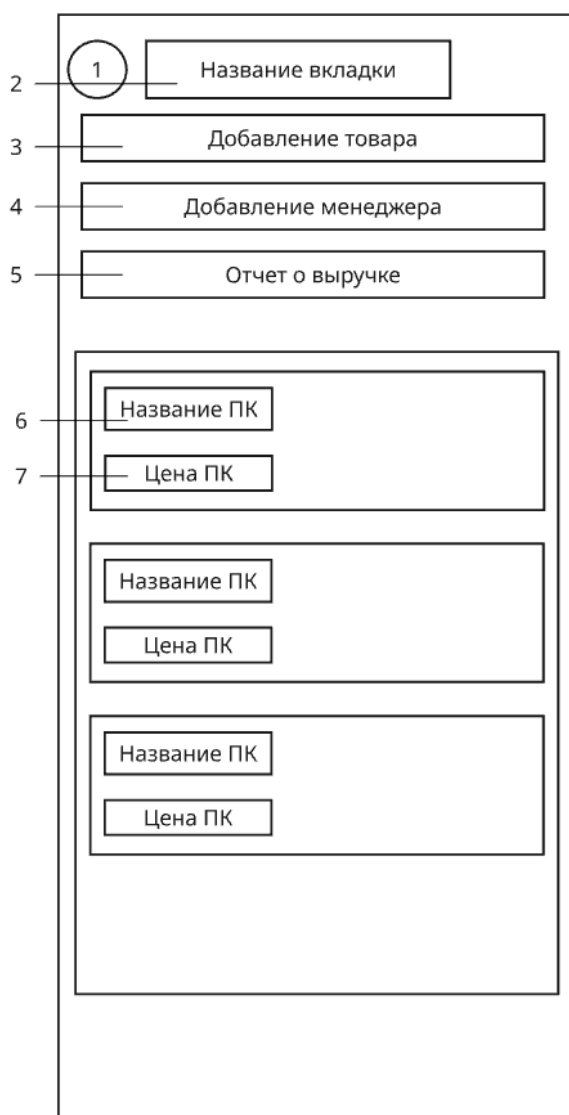


Рисунок 47 – Макет интерфейса окна административной панели

В макете интерфейса окна административной панели присутствуют следующие элементы, перечисленные ниже:

- 1 – кнопка перехода на предыдущее окно;
- 2 – текст с названием окна;
- 3 – кнопка перехода на форму добавления товара;
- 4 – кнопка перехода на форму добавления менеджера;
- 5 – переход к анализу выручки;
- 6 – название существующего ПК;
- 7 – цена существующего ПК.

При нажатии на любой существующий ПК будет совершен переход на форму редактирования ПК.

Ниже на рисунке 48 представлен макет окна добавления и изменения данных о ПК. Разница между окнами будет в том, что на окне изменения данных о ПК будут автоматически выбраны уже указанные характеристики.

Рисунок 48 – Макет интерфейса окна добавления или редактирования ПК

В макете интерфейса окна добавления или редактирования ПК присутствуют следующие элементы, перечисленные ниже:

- 1 – кнопка перехода на предыдущее окно;
- 2 – текст с названием окна;
- 3 – поле ввода названия ПК;
- 4 – поле выбора категории ПК;
- 5 – поле ввода описания ПК;

- 6 – поле ввода цены ПК;
- 7 – поле выбора процессора ПК;
- 8 – поле выбора типа ОЗУ ПК;
- 9 – поле выбора объема ОЗУ ПК;
- 10 – поле выбора видеокарты ПК;
- 11 – поле выбора объема диска ПК;
- 12 – поле выбора изображения для ПК;
- 13 – кнопка сохранения ПК в базе данных.

Ниже на рисунке 49 представлен макет окна добавления менеджера по работе с клиентами.

Рисунок 49 – Макет интерфейса окна добавления менеджера по работе с клиентами

В макете интерфейса окна добавления менеджера по работе с клиентами присутствуют следующие элементы, перечисленные ниже:

- 1 – кнопка перехода на предыдущее окно;

- 2 – текст с названием окна;
- 3 – поле ввода почты менеджера;
- 4 – поле ввода пароля менеджера;
- 5 – поле ввода имени менеджера;
- 6 – кнопка добавления менеджера в базу данных;
- 7 – почта существующего менеджера;
- 8 – имя существующего менеджера.

Ниже на рисунке 50 представлен макет окна просмотра данных о выручке за период времени.

The wireframe shows a window layout with the following elements:

- 1**: A circular button in the top-left corner.
- 2**: A text label "Название вкладки" (Tab name) next to the button.
- 3**: A text input field labeled "Дата начала расчета" (Calculation start date).
- 4**: A text input field labeled "Дата окончания расчета" (Calculation end date).
- 5**: A button labeled "Кнопка подтверждения расчета" (Calculation confirmation button).
- 6**: A text input field labeled "Общая выручка" (Total revenue).
- 7**: A text input field labeled "Количество заказов" (Number of orders).
- 8**: A table with 4 columns: "№" (Number), "Дата" (Date), "Сумма" (Sum), and "Статус" (Status). The table contains three rows of data.

Рисунок 50 – Макет интерфейса окна просмотра данных о выручке за период времени

В макете интерфейса окна просмотра данных о выручке за период времени присутствуют следующие элементы, перечисленные ниже:

- 1 – кнопка перехода на предыдущее окно;

- 2 – текст с названием окна;
- 3 – поле ввода начала расчета;
- 4 – поле ввода конца расчета;
- 5 – кнопка подтверждения расчета;
- 6 – текст с общей выручкой;
- 7 – текст количества заказов;
- 8 – данные отдельного заказа за указанный период.

Таким образом, на этапе проектирования были выбраны необходимые инструментальные средства разработки и определена архитектура программного обеспечения в виде трехуровневой клиент-серверной. Также были разработаны концептуальная и логическая модели базы данных, определяющий основные сущности, которыми будут оперировать, и их связи. Были подобраны архитектурные стили серверной части, играющие решающую роль в способе обмена данными клиентом и сервером. На заключительном подэтапе были разработаны макеты пользовательского интерфейса, визуализирующие функции пользователей, определенных на этапе функционального анализа приложения.

3 Реализация программного обеспечения

3.1 Реализация базы данных

На базе спроектированной логической модели была разработана физическая модель базы данных для СУБД PostgreSQL 18.2, она представлена ниже на рисунке 51.

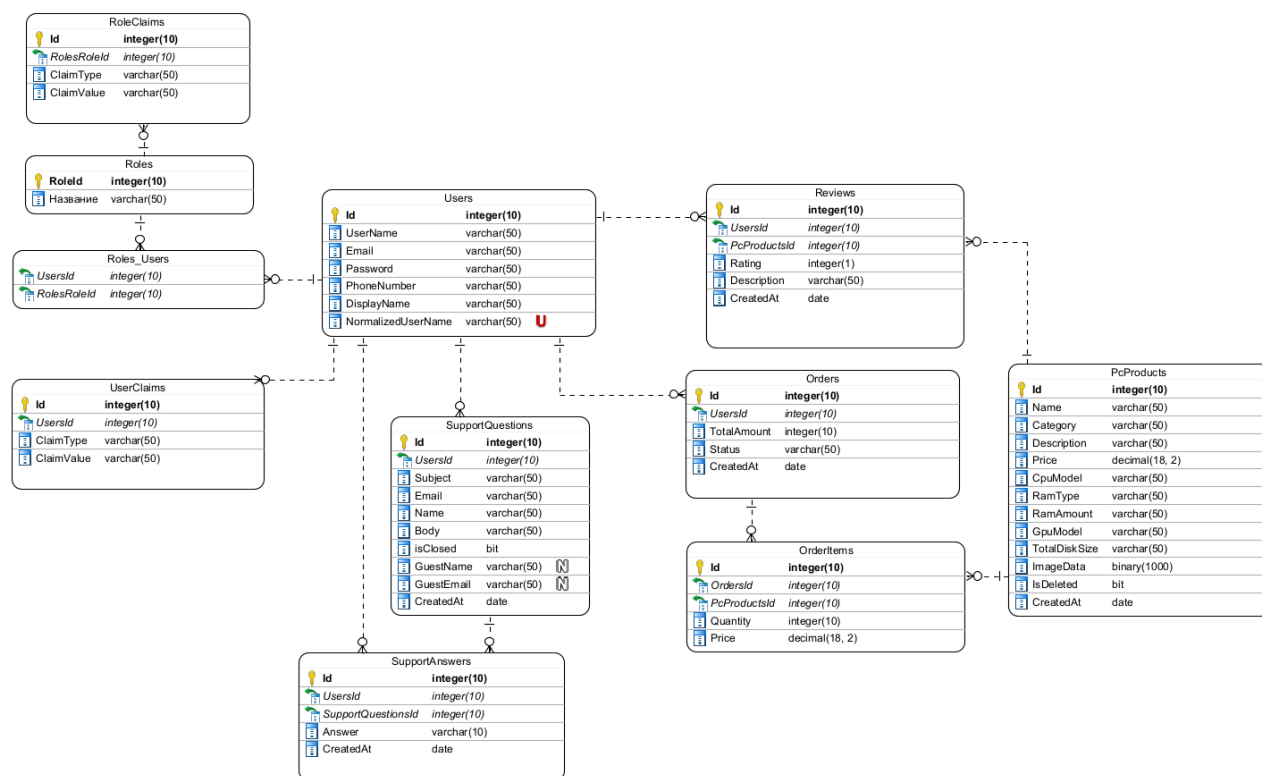


Рисунок 51 – Физическая модель базы данных

Ниже в таблицах приведено описание атрибутов сущностей физической модели базы данных, включая их название, описание, тип данных, уникальность, возможность значения быть пустым, является ли атрибут ключом [20-21]. Ниже в таблице 10 приведено описание атрибутов сущности «Users».

Таблица 10 – Описание атрибутов сущности «Users»

Название	Описание	Тип данных	Уникальность	Nullable	Ключ
Id	Идентификатор	Integer(10)	Да	Нет	PK
UserName	Логин	Varchar(50)	Нет	Нет	Нет

Продолжение таблицы 10

Название	Описание	Тип данных	Уникальность	Nullable	Ключ
Email	Почта	Varchar(50)	Нет	Нет	Нет
Password	Пароль	Varchar(50)	Нет	Нет	Нет
PhoneNumber	Номер телефона	Varchar(50)	Нет	Нет	Нет
DisplayName	Отображаемое имя	Varchar(50)	Нет	Нет	Нет
NormalizedUserName	Нормализованный логин	Varchar(50)	Да	Нет	Нет

Далее в таблице 11 приведено описание атрибутов сущности «Roles_Users».

Таблица 11 – Описание атрибутов сущности «Roles_Users»

Название	Описание	Тип данных	Уникальность	Nullable	Ключ
UsersId	Идентификатор пользователя	Integer(10)	Нет	Нет	FK
RolesRoleId	Идентификатор роли	Integer(10)	Нет	Нет	FK

Ниже в таблице 12 приведено описание атрибутов сущности «Roles».

Таблица 12 – Описание атрибутов сущности «Roles»

Название	Описание	Тип данных	Уникальность	Nullable	Ключ
RoleId	Идентификатор	Integer(10)	Да	Нет	PK
RoleName	Название	Varchar(50)	Да	Нет	Нет

Ниже в таблице 13 приведено описание атрибутов сущности «RoleClaims».

Таблица 13 – Описание атрибутов сущности «RoleClaims»

Название	Описание	Тип данных	Уникальность	Nullable	Ключ
Id	Идентификатор	Integer(10)	Да	Нет	PK
RolesRoleId	Идентификатор роли	Varchar(50)	Нет	Нет	FK

Продолжение таблицы 13

Название	Описание	Тип данных	Уникальность	Nullable	Ключ
ClaimType	Тип доступа	Varchar(50)	Нет	Нет	Нет
ClaimValue	Значение доступа	Varchar(50)	Нет	Нет	Нет

Ниже в таблице 14 приведено описание атрибутов сущности «UserClaims».

Таблица 14 – Описание атрибутов сущности «UserClaims»

Название	Описание	Тип данных	Уникальность	Nullable	Ключ
Id	Идентификатор	Integer(10)	Да	Нет	PK
UsersId	Идентификатор пользователя	Varchar(50)	Нет	Нет	FK
ClaimType	Тип доступа	Varchar(50)	Нет	Нет	Нет
ClaimValue	Значение доступа	Varchar(50)	Нет	Нет	Нет

Ниже в таблице 15 приведено описание атрибутов сущности «SupportQuestions».

Таблица 15 – Описание атрибутов сущности «SupportQuestions»

Название	Описание	Тип данных	Уникальность	Nullable	Ключ
Id	Идентификатор	Integer(10)	Да	Нет	PK
UsersId	Идентификатор пользователя	Varchar(50)	Нет	Нет	FK
ClaimType	Тип доступа	Varchar(50)	Нет	Нет	Нет
ClaimValue	Значение доступа	Varchar(50)	Нет	Нет	Нет
Subject	Тема	Varchar(50)	Нет	Нет	Нет
Email	Почта	Varchar(50)	Нет	Нет	Нет
Name	Имя	Varchar(50)	Нет	Нет	Нет
Body	Тело вопроса	Varchar(50)	Нет	Нет	Нет
IsClosed	Закрыт	Bit	Нет	Нет	Нет

Продолжение таблицы 15

Название	Описание	Тип данных	Уникальность	Nullable	Ключ
CuestName	Имя для гостя	Varchar(50)	Нет	Да	Нет
GuestEmail	Почта для гостя	Varchar(50)	Нет	Да	Нет
CreatedAt	Дата создания	Varchar(50)	Нет	Нет	Нет

Ниже в таблице 16 приведено описание атрибутов сущности «SupportAnswers».

Таблица 16 – Описание атрибутов сущности «SupportAnswers»

Название	Описание	Тип данных	Уникальность	Nullable	Ключ
Id	Идентификатор	Integer(10)	Да	Нет	PK
UsersId	Идентификатор пользователя	Varchar(50)	Нет	Нет	FK
SupportQuestionsId	Идентификатор вопроса	Varchar(50)	Нет	Нет	FK
Answer	Тело ответа	Varchar(50)	Нет	Нет	Нет
CreatedAt	Дата создания	Varchar(50)	Нет	Нет	Нет

Ниже в таблице 17 приведено описание атрибутов сущности «Orders».

Таблица 17 – Описание атрибутов сущности «Orders»

Название	Описание	Тип данных	Уникальность	Nullable	Ключ
Id	Идентификатор	Integer(10)	Да	Нет	PK
UsersId	Идентификатор пользователя	Varchar(50)	Нет	Нет	FK
TotalAmount	Сумма заказа	Decimal(18,2)	Нет	Нет	Нет
Status	Статус заказа	Varchar(50)	Нет	Нет	Нет
CreatedAt	Дата создания	Varchar(50)	Нет	Нет	Нет

Ниже в таблице 18 приведено описание атрибутов сущности «OrderItems».

Таблица 18 – Описание атрибутов сущности «OrderItems»

Название	Описание	Тип данных	Уникальность	Nullable	Ключ
Id	Идентификатор	Integer(10)	Да	Нет	PK
OrdersId	Идентификатор заказа	Integer (10)	Нет	Нет	FK
PcProductsId	Идентификатор компьютера	Integer(10)	Нет	Нет	FK
Quantity	Количество	Varchar(50)	Нет	Нет	Нет
Price	Цена	Varchar(50)	Нет	Нет	Нет

Ниже в таблице 19 приведено описание атрибутов сущности «Reviews».

Таблица 19 – Описание атрибутов сущности «Reviews»

Название	Описание	Тип данных	Уникальность	Nullable	Ключ
Id	Идентификатор	Integer(10)	Да	Нет	PK
UsersId	Идентификатор пользователя	Integer (10)	Нет	Нет	FK
PcProductsId	Идентификатор компьютера	Integer(10)	Нет	Нет	FK
Rating	Оценка	Varchar(50)	Нет	Нет	Нет
Description	Тело отзыва	Varchar(50)	Нет	Нет	Нет
CreatedAt	Дата создания	Varchar(50)	Нет	Нет	Нет

Ниже в таблице 20 приведено описание атрибутов сущности «PcProducts».

Таблица 20 – Описание атрибутов сущности «PcProducts»

Название	Описание	Тип данных	Уникальность	Nullable	Ключ
Id	Идентификатор	Integer(10)	Да	Нет	PK
Name	Идентификатор пользователя	Integer (10)	Нет	Нет	FK

Продолжение таблицы 20

Название	Описание	Тип данных	Уникальность	Nullable	Ключ
Category	Идентификатор компьютера	Integer(10)	Нет	Нет	FK
Rating	Оценка	Varchar(50)	Нет	Нет	Нет
Description	Тело отзыва	Varchar(50)	Нет	Нет	Нет
Price	Дата создания	Decimal(18,2)	Нет	Нет	Нет
CpuModel	Модель процессора	Varchar(50)	Нет	Нет	Нет
RamType	Тип ОЗУ	Varchar(50)	Нет	Нет	Нет
RamAmount	Количество ОЗУ	Varchar(50)	Нет	Нет	Нет
GpuModel	Модель видеокарты	Varchar(50)	Нет	Нет	Нет
TotalDiskSize	Объем диска	Varchar(50)	Нет	Нет	Нет
ImageData	Изображение	Binary(1000)	Нет	Нет	Нет
IsDeleted	Удален ли ПК	Varchar(50)	Нет	Нет	Нет
CreatedAt	Дата создания	Varchar(50)	Нет	Нет	Нет

Таким образом, была разработана физическая модель базы данных с учетом развертывания в СУБД PostgreSQL 18.2 и описаны сущности и атрибуты базы данных.

3.2 Реализация серверной части приложения

Разработка серверной части приложения проводилась в редакторе кода Visual Studio Code на популярном языке C# на универсальной платформе ASP.NET Core [22]. Серверная часть приложения ориентирована на обслуживание внешних клиентов, например мобильных приложений на Kotlin.

В качестве архитектурного подхода при разработке использовалась ориентация на модульный монолит. Это подразумевает работу приложения в одном процессе, при этом оно логически разбивается на модули, группируемые по семантике [19]. В приложении были выделены следующие модули: каталог

товаров, аутентификация, поддержка, административный модуль, модуль заказов и отзывов. При таком подходе наблюдается низкая связанность модулей между собой, что дает возможность плавного перехода к архитектуре на основе сервисов в случае необходимости [23]. Ниже на рисунке 52 представлена диаграмма классов для контроллеров модулей, перечисленных выше [24].

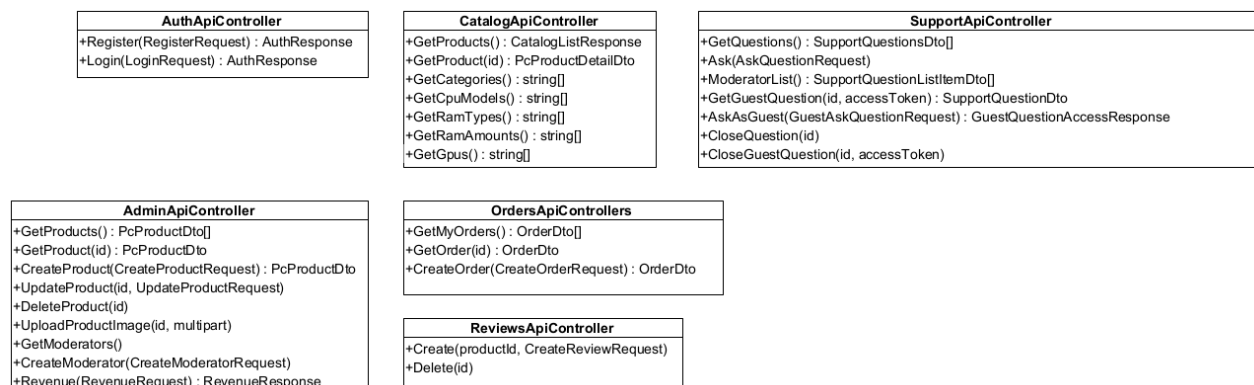


Рисунок 52 – Диаграмма классов контроллеров серверной части

Ниже в таблице 21 приведено описание классов контроллеров с их непосредственным назначением.

Таблица 21 – Описание классов контроллеров

Название	Описание
AuthApiController	Отвечает за запросы регистрации и входа в аккаунт.
CatalogApiController	Отвечает за отображение данных товаров, возможности фильтрации.
OrdersApiController	Отвечает за просмотр заказов пользователя и оформление заказов.
ReviewsApiController	Отвечает за публикацию и удаление отзывов на товары.
SupportApiController	Отвечает за создание и ответы на вопросы в поддержку.
AdminApiController	Отвечает за возможности администратора: создание, управление товарами, управление менеджерами, получение сводки о продажах.

Помимо этого, внутри модулей организованы модели данных, с которыми работает сервер, они представлены ниже на рисунке 53 в виде диаграммы классов [25].

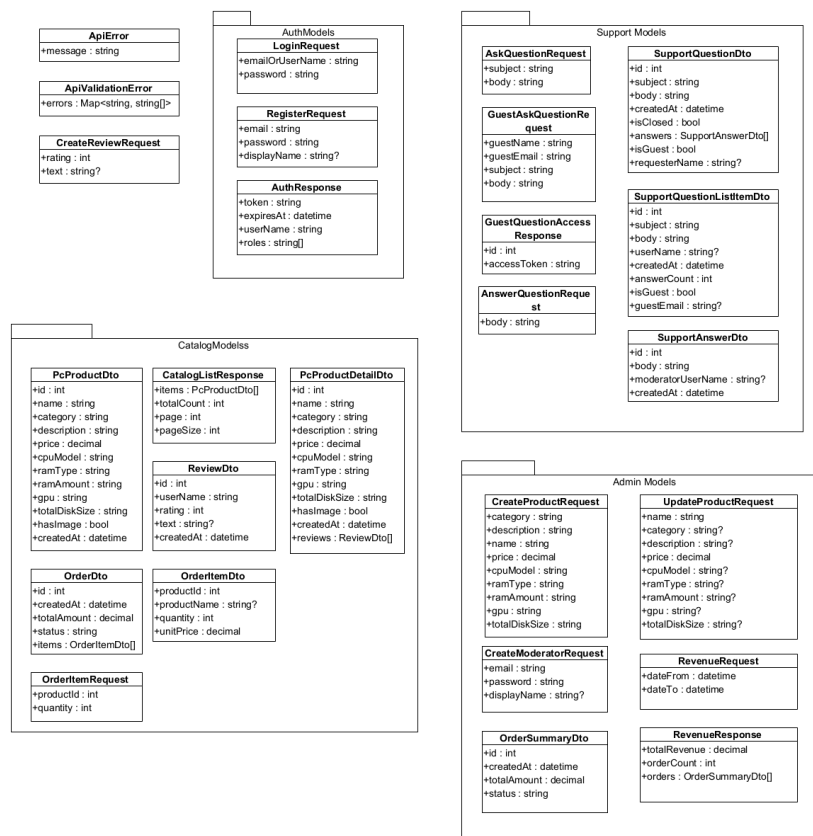


Рисунок 53 – Диаграмма классов моделей серверной части

Взаимодействие сервера с клиентами осуществляется на основе принципов подхода REST [26]. Обмен информацией работает через HTTP-запросы с телом вида JSON по конечным точкам `/api/*`, что определяет каждый отдельный ресурс на сервере. Такой подход обеспечивает независимость от платформы, что упрощает реализацию внешних клиентов при документировании API сервера.

С помощью ASP.NET Core сервер реализует конвейер HTTP-запросов, проходящих через соответствующие этапы обработки. Последовательно происходит проверка CORS-политик [27], маршрутизация запроса, аутентификация и авторизация. Затем запрос передается контроллеру, который взаимодействует с базой данных напрямую. Также ASP.NET Core реализует механизм DI: все основные зависимости определяются внутри точки входа в программу (Program.cs), то есть интерфейс `ICatalogService` связывается с реализацией `CatalogService`, что устраняет жесткую зависимость и позволяет

изменять реализацию без внесения изменения в код потребителей [28]. Пример реализации DI представлен ниже на рисунке 54.

Dependency Injection Architecture

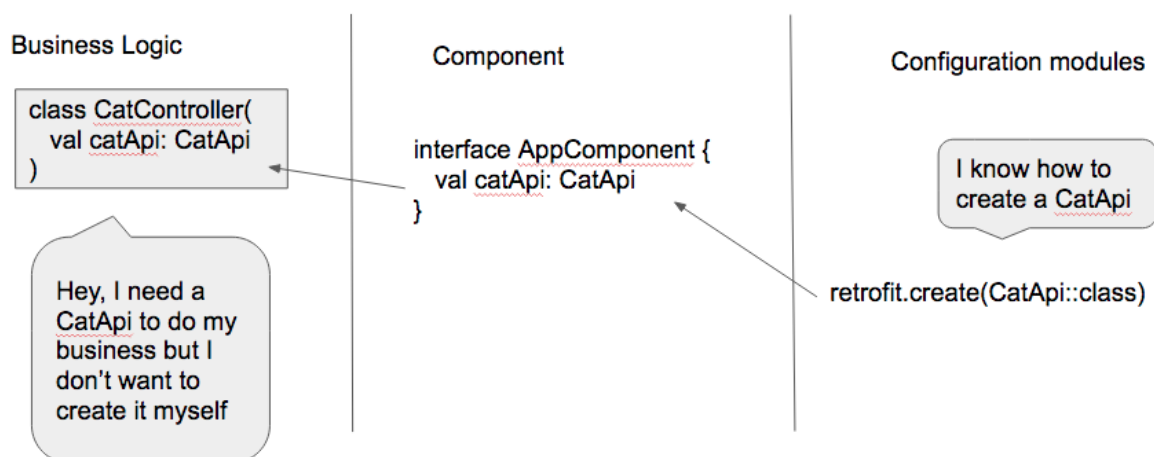


Рисунок 54 – Механизм DI на примере кода

Управление пользователями происходит с помощью ASP.NET Core Identity [29]. Данный пакет реализует правила регистрации, хранения данных пользователей, проверку пароля и систему ролей приложения (гость, зарегистрированный пользователь, менеджер по работе с клиентами, администратор).

Доступ к API реализуется с помощью RBAC [30]. У некоторых маршрутов имеется атрибут [Authorize], ограничивающий доступ. Права доступа встраиваются в JWT токен, механизм которого рассмотрен ниже.

Для работы с мобильными клиентами был реализован механизм JWT-аутентификации [31]. После успешного входа в приложение пользователь получает токен доступа, содержащий идентификатор и роли. При последующих запросах этот токен передается на сервер в заголовке запроса. После сверки подписи с помощью HMAC-SHA256 и срока действия сервер отдает нужные данные клиенту или отклоняет запрос [32]. Реализация JWT-аутентификации показана ниже на рисунке 55.

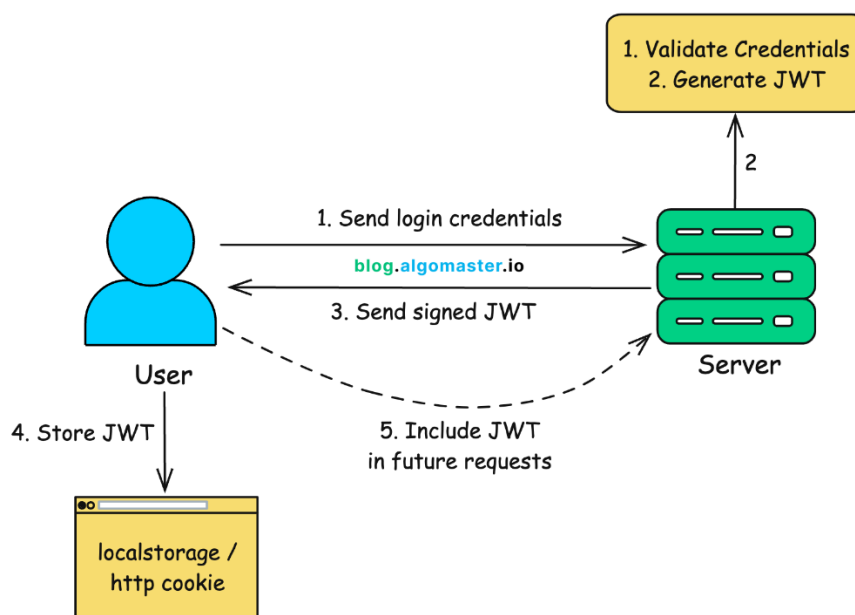


Рисунок 55 – Способ аутентификации с помощью JWT

Хранение данных реализуют Entity Framework Core вместе с PostgreSQL 18.2 [33-34]. На сервере реализуется объектно-реляционное отображение, что позволяет осуществить автоматическое сопоставление сущностей C# таблицам БД. Классом взаимодействия в данном случае является IdentityDbContext, объединяющий генерируемые таблицы с сущностями магазина. Таким образом, с помощью данных технологий появляется возможность использования LINQ-запросов для выборки данных вместо SQL [35]. Одной из особенностей PostgreSQL является «Enum type», используемый для описания характеристик товара, которые определены словами на уровне API [36]. Для преобразования используется специальный уровень PcProductOptions, выполняющий разбор строк и валидацию параметров. Также при удалении товара используется мягкое удаление – вместо удаления записи ставится флаг, скрывающий товар.

Сервер разворачивается с помощью контейнеризации Docker [37]. Сервер работает как отдельный контейнер и связан с контейнером PostgreSQL, который использует постоянный «диск» внутри докера, обеспечивающий хранение данных. На рисунке 56 представлен пример развертывания сервера в Docker.

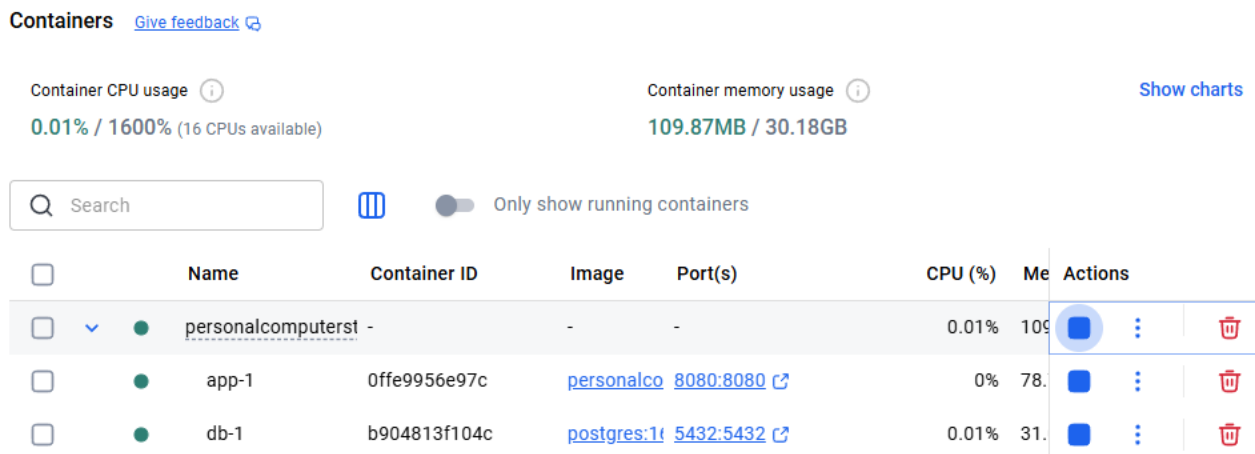


Рисунок 56 – Развертывание сервера в Docker Desktop

Таким образом, на данном этапе была реализована серверная часть приложения салона по продаже компьютеров. С помощью использованных методов разработки и технологий разработанный сервер является масштабируемым и легковесным, обеспечивает простоту изменения существующего кода и добавление нового, а также перехода на другую архитектуру. Листинг контроллеров приложения находится в приложении Б, а полный исходный код серверной части приложения можно увидеть в публичном репозитории GitHub [38].

3.3 Реализация клиентской части приложения

Разработка клиентского приложения проводилась на языке Kotlin для платформы Android и ориентировано на взаимодействие с сервером через REST API. Архитектура мобильного клиента построена на основе шаблонов MVVM с применением Repository, что обеспечивает разделение приложения на независимые уровни [39]. Пользовательский интерфейс отвечает за отображение и реакцию на действия пользователя, ViewModel отвечает за бизнес-логику, а Repository отвечает за работу с источником данных.

Графический интерфейс был реализован с помощью фреймворка JetpackCompose, где интерфейс описывается с помощью функций, обновляющихся автоматически при изменении состояния данных [40].

Навигация по экранам приложения реализована с помощью библиотеки Navigation Compose, в основе которой лежит принцип графа переходов между всеми существующими экранами.

Главным принципом реализации интерфейса является реактивная модель StateFlow [41]. Каждая ViewModel сохраняет состояние экрана в виде данных, которые отслеживаются интерфейсом, что предоставляет автоматическое обновление экрана при изменении данных. Ниже на рисунке 56 представлена диаграмма компонентов мобильного клиента салона по продаже ПК.

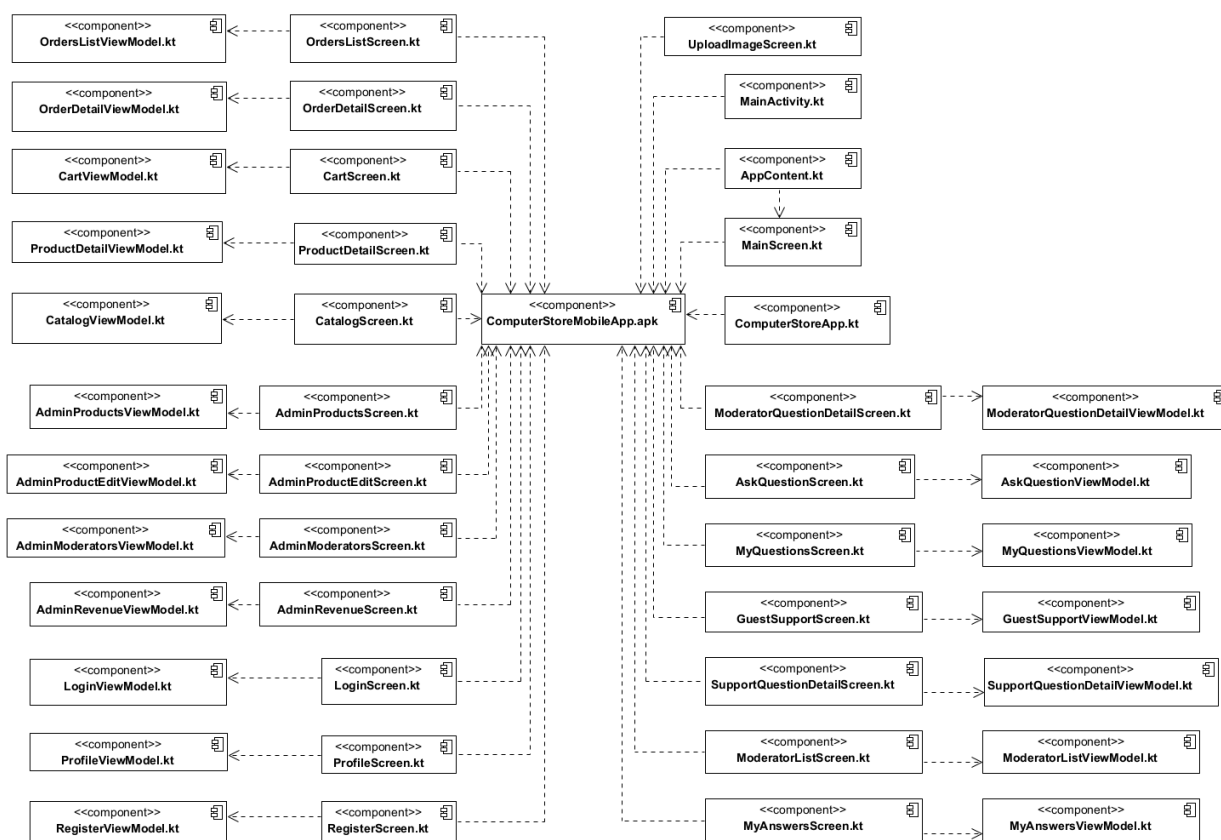


Рисунок 57 – Диаграмма компонентов мобильного приложения

Ниже в таблице 22 приведено описание функций компонентов.

Таблица 22 – Описание функций компонентов мобильного приложения

Название компонента	Функционал компонента
ComputerStoreMobileApp.kt	Собранное Android клиентское приложение
UploadImageScreen.kt	Вызов системного экрана выбора изображения

Продолжение таблицы 22

Название компонента	Функционал компонента
AppContent.kt	Корневой навигационный контейнер
MainScreen.kt	Отображение главных разделов приложения
MainActivity.kt	Точка входа в приложение
ComputerStoreApp.kt	Компонент инициализации приложения
ModeratorQuestionDetailScreen.kt	Экран ответа на вопрос
ModeratorQuestionDetailViewModel.kt	Логика обработки данных ответа на вопрос
AskQuestionScreen.kt	Экран отправки вопроса
AskQuestionViewModel.kt	Логика обработки отправки вопроса
MyQuestionsScreen.kt	Экран просмотра заданных вопросов
MyQuestionsViewModel.kt	Логика обработки данных заданных вопросов
GuestSupportScreen.kt	Экран поддержки для гостя
GuestSupportViewModel.kt	Логика обработки данных поддержки для гостя
SupportQuestionDetailScreen.kt	Экран просмотра заданного вопроса
SupportQuestionDetailViewModel.kt	Логика обработки данных заданного вопроса
ModeratorListScreen.kt	Экран просмотра всех вопросов
ModeratorListViewModel.kt	Логика обработки данных всех вопросов
MyAnswersScreen.kt	Экран просмотра ответов пользователя
MyAnswersViewModel.kt	Логика обработки данных ответов пользователя
OrdersListScreen.kt	Экран заказов
OrdersListViewModel.kt	Логика обработки данных заказов
OrderDetailScreen.kt	Экран деталей заказа
OrderDetailViewModel.kt	Логика обработки данных деталей заказа

Окончание таблицы 22

Название компонента	Функционал компонента
CartScreen.kt	Экран корзины
CartViewModel.kt	Логика обработки данных корзины
ProductDetailScreen.kt	Экран карточки товара
ProductDetailViewModel.kt	Логика обработки данных карточки товара
CatalogScreen.kt	Экран каталога
CatalogViewModel.kt	Логика обработки данных каталога
AdminProductsScreen.kt	Экран отображения товаров администратора
AdminProductsViewModel.kt	Логика обработки данных товаров администратора
AdminProductEditScreen.kt	Экран изменения товара
AdminProductEditViewModel.kt	Логика обработки данных изменения товара
AdminModeratorsScreen.kt	Экран управления менеджерами
AdminModeratorsViewModel.kt	Логика обработки данных управления менеджерами
AdminRevenueScreen.kt	Экран сводки о прибыли
AdminRevenueViewModel.kt	Логика обработки данных о прибыли
LoginScreen.kt	Экран входа в аккаунт
LoginViewModel.kt	Логика обработки входа в аккаунт
ProfileScreen.kt	Экран личного профиля
ProfileViewModel.kt	Логика обработки данных личного профиля
RegisterScreen.kt	Экран регистрации
RegisterViewModel.kt	Логика обработки данных регистрации

Таким образом, были описаны компоненты мобильного приложения.

Взаимодействие с сервером реализовано за счет библиотеки Retrofit, которая исполняет роль REST-клиента, который автоматически преобразует интерфейсы Kotlin в HTTP-запросы и связывает их с конечными точками [42]. Отправку запросов выполняет HTTP-клиент OkHttp, который управляет заголовками запроса, перехватывает трафик и устанавливает соединение [43]. Безопасность пользовательских данных реализуется с помощью шифрования JWT-токена.

Приложение поддерживает несколько сценариев взаимодействия с системой. Незарегистрированный пользователь может просматривать каталог и обращаться в поддержку без авторизации. Зарегистрированные пользователи получают доступ к заказам, корзине и отзывам. Менеджеры работают с системой обработки обращений пользователей, а администраторы – с CRUD-операциями каталога, загрузкой изображений товаров, управлением менеджерами и аналитикой выручки.

На основе разработанного мобильного приложения и сервера была построена диаграмма развертывания, она представлена ниже на рисунке 57.

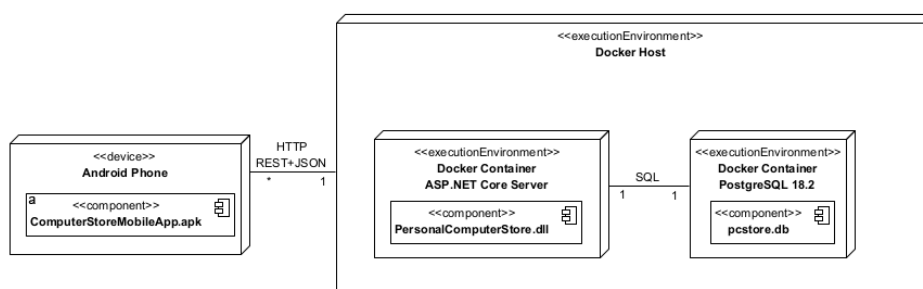


Рисунок 57 – Диаграмма развертывания мобильного приложения

Таким образом, на данном этапе был реализован мобильный клиент для салона по продаже персональных компьютеров с использованием Kotlin. Листинг файлов маршрутов конечных точек находится в приложении В, а полный исходный код мобильного приложения доступен в репозитории GitHub в свободном доступе [44].

4 Тестирование программного продукта

4.1 Функциональное тестирование черным ящиком

При тестировании черным ящиком проверяется работоспособность функций приложения с игнорированием непосредственной логики обработки данных запросов и фокусировкой на входных и выходных значениях и сравнения с ожидаемыми результатами.

Для проведения тестирования выделим необходимые границы, в рамках которых определяется правильная реакция программы на входные данные поля названия персонального компьютера. Они представлены в таблице 23.

Таблица 23 – Границы тестирования поля названия ПК

Границы	Описание
Физические	Строка отрицательной длины (Ф1)
Логические	Нулевая длина названия (Л1)
Технологические	Длина строки больше String.Length (2,147,483,647) = Int32.MaxValue (Т1)
Произвольные	Минимальная длина 10 (П1), максимальная длина 20 (П2), каждое слово начинается с заглавной буквы (П3), допустимые символы (кириллица, латиница, цифры, дефис, пробелы) (П4)

На основе выделенных границ проведем тестирование черным ящиком для данного поля. Результаты тестирования представлены ниже в таблице 24.

Таблица 24 – Результаты тестирования поля черным ящиком

№ теста	Тестируемые границы	Описание теста	Тестовые данные	Ожидаемый результат	Наблюдаемый результат	Тест пройден / не пройден
1	П1	Проверка перехвата короткого названия	Игровой	Появилось окно с описанием ошибки в названии	Появилось окно с описанием ошибки в названии (Рисунок Г.1)	Пройден
2	П2	Проверка перехвата длинного названия	Гипер компьютер для тяжелых игр	Появилось окно с описанием ошибки в названии	Появилось окно с описанием ошибки в названии (Рисунок Г.2)	Пройден

Продолжение таблицы 24

№ теста	Тестируемые границы	Описание теста	Тестовые данные	Ожидаемый результат	Наблюдаемый результат	Тест пройден / не пройден
3	ПЗ	Проверка перехвата названия со словами со строчных букв	офисный компьютер	Появилось окно с описанием ошибки в названии	Появилось окно с описанием ошибки в названии (Рисунок Г.3)	Пройден
4	П4	Проверка перехвата названия с недопустимыми символами	Перехватчик №1	Появилось окно с описанием ошибки в названии	Появилось окно с описанием ошибки в названии (Рисунок Г.4)	Пройден
5	Л1	Проверка названия нулевой длины	«»	Появилось окно с описанием ошибки в названии	Появилось окно с описанием ошибки в названии (Рисунок Г.5)	Пройден
6	П1, П2, ПЗ, П4	Проверка названия, соответствующего всем границам	Office Basic	Появилось окно об успешном добавлении товара	Появилось окно об успешном добавлении товара (Рисунок Г.6)	Пройден

На основе выходных данных можно сделать вывод о том, что программное обеспечение корректно обрабатывает входные данные с учетом возможных ошибок и обрабатывает их в соответствии с установленными рамками.

4.2 Модульное тестирование

Модульные тесты – это автоматизированные проверки наименьших частей кода (функций, методов, классов) в изоляции от остальной системы. Они подтверждают, что конкретный модуль работает корректно при различных входящих данных.

В рамках выпускной квалификационной работы были написаны тесты для покрытия серверной части. Их фрагмент приведен на рисунке 58, а результаты тестирования на рисунке 59.

```
[Fact]
public async Task CreateOrderAsync_IgnoresInvalidLinesAndCalculatesTotal()
{
    await using var db = CreateDbContext();
    var p1 = new PcProduct { Name = "A", Price = 1000m };
    var p2 = new PcProduct { Name = "B", Price = 250m };
    db.PcProducts.AddRange(p1, p2);
    await db.SaveChangesAsync();

    var sut = new OrderService(db);
    var order = await sut.CreateOrderAsync(
        "user-1",
        new List<(int ProductId, int Quantity)>
        {
            (p1.Id, 2),
            (p2.Id, 0),
            (99999, 1)
        });

    Assert.NotNull(order);
    Assert.Equal(OrderStatus.Pending, order.Status);
    Assert.Equal(2000m, order.TotalAmount);

    var savedItems = await db.OrderItems.Where(x => x.OrderId == order.Id).ToListAsync();
    Assert.Single(savedItems);
    Assert.Equal(p1.Id, savedItems[0].PcProductId);
    Assert.Equal(2, savedItems[0].Quantity);
    Assert.Equal(1000m, savedItems[0].UnitPrice);
}
```

Рисунок 58 – Фрагмент тестирования модуля заказов

На рисунке приведен фрагмент кода описания теста оформления заказа пользователем — задаются значения товаров и стоимости заказа.

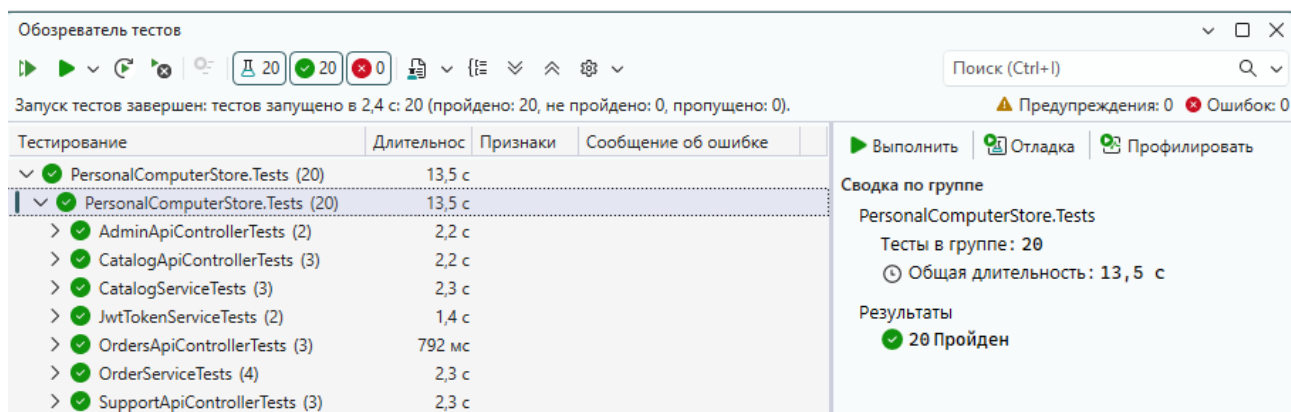


Рисунок 59 – Выполнение тестирования

На рисунке выше показано выполненное модульное тестирование серверной части с покрытием тестами в размере 20 штук тестовым классом PersonalComputeStore.Tests. Фрагмент кода тестов находится в листингах приложения Д, а полный исходный код тестов представлен в репозитории GitHub серверного приложения [38]. Все тесты были пройдены успешно, что подтверждает корректную работу разработанного программного обеспечения.

4.3 Метрики кода

Метрики кода позволяют количественно оценить заданные характеристики программного кода с целью его возможного улучшения в последующих итерациях разработки. В рамках расчета метрик приведем метод «CreateOrderAsync», содержащий логику обработки сохранения заказа пользователя. Его код представлен ниже в листинге 1.

Листинг 1 – Метод «CreateOrderAsync»

```
public async Task<Order?> CreateOrderAsync(string userId,
ReadOnlyList<(int ProductId, int Quantity)> items, CancellationToken
cancellation_token = default)
{
    if (items.Count == 0) return null;
    var productIds = items.Select(x => x.ProductId).Distinct().ToList();
    var products = await _db.PcProducts
        .Where(p => productIds.Contains(p.Id))
        .ToDictionaryAsync(p => p.Id, cancellation_token);
    decimal total = 0;
    var orderItems = new List<OrderItem>();
    foreach (var (productId, quantity) in items)
    {
        if (quantity <= 0) continue;
        if (!products.TryGetValue(productId, out var product)) continue;
        var unitPrice = product.Price;
        orderItems.Add(new OrderItem { PcProductId = productId, Quantity
= quantity, UnitPrice = unitPrice });
        total += unitPrice * quantity;
        if (orderItems.Count == 0) return null;
        var order = new Order
        {
            UserId = userId,
            TotalAmount = total,
            Status = OrderStatus.Pending
        };
        _db.Orders.Add(order);
        await _db.SaveChangesAsync(cancellation_token);
        foreach (var oi in orderItems)
        {
            oi.OrderId = order.Id;
            _db.OrderItems.Add(oi);
        }
        await _db.SaveChangesAsync(cancellation_token);
        return order;
    }
}
```

На основе вышеприведенного метода рассчитаем метрики SLOC и Холстеда. Расчеты приведены ниже в разделах 4.3.1 и 4.3.2 соответственно.

4.3.1 Метрика исходных строк кода

SLOC (Source Lines of Code) – метрика оценки объема написанного кода. Данная метрика разделяется на физическую и логическую: физическая считает все строки в файле, включая пустые и комментарии, логическая считает количество выполняемых инструкций (разделяемых точкой с запятой) [45]. Данная метрика позволяет быстро оценить размер кода и понять, является ли его объем избыточным и есть ли смысл разбивать его на более мелкие функциональные блоки.

Для расчета физического SLOC посчитаем все строки – их количество равно 33, что говорит о малом размере метода и легкости его восприятия.

Теперь рассчитаем логический SLOC, он будет включать объявление метода, объявления, присвоения, условия, циклы и вызовы с возвратом значения. Таким образом, можно насчитать 21 инструкцию, что говорит о компактной реализации метода и эффективном использовании данных с хорошей структурированностью.

4.3.2 Метрики Холстеда

Данные метрики помогают оценить сложность программы с разных сторон, чтобы лучше понять, какой подход выбирать в рамках последующей оптимизации, если она требуется [46]. Чтобы начать вычисления метрик, необходимо привести словарь операторов и операндов выбранного метода. Словарь операторов представлен ниже в таблице 25.

Таблица 25 – Словарь операторов метода

Операторы	Количество
public	1
async	1
task	1
await	3
return	3
if	3
foreach	2
continue	2
var	4

Продолжение таблицы 25

Операторы	Количество
decimal	1
new	3
out	1
=	9
<=	1
==	1
!	1
+=	1
*	1
=>	2
.	23
()	20
{}	7

Словарь операндов приведен ниже в таблице 26.

Таблица 26 – Словарь операндов

Операнды	Количество
Order	4
CreateOrderAsync	1
userId	3
string	1
items	4
ProductId	2
Quantity	4
cancellationToken	5
default	1
Count	2
0	3
null	2
productIds	3
Select	1
x	1
Distinct	1
ToList	1
products	3
db	6
PcProducts	1
Where	1
p	3
Id	3
Contains	1
ToDictionaryAsync	1
total	3
orderItems	4
List	1

Продолжение таблицы 26

Операнды	Количество
OrderItem	2
productId	4
quantity	5
TryGetValue	1
product	2
unitPrice	3
Price	1
Add	3
PcProductId	1
UnitPrice	1
OrderStatus.Pending	1
order	5
UserId	1
TotalAmount	1
Status	1
Orders	1
SaveChangesAsync	2
oi	3
OrderId	2
OrderItems	1

Рассчитаем N – длину программы, на основе $N1$ и $N2$ – общего числа операторов и операндов соответственно, исходя из значений таблиц.

$$N = N1 + N2 = 205, \text{ при } N1 = 92, N2 = 113,$$

где $N1$ – число вхождений всех операторов в программе;

$N2$ – число вхождений всех операндов программе.

Далее рассчитаем n – словарь программы, на основе $n1$ и $n2$ – числа уникальных операторов и операндов программы.

$$n = n1 + n2 = 72, \text{ при } n1 = 22, n2 = 50,$$

где $n1$ – количество операторов в программе;

$n2$ – количество операндов программе.

На основе полученных значений N и n можно сделать вывод о том, что длина метода и его словарь малы, что обеспечивает легкость понимания и доработки при необходимости.

Далее рассчитаем объем V метода:

$$V = N * \log_2 n = 205 * \log_2 72 = 1264.85,$$

где N – длина программы;

n – словарь программы.

Также рассчитаем сложность D :

$$D = (n1/2) * (N2/n2) = (22 / 2) * (113 / 50) = 24.86,$$

где $n1$ – количество операторов в программе;

$n2$ – количество операндов программе;

$N2$ – число вхождений всех операндов программе.

Сравнительно небольшие объем и сложность метода также говорят о том, что разработчик потратит меньше времени на понимание его логики.

В заключение рассчитаем количество усилий на написание метода:

$$E = D * V = 24.86 * 1264.85 = 31444.17,$$

где D – сложность программы;

V – объем программы.

На основе рассчитанных значений можно заключить, что количественные метрики Холстеда для данного метода показывают легкость его восприятия разработчиком, что дает возможность быстрой реакции для изменений. Следует отметить, что интерпретация значений выполнена на основе обобщенных практических рекомендаций, так как изначальная модель Холстеда не содержит фиксированных порогов качества [47].

Заключение

В ходе выполнения выпускной квалификационной работы было разработано программное мобильное приложение для поддержки функционирования процессов салона по продаже настольных компьютеров.

Был проведен анализ предметной области, в ходе которого были выделены основные функции разрабатываемого приложения. Также были рассмотрены аналоги программных продуктов и выделены их недостатки, на основе чего были выделены основные преимущества разрабатываемого приложения, отсутствующие у конкурентов. Помимо этого, был выполнен анализ целевой аудитории, показавший доли рынка мобильных устройств и операционных систем, среди которых лидировал Android, под который будет разрабатываться приложение. На заключительном этапе анализа было выполнено планирование проекта с учетом всех возможных затрат. На основе проведенных расчетов был выявлен срок разработки в 39 рабочих дней, а затраты составляют 261 260,00 рублей, что свидетельствует о экономической целесообразности данного проекта.

Этап проектирования был начат с выбора технологий и инструментальных средств разработки, затем последовал выбор архитектурного шаблона программного обеспечения, после чего были смоделированы концептуальная и логическая модели базы данных серверной составляющей приложения, спроектирована серверная часть и интерфейс мобильного приложения.

На этапе реализации программного обеспечения была описана физическая модель базы данных, на основе которой она была развернута в СУБД PostgreSQL 18. Серверная составляющая разрабатывалась с помощью эффективного объектно-ориентированного языка C# на платформе ASP.NET в редакторе кода Visual Studio Code. Мобильное приложение было разработано в среде Android Studio на языке Kotlin с набором инструментов Jetpack Compose.

Заключительным этапом стало тестирование разработанного приложения, в ходе которого функционал приложения был подвергнут проверке тестирования

черным ящиком для подтверждения корректной работы функций на практике, затем было применено модульное тестирование, изолирующее классы при проверке, что дает возможность быстро реагировать в случае появления ошибок. Также были рассчитаны метрики SLOC и Холстеда, которые показывают легкость восприятия кода.

Таким образом, все поставленные задачи выпускной квалификационной работы были успешно выполнены.

Разработанное приложение планируется в дальнейшем к публикации в магазине Google Play Store, RuStore. Также планируется портировать приложение на операционную систему iOS для публикации в магазине App Store.

Список использованных источников

1. Глобализация СМИ и масс-медиа России. [Электронный ресурс]. URL: <https://cyberleninka.ru/article/n/globalizatsiya-smi-i-mass-media-rossii> (дата обращения 10.04.2026).
2. Программа «Цифровая экономика Российской Федерации». Утверждена распоряжением Правительства Российской Федерации от 28 июля 2017 г. № 1632-р. [Электронный ресурс]. URL: <http://static.government.ru/media/files/9gFM4FHj4PsB79I5v7yLVuPgu4bvR7M0.pdf> (дата обращения 10.04.2026).
3. Мобильный трафик в России. [Электронный ресурс]. URL: [https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%9C%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%82%D1%80%D0%B0%D1%84%D0%B8%D0%BA_\(%D1%80%D0%BE%D1%81%D1%81%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%80%D1%8B%D0%BD%D0%BE%D0%BA\)](https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%9C%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%82%D1%80%D0%B0%D1%84%D0%B8%D0%BA_(%D1%80%D0%BE%D1%81%D1%81%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%80%D1%8B%D0%BD%D0%BE%D0%BA)) (дата обращения 10.04.2026).
4. Анализ магазина «HYPERPC». [Электронный ресурс]. URL: <https://hyperpc.ru> (дата обращения 05.04.2026).
5. Анализ магазина «Delta Game». [Электронный ресурс]. URL: <https://delta-game.ru/> (дата обращения 05.04.2026).
6. Анализ магазина «COMPSHOP». [Электронный ресурс]. URL: <https://compshop.store/> (дата обращения 05.04.2026).
7. Исследование данных рынка персональных компьютеров. [Электронный ресурс]. URL: [https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D1%8B_\(%D1%80%D1%8B%D0%BD%D0%BE%D0%BA_%D0%A0%D0%BE%D1%81%D1%81%D0%B8%D0%B8\)#.D0.A0.D1.8B.D0.BD.D0](https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D1%8B_(%D1%80%D1%8B%D0%BD%D0%BE%D0%BA_%D0%A0%D0%BE%D1%81%D1%81%D0%B8%D0%B8)#.D0.A0.D1.8B.D0.BD.D0)

BE.D0.BA_.D0.BF.D0.BB.D0.B0.D0.BD.D1.88.D0.B5.D1.82.D0.BE.D0.B2 (дата обращения 06.04.2026).

8. Статистика оборота бизнеса целевой аудитории. [Электронный ресурс]. URL: <https://tass.ru/ekonomika/25190135> (дата обращения 06.04.2026).

9. Мировые доли рынка мобильных операционных систем. [Электронный ресурс]. URL: <https://gs.statcounter.com/os-market-share/mobile/worldwide> (дата обращения 06.04.2026).

10. Российский рынок мобильных ОС. [Электронный ресурс]. URL: <https://cra.rip/articles/mobile-os-map/> (дата обращения 06.04.2026).

11. Отечественные аналоги ОС Android. [Электронный ресурс]. URL: <https://3dnews.ru/1125127/russian-mobile-os-review> (дата обращения 06.04.2026).

12. Распределение версий Android. [Электронный ресурс]. URL: <https://composables.com/android-distribution-chart> (дата обращения 07.04.2026).

13. Справка Google Play Store по публикации приложений. [Электронный ресурс]. URL: <https://developer.android.com/google/play/requirements/target-sdk?hl=ru> (дата обращения 07.04.2026).

14. Ехлаков, Ю.П. Управление программными проектами. Стандарты, модели: Учебное пособие / Ю.П. Ехлаков // СПб.: Лань, 2019. – 244 с.

15. Project Management Institute. Руководство PMBOK 6-е издание + Agile / PMI // М.: Олимп-Бизнес, 2022. – 974 с.

16. Фаулер, М. UML. Основы, 3-е издание / Мартин Фаулер // СПб.: Символ-Плюс, 2020. – 192 с.

17. Японская Ассоциация Менеджмента. Канбан и «точно вовремя» на Toyota. Менеджмент начинается на рабочем месте / Японская Ассоциация Менеджмента // М. Альпина Паблишер, 2022. – 214 с.

18. Типы клиент-серверной архитектуры. [Электронный ресурс]. URL: <https://servergate.ru/articles/klient-servernaya-arkhitektura/> (дата обращения 18.04.2026).

19. Марк, Р. Фундаментальный подход к программной архитектуре: паттерны, свойства, проверенные методы / Марк Ричардс, Форд Нил // СПб.: Питер, 2024. – 448 с.
20. Комаров, В. Путеводитель по базам данных / Комаров Владимир // ДМК Пресс, 2024. – 520 с.
21. Дейт, К. Введение в системы баз данных / Дейт Кристофер Дж. // Диалектика, 2024. – 1368 с.
22. Документация и применения ASP.NET Core. [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/dotnet/?WT.mc_id=dotnet-35129-website (дата обращения 10.05.2026).
23. Сравнения связности и зацепления. [Электронный ресурс]. URL: <https://habr.com/ru/articles/568216/> (дата обращения 10.05.2026).
24. Обзор контроллера MVC в ASP.NET. [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/dotnet/?WT.mc_id=dotnet-35129-website (дата обращения 10.05.2026).
25. Обзор модели MVC. [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/dotnet/?WT.mc_id=dotnet-35129-website (дата обращения 10.05.2026).
26. Принципы архитектуры REST. [Электронный ресурс]. URL: <https://habr.com/ru/articles/38730/> (дата обращения 12.05.2026).
27. Управление CORS в ASP.NET Core. [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/aspnet/core/security/cors?view=aspnetcore-10.0> (дата обращения 12.05.2026).
28. Роберт, М. Чистая архитектура. Искусство разработки программного обеспечения / Мартин Роберт // СПб.: Питер, 2022. – 509 с.
29. Сведения о Identity на ASP.NET Core. [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/aspnet/core/security/authentication/identity?view=aspnetcore-10.0&tabs=visual-studio> (дата обращения 12.05.2026).

30. Авторизация на основе ролей в ASP.NET Core. [Электронный ресурс]. URL: <https://learn.microsoft.com/en-us/aspnet/core/security/authorization/roles?view=aspnetcore-10.0> (дата обращения 12.05.2026).

31. Принципы управления JWT. [Электронный ресурс]. URL: <https://habr.com/ru/articles/842056/> (дата обращения 14.05.2026).

32. Проверка подлинности сообщений с помощью HMAC. [Электронный ресурс]. URL: <https://habr.com/ru/articles/262341/> (дата обращения 14.05.2026).

33. Сведения о Entity Framework Core. [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/ef/core/> (дата обращения 14.05.2026).

34. Документация PostgreSQL. [Электронный ресурс]. URL: <https://www.postgresql.org/docs/> (дата обращения 14.05.2026).

35. Описания языка запросов LINQ. [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/dotnet/csharp/linq/> (дата обращения 14.05.2026).

36. Enum type в PostgreSQL. [Электронный ресурс]. URL: <https://www.postgresql.org/docs/current/datatype-enum.html> (дата обращения 14.05.2026).

37. Основы управления Docker. [Электронный ресурс]. URL: <https://habr.com/ru/companies/netologyru/articles/967546/> (дата обращения 14.05.2026).

38. Мобильное программное приложение для поддержки процессов функционирования компьютерного салона, репозиторий серверной части. [Электронный ресурс]. URL: <https://github.com/Skiiper/ComputerStoreServerApp> (дата обращения 1.06.2026).

39. Архитектура MVVM и паттерн Repository на Kotlin. [Электронный ресурс]. URL: <https://habr.com/ru/articles/432826/> (дата обращения 17.05.2026).

40. Особенности набора инструментов Jetpack Compose. [Электронный ресурс]. URL: <https://habr.com/ru/articles/722040/> (дата обращения 17.05.2026).

41. Способы использования корутин StateFlow. [Электронный ресурс]. URL: <https://habr.com/ru/articles/501308/> (дата обращения 17.05.2026).

42. Использование библиотеки Retrofit с Kotlin. [Электронный ресурс]. URL: <https://habr.com/ru/articles/520544/> (дата обращения 17.05.2026).

43. Основные сведения о библиотеке OkHttp. [Электронный ресурс]. URL: <https://habr.com/ru/articles/717900/> (дата обращения 17.05.2026).

44. Мобильное программное приложение для поддержки процессов функционирования компьютерного салона, репозиторий мобильного приложения. [Электронный ресурс]. URL: <https://github.com/Skiipering/ComputerStoreMobileApp> (дата обращения 1.06.2026).

45. Сведения о метрике SLOC. [Электронный ресурс]. URL: <https://www.embeddedrelated.com/showarticle/454.php> (дата обращения 20.05.2026).

46. Информация о метриках Холстеда. [Электронный ресурс]. URL: <https://www.geeksforgeeks.org/software-engineering/software-engineering-halsteads-software-metrics/> (дата обращения 20.05.2026).

47. Прессман, Р. С. Software Engineering: A Practitioner's Approach / R. S. Pressman // Нью-Йорк: McGraw-Hill, 2010. – 780 с.

Приложение А
(справочное)
Глоссарий

SDK (Software Development Kit) – набор инструментов разработки программного обеспечения, включающий библиотеки, документацию, утилиты и примеры кода, предназначенные для создания приложений под конкретную платформу.

Agile – гибкая методология разработки ПО, основанная на итеративной поставке продукта, постоянной обратной связи и адаптации к изменениям требований.

Kanban – метод управления разработкой, основанный на визуализации задач, ограничении незавершённой работы (WIP) и непрерывном потоке выполнения задач.

ASP.NET Core – кроссплатформенный фреймворк для разработки веб-приложений и API на языке C# от Microsoft.

REST (Representational State Transfer) – архитектурный стиль проектирования распределённых систем, основанный на использовании HTTP и стандартных операций (GET, POST, PUT, DELETE).

JSON (JavaScript Object Notation) – текстовый формат обмена данными, используемый для представления структурированной информации в виде пар ключ–значение.

Entity Framework Core – ORM-фреймворк для .NET, обеспечивающий работу с базой данных через объектную модель без написания SQL-запросов вручную.

MVVM (Model-View-Viewmodel) – архитектурный шаблон, разделяющий логику представления, данных и бизнес-логики для упрощения тестирования и сопровождения приложений.

Jetpack Compose – декларативный UI-фреймворк для Android, позволяющий создавать интерфейсы на Kotlin без XML-разметки.

Сетевой граф проекта – графическая модель проекта, отображающая последовательность и зависимости между работами.

Критический путь проекта – последовательность задач в сетевом графе, определяющая минимальную длительность выполнения проекта.

Диаграмма Ганта – инструмент визуализации календарного плана проекта, отображающий задачи, их длительность и взаимосвязи.

UML (Unified Modeling Language) – унифицированный язык моделирования программных систем, используемый для описания структуры и поведения системы.

API (Application Programming Interface) – интерфейс взаимодействия между программными компонентами, определяющий способы обмена данными и вызова функций.

HTTP (HyperText Transfer Protocol) – протокол передачи данных в вебе, основанный на модели запрос–ответ между клиентом и сервером.

JWT (JSON Web Token) – стандарт безопасной передачи информации в виде токена, содержащего зашифрованные данные пользователя и подпись.

Identity ASP.NET – система управления пользователями в ASP.NET, обеспечивающая аутентификацию, авторизацию и управление ролями.

MVC (Model-View-Controller) – архитектурный шаблон, разделяющий приложение на модель, представление и контроллер для упрощения разработки.

MVC контроллер – компонент MVC, отвечающий за обработку входящих запросов и управление логикой взаимодействия между моделью и представлением.

MVC модель – компонент MVC, отвечающий за бизнес-логику и данные приложения.

Конечная точка – URL-адрес в API, через который осуществляется доступ к определённой функции сервера.

CORS (Cross-Origin Resource Sharing) – механизм, определяющий правила доступа к ресурсам сервера с других доменов.

Маршрутизация запроса – процесс определения обработчика (контроллера/метода) на основе URL входящего HTTP-запроса.

DI (Dependency Injection) – паттерн внедрения зависимостей, при котором объекты получают свои зависимости извне, а не создают их самостоятельно.

RBAC (Role-Based Access Control) – модель управления доступом, основанная на назначении прав пользователям через роли.

HMAC-SHA256 – алгоритм хеширования с секретным ключом, используемый для проверки целостности и подлинности данных.

LINQ (Language Integrated Query) – технология .NET для выполнения запросов к данным прямо в коде C#.

SQL (Structured Query Language) – язык структурированных запросов для управления и обработки данных в реляционных базах данных.

Docker – платформа контейнеризации приложений, позволяющая упаковывать приложение и его зависимости в изолированные контейнеры.

StateFlow (Kotlin) – реактивный поток данных из библиотеки Kotlin Coroutines, используемый для хранения и наблюдения состояния UI.

Retrofit – HTTP-клиент для Android, упрощающий работу с REST API.

OkHttp – низкоуровневая HTTP-библиотека для Android и Java, используемая для сетевых запросов.

EncryptedSharedPreferences – механизм Android для безопасного хранения данных с автоматическим шифрованием.

CRUD (Create, Read, Update, Delete) – базовые операции над данными в системах хранения информации.

Тестирование черным ящиком – метод тестирования программ, при котором проверяется только функциональность без анализа внутренней структуры кода.

Модульное тестирование – вид тестирования, при котором проверяются отдельные модули или функции программы изолированно от системы.

Приложение Б
(обязательное)
Исходный код серверной части

Листинг Б.1 – Файл AdminApiController.cs

```

using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using PersonalComputerStore.Api.Dto;
using PersonalComputerStore.Data;
using PersonalComputerStore.Services;

namespace PersonalComputerStore.Api;

[ApiController]
[Route("api/admin")]
[Authorize(AuthenticationSchemes = ApiAuthConstants.Schemes, Roles = "Administrator")]
public class AdminApiController(AppDbContext db, ICatalogService catalog, UserManager<ApplicationUser> userManager) :
    ControllerBase
{
    private readonly AppDbContext _db = db;
    private readonly ICatalogService _catalog = catalog;
    private readonly UserManager<ApplicationUser> _userManager = userManager;

    [HttpGet("products")]
    public async Task<ActionResult<IReadOnlyList<PcProductDto>>>
        GetProducts(CancellationToken cancellationToken = default)
    {
        var list = await
            _db.PcProducts.IgnoreQueryFilters().OrderBy(p =>
                p.Name).AsNoTracking().ToListAsync(cancellationToken);
        var dtos = list.Select(p => new PcProductDto(
            p.Id, p.Name, PcProductOptions.ToLabel(p.Category),
            p.Description, p.Price,
            PcProductOptions.ToLabel(p.CpuModel),
            PcProductOptions.ToLabel(p.RamType),
            PcProductOptions.ToLabel(p.RamAmount),
            PcProductOptions.ToLabel(p.Gpu),
            PcProductOptions.ToLabel(p.TotalDiskSize),
            p.ImageData != null && p.ImageData.Length > 0,
            p.CreatedAt)).ToList();
        return Ok(dtos);
    }

    [HttpGet("products/{id:int}")]
    public async Task<ActionResult<PcProductDto>> GetProduct(int
        id, CancellationToken cancellationToken = default)
    {
        var p = await
            _db.PcProducts.IgnoreQueryFilters().AsNoTracking().FirstOrDefaultAsync(x => x.Id == id, cancellationToken);
    }
}

```

```

        if (p == null) return NotFound(new ApiError("Product not
found."));
        return Ok(new PcProductDto(p.Id, p.Name,
PcProductOptions.ToLabel(p.Category), p.Description, p.Price,
PcProductOptions.ToLabel(p.CpuModel),
PcProductOptions.ToLabel(p.RamType),
PcProductOptions.ToLabel(p.RamAmount),
PcProductOptions.ToLabel(p.Gpu),
PcProductOptions.ToLabel(p.TotalDiskSize), p.ImageData != null &&
p.ImageData.Length > 0, p.CreatedAt));
    }

    [HttpPost("products")]
    public async Task<IActionResult> CreateProduct([FromBody]
CreateProductRequest request, CancellationToken cancellationTok
= default)
    {
        if (!TryParseSpecs(request.Category, request.CpuModel,
request.RamType, request.RamAmount, request.Gpu,
request.TotalDiskSize,
            out var category, out var cpuModel, out var
ramType, out var ramAmount, out var gpu, out var diskSize, out var
validationError))
            return BadRequest(validationError);

        var product = new PcProduct
        {
            Name = request.Name,
            Category = category,
            Description = request.Description,
            Price = request.Price,
            CpuModel = cpuModel,
            RamType = ramType,
            RamAmount = ramAmount,
            Gpu = gpu,
            TotalDiskSize = diskSize
        };
        _db.PcProducts.Add(product);
        await _db.SaveChangesAsync(cancellationToken);
        return CreatedAtAction(nameof(GetProduct), new { id =
product.Id }, new PcProductDto(product.Id, product.Name,
PcProductOptions.ToLabel(product.Category), product.Description,
product.Price, PcProductOptions.ToLabel(product.CpuModel),
PcProductOptions.ToLabel(product.RamType),
PcProductOptions.ToLabel(product.RamAmount),
PcProductOptions.ToLabel(product.Gpu),
PcProductOptions.ToLabel(product.TotalDiskSize), false,
product.CreatedAt));
    }

    [HttpPut("products/{id:int}")]

```

```

    public async Task<IActionResult> UpdateProduct(int id,
[FromBody] UpdateProductRequest request, CancellationToken
cancellation_token = default)
    {
        if (!TryParseSpecs(request.Category, request.CpuModel,
request.RamType, request.RamAmount, request.Gpu,
request.TotalDiskSize,
            out var category, out var cpuModel, out var
ramType, out var ramAmount, out var gpu, out var diskSize, out var
validationError))
            return BadRequest(validationError);

        var existing = await
_db.PcProducts.IgnoreQueryFilters().FirstOrDefaultAsync(x => x.Id
== id, cancellation_token);
        if (existing == null) return NotFound(new
ApiError("Product not found."));
        existing.Name = request.Name;
        existing.Category = category;
        existing.Description = request.Description;
        existing.Price = request.Price;
        existing.CpuModel = cpuModel;
        existing.RamType = ramType;
        existing.RamAmount = ramAmount;
        existing.Gpu = gpu;
        existing.TotalDiskSize = diskSize;
        await _db.SaveChangesAsync(cancellation_token);
        return NoContent();
    }

    private static bool TryParseSpecs(
        string? categoryValue,
        string? cpuValue,
        string? ramTypeValue,
        string? ramAmountValue,
        string? gpuValue,
        string? diskValue,
        out ProductCategory? category,
        out CpuModel? cpuModel,
        out RamType? ramType,
        out RamAmount? ramAmount,
        out GpuModel? gpu,
        out DiskSize? disk,
        out ApiValidationError? validationError)
    {
        var errors = new Dictionary<string, string[]>();
        if (!PcProductOptions.TryParseCategory(categoryValue, out
category))
            errors["Category"] = ["Invalid category. Use one of
the allowed enum labels."];
        if (!PcProductOptions.TryParseCpuModel(cpuValue, out
cpuModel))

```

```

        errors["CpuModel"] = ["Invalid CPU model. Use one of
the allowed enum labels."];
        if (!PcProductOptions.TryParseRamType(ramTypeValue, out
ramType))
            errors["RamType"] = ["Invalid RAM type. Use one of the
allowed enum labels."];
        if (!PcProductOptions.TryParseRamAmount(ramAmountValue,
out ramAmount))
            errors["RamAmount"] = ["Invalid RAM amount. Use one of
the allowed enum labels."];
        if (!PcProductOptions.TryParseGpu(gpuValue, out gpu))
            errors["Gpu"] = ["Invalid GPU model. Use one of the
allowed enum labels."];
        if (!PcProductOptions.TryParseDiskSize(diskValue, out
disk))
            errors["TotalDiskSize"] = ["Invalid disk size. Use one
of the allowed enum labels."];

        if (errors.Count == 0)
        {
            validationError = null;
            return true;
        }

        validationError = new ApiValidationErrors(errors);
        return false;
    }

    [HttpDelete("products/{id:int}")]
    public async Task<IActionResult> DeleteProduct(int id,
CancellationTokens cancellationTokens = default)
    {
        var p = await
_db.PcProducts.IgnoreQueryFilters().FirstOrDefaultAsync(x => x.Id
== id, cancellationTokens);
        if (p == null) return NotFound(new ApiError("Product not
found.));
        p.IsDeleted = true;
        await _db.SaveChangesAsync(cancellationTokens);
        return NoContent();
    }

    [HttpPost("products/{id:int}/image")]
    [Consumes("multipart/form-data")]
    public async Task<IActionResult> UploadProductImage(int id,
IFormFile? image, CancellationTokens cancellationTokens = default)
    {
        var error = ProductImageService.ValidatePng(image);
        if (error != null) return BadRequest(new ApiError(error));
        if (image == null || image.Length == 0) return
BadRequest(new ApiError("No image file provided.));
    }

```

```

        var imageData = await
ProductImageService.ReadPngBytesAsync(image, cancellationTokens);
        if (imageData == null) return BadRequest(new
ApiError("File is not a valid PNG.));
        var p = await
_db.PcProducts.IgnoreQueryFilters().FirstOrDefaultAsync(x => x.Id
== id, cancellationTokens);
        if (p == null) return NotFound(new ApiError("Product not
found.));
        p.ImageData = imageData;
        await _db.SaveChangesAsync(cancellationTokens);
        return NoContent();
    }

    [HttpGet("moderators")]
    public async Task<ActionResult<object>>
GetModerators(CancellationTokens cancellationTokens = default)
    {
        var users = await
_userManager.GetUsersInRoleAsync("Moderator");
        var list = users.Select(u => new { u.Id, u.Email,
u.UserName, u.DisplayName }).ToList();
        return Ok(list);
    }

    [HttpPost("moderators")]
    public async Task<IAActionResult> CreateModerator([FromBody]
CreateModeratorRequest request, CancellationTokens
cancellationTokens = default)
    {
        var user = new ApplicationUser { UserName = request.Email,
Email = request.Email, DisplayName = request.DisplayName,
EmailConfirmed = true };
        var result = await _userManager.CreateAsync(user,
request.Password);
        if (!result.Succeeded)
            return BadRequest(new
ApiValidationError(result.Errors.GroupBy(e =>
e.Code).ToDictionary(g => g.Key, g => g.Select(e =>
e.Description).ToArray())));
        await _userManager.AddToRoleAsync(user, "Moderator");
        return StatusCode(201, new { id = user.Id, email =
user.Email });
    }

    [HttpPost("revenue")]
    public async Task<ActionResult<RevenueResponse>>
Revenue([FromBody] RevenueRequest request, CancellationTokens
cancellationTokens = default)
    {
        if (request.DateFrom > request.DateTo)

```

```

        return BadRequest(new ApiError("DateFrom must be
before or equal to DateTo.));
        var from = DateTime.SpecifyKind(request.DateFrom.Date,
DateTimeKind.Utc);
        var to =
DateTime.SpecifyKind(request.DateTo.Date.AddDays(1),
DateTimeKind.Utc);
        var orders = await _db.Orders
            .AsNoTracking()
            .Where(o => o.CreatedAt >= from && o.CreatedAt < to &&
(o.Status == OrderStatus.Completed || o.Status ==
OrderStatus.Paid))
            .OrderBy(o => o.CreatedAt)
            .ToListAsync(cancellationToken);
        var total = orders.Sum(o => o.TotalAmount);
        var orderDtos = orders.Select(o => new
OrderSummaryDto(o.Id, o.CreatedAt, o.TotalAmount,
o.Status)).ToList();
        return Ok(new RevenueResponse(total, orders.Count,
orderDtos));
    }
}

```

Листинг Б.2 – Файл AuthApiController.cs

```

using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using PersonalComputerStore.Api.Dto;
using PersonalComputerStore.Data;
using PersonalComputerStore.Services;

namespace PersonalComputerStore.Api;

[ApiController]
[Route("api/auth")]
public class AuthApiController(
    UserManager<ApplicationUser> userManager,
    SignInManager<ApplicationUser> signInManager,
    IJwtTokenService jwt,
    IConfiguration config) : ControllerBase
{
    private readonly UserManager<ApplicationUser> _userManager =
userManager;
    private readonly SignInManager<ApplicationUser> _signInManager
= signInManager;
    private readonly IJwtTokenService _jwt = jwt;
    private readonly IConfiguration _config = config;

    [HttpPost("register")]
    [AllowAnonymous]

```

```

        public async Task<IActionResult> Register([FromBody]
RegisterRequest request, CancellationToken cancellationToken =
default)
        {
            var user = new ApplicationUser
            {
                UserName = request.Email,
                Email = request.Email,
                DisplayName = request.DisplayName
            };
            var result = await _userManager.CreateAsync(user,
request.Password);
            if (!result.Succeeded)
                return BadRequest(new
ApiValidationErrors(result.Errors.GroupBy(e =>
e.Code).ToDictionary(g => g.Key, g => g.Select(e =>
e.Description).ToArray())));
            await _userManager.AddToRoleAsync(user, "RegisteredUser");
            var token = await _jwt.GenerateTokenAsync(user,
cancellationToken);
            var expMinutes =
int.TryParse(_config["Jwt:ExpirationMinutes"], out var m) ? m :
60;
            var roles = await _userManager.GetRolesAsync(user);
            return StatusCode(201, new AuthResponse(token,
user.UserName ?? user.Email ?? "",
DateTime.UtcNow.AddMinutes(expMinutes), [... roles]));
        }

        [HttpPost("login")]
        [AllowAnonymous]
        public async Task<IActionResult> Login([FromBody] LoginRequest
request, CancellationToken cancellationToken = default)
        {
            var user = await
_userManager.FindByNameAsync(request.EmailOrUserName)
                ?? await
_userManager.FindByEmailAsync(request.EmailOrUserName);
            if (user == null)
                return Unauthorized(new ApiError("Invalid
credentials.));
            var result = await
_signInManager.CheckPasswordSignInAsync(user, request.Password,
lockoutOnFailure: false);
            if (!result.Succeeded)
                return Unauthorized(new ApiError("Invalid
credentials.));
            var token = await _jwt.GenerateTokenAsync(user,
cancellationToken);
            var expMinutes =
int.TryParse(_config["Jwt:ExpirationMinutes"], out var m) ? m :
60;

```

```

        var roles = await _userManager.GetRolesAsync(user);
        return Ok(new AuthResponse(token, user.UserName ??
user.Email ?? "", DateTime.UtcNow.AddMinutes(expMinutes), [...
roles]));
    }
}

```

Листинг Б.3 – Файл CatalogApiController.cs

```

using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using PersonalComputerStore.Api.Dto;
using PersonalComputerStore.Data;
using PersonalComputerStore.Services;

namespace PersonalComputerStore.Api;

[ApiController]
[Route("api/catalog")]
public class CatalogApiController(ICatalogService catalog,
AppDbContext db) : ControllerBase
{
    private const int DefaultPageSize = 20;
    private readonly ICatalogService _catalog = catalog;
    private readonly AppDbContext _db = db;

    [HttpGet]
    [AllowAnonymous]
    public async Task<ActionResult<CatalogListResponse>>
GetProducts(
    [FromQuery] string? search,
    [FromQuery] string? category,
    [FromQuery] decimal? minPrice,
    [FromQuery] decimal? maxPrice,
    [FromQuery] string? cpuModel,
    [FromQuery] string? ramType,
    [FromQuery] string? ramAmount,
    [FromQuery] string? gpu,
    [FromQuery] string? totalDiskSize,
    [FromQuery] string? sortBy,
    [FromQuery] bool? sortDesc,
    [FromQuery] int page = 1,
    [FromQuery] int pageSize = 20,
    CancellationToken cancellationToken = default)
    {
        if (page < 1) page = 1;
        if (pageSize < 1 || pageSize > 100) pageSize =
DefaultPageSize;
        sortBy ??= "name";
        sortDesc ??= false;
        var (items, totalCount) = await _catalog.GetProductsAsync(

```

```

        search, category, minPrice, maxPrice, cpuModel,
        ramType, ramAmount, gpu, totalDiskSize,
        sortBy, sortDesc.Value, page, pageSize,
        cancellationToken);
        var dtos = items.Select(p => new PcProductDto(
            p.Id, p.Name, PcProductOptions.ToLabel(p.Category),
            p.Description, p.Price,
            PcProductOptions.ToLabel(p.CpuModel),
            PcProductOptions.ToLabel(p.RamType),
            PcProductOptions.ToLabel(p.RamAmount),
            PcProductOptions.ToLabel(p.Gpu),
            PcProductOptions.ToLabel(p.TotalDiskSize),
            p.ImageData != null && p.ImageData.Length > 0,
            p.CreatedAt)).ToList();
        return Ok(new CatalogListResponse(dtos, totalCount, page,
            pageSize));
    }

    [HttpGet("{id:int}")]
    [AllowAnonymous]
    public async Task<ActionResult<PcProductDetailDto>>
    GetProduct(int id, CancellationToken cancellationToken = default)
    {
        var product = await _catalog.GetByIdAsync(id,
            includeDeleted: false, cancellationToken);
        if (product == null) return NotFound(new ApiError("Product
            not found.));
        var reviewDtos = product.Reviews
            .OrderByDescending(r => r.CreatedAt)
            .Select(r => new ReviewDto(r.Id, r.User?.UserName ??
                "User", r.Rating, r.Text, r.CreatedAt))
            .ToList();
        var dto = new PcProductDetailDto(
            product.Id, product.Name,
            PcProductOptions.ToLabel(product.Category), product.Description,
            product.Price,
            PcProductOptions.ToLabel(product.CpuModel),
            PcProductOptions.ToLabel(product.RamType),
            PcProductOptions.ToLabel(product.RamAmount),
            PcProductOptions.ToLabel(product.Gpu),
            PcProductOptions.ToLabel(product.TotalDiskSize),
            product.ImageData != null && product.ImageData.Length
            > 0, product.CreatedAt, reviewDtos);
        return Ok(dto);
    }

    [HttpGet("categories")]
    [AllowAnonymous]
    public async Task<ActionResult<IReadOnlyList<string>>>
    GetCategories(CancellationToken cancellationToken = default)
    {

```

```

        var categories = await
_catalog.GetCategoriesAsync(cancellationToken);
        return Ok(categories);
    }

    [HttpGet("options/cpu-models")]
    [AllowAnonymous]
    public ActionResult<IReadOnlyList<string>> GetCpuModels()
    {
        return Ok(PcProductOptions.CpuModels);
    }

    [HttpGet("options/ram-types")]
    [AllowAnonymous]
    public ActionResult<IReadOnlyList<string>> GetRamTypes()
    {
        return Ok(PcProductOptions.RamTypes);
    }

    [HttpGet("options/ram-amounts")]
    [AllowAnonymous]
    public ActionResult<IReadOnlyList<string>> GetRamAmounts()
    {
        return Ok(PcProductOptions.RamAmounts);
    }

    [HttpGet("options/gpus")]
    [AllowAnonymous]
    public ActionResult<IReadOnlyList<string>> GetGpus()
    {
        return Ok(PcProductOptions.Gpus);
    }

    [HttpGet("options/disk-sizes")]
    [AllowAnonymous]
    public ActionResult<IReadOnlyList<string>> GetDiskSizes()
    {
        return Ok(PcProductOptions.TotalDiskSizes);
    }

    [HttpGet("{id:int}/image")]
    [AllowAnonymous]
    [ResponseCache(NoStore = true)]
    public async Task<IActionResult> GetProductImage(int id,
Cancellation token cancellationToken = default)
    {
        var product = await _db.PcProducts
            .AsNoTracking()
            .Where(p => p.Id == id && !p.IsDeleted)
            .Select(p => new { p.ImageData })
            .FirstOrDefaultAsync(cancellationToken);
    }

```

```

        if (product?.ImageData == null || product.ImageData.Length
== 0)
            return NotFound();
        return File(product.ImageData, "image/png",
enableRangeProcessing: false);
    }
}

```

Листинг Б.4 – Файл OrdersApiController.cs

```

using System.Security.Claims;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using PersonalComputerStore.Api.Dto;
using PersonalComputerStore.Services;

namespace PersonalComputerStore.Api;

[ApiController]
[Route("api/orders")]
[Authorize(AuthenticationSchemes = ApiAuthConstants.Schemes)]
public class OrdersApiController(IOrderService orderService) :
ControllerBase
{
    private readonly IOrderService _orderService = orderService;

    private string UserId =>
User.FindFirstValue(ClaimTypes.NameIdentifier) ?? "";

    [HttpGet]
    [Authorize(AuthenticationSchemes = ApiAuthConstants.Schemes,
Roles = "RegisteredUser,Administrator")]
    public async Task<ActionResult<IReadOnlyList<OrderDto>>>
GetMyOrders(Cancellation_token cancellationToken = default)
    {
        var orders = await
_orderService.GetUserOrdersAsync(UserId, cancellationToken);
        var dtos = orders.Select(o => new OrderDto(
            o.Id, o.CreatedAt, o.TotalAmount, o.Status,
            [... o.Items.Select(i => new
OrderItemDto(i.PcProductId, i.PcProduct?.Name, i.Quantity,
i.UnitPrice))])).ToList();
        return Ok(dtos);
    }

    [HttpGet("{id:int}")]
    [Authorize(AuthenticationSchemes = ApiAuthConstants.Schemes,
Roles = "RegisteredUser,Administrator")]
    public async Task<ActionResult<OrderDto>> GetOrder(int id,
Cancellation_token cancellationToken = default)
    {
        var order = await _orderService.GetOrderByIdAsync(id,
UserId, cancellationToken);
    }
}

```

```

        if (order == null) return NotFound(new ApiError("Order not
found."));
        var dto = new OrderDto(
            order.Id, order.CreatedAt, order.TotalAmount,
order.Status,
           [.. order.Items.Select(i => new
OrderItemDto(i.PcProductId, i.PcProduct?.Name, i.Quantity,
i.UnitPrice))]);
        return Ok(dto);
    }

    [HttpPost]
    [Authorize(AuthenticationSchemes = ApiAuthConstants.Schemes,
Roles = "RegisteredUser,Administrator")]
    public async Task<ActionResult<OrderDto>>
CreateOrder([FromBody] CreateOrderRequest request,
CancellationTokens cancellationTokens = default)
    {
        if (request.Items == null || request.Items.Count == 0)
            return BadRequest(new ApiError("At least one item is
required."));
        var items = request.Items.Select(i => (i.ProductId,
i.Quantity)).ToList();
        var order = await _orderService.CreateOrderAsync(UserId,
items, cancellationTokens);
        if (order == null)
            return BadRequest(new ApiError("Could not create
order. Check product IDs and quantities."));
        await _orderService.ProcessPaymentAsync(order.Id,
cancellationTokens);
        var created = await
_orderService.GetOrderByIdAsync(order.Id, UserId,
cancellationTokens);
        if (created == null) return Ok(new OrderDto(order.Id,
order.CreatedAt, order.TotalAmount, order.Status, []));
        var dto = new OrderDto(
            created.Id, created.CreatedAt, created.TotalAmount,
created.Status,
           [.. created.Items.Select(i => new
OrderItemDto(i.PcProductId, i.PcProduct?.Name, i.Quantity,
i.UnitPrice))]);
        return CreatedAtAction(nameof(GetOrder), new { id = dto.Id
}, dto);
    }
}

```

Листинг Б.5 – Файл ReviewsApiController.cs

```

using System.Security.Claims;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using PersonalComputerStore.Api.Dto;
using PersonalComputerStore.Data;

```

```

using PersonalComputerStore.Services;

namespace PersonalComputerStore.Api;

[ApiController]
[Route("api/reviews")]
public class ReviewsApiController(AppDbContext db, ICatalogService
catalog) : ControllerBase
{
    private readonly AppDbContext _db = db;
    private readonly ICatalogService _catalog = catalog;

    [HttpPost]
    [Authorize(AuthenticationSchemes = ApiAuthConstants.Schemes,
Roles = "RegisteredUser")]
    public async Task<IActionResult> Create([FromBody]
CreateReviewRequest request, [FromQuery] int productId,
CancellationTokens cancellationTokens = default)
    {
        if (productId <= 0) return BadRequest(new ApiError("Valid
productId is required.));
        var product = await _catalog.GetByIdAsync(productId,
includeDeleted: false, cancellationTokens);
        if (product == null) return NotFound(new ApiError("Product
not found.));
        var userId =
User.FindFirstValue(ClaimTypes.NameIdentifier);
        if (string.IsNullOrEmpty(userId)) return Unauthorized();
        var review = new Review
        {
            PcProductId = productId,
            UserId = userId,
            Rating = request.Rating,
            Text = request.Text
        };
        _db.Reviews.Add(review);
        await _db.SaveChangesAsync(cancellationTokens);
        return StatusCode(201, new { id = review.Id, productId =
review.PcProductId });
    }

    [HttpDelete("{id:int}")]
    [Authorize(AuthenticationSchemes = ApiAuthConstants.Schemes,
Roles = "Moderator,Administrator")]
    public async Task<IActionResult> Delete(int id,
CancellationTokens cancellationTokens = default)
    {
        var review = await _db.Reviews.FindAsync([id],
cancellationTokens);
        if (review == null) return NotFound(new ApiError("Review
not found.));
        _db.Reviews.Remove(review);
    }
}

```

```

        await _db.SaveChangesAsync(cancellationToken);
        return NoContent();
    }
}

```

Листинг Б.6 – Файл SupportApiController.cs

```

using System.Security.Claims;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using PersonalComputerStore.Api.Dto;
using PersonalComputerStore.Data;

namespace PersonalComputerStore.Api;

[ApiController]
[Route("api/support")]
[Authorize(AuthenticationSchemes = ApiAuthConstants.Schemes)]
public class SupportApiController(AppDbContext db) :
    ControllerBase
{
    private readonly AppDbContext _db = db;

    private string? UserId =>
        User.FindFirstValue(ClaimTypes.NameIdentifier);

    private static SupportQuestionDto
        ToQuestionDto(SupportQuestion q)
        => new(
            q.Id,
            q.Subject,
            q.Body,
            q.CreatedAt,
            q.IsClosed,
            [.. q.Answers.OrderBy(a => a.CreatedAt).Select(a =>
                new SupportAnswerDto(a.Id, a.Body, a.Moderator?.UserName,
                    a.CreatedAt))],
            q.UserId == null,
            q.User?.UserName ?? q.GuestName);

    [HttpGet("my")]
    public async
        Task<ActionResult<IReadOnlyList<SupportQuestionDto>>>
        GetMyQuestions(CancellationTokentoken cancellationToken = default)
    {
        var userId = UserId;
        if (userId == null) return Unauthorized();
        var questions = await _db.SupportQuestions
            .AsNoTracking()
            .Where(q => q.UserId == userId)
            .OrderByDescending(q => q.CreatedAt)
            .Include(q => q.Answers)

```

```

        .ThenInclude(a => a.Moderator)
        .ToListAsync(cancellationToken);
var dtos = questions.Select(ToQuestionDto).ToList();
return Ok(dtos);
}

[HttpPost("ask")]
public async Task<IActionResult> Ask([FromBody]
AskQuestionRequest request, CancellationToken cancellationToken =
default)
{
    var userId = UserId;
    if (userId == null) return Unauthorized();
    var question = new SupportQuestion { UserId = userId,
Subject = request.Subject, Body = request.Body };
    _db.SupportQuestions.Add(question);
    await _db.SaveChangesAsync(cancellationToken);
    return StatusCode(201, new { id = question.Id });
}

[HttpPost("guest/ask")]
[AllowAnonymous]
public async Task<ActionResult<GuestQuestionAccessResponse>>
AskAsGuest([FromBody] GuestAskQuestionRequest request,
CancellationToken cancellationToken = default)
{
    var accessToken = Guid.NewGuid();
    var question = new SupportQuestion
    {
        Subject = request.Subject,
        Body = request.Body,
        GuestName = request.GuestName,
        GuestEmail = request.GuestEmail,
        GuestAccessToken = accessToken
    };
    _db.SupportQuestions.Add(question);
    await _db.SaveChangesAsync(cancellationToken);
    return StatusCode(201, new
GuestQuestionAccessResponse(question.Id,
accessToken.ToString("D")));
}

[HttpGet("guest/{id:int}")]
[AllowAnonymous]
public async Task<ActionResult<SupportQuestionDto>>
GetGuestQuestion(
    int id,
    [FromQuery] string accessToken,
    CancellationToken cancellationToken = default)
{
    if (!Guid.TryParse(accessToken, out var parsedToken))

```

```

        return BadRequest(new ApiError("Invalid access token
format."));

        var question = await _db.SupportQuestions
            .AsNoTracking()
            .Where(q => q.Id == id && q.UserId == null &&
q.GuestAccessToken == parsedToken)
            .Include(q => q.Answers)
            .ThenInclude(a => a.Moderator)
            .FirstOrDefaultAsync(cancellationTokens);
        if (question == null) return NotFound(new
ApiError("Question not found."));

        return Ok(ToQuestionDto(question));
    }

    [HttpGet("moderator/list")]
    [Authorize(AuthenticationSchemes = ApiAuthConstants.Schemes,
Roles = "Moderator,Administrator")]
    public async
Task<ActionResult<IReadOnlyList<SupportQuestionListItemDto>>>
ModeratorList(CancellationTokens cancellationTokens = default)
    {
        var questions = await _db.SupportQuestions
            .AsNoTracking()
            .Where(q => !q.IsClosed)
            .OrderByDescending(q => q.CreatedAt)
            .Include(q => q.Answers)
            .Include(q => q.User)
            .ToListAsync(cancellationTokens);
        var dtos = questions.Select(q => new
SupportQuestionListItemDto(
            q.Id,
            q.Subject,
            q.Body,
            q.User?.UserName ?? q.GuestName,
            q.CreatedAt,
            q.Answers.Count,
            q.UserId == null,
            q.GuestEmail)).ToList();
        return Ok(dtos);
    }

    [HttpPost("my/{id:int}/close")]
    public async Task<ActionResult> CloseQuestion(int id,
CancellationTokens cancellationTokens = default)
    {
        var userId = UserId;
        if (userId == null) return Unauthorized();
        var question = await _db.SupportQuestions.FindAsync([id],
cancellationTokens);

```

```

        if (question == null) return NotFound(new
ApiError("Question not found.));
        if (question.UserId != userId) return Forbid();
        question.IsClosed = true;
        await _db.SaveChangesAsync(cancellationToken);
        return NoContent();
    }

    [HttpPost("quest/{id:int}/close")]
    [AllowAnonymous]
    public async Task<IActionResult> CloseGuestQuestion(
        int id,
        [FromQuery] string accessToken,
        CancellationToken cancellationToken = default)
    {
        if (!Guid.TryParse(accessToken, out var parsedToken))
            return BadRequest(new ApiError("Invalid access token
format.));

        var question = await _db.SupportQuestions
            .Where(q => q.Id == id && q.UserId == null &&
q.GuestAccessToken == parsedToken)
            .FirstOrDefaultAsync(cancellationToken);
        if (question == null) return NotFound(new
ApiError("Question not found.));

        question.IsClosed = true;
        await _db.SaveChangesAsync(cancellationToken);
        return NoContent();
    }

    [HttpPost("moderator/answer")]
    [Authorize(AuthenticationSchemes = ApiAuthConstants.Schemes,
Roles = "Moderator,Administrator")]
    public async Task<IActionResult> Answer([FromBody]
AnswerQuestionRequest request, [FromQuery] int questionId,
CancellationToken cancellationToken = default)
    {
        var moderatorId = UserId;
        if (moderatorId == null) return Unauthorized();
        var question = await
_db.SupportQuestions.FindAsync([questionId], cancellationToken);
        if (question == null) return NotFound(new
ApiError("Question not found.));
        var answer = new SupportAnswer { QuestionId = questionId,
ModeratorId = moderatorId, Body = request.Body };
        _db.SupportAnswers.Add(answer);
        await _db.SaveChangesAsync(cancellationToken);
        return StatusCode(201, new { id = answer.Id });
    }

    [HttpGet("moderator/my-answers")]

```

```

        [Authorize(AuthenticationSchemes = ApiAuthConstants.Schemes,
Roles = "Moderator,Administrator")]
        public async
Task<ActionResult<IReadOnlyList<SupportAnswerDto>>>
MyAnswers(CancellationToken cancellationToken = default)
        {
            var moderatorId = UserId;
            if (moderatorId == null) return Unauthorized();
            var answers = await _db.SupportAnswers
                .AsNoTracking()
                .Where(a => a.ModeratorId == moderatorId)
                .OrderByDescending(a => a.CreatedAt)
                .Include(a => a.Question)
                .ToListAsync(cancellationToken);
            var dtos = answers.Select(a => new SupportAnswerDto(a.Id,
a.Body, null, a.CreatedAt)).ToList();
            return Ok(dtos);
        }
    }
}

```

Приложение В

(обязательное)

Исходный код мобильного приложения

Листинг В.1 – Файл AdminApi.kt

```
package com.computerstoremobileapp.data.api

import com.computerstoremobileapp.data.dto.CreateModeratorRequest
import com.computerstoremobileapp.data.dto.ModeratorDto
import com.computerstoremobileapp.data.dto.PcProductDto
import com.computerstoremobileapp.data.dto.ProductCreateUpdateDto
import com.computerstoremobileapp.data.dto.RevenueRequest
import com.computerstoremobileapp.data.dto.RevenueResponse
import okhttp3.MultipartBody
import retrofit2.Response
import retrofit2.http.Body
import retrofit2.http.DELETE
import retrofit2.http.Multipart
import retrofit2.http.POST
import retrofit2.http.PUT
import retrofit2.http.Part
import retrofit2.http.Path
import retrofit2.http.GET

interface AdminApi {
    @GET("api/admin/products")
    suspend fun getProducts(): List<PcProductDto>

    @GET("api/admin/products/{id}")
    suspend fun getProduct(@Path("id") id: Int): PcProductDto

    @POST("api/admin/products")
    suspend fun createProduct(@Body product:
ProductCreateUpdateDto): PcProductDto

    @PUT("api/admin/products/{id}")
    suspend fun updateProduct(@Path("id") id: Int, @Body product:
ProductCreateUpdateDto): Response<Void>

    @DELETE("api/admin/products/{id}")
    suspend fun deleteProduct(@Path("id") id: Int):
retrofit2.Response<Unit>

    @Multipart
    @POST("api/admin/products/{id}/image")
    suspend fun uploadProductImage(
        @Path("id") id: Int,
        @Part image: MultipartBody.Part
    ): retrofit2.Response<Unit>

    @GET("api/admin/moderators")
    suspend fun getModerators(): List<ModeratorDto>

    @POST("api/admin/moderators")
    suspend fun createModerator(@Body request:
CreateModeratorRequest): retrofit2.Response<Unit>
}
```

```

        @POST("api/admin/revenue")
        suspend fun getRevenue(@Body request: RevenueRequest):
RevenueResponse
    }

```

Листинг В.2 – Файл AuthApi.kt

```

package com.computerstoremobileapp.data.api

import com.computerstoremobileapp.data.dto.AuthResponse
import com.computerstoremobileapp.data.dto.LoginRequest
import com.computerstoremobileapp.data.dto.RegisterRequest
import retrofit2.http.Body
import retrofit2.http.POST

interface AuthApi {
    @POST("api/auth/register")
    suspend fun register(@Body request: RegisterRequest):
AuthResponse

    @POST("api/auth/login")
    suspend fun login(@Body request: LoginRequest): AuthResponse
}

```

Листинг В.3 – Файл CatalogApi.kt

```

package com.computerstoremobileapp.data.api

import com.computerstoremobileapp.data.dto.CatalogListResponse
import com.computerstoremobileapp.data.dto.PcProductDetailDto
import okhttp3.ResponseBody
import retrofit2.http.GET
import retrofit2.http.Path
import retrofit2.http.Query

interface CatalogApi {
    @GET("api/catalog")
    suspend fun getProducts(
        @Query("search") search: String? = null,
        @Query("category") category: String? = null,
        @Query("minPrice") minPrice: Double? = null,
        @Query("maxPrice") maxPrice: Double? = null,
        @Query("cpuModel") cpuModel: String? = null,
        @Query("ramType") ramType: String? = null,
        @Query("ramAmount") ramAmount: String? = null,
        @Query("gpu") gpu: String? = null,
        @Query("totalDiskSize") totalDiskSize: String? = null,
        @Query("sortBy") sortBy: String? = "name",
        @Query("sortDesc") sortDesc: Boolean? = false,
        @Query("page") page: Int = 1,
        @Query("pageSize") pageSize: Int = 20
    ): CatalogListResponse
}

```

```

    @GET("api/catalog/{id}")
    suspend fun getProduct(@Path("id") id: Int):
PcProductDetailDto

    @GET("api/catalog/categories")
    suspend fun getCategories(): List<String>

    @GET("api/catalog/options/cpu-models")
    suspend fun getCpuModels(): List<String>

    @GET("api/catalog/options/ram-types")
    suspend fun getRamTypes(): List<String>

    @GET("api/catalog/options/ram-amounts")
    suspend fun getRamAmounts(): List<String>

    @GET("api/catalog/options/gpus")
    suspend fun getGpuModels(): List<String>

    @GET("api/catalog/options/disk-sizes")
    suspend fun getDiskSizes(): List<String>

    @GET("api/catalog/{id}/image")
    suspend fun getProductImage(@Path("id") id: Int): ResponseBody
}

```

Листинг В.4 – Файл OrdersApi.kt

```

package com.computerstoremobileapp.data.api

import com.computerstoremobileapp.data.dto.CreateOrderRequest
import com.computerstoremobileapp.data.dto.OrderDto
import retrofit2.http.Body
import retrofit2.http.GET
import retrofit2.http.POST
import retrofit2.http.Path

interface OrdersApi {
    @GET("api/orders")
    suspend fun getOrders(): List<OrderDto>

    @GET("api/orders/{id}")
    suspend fun getOrder(@Path("id") id: Int): OrderDto

    @POST("api/orders")
    suspend fun createOrder(@Body request: CreateOrderRequest):
OrderDto
}

```

Листинг В.5 – Файл ReviewsApi.kt

```

package com.computerstoremobileapp.data.api

import com.computerstoremobileapp.data.dto.CreateReviewRequest
import com.computerstoremobileapp.data.dto.CreateReviewResponse
import retrofit2.http.Body
import retrofit2.http.DELETE
import retrofit2.http.POST
import retrofit2.http.Path
import retrofit2.http.Query

interface ReviewsApi {
    @POST("api/reviews")
    suspend fun createReview(
        @Query("productId") productId: Int,
        @Body request: CreateReviewRequest
    ): CreateReviewResponse

    @DELETE("api/reviews/{id}")
    suspend fun deleteReview(@Path("id") id: Int):
retrofit2.Response<Unit>
}

```

Листинг В.6 – Файл SupportApi.kt

```

package com.computerstoremobileapp.data.api

import com.computerstoremobileapp.data.dto.AnswerQuestionRequest
import com.computerstoremobileapp.data.dto.AskQuestionRequest
import com.computerstoremobileapp.data.dto.GuestAskQuestionRequest
import
com.computerstoremobileapp.data.dto.GuestQuestionAccessResponse
import com.computerstoremobileapp.data.dto.SupportAnswerDto
import com.computerstoremobileapp.data.dto.SupportQuestionDto
import
com.computerstoremobileapp.data.dto.SupportQuestionListItemDto
import retrofit2.http.Body
import retrofit2.http.GET
import retrofit2.http.POST
import retrofit2.http.Path
import retrofit2.http.Query

interface SupportApi {
    @POST("api/support/guest/ask")
    suspend fun askAsGuest(@Body request:
GuestAskQuestionRequest): GuestQuestionAccessResponse

    @GET("api/support/guest/{id}")
    suspend fun getGuestQuestion(
        @Path("id") id: Int,
        @Query("accessToken") accessToken: String
    ): SupportQuestionDto
}

```

```

@POST("api/support/guest/{id}/close")
suspend fun closeGuestQuestion(
    @Path("id") id: Int,
    @Query("accessToken") accessToken: String
): retrofit2.Response<Unit>

@GET("api/support/my")
suspend fun getMyQuestions(): List<SupportQuestionDto>

@POST("api/support/ask")
suspend fun ask(@Body request: AskQuestionRequest):
retrofit2.Response<Unit>

@POST("api/support/my/{id}/close")
suspend fun closeQuestion(@Path("id") id: Int):
retrofit2.Response<Unit>

@GET("api/support/moderator/list")
suspend fun getModeratorList():
List<SupportQuestionListItemDto>

@GET("api/support/moderator/{id}")
suspend fun getModeratorQuestion(@Path("id") id: Int):
SupportQuestionDto

@POST("api/support/moderator/answer")
suspend fun answer(
    @Query("questionId") questionId: Int,
    @Body request: AnswerQuestionRequest
): retrofit2.Response<Unit>

@GET("api/support/moderator/my-answers")
suspend fun getMyAnswers(): List<SupportAnswerDto>
}

```

Листинг В.7 – Файл SecureTokenStorage.kt

```

package com.computerstoremobileapp.data.auth

import android.content.Context
import androidx.security.crypto.EncryptedSharedPreferences
import androidx.security.crypto.MasterKey

class SecureTokenStorage(context: Context) {

    private val masterKey = MasterKey.Builder(context)
        .setKeyScheme(MasterKey.KeyScheme.AES256_GCM)
        .build()

    private val sharedPreferences =
        EncryptedSharedPreferences.create(
            context,

```

```

        "secure_prefs",
        masterKey,

EncryptedSharedPreferences.PrefKeyEncryptionScheme.AES256_SIV,

EncryptedSharedPreferences.PrefValueEncryptionScheme.AES256_GCM
    )

    private val tokenKey = "auth_token"
    private val userNameKey = "user_name"
    private val rolesKey = "user_roles"
    private val guestQuestionIdKey = "guest_question_id"
    private val guestAccessTokenKey = "guest_access_token"

    fun getToken(): String? =
        sharedPreferences.getString(tokenKey, null)

    fun setToken(token: String?) {
        sharedPreferences.edit().putString(tokenKey,
token).apply()
    }

    fun getUserName(): String? =
        sharedPreferences.getString(userNameKey, null)

    fun setUserName(userName: String?) {
        sharedPreferences.edit().putString(userNameKey,
userName).apply()
    }

    fun getRoles(): Set<String> =
        sharedPreferences.getStringSet(rolesKey, emptySet()) ?:
emptySet()

    fun setRoles(roles: List<String>) {
        sharedPreferences.edit().putStringSet(rolesKey,
roles.toSet()).apply()
    }

    fun getGuestQuestionId(): Int? {
        if (!sharedPreferences.contains(guestQuestionIdKey))
return null
        return sharedPreferences.getInt(guestQuestionIdKey, -
1).takeIf { it >= 0 }
    }

    fun setGuestQuestionId(id: Int?) {
        val e = sharedPreferences.edit()
        if (id == null) e.remove(guestQuestionIdKey) else
e.putInt(guestQuestionIdKey, id)
        e.apply()
    }

```

```

fun getGuestAccessToken(): String? =
    sharedPreferences.getString(guestAccessTokenKey, null)

fun setGuestAccessToken(token: String?) {
    sharedPreferences.edit().putString(guestAccessTokenKey,
token).apply()
}

fun clearGuestSupport() {
    sharedPreferences.edit()
        .remove(guestQuestionIdKey)
        .remove(guestAccessTokenKey)
        .apply()
}

fun clear() {
    sharedPreferences.edit()
        .remove(tokenKey)
        .remove(userNameKey)
        .remove(rolesKey)
        .apply()
}
}

```

Приложение Г
(рекомендуемое)
Результаты тестирования серверной части

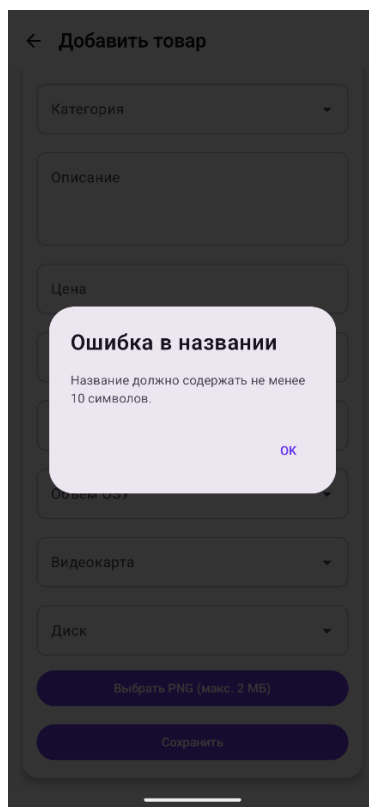


Рисунок Г.1 – Ввод слишком короткого названия

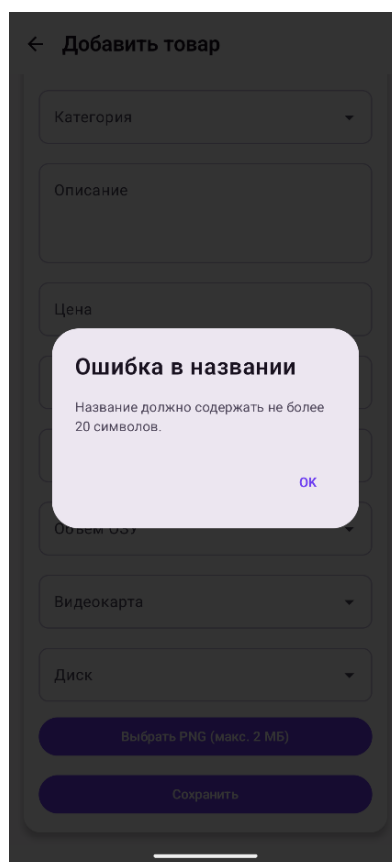


Рисунок Г.2 – Ввод слишком длинного названия

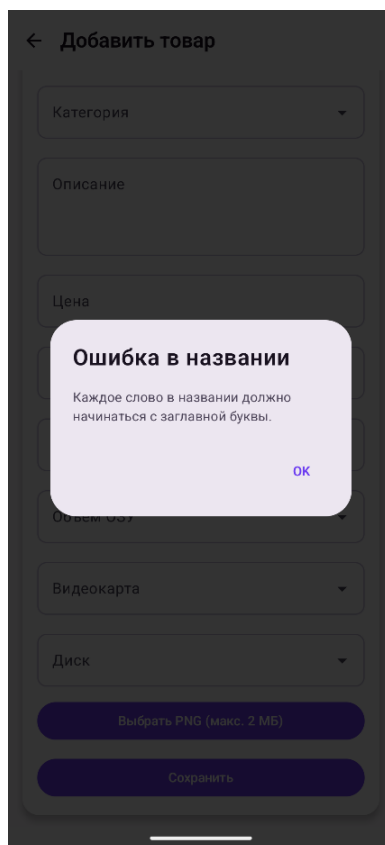


Рисунок Г.3 – Ввод названия со строчных букв

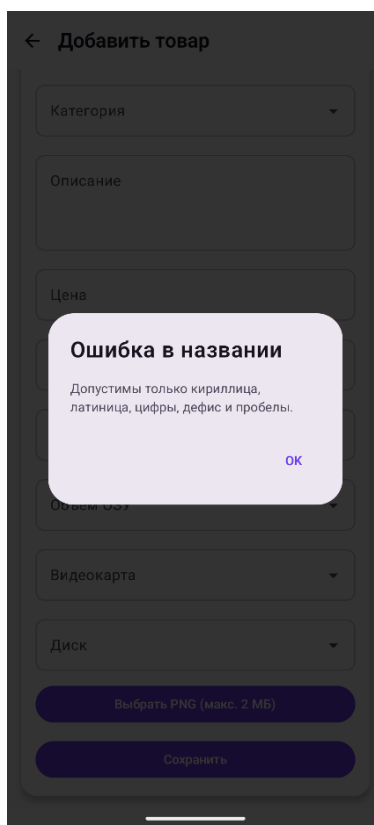


Рисунок Г.4 – Ввод названия с недопустимыми символами

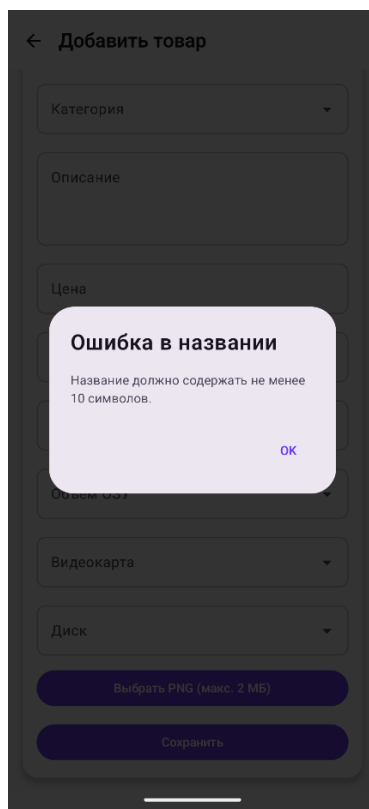


Рисунок Г.5 – Ввод названия нулевой длины

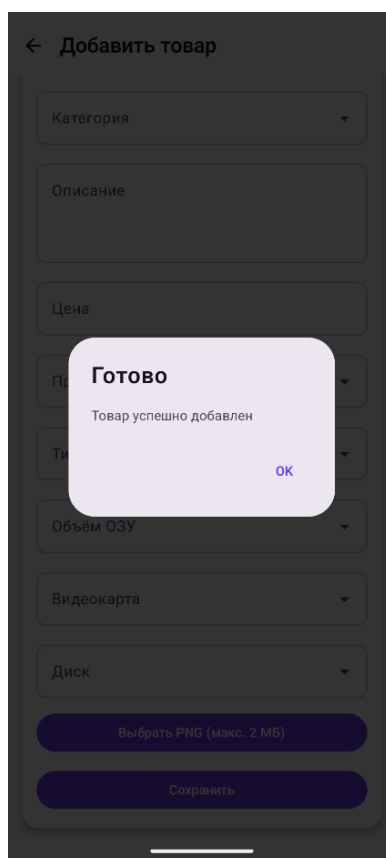


Рисунок Г.6 – Ввод названия, соответствующего всем ограничениям

Приложение Д
(рекомендуемое)
Исходный код класса тестирования

Листинг Д.1 – Файл CatalogApiControllerTests.cs

```

using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Moq;
using PersonalComputerStore.Api;
using PersonalComputerStore.Api.Dto;
using PersonalComputerStore.Data;
using PersonalComputerStore.Services;

namespace PersonalComputerStore.Tests;

public class CatalogApiControllerTests
{
    [Fact]
    public async Task
    GetProducts_NormalizesPagingAndReturnsMappedPayload()
    {
        var catalog = new Mock<ICatalogService>();
        var product = new PcProduct { Id = 7, Name = "Build",
Price = 1500m, ImageData = [1, 2, 3] };
        catalog
            .Setup(x => x.GetProductsAsync(
                null, null, null, null, null, null, null, null,
                "name", false, 1, 20,
                It.IsAny<CancellationToken>()))
            .ReturnsAsync((new List<PcProduct> { product }, 1));

        await using var db = CreateDbContext();
        var sut = new CatalogApiController(catalog.Object, db);

        var result = await sut.GetProducts(
            search: null,
            category: null,
            minPrice: null,
            maxPrice: null,
            cpuModel: null,
            ramType: null,
            ramAmount: null,
            gpu: null,
            totalDiskSize: null,
            sortBy: null,
            sortDesc: null,
            page: 0,
            pageSize: 101);

        var ok = Assert.IsType<OkObjectResult>(result.Result);
        var payload =
        Assert.IsType<CatalogListResponse>(ok.Value);
        Assert.Equal(1, payload.TotalCount);
        Assert.Equal(1, payload.Page);
        Assert.Equal(20, payload.PageSize);
    }
}

```

```

        Assert.Single(payload.Items);
        Assert.True(payload.Items[0].HasImage);
    }

    [Fact]
    public async Task
    GetProduct_ReturnsNotFoundWhenProductMissing()
    {
        var catalog = new Mock<ICatalogService>();
        catalog.Setup(x => x.GetByIdAsync(99, false,
            It.IsAny<CancellationToken>()))
            .ReturnsAsync((PcProduct?)null);

        await using var db = CreateDbContext();
        var sut = new CatalogApiController(catalog.Object, db);

        var result = await sut.GetProduct(99);

        var notFound =
        Assert.IsType<NotFoundObjectResult>(result.Result);
        var error = Assert.IsType<ApiError>(notFound.Value);
        Assert.Equal("Product not found.", error.Message);
    }

    [Fact]
    public async Task
    GetProductImage_ReturnsPngFileWhenImageExists()
    {
        var catalog = new Mock<ICatalogService>();
        await using var db = CreateDbContext();
        db.PcProducts.Add(new PcProduct { Name = "Image PC", Price
= 1000m, ImageData = [137, 80, 78, 71] });
        await db.SaveChangesAsync();
        var id = await db.PcProducts.Select(p =>
p.Id).FirstAsync();

        var sut = new CatalogApiController(catalog.Object, db);
        var result = await sut.GetProductImage(id);

        var file = Assert.IsType<FileContentResult>(result);
        Assert.Equal("image/png", file.ContentType);
        Assert.Equal(4, file.FileContents.Length);
    }

    private static AppDbContext CreateDbContext()
    {
        var options = new DbContextOptionsBuilder<AppDbContext>()
            .UseInMemoryDatabase(Guid.NewGuid().ToString())
            .Options;
        return new AppDbContext(options);
    }
}

```

Листинг Д.2 – Файл JwtTokenServiceTests.cs

```

using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using Microsoft.AspNetCore.Identity;
using Microsoft.Extensions.Configuration;
using Moq;
using PersonalComputerStore.Data;
using PersonalComputerStore.Services;

namespace PersonalComputerStore.Tests;

public class JwtTokenServiceTests
{
    [Fact]
    public async Task
GenerateTokenAsync_CreatesJwtWithExpectedClaimsAndRoles()
    {
        var user = new ApplicationUser
        {
            Id = "user-123",
            Email = "user@test.local",
            UserName = "test-user"
        };

        var configuration = new ConfigurationBuilder()
            .AddInMemoryCollection(new Dictionary<string, string?>
            {
                ["Jwt:Key"] = "super-secret-key-for-tests-super-secret-key",
                ["Jwt:Issuer"] = "pc-store-tests",
                ["Jwt:Audience"] = "pc-store-clients",
                ["Jwt:ExpirationMinutes"] = "60"
            })
            .Build();

        var userManager = CreateUserManagerMock();
        userManager
            .Setup(x => x.GetRolesAsync(user))
            .ReturnsAsync(["Admin", "Customer"]);

        var sut = new JwtTokenService(configuration,
userManager.Object);
        var token = await sut.GenerateTokenAsync(user);

        var jwt = new
JwtSecurityTokenHandler().ReadJwtToken(token);
        Assert.Equal("pc-store-tests", jwt.Issuer);
        Assert.Contains("pc-store-clients", jwt.Audiences);
        Assert.Equal("user-123", jwt.Claims.First(c => c.Type ==
JwtRegisteredClaimNames.Sub).Value);
        Assert.Equal("user@test.local", jwt.Claims.First(c =>
c.Type == JwtRegisteredClaimNames.Email).Value);
    }
}

```

```

        Assert.Equal("test-user", jwt.Claims.First(c => c.Type ==
JwtRegisteredClaimNames.UniqueName).Value);
        Assert.Equal("user-123", jwt.Claims.First(c => c.Type ==
ClaimTypes.NameIdentifier).Value);

        var roles = jwt.Claims.Where(c => c.Type ==
ClaimTypes.Role).Select(c => c.Value).ToList();
        Assert.Contains("Admin", roles);
        Assert.Contains("Customer", roles);
    }

    [Fact]
    public async Task GenerateTokenAsync_ThrowsWhenJwtKeyMissing()
    {
        var configuration = new ConfigurationBuilder()
            .AddInMemoryCollection([])
            .Build();
        var userManager = CreateUserManagerMock();
        var user = new ApplicationUser { Id = "u1", UserName =
"u1" };
        var sut = new JwtTokenService(configuration,
userManager.Object);

        await Assert.ThrowsAsync<InvalidOperationException>(() =>
sut.GenerateTokenAsync(user));
    }

    private static Mock<UserManager<ApplicationUser>>
CreateUserManagerMock()
    {
        var store = new Mock<IUserStore<ApplicationUser>>();
        return new Mock<UserManager<ApplicationUser>>(
            store.Object,
            null!,
            null!,
            null!,
            null!,
            null!,
            null!,
            null!,
            null!);
    }
}

```

Листинг Д.3 – Файл OrdersApiControllerTests.cs

```

using System.Security.Claims;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Moq;
using PersonalComputerStore.Api;

```

```

using PersonalComputerStore.Api.Dto;
using PersonalComputerStore.Data;
using PersonalComputerStore.Services;

namespace PersonalComputerStore.Tests;

public class OrdersApiControllerTests
{
    [Fact]
    public async Task
CreateOrder_ReturnsBadRequestWhenItemsAreMissing()
    {
        var orderService = new Mock<IOrderService>();
        var sut = BuildController(orderService.Object, "user-1");

        var result = await sut.CreateOrder(new
CreateOrderRequest([]));

        var badRequest =
Assert.IsType<BadRequestObjectResult>(result.Result);
        var error = Assert.IsType<ApiError>(badRequest.Value);
        Assert.Equal("At least one item is required.",
error.Message);
    }

    [Fact]
    public async Task
CreateOrder_ReturnsCreatedAndProcessesPayment()
    {
        var orderService = new Mock<IOrderService>();
        var createdOrder = new Order
        {
            Id = 42,
            UserId = "user-1",
            TotalAmount = 999m,
            Status = OrderStatus.Pending,
            CreatedAt = DateTime.UtcNow
        };

        orderService
            .Setup(x => x.CreateOrderAsync("user-1",
It.IsAny<IReadOnlyList<(int ProductId, int Quantity)>>(),
It.IsAny<CancellationTokens>()))
            .ReturnsAsync(createdOrder);

        orderService
            .Setup(x => x.GetOrderByIdAsync(42, "user-1",
It.IsAny<CancellationTokens>()))
            .ReturnsAsync(new Order
            {
                Id = 42,
                UserId = "user-1",

```

```

        TotalAmount = 999m,
        Status = OrderStatus.Paid,
        Items =
        [
            new() { PcProductId = 3, Quantity = 1,
UnitPrice = 999m, PcProduct = new PcProduct { Name = "PC #3",
Price = 999m } }
        ]
    });

    var sut = BuildController(orderService.Object, "user-1");

    var result = await sut.CreateOrder(new
CreateOrderRequest([new(3, 1)]));

    var createdAt =
Assert.IsType<CreatedAtActionResult>(result.Result);
    var dto = Assert.IsType<OrderDto>(createdAt.Value);
    Assert.Equal(42, dto.Id);
    Assert.Single(dto.Items);

    orderService.Verify(x => x.ProcessPaymentAsync(42,
It.IsAny<CancellationToken>()), Times.Once);
}

[Fact]
public async Task GetOrder_ReturnsNotFoundWhenMissing()
{
    var orderService = new Mock<IOrderService>();
    orderService.Setup(x => x.GetOrderByIdAsync(10, "user-1",
It.IsAny<CancellationToken>()))
        .ReturnsAsync((Order?)null);

    var sut = BuildController(orderService.Object, "user-1");
    var result = await sut.GetOrder(10);

    var notFound =
Assert.IsType<NotFoundObjectResult>(result.Result);
    var error = Assert.IsType<ApiError>(notFound.Value);
    Assert.Equal("Order not found.", error.Message);
}

private static OrdersApiController
BuildController(IOrderService orderService, string userId)
{
    var sut = new OrdersApiController(orderService)
    {
        ControllerContext = new ControllerContext
        {
            HttpContext = new DefaultHttpContext
            {
                User = new ClaimsPrincipal(

```

```
new ClaimsIdentity(  
    [new Claim(ClaimTypes.NameIdentifier,  
        userId)],  
    "test-auth"))  
    }  
    }  
};  
return sut;  
}  
}
```