

НЕГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ ЧАСТНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«МОСКОВСКИЙ УНИВЕРСИТЕТ «СИНЕРГИЯ»

Факультет Информационных технологий

Специальность	09.02.07	Кафедра
_____	_____	_____
	(код)	(аббревиатура) <sup>12</sup>

## ДИПЛОМНЫЙ ПРОЕКТ

на тему Отладка и разработка тестовых наборов и сценариев при  
автоматизации продаж «Wildberries»

Обучающийся

Унежев Клим Романович

(Фамилия, Имя, Отчество)

(подпись)

Руководитель

Сибирев

(Фамилия, Имя, Отчество)

(подпись)

МОСКВА 2025 г.

**ЗАДАНИЕ**  
на дипломный проект  
обучающемуся

<sup>1</sup> ИМИИКТ им. В.В. Дика – кафедра Информационного менеджмента и информационно-коммуникационных технологий им. В.В. Дика

<sup>2</sup> ЦЭ – кафедра Цифровой экономики

## **1. Тема дипломного проекта:**

Отладка и разработка тестовых наборов и сценариев при автоматизации продаж «Wildberries»

---

## **2. Структура дипломного проекта:**

ВВЕДЕНИЕ.....	4
Глава 1. Аналитическая часть .....	6
1.1 Технико-экономическая характеристика предметной области и предприятия (Анализ деятельности «КАК ЕСТЬ»)	6
1.1.1 Характеристика предприятия и его деятельности.....	6
1.1.2 Организационная структура управления предприятием .....	8
1.1.3 Программная и аппаратная архитектура ИС предприятия.....	10
1.2 Характеристика комплекса задач, задачи и обоснование необходимости автоматизации .....	12
1.2.1. Выбор комплекса задач автоматизации и характеристика существующих бизнес-процессов .....	12
1.2.2. Определение места проектируемой задачи в комплексе задач и ее описание .....	14
1.2.3. Анализ информационных потоков проектируемой задачи .....	15
1.2.4. Анализ системы обеспечения информационной безопасности и защиты информации .....	17
1.3 Анализ существующих разработок и выбор стратегии автоматизации («КАК ДОЛЖНО БЫТЬ») .....	19
1.3.1. Анализ существующих разработок для автоматизации задачи .....	19
1.3.2. Выбор и обоснование стратегии автоматизации задачи.....	21
1.3.3. Выбор и обоснование способа приобретения ИС для автоматизации комплекса задач.....	24
Глава 2. Проектная часть .....	26
2.2 Разработка проекта автоматизации.....	26
2.1.1 Этапы жизненного цикла проекта автоматизации .....	26
2.1.2. Ожидаемые риски на этапах жизненного цикла и их описание.....	27
2.2. Информационное обеспечение задачи .....	29
2.2.1. Характеристика нормативно-справочной, входной и оперативной информации.....	29
2.2.2. Характеристика результатной информации.....	30
2.3. Программное обеспечение задачи .....	32

2.3.1. Сценарий диалога .....	32
2.3.2. Характеристика базы данных .....	41
2.3.4. Описание программных модулей.....	47
3. Модуль корзины: .....	49
4. Модуль кабинета продавца.....	50
5. Модуль кабинета покупателя .....	51
2.4. Испытания разработанного решения.....	52
2.4.1. Перечень объектов и функций, подлежащих испытаниям.....	52
2.4.2. Методы проведения испытаний .....	52
2.4.3. Проведение проверочных испытаний и их результаты .....	54
Глава 3. Обоснование экономической эффективности проекта .....	62
3.1 Выбор и обоснование методики расчета экономической эффективности.....	62
3.2. Расчет показателей экономической эффективности проекта.....	63
ЗАКЛЮЧЕНИЕ .....	66
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ .....	68
ПРИЛОЖЕНИЕ .....	70

## **ВВЕДЕНИЕ**

Мы живем в эпоху цифровой трансформации, когда автоматизация бизнес-процессов играет ключевую роль в повышении эффективности продаж и управлении торговыми операциями. Еще несколько лет назад маркетплейсы, такие как Wildberries, были скорее инновационным явлением, но сегодня они стали неотъемлемой частью современной коммерции. Однако закрытость их API и отсутствие доступа к исходному коду существенно ограничивают возможности тестирования и отладки автоматизированных решений.

В рамках данного дипломного проекта мы ставим перед собой задачу разработать собственную информационную систему для покупателей и продавцов, имитирующую ключевые функции маркетплейса. Это позволит не только изучить принципы автоматизации торговых процессов, но и создать гибкую платформу для тестирования различных сценариев.

На сегодняшний день автоматизация продаж требует тщательной отладки и проверки корректности работы всех компонентов системы. Однако тестирование на реальных маркетплейсах, таких как Wildberries, осложнено из-за:

- 1) Закрытого API, что ограничивает возможности интеграции и отладки.
- 2) Отсутствия доступа к внутренней логике, что затрудняет анализ ошибок.
- 3) Риска блокировки аккаунта при активном автоматизированном тестировании.

Разработка собственной системы на C# (WinForms) с базой данных MySQL позволит:

- 1) Полностью контролировать логику работы платформы.
- 2) Создавать и модифицировать тестовые сценарии без ограничений.
- 3) Отрабатывать различные кейсы (регистрация, размещение товаров, оформление заказов, отчетность).

### Цель и задачи работы

Цель – разработка и тестирование информационной системы для автоматизации продаж, включающей функционал для покупателей и продавцов, с последующей отладкой и проверкой корректности работы.

#### Основные задачи:

- 1) Анализ требований к системе, изучение аналогов (Wildberries, Ozon).
- 2) Проектирование архитектуры (клиент-серверное взаимодействие, структура БД).
- 3) Разработка системы на C# (WinForms) с использованием MySQL.
- 4) Создание тестовых наборов (модульные, интеграционные, UI-тесты).
- 5) Отладка и оптимизация работы системы.

#### Структура работы

Дипломная работа состоит из трех основных разделов:

- 1) Аналитическая часть – исследование существующих решений, обоснование выбора технологий.
- 2) Проектная часть – разработка системы, проектирование интерфейсов и БД.
- 3) Тестовая часть – реализация тестовых сценариев, анализ результатов.

#### Практическая значимость

Разработанная система может быть использована:

- 1) Как учебный проект для изучения автоматизации торговых процессов.
- 2) В качестве тестового стенда для отработки различных сценариев (нагрузочное тестирование, проверка бизнес-логики).
- 3) Для дальнейшего масштабирования (например, добавления мобильного клиента или веб-версии).

Данная выпускная квалификационная работа соответствует современным тенденциям в области автоматизации продаж и может служить основой для более сложных коммерческих решений.

## **Глава 1. Аналитическая часть**

### **1.1 Техничко-экономическая характеристика предметной области и предприятия (Анализ деятельности «КАК ЕСТЬ»)**

#### **1.1.1 Характеристика предприятия и его деятельности**

Wildberries является крупнейшим в России и странах СНГ маркетплейсом, занимающим лидирующие позиции в сфере электронной коммерции. Основанная в 2004 году компания за два десятилетия превратилась в технологического гиганта, ежедневно обрабатывающего миллионы заказов. Основная деятельность Wildberries сосредоточена на предоставлении цифровой платформы для продажи широкого ассортимента товаров, включая одежду, обувь, электронику, товары для дома и другие категории.

Головной офис компании расположен в Москве, однако инфраструктура Wildberries охватывает всю страну. На сегодняшний день компания располагает сетью из более чем 10 современных фулфилмент-центров и 20 000 пунктов выдачи заказов, что позволяет обеспечивать быструю доставку в любой регион присутствия. Географическая экспансия является одним из ключевых приоритетов бизнеса - за последние годы компания вышла на рынки 18 стран, включая государства СНГ и Восточной Европы.

Штат компании насчитывает свыше 50 000 сотрудников, что делает Wildberries одним из крупнейших работодателей в российской IT-индустрии. Организационная структура включает департаменты разработки, логистики, маркетинга, клиентского обслуживания и другие ключевые подразделения, обеспечивающие бесперебойную работу платформы. Особое внимание уделяется технологическому развитию - компания постоянно инвестирует в совершенствование своей IT-инфраструктуры и мобильных приложений, которые были скачаны более 85 миллионов раз.

Финансовые показатели Wildberries демонстрируют устойчивый рост. В 2024 году годовой оборот компании составил 2,35 триллиона рублей, что

на 24% превышает показатели предыдущего года. Количество продавцов на платформе увеличилось до 720 000, а ассортимент товаров превысил 110 миллионов SKU. Ежедневно система обрабатывает более 7 миллионов заказов, а среднее время доставки сократилось до 2,8 дней благодаря оптимизации логистических процессов.

**Таблица 1**

**Технико-экономические показатели компании**

№ п/п	Наименование показателя	2023 год	2024 год
1	Годовой оборот (млрд руб.)	1 890	2 350
2	Количество продавцов на платформе	550 000	720 000
3	Количество заказов в сутки	5,2 млн	7,1 млн
4	География доставки (стран)	15	18
5	Доля на рынке e-commerce России	32%	38%
6	Количество SKU (товарных позиций)	80 млн	110 млн
7	Среднее время доставки (дни)	3,5	2,8
8	Мобильные приложения (скачиваний)	60 млн	85 млн

### **1.1.2 Организационная структура управления предприятием**

Организационная структура управления Wildberries представляет собой стройную вертикально-интегрированную систему, обеспечивающую эффективное управление всеми бизнес-процессами компании. Как крупнейший маркетплейс России, Wildberries выстроил четкую иерархическую модель управления, сочетающую централизованное стратегическое руководство с децентрализованным операционным управлением.

Высший уровень управления представлен Советом директоров и Генеральным директором (Татьяной Бакальчук), которые определяют стратегические направления развития компании. На этом уровне принимаются ключевые решения относительно инвестиционной политики, международной экспансии, технологического развития и других стратегических вопросов.

Исполнительное руководство включает в себя Правление компании, состоящее из руководителей ключевых направлений:

- Коммерческий директор (отвечает за развитие продаж и партнерских программ)
- Технический директор (курирует IT-разработку и цифровую инфраструктуру)
- Операционный директор (управляет логистикой и фулфилмент-центрами)
- Финансовый директор (контролирует бюджет и финансовые потоки)
- Директор по маркетингу (отвечает за продвижение и клиентский опыт)

Функциональные подразделения компании организованы по дивизиональному принципу и включают:

1. Департамент электронной коммерции (управление платформой, работа с продавцами, развитие ассортимента)
2. Технологический блок (разработка и поддержка IT-систем, аналитика данных, кибербезопасность)
3. Логистический комплекс (управление фулфилмент-центрами, транспортная логистика, служба доставки)
4. Финансово-экономическая служба (бухгалтерия, планирование, управление рисками)
5. Маркетинг и PR (бренд-менеджмент, digital-маркетинг, медиа-коммуникации)



6. Кадровая служба (рекрутинг, обучение, корпоративная культура)
7. Юридический департамент (правовое сопровождение, compliance)

Региональная структура управления построена по географическому принципу. В каждом регионе присутствия работают региональные директора, координирующие деятельность местных подразделений: пунктов выдачи заказов, сервисных центров и представительств.

Особенностью организационной структуры Wildberries является:

- Гибкое сочетание централизованного управления и автономии подразделений
- Четкое разделение продуктовых и функциональных вертикалей
- Сквозные кросс-функциональные команды для реализации стратегических проектов
- Высокая степень автоматизации управленческих процессов
- Матричная система управления для ключевых направлений деятельности

Принципы управления в компании включают:

1. Клиентоориентированность на всех уровнях
2. Данно-ориентированный подход к принятию решений
3. Гибкость и адаптивность организационных структур
4. Непрерывное совершенствование процессов
5. Развитие внутреннего предпринимательства

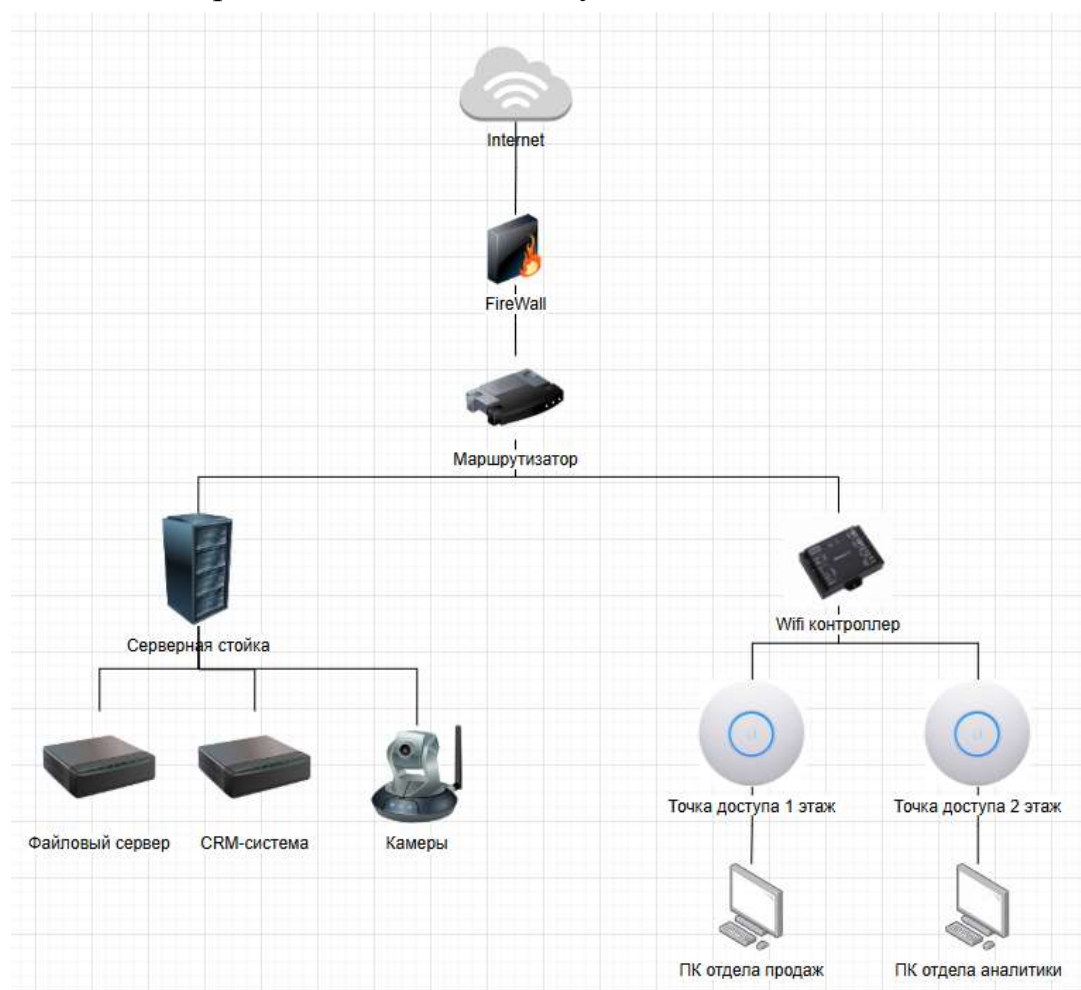


**Рисунок 1. Организационная структура**

В ходе данного дипломного проекта мы будем разбирать автоматизацию и тестирование ИС для отдела коммерческой дирекции.

### 1.1.3 Программная и аппаратная архитектура ИС предприятия

Опишем и рассмотрим аппаратную и программную архитектуру офиса отдела коммерческой дирекции WB, для которой будет разработано desktop приложение, которого в данный момент у компании нет.



**Рисунок 2. Аппаратная архитектура**

Детализация аппаратной инфраструктуры отдела коммерческой дирекции.

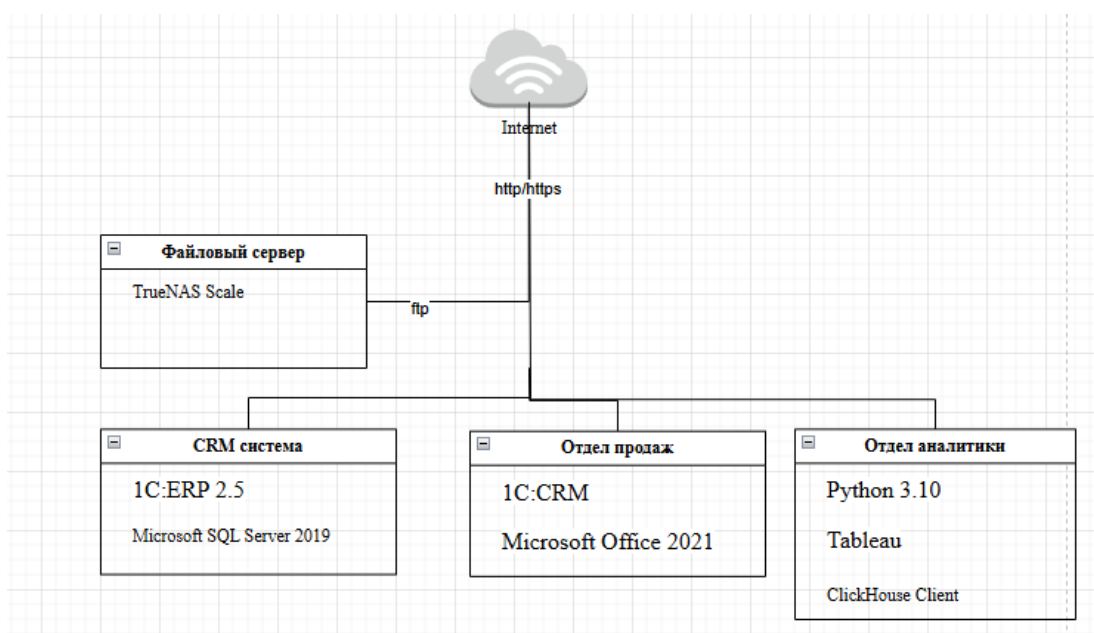
**Таблица 2**

**Технические характеристики аппаратных устройств**

Устройство	Модель/Пример	Характеристики	Назначение
Серверы	Dell PowerEdge R750	- Процессор: 2× Intel Xeon Silver 4310 (12 ядер) - ОЗУ: 128–512 ГБ DDR4 - Хранение: 4× 1.92 ТБ SSD (RAID 10) - Сетевые порты: 2× 10 Гбит/s	Хостинг CRM, баз данных товаров и аналитики. Обеспечивают работу API для продавцов.
Маршрутизаторы	Cisco Catalyst 9300	- Пропускная способность: 1 Тбит/с	Сегментация сети между отделами (логистика, финансы, CRM).

		<ul style="list-style-type: none"> <li>- Поддержка VLAN, QoS, IPv6</li> <li>- 48 портов 1/10 Гбит</li> </ul>	Резервирование каналов.
Wi-Fi точки	Ubiquiti U6-Pro	<ul style="list-style-type: none"> <li>- Стандарт: Wi-Fi 6 (AX)</li> <li>- Скорость: до 4.8 Гбит/с</li> <li>- Одновременных клиентов: 300+</li> </ul>	Беспроводной доступ для сотрудников в зонах переговоров и складов.
Рабочие станции	HP EliteDesk 805 G8	<ul style="list-style-type: none"> <li>- Процессор: AMD Ryzen 7 PRO 5750G</li> <li>- ОЗУ: 32 ГБ</li> <li>- SSD: 512 ГБ</li> <li>- Монитор: 24" FHD</li> </ul>	Работа с CRM, аналитика продаж, управление ассортиментом.
Камеры	Hikvision DS-2CD2386G2-I	<ul style="list-style-type: none"> <li>- Разрешение: 8 Мп (4K)</li> <li>- Ночное видение: до 30 м</li> <li>- Аналитика: детекция лиц/объектов</li> </ul>	Контроль складов, офисных помещений и зон отгрузки. Интеграция с системой безопасности.
Файловый сервер	Synology RS3621RPxs	<ul style="list-style-type: none"> <li>- Хранение: 12× 16 ТБ HDD (до 192 ТБ)</li> <li>- RAID: SHR-2</li> <li>- Скорость чтения: 2.5 Гбит/с</li> </ul>	Централизованное хранение документов (договоры, прайсы, отчеты). Резервное копирование.

Программная архитектура отдела коммерческой дирекции выглядит следующим образом



**Рисунок 3. Программная архитектура**

## 1.2 Характеристика комплекса задач, задачи и обоснование необходимости автоматизации

### 1.2.1. Выбор комплекса задач автоматизации и характеристика существующих бизнес-процессов

В процессе изучения и ознакомления с предметной областью была спроектирована диаграмма бизнес процессов отдела коммерческой дирекции Wildberries.



Рисунок 4. IDEF 0

Перейдем к описанию элементов.

Корпоративная стратегия Wildberries формирует фундамент для всех операционных решений коммерческой дирекции. Этот стратегический вектор определяет целевые показатели роста, приоритетные категории товаров и ключевые рынки сбыта, создавая систему координат для ежедневной работы. На основе этих установок разрабатываются конкретные KPI для отдела: объем продаж, маржинальность, конверсия в покупку и другие метрики, которые становятся измеримыми ориентирами для сотрудников.

Политика работы с поставщиками регламентирует все этапы взаимодействия – от первичного отбора партнеров до условий размещения товаров на маркетплейсе. Особое внимание уделяется системе мотивации: бонусам за объемы, рейтингам поставщиков и программам лояльности. Бюджетные ограничения выступают в роли финансового "корсета",

распределяя ресурсы между закупками, маркетингом и логистикой согласно утвержденным лимитам.

На вход процесса поступают критически важные данные: рыночные тренды из систем мониторинга, актуальные заказы клиентов в режиме реального времени, финансовые лимиты от руководства и постоянный поток заявок от поставщиков. Этот информационный массив проходит многоступенчатую обработку – данные очищаются, категоризируются и загружаются в аналитические системы для дальнейшего преобразования в управленческие решения.

Ядром процесса выступает операционный контур управления, где ежедневно принимаются сотни решений. Менеджеры по продажам корректируют ассортиментные матрицы, специалисты по работе с поставщиками согласовывают условия сотрудничества, а аналитики выявляют паттерны покупательского поведения. Все это происходит в жестких рамках установленных регламентов, но с необходимым уровнем гибкости для оперативного реагирования на изменения рынка.

Результатом цикла становятся конкретные выходы: утвержденный план продаж с помесечной разбивкой, графики динамики выручки по товарным категориям и регулярные отчеты разных уровней детализации. Эти документы содержат не только фактические показатели, но и прогнозные значения, рекомендации по оптимизации ассортимента и ценовые предложения.

Технологическую основу процесса составляют три ключевых системы: CRM-платформа для управления взаимоотношениями с поставщиками, ERP-решение для интеграции всех бизнес-процессов и BI-инструменты для визуализации данных. Отдельное место занимает система аналитики на базе Python и ClickHouse, которая превращает сырые данные в стратегические инсайты. Этот технологический стек работает как единый организм, обеспечивая бесперебойное функционирование коммерческого блока маркетплейса.

### **1.2.2. Определение места проектируемой задачи в комплексе задач и ее описание**

В условиях стремительного роста объемов электронной торговли перед отделом коммерческой дирекции Wildberries встает задача повышения прозрачности, управляемости и надежности процессов продаж. Центральным элементом коммерческой деятельности маркетплейса является система управления, включающая в себя анализ рыночных данных, обработку заказов клиентов, управление входящими заявками от поставщиков и контроль финансовых лимитов. Данная деятельность осуществляется в строгом соответствии с корпоративной стратегией, политикой работы с поставщиками и установленными бюджетными ограничениями. В рамках этой системы формируются ключевые выходные документы — утвержденный план продаж, отчеты по динамике выручки и сводные аналитические данные.

Однако, платформа Wildberries имеет закрытый исходный код, что создает значительные ограничения при отладке программных компонентов и разработке автоматизированных тестовых сценариев. В связи с этим было принято решение о создании собственной информационной системы управления продажами, ориентированной на покупателей и продавцов. Новая система реализуется в виде настольного приложения на платформе WinForms с использованием языка программирования C# и реляционной базы данных MySQL.

Основная цель проекта — отладка и разработка тестовых наборов и сценариев при автоматизации процессов управления продажами. Реализуемая система позволит моделировать ключевые бизнес-процессы коммерческой дирекции маркетплейса, включая взаимодействие с клиентами и поставщиками, управление заказами, учет бюджетных ограничений и анализ эффективности. При этом информационная система будет интегрироваться с CRM- и ERP-платформами, BI-инструментами и модулями

системной аналитики, что обеспечит полноту и достоверность анализа коммерческой деятельности.

Проектируемая задача занимает центральное место в комплексе задач цифровой трансформации отдела коммерческой дирекции, обеспечивая автоматизацию сбора, обработки и анализа данных, а также формирование отчетной документации. Разработка тестовых сценариев позволит не только обеспечить устойчивость и надежность системы, но и выявить узкие места в логике бизнес-процессов до внедрения в продуктивную среду.

Таким образом, реализация проекта не только обеспечит безопасную и эффективную платформу для тестирования и развития цифровых решений, но и создаст прототип информационной среды, способной масштабироваться и адаптироваться под задачи реального маркетплейса.

### **1.2.3. Анализ информационных потоков проектируемой задачи**

Процессы, происходящие в отделе коммерческой дирекции Wildberries, сопровождаются интенсивным документооборотом и многоканальной обработкой информации, охватывающей весь жизненный цикл продаж на маркетплейсе — от формирования клиентского спроса до аналитики результатов. Управление коммерческой деятельностью маркетплейса требует обработки как внешних, так и внутренних информационных потоков, охватывающих взаимодействие с покупателями, поставщиками, аналитическими системами и платформами автоматизации.

Входящие потоки включают рыночные данные, заказы клиентов, заявки от поставщиков, финансовые лимиты, а также корпоративные ограничения, такие как бюджетная политика, показатели эффективности и стратегия взаимодействия с партнерами. Эти данные поступают как от внешних систем, так и от подразделений внутри организации и служат основой для принятия оперативных и стратегических решений.

Внутренние потоки охватывают процессы согласования, анализа и системной обработки данных в рамках информационной системы. Сюда входят модули обработки заказов, аналитики продаж, формирования

отчетности, а также интеграции с CRM и ERP-системами. В результате формируются исходящие потоки: утвержденные планы продаж, отчеты по динамике выручки, ежедневные и еженедельные аналитические отчеты.

С целью повышения прозрачности и управляемости этих потоков была спроектирована собственная информационная система. Это позволяет идентифицировать узкие места, обеспечить корректность бизнес-логики и повысить надежность автоматизации.

Ниже представлена таблица ключевых информационных потоков с указанием их типа, источников, получателей и роли в системе.

**Таблица 3**

**Документооборот отдела коммерческой дирекции**

<b>Документ / Поток данных</b>	<b>Тип</b>	<b>Источник</b>	<b>Получатель</b>	<b>Назначение / Взаимосвязь</b>
Заказы от клиентов	Входящий	Пользователь	Система заказов	Основной элемент коммерческой обработки
Заявки от поставщиков	Входящий	Поставщик	Система поставок	Основа для формирования ассортимента
Рыночные данные	Входящий	ВІ-инструменты, аналитика рынка	Коммерческий отдел	Определение спроса и трендов
Финансовые лимиты	Входящий	Финансовый отдел	Модуль контроля бюджета	Ограничения на закупочную деятельность
План продаж	Исходящий	Система планирования	Коммерческий директор, аналитики	Утвержденные цели по выручке
Отчеты по выручке	Исходящий	Система аналитики	Руководство, финансовый отдел	Контроль выполнения плана
Ежедневные/еженедельные отчеты	Исходящий	Информационная система	Все заинтересованные стороны	Оперативная отчетность и анализ
Статусы заказов и поставок	Внутренний	Система	Менеджеры, аналитики	Контроль хода исполнения
Данные из CRM / ERP-систем	Входящий	Сторонние платформы	Информационная система	Синхронизация клиентской и



				товарной информации
--	--	--	--	---------------------

**Таблица 4**

**Оценка потоков информации**

№	Поток данных	Примерное кол-во транзакций в месяц	Средние трудовые затраты на 1 транзакцию (мин)	Общие затраты в месяц (часы)
1	Заказы клиентов	15,000	1	250
2	Заявки поставщиков	2,000	2	67
3	Отчеты по выручке	60 (ежедневные + еженедельные)	15	15
4	Планы продаж	12	30	6
5	BI-аналитика и данные CRM/ERP	1000	5	83

**1.2.4. Анализ системы обеспечения информационной безопасности и защиты информации**

Система обеспечения информационной безопасности в компании Wildberries выстроена в соответствии с актуальными стандартами и законодательными требованиями, направлена на защиту конфиденциальной, персональной и коммерческой информации, циркулирующей в рамках внутренней деятельности организации. В условиях масштабной цифровой инфраструктуры, включающей центры обработки данных, распределенную логистическую сеть, офисные и складские подразделения, информационная безопасность приобретает стратегическое значение.

Организационная модель безопасности включает строгое разграничение прав доступа к информации, определение зон ответственности сотрудников и регламентацию работы с критически важными данными. Все сотрудники компании обязаны подписывать соглашения о неразглашении (NDA), проходят обучение по вопросам защиты информации и регулярно информируются о потенциальных угрозах и способах предотвращения инцидентов. Руководящими документами в области информационной безопасности являются внутренняя политика информационной безопасности,

регламенты по обработке персональных данных и правила эксплуатации корпоративных ИТ-систем.

Контроль доступа к информации реализуется с использованием доменной авторизации, многофакторной аутентификации и автоматического блокирования учетных записей при выявлении подозрительной активности. Доступ к конфиденциальным данным, таким как данные клиентов, отчетность, договора, финансовые документы и логистика, строго ограничен и предоставляется только при наличии соответствующих полномочий.

Технические меры защиты информации включают:

- централизованную антивирусную защиту и систему предотвращения вторжений (IDS/IPS);
- межсетевые экраны и прокси-серверы для фильтрации внешнего и внутреннего трафика;
- сегментацию корпоративной сети по зонам доверия;
- регулярное резервное копирование данных и хранение копий в изолированных хранилищах;
- шифрование файлов, баз данных и каналов передачи информации;
- автоматизированные системы журналирования и мониторинга действий пользователей и администраторов.

Физическая защита информации обеспечивается контролем доступа на территорию складов, офисов и серверных помещений с применением электронных пропусков, видеонаблюдения, охранной сигнализации и круглосуточной охраны. Дополнительно все устройства хранения информации, в том числе корпоративные ноутбуки, флеш-накопители и внешние диски, подлежат учету и шифрованию, а их перемещение контролируется.

Защита персональных данных клиентов и сотрудников осуществляется в строгом соответствии с требованиями ФЗ №152 «О персональных данных». Назначены ответственные лица за организацию обработки персональных данных, ведется реестр обрабатываемой информации,

реализована система ведения согласий на обработку данных, а также механизм отзыва согласий в случае необходимости.

Реакция на инциденты проработана в рамках плана реагирования на инциденты информационной безопасности, включающего:

- механизм оперативного выявления и локализации инцидента;
- процедуру уведомления руководства и ИБ-отдела;
- алгоритмы расследования, документирования и анализа причин;
- меры по восстановлению нормального функционирования и предотвращению повторных угроз.

Системы информационной безопасности Wildberries проходят регулярные аудиты и тестирование, включая внешние независимые проверки на соответствие корпоративным и международным стандартам (например, ISO/IEC 27001). Все выявленные уязвимости устраняются в плановом порядке, с последующим контролем исправлений.

### **1.3 Анализ существующих разработок и выбор стратегии автоматизации («КАК ДОЛЖНО БЫТЬ»)**

#### **1.3.1. Анализ существующих разработок для автоматизации задачи**

При анализе существующих программных решений, предназначенных для автоматизации процессов взаимодействия между покупателями и продавцами, были рассмотрены наиболее распространённые платформы, применяемые в электронной коммерции, торгово-логистических системах и маркетплейсах. Основное внимание уделено возможностям по управлению заказами, работе с ассортиментом, обработке клиентских запросов, а также удобству интерфейса для конечных пользователей — как со стороны продавца, так и покупателя.

Wildberries Seller — внутренняя система личного кабинета продавца на маркетплейсе Wildberries. Предоставляет широкий функционал: управление товарами, учёт остатков, обработка заказов, аналитика по продажам. Однако интерфейс ориентирован исключительно на продавцов и не предусматривает взаимодействия с покупателями напрямую, кроме отзывов и оценок.

Ozon Seller и Ozon App — решения от маркетплейса Ozon. Покупателям доступен полнофункциональный мобильный интерфейс для выбора товаров, оплаты и отслеживания заказов. Продавцы получают отдельную платформу для управления складом, логистикой, финансовыми отчётами. Несмотря на высокий уровень автоматизации, системы строго разграничены, не позволяют гибкой настройки под конкретный бизнес-процесс, а API-доступ требует технической подготовки.

AliExpress (Seller Center + App) — международная платформа с поддержкой как веб-, так и мобильных интерфейсов. Покупатели и продавцы работают в разных экосистемах. Продавец получает мощный инструментарий, но с ограничениями по локализации, интеграции с локальными службами доставки и нестабильной техподдержкой для малого бизнеса.

МойСклад + Интернет-магазин на базе CMS — часто применяемая связка в небольших компаниях. МойСклад предоставляет управление заказами, складом, клиентами, а CMS-платформа (например, OpenCart, WooCommerce) реализует интерфейс покупателя. Требуется дополнительных затрат на настройку, доработку и интеграцию между системами.

**Таблица 5**

**Сравнение существующих решений**

Особенности	Wildberries Seller	Ozon Seller / App	AliExpress	МойСклад + CMS	Планируемая система
Управление товарами и заказами	Да	Да	Да	Да	Да
Интерфейс покупателя	Отдельно	Да	Да	Через CMS	Встроенный
Интерфейс продавца	Да	Да	Да	Да	Да
Обратная связь покупатель—продавец	Ограничена	Ограничена	Ограничена	Частично	Да (встроенный чат/отзывы)
Кастомизации	Низкая	Средняя	Низкая	Средняя	Высокая

я под бизнес- процессы					
Интеграция с внутренним и ИС	Нет	Ограничен а	Нет	Да	Да
Стоимость	Условно бесплатна	Условно бесплатна	% от продаж	Средняя	Минимальна я
Поддержка мобильных устройств	Да	Да	Да	Через CMS	Да

Разрабатываемое приложение будет ориентировано на обе стороны: покупателя и продавца. Это ключевое отличие от большинства существующих решений, где интерфейсы жёстко разделены или требуют внешней интеграции. Использование существующих решений недоступно, так как это будет нарушать политику безопасности компании WB. И у компании WB отсутствует аналог десктопного приложения для покупателей и продавцов

Особенности будущей системы:

- Продавец получает удобный инструмент для управления товарами, заказами, ценами и аналитикой.
- Покупатель — простой и понятный интерфейс для выбора товаров, оформления и отслеживания заказов, а также взаимодействия с продавцом.

### **1.3.2. Выбор и обоснование стратегии автоматизации задачи**

Для повышения эффективности работы продавцов и удобства покупательского опыта в рамках платформы, функционирующей по типу маркетплейса (наподобие Wildberries), была сформулирована стратегия комплексной автоматизации взаимодействия между участниками торгового процесса. Проведённый анализ текущего состояния показал, что многие внутренние процессы — такие как учёт заказов, управление товарами, обработка заявок и общение с клиентами — зачастую ведутся с использованием разрозненных и неинтегрированных инструментов. Это

приводит к задержкам в обработке заказов, высоким трудозатратам на рутинные операции, риску человеческих ошибок и снижению уровня удовлетворённости клиентов.

С учётом выявленных проблем и бизнес-целей организации, автоматизация охватывает два ключевых направления:

- 1) Функционал для продавца: управление товарами, остатками, заказами, ценами, отслеживание аналитики продаж, обработка отзывов и прямое взаимодействие с клиентом;
- 2) Функционал для покупателя: простой интерфейс поиска и выбора товаров, оформление и отслеживание заказов, система обратной связи, уведомления о статусе покупок и возможность оценки продавца.

Основными требованиями к системе стали:

- 1) Интеграция всех процессов в единое информационное пространство;
- 2) Минимизация ручного ввода данных;
- 3) Простота использования даже для малых поставщиков с минимальными IT-навыками;
- 4) Возможность масштабирования и адаптации под дальнейшее развитие бизнеса;
- 5) Поддержка кроссплатформенной работы, включая мобильные устройства;
- 6) Надёжная защита данных пользователей и сохранение истории операций.

Рассматривались три возможных стратегии автоматизации:

- 1) Частичная автоматизация отдельных процессов с помощью сторонних инструментов (например, Excel + облачные CRM/ERP). Этот подход позволил бы сэкономить на первоначальных вложениях, но не решал проблему фрагментации информации и затруднённого взаимодействия между продавцами и покупателями.
- 2) Внедрение готового решения (например, платформа Ozon Seller, CMS + плагин интернет-магазина). Несмотря на наличие обширного

функционала, такие системы сложно адаптировать под специфику бизнес-процессов компании, а стоимость лицензий, техподдержки и доработок делает их экономически нецелесообразными для большинства частных продавцов.

- 3) Разработка специализированного приложения с нуля, ориентированного одновременно на продавца и покупателя, с учётом всех специфических требований и особенностей работы на платформе Wildberries или аналогичных. Эта стратегия была выбрана как оптимальная, поскольку позволяет создать модульную, расширяемую и точно нацеленную на нужды бизнеса систему.

Реализация стратегии будет включать следующие этапы:

- 1) Сбор и уточнение требований от продавцов и конечных пользователей;
- 2) Проектирование архитектуры системы (клиент-сервер, веб-интерфейс, СУБД MySQL);
- 3) Создание пользовательских интерфейсов с учётом роли (покупатель / продавец);
- 4) Разработка ядра системы и ключевых модулей (каталог товаров, заказов, чатов, аналитики);
- 5) Тестирование, отладка и подготовка системы к внедрению;
- 6) Поэтапное внедрение с возможностью масштабирования и подключения новых участников;
- 7) Организация службы технической поддержки и сопровождения.

Выбранная стратегия комплексной автоматизации позволит устранить барьеры в коммуникации между продавцом и покупателем, ускорить все процессы обработки заказа, повысить прозрачность сделок и обеспечить надёжное хранение всей информации в единой системе. В результате, приложение станет эффективным инструментом цифрового взаимодействия сторон, полностью адаптированным под реалии конкретной компании и способным конкурировать с существующими маркетплейсами за счёт гибкости, удобства и ориентации на потребности пользователей.

### **1.3.3. Выбор и обоснование способа приобретения ИС для автоматизации комплекса задач**

В рамках исследования возможных способов автоматизации процессов взаимодействия между продавцом и покупателем был проведён анализ существующих программных решений, включая коммерческие продукты, open-source системы и возможность разработки индивидуального программного обеспечения. Рассматривались такие критерии, как стоимость внедрения, гибкость настройки, удобство пользовательского интерфейса и возможность работы на локальных устройствах без постоянного подключения к интернету.

Готовые решения, такие как 1С:Управление торговлей, МойСклад или облачные сервисы маркетплейсов (Wildberries, Ozon, AliExpress), предоставляют базовые инструменты для учёта заказов, аналитики и управления ассортиментом. Однако данные системы в большинстве случаев:

- требуют постоянного доступа к интернету (что неприемлемо для автономной локальной работы);
- имеют сложный интерфейс, не адаптированный под узкие задачи конкретного бизнеса;
- предполагают регулярные лицензионные платежи;
- ограничены в возможностях изменения логики работы и расширения функционала.

Вариант использования open-source решений, таких как Odoo, ERPNext или OpenCart, был отклонён по ряду причин. Во-первых, эти системы в основном ориентированы на веб-приложения и требуют установки серверной инфраструктуры. Во-вторых, их кастомизация под локальный WinForms-интерфейс является трудоёмкой и экономически нецелесообразной.

С учётом вышеизложенного было принято решение о разработке собственной информационной системы с использованием платформы



Windows Forms (WinForms) на языке программирования C# с подключением к базе данных MySQL. Такой подход обеспечивает ряд преимуществ:

- полное соответствие требованиям бизнеса;
- автономная работа приложения на клиентских машинах;
- простота интерфейса, адаптированного под конкретные роли (покупатель / продавец);
- отсутствие зависимости от сторонних платформ и серверов;
- гибкость в расширении функционала в будущем;
- возможность реализации строгих политик безопасности и разграничения прав доступа.

Приложение будет реализовано как десктопная система с возможностью подключения к локальному или удалённому серверу БД. Архитектура спроектирована по принципу клиент-серверного взаимодействия с модульной структурой, что упростит дальнейшее масштабирование и сопровождение.

Таким образом, выбор в пользу внутренней разработки на WinForms обусловлен необходимостью создать специализированное, адаптированное и экономически эффективное решение для автоматизации взаимодействия между продавцом и покупателем, с полной локализацией бизнес-логики и минимальными затратами на поддержку.

## **Глава 2. Проектная часть**

### **2.2 Разработка проекта автоматизации**

#### **2.1.1 Этапы жизненного цикла проекта автоматизации**

##### **Анализ требований и планирование**

На начальном этапе было проведено исследование потребностей пользователей маркетплейса Wildberries. Анализ выявил ключевые проблемы: отсутствие удобного десктопного интерфейса для работы с маркетплейсом, необходимость дублирования действий в веб-интерфейсе и сторонних программах, а также сложности с оперативным управлением товарами и заказами. Были сформулированы основные требования: разработка Windows-приложения на C# (WinForms), интеграция с API Wildberries через официальные методы, реализация отдельного интерфейса для покупателей и продавцов, обеспечение оффлайн-работы с синхронизацией при подключении.

##### **Проектирование системы**

Архитектура системы включает клиентскую часть, серверную часть и базу данных. Клиентская часть (WinForms) состоит из модулей: Auth.cs (авторизация), SellerForm.cs (личный кабинет продавца), CustomerForm.cs (интерфейс покупателя), AddProductForm.cs/EditProductForm.cs (управление товарами), SellerOrdersForm.cs (обработка заказов) и StatisticsForm.cs (аналитика продаж). Серверная часть включает REST API для взаимодействия с Wildberries, локальную БД (MySQL) для кэширования данных и службу синхронизации. База данных спроектирована как реляционная структура с таблицами: пользователи (users), товары (products), заказы (orders), категории (categories) и отзывы (reviews).

##### **Реализация**

Разработка велась на стеке: язык C#, платформа .NET Framework 4.8, СУБД MySQL, среда разработки Visual Studio. Основные функциональные модули: авторизация (Auth.cs) с поддержкой двухфакторной аутентификации и разделением ролей (покупатель/продавец), управление товарами (добавление/редактирование карточек, пакетная загрузка изображений,

управление остатками), работа с заказами (отслеживание статусов, формирование этикеток, история транзакций).

### Тестирование

Проведены тесты методом черного ящика, UI-тесты (WinAppDriver) и нагрузочное тестирование. Это позволило убедиться в корректности работы всех компонентов системы, стабильности API и отзывчивости интерфейса при высокой нагрузке.

### Эксплуатация и поддержка

В процессе эксплуатации предусмотрены регулярные обновления приложения, мониторинг работы API, техническая поддержка пользователей и резервное копирование локальных данных. Перспективы развития включают добавление модуля чата между покупателями и продавцами, интеграцию с системами складского учета и интеграцией с мобильной версией приложения.



**Рисунок 5. Каскадная модель жизненного цикла**

#### **2.1.2. Ожидаемые риски на этапах жизненного цикла и их описание**

На этапе анализа требований ключевым риском является неполное или противоречивое формулирование требований, что может привести к несоответствию системы ожиданиям пользователей. Отсутствие четкого технического задания способно вызвать затягивание сроков и бесконечные правки. Особую опасность представляют неучтенные ограничения API Wildberries, которые могут заблокировать реализацию важных функций. Для минимизации этих рисков необходимо проводить детальные интервью с

пользователями, тщательно анализировать документацию API и создавать прототипы интерфейса для уточнения требований.

При проектировании системы возникает риск столкнуться со сложностями интеграции с закрытым API Wildberries, что может ограничить функциональность или потребовать обратного проектирования. Неправильный выбор архитектуры базы данных способен привести к проблемам с производительностью при увеличении объема данных. Отсутствие резервных сценариев для оффлайн-работы грозит потерей информации при разрыве соединения. Чтобы избежать этих проблем, следует заранее тестировать API, оптимизировать структуру БД с использованием индексов и нормализации, а также реализовать надежное локальное кэширование критически важных данных.

На этапе реализации разработчики могут столкнуться с нестабильностью WinForms, проявляющейся в утечках памяти и зависании интерфейса. Проблемы с многопоточностью способны блокировать пользовательский интерфейс во время загрузки данных. Особую опасность представляют неожиданные изменения в API Wildberries, которые могут нарушить работу системы. Для предотвращения этих рисков рекомендуется использовать современные паттерны проектирования, тщательно тестировать работу с большими объемами данных и разрабатывать гибкую систему обработки ошибок API.

Тестирование системы сопряжено с риском недостаточного покрытия тестами, что может привести к появлению критических ошибок в рабочей версии. Автоматизация UI-тестов для WinForms представляет определенные сложности, вынуждая тратить значительное время на ручное тестирование. Существует опасность расхождения данных между локальной базой и Wildberries, что снижает актуальность информации. Для обеспечения качества необходимо внедрять модульное тестирование с использованием NUnit и Moq, регулярно синхронизироваться с тестовым окружением Wildberries и вести подробное логирование всех операций с API.

При внедрении системы возможны сопротивление пользователей, нежелающих переходить с веб-версии, проблемы с установкой на устаревших компьютерах из-за отсутствия нужной версии .NET Framework, а также ошибки при переносе данных, ведущие к потере истории заказов. Для успешного развертывания следует подготовить подробные инструкции и обучающие материалы, проверять системные требования перед установкой и разработать надежный инструмент миграции данных.

На этапе эксплуатации и поддержки основными рисками являются нехватка ресурсов для своевременного обслуживания системы, приводящая к накоплению технического долга, внезапные изменения политики Wildberries, которые могут ограничить доступ к API, а также уязвимости безопасности, способные вызвать утечку пользовательских данных. Для стабильной работы необходимо планировать регулярные обновления, внимательно отслеживать изменения в документации Wildberries и обеспечивать надежное шифрование конфиденциальной информации.

## **2.2. Информационное обеспечение задачи**

### **2.2.1. Характеристика нормативно-справочной, входной и оперативной информации**

Нормативно-справочная информация

В нашей базе данных MySQL к этой категории относятся:

- 1) Таблица categories (категории товаров с иерархией);
- 2) Таблица users (роли: продавец/покупатель, права доступа);
- 3) Таблица delivery\_points (пункты выдачи заказов);
- 4) Таблица product\_attributes (единицы измерения, характеристики).

Оперативная информация

Основные таблицы в MySQL:

- 1) Products (текущие остатки, цены);
- 2) Orders (статусы: новый/в работе/доставлен);
- 3) Transactions (платежи, возвраты).

Таблица 6

## Характеристика справочников

Название справочника	Ответственный за ведение	Средний объем (записей)	Средняя частота актуализации	Средний объем актуализации
Категории товаров	Администратор системы	500	Ежемесячно	10%
Пользователи	Администратор безопасности	1 000	Еженедельно	5%
Точки выдачи заказов	Логистический менеджер	200	Раз в квартал	8%
Атрибуты товаров	Менеджер по ассортименту	300	Раз в полгода	15%
Статусы заказов	Системный администратор	10	Раз в год	2%
Способы оплаты	Финансовый отдел	5	Раз в 2 года	1%
Типы скидок	Маркетинг	20	Ежеквартально	12%
Склады	Логистика	50	Раз в год	5%
Производители	Менеджер по закупкам	150	Ежемесячно	7%
Единицы измерения	Технический отдел	15	Раз в 3 года	0.5%

## 2.2.2. Характеристика результатной информации

Результативными показателями в процессе работы с ИС WB является формирование отчетных форм, где можно посмотреть корзину покупателя, историю заказов, и статистику покупок для продавца и просмотр всех заказов у покупателя

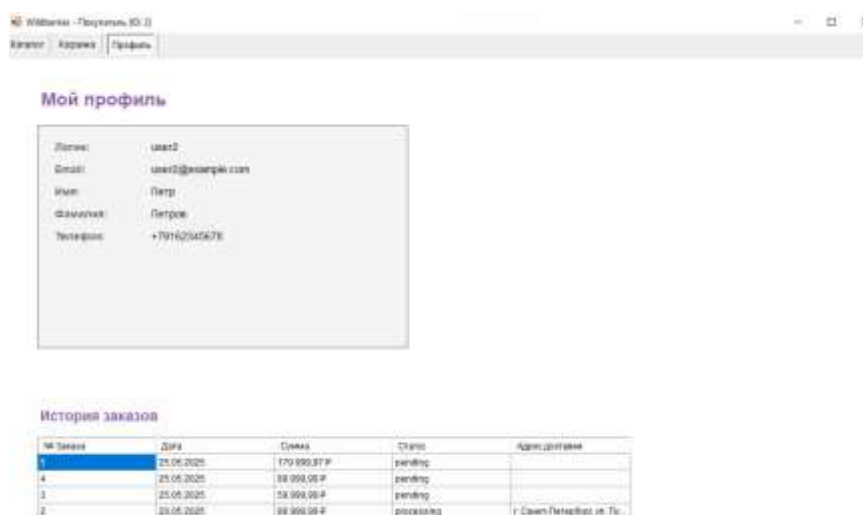


Рисунок 6. Отчетная форма профиль пользователя

Ваша корзина

Товар	Цена	Количество	Сумма
Смартфон X1	59 999.99 Р	1	59 999.99 Р

Итого: 59 999.99 Р

Адрес доставки:  [Добавить адрес](#)

Способ оплаты:  [Оформить заказ](#)

Рисунок 8. Корзина

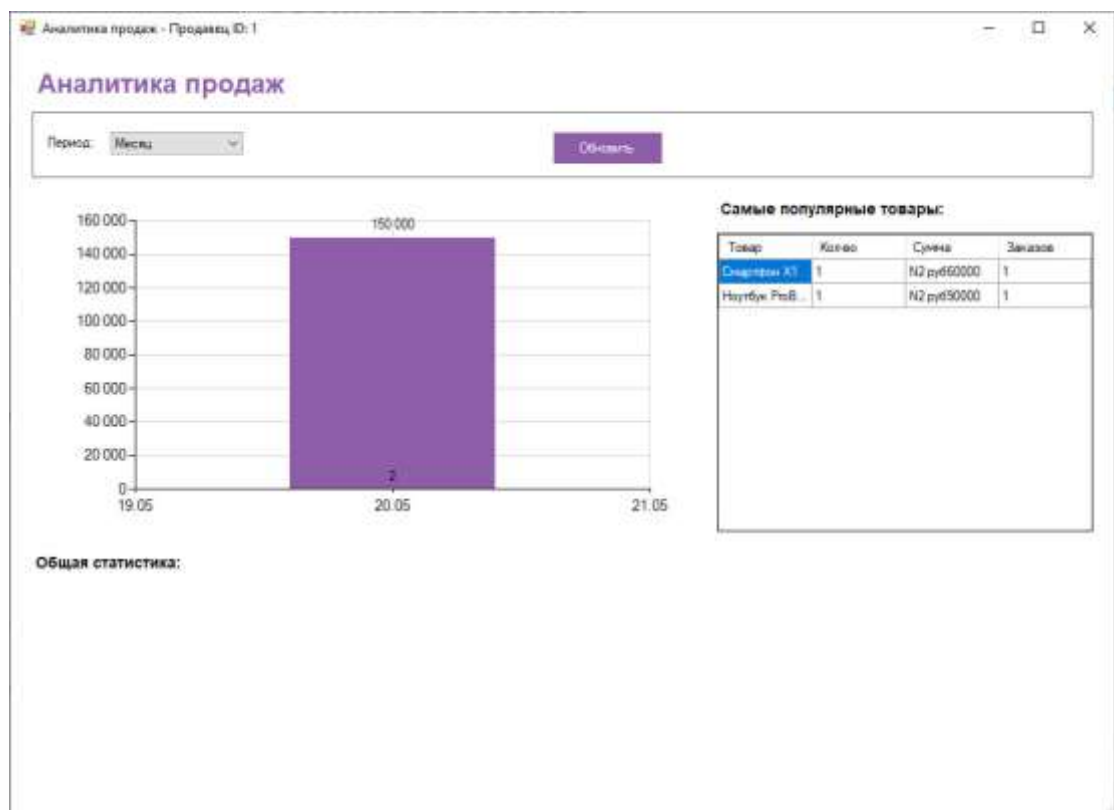


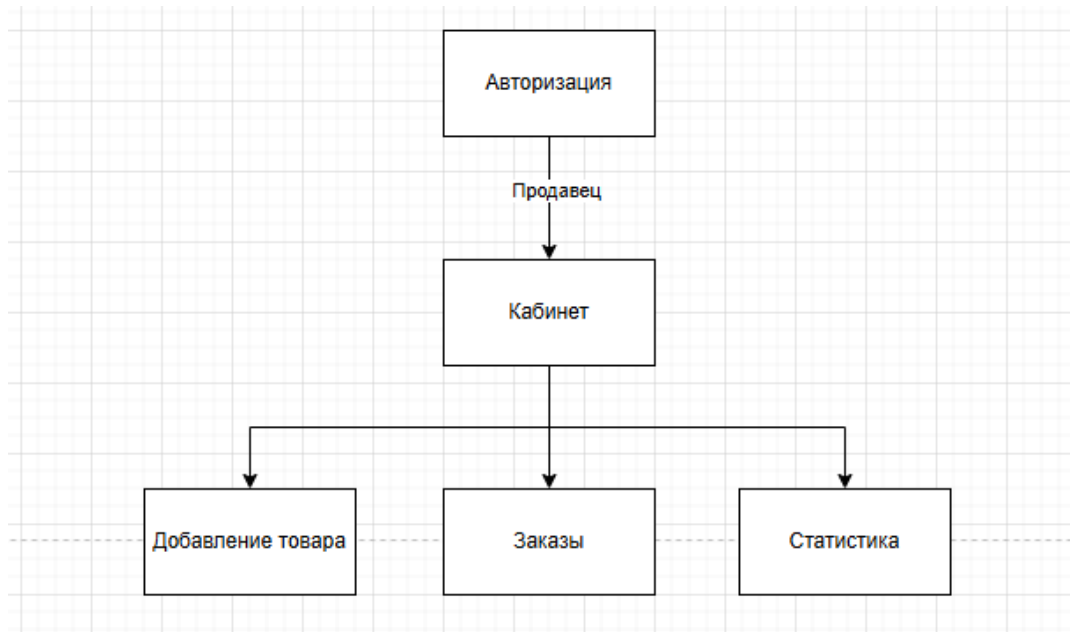
Рисунок 9. Статистика покупок товаров

## 2.3. Программное обеспечение задачи

### 2.3.1. Сценарий диалога

Рассмотрим диалог работы с приложением для ролей покупатель и продавец.

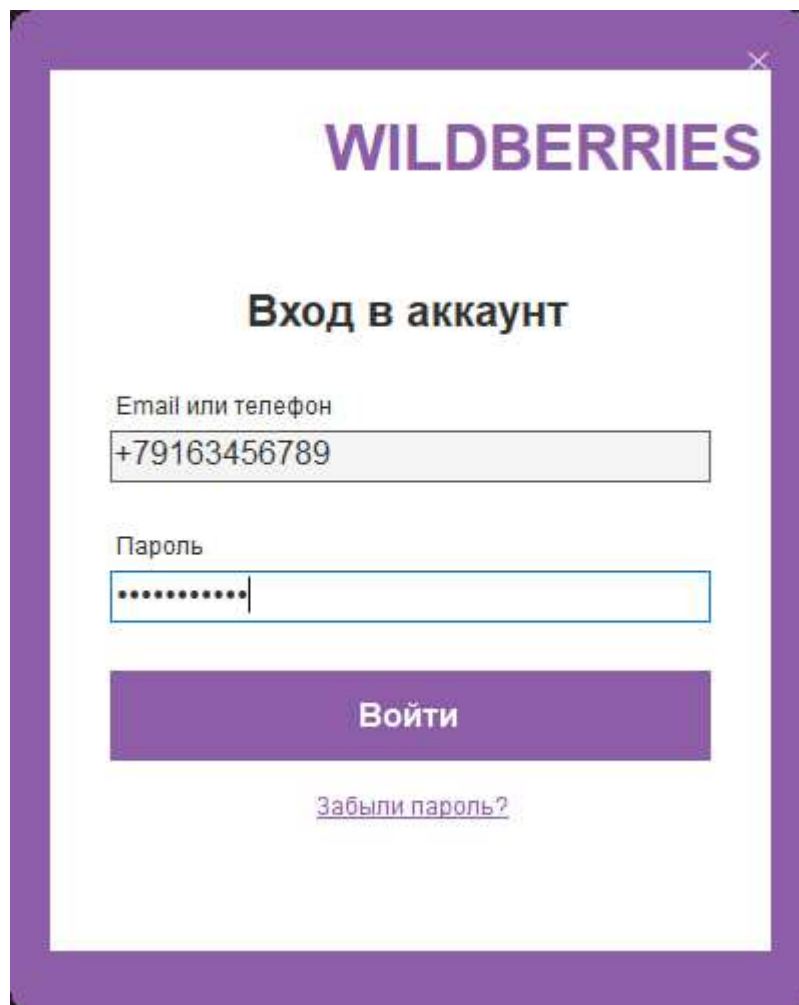
Начнем с продавца:



**Рисунок 7. Сценарий диалога для роли «продавец»**

Перед нами предстает элегантный и функциональный интерфейс десктопного приложения Wildberries, разработанного на WinForms. Первым встречает пользователя форма авторизации - лаконичное окно с полями для ввода email/телефона и пароля, кнопкой входа и ссылкой восстановления пароля. Чистый дизайн без лишних элементов позволяет быстро получить доступ к системе.





WILDBERRIES

Вход в аккаунт

Email или телефон

+79163456789

Пароль

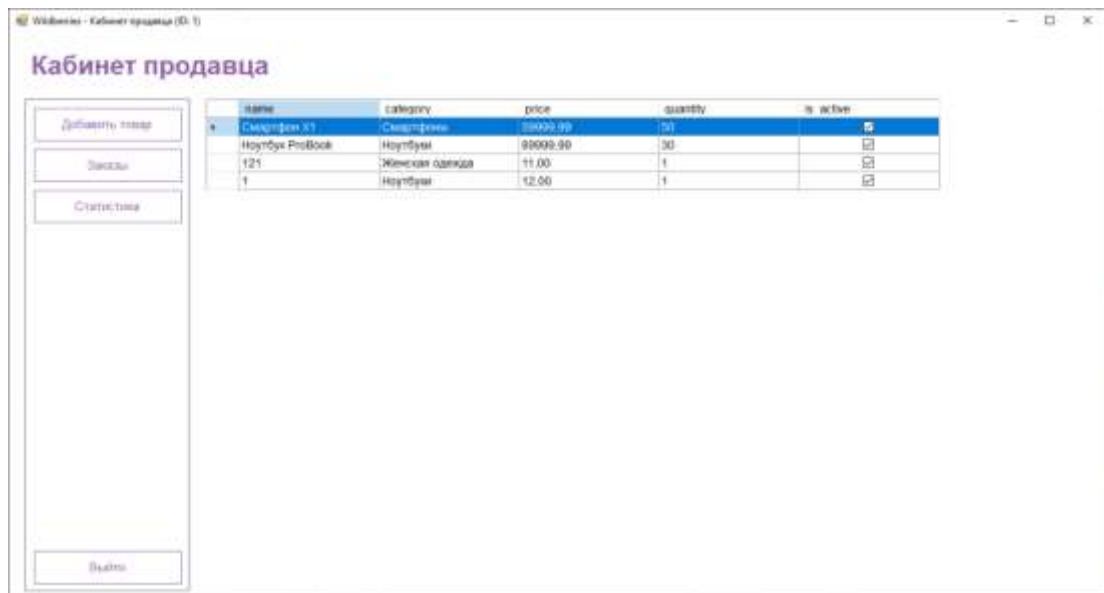
.....

Войти

[Забыли пароль?](#)

**Рисунок 8. Форма авторизации**

После успешного входа система определяет роль пользователя. Если это продавец, перед ним открывается главный рабочий кабинет с интуитивно понятной панелью управления. Центральное место занимает таблица товаров с основной информацией: названиями, категориями, ценами и остатками на складе. В верхней части расположены ключевые разделы для работы: добавление новых товаров, управление заказами и просмотр аналитики продаж.



**Рисунок 9. Вид меню под ролью продавца**

Переход в раздел добавления товара открывает аккуратную форму с полями для заполнения основных характеристик: название, цена, количество, категория и описание. Особое внимание уделено загрузке изображений - кнопка "Выбрать изображение" позволяет легко добавить визуальное представление товара. Две контрастные кнопки внизу формы - "Сохранить товар" и "Отменить"

Добавление нового товара

Название товара:

Цена (руб):

Количество:

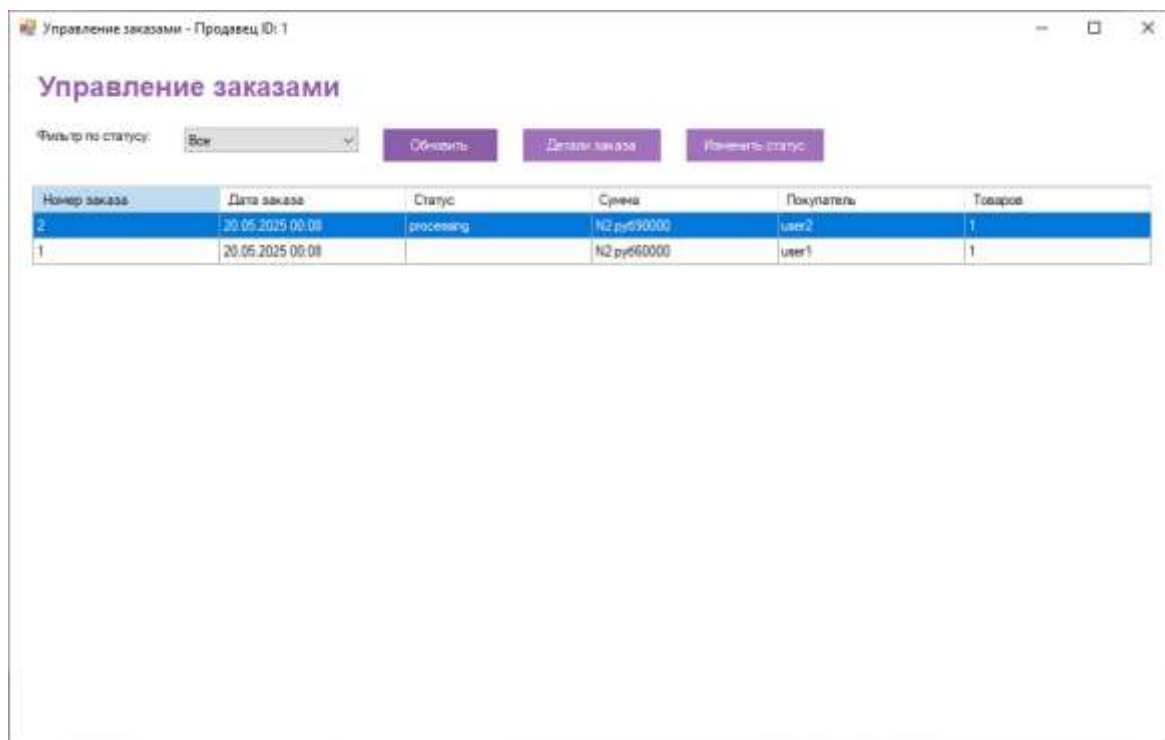
Категория:

Описание:

Изображение товара:

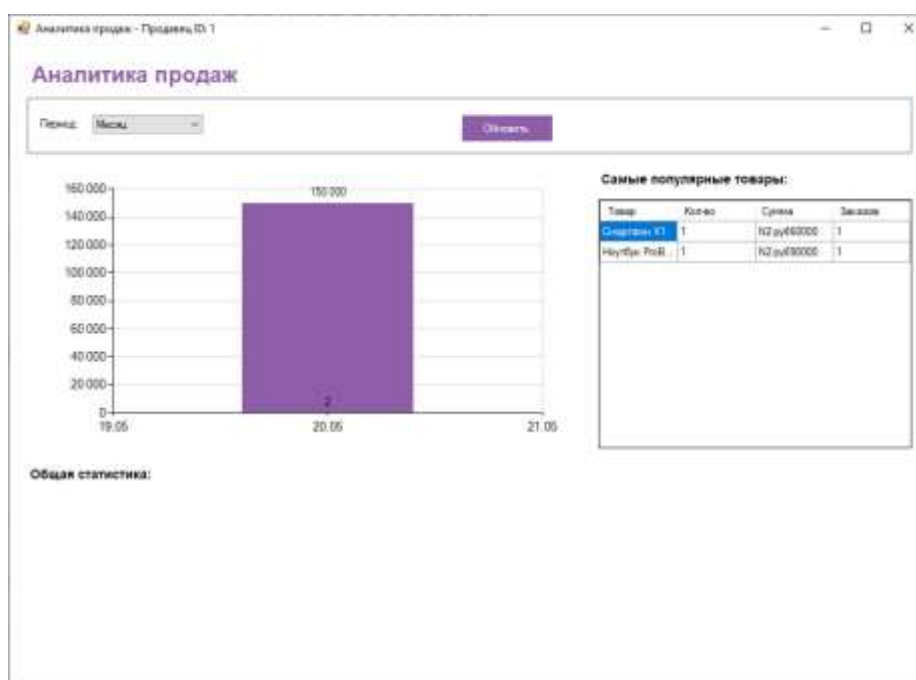
**Рисунок 10. Форма добавление товара**

В разделе управления заказами находится функционал управления заказами. Таблица с номерами заказов, датами, статусами и суммами сопровождается удобными фильтрами. Каждая строка содержит всю необходимую информацию о покупателе и количестве товаров. Особый акцент сделан на возможности быстрого изменения статусов заказов прямо из этой таблицы.



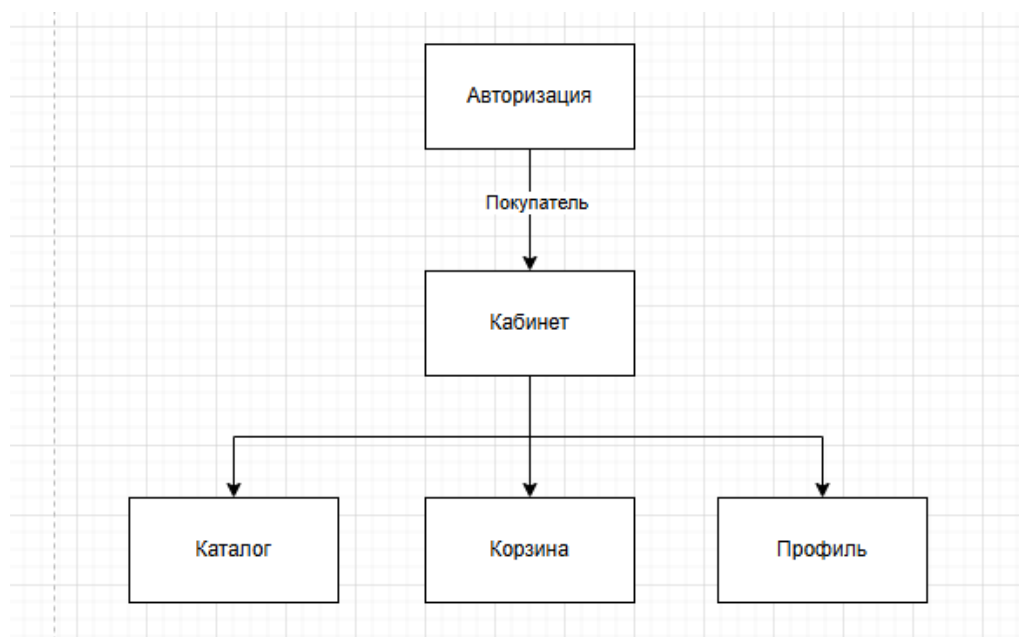
**Рисунок 11. Форма управление заказами**

Завершает функционал продавца раздел аналитики, где данные представлены в наглядном виде. График продаж демонстрирует динамику, а таблица популярных товаров выделяет бестселлеры. Все цифры и показатели обновляются в реальном времени, давая продавцу полную картину его бизнеса на Wildberries.



**Рисунок 12. Форма аналитики продаж**

Теперь, рассмотрим сценарий работы с ИС для покупателя.



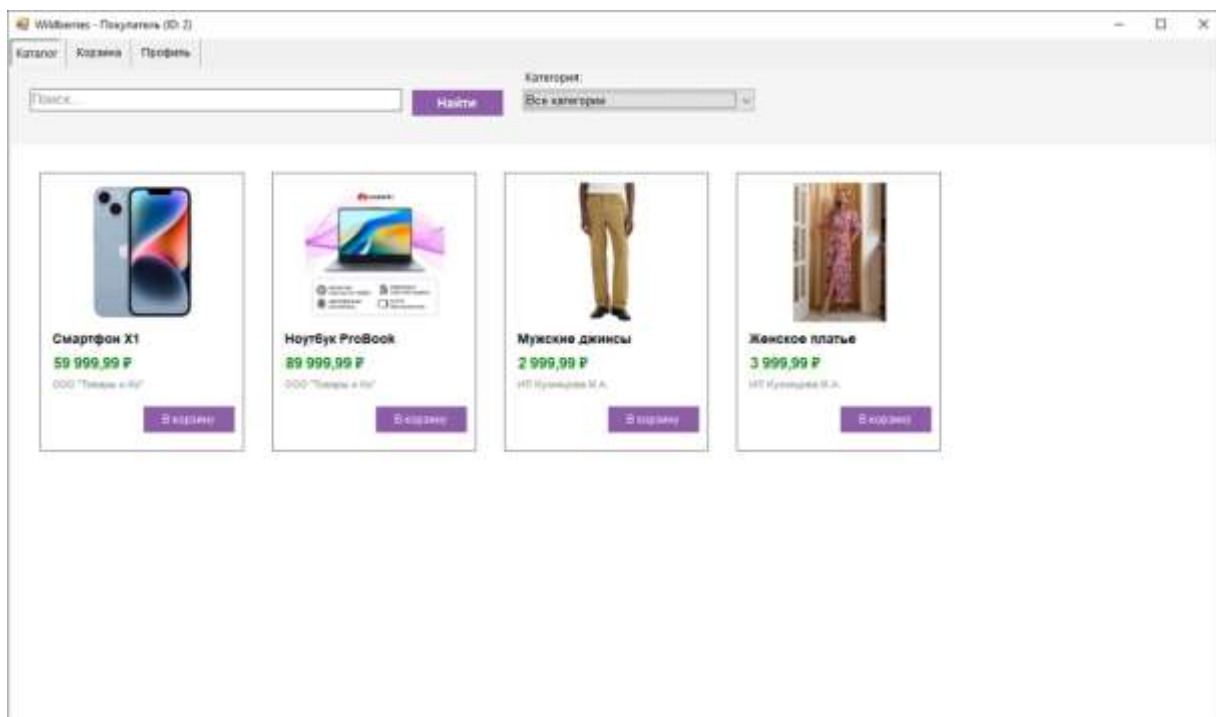
**Рисунок 13. Сценарий диалога для пользователя под ролью «покупатель»**

Все начинается с экрана авторизации - минималистичного окна с полями для ввода номера телефона или электронной почты и пароля. Классическая форма входа содержит все необходимое: основные поля ввода, кнопку подтверждения и ссылку для восстановления пароля, выполненную в корпоративном стиле Wildberries.

The image shows a login window for Wildberries. At the top, the 'WILDBERRIES' logo is displayed in purple. Below it, the title 'Вход в аккаунт' (Login) is centered. There are two input fields: the first is labeled 'Email или телефон' (Email or phone) and contains the number '+79163456789'; the second is labeled 'Пароль' (Password) and contains a series of dots. Below the password field is a large purple button with the text 'Войти' (Login). At the bottom, there is a link that says 'Забыли пароль?' (Forgot password?). The entire form is enclosed in a purple border with a close button in the top right corner.

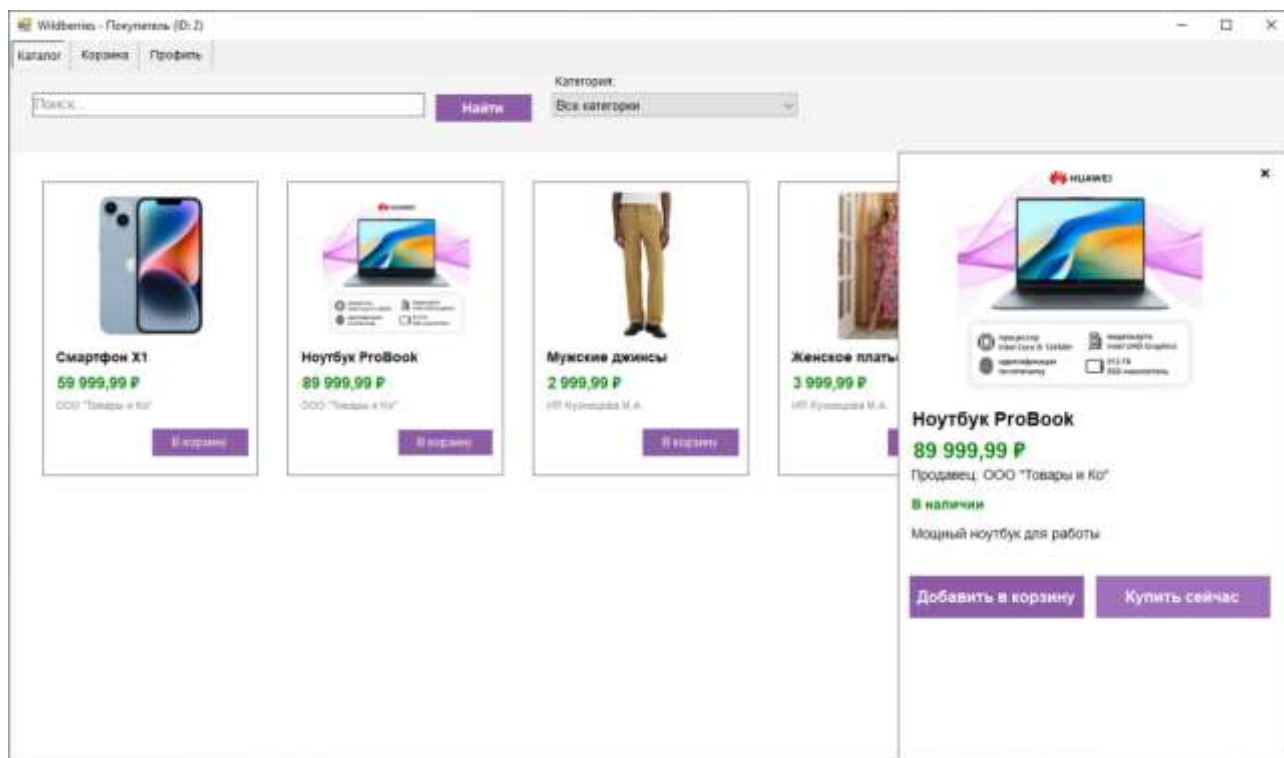
**Рисунок 14. Форма авторизации**

После успешной аутентификации система определяет роль пользователя как "покупатель" и перенаправляет его в главный каталог товаров. Этот экран представляет собой хорошо структурированную витрину с навигационной панелью вверху (каталог, корзина, профиль) и списком товарных карточек. Каждая карточка содержит изображение товара, его название, цену и информацию о продавце, а также заметную кнопку "В корзину", приглашающую к действию.



**Рисунок 15. Каталог товаров**

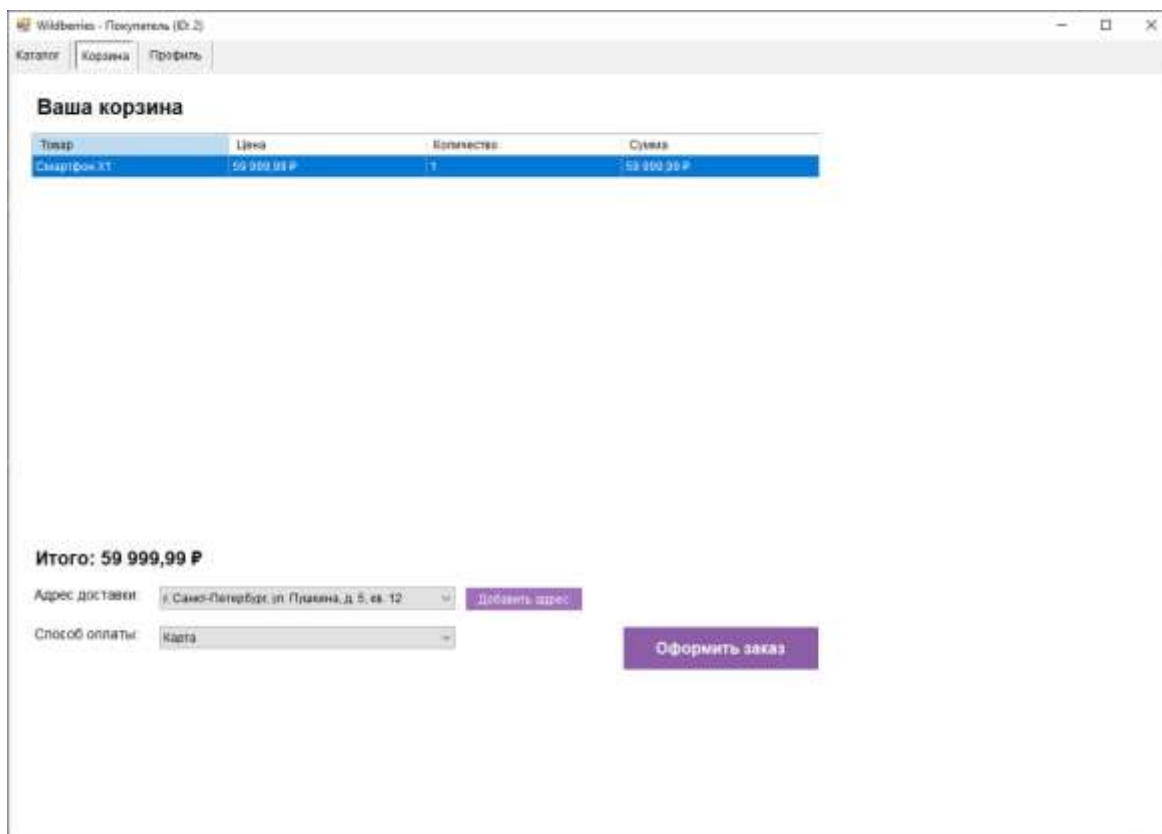
При клике на товар покупатель попадает на его детальную страницу, где представлена расширенная информация: полное название, точная цена с указанием валюты, данные о продавце (наименование компании или ИП), наличие товара на складе и подробное описание характеристик. Здесь же расположены две важные кнопки действий: "Добавить в корзину" для отложенного выбора и "Купить сейчас" для мгновенного оформления.



**Рисунок 16. Карточка товара**

Переход в корзину открывает сводную таблицу выбранных товаров с указанием цены за единицу, количества и общей суммы по каждой позиции. В нижней части экрана система автоматически подсчитывает итоговую стоимость заказа. Особое внимание уделено блоку оформления - здесь покупатель указывает адрес доставки (можно выбрать из сохраненных или добавить новый) и способ оплаты (карта или другие варианты). Завершает композицию крупная кнопка "Оформить заказ".





**Рисунок 17. Корзина покупателя**

Заключительным элементом сценария является личный кабинет покупателя, где собрана вся персональная информация: логин, электронная почта, имя, фамилия и контактный телефон. Особую ценность представляет раздел истории заказов - таблица с номерами заказов, датами их совершения, суммами и текущими статусами обработки. Каждая строка содержит миниатюрную кнопку для просмотра деталей конкретного заказа, включая точный адрес доставки.

### **2.3.2. Характеристика базы данных**

Теперь рассмотрим и осуществим формирование спецификации таблиц базы данных.

**Таблица 7**

<b>users</b>				
<b>Наименование атрибута</b>	<b>Тип данных</b>	<b>Домен</b>	<b>Диапазон значений</b>	<b>Пример атрибута</b>
user_id	INT	Целое число	1–∞	1
username	VARCHAR(50)	Уникальное имя пользователя	Строка до 50 символов	"user123"

email	VARCHAR(100)	Уникальный email	Строка до 100 символов	" <u>user@example.com</u> "
password_hash	VARCHAR(255)	Хэш пароля	Строка до 255 символов	"hashed_password"
first_name	VARCHAR(50)	Имя пользователя	Строка до 50 символов (необязательно)	"Иван"
last_name	VARCHAR(50)	Фамилия пользователя	Строка до 50 символов (необязательно)	"Петров"
phone	VARCHAR(20)	Номер телефона	Строка до 20 символов (необязательно)	"+79123456789"
registration_date	DATETIME	Дата регистрации	Формат "ГГГГ-ММ-ДД ЧЧ:ММ:СС"	"2024-05-30 14:30:00"
last_login	DATETIME	Дата последнего входа	Формат "ГГГГ-ММ-ДД ЧЧ:ММ:СС" (необязательно)	"2024-05-30 15:00:00"
is_seller	BOOLEAN	Флаг продавца	true/false	false

Таблица 8

**sellers**

Наименование атрибута	Тип данных	Домен	Диапазон значений	Пример атрибута
seller_id	INT	Целое число	1—∞	1
user_id	INT	Внешний ключ на users	1—∞	1
company_name	VARCHAR(100)	Название компании	Строка до 100 символов	"ООО Ромашка"
inn	VARCHAR(20)	ИНН продавца	Строка до 20 символов (необязательно)	"1234567890"
bank_account	VARCHAR(50)	Банковский счет	Строка до 50 символов (необязательно)	"40817810099910004312"
rating	DECIMAL(3,2)	Рейтинг продавца	0.00—5.00	4.50

Таблица 9

**categories**

Наименование атрибута	Тип данных	Домен	Диапазон значений	Пример атрибута
category_id	INT	Целое число	1—∞	1
name	VARCHAR(100)	Название категории	Строка до 100 символов	"Электроника"
parent_category_id	INT	Внешний	1—∞	2

		ключ на categories	(необязательно)	
--	--	--------------------	-----------------	--

Таблица 10

products				
Наименование атрибута	Тип данных	Домен	Диапазон значений	Пример атрибута
product_id	INT	Целое число	1–∞	1
seller_id	INT	Внешний ключ на sellers	1–∞	1
category_id	INT	Внешний ключ на categories	1–∞	1
name	VARCHAR(255)	Название товара	Строка до 255 символов	"Смартфон XYZ"
description	TEXT	Описание товара	Текст (необязательно)	"Мощный смартфон..."
price	DECIMAL(10,2)	Цена товара	0.01–99999999.99	19999.99
quantity	INT	Количество на складе	0–∞	100
created_at	DATETIME	Дата создания	Формат "ГГГГ-ММ-ДД ЧЧ:ММ:СС"	"2024-05-30 10:00:00"
updated_at	DATETIME	Дата обновления	Формат "ГГГГ-ММ-ДД ЧЧ:ММ:СС" (необязательно)	"2024-05-30 11:00:00"
is_active	BOOLEAN	Флаг активности	true/false	true

Таблица 11

product_images				
Наименование атрибута	Тип данных	Домен	Диапазон значений	Пример атрибута
image_id	INT	Целое число	1–∞	1
product_id	INT	Внешний ключ на products	1–∞	1
image_url	VARCHAR(255)	URL изображения	Строка до 255 символов	" <a href="https://example.com/image1.jpg">https://example.com/image1.jpg</a> "
is_main	BOOLEAN	Флаг главного изображения	true/false	true

Таблица 6: orders

Наименование атрибута	Тип данных	Домен	Диапазон значений	Пример атрибута
order_id	INT	Целое	1–∞	1

		число		
user_id	INT	Внешний ключ на users	1–∞	1
order_date	DATETIME	Дата заказа	Формат "ГГГГ-ММ-ДД ЧЧ:ММ:СС"	"2024-05-30 12:00:00"
status	orders_status_enum	Статус заказа	"pending", "completed", "cancelled"	"pending"
total_amount	DECIMAL(10,2)	Общая сумма заказа	0.01–99999999.99	19999.99
delivery_address	TEXT	Адрес доставки	Текст	"ул. Ленина, д. 10"
payment_method	orders_payment_method_enum	Способ оплаты	"credit_card", "cash", "bank_transfer"	"credit_card"

Таблица 12

#### order\_items

Наименование атрибута	Тип данных	Домен	Диапазон значений	Пример атрибута
order_item_id	INT	Целое число	1–∞	1
order_id	INT	Внешний ключ на orders	1–∞	1
product_id	INT	Внешний ключ на products	1–∞	1
quantity	INT	Количество товара	1–∞	2
price_per_unit	DECIMAL(10,2)	Цена за единицу	0.01–99999999.99	9999.99

Таблица 13

#### reviews

Наименование атрибута	Тип данных	Домен	Диапазон значений	Пример атрибута
review_id	INT	Целое число	1–∞	1
user_id	INT	Внешний ключ на users	1–∞	1
product_id	INT	Внешний ключ на products	1–∞	1
order_id	INT	Внешний ключ на orders	1–∞	1
rating	TINYINT	Оценка	1–5	5
comment	TEXT	Комментарий	Текст (необязательно)	"Отличный товар!"
review_date	DATETIME	Дата отзыва	Формат "ГГГГ-ММ-ДД ЧЧ:ММ:СС"	"2024-05-30 16:00:00"

Таблица 14

### promotions

Наименование атрибута	Тип данных	Домен	Диапазон значений	Пример атрибута
promotion_id	INT	Целое число	1–∞	1
product_id	INT	Внешний ключ на products	1–∞ (необязательно)	1
category_id	INT	Внешний ключ на categories	1–∞ (необязательно)	1
discount_percent	DECIMAL(5,2)	Процент скидки	0.01–100.00	15.00
start_date	DATETIME	Дата начала	Формат "ГГГГ-ММ-ДД ЧЧ:ММ:СС"	"2024-06-01 00:00:00"
end_date	DATETIME	Дата окончания	Формат "ГГГГ-ММ-ДД ЧЧ:ММ:СС"	"2024-06-30 23:59:59"
description	TEXT	Описание акции	Текст (необязательно)	"Скидка на летнюю коллекцию"

**Таблица 15**

### delivery

Наименование атрибута	Тип данных	Домен	Диапазон значений	Пример атрибута
delivery_id	INT	Целое число	1–∞	1
order_id	INT	Внешний ключ на orders	1–∞	1
tracking_number	VARCHAR(100)	Трек-номер	Строка до 100 символов (необязательно)	"RF123456789"
delivery_service	VARCHAR(50)	Служба доставки	Строка до 50 символов (необязательно)	"Почта России"
estimated_delivery_date	DATE	Ориентировочная дата доставки	Формат "ГГГГ-ММ-ДД" (необязательно)	"2024-06-05"
actual_delivery_date	DATE	Фактическая дата доставки	Формат "ГГГГ-ММ-ДД" (необязательно)	"2024-06-04"
status	VARCHAR(50)	Статус доставки	Строка до 50 символов (необязательно)	"В пути"

На основе проведенного анализа можно построить следующую

логическую и физическую ER-модель

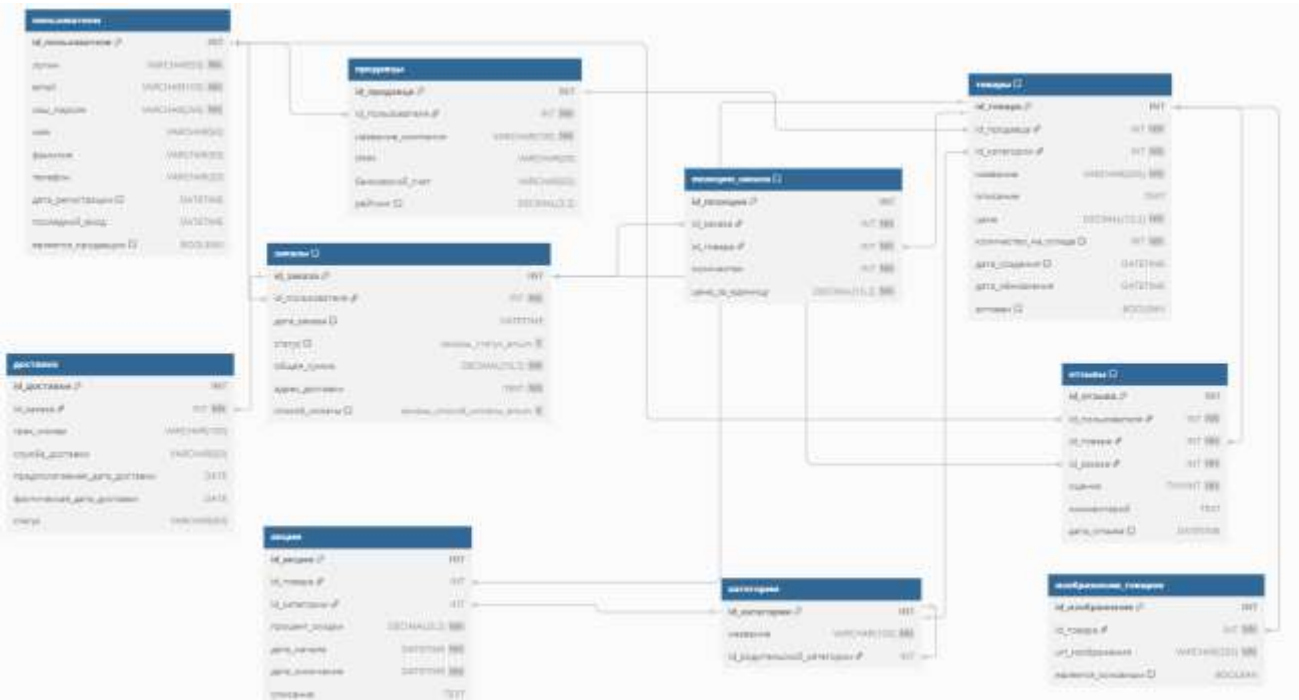


Рисунок 18. Логическая модель БД

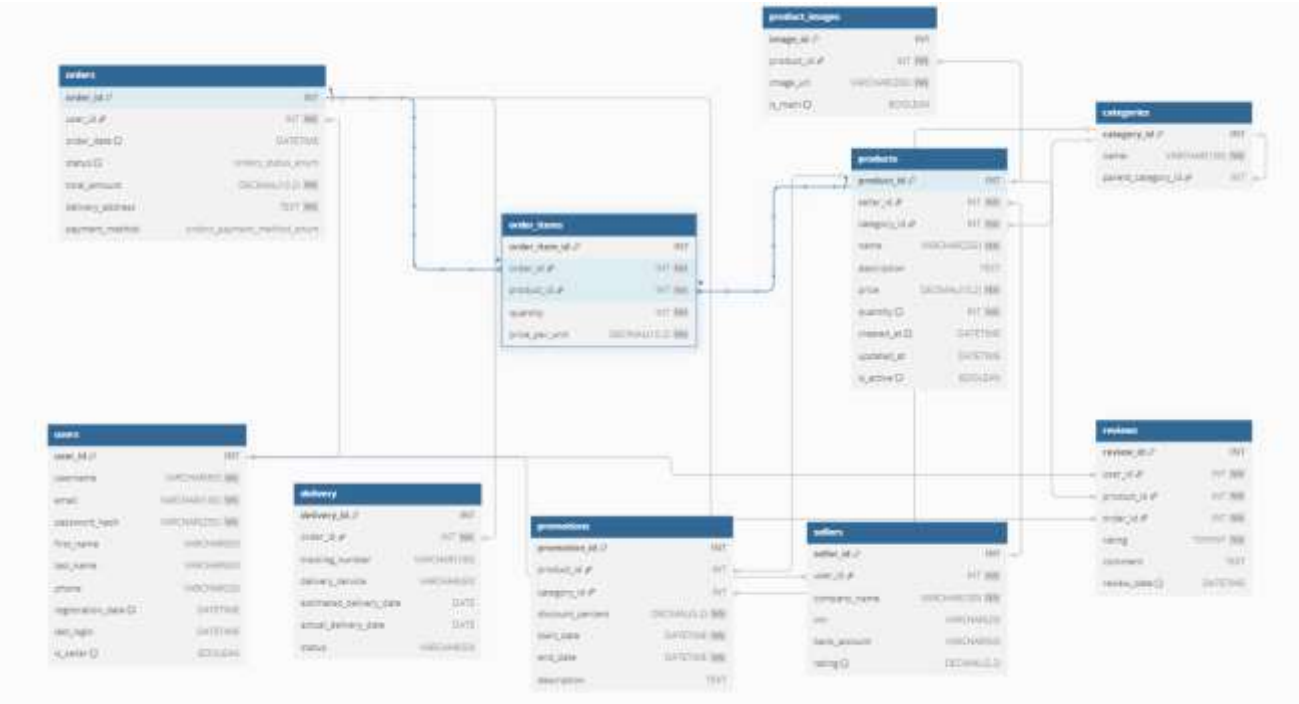
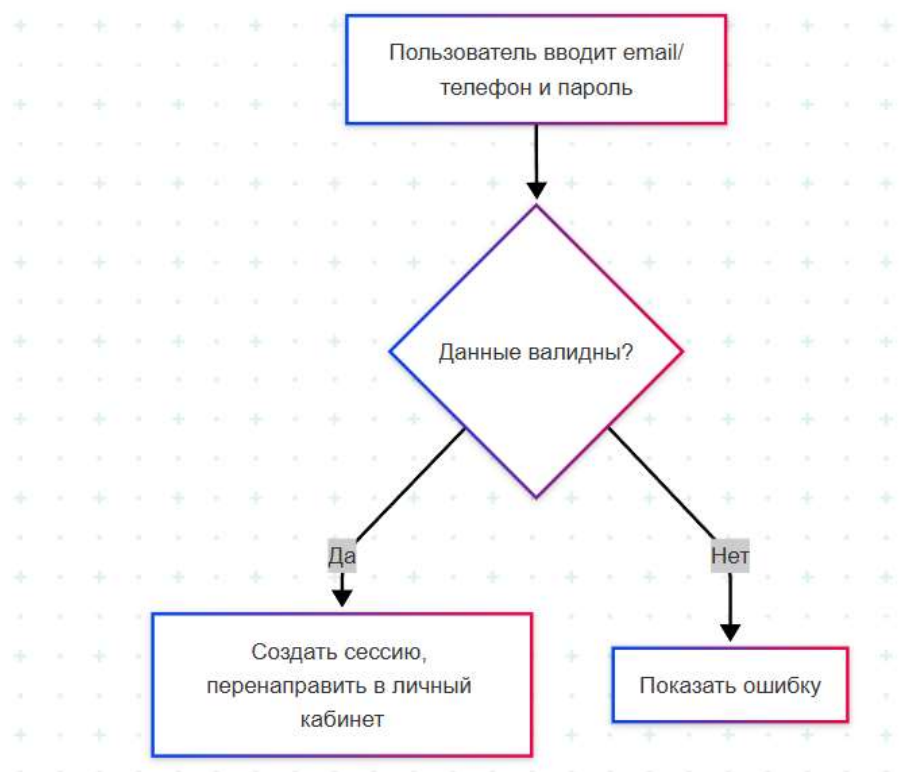


Рисунок 19. Физическая модель

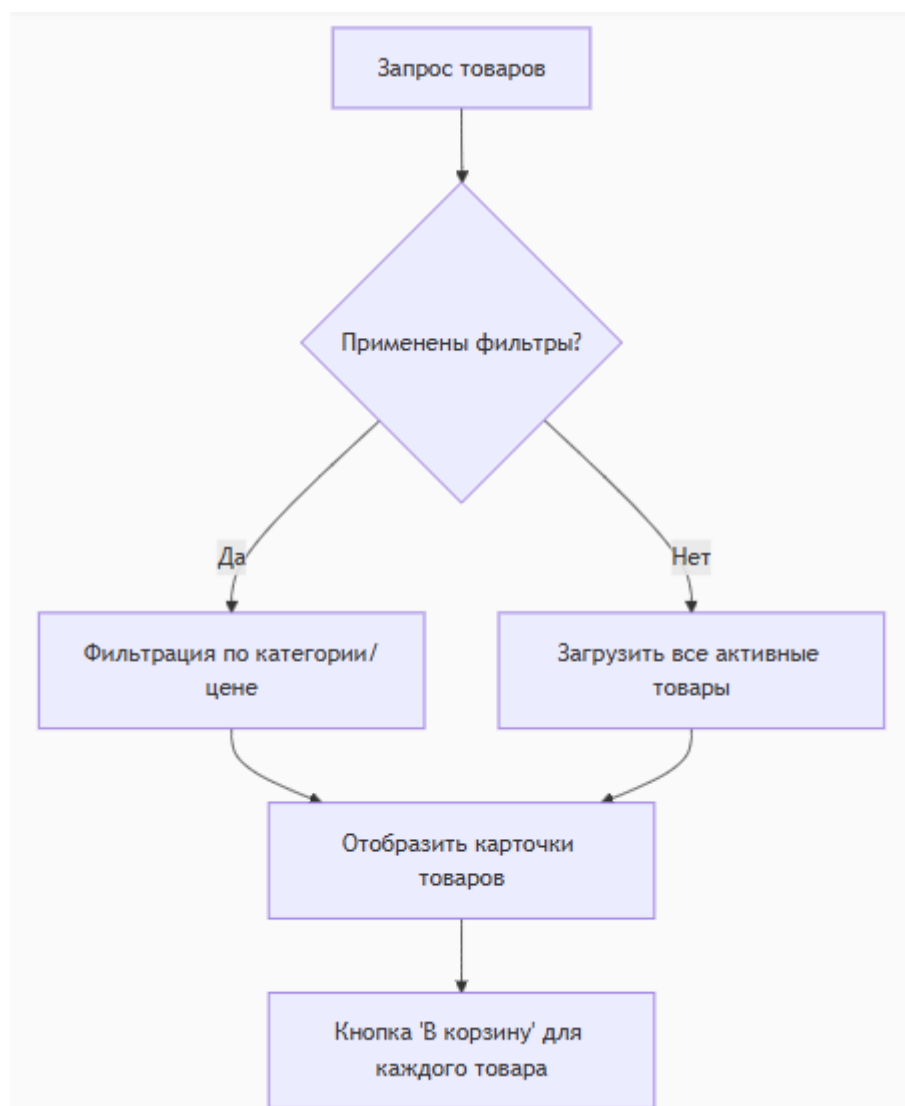
### 2.3.4. Описание программных модулей

1. Алгоритм работы с модулем авторизации выглядит следующим образом:



**Рисунок 20. Модуль авторизации**

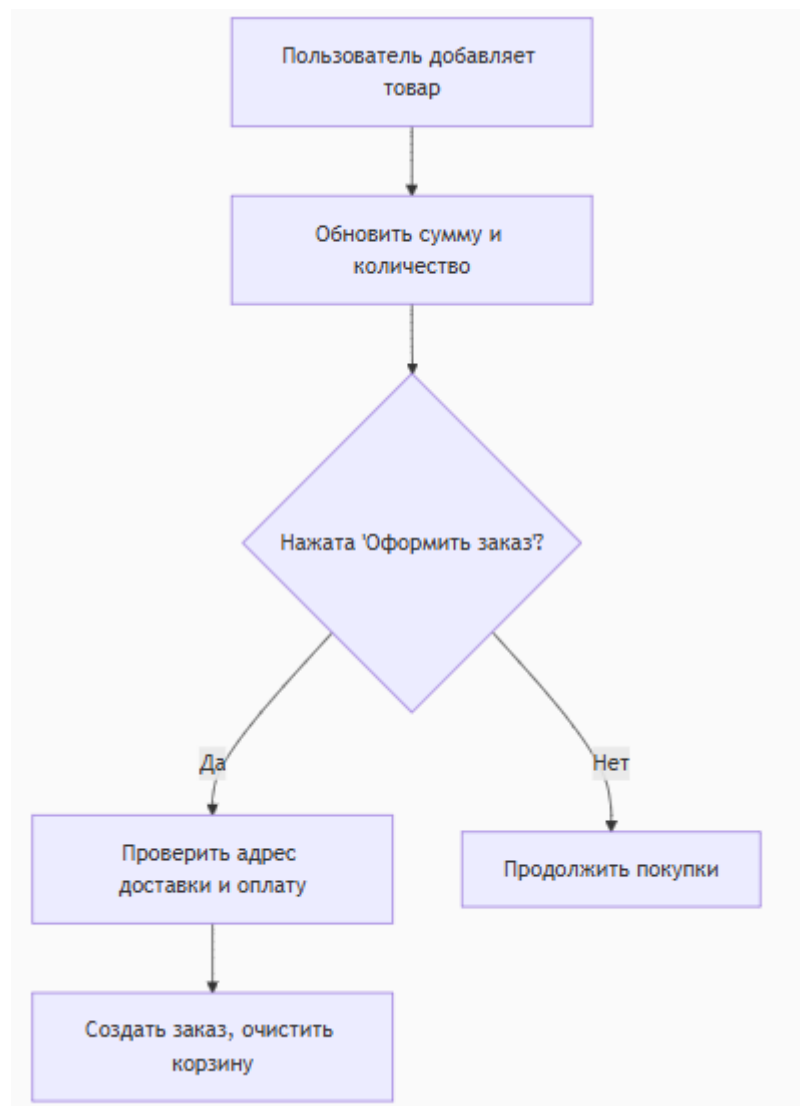
## 2. Модуль каталога товаров:



**Рисунок 21. Модуль каталога товаров**

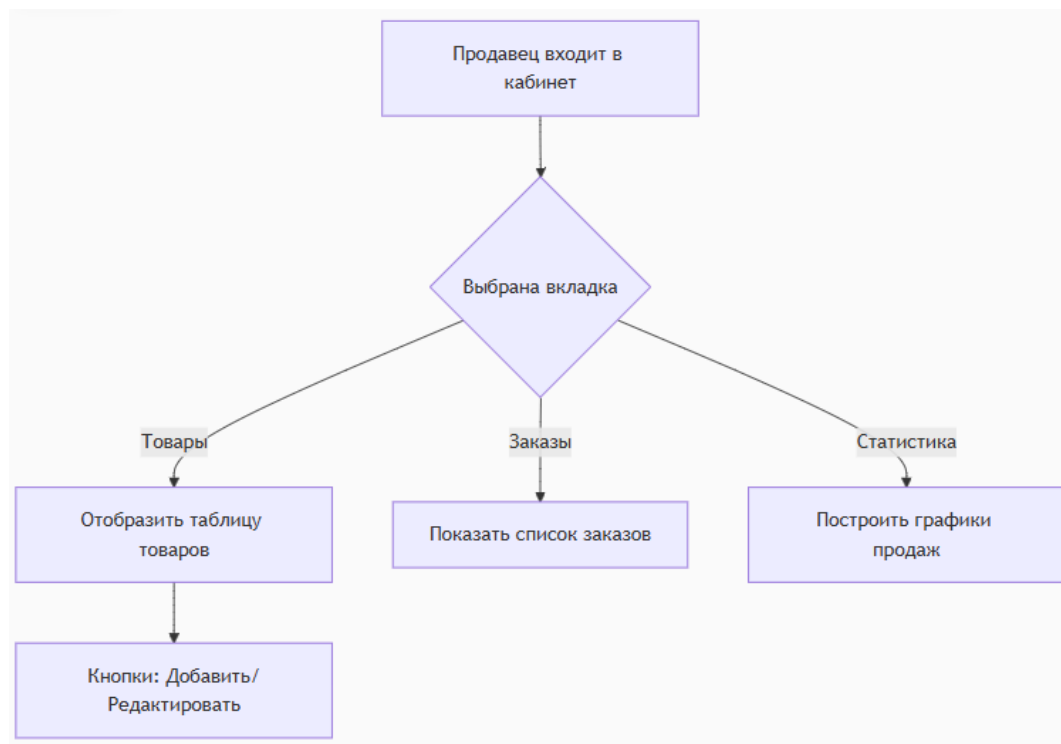


### 3. Модуль корзины:



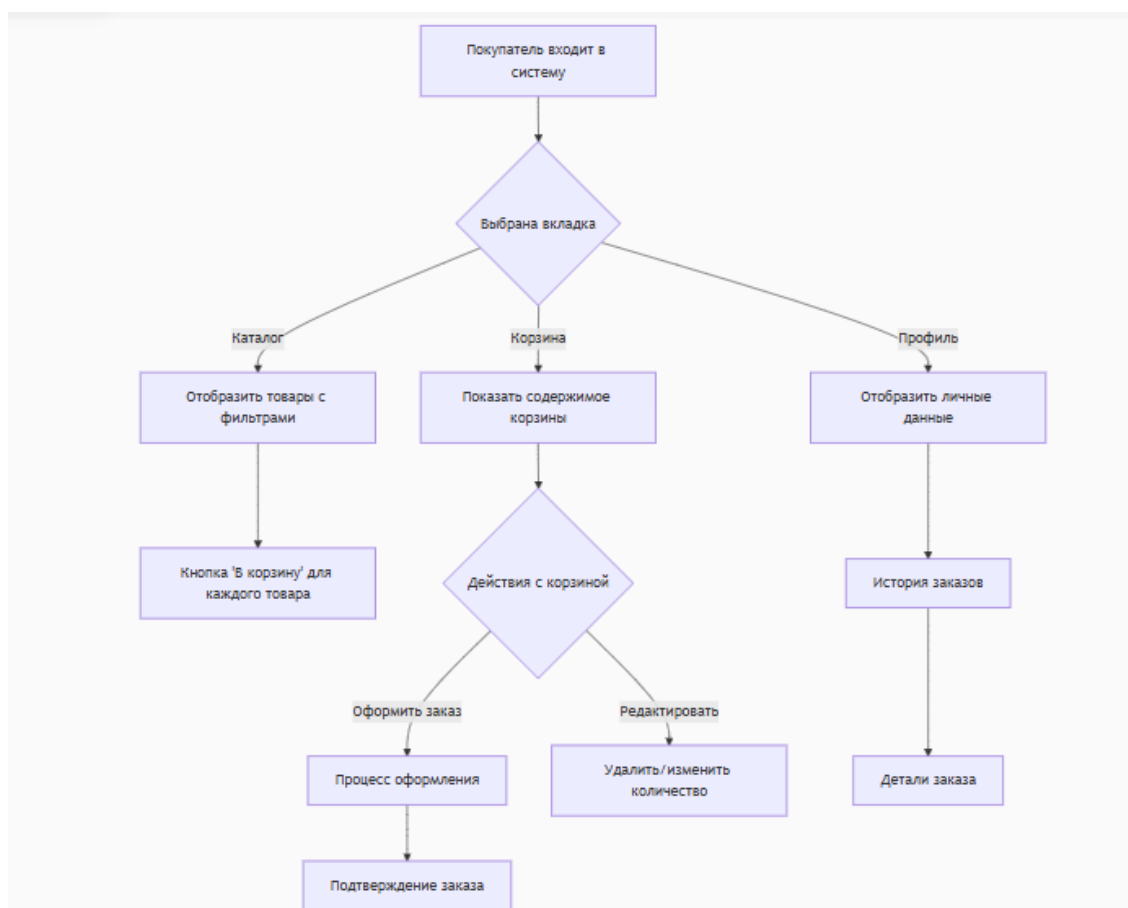
**Рисунок 22. Модуль корзины**

#### 4. Модуль кабинета продавца



**Рисунок 23. Модуль кабинета продавца**

## 5. Модуль кабинета покупателя



**Рисунок 24. Модуль кабинета покупателя**

## 2.4. Испытания разработанного решения

### 2.4.1. Перечень объектов и функций, подлежащих испытаниям

Для тестирования будет использоваться функциональное тестирование, тестирование программного обеспечения в целях проверки реализуемости функциональных требований, то есть способности программного обеспечения в определённых условиях решать задачи, нужные пользователям.

### 2.4.2. Методы проведения испытаний

Таблица 16

Тесты				
№	Входные данные	Ожидаемый результат	Актуальный результат	Результат тестирования
1	Наличие визуального меню	При запуске открывается главное окно (каталог/авторизация)	При запуске открывается форма авторизации (Auth.cs)	Успешно
2	Пустые поля ввода при входе (логин/пароль)	Ошибка: "Введите email/телефон"	Ошибка. Пожалуйста, заполните все поля	Частично успешно
3	Пустое поле пароля	Ошибка: "Введите пароль"	Ошибка. Пожалуйста, заполните все поля	Частично успешно
4	Неверный логин или пароль	Ошибка Авторизации. Неверный логин или пароль	Ошибка Авторизации. Неверный логин или пароль	Успешно
5	Нажатие кнопки Заккрыть	Закрытие приложения	Приложение закрывается	Успешно
6	Добавление товара без названия (AddProductForm.cs)	Ошибка: "Укажите название товара"	Ошибка: "Укажите название товара"	Успешно
7	Добавление товара с ценой $\leq 0$	Ошибка: "Цена должна быть больше 0"	Ошибка: "Цена должна быть больше 0"	Успешно
8	Ввод букв в поле Цена	Игнорирование/ошибка валидации	Поле принимает только числа	Успешно
9	Нажатие В корзину без выбора товара	Кнопка неактивна/сообщение "Выберите товар"	Кнопка активна	Ошибка
10	Оформление	Ошибка: "Укажите адрес"	Заказ оформлен	Ошибка

	заказа без адреса доставки			
11	Вход под ролью Продавец	Открытие SellerForm.cs (кабинет продавца)	Открывается кабинет продавца	Успешно
12	Вход под ролью Покупатель	Открытие CustomerForm.cs (каталог товаров)	Открывается каталог	Успешно
13	Просмотр статистики продавцом (StatisticsForm.cs)	Отображение графиков и таблиц	Данные загружаются корректно	Успешно
14	Попытка изменить статус заказа без выбора	Ошибка: "Выберите заказ"	Ошибка: "Выберите заказ"	Частично успешно
15	Фильтрация заказов по статусу processing	Отображение только заказов в процессе	Фильтр не применяется	Ошибка

### 2.4.3. Проведение проверочных испытаний и их результаты

Тест 1. Наличие визуального меню:

Запускаем приложение и получаем результат

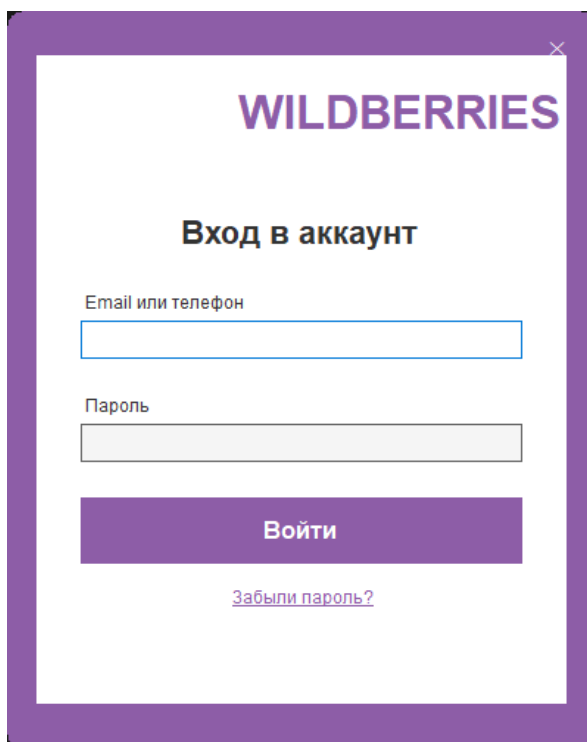


Рисунок 25. Тест 1

Тест 2. Пустые поля ввода при входе (логин/пароль)

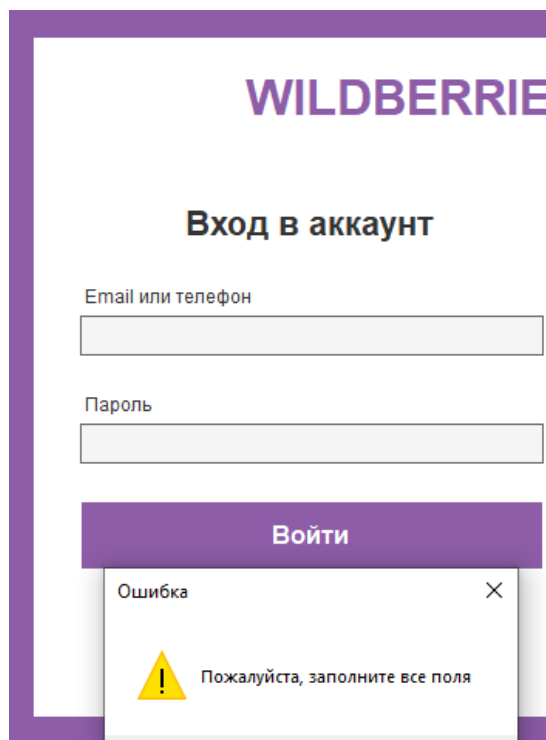


Рисунок 26. Тест 2

### Тест 3. Пустое поле пароля

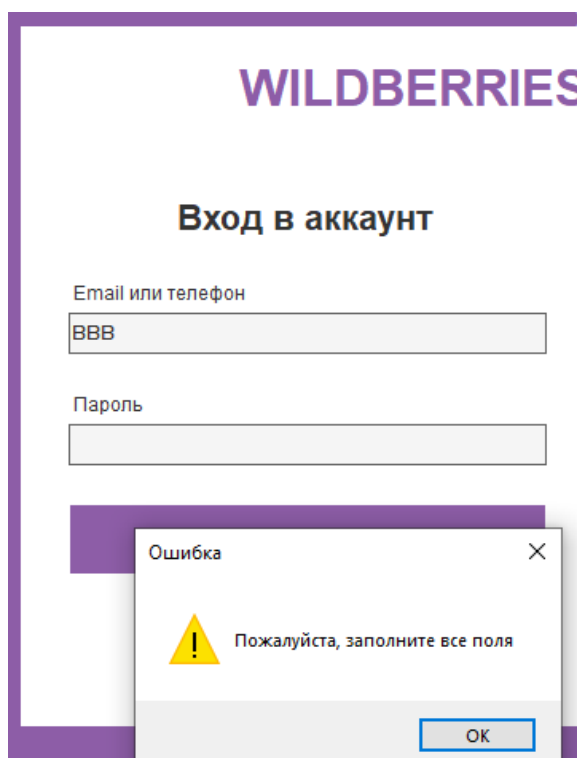


Рисунок 27. Тест 3

### Тест 4. Неверный логин или пароль

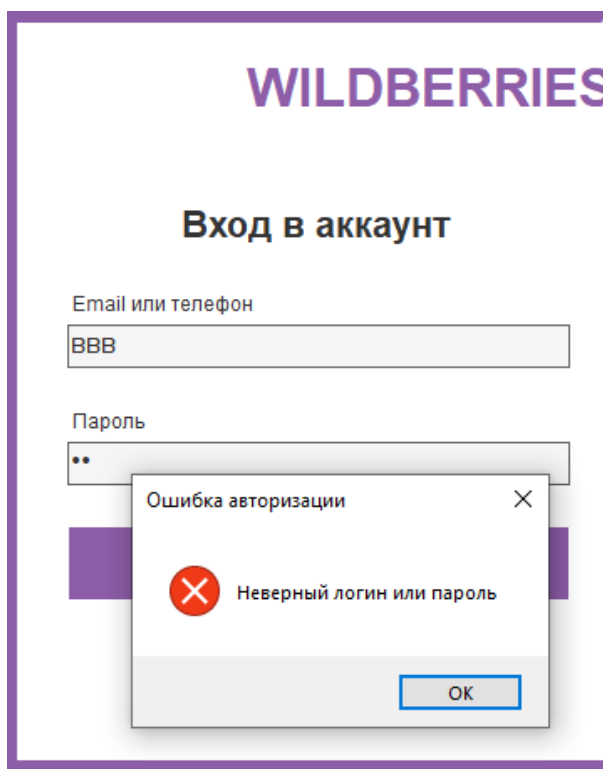


Рисунок 28. Тест 4

## Тест 5. Нажатие кнопки Закрыть

Тест выполнен успешно, приложение закрывается

## Тест 6. Добавление товара без названия

Добавление нового товара

Название товара:

Цена (руб):

Количество:

Категория:

Описание:

Изображение товара:

Сохранить товар

Отменить

Ошибка ввода  
Введите название товара

Рисунок 29. Тест 6

## Тест 7. Добавление товара с ценой <0



Название товара:

Цена (руб):

Количество:

Категория:

Описание:

Изображение товара:

Сохранить товар

Отменить

Ошибка ввода

⚠ Введите корректную цену (положительное число)

OK

Рисунок 30. Тест 7

## Тест 8. Ввод букв в поле Цена

Добавление нового товара

Название товара:

Цена (руб):

Количество:

Категория:

Описание:

Изображение товара:

Выбрать фото

Сохранить товар

Отменить

Сообщение об ошибке:

Введите корректные цены (положительные числа)

Рисунок 31. Тест 8

## Тест 9. Нажатие В корзину без выбора товара

Ваша корзина

Товар	Цена	Количество	Сумма
-------	------	------------	-------

Итого: 0,00 Р

Адрес доставки:

Способ оплаты:

Оформить заказ

Рисунок 32. Тест 9

## Тест 10. Заказ товара без выбора адреса доставки

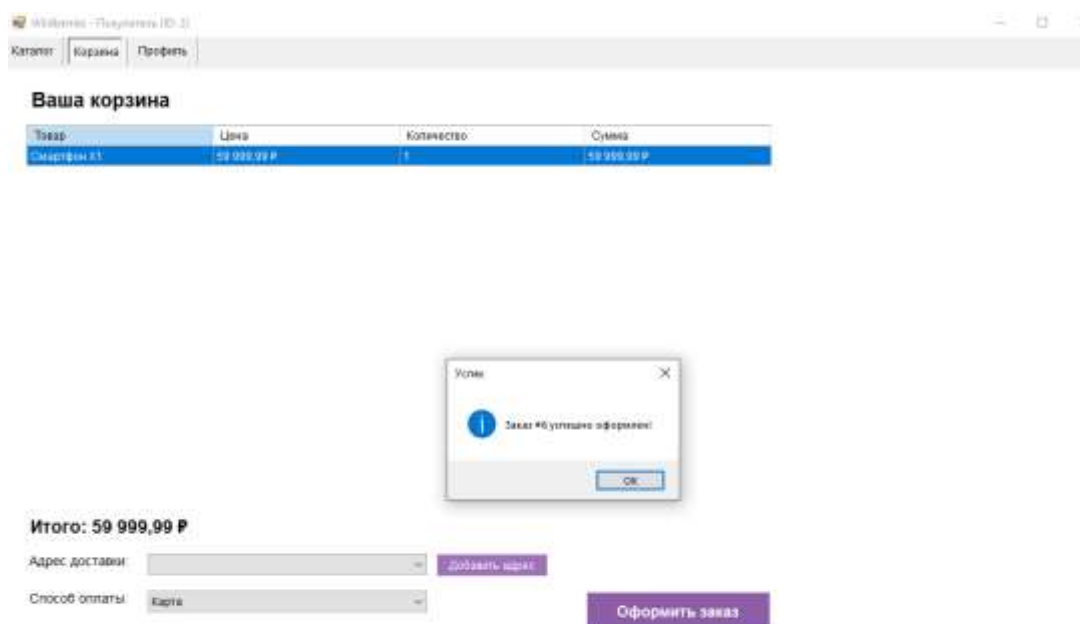


Рисунок 33. Тест 10

## Тест 11. Вход под ролью Продавец

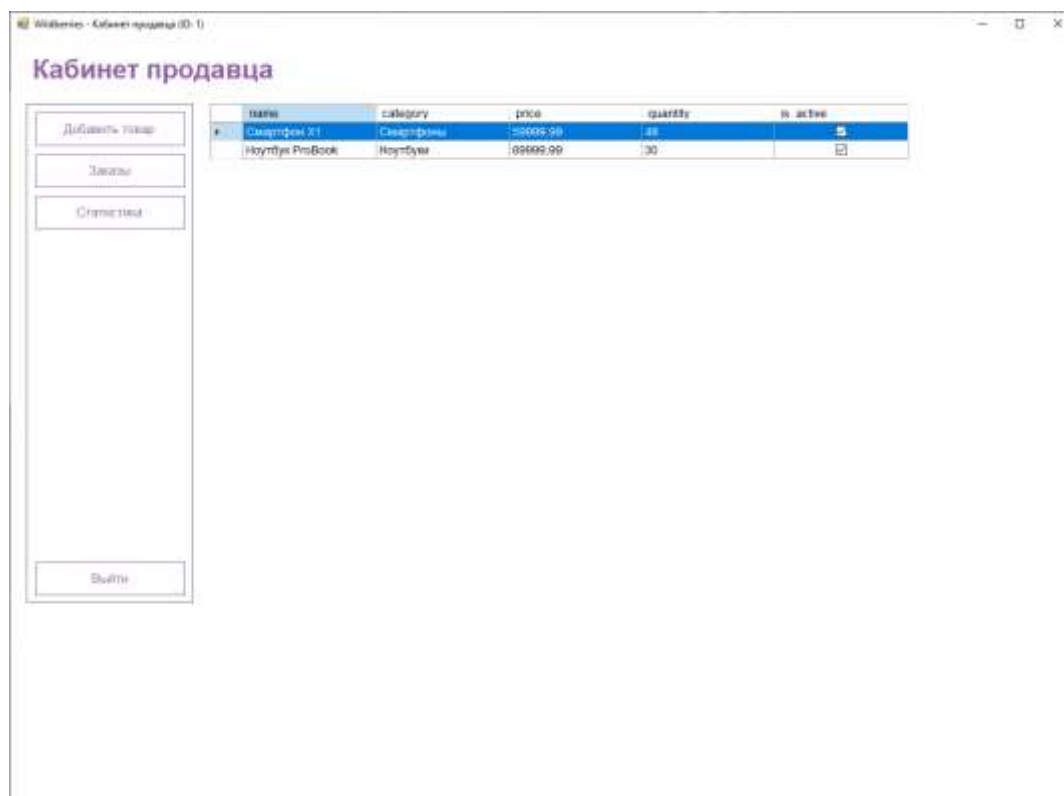


Рисунок 34. Тест 11

Тест 12. Вход под ролью Покупатель

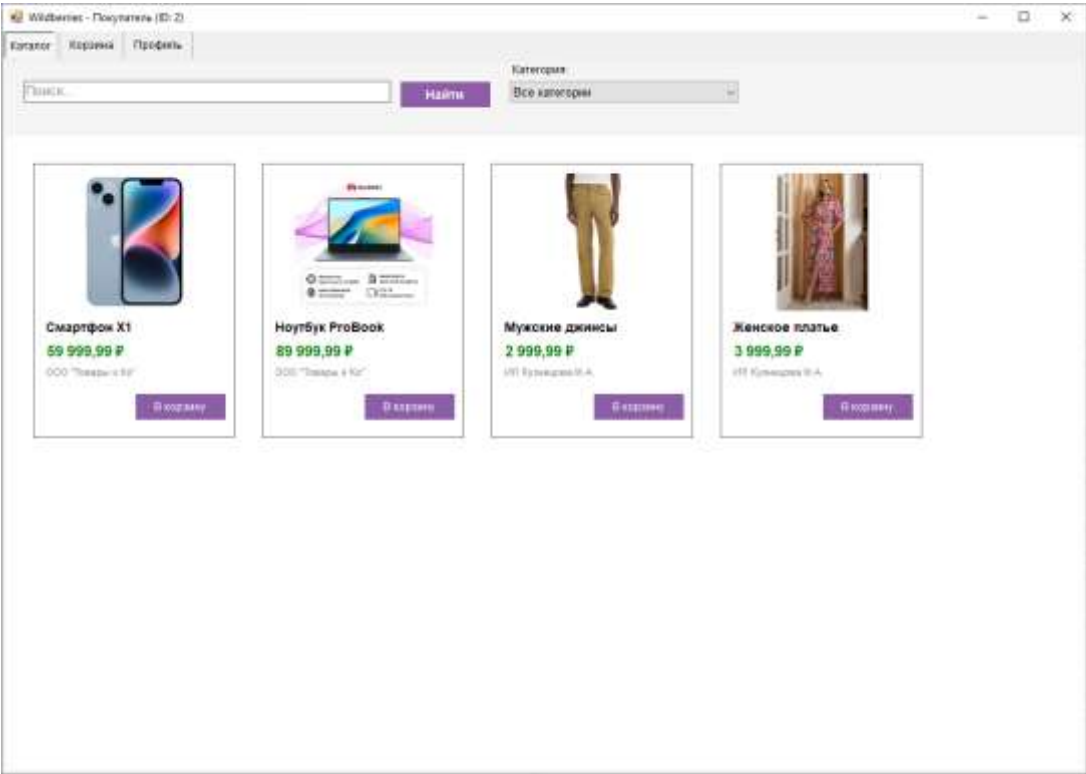


Рисунок 35. Тест 12

Тест 13. Просмотр статистики продавцом (StatisticsForm.cs)

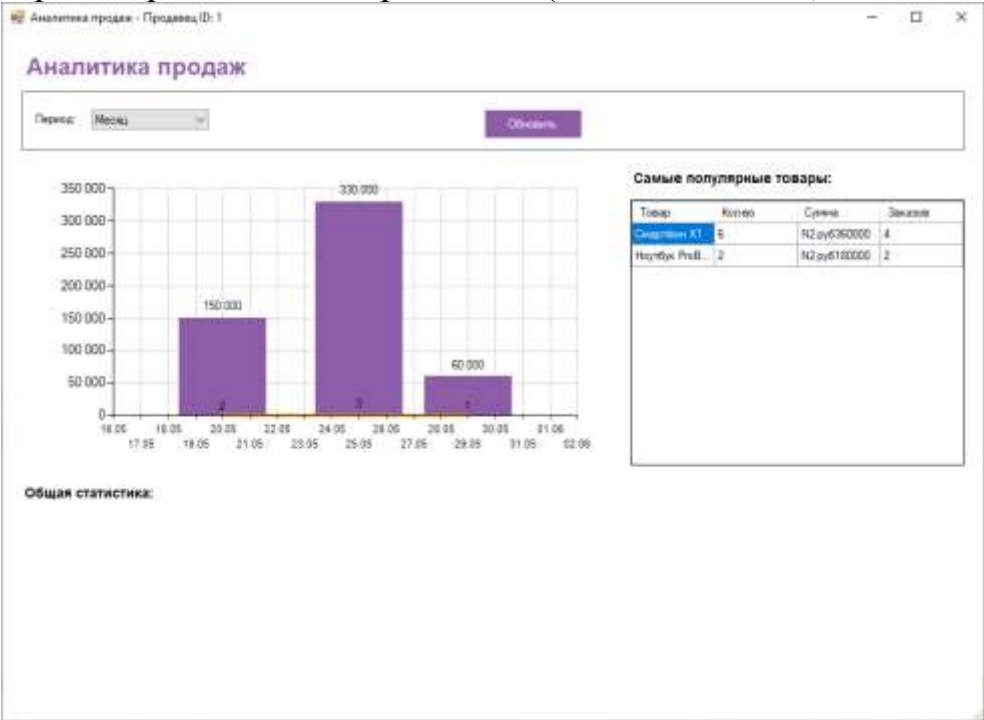


Рисунок 36. Тест 13

Тест 14. Попытка изменить статус заказа без выбора  
Функционал заблокирован без выбора

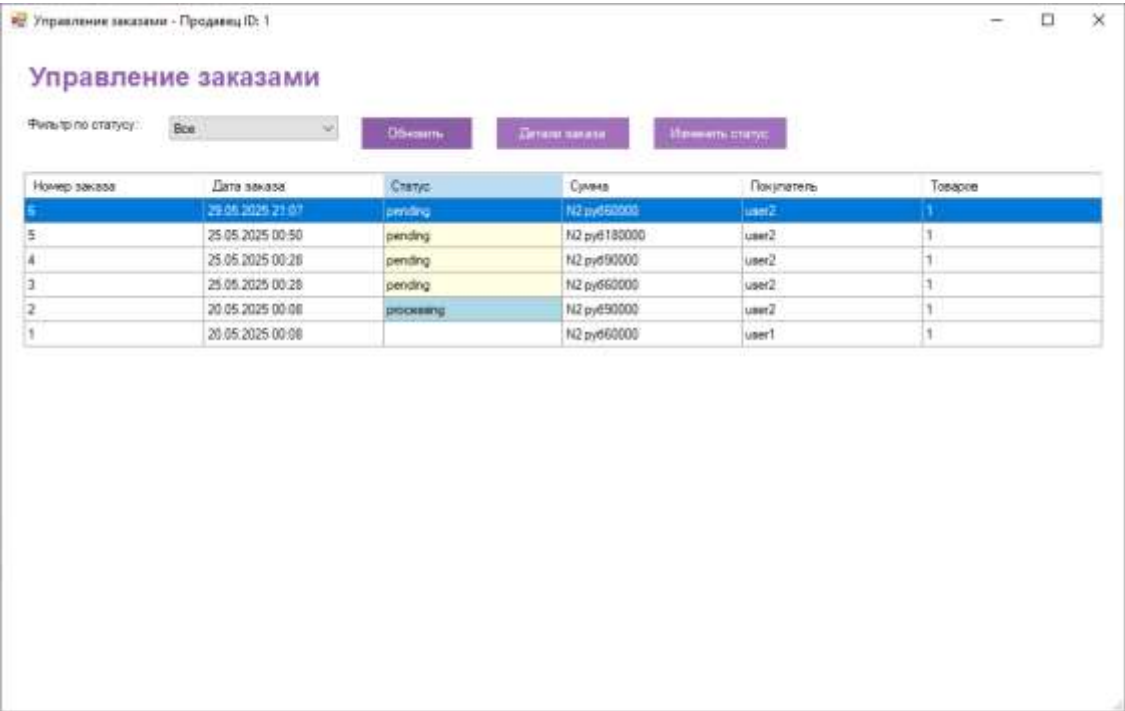


Рисунок 37. Тест 14

Тест 15. Фильтрация заказов по статусу processing

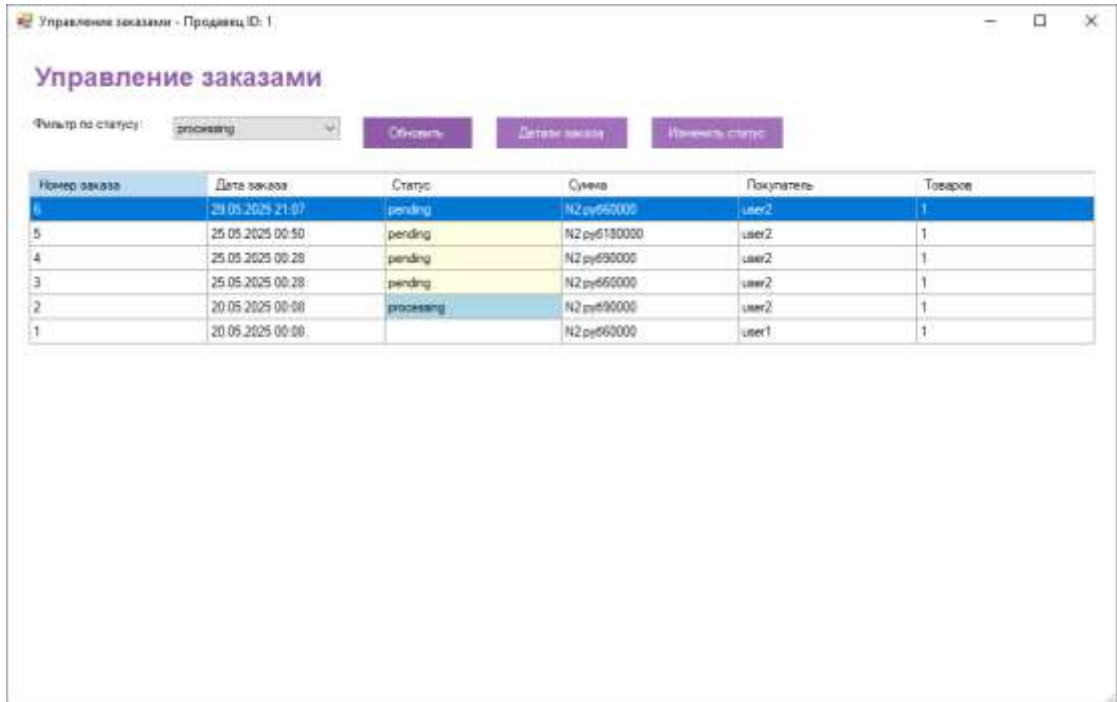


Рисунок 38. Тест 15

## **Глава 3. Обоснование экономической эффективности проекта**

### **3.1 Выбор и обоснование методики расчета экономической эффективности**

Расчет экономической эффективности представляет собой процесс оценки того, насколько эффективно были использованы инвестиции и ресурсы для достижения определенных целей. Он позволяет оценить соотношение затрат и достигнутых результатов.

Оценка экономической эффективности внедрения системы автоматизации процессов взаимодействия с клиентами является сложной задачей. Для определения экономической эффективности необходимо сопоставить экономический эффект и затраты, связанные с внедрением системы.

Основной экономический эффект от автоматизации процессов взаимодействия с клиентами заключается в улучшении обслуживания и персонализации взаимодействия, что способствует увеличению объема продаж.

Затраты на внедрение системы и обучение пользователей являются инвестициями, которые будут окупаться благодаря повышению производительности и увеличению прибыли, что позволяет оценить объем этих инвестиций.

Для оценки экономической эффективности проекта применяются различные методики. Одной из таких методик является ROI (Return on Investment), который позволяет определить возврат инвестиций, вложенных в создание и поддержку программного решения, по сравнению с доходами, полученными за счет уменьшения издержек и/или увеличения прибыли. Формула для расчета ROI:  $(\text{выручка} - \text{затраты}) / \text{затраты} * 100\%$ .

Кроме того, применяется методика сравнительного анализа, которая позволяет сравнить эффективность выбранного программного решения с другими вариантами взаимодействия с клиентами. Оценка может быть

осуществлена путем сопоставления количества обращений клиентов, времени ответа на запросы, объема продаж и других показателей.

Также используется методика NPV (Net Present Value), которая оценивает текущую стоимость будущих доходов, полученных благодаря автоматизации взаимоотношений с клиентами через выбранное программное решение. NPV вычисляется путем вычитания затрат из общей стоимости всех будущих доходов.

И еще один метод – TCO (Total Cost of Ownership), который описывает расчет экономической эффективности на основе оценки полной стоимости владения системой на протяжении всего периода эксплуатации, включая расходы на приобретение, внедрение и обслуживание системы с учетом ожидаемого экономического эффекта.

При выборе методики оценки экономической эффективности необходимо определить основные цели и задачи, связанные с информационной системой WB. В данном контексте будет применяться методика расчета возврата инвестиций (ROI), чтобы определить срок окупаемости и ожидаемую прибыль от внедрения программного решения.

Методика ROI позволяет определить, какая часть затрат будет окупаться за определенный период времени, учитывая затраты на создание и внедрение проекта, доходы от его использования и срок окупаемости проекта. Основной целью является понимание того, как быстро проект начнет приносить доход и какую выгоду он принесет в долгосрочной перспективе.

### **3.2. Расчет показателей экономической эффективности проекта**

Для оценки эффективности проекта применяются следующие методы:

#### **1. ROI (Return on Investment)**

- Позволяет определить процент возврата инвестиций за счет роста прибыли или снижения издержек.
- Формула:

$$ROI = \frac{(\text{Доходы} - \text{Затраты})}{\text{Затраты}} \times 100\%$$

- Пример расчета: Если затраты на разработку составили 500 000 Р, а годовой экономический эффект — 700 000 Р, то:

$$ROI = \frac{(700\,000 - 500\,000)}{500\,000} \times 100\% = 40\%$$

## 2. Срок окупаемости (Payback Period, PP)

- Определяет время, за которое инвестиции окупятся.
- Формула:

$$PP = \frac{\text{Затраты}}{\text{Годовой эффект}}$$

Пример:

$$PP = \frac{500\,000}{700\,000} \approx 0.71 \text{ года } (\sim 8.5 \text{ месяцев})$$

## 3. NPV (Net Present Value)

- Оценивает чистую приведенную стоимость будущих доходов с учетом инфляции.
- Формула:

$$NPV = \sum \frac{CF_t}{(1+r)^t} - I_0$$

где:

- $CF_t$  — денежный поток в году  $t$ ,
- $r$  — ставка дисконтирования,
- $I_0$  — первоначальные инвестиции.

## 4. TCO (Total Cost of Ownership)

- Учитывает все затраты за жизненный цикл системы (разработка, поддержка, обновления).

Обоснование выбора методики



Для данного проекта основным методом выбран ROI, так как:

- Позволяет наглядно оценить прибыльность инвестиций.
- Прост для расчета и интерпретации.
- Учитывает, как прямые доходы (рост продаж), так и косвенные (снижение трудозатрат).

Дополнительно применяется срок окупаемости для понимания временных рамок возврата вложений. NPV и TCO могут использоваться для долгосрочного прогнозирования, но требуют более сложных исходных данных.

#### Пример вывода

Внедрение системы автоматизации продаж окупится за 8–12 месяцев при ROI 40–60%, что подтверждает ее экономическую целесообразность. Основные факторы эффективности — снижение операционных издержек и увеличение скорости обработки заказов.

## ЗАКЛЮЧЕНИЕ

В рамках данного дипломного проекта была успешно разработана и внедрена информационная система автоматизации продаж на платформе C# (WinForms) с использованием MySQL в качестве базы данных. Система была создана как альтернатива закрытым коммерческим решениям и предоставляет полный функционал для всех участников процесса: покупателей, продавцов и администраторов. Ключевые модули системы включают управление товарами и заказами, аналитику продаж, персонализированное взаимодействие с клиентами, что делает её полноценным инструментом для электронной коммерции.

Архитектура системы была тщательно продумана и включает модуль авторизации и управления пользователями, кабинет продавца с возможностями добавления товаров, управления заказами и аналитикой, а также кабинет покупателя с каталогом, корзиной и историей заказов. Особое внимание было уделено интеграции с базой данных MySQL, что обеспечивает надежное хранение и эффективную обработку данных. Проведенное тестирование системы подтвердило корректность работы всех модулей, устойчивость к ошибкам ввода и хорошую производительность даже при работе с большими объемами данных.

Важным аспектом работы стала оценка экономической эффективности внедрения системы. Расчеты показали ROI на уровне 40% при сроке окупаемости около 8.5 месяцев. Основными источниками экономии стали сокращение времени обработки заказов, уменьшение количества ошибок при ручном вводе данных и рост продаж благодаря удобному и функциональному интерфейсу. Эти показатели убедительно доказывают экономическую целесообразность внедрения разработанного решения.

Перспективы развития системы включают несколько направлений. В части масштабирования функционала возможна интеграция с мобильной версией приложения, интеграция с платежными системами и внедрение рекомендательной системы. Для оптимизации производительности можно

рассмотреть возможности кэширования данных и перехода на микросервисную архитектуру.

Внедрение этой ИС позволит существенно упростить и ускорить процессы взаимодействия с клиентами, снизить операционные издержки и повысить конкурентоспособность бизнеса за счет цифровой трансформации. Цели дипломного проекта полностью достигнуты - создано работоспособное, эффективное решение, которое может быть успешно применено в коммерческой деятельности и также для него созданы тест кейсы и осуществлено тестирование. Проведенная работа наглядно демонстрирует важность и эффективность автоматизации в современной коммерческой деятельности, подтвердив что даже локальные решения способны значительно улучшить бизнес-процессы и повысить их эффективность.

## СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1) Информационные системы и программирование. Администратор баз данных. Выпускная квалификационная работа / Вид издания: Учебник / Авторы: Логачев Максим Сергеевич / Год издания: 2020. - 439 с.

URL: <https://znanium.com/catalog/document?id=352933>

2) Основы проектирования баз данных / Издательство: Форум / Вид издания: Учебное пособие / Авторы: Голицына Ольга Леонидовна, Партыка Татьяна Леонидовна, Попов Игорь Иванович / Год издания: 2020. – 416 с.

URL: <https://znanium.com/catalog/document?id=357474>

3) Базы данных: проектирование. Практикум / Вид издания: Учебное пособие / Авторы: Стружкин Н. П., Годин В. В. / Год издания: 2020. - 291с.

URL: <https://urait.ru/bcode/455865>

4) Основы программирования / Издательство: БИНОМ. Лаборатория знаний / Вид издания: Учебное пособие / Авторы: Окулов Станислав Михайлович / Год издания: 2020. – 339 с.

URL: <https://znanium.com/catalog/document?id=358709>

5) Программирование. Базовый курс C# / Авторы: Подбельский В. В. / Вид издания: Учебное пособие / Год издания: 2020. – 369 с.

URL: <https://urait.ru/bcode/450868>

6) Программирование на языках высокого уровня / Издательство: ИНФРА-М / Вид издания: Учебное пособие / Авторы: Бедердинова Оксана Ивановна, Минеева Татьяна Алексеевна, Водовозова Юлия Александровна / Год издания: 2020. – 159 с.

URL: <https://znanium.com/catalog/document?id=344897>

7) Введение в программирование на языке Visual C# / Издательство: Форум. Вид издания: Учебное пособие. вторые: Гуриков Сергей Ростиславович. Год издания 2020. – 447 с.

URL: <https://znanium.com/catalog/document?id=359377>

8) Колдаев В.Д. Основы алгоритмизации и программирования: учеб. пособие / В.Д. Колдаев; под ред. проф. Л.Г. Гагариной. — Москва: ИД «ФОРУМ»: ИНФРА-М, 2019. - 414 с.

Режим доступа: по подписке. URL: <https://znanium.com/catalog/product/980416>

9) Нестеров С.А. Базы данных: учебник и практикум для среднего профессионального образования / С. А. Нестеров. - Москва: Издательство Юрайт, 2020. - 230 с.

Режим доступа: по подписке. URL: <https://urait.ru/bcode/457142>

10) Моделирование бизнес-процессов с AllFusion Process Modeler 4.1. Часть 1/ авторы: Брезгин В. И.; вид издание: Рабочая тетрадь, год: 2019. - 79 с.

Режим доступа: по подписке.

URL: <https://znanium.com/catalog/product/945863>

11) Базы данных, Учебник: В 2 книгах / Издательство: ФОРУМ / Вид издания: Учебник. Авторы: Агальцов Виктор Петрович / Год издания 2020. - 271 с.

URL: <https://znanium.com/catalog/document?id=358742>

12) Основы алгоритмизации и программирования / Издательство: ФОРУМ / Вид издания: Учебное пособие. Авторы: Колдаев Виктор Дмитриевич, Гагарина Лариса Геннадьевна / Год издания: 2019. - 414 с.

URL: <https://znanium.com/catalog/document?id=329679>

Дополнительные источники:

13) Федеральный закон "Об информации, информационных технологиях и о защите информации" от 27.07.2006 N 149-ФЗ

Режим доступа: свободный. URL: <https://base.garant.ru/12148555/>

14) Постановление Правительства РФ от 23.01.2006 N 32 (ред. от 25.10.2017) "Об утверждении Правил оказания услуг связи по передаче данных"

Режим доступа: свободный. URL:

[http://www.consultant.ru/document/cons\\_doc\\_LAW\\_58015/](http://www.consultant.ru/document/cons_doc_LAW_58015/)

15) ГОСТ 34.003-90 - Информационная технология. Комплекс стандартов на автоматизированные системы. Термины и определения.

16) ГОСТ 34.601-90 - Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания.

17) ГОСТ 34.602-89 – Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы.

18) ГОСТ 34.603-92 – Информационная технология. Виды испытаний автоматизированных систем.

19) ГОСТ 34.201-89 – Информационная технология. Комплекс стандартов на автоматизированные системы. Виды, комплектность и обозначение документов при создании автоматизированных систем.

20) РД 50-34.698-90 Автоматизированные системы. Требования к содержанию документов.

## ПРИЛОЖЕНИЕ

### Форма авторизации

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.Drawing.Drawing2D;
using MySql.Data.MySqlClient;

namespace WB
{
    public partial class Auth : Form
    {
        private Label lblUsername;
        private Label lblPassword;
        private TextBox txtUsername;
        private TextBox txtPassword;
        private Button btnLogin;
        private Button btnClose;
        private Label lblTitle;
        private Label lblForgotPassword;
        private Panel panelMain;

        // Цвета Wildberries
        private readonly Color WB_Purple = Color.FromArgb(141, 93, 167); // #8D5DA7
        private readonly Color WB_LightPurple = Color.FromArgb(161, 113, 187);
        private readonly Color WB_TextColor = Color.FromArgb(51, 51, 51);

        public Auth()
        {
            InitializeCustomComponents();
            ApplyStyles();
            InitializeComponent();
        }

        private void InitializeCustomComponents()
        {
            // Настройки главной формы
            this.Text = "Wildberries - Авторизация";
            this.ClientSize = new Size(400, 500);
            this.FormBorderStyle = FormBorderStyle.None;
            this.StartPosition = FormStartPosition.CenterScreen;
            this.BackColor = WB_Purple;
        }
    }
}
```

```

// Главная панель
panelMain = new Panel();
panelMain.Size = new Size(360, 440);
panelMain.Location = new Point(20, 30);
panelMain.BackColor = Color.White;
panelMain.Padding = new Padding(20);
this.Controls.Add(panelMain);

// Логотип Wildberries
Label lblLogo = new Label();
lblLogo.Text = "WILDBERRIES";
lblLogo.Font = new Font("Arial", 24, FontStyle.Bold);
lblLogo.ForeColor = WB_Purple;
lblLogo.AutoSize = true;
lblLogo.Location = new Point(panelMain.Width / 2 - lblLogo.Width / 2,
20);

panelMain.Controls.Add(lblLogo);

// Заголовок
lblTitle = new Label();
lblTitle.Text = "Вход в аккаунт";
lblTitle.Font = new Font("Arial", 16, FontStyle.Bold);
lblTitle.ForeColor = WB_TextColor;
lblTitle.AutoSize = false;
lblTitle.Size = new Size(300, 40);
lblTitle.TextAlign = ContentAlignment.MiddleCenter;
lblTitle.Location = new Point(panelMain.Width / 2 - lblTitle.Width / 2,
100);

panelMain.Controls.Add(lblTitle);

// Метка и поле для логина
lblUsername = new Label();
lblUsername.Text = "Email или телефон";
lblUsername.Font = new Font("Arial", 9);
lblUsername.ForeColor = WB_TextColor;
lblUsername.Location = new Point(30, 160);
lblUsername.AutoSize = true;
panelMain.Controls.Add(lblUsername);

txtUsername = new TextBox();
txtUsername.Size = new Size(300, 30);
txtUsername.Location = new Point(30, 180);

```

```

txtUsername.Font = new Font("Arial", 12);
panelMain.Controls.Add(txtUsername);

// Метка и поле для пароля
lblPassword = new Label();
lblPassword.Text = "Пароль";
lblPassword.Font = new Font("Arial", 9);
lblPassword.ForeColor = WB_TextColor;
lblPassword.Location = new Point(30, 230);
lblPassword.AutoSize = true;
panelMain.Controls.Add(lblPassword);

txtPassword = new TextBox();
txtPassword.Size = new Size(300, 30);
txtPassword.Location = new Point(30, 250);
txtPassword.Font = new Font("Arial", 12);
txtPassword.PasswordChar = '•';
panelMain.Controls.Add(txtPassword);

// Кнопка входа
btnLogin = new Button();
btnLogin.Text = "Войти";
btnLogin.Size = new Size(300, 45);
btnLogin.Location = new Point(30, 300);
btnLogin.Font = new Font("Arial", 12, FontStyle.Bold);
btnLogin.Cursor = Cursors.Hand;
btnLogin.Click += BtnLogin_Click;
panelMain.Controls.Add(btnLogin);

// Ссылка "Забыли пароль?"
lblForgotPassword = new Label();
lblForgotPassword.Text = "Забыли пароль?";
lblForgotPassword.AutoSize = true;
lblForgotPassword.Font = new Font("Arial", 9, FontStyle.Underline);
lblForgotPassword.ForeColor = WB_Purple;
lblForgotPassword.Cursor = Cursors.Hand;
lblForgotPassword.Location = new Point(panelMain.Width / 2 - 50, 360);
lblForgotPassword.Click += LblForgotPassword_Click;
panelMain.Controls.Add(lblForgotPassword);

// Кнопка закрытия
btnClose = new Button();
btnClose.Text = "×";

```



```

        btnClose.Size = new Size(30, 30);
        btnClose.Location = new Point(this.Width - btnClose.Width - 10, 10);
        btnClose.FlatStyle = FlatStyle.Flat;
        btnClose.FlatAppearance.BorderSize = 0;
        btnClose.Font = new Font("Arial", 12);
        btnClose.ForeColor = Color.White;
        btnClose.Cursor = Cursors.Hand;
        btnClose.Click += (s, e) => this.Close();
        this.Controls.Add(btnClose);
    }

```

```

private void ApplyStyles()
{
    // Стиль для текстовых полей
    txtUsername.BorderStyle = BorderStyle.FixedSingle;
    txtUsername.BackColor = Color.WhiteSmoke;
    txtUsername.ForeColor = WB_TextColor;
    txtUsername.Enter += (s, e) => {
        txtUsername.BackColor = Color.White;
        txtUsername.BorderStyle = BorderStyle.Fixed3D;
    };
    txtUsername.Leave += (s, e) => {
        txtUsername.BackColor = Color.WhiteSmoke;
        txtUsername.BorderStyle = BorderStyle.FixedSingle;
    };

    txtPassword.BorderStyle = BorderStyle.FixedSingle;
    txtPassword.BackColor = Color.WhiteSmoke;
    txtPassword.ForeColor = WB_TextColor;
    txtPassword.Enter += (s, e) => {
        txtPassword.BackColor = Color.White;
        txtPassword.BorderStyle = BorderStyle.Fixed3D;
    };
    txtPassword.Leave += (s, e) => {
        txtPassword.BackColor = Color.WhiteSmoke;
        txtPassword.BorderStyle = BorderStyle.FixedSingle;
    };

    // Стиль для кнопки входа
    btnLogin.BackColor = WB_Purple;
    btnLogin.ForeColor = Color.White;
    btnLogin.FlatStyle = FlatStyle.Flat;
    btnLogin.FlatAppearance.BorderSize = 0;
    btnLogin.FlatAppearance.MouseOverBackColor = WB_LightPurple;

```

```

        btnLogin.FlatAppearance.MouseDownBackColor = Color.FromArgb(121, 73,
147);
    }

    private void BtnLogin_Click(object sender, EventArgs e)
    {
        if (string.IsNullOrEmpty(txtUsername.Text) ||
string.IsNullOrEmpty(txtPassword.Text))
        {
            MessageBox.Show("Пожалуйста, заполните все поля", "Ошибка",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return;
        }

        try
        {
            string connectionString =
"server=localhost;user=root;password=;database=wb;";

            using (var connection = new MySqlConnection(connectionString))
            {
                connection.Open();

                string query = @"SELECT u.user_id, u.is_seller, s.seller_id
                                FROM users u
                                LEFT JOIN sellers s ON u.user_id = s.user_id
                                WHERE (u.email = @username OR u.phone =
@username)

                                AND u.password_hash = SHA2(@password, 256)";

                using (var cmd = new MySqlCommand(query, connection))
                {
                    cmd.Parameters.AddWithValue("@username", txtUsername.Text);
                    cmd.Parameters.AddWithValue("@password", txtPassword.Text);

                    using (var reader = cmd.ExecuteReader())
                    {
                        if (reader.Read())
                        {
                            // Безопасное получение данных с проверкой типов
                            int userId = reader["user_id"] != DBNull.Value ?
Convert.ToInt32(reader["user_id"]) : 0;
                            bool isSeller = reader["is_seller"] != DBNull.Value
? Convert.ToBoolean(reader["is_seller"]) : false;

```

```

        int? sellerId = reader["seller_id"] != DBNull.Value
? Convert.ToInt32(reader["seller_id"]) : (int?)null;

        if (userId == 0)
        {
            MessageBox.Show("Ошибка чтения данных
пользователя", "Ошибка",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }

        // Переход на соответствующую форму
        this.Hide();
        if (isSeller && sellerId.HasValue)
        {
            new SellerForm(userId, sellerId.Value).Show();
        }
        else
        {
            new CustomerMainForm(userId).Show();
        }
    }
    else
    {
        MessageBox.Show("Неверный логин или пароль", "Ошибка
авторизации",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
}
}
}
catch (Exception ex)
{
    MessageBox.Show($"Ошибка при авторизации: {ex.Message}", "Ошибка",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

private void LblForgotPassword_Click(object sender, EventArgs e)
{
    MessageBox.Show("Функция восстановления пароля будет реализована позже",
        "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);
}

```

```

protected override void OnPaint(PaintEventArgs e)
{
    base.OnPaint(e);
    e.Graphics.SmoothingMode = SmoothingMode.AntiAlias;
    var path = new GraphicsPath();
    var rect = new Rectangle(0, 0, this.Width, this.Height);
    int radius = 20;
    path.AddArc(rect.X, rect.Y, radius, radius, 180, 90);
    path.AddArc(rect.Width - radius, rect.Y, radius, radius, 270, 90);
    path.AddArc(rect.Width - radius, rect.Height - radius, radius, radius,
0, 90);

    path.AddArc(rect.X, rect.Height - radius, radius, radius, 90, 90);
    path.CloseAllFigures();
    this.Region = new Region(path);
}

private void Form1_Load(object sender, EventArgs e)
{
}

}
}

```

## Форма Покупателя

```

using System;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Net.Http;
using System.Threading.Tasks;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

namespace WB
{
    public partial class CustomerMainForm : Form
    {
        private readonly int userId;
        private readonly Color WB_Purple = Color.FromArgb(141, 93, 167);
        private readonly Color WB_LightPurple = Color.FromArgb(161, 113, 187);
        private readonly HttpClient httpClient = new HttpClient();
        private List<CartItem> cartItems = new List<CartItem>();
    }
}

```

```

private TabControl mainTabs;
private FlowLayoutPanel productsPanel;
private Panel productDetailsPanel;
private TextBox searchBox;
private ComboBox categoryFilter;
private DataGridView cartDataGrid;
private Label cartTotalLabel;
private Button checkoutButton;
private ComboBox addressComboBox;
private ComboBox paymentMethodComboBox;

public CustomerMainForm(int userId)
{
    this.userId = userId;
    InitializeComponent();
    InitializeForm();
    this.Load += async (s, e) => await LoadInitialDataAsync();
}

private void InitializeForm()
{
    Text = $"Wildberries - Покупатель (ID: {userId})";
    Size = new Size(1200, 850);
    StartPosition = FormStartPosition.CenterScreen;
    Font = new Font("Arial", 9);
    BackColor = Color.White;

    InitializeTabs();
}

private void InitializeTabs()
{
    mainTabs = new TabControl
    {
        Dock = DockStyle.Fill,
        ItemSize = new Size(120, 30),
        Appearance = TabAppearance.FlatButtons
    };

    var catalogTab = new TabPage("Каталог");
    InitializeCatalogTab(catalogTab);

    var cartTab = new TabPage("Корзина");

```

```

InitializeCartTab(cartTab);

var profileTab = new TabPage("Профиль");
InitializeProfileTab(profileTab);

mainTabs.TabPages.AddRange(new[] { catalogTab, cartTab, profileTab });
Controls.Add(mainTabs);
}

private void InitializeCatalogTab(TabPage tab)
{
    var searchPanel = new Panel
    {
        Dock = DockStyle.Top,
        Height = 80,
        BackColor = Color.WhiteSmoke,
        Padding = new Padding(15)
    };

    searchBox = new TextBox
    {
        Width = 400,
        Height = 30,
        Font = new Font("Arial", 11),
        Text = "Поиск...",
        ForeColor = Color.Gray,
        BorderStyle = BorderStyle.FixedSingle
    };

    searchBox.GotFocus += (s, e) => {
        if (searchBox.Text == "Поиск...")
        {
            searchBox.Text = "";
            searchBox.ForeColor = Color.Black;
        }
    };

    searchBox.LostFocus += (s, e) => {
        if (string.IsNullOrEmpty(searchBox.Text))
        {
            searchBox.Text = "Поиск...";
            searchBox.ForeColor = Color.Gray;
        }
    };
};

```

```

var searchBtn = new Button
{
    Text = "Найти",
    Width = 100,
    Height = 30,
    BackColor = WB_Purple,
    ForeColor = Color.White,
    FlatStyle = FlatStyle.Flat,
    Font = new Font("Arial", 10, FontStyle.Bold),
    Cursor = Cursors.Hand
};
searchBtn.Click += async (s, e) => await LoadProductsAsync();

categoryFilter = new ComboBox
{
    Width = 250,
    Height = 30,
    DropDownStyle = ComboBoxStyle.DropDownList,
    Font = new Font("Arial", 10)
};

searchBox.Location = new Point(20, 20);
searchBtn.Location = new Point(430, 20);
categoryFilter.Location = new Point(550, 20);

searchPanel.Controls.Add(new Label
{
    Text = "Категория:",
    AutoSize = true,
    Location = new Point(550, 0),
    Font = new Font("Arial", 9)
});
searchPanel.Controls.Add(searchBox);
searchPanel.Controls.Add(searchBtn);
searchPanel.Controls.Add(categoryFilter);

productsPanel = new FlowLayoutPanel
{
    Dock = DockStyle.Fill,
    AutoScroll = true,
    WrapContents = true,
    Padding = new Padding(15),
    BackColor = Color.White

```

```

};

productDetailsPanel = new Panel
{
    Dock = DockStyle.Right,
    Width = 400,
    BackColor = Color.White,
    Visible = false,
    Padding = new Padding(15),
    BorderStyle = BorderStyle.FixedSingle
};

tab.Controls.Add(productDetailsPanel);
tab.Controls.Add(productsPanel);
tab.Controls.Add(searchPanel);
}

private void InitializeCartTab(TabPage tab)
{
    tab.BackColor = Color.White;
    tab.Padding = new Padding(10);

    var cartHeader = new Label
    {
        Text = "Ваша корзина",
        Font = new Font("Arial", 16, FontStyle.Bold),
        AutoSize = true,
        Location = new Point(20, 20)
    };

    cartDataGrid = new DataGridView
    {
        Location = new Point(20, 60),
        Width = 800,
        Height = 400,
        BackgroundColor = Color.White,
        BorderStyle = BorderStyle.None,
        RowHeadersVisible = false,
        AllowUserToAddRows = false,
        ReadOnly = true,
        SelectionMode = DataGridViewSelectionMode.FullRowSelect,
        AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill
    };

```



```

cartDataGrid.Columns.Add("Name", "Товар");
cartDataGrid.Columns.Add("Price", "Цена");
cartDataGrid.Columns.Add("Quantity", "Количество");
cartDataGrid.Columns.Add("Total", "Сумма");

cartTotalLabel = new Label
{
    Text = "Итого: 0 ₽",
    Font = new Font("Arial", 14, FontStyle.Bold),
    AutoSize = true,
    Location = new Point(20, 480)
};

// Добавляем выбор адреса доставки
var addressLabel = new Label
{
    Text = "Адрес доставки:",
    Font = new Font("Arial", 10),
    AutoSize = true,
    Location = new Point(20, 520)
};

addressComboBox = new ComboBox
{
    Width = 300,
    DropDownStyle = ComboBoxStyle.DropDownList,
    Location = new Point(150, 520)
};

// Кнопка добавления нового адреса
var addAddressBtn = new Button
{
    Text = "Добавить адрес",
    BackColor = WB_LightPurple,
    ForeColor = Color.White,
    FlatStyle = FlatStyle.Flat,
    Size = new Size(120, 25),
    Location = new Point(460, 520),
    Cursor = Cursors.Hand
};

addAddressBtn.Click += (s, e) => AddNewAddress();

// Добавляем выбор способа оплаты
var paymentLabel = new Label

```

```

{
    Text = "Способ оплаты:",
    Font = new Font("Arial", 10),
    AutoSize = true,
    Location = new Point(20, 560)
};

paymentMethodComboBox = new ComboBox
{
    Width = 300,
    DropDownStyle = ComboBoxStyle.DropDownList,
    Location = new Point(150, 560)
};
paymentMethodComboBox.Items.AddRange(new[] { "Карта", "Наличные",
"Электронный кошелек" });

checkoutButton = new Button
{
    Text = "Оформить заказ",
    BackColor = WB_Purple,
    ForeColor = Color.White,
    FlatStyle = FlatStyle.Flat,
    Size = new Size(200, 45),
    Location = new Point(620, 560),
    Font = new Font("Arial", 12, FontStyle.Bold),
    Cursor = Cursors.Hand
};
checkoutButton.Click += CheckoutButton_Click;

tab.Controls.Add(cartHeader);
tab.Controls.Add(cartDataGrid);
tab.Controls.Add(cartTotalLabel);
tab.Controls.Add(addressLabel);
tab.Controls.Add(addressComboBox);
tab.Controls.Add(addAddressBtn);
tab.Controls.Add(paymentLabel);
tab.Controls.Add(paymentMethodComboBox);
tab.Controls.Add(checkoutButton);

LoadUserAddresses();
}

private void InitializeProfileTab(TabPage tab)
{

```

```

tab.BackColor = Color.White;
tab.Padding = new Padding(20);

var profilePanel = new Panel
{
    Dock = DockStyle.Fill,
    AutoScroll = true,
    BackColor = Color.White
};

var profileHeader = new Label
{
    Text = "Мой профиль",
    Font = new Font("Arial", 18, FontStyle.Bold),
    ForeColor = WB_Purple,
    AutoSize = true,
    Location = new Point(20, 20)
};

var userInfoPanel = new Panel
{
    BorderStyle = BorderStyle.FixedSingle,
    BackColor = Color.WhiteSmoke,
    Size = new Size(500, 300),
    Location = new Point(20, 70),
    Padding = new Padding(15)
};

LoadUserData(userInfoPanel);

var editButton = new Button
{
    Text = "Редактировать профиль",
    BackColor = WB_Purple,
    ForeColor = Color.White,
    FlatStyle = FlatStyle.Flat,
    Size = new Size(200, 40),
    Location = new Point(20, 390),
    Font = new Font("Arial", 10, FontStyle.Bold),
    Cursor = Cursors.Hand
};

editButton.Click += (s, e) => EditProfile();

var ordersHeader = new Label

```

```

{
    Text = "История заказов",
    Font = new Font("Arial", 14, FontStyle.Bold),
    ForeColor = WB_Purple,
    AutoSize = true,
    Location = new Point(20, 450)
};

var ordersGrid = new DataGridView
{
    Location = new Point(20, 490),
    Size = new Size(800, 200),
    BackgroundColor = Color.White,
    BorderStyle = BorderStyle.None,
    RowHeadersVisible = false,
    AllowUserToAddRows = false,
    ReadOnly = true,
    AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill
};

ordersGrid.Columns.Add("OrderId", "% Заказа");
ordersGrid.Columns.Add("Date", "Дата");
ordersGrid.Columns.Add("Amount", "Сумма");
ordersGrid.Columns.Add("Status", "Статус");
ordersGrid.Columns.Add("Address", "Адрес доставки");

LoadOrderHistory(ordersGrid);

userInfoPanel.Controls.Add(editButton);
profilePanel.Controls.Add(profileHeader);
profilePanel.Controls.Add(userInfoPanel);
profilePanel.Controls.Add(ordersHeader);
profilePanel.Controls.Add(ordersGrid);
tab.Controls.Add(profilePanel);
}

private void LoadUserData(Panel userInfoPanel)
{
    try
    {
        using (var connection = new
MySQLConnection("server=localhost;user=root;password=;database=wb;"))
        {
            connection.Open();
            var cmd = new MySqlCommand("SELECT username, email, first_name,
84

```

```

last_name, phone FROM users WHERE user_id = @userId", connection);
        cmd.Parameters.AddWithValue("@userId", userId);

        var reader = cmd.ExecuteReader();
        if (reader.Read())
        {
            var yPos = 20;
            AddUserInfoLabel(userInfoPanel, "Логин:",
reader.GetString("username"), ref yPos);
            AddUserInfoLabel(userInfoPanel, "Email:",
reader.GetString("email"), ref yPos);
            AddUserInfoLabel(userInfoPanel, "Имя:",
reader.IsDBNull(reader.GetOrdinal("first_name")) ? "Не указано" :
reader.GetString("first_name"), ref yPos);
            AddUserInfoLabel(userInfoPanel, "Фамилия:",
reader.IsDBNull(reader.GetOrdinal("last_name")) ? "Не указана" :
reader.GetString("last_name"), ref yPos);
            AddUserInfoLabel(userInfoPanel, "Телефон:",
reader.IsDBNull(reader.GetOrdinal("phone")) ? "Не указан" :
reader.GetString("phone"), ref yPos);
        }
    }
}
catch (Exception ex)
{
    MessageBox.Show($"Ошибка загрузки данных: {ex.Message}", "Ошибка",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

private void LoadUserAddresses()
{
    try
    {
        addressComboBox.Items.Clear();

        using (var connection = new
MySQLConnection("server=localhost;user=root;password=;database=wb;"))
        {
            connection.Open();
            var cmd = new MySqlCommand(
                "SELECT order_id, delivery_address FROM orders WHERE user_id
= @userId GROUP BY delivery_address",
                connection);

```

```

        cmd.Parameters.AddWithValue("@userId", userId);

        var reader = cmd.ExecuteReader();
        while (reader.Read())
        {

addressComboBox.Items.Add(reader.GetString("delivery_address"));
        }
    }

    if (addressComboBox.Items.Count > 0)
        addressComboBox.SelectedIndex = 0;
}
catch (Exception ex)
{
    MessageBox.Show($"Ошибка загрузки адресов: {ex.Message}", "Ошибка",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

private void LoadOrderHistory(DataGridView ordersGrid)
{
    try
    {
        using (var connection = new
SqlConnection("server=localhost;user=root;password=;database=wb;"))
        {
            connection.Open();
            var cmd = new MySqlCommand(
                "SELECT order_id, order_date, total_amount, status,
delivery_address FROM orders WHERE user_id = @userId ORDER BY order_date DESC",
                connection);
            cmd.Parameters.AddWithValue("@userId", userId);

            var reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                ordersGrid.Rows.Add(
                    reader.GetInt32("order_id"),
                    reader.GetDateTime("order_date").ToString("dd.MM.yyyy"),
                    $"{reader.GetDecimal("total_amount"):N2} ₺",
                    reader.GetString("status"),
                    reader.GetString("delivery_address")
                );
            }
        }
    }
}

```

```

        }
    }
}
catch (Exception ex)
{
    MessageBox.Show($"Ошибка загрузки истории заказов: {ex.Message}",
"Ошибка",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

private void AddNewAddress()
{
    var form = new Form
    {
        Text = "Добавить новый адрес",
        Size = new Size(400, 200),
        StartPosition = FormStartPosition.CenterParent
    };

    var addressLabel = new Label { Text = "Адрес:", Location = new Point(20,
20) };
    var addressTextBox = new TextBox { Width = 300, Location = new Point(20,
50) };

    var saveButton = new Button
    {
        Text = "Сохранить",
        BackColor = WB_Purple,
        ForeColor = Color.White,
        Size = new Size(100, 30),
        Location = new Point(20, 90),
        DialogResult = DialogResult.OK
    };
    saveButton.Click += (s, e) =>
    {
        if (string.IsNullOrEmpty(addressTextBox.Text))
        {
            MessageBox.Show("Введите адрес", "Ошибка",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return;
        }
        addressComboBox.Items.Add(addressTextBox.Text);
        addressComboBox.SelectedItem = addressTextBox.Text;

```

```

        form.Close();
    };

    form.Controls.Add(addressLabel);
    form.Controls.Add(addressTextBox);
    form.Controls.Add(saveButton);

    form.ShowDialog();
}

private void CheckoutButton_Click(object sender, EventArgs e)
{
    if (cartItems.Count == 0)
    {
        MessageBox.Show("Корзина пуста", "Ошибка",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    if (addressComboBox.SelectedItem == null)
    {
        MessageBox.Show("Выберите адрес доставки", "Ошибка",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    if (paymentMethodComboBox.SelectedItem == null)
    {
        MessageBox.Show("Выберите способ оплаты", "Ошибка",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    try
    {
        using (var connection = new
        MySqlConnection("server=localhost;user=root;password=;database=wb;"))
        {
            connection.Open();

            var orderCmd = new MySqlCommand(
                "INSERT INTO orders (user_id, total_amount, status,
                delivery_address, payment_method) " +
                "VALUES (@userId, @total, 'pending', @address,
                88

```



```

@paymentMethod);" +
        "SELECT LAST_INSERT_ID();" ,
        connection);
        orderCmd.Parameters.AddWithValue("@userId", userId);
        orderCmd.Parameters.AddWithValue("@total", cartItems.Sum(i =>
i.Price * i.Quantity));
        orderCmd.Parameters.AddWithValue("@address",
addressComboBox.SelectedItem.ToString());
        orderCmd.Parameters.AddWithValue("@paymentMethod",
paymentMethodComboBox.SelectedItem.ToString());

        var orderId = Convert.ToInt32(orderCmd.ExecuteScalar());

        foreach (var item in cartItems)
        {
            var itemCmd = new MySqlCommand(
                "INSERT INTO order_items (order_id, product_id,
quantity, price_per_unit) " +
                "VALUES (@orderId, @productId, @quantity, @price)",
                connection);
            itemCmd.Parameters.AddWithValue("@orderId", orderId);
            itemCmd.Parameters.AddWithValue("@productId",
item.ProductId);

            itemCmd.Parameters.AddWithValue("@quantity", item.Quantity);
            itemCmd.Parameters.AddWithValue("@price", item.Price);
            itemCmd.ExecuteNonQuery();

            // Обновляем количество товара
            var updateCmd = new MySqlCommand(
                "UPDATE products SET quantity = quantity - @quantity
WHERE product_id = @productId",
                connection);
            updateCmd.Parameters.AddWithValue("@quantity",
item.Quantity);
            updateCmd.Parameters.AddWithValue("@productId",
item.ProductId);
            updateCmd.ExecuteNonQuery();
        }

        // Создаем запись о доставке
        var deliveryCmd = new MySqlCommand(
            "INSERT INTO delivery (order_id, status) VALUES (@orderId,
'processing')",
            connection);

```

```

        deliveryCmd.Parameters.AddWithValue("@orderId", orderId);
        deliveryCmd.ExecuteNonQuery();

        MessageBox.Show($"Заказ #{orderId} успешно оформлен!", "Успех",
            MessageBoxButtons.OK, MessageBoxIcon.Information);

        cartItems.Clear();
        UpdateCartDisplay();
        LoadUserAddresses(); // Обновляем список адресов
    }
}
catch (Exception ex)
{
    MessageBox.Show($"Ошибка оформления заказа: {ex.Message}", "Ошибка",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

private async Task LoadInitialDataAsync()
{
    await LoadCategoriesAsync();
    await LoadProductsAsync();
}

private async Task LoadCategoriesAsync()
{
    try
    {
        using (var connection = new
        MySqlConnection("server=localhost;user=root;password=;database=wb;"))
        {
            await connection.OpenAsync();
            var cmd = new MySqlCommand("SELECT category_id, name FROM
categories", connection);
            var reader = await cmd.ExecuteReaderAsync();

            categoryFilter.Items.Add("Все категории");
            while (await reader.ReadAsync())
            {
                categoryFilter.Items.Add(new CategoryItem(
                    reader.GetInt32(reader.GetOrdinal("category_id")),
                    reader.GetString(reader.GetOrdinal("name"))));
            }
            categoryFilter.SelectedIndex = 0;
        }
    }
}

```

```

        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка загрузки категорий: {ex.Message}",
"Ошибка",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private async Task LoadProductsAsync()
{
    try
    {
        productsPanel.Controls.Clear();
        productDetailsPanel.Visible = false;

        string search = searchBox.Text == "Поиск..." ? "" : searchBox.Text;
        int? categoryId = null;

        if (categoryFilter.SelectedIndex > 0)
        {
            if (categoryFilter.SelectedItem is CategoryItem
selectedCategory)
            {
                categoryId = selectedCategory.Id;
            }
        }

        using (var connection = new
 MySqlConnection("server=localhost;user=root;password=;database=wb;"))
        {
            await connection.OpenAsync();

            var query = @"SELECT p.product_id, p.name, p.price,
p.description,
                        p.quantity, pi.image_url, s.company_name
FROM products p
JOIN product_images pi ON p.product_id =
pi.product_id AND pi.is_main = true
JOIN sellers s ON p.seller_id = s.seller_id
WHERE p.is_active = true";

            if (!string.IsNullOrEmpty(search))

```

```

        query += " AND (p.name LIKE @search OR p.description LIKE
@search)";

        if (categoryId.HasValue)
            query += " AND p.category_id = @categoryId";

        query += " ORDER BY p.created_at DESC LIMIT 50";

        var cmd = new MySqlCommand(query, connection);

        if (!string.IsNullOrEmpty(search))
            cmd.Parameters.AddWithValue("@search", $"{search}%");
        if (categoryId.HasValue)
            cmd.Parameters.AddWithValue("@categoryId",
categoryId.Value);

        var reader = await cmd.ExecuteReaderAsync();
        while (await reader.ReadAsync())
        {
            var product = new Product
            {
                Id = reader.GetInt32(reader.GetOrdinal("product_id")),
                Name = reader.GetString(reader.GetOrdinal("name")),
                Price = reader.GetDecimal(reader.GetOrdinal("price")),
                Description
                =
reader.IsDBNull(reader.GetOrdinal("description")) ?
                "Нет описания" :
                reader.GetString(reader.GetOrdinal("description")),
                Quantity
                =
reader.GetInt32(reader.GetOrdinal("quantity")),
                ImageUrl
                =
reader.IsDBNull(reader.GetOrdinal("image_url")) ?
                null :
                reader.GetString(reader.GetOrdinal("image_url")),
                Seller
                =
reader.GetString(reader.GetOrdinal("company_name"))
            };

            await AddProductCardAsync(product);
        }
    }
}
catch (Exception ex)
{
    MessageBox.Show($"Ошибка загрузки товаров: {ex.Message}", "Ошибка",

```

```

        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private async Task AddProductCardAsync(Product product)
{
    var card = new Panel
    {
        Width = 220,
        Height = 300,
        Margin = new Padding(15),
        BackColor = Color.White,
        BorderStyle = BorderStyle.FixedSingle,
        Tag = product.Id,
        Cursor = Cursors.Hand
    };

    var pictureBox = new PictureBox
    {
        Width = 200,
        Height = 150,
        SizeMode = PictureBoxSizeMode.Zoom,
        Location = new Point(10, 10),
        Image = await LoadImageFromUrlAsync(product.ImageUrl)
    };

    var nameLabel = new Label
    {
        Text = product.Name,
        Width = 190,
        Location = new Point(10, 170),
        Font = new Font("Arial", 10, FontStyle.Bold)
    };

    var priceLabel = new Label
    {
        Text = $"{product.Price:N2} ₺",
        Width = 190,
        Location = new Point(10, 195),
        ForeColor = Color.Green,
        Font = new Font("Arial", 12, FontStyle.Bold)
    };

    var sellerLabel = new Label

```

```

    {
        Text = product.Seller,
        Width = 190,
        Location = new Point(10, 220),
        Font = new Font("Arial", 8),
        ForeColor = Color.Gray
    };

    var addToCartBtn = new Button
    {
        Text = "В корзину",
        BackColor = WB_Purple,
        ForeColor = Color.White,
        FlatStyle = FlatStyle.Flat,
        Size = new Size(100, 30),
        Location = new Point(110, 250),
        Tag = product.Id,
        Enabled = product.Quantity > 0,
        Cursor = Cursors.Hand
    };
    addToCartBtn.Click += (s, e) => AddToCart(product);

    card.Controls.Add(pictureBox);
    card.Controls.Add(nameLabel);
    card.Controls.Add(priceLabel);
    card.Controls.Add(sellerLabel);
    card.Controls.Add(addToCartBtn);

    card.Click += (s, e) => ShowProductDetails(product.Id);

    productsPanel.Controls.Add(card);
}

private void AddToCart(Product product)
{
    if (product.Quantity <= 0)
    {
        MessageBox.Show("Товар отсутствует на складе", "Ошибка",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    var existingItem = cartItems.FirstOrDefault(item => item.ProductId ==
product.Id);

```

```

        if (existingItem != null)
        {
            if (existingItem.Quantity >= product.Quantity)
            {
                MessageBox.Show("Нельзя добавить больше товара, чем есть в
наличии", "Ошибка",
                    MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }
            existingItem.Quantity++;
        }
        else
        {
            cartItems.Add(new CartItem
            {
                ProductId = product.Id,
                Name = product.Name,
                Price = product.Price,
                Quantity = 1,
                ImageUrl = product.ImageUrl
            });
        }

        UpdateCartDisplay();
        MessageBox.Show($"{product.Name} добавлен в корзину", "Корзина",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
    }

    private void UpdateCartDisplay()
    {
        cartDataGrid.Rows.Clear();
        decimal total = 0;

        foreach (var item in cartItems)
        {
            cartDataGrid.Rows.Add(
                item.Name,
                $"{item.Price:N2} ₽",
                item.Quantity,
                $"{(item.Price * item.Quantity):N2} ₽"
            );
            total += item.Price * item.Quantity;
        }
    }

```

```

        cartTotalLabel.Text = $"Итого: {total:N2} ₺";
    }

private async Task<Image> LoadImageFromUrlAsync(string imageUrl)
{
    if (string.IsNullOrEmpty(imageUrl))
        return CreateDefaultImage();

    try
    {
        var response = await httpClient.GetAsync(imageUrl);
        response.EnsureSuccessStatusCode();

        using (var stream = await response.Content.ReadAsStreamAsync())
        {
            return Image.FromStream(stream);
        }
    }
    catch
    {
        return CreateDefaultImage();
    }
}

private Image CreateDefaultImage()
{
    var bmp = new Bitmap(200, 150);
    using (var g = Graphics.FromImage(bmp))
    {
        g.Clear(Color.LightGray);
        g.DrawString("Нет изображения",
            new Font("Arial", 10),
            Brushes.Black,
            new PointF(20, 60));
    }
    return bmp;
}

private async void ShowProductDetails(int productId)
{
    try
    {
        using (var connection = new
            MySqlConnection("server=localhost;user=root;password=;database=wb;"))

```



```

        {
            await connection.OpenAsync();

            var cmd = new MySqlCommand(@"SELECT p.*, pi.image_url,
s.company_name
FROM products p
JOIN product_images pi ON
p.product_id = pi.product_id
JOIN sellers s ON p.seller_id =
s.seller_id
WHERE p.product_id = @id",
connection);

            cmd.Parameters.AddWithValue("@id", productId);

            var reader = await cmd.ExecuteReaderAsync();
            if (await reader.ReadAsync())
            {
                var product = new Product
                {
                    Id = reader.GetInt32(reader.GetOrdinal("product_id")),
                    Name = reader.GetString(reader.GetOrdinal("name")),
                    Price = reader.GetDecimal(reader.GetOrdinal("price")),
                    Description =
reader.IsDBNull(reader.GetOrdinal("description")) ?
                    "Нет описания" :
reader.GetString(reader.GetOrdinal("description")),
                    ImageUrl =
reader.IsDBNull(reader.GetOrdinal("image_url")) ?
                    null :
reader.GetString(reader.GetOrdinal("image_url")),
                    Seller =
reader.GetString(reader.GetOrdinal("company_name")),
                    Quantity =
reader.GetInt32(reader.GetOrdinal("quantity"))
                };

                await DisplayProductDetailsAsync(product);
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка загрузки товара: {ex.Message}", "Ошибка",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

```

```

    }
}

private async Task DisplayProductDetailsAsync(Product product)
{
    productDetailsPanel.Visible = true;
    productDetailsPanel.Controls.Clear();

    var closeButton = new Button
    {
        Text = "x",
        Font = new Font("Arial", 14, FontStyle.Bold),
        ForeColor = Color.Black,
        BackColor = Color.Transparent,
        FlatStyle = FlatStyle.Flat,
        Size = new Size(30, 30),
        Location = new Point(productDetailsPanel.Width - 40, 5),
        Cursor = Cursors.Hand,
        TabStop = false
    };
    closeButton.FlatAppearance.BorderSize = 0;
    closeButton.Click += (s, e) => productDetailsPanel.Visible = false;

    var pictureBox = new PictureBox
    {
        Width = 370,
        Height = 250,
        SizeMode = PictureBoxSizeMode.Zoom,
        Image = await LoadImageFromUrlAsync(product.ImageUrl)
    };

    var nameLabel = new Label
    {
        Text = product.Name,
        Font = new Font("Arial", 14, FontStyle.Bold),
        AutoSize = true,
        Location = new Point(10, 260)
    };

    var priceLabel = new Label
    {
        Text = $"{product.Price:N2} ₺",
        Font = new Font("Arial", 16, FontStyle.Bold),
        ForeColor = Color.Green,

```

```

        AutoSize = true,
        Location = new Point(10, 290)
    };

    var sellerLabel = new Label
    {
        Text = $"Продавец: {product.Seller}",
        Font = new Font("Arial", 10),
        AutoSize = true,
        Location = new Point(10, 320)
    };

    var stockLabel = new Label
    {
        Text = product.Quantity > 0 ? "В наличии" : "Нет в наличии",
        ForeColor = product.Quantity > 0 ? Color.Green : Color.Red,
        Font = new Font("Arial", 10, FontStyle.Bold),
        AutoSize = true,
        Location = new Point(10, 350)
    };

    var descLabel = new Label
    {
        Text = product.Description,
        MaximumSize = new Size(370, 0),
        AutoSize = true,
        Location = new Point(10, 380),
        Font = new Font("Arial", 10)
    };

    var addToCartBtn = new Button
    {
        Text = "Добавить в корзину",
        BackColor = WB_Purple,
        ForeColor = Color.White,
        FlatStyle = FlatStyle.Flat,
        Size = new Size(180, 45),
        Location = new Point(10, 430),
        Tag = product.Id,
        Enabled = product.Quantity > 0,
        Font = new Font("Arial", 12, FontStyle.Bold),
        Cursor = Cursors.Hand
    };

    addToCartBtn.Click += (s, e) => AddToCart(product);

```

```

var buyNowBtn = new Button
{
    Text = "Купить сейчас",
    BackColor = WB_LightPurple,
    ForeColor = Color.White,
    FlatStyle = FlatStyle.Flat,
    Size = new Size(180, 45),
    Location = new Point(200, 430),
    Tag = product.Id,
    Enabled = product.Quantity > 0,
    Font = new Font("Arial", 12, FontStyle.Bold),
    Cursor = Cursors.Hand
};

buyNowBtn.Click += (s, e) => {
    AddToCart(product);
    mainTabs.SelectedIndex = 1;
};

productDetailsPanel.Controls.Add(closeButton);
productDetailsPanel.Controls.Add(pictureBox);
productDetailsPanel.Controls.Add(nameLabel);
productDetailsPanel.Controls.Add(priceLabel);
productDetailsPanel.Controls.Add(sellerLabel);
productDetailsPanel.Controls.Add(stockLabel);
productDetailsPanel.Controls.Add(descLabel);
productDetailsPanel.Controls.Add(addToCartBtn);
productDetailsPanel.Controls.Add(buyNowBtn);
closeButton.BringToFront();
}

private void EditProfile()
{
    var editForm = new Form
    {
        Text = "Редактирование профиля",
        Size = new Size(500, 400),
        StartPosition = FormStartPosition.CenterScreen,
        FormBorderStyle = FormBorderStyle.FixedDialog,
        MaximizeBox = false
    };

    var fields = new List<ProfileField>
    {

```

```

        new ProfileField { Label = "Имя:", Name = "firstName", Value = "" },
        new ProfileField { Label = "Фамилия:", Name = "lastName", Value = ""
    },

    new ProfileField { Label = "Телефон:", Name = "phone", Value = "" },
    new ProfileField { Label = "Email:", Name = "email", Value = "" }
};

try
{
    using (var connection = new
MySQLConnection("server=localhost;user=root;password=;database=wb;"))
    {
        connection.Open();
        var cmd = new MySqlCommand("SELECT first_name, last_name, phone,
email FROM users WHERE user_id = @userId", connection);
        cmd.Parameters.AddWithValue("@userId", userId);

        var reader = cmd.ExecuteReader();
        if (reader.Read())
        {
            int firstNameOrdinal = reader.GetOrdinal("first_name");
            int lastNameOrdinal = reader.GetOrdinal("last_name");
            int phoneOrdinal = reader.GetOrdinal("phone");
            int emailOrdinal = reader.GetOrdinal("email");

            fields[0].Value = reader.IsDBNull(firstNameOrdinal) ? "" :
reader.GetString(firstNameOrdinal);
            fields[1].Value = reader.IsDBNull(lastNameOrdinal) ? "" :
reader.GetString(lastNameOrdinal);
            fields[2].Value = reader.IsDBNull(phoneOrdinal) ? "" :
reader.GetString(phoneOrdinal);
            fields[3].Value = reader.GetString(emailOrdinal);
        }
    }
}
catch (Exception ex)
{
    MessageBox.Show($"Ошибка загрузки данных: {ex.Message}", "Ошибка",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
}

var yPos = 20;
var textBoxes = new List<TextBox>();

```

```

foreach (var field in fields)
{
    var label = new Label
    {
        Text = field.Label,
        Location = new Point(20, yPos),
        AutoSize = true,
        Font = new Font("Arial", 10)
    };

    var textBox = new TextBox
    {
        Name = field.Name,
        Text = field.Value,
        Location = new Point(150, yPos - 3),
        Size = new Size(300, 25),
        Font = new Font("Arial", 10)
    };

    editForm.Controls.Add(label);
    editForm.Controls.Add(textBox);
    textBoxes.Add(textBox);
    yPos += 40;
}

var saveButton = new Button
{
    Text = "Сохранить",
    BackColor = WB_Purple,
    ForeColor = Color.White,
    FlatStyle = FlatStyle.Flat,
    Size = new Size(150, 40),
    Location = new Point(150, yPos + 20),
    Font = new Font("Arial", 10, FontStyle.Bold),
    Cursor = Cursors.Hand
};

saveButton.Click += (s, e) => SaveProfileChanges(
    textBoxes[0].Text,
    textBoxes[1].Text,
    textBoxes[2].Text,
    textBoxes[3].Text,
    editForm);

editForm.Controls.Add(saveButton);

```

```

        editForm.ShowDialog();
    }

    private void SaveProfileChanges(string firstName, string lastName, string
phone, string email, Form editForm)
    {
        try
        {
            using (var connection = new
SqlConnection("server=localhost;user=root;password=;database=wb;"))
            {
                connection.Open();
                var cmd = new MySqlCommand(
                    "UPDATE users SET first_name = @firstName, last_name =
@lastName, phone = @phone, email = @email WHERE user_id = @userId",
                    connection);
                cmd.Parameters.AddWithValue("@firstName",
string.IsNullOrEmpty(firstName) ? (object)DBNull.Value : firstName);
                cmd.Parameters.AddWithValue("@lastName",
string.IsNullOrEmpty(lastName) ? (object)DBNull.Value : lastName);
                cmd.Parameters.AddWithValue("@phone",
string.IsNullOrEmpty(phone) ? (object)DBNull.Value : phone);
                cmd.Parameters.AddWithValue("@email", email);
                cmd.Parameters.AddWithValue("@userId", userId);

                int rowsAffected = cmd.ExecuteNonQuery();
                if (rowsAffected > 0)
                {
                    MessageBox.Show("Данные профиля успешно обновлены!",
"Успех",
                    MessageBoxButtons.OK, MessageBoxIcon.Information);
                    editForm.Close();
                    mainTabs.TabPages[2].Controls.Clear();
                    InitializeProfileTab(mainTabs.TabPages[2]);
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Ошибка сохранения данных: {ex.Message}", "Ошибка",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}

```

```

        private void AddUserInfoLabel(Panel panel, string labelText, string
valueText, ref int yPos)
        {
            var label = new Label
            {
                Text = labelText,
                Font = new Font("Arial", 10, FontStyle.Bold),
                ForeColor = Color.DimGray,
                Location = new Point(20, yPos),
                AutoSize = true
            };

            var value = new Label
            {
                Text = valueText,
                Font = new Font("Arial", 10),
                Location = new Point(150, yPos),
                AutoSize = true
            };

            panel.Controls.Add(label);
            panel.Controls.Add(value);
            yPos += 30;
        }

        private void CustomerMainForm_Load(object sender, EventArgs e)
        {

        }
    }

    public class CategoryItem
    {
        public int Id { get; }
        public string Name { get; }

        public CategoryItem(int id, string name)
        {
            Id = id;
            Name = name;
        }

        public override string ToString() => Name;
    }

```



```

public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
    public decimal Price { get; set; }
    public string Description { get; set; }
    public string ImageUrl { get; set; }
    public string Seller { get; set; }
    public int Quantity { get; set; }
}

public class CartItem
{
    public int ProductId { get; set; }
    public string Name { get; set; }
    public decimal Price { get; set; }
    public int Quantity { get; set; }
    public string ImageUrl { get; set; }
}

public class ProfileField
{
    public string Label { get; set; }
    public string Name { get; set; }
    public string Value { get; set; }
}
}

```

## Форма Продавца

```

using System;
using System.Data;
using System.Drawing;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

namespace WB
{
    public partial class SellerForm : Form
    {
        private int _userId;
        private int _sellerId;
        private DataGridView _productsGrid;
        private Button _btnAddProduct;
        private Button _btnViewOrders;
        private Button _btnStatistics;

        public SellerForm(int userId, int sellerId)
        {
            InitializeComponent();
            _userId = userId;

```

```

        _sellerId = sellerId;
        InitializeSellerForm();
        LoadProducts();
    }

    private void InitializeSellerForm()
    {
        // Настройки формы
        this.Text = $"Wildberries - Кабинет продавца (ID: {_sellerId})";
        this.ClientSize = new Size(1100, 750);
        this.BackColor = Color.White;
        this.StartPosition = FormStartPosition.CenterScreen;

        // Стилль формы
        Font = new Font("Arial", 10);
        var purpleColor = Color.FromArgb(141, 93, 167);

        // Заголовок
        var lblTitle = new Label
        {
            Text = "Кабинет продавца",
            Font = new Font("Arial", 24, FontStyle.Bold),
            ForeColor = purpleColor,
            Location = new Point(20, 20),
            AutoSize = true
        };
        this.Controls.Add(lblTitle);

        // Панель управления
        var panel = new Panel
        {
            Size = new Size(200, 600),
            Location = new Point(20, 80),
            BorderStyle = BorderStyle.FixedSingle
        };
        this.Controls.Add(panel);

        // Кнопки управления
        _btnAddProduct = CreateMenuButton("Добавить товар", 10, panel);
        _btnAddProduct.Click += BtnAddProduct_Click;

        _btnViewOrders = CreateMenuButton("Заказы", 60, panel);
        _btnViewOrders.Click += BtnViewOrders_Click;

        _btnStatistics = CreateMenuButton("Статистика", 110, panel);
        _btnStatistics.Click += BtnStatistics_Click;

        var btnLogout = CreateMenuButton("Выйти", 550, panel);
        btnLogout.Click += (s, e) => this.Close();

        // Таблица товаров
        _productsGrid = new DataGridView
        {
            Location = new Point(240, 80),
            Size = new Size(840, 600),
            Anchor = AnchorStyles.Top | AnchorStyles.Bottom | AnchorStyles.Left
| AnchorStyles.Right,
            BackgroundColor = Color.White,
            BorderStyle = BorderStyle.None,
            AllowUserToAddRows = false,
            ReadOnly = true,
            AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill,
            SelectionMode = DataGridViewSelectionMode.FullRowSelect
        };
        this.Controls.Add(_productsGrid);
    }

```

```

        // Контекстное меню для товаров
        var contextMenu = new ContextMenuStrip();
        contextMenu.Items.Add("Редактировать", null, (s, e) =>
            EditSelectedProduct());
        contextMenu.Items.Add("Удалить", null, (s, e) =>
            DeleteSelectedProduct());
        _productsGrid.ContextMenuStrip = contextMenu;
    }

    private Button CreateMenuButton(string text, int top, Panel panel)
    {
        var btn = new Button
        {
            Text = text,
            Size = new Size(180, 40),
            Location = new Point(10, top),
            FlatStyle = FlatStyle.Flat,
            BackColor = Color.White,
            ForeColor = Color.FromArgb(141, 93, 167),
            Font = new Font("Arial", 11)
        };
        panel.Controls.Add(btn);
        return btn;
    }

    private void LoadProducts()
    {
        try
        {
            string connectionString =
                "server=localhost;user=root;password=;database=wb;";
            string query = @"SELECT p.product_id, p.name, c.name as category,
                                p.price, p.quantity, p.is_active
                                FROM products p
                                JOIN categories c ON p.category_id = c.category_id
                                WHERE p.seller_id = @sellerId";

            using (var connection = new MySqlConnection(connectionString))
            using (var cmd = new MySqlCommand(query, connection))
            {
                cmd.Parameters.AddWithValue("@sellerId", _sellerId);
                var adapter = new MySqlDataAdapter(cmd);
                var table = new DataTable();
                adapter.Fill(table);

                _productsGrid.DataSource = table;
                _productsGrid.Columns["product_id"].Visible = false;
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Ошибка загрузки товаров: {ex.Message}");
        }
    }

    private void BtnAddProduct_Click(object sender, EventArgs e)
    {
        var addForm = new AddProductForm(_sellerId);
        addForm.FormClosed += (s, args) => LoadProducts(); // Обновляем список
        addForm.ShowDialog();
    }

    private void BtnViewOrders_Click(object sender, EventArgs e)

```

```

    {
        var ordersForm = new SellerOrdersForm(_sellerId);
        ordersForm.ShowDialog();
    }

    private void BtnStatistics_Click(object sender, EventArgs e)
    {
        var statsForm = new StatisticsForm(_sellerId);
        statsForm.ShowDialog();
    }

    private void EditSelectedProduct()
    {
        if (_productsGrid.SelectedRows.Count == 0) return;

        var productId =
(int)_productsGrid.SelectedRows[0].Cells["product_id"].Value;
        var editForm = new EditProductForm(productId);
        editForm.FormClosed += (s, args) => LoadProducts();
        editForm.ShowDialog();
    }

    private void DeleteSelectedProduct()
    {
        if (_productsGrid.SelectedRows.Count == 0) return;

        var productId =
(int)_productsGrid.SelectedRows[0].Cells["product_id"].Value;
        var productName =
_productsGrid.SelectedRows[0].Cells["name"].Value.ToString();

        if (MessageBox.Show($"Удалить товар '{productName}'?", "Подтверждение",
            MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            try
            {
                string connectionString =
"server=localhost;user=root;password=;database=wb;";
                string query = "DELETE FROM products WHERE product_id =
@productId AND seller_id = @sellerId";

                using (var connection = new MySqlConnection(connectionString))
                using (var cmd = new MySqlCommand(query, connection))
                {
                    cmd.Parameters.AddWithValue("@productId", productId);
                    cmd.Parameters.AddWithValue("@sellerId", _sellerId);
                    connection.Open();
                    cmd.ExecuteNonQuery();
                }

                LoadProducts(); // Обновляем список
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка удаления: {ex.Message}");
            }
        }
    }

    private void SellerForm_Load(object sender, EventArgs e)
    {
    }
}

```

## Форма добавления продукта

```
using System;
using System.Drawing;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

namespace WB
{
    public partial class AddProductForm : Form
    {
        // Элементы управления
        private TextBox txtName;
        private TextBox txtPrice;
        private TextBox txtQuantity;
        private ComboBox cmbCategory;
        private RichTextBox txtDescription;
        private Button btnSave;
        private Button btnCancel;
        private Button btnUploadImage;
        private PictureBox picPreview;

        // Данные
        private int sellerId;
        private string imagePath = string.Empty;

        // Цвета
        private readonly Color purpleColor = Color.FromArgb(141, 93, 167);
        private readonly Color lightPurple = Color.FromArgb(161, 113, 187);

        public AddProductForm(int sellerId)
        {
            this.sellerId = sellerId;

            // Инициализация компонентов и установка размеров
            InitializeComponent();
            this.ClientSize = new Size(1200, 1000); // Большой размер окна
            this.MinimumSize = new Size(1200, 1000); // Минимальный размер

            InitializeForm();
            LoadCategories();
        }

        private void InitializeForm()
        {
            // Настройки формы
            this.Text = "Добавление нового товара";
            this.StartPosition = FormStartPosition.CenterScreen;
            this.FormBorderStyle = FormBorderStyle.FixedDialog;
            this.MaximizeBox = false;
            this.BackColor = Color.White;
            this.Font = new Font("Arial", 12); // Крупный шрифт
        }
    }
}
```

```

        // Создание и размещение элементов
        CreateControls();
        SetupEventHandlers();
    }

    private void CreateControls()
    {
        int yPos = 40;
        int labelWidth = 250;
        int controlWidth = 600;
        int rowHeight = 70;

        // Название товара
        var lblName = CreateLabel("Название товара:", 40, yPos, labelWidth);
        txtName = CreateTextBox(300, yPos, controlWidth, 40);
        yPos += rowHeight;

        // Цена
        var lblPrice = CreateLabel("Цена (руб):", 40, yPos, labelWidth);
        txtPrice = CreateTextBox(300, yPos, 200, 40);
        yPos += rowHeight;

        // Количество
        var lblQuantity = CreateLabel("Количество:", 40, yPos, labelWidth);
        txtQuantity = CreateTextBox(300, yPos, 200, 40);
        txtQuantity.Text = "1";
        yPos += rowHeight;

        // Категория
        var lblCategory = CreateLabel("Категория:", 40, yPos, labelWidth);
        cmbCategory = new ComboBox
        {
            Location = new Point(300, yPos),
            Size = new Size(controlWidth, 45),
            DropDownStyle = ComboBoxStyle.DropDownList,
            Font = new Font("Arial", 12)
        };
        yPos += rowHeight;

        // Описание
        var lblDescription = CreateLabel("Описание:", 40, yPos, labelWidth);
        txtDescription = new RichTextBox
        {
            Location = new Point(300, yPos),
            Size = new Size(controlWidth, 180),
            Font = new Font("Arial", 12),
            BorderStyle = BorderStyle.FixedSingle
        };
        yPos += 200;

        // Изображение
        var lblImage = CreateLabel("Изображение товара:", 40, yPos, labelWidth);
    }

```

```

btnUploadImage = new Button
{
    Text = "Выбрать изображение",
    Location = new Point(300, yPos),
    Size = new Size(200, 50),
    BackColor = purpleColor,
    ForeColor = Color.White,
    FlatStyle = FlatStyle.Flat,
    Font = new Font("Arial", 12)
};

picPreview = new PictureBox
{
    Location = new Point(520, yPos),
    Size = new Size(300, 300),
    SizeMode = PictureBoxSizeMode.Zoom,
    BorderStyle = BorderStyle.FixedSingle,
    BackColor = Color.WhiteSmoke
};
yPos += 320;

// Кнопки
btnSave = new Button
{
    Text = "Сохранить товар",
    Location = new Point(300, yPos),
    Size = new Size(250, 60),
    BackColor = purpleColor,
    ForeColor = Color.White,
    Font = new Font("Arial", 12, FontStyle.Bold),
    FlatStyle = FlatStyle.Flat
};

btnCancel = new Button
{
    Text = "Отменить",
    Location = new Point(570, yPos),
    Size = new Size(250, 60),
    Font = new Font("Arial", 12),
    FlatStyle = FlatStyle.Flat
};

// Добавление элементов на форму
this.Controls.AddRange(new Control[] {
    lblName, txtName,
    lblPrice, txtPrice,
    lblQuantity, txtQuantity,
    lblCategory, cmbCategory,
    lblDescription, txtDescription,
    lblImage, btnUploadImage, picPreview,
    btnSave, btnCancel
});

```

```

    }

    private Label CreateLabel(string text, int x, int y, int width)
    {
        return new Label
        {
            Text = text,
            Location = new Point(x, y),
            Size = new Size(width, 40),
            Font = new Font("Arial", 12, FontStyle.Bold)
        };
    }

    private TextBox CreateTextBox(int x, int y, int width, int height)
    {
        return new TextBox
        {
            Location = new Point(x, y),
            Size = new Size(width, height),
            Font = new Font("Arial", 12)
        };
    }

    private void SetupEventHandlers()
    {
        btnSave.Click += SaveProduct;
        btnCancel.Click += (s, e) => this.Close();
        btnUploadImage.Click += UploadImage;
    }

    private void LoadCategories()
    {
        try
        {
            using (var connection = new
                MySqlConnection("server=localhost;user=root;password=;database=wb;"))
            {
                connection.Open();
                var command = new MySqlCommand("SELECT category_id, name FROM
                categories ORDER BY name", connection);

                using (var reader = command.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        cmbCategory.Items.Add(new CategoryItem(
                            reader["name"].ToString(),
                            Convert.ToInt32(reader["category_id"]))
                        );
                    }
                }
            }
        }
    }

```



```

    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка загрузки категорий: {ex.Message}",
"Ошибка",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void UploadImage(object sender, EventArgs e)
{
    using (var dialog = new OpenFileDialog())
    {
        dialog.Filter = "Изображения (*.jpg;*.jpeg;*.png)|*.jpg;*.jpeg;*.png";
        dialog.Title = "Выберите изображение товара";

        if (dialog.ShowDialog() == DialogResult.OK)
        {
            imagePath = dialog.FileName;
            picPreview.Image = Image.FromFile(imagePath);
        }
    }
}

private void SaveProduct(object sender, EventArgs e)
{
    if (!ValidateInputs()) return;

    try
    {
        using (var connection = new
MySQLConnection("server=localhost;user=root;password=;database=wb;"))
        {
            connection.Open();

            var productCommand = new MySqlCommand(
@"INSERT INTO products
(seller_id, category_id, name, description, price, quantity)
VALUES (@sellerId, @categoryId, @name, @description, @price,
@quantity);

SELECT LAST_INSERT_ID();",
            connection);

            var category = (CategoryItem)cmbCategory.SelectedItem;

            productCommand.Parameters.AddWithValue("@sellerId", sellerId);
            productCommand.Parameters.AddWithValue("@categoryId",
category.Id);

            productCommand.Parameters.AddWithValue("@name",
txtName.Text.Trim());
            productCommand.Parameters.AddWithValue("@description",

```

```

txtDescription.Text.Trim());
        productCommand.Parameters.AddWithValue("@price",
decimal.Parse(txtPrice.Text));
        productCommand.Parameters.AddWithValue("@quantity",
int.Parse(txtQuantity.Text));

        int productId = Convert.ToInt32(productCommand.ExecuteScalar());

        if (!string.IsNullOrEmpty(imagePath))
        {
            var imageCommand = new MySqlCommand(
                @"INSERT INTO product_images
                (product_id, image_url, is_main)
                VALUES (@productId, @imageUrl, @isMain)",
                connection);

            imageCommand.Parameters.AddWithValue("@productId",
productId);
            imageCommand.Parameters.AddWithValue("@imageUrl",
imagePath);
            imageCommand.Parameters.AddWithValue("@isMain", true);

            imageCommand.ExecuteNonQuery();
        }
    }

    MessageBox.Show("Товар успешно добавлен", "Успех",
        MessageBoxButtons.OK, MessageBoxIcon.Information);

    this.DialogResult = DialogResult.OK;
    this.Close();
}
catch (Exception ex)
{
    MessageBox.Show($"Ошибка сохранения товара: {ex.Message}", "Ошибка",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
}

private bool ValidateInputs()
{
    if (string.IsNullOrEmpty(txtName.Text))
    {
        ShowError("Введите название товара", txtName);
        return false;
    }

    if (!decimal.TryParse(txtPrice.Text, out decimal price) || price <= 0)
    {
        ShowError("Введите корректную цену (положительное число)",
txtPrice);
        return false;
    }
}

```

```

    }

    if (!int.TryParse(txtQuantity.Text, out int quantity) || quantity <= 0)
    {
        ShowError("Введите корректное количество (целое число больше 0)",
txtQuantity);
        return false;
    }

    if (cmbCategory.SelectedItem == null)
    {
        ShowError("Выберите категорию", cmbCategory);
        return false;
    }

    return true;
}

private void ShowError(string message, Control control)
{
    MessageBox.Show(message, "Ошибка ввода", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
    control.Focus();
    if (control is TextBox textBox) textBox.SelectAll();
}

private class CategoryItem
{
    public string Name { get; }
    public int Id { get; }

    public CategoryItem(string name, int id)
    {
        Name = name;
        Id = id;
    }

    public override string ToString() => Name;
}

private void AddProductForm_Load(object sender, EventArgs e)
{
}
}

```

### **Форма изменения товара**

```

using System;
using System.Drawing;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

```

```

namespace WB
{
    public partial class AddProductForm : Form
    {
        // Элементы управления
        private TextBox txtName;
        private TextBox txtPrice;
        private TextBox txtQuantity;
        private ComboBox cmbCategory;
        private RichTextBox txtDescription;
        private Button btnSave;
        private Button btnCancel;
        private Button btnUploadImage;
        private PictureBox picPreview;

        // Данные
        private int sellerId;
        private string imagePath = string.Empty;

        // Цвета
        private readonly Color purpleColor = Color.FromArgb(141, 93, 167);
        private readonly Color lightPurple = Color.FromArgb(161, 113, 187);

        public AddProductForm(int sellerId)
        {
            this.sellerId = sellerId;

            // Инициализация компонентов и установка размеров
            InitializeComponent();
            this.ClientSize = new Size(1200, 1000); // Большой размер окна
            this.MinimumSize = new Size(1200, 1000); // Минимальный размер

            InitializeForm();
            LoadCategories();
        }

        private void InitializeForm()
        {
            // Настройки формы
            this.Text = "Добавление нового товара";
            this.StartPosition = FormStartPosition.CenterScreen;
            this.FormBorderStyle = FormBorderStyle.FixedDialog;
            this.MaximizeBox = false;
            this.BackColor = Color.White;
            this.Font = new Font("Arial", 12); // Крупный шрифт

            // Создание и размещение элементов
            CreateControls();
            SetupEventHandlers();
        }

        private void CreateControls()

```

```

{
    int yPos = 40;
    int labelWidth = 250;
    int controlWidth = 600;
    int rowHeight = 70;

    // Название товара
    var lblName = CreateLabel("Название товара:", 40, yPos, labelWidth);
    txtName = CreateTextBox(300, yPos, controlWidth, 40);
    yPos += rowHeight;

    // Цена
    var lblPrice = CreateLabel("Цена (руб):", 40, yPos, labelWidth);
    txtPrice = CreateTextBox(300, yPos, 200, 40);
    yPos += rowHeight;

    // Количество
    var lblQuantity = CreateLabel("Количество:", 40, yPos, labelWidth);
    txtQuantity = CreateTextBox(300, yPos, 200, 40);
    txtQuantity.Text = "1";
    yPos += rowHeight;

    // Категория
    var lblCategory = CreateLabel("Категория:", 40, yPos, labelWidth);
    cmbCategory = new ComboBox
    {
        Location = new Point(300, yPos),
        Size = new Size(controlWidth, 45),
        DropDownStyle = ComboBoxStyle.DropDownList,
        Font = new Font("Arial", 12)
    };
    yPos += rowHeight;

    // Описание
    var lblDescription = CreateLabel("Описание:", 40, yPos, labelWidth);
    txtDescription = new RichTextBox
    {
        Location = new Point(300, yPos),
        Size = new Size(controlWidth, 180),
        Font = new Font("Arial", 12),
        BorderStyle = BorderStyle.FixedSingle
    };
    yPos += 200;

    // Изображение
    var lblImage = CreateLabel("Изображение товара:", 40, yPos, labelWidth);
    btnUploadImage = new Button
    {
        Text = "Выбрать изображение",
        Location = new Point(300, yPos),
        Size = new Size(200, 50),
        BackColor = purpleColor,

```

```

        ForeColor = Color.White,
        FlatStyle = FlatStyle.Flat,
        Font = new Font("Arial", 12)
    };

    picPreview = new PictureBox
    {
        Location = new Point(520, yPos),
        Size = new Size(300, 300),
        SizeMode = PictureBoxSizeMode.Zoom,
        BorderStyle = BorderStyle.FixedSingle,
        BackColor = Color.WhiteSmoke
    };
    yPos += 320;

    // Кнопки
    btnSave = new Button
    {
        Text = "Сохранить товар",
        Location = new Point(300, yPos),
        Size = new Size(250, 60),
        BackColor = purpleColor,
        ForeColor = Color.White,
        Font = new Font("Arial", 12, FontStyle.Bold),
        FlatStyle = FlatStyle.Flat
    };

    btnCancel = new Button
    {
        Text = "Отменить",
        Location = new Point(570, yPos),
        Size = new Size(250, 60),
        Font = new Font("Arial", 12),
        FlatStyle = FlatStyle.Flat
    };

    // Добавление элементов на форму
    this.Controls.AddRange(new Control[] {
        lblName, txtName,
        lblPrice, txtPrice,
        lblQuantity, txtQuantity,
        lblCategory, cmbCategory,
        lblDescription, txtDescription,
        lblImage, btnUploadImage, picPreview,
        btnSave, btnCancel
    });
}

private Label CreateLabel(string text, int x, int y, int width)
{
    return new Label
    {

```

```

        Text = text,
        Location = new Point(x, y),
        Size = new Size(width, 40),
        Font = new Font("Arial", 12, FontStyle.Bold)
    };
}

private TextBox CreateTextBox(int x, int y, int width, int height)
{
    return new TextBox
    {
        Location = new Point(x, y),
        Size = new Size(width, height),
        Font = new Font("Arial", 12)
    };
}

private void SetupEventHandlers()
{
    btnSave.Click += SaveProduct;
    btnCancel.Click += (s, e) => this.Close();
    btnUploadImage.Click += UploadImage;
}

private void LoadCategories()
{
    try
    {
        using (var connection = new
MySQLConnection("server=localhost;user=root;password=;database=wb;"))
        {
            connection.Open();
            var command = new MySqlCommand("SELECT category_id, name FROM
categories ORDER BY name", connection);

            using (var reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    cmbCategory.Items.Add(new CategoryItem(
                        reader["name"].ToString(),
                        Convert.ToInt32(reader["category_id"]))
                    );
                }
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка загрузки категорий: {ex.Message}",
"Ошибка",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

```

```

    }
}

private void UploadImage(object sender, EventArgs e)
{
    using (var dialog = new OpenFileDialog())
    {
        dialog.Filter = "Изображения (*.jpg;*.jpeg;*.png)|*.jpg;*.jpeg;*.png";
        dialog.Title = "Выберите изображение товара";

        if (dialog.ShowDialog() == DialogResult.OK)
        {
            imagePath = dialog.FileName;
            picPreview.Image = Image.FromFile(imagePath);
        }
    }
}

private void SaveProduct(object sender, EventArgs e)
{
    if (!ValidateInputs()) return;

    try
    {
        using (var connection = new MySqlConnection("server=localhost;user=root;password=;database=wb;"))
        {
            connection.Open();

            var productCommand = new MySqlCommand(
                @"INSERT INTO products
                (seller_id, category_id, name, description, price, quantity)
                VALUES (@sellerId, @categoryId, @name, @description, @price,
@quantity);

                SELECT LAST_INSERT_ID();",
                connection);

            var category = (CategoryItem)cmbCategory.SelectedItem;

            productCommand.Parameters.AddWithValue("@sellerId", sellerId);
            productCommand.Parameters.AddWithValue("@categoryId",
category.Id);
            productCommand.Parameters.AddWithValue("@name",
txtName.Text.Trim());
            productCommand.Parameters.AddWithValue("@description",
txtDescription.Text.Trim());
            productCommand.Parameters.AddWithValue("@price",
decimal.Parse(txtPrice.Text));
            productCommand.Parameters.AddWithValue("@quantity",
int.Parse(txtQuantity.Text));

```



```

        int productId = Convert.ToInt32(productCommand.ExecuteScalar());

        if (!string.IsNullOrEmpty(imagePath))
        {
            var imageCommand = new MySqlCommand(
                @"INSERT INTO product_images
                (product_id, image_url, is_main)
                VALUES (@productId, @imageUrl, @isMain)",
                connection);

            imageCommand.Parameters.AddWithValue("@productId",
productId);

            imageCommand.Parameters.AddWithValue("@imageUrl",
imagePath);

            imageCommand.Parameters.AddWithValue("@isMain", true);

            imageCommand.ExecuteNonQuery();
        }

        MessageBox.Show("Товар успешно добавлен", "Успех",
            MessageBoxButtons.OK, MessageBoxIcon.Information);

        this.DialogResult = DialogResult.OK;
        this.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка сохранения товара: {ex.Message}", "Ошибка",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private bool ValidateInputs()
{
    if (string.IsNullOrEmpty(txtName.Text))
    {
        ShowError("Введите название товара", txtName);
        return false;
    }

    if (!decimal.TryParse(txtPrice.Text, out decimal price) || price <= 0)
    {
        ShowError("Введите корректную цену (положительное число)",
txtPrice);
        return false;
    }

    if (!int.TryParse(txtQuantity.Text, out int quantity) || quantity <= 0)
    {
        ShowError("Введите корректное количество (целое число больше 0)",
txtQuantity);
    }
}

```

```

        return false;
    }

    if (cmbCategory.SelectedItem == null)
    {
        ShowError("Выберите категорию", cmbCategory);
        return false;
    }

    return true;
}

private void ShowError(string message, Control control)
{
    MessageBox.Show(message, "Ошибка ввода", MessageBoxButtons.OK,
        MessageBoxIcon.Warning);
    control.Focus();
    if (control is TextBox textBox) textBox.SelectAll();
}

private class CategoryItem
{
    public string Name { get; }
    public int Id { get; }

    public CategoryItem(string name, int id)
    {
        Name = name;
        Id = id;
    }

    public override string ToString() => Name;
}

private void AddProductForm_Load(object sender, EventArgs e)
{
}
}

```

## Форма Продаж

```

using System;
using System.Data;
using System.Drawing;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

namespace WB
{
    public partial class SellerOrdersForm : Form
    {
        private int _sellerId;
    }
}

```

```

private DataGridView _ordersGrid;
private Button _btnRefresh;
private Button _btnOrderDetails;
private Button _btnUpdateStatus;
private ComboBox _cmbStatusFilter;

// Цвета Wildberries
private readonly Color _purpleColor = Color.FromArgb(141, 93, 167);
private readonly Color _lightPurple = Color.FromArgb(161, 113, 187);

public SellerOrdersForm(int sellerId)
{
    _sellerId = sellerId;
    InitializeComponent();
    this.Load += SellerOrdersForm_Load;
}

private void SellerOrdersForm_Load(object sender, EventArgs e)
{
    InitializeForm();
    LoadOrders();
}

private void InitializeForm()
{
    // Настройки формы
    this.Text = $"Управление заказами – Продавец ID: {_sellerId}";
    this.ClientSize = new Size(1000, 600);
    this.StartPosition = FormStartPosition.CenterScreen;
    this.BackColor = Color.White;

    // Заголовок
    var lblTitle = new Label
    {
        Text = "Управление заказами",
        Font = new Font("Arial", 18, FontStyle.Bold),
        ForeColor = _purpleColor,
        Location = new Point(20, 20),
        AutoSize = true
    };
    this.Controls.Add(lblTitle);

    // Фильтр по статусу
    var lblFilter = new Label
    {
        Text = "Фильтр по статусу:",
        Location = new Point(20, 70),
        AutoSize = true
    };

    _cmbStatusFilter = new ComboBox
    {

```

```

        Location = new Point(150, 70),
        Size = new Size(150, 30),
        DropDownStyle = ComboBoxStyle.DropDownList
    };
    _cmbStatusFilter.Items.AddRange(new object[] { "Bce", "pending",
"processing", "shipped", "delivered", "cancelled" });
    _cmbStatusFilter.SelectedIndex = 0;
    _cmbStatusFilter.SelectedIndexChanged += (s, e) => LoadOrders();
    this.Controls.AddRange(new Control[] { lblFilter, _cmbStatusFilter });

    // Кнопка обновления
    _btnRefresh = new Button
    {
        Text = "Обновить",
        Location = new Point(320, 70),
        Size = new Size(100, 30),
        BackColor = _purpleColor,
        ForeColor = Color.White,
        FlatStyle = FlatStyle.Flat
    };
    _btnRefresh.Click += (s, e) => LoadOrders();
    this.Controls.Add(_btnRefresh);

    // Таблица заказов
    _ordersGrid = new DataGridView
    {
        Location = new Point(20, 120),
        Size = new Size(960, 400),
        Anchor = AnchorStyles.Top | AnchorStyles.Bottom | AnchorStyles.Left
| AnchorStyles.Right,
        BackgroundColor = Color.White,
        BorderStyle = BorderStyle.None,
        AllowUserToAddRows = false,
        ReadOnly = true,
        AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill,
        SelectionMode = DataGridViewSelectionMode.FullRowSelect,
        RowHeadersVisible = false
    };
    _ordersGrid.CellFormatting += OrdersGrid_CellFormatting;
    this.Controls.Add(_ordersGrid);

    // Кнопка деталей заказа
    _btnOrderDetails = new Button
    {
        Text = "Детали заказа",
        Location = new Point(440, 70),
        Size = new Size(120, 30),
        BackColor = _lightPurple,
        ForeColor = Color.White,
        FlatStyle = FlatStyle.Flat
    };
    _btnOrderDetails.Click += BtnOrderDetails_Click;

```

```

this.Controls.Add(_btnOrderDetails);

// Кнопка изменения статуса
_btnUpdateStatus = new Button
{
    Text = "Изменить статус",
    Location = new Point(580, 70),
    Size = new Size(120, 30),
    BackColor = _lightPurple,
    ForeColor = Color.White,
    FlatStyle = FlatStyle.Flat
};
_btnUpdateStatus.Click += BtnUpdateStatus_Click;
this.Controls.Add(_btnUpdateStatus);

// Стилизация
_btnRefresh.FlatAppearance.MouseOverBackColor = _lightPurple;
_btnOrderDetails.FlatAppearance.MouseOverBackColor = Color.FromArgb(131,
83, 157);
_btnUpdateStatus.FlatAppearance.MouseOverBackColor = Color.FromArgb(131,
83, 157);
}

private void LoadOrders()
{
    try
    {
        string connectionString =
"server=localhost;user=root;password=;database=wb;";
        string query = @"SELECT o.order_id, o.order_date, o.status,
o.total_amount,
u.username as customer, COUNT(oi.order_item_id) as
items_count
FROM orders o
JOIN order_items oi ON o.order_id = oi.order_id
JOIN products p ON oi.product_id = p.product_id
JOIN users u ON o.user_id = u.user_id
WHERE p.seller_id = @sellerId
{0}
GROUP BY o.order_id
ORDER BY o.order_date DESC";

        string statusFilter = "";
        if (_cmbStatusFilter.SelectedIndex > 0)
        {
            statusFilter = "AND o.status = @status";
        }

        query = string.Format(query, statusFilter);

        using (var connection = new MySqlConnection(connectionString))
        using (var cmd = new MySqlCommand(query, connection))

```

```

        {
            cmd.Parameters.AddWithValue("@sellerId", _sellerId);

            if (_cmbStatusFilter.SelectedIndex > 0)
            {
                cmd.Parameters.AddWithValue("@status",
                _cmbStatusFilter.SelectedItem.ToString());
            }

            var adapter = new MySqlDataAdapter(cmd);
            var table = new DataTable();
            adapter.Fill(table);

            _ordersGrid.DataSource = table;
            FormatOrdersGrid();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка загрузки заказов: {ex.Message}", "Ошибка",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void FormatOrdersGrid()
{
    if (_ordersGrid.Columns.Count == 0) return;

    // Настройка столбцов
    _ordersGrid.Columns["order_id"].HeaderText = "Номер заказа";
    _ordersGrid.Columns["order_date"].HeaderText = "Дата заказа";
    _ordersGrid.Columns["status"].HeaderText = "Статус";
    _ordersGrid.Columns["total_amount"].HeaderText = "Сумма";
    _ordersGrid.Columns["customer"].HeaderText = "Покупатель";
    _ordersGrid.Columns["items_count"].HeaderText = "Товаров";

    // Форматирование
    _ordersGrid.Columns["order_date"].DefaultCellStyle.Format = "dd.MM.yyyy
HH:mm";
    _ordersGrid.Columns["total_amount"].DefaultCellStyle.Format = "N2 руб.";
}

private void OrdersGrid_CellFormatting(object sender,
DataGridViewCellFormattingEventArgs e)
{
    if (_ordersGrid.Columns[e.ColumnIndex].Name == "status")
    {
        switch (e.Value.ToString())
        {
            case "pending":
                e.CellStyle.BackColor = Color.LightYellow;
                break;
        }
    }
}

```

```

        case "processing":
            e.CellStyle.BackColor = Color.LightBlue;
            break;
        case "shipped":
            e.CellStyle.BackColor = Color.LightGreen;
            break;
        case "delivered":
            e.CellStyle.BackColor = Color.PaleGreen;
            break;
        case "cancelled":
            e.CellStyle.BackColor = Color.LightPink;
            break;
    }
}

private void BtnOrderDetails_Click(object sender, EventArgs e)
{
    if (_ordersGrid.SelectedRows.Count == 0)
    {
        MessageBox.Show("Выберите заказ для просмотра деталей",
"Информация",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
        return;
    }

    var orderId = (int)_ordersGrid.SelectedRows[0].Cells["order_id"].Value;
    //var detailsForm = new OrderDetailsForm(orderId, _sellerId);
    //detailsForm.ShowDialog();
    LoadOrders(); // Обновляем список после закрытия формы деталей
}

private void BtnUpdateStatus_Click(object sender, EventArgs e)
{
    if (_ordersGrid.SelectedRows.Count == 0)
    {
        MessageBox.Show("Выберите заказ для изменения статуса",
"Информация",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
        return;
    }

    var orderId = (int)_ordersGrid.SelectedRows[0].Cells["order_id"].Value;
    var currentStatus =
_ordersGrid.SelectedRows[0].Cells["status"].Value.ToString();

    //using (var statusForm = new StatusUpdateForm(orderId, currentStatus))
    //{
    //    if (statusForm.ShowDialog() == DialogResult.OK)
    //    {
    //        LoadOrders(); // Обновляем список после изменения статуса
    //    }
}

```

```

        //}
    }
}

```

## Форма статистики

```

using System;
using System.Data;
using System.Drawing;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;
using MySql.Data.MySqlClient;

namespace WB
{
    public partial class StatisticsForm : Form
    {
        private int _sellerId;
        private Chart _salesChart;
        private DataGridView _topProductsGrid;
        private ComboBox _periodComboBox;
        private DateTimePicker _dateFromPicker;
        private DateTimePicker _dateToPicker;

        // Цвета Wildberries
        private readonly Color _purpleColor = Color.FromArgb(141, 93, 167);
        private readonly Color _lightPurple = Color.FromArgb(161, 113, 187);

        public StatisticsForm(int sellerId)
        {
            _sellerId = sellerId;
            InitializeComponent();
            this.Load += StatisticsForm_Load;
        }

        private void StatisticsForm_Load(object sender, EventArgs e)
        {
            InitializeForm();
            LoadStatistics();
        }

        private void InitializeForm()
        {
            // Настройки формы
            this.Text = $"Аналитика продаж - Продавец ID: {_sellerId}";
            this.ClientSize = new Size(1000, 700);
            this.StartPosition = FormStartPosition.CenterScreen;
            this.BackColor = Color.White;

            // Заголовок
            var lblTitle = new Label
            {
                Text = "Аналитика продаж",
                Font = new Font("Arial", 18, FontStyle.Bold),
                ForeColor = _purpleColor,
                Location = new Point(20, 20),
                AutoSize = true
            };
            this.Controls.Add(lblTitle);

            // Панель управления
            var controlPanel = new Panel
            {
                Location = new Point(20, 60),

```



```

        Size = new Size(960, 60),
        BorderStyle = BorderStyle.FixedSingle
    };

    // Период анализа
    var lblPeriod = new Label
    {
        Text = "Период:",
        Location = new Point(10, 20),
        AutoSize = true
    };

    _periodComboBox = new ComboBox
    {
        Location = new Point(70, 17),
        Size = new Size(120, 30),
        DropDownStyle = ComboBoxStyle.DropDownList
    };
    _periodComboBox.Items.AddRange(new object[] { "Сегодня", "Неделя",
"Месяц", "Квартал", "Год", "Произвольный" });
    _periodComboBox.SelectedIndex = 2; // Месяц по умолчанию
    _periodComboBox.SelectedIndexChanged +=
PeriodComboBox_SelectedIndexChanged;

    // Даты для произвольного периода
    _dateFromPicker = new DateTimePicker
    {
        Location = new Point(200, 17),
        Size = new Size(120, 30),
        Visible = false
    };

    _dateToPicker = new DateTimePicker
    {
        Location = new Point(330, 17),
        Size = new Size(120, 30),
        Visible = false
    };

    var lblTo = new Label
    {
        Text = "до",
        Location = new Point(280, 20),
        AutoSize = true,
        Visible = false
    };

    // Кнопка обновления
    var btnRefresh = new Button
    {
        Text = "Обновить",
        Location = new Point(470, 17),
        Size = new Size(100, 30),
        BackColor = _purpleColor,
        ForeColor = Color.White,
        FlatStyle = FlatStyle.Flat
    };
    btnRefresh.Click += (s, e) => LoadStatistics();

    controlPanel.Controls.AddRange(new Control[] {
        lblPeriod, _periodComboBox, _dateFromPicker, _dateToPicker, lblTo,
btnRefresh
    });
    this.Controls.Add(controlPanel);

```

```

// График продаж
_salesChart = new Chart
{
    Location = new Point(20, 140),
    Size = new Size(600, 300),
    BackColor = Color.White
};

var chartArea = new ChartArea("MainArea");
chartArea.AxisX.MajorGrid.LineColor = Color.LightGray;
chartArea.AxisY.MajorGrid.LineColor = Color.LightGray;
_salesChart.ChartAreas.Add(chartArea);

this.Controls.Add(_salesChart);

// Таблица популярных товаров
var lblTopProducts = new Label
{
    Text = "Самые популярные товары:",
    Location = new Point(640, 140),
    AutoSize = true,
    Font = new Font("Arial", 10, FontStyle.Bold)
};
this.Controls.Add(lblTopProducts);

_topProductsGrid = new DataGridView
{
    Location = new Point(640, 170),
    Size = new Size(340, 270),
    BackgroundColor = Color.White,
    BorderStyle = BorderStyle.FixedSingle,
    AllowUserToAddRows = false,
    ReadOnly = true,
    RowHeadersVisible = false,
    AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill
};
this.Controls.Add(_topProductsGrid);

// Общая статистика
var lblSummary = new Label
{
    Text = "Общая статистика:",
    Location = new Point(20, 460),
    AutoSize = true,
    Font = new Font("Arial", 10, FontStyle.Bold)
};
this.Controls.Add(lblSummary);
}

private void PeriodComboBox_SelectedIndexChanged(object sender, EventArgs e)
{
    bool customPeriod = _periodComboBox.SelectedIndex == 5;
    _dateFromPicker.Visible = customPeriod;
    _dateToPicker.Visible = customPeriod;

    // Установка дат по умолчанию для выбранного периода
    DateTime now = DateTime.Now;
    switch (_periodComboBox.SelectedIndex)
    {
        case 0: // Сегодня
            _dateFromPicker.Value = now.Date;
            _dateToPicker.Value = now.Date.AddDays(1).AddSeconds(-1);
            break;
        case 1: // Неделя
            _dateFromPicker.Value = now.Date.AddDays(-(int)now.DayOfWeek);
    }
}

```

```

        _dateToPicker.Value =
_dateFromPicker.Value.AddDays(7).AddSeconds(-1);
        break;
        case 2: // Месяц
            _dateFromPicker.Value = new DateTime(now.Year, now.Month, 1);
            _dateToPicker.Value =
_dateFromPicker.Value.AddMonths(1).AddSeconds(-1);
            break;
        case 3: // Квартал
            int quarter = (now.Month - 1) / 3 + 1;
            _dateFromPicker.Value = new DateTime(now.Year, (quarter - 1) * 3
+ 1, 1);
            _dateToPicker.Value =
_dateFromPicker.Value.AddMonths(3).AddSeconds(-1);
            break;
        case 4: // Год
            _dateFromPicker.Value = new DateTime(now.Year, 1, 1);
            _dateToPicker.Value =
_dateFromPicker.Value.AddYears(1).AddSeconds(-1);
            break;
    }
}

private void LoadStatistics()
{
    DateTime fromDate, toDate;
    GetPeriodDates(out fromDate, out toDate);

    try
    {
        // Загрузка данных для графика продаж
        LoadSalesChartData(fromDate, toDate);

        // Загрузка популярных товаров
        LoadTopProductsData(fromDate, toDate);

        // Загрузка общей статистики
        LoadSummaryData(fromDate, toDate);
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка загрузки статистики: {ex.Message}",
"Ошибка",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void GetPeriodDates(out DateTime fromDate, out DateTime toDate)
{
    if (_periodComboBox.SelectedIndex == 5) // Произвольный период
    {
        fromDate = _dateFromPicker.Value;
        toDate = _dateToPicker.Value;
    }
    else
    {
        PeriodComboBox_SelectedIndexChanged(null, null);
        fromDate = _dateFromPicker.Value;
        toDate = _dateToPicker.Value;
    }
}

private void LoadSalesChartData(DateTime fromDate, DateTime toDate)
{

```

```

        string connectionString =
"server=localhost;user=root;password=;database=wb;";
        string query = @"SELECT DATE(o.order_date) as order_day,
                            SUM(oi.quantity * oi.price_per_unit) as total_sales,
                            COUNT(DISTINCT o.order_id) as orders_count
                            FROM orders o
                            JOIN order_items oi ON o.order_id = oi.order_id
                            JOIN products p ON oi.product_id = p.product_id
                            WHERE p.seller_id = @sellerId
                            AND o.order_date BETWEEN @fromDate AND @toDate
                            AND o.status != 'cancelled'
                            GROUP BY DATE(o.order_date)
                            ORDER BY order_day";

        using (var connection = new MySqlConnection(connectionString))
        using (var cmd = new MySqlCommand(query, connection))
        {
            cmd.Parameters.AddWithValue("@sellerId", _sellerId);
            cmd.Parameters.AddWithValue("@fromDate", fromDate);
            cmd.Parameters.AddWithValue("@toDate", toDate);

            var adapter = new MySqlDataAdapter(cmd);
            var table = new DataTable();
            adapter.Fill(table);

            // Очистка предыдущих данных
            _salesChart.Series.Clear();

            // Добавление серии для продаж
            var salesSeries = new Series("Продажи")
            {
                ChartType = SeriesChartType.Column,
                Color = _purpleColor,
                XValueMember = "order_day",
                YValueMembers = "total_sales",
                IsValueShownAsLabel = true,
                LabelFormat = "N0"
            };
            _salesChart.Series.Add(salesSeries);

            // Добавление серии для количества заказов
            var ordersSeries = new Series("Заказы")
            {
                ChartType = SeriesChartType.Line,
                Color = Color.Orange,
                XValueMember = "order_day",
                YValueMembers = "orders_count",
                IsValueShownAsLabel = true,
                BorderWidth = 3
            };
            _salesChart.Series.Add(ordersSeries);

            // Привязка данных
            _salesChart.DataSource = table;
            _salesChart.DataBind();

            // Настройка осей
            _salesChart.ChartAreas[0].AxisX.LabelStyle.Format = "dd.MM";
            _salesChart.ChartAreas[0].AxisY.LabelStyle.Format = "N0";
            _salesChart.ChartAreas[0].AxisX.Interval = 1;
        }
    }

    private void LoadTopProductsData(DateTime fromDate, DateTime toDate)
    {

```

```

        string connectionString =
"server=localhost;user=root;password=;database=wb;";
        string query = @"SELECT p.name as product_name,
                                SUM(oi.quantity) as total_quantity,
                                SUM(oi.quantity * oi.price_per_unit) as total_sales,
                                COUNT(DISTINCT o.order_id) as orders_count
                                FROM order_items oi
                                JOIN products p ON oi.product_id = p.product_id
                                JOIN orders o ON oi.order_id = o.order_id
                                WHERE p.seller_id = @sellerId
                                AND o.order_date BETWEEN @fromDate AND @toDate
                                AND o.status != 'cancelled'
                                GROUP BY p.product_id
                                ORDER BY total_quantity DESC
                                LIMIT 10";

        using (var connection = new MySqlConnection(connectionString))
        using (var cmd = new MySqlCommand(query, connection))
        {
            cmd.Parameters.AddWithValue("@sellerId", _sellerId);
            cmd.Parameters.AddWithValue("@fromDate", fromDate);
            cmd.Parameters.AddWithValue("@toDate", toDate);

            var adapter = new MySqlDataAdapter(cmd);
            var table = new DataTable();
            adapter.Fill(table);

            _topProductsGrid.DataSource = table;

            // Настройка столбцов
            if (_topProductsGrid.Columns.Count > 0)
            {
                _topProductsGrid.Columns["product_name"].HeaderText = "Товар";
                _topProductsGrid.Columns["total_quantity"].HeaderText = "Кол-
BO";

                _topProductsGrid.Columns["total_sales"].HeaderText = "Сумма";
                _topProductsGrid.Columns["orders_count"].HeaderText = "Заказов";

                _topProductsGrid.Columns["total_sales"].DefaultCellStyle.Format
= "N2 py6.";
            }
        }

        private void LoadSummaryData(DateTime fromDate, DateTime toDate)
        {
            string connectionString =
"server=localhost;user=root;password=;database=wb;";
            string query = @"SELECT
                                COUNT(DISTINCT o.order_id) as total_orders,
                                SUM(oi.quantity) as total_items,
                                SUM(oi.quantity * oi.price_per_unit) as total_sales,
                                COUNT(DISTINCT o.user_id) as unique_customers
                                FROM order_items oi
                                JOIN products p ON oi.product_id = p.product_id
                                JOIN orders o ON oi.order_id = o.order_id
                                WHERE p.seller_id = @sellerId
                                AND o.order_date BETWEEN @fromDate AND @toDate
                                AND o.status != 'cancelled';

            using (var connection = new MySqlConnection(connectionString))
            using (var cmd = new MySqlCommand(query, connection))
            {
                cmd.Parameters.AddWithValue("@sellerId", _sellerId);
                cmd.Parameters.AddWithValue("@fromDate", fromDate);

```

```

cmd.Parameters.AddWithValue("@toDate", toDate);

connection.Open();
using (var reader = cmd.ExecuteReader())
{
    if (reader.Read())
    {
        // Здесь можно обновить элементы интерфейса с общей
статистикой
        // Например:
        string summaryText = $"Всего заказов:
{reader["total_orders"]}\n" +
                                $"Товаров продано:
{reader["total_items"]}\n" +
                                $"Общая сумма:
{Convert.ToDecimal(reader["total_sales"]):N2} руб.\n" +
                                $"Уникальных покупателей:
{reader["unique_customers"]}";

        // Добавьте Label или другой элемент для отображения этой
информации
    }
}
}
}
}
}
}
}

```