Федеральное государственное бюджетное образовательное учреждение высшего образования

**РОССИЙСКАЯ АКАДЕМИЯ  
НАРОДНОГО ХОЗЯЙСТВА и ГОСУДАРСТВЕННОЙ СЛУЖБЫ  
при ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ЗАПАДНЫЙ ФИЛИАЛ**

**ОТЧЕТ**

о прохождении учебной практики по профилю специальности

ПМ.02 Осуществление интеграции программных модулей

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| студента | *23ИСПп5-о9* | | | группы | | | *3* | | курса |
| *Дехтеренко Максим Дмитриевич* | | | | | | | | | |
| *Фамилия Имя Отчество* | | | | | | | | | |
| по специальности | | *09.02.07 Информационные системы и программирование* | | | | | | | |
|  |  | *Код, наименование специальности* | | | | | | | |
| Место прохождения практики | | | | | | *Западный филиал РАНХиГС* | | | |
|  | | | | | | | | | |
| Период прохождения практики | | | | | | *с 22.09.2025 по 05.10.2025* | | | |
|  |  | |  | | |  | | |  |
|  | Дата сдачи | | | | | *05.10.2025* | | | |
|  | Дата защиты | | | | | *05.10.2025* | | | |
|  | Оценка | | | | |  | | | |
|  |  | | | | |  | | | |
|  | | | | | | Руководитель практики | | | |
|  |  | | | | |  | | | |
|  |  | | | | |  | | */Большакова-Стрекалова А.В./* | |
|  |  | | | | | *Подпись* | | *Фамилия И.О.* | |
|  | | | | |  | *05 октября 2025 г.* | | | | |  |
|  | | | | | |  | | | |

Калининград, 2025

Содержание

Введение

1 Разработка программного обеспечения

1.1 Анализ предметной области

1.2 Разработка и оформление технического задания

1.3 Построение архитектуры программного средства

1.4 Изучение работы в системе контроля версий

1.5 Построение диаграмм

1.6 Тестовый сценарий. Оценка программных средств с помощью метрик

1.7 Инспекция программного кода на предмет соответствия стандартам кодирования

2 Разработка АИС « »

2.1 Разработка структуры проекта

2.2 Разработка и интеграция модулей проекта

2.3 Отладка отдельных модулей программного проекта

2.4 Организация обработки исключений

2.5 Тестирование интерфейса пользователя средствами инструментальной среды разработки

2.6 Функциональное тестирование и тестирование интеграции

3 Моделирование в программных системах

Заключение

Список использованных источников

Приложение А. Текст программы

Приложение Б. Тест-кейсы и отчет о тестировании

Введение

Цель и задачи практики.

Настоящий отчёт представляет собой документально оформленный результат прохождения учебной практики по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей». Практика проводилась с целью формирования практических навыков и компетенций в области разработки комплексных программных решений для управления складскими операциями. Актуальность выбранной темы обусловлена стремительным развитием логистического сектора, ростом объёмов товарооборота и критической важностью эффективного управления товарными запасами для обеспечения рентабельности современных торговых предприятий.

В условиях высокой конкуренции и постоянно растущих требований к скорости выполнения заказов, оптовые склады сталкиваются с существенными вызовами при ведении учётных операций традиционными методами. Использование разрозненных электронных таблиц, бумажных журналов и рутинных методов инвентаризации для управления тысячами товарных позиций приводит к неизбежным ошибкам, дублированию данных, затрудняет оперативный доступ к актуальной информации об остатках и перемещениях товаров, значительно увеличивает время обработки заказов и проведения плановых инвентаризаций. Это влечёт за собой прямые финансовые потери из-за затоваривания или, наоборот, нехватки продукции, неэффективное использование складских площадей, снижение общей производительности труда сотрудников склада и, как следствие, ухудшение качества обслуживания клиентов.

Целью практики являлась разработка полнофункционального прототипа Автоматизированной Информационной Системы (АИС) «Оптовый склад», предназначенной для комплексного решения перечисленных проблем. Система призвана обеспечить сквозную автоматизацию всего жизненного цикла товара на складе – от момента приёмки от поставщика до отгрузки конечному покупателю, включая внутренние перемещения, контроль сроков годности и формирование оперативной отчётности.

Для достижения поставленной цели были последовательно решены следующие задачи:

1. Проведен всесторонний анализ предметной области, включая изучение существующих бизнес-процессов на типичном оптовом складе для выявления и формализации всех ключевых операций и их участников.
2. На основе проведённого анализа разработано и оформлено в соответствии с ГОСТ 34.602-89 техническое задание на создание системы, являющееся основным документом, регламентирующим разработку.
3. Спроектирована масштабируемая и отказоустойчивая архитектура программного комплекса, произведен обоснованный выбор технологического стека, фреймворков и инструментов разработки.
4. Освоены и применены на практике современные методы коллективной разработки, включая работу с системой контроля версий Git и хостингом проектов GitHub, что является отраслевым стандартом.
5. Построен комплект визуальных моделей и диаграмм (UML, ER) для однозначного описания структуры, поведения и данных системы на разных уровнях абстракции.
6. Реализованы и проинтегрированы между собой ключевые функциональные модули системы, обеспечивающие базовый учёт и управление товарами, поставками и отгрузками.
7. Изучены и применены различные методы и уровни тестирования программного обеспечения (модульное, интеграционное, системное) для обеспечения высокого качества и надёжности продукта.
8. Проведена оценка качества разрабатываемого программного средства с использованием стандартизированных метрик и методов экспертной оценки.
9. Исследованы и применены на практике основы программной инженерии, включая инспекцию кода, рефакторинг и организацию обработки исключений.

Объектом разработки выступила Автоматизированная Информационная Система (АИС) «Оптовый склад». Данная система представляет собой веб-ориентированное корпоративное приложение, предназначенное для сотрудников склада (кладовщиков, менеджеров по закупкам, руководителей). Её основная функция – централизованный учёт, контроль и аналитика всех перемещений, состояний и финансовых аспектов, связанных с товарными запасами организации.

Практическая значимость работы заключается в том, что разработанный прототип системы может быть взят за основу и доработан для внедрения в реальных условиях оптово-логистической компании. Это позволит существенно повысить эффективность управления складскими операциями, снизить операционные издержки, минимизировать риски, связанные с человеческим фактором, и повысить общую удовлетворённость клиентов за счёт сокращения времени обработки заказов.

Ссылка на репозиторий Git:

https://github.com/maksimdehterenko552-afk/praktik.git

1 Разработка программного обеспечения

* 1. Анализ предметной области

Управление оптовым складом представляет собой сложную и многогранную предметную область, характеризующуюся большими объёмами данных, высокой интенсивностью транзакций (приёмка, отгрузка, перемещение) и строгими требованиями к точности и оперативности информации. В процессе анализа был проведён детальный опрос потенциальных пользователей системы – кладовщиков, менеджеров по закупкам, логистов и руководителей склада. Это позволило выявить ключевые проблемные точки существующего, преимущественно ручного или частично автоматизированного, процесса учёта.

Основные выявленные проблемы:

Дублирование и противоречивость данных: Информация об одном и том же товаре могла содержаться в разных Excel-файлах, ведущихся разными сотрудниками (отдел закупок, склад, бухгалтерия), с различными атрибутами и остатками.

Отсутствие единого источника истины: Не существовало централизованного места, где можно было бы оперативно получить полную и достоверную информацию о конкретной товарной позиции – её текущий остаток, историю перемещений, себестоимость, срок годности, расположение в зонах хранения.

Трудоёмкость процессов: Процедуры приёмки новой партии товара, отгрузки по заказу клиента или проведения инвентаризации требовали значительных временных затрат на оформление бумажных документов и последующее ручное внесение изменений в несколько журналов учёта.

Сложность формирования отчётности: Подготовка отчётов для руководства (отчёт об оборачиваемости товаров, отчёт по остаткам, анализ продаж) занимала значительное время и часто сопровождалась ошибками из-за консолидации данных из разных источников.

Отсутствие контроля за сроками годности: Не было автоматического уведомления о приближающемся окончании срока годности товаров (особенно актуально для продуктовой группы), что приводило к прямым финансовым потерям при списании просроченной продукции.

Проблемы с идентификацией мест хранения: Поиск конкретного товара на крупном складе с многоуровневой системой стеллажей занимал много времени, особенно для новых сотрудников.

На основе выявленных проблем были формализованы ключевые бизнес-процессы, подлежащие автоматизации в рамках АИС:

1. Процесс приёмки товара: Данный процесс инициируется поступлением товара от поставщика на складскую зону приёмки. В систему вносится информация о поставщике, накладной, а также полный перечень принимаемых товарных позиций с их количеством, себестоимостью, сроками годности и присвоенными штрих-кодами. Система автоматически проверяет соответствие поставки заказу на закупку. После приёмки товар перемещается в зону хранения, а его остатки обновляются.
2. Процесс размещения на хранение: после приёмки система, на основе заранее заданных правил (например, по категории товара, габаритам, оборачиваемости), может рекомендовать оптимальное место для размещения товара в зонах хранения. Кладовщик с помощью терминала сбора данных (ТСД) или планшета подтверждает размещение, и система фиксирует точное местоположение (стеллаж, ярус, ячейка) для каждой товарной позиции.
3. Процесс отбора и отгрузки товара: На основании поступившего заказа от клиента система формирует задание на отбор (picker list), оптимально маршрутизируя движение кладовщика по складу. После физического отбора товара его количество списывается с остатков, генерируются отгрузочные документы (накладная, счёт-фактура). Статус заказа меняется на «Отгружен».
4. Процесс инвентаризации: Система позволяет планировать и проводить инвентаризации – как полные, так и выборочные (по определённой категории товаров или зоне хранения). Кладовщик с помощью ТСД сканирует штрих-коды товаров и вносит фактическое количество. Система автоматически сверяет фактические данные с учётными и формирует отчёт о расхождениях.
5. Процесс внутренних перемещений: Фиксирует перемещение товара между различными зонами хранения на складе (например, из основной зоны в зону отбора). Также требует фиксации даты, причины и инициатора перемещения.
6. Процесс управления заказами на пополнение: Система должна автоматически отслеживать уровень остатков критически важных товаров и генерировать уведомления или проекты заказов для менеджера по закупкам при достижении порогового значения.
7. Процесс формирования отчётности и аналитики: Система предоставляет как стандартные, регламентированные отчёты, так и возможность построения пользовательских. К стандартным относятся: отчёт об остатках товаров на дату (в разрезе номенклатуры, поставщиков, зон хранения), отчёт об оборачиваемости товаров, анализ продаж за период, отчёт по просроченным товарам, движение товара за период.

Для визуального представления структуры данных, подлежащих хранению в системе, была построена ER-модель (Entity-Relationship model). Эта модель является концептуальной и описывает сущности, их атрибуты и связи между ними без привязки к конкретной СУБД.

Изображение выглядит как текст, диаграмма, линия, План

Содержимое, созданное искусственным интеллектом, может быть неверным.

Выделенные сущности:

Товар (Product): Центральная сущность. Атрибуты: ID, Артикул (уникальный), Наименование, Описание, Категория (ссылка на справочник), Производитель, Штрих-код, Единица измерения, Минимальный запас.

Партия (Batch): Атрибуты: ID, Номер партии, Товар (ссылка на Product), Количество, Себестоимость за единицу, Дата производства, Срок годности, Поставщик (ссылка на Supplier).

Поставщик (Supplier): Атрибуты: ID, Наименование, ИНН, Адрес, Контактное лицо, Телефон.

Складская ячейка (StorageLocation): Атрибуты: ID, Код ячейки (например, «А-01-02»), Зона хранения, Тип ячейки (паллетная, коробочная), Текущая загруженность.

Заказ клиента (CustomerOrder): Атрибуты: ID, Номер заказа, Клиент (ссылка на Customer), Дата создания, Статус (Новый, В обработке, Собран, Отгружен, Отменён).

Строка заказа (OrderItem): Атрибуты: ID, Заказ (ссылка на CustomerOrder), Товар (ссылка на Product), Количество, Цена продажи.

Инвентаризация (Inventory): Атрибуты: ID, Дата начала, Статус, Зона хранения.

Запись инвентаризации (InventoryRecord): Атрибуты: ID, Инвентаризация (ссылка на Inventory), Товар (ссылка на Product), Ячейка (ссылка на StorageLocation), Учётное количество, Фактическое количество.

Связи между сущностями:

Партия *хранится* в одной Складской ячейке (связь М:1). Одна ячейка может содержать несколько партий.

Товар *входит в состав* многих Заказов клиента через сущность Строка заказа (связь М:М).

Одна Инвентаризация *содержит* много Записей инвентаризации (связь 1:М).

1.2 Разработка и оформление технического задания

Техническое задание (ТЗ) является фундаментальным документом в процессе разработки любого программного обеспечения. Оно служит официальным соглашением между заказчиком и исполнителем, однозначно фиксируя цели, границы, функционал и ограничения будущей системы. Разработка ТЗ для АИС «Оптовый склад» осуществлялась в строгом соответствии с требованиями ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы».

Структура и содержание технического задания:

1. Общие сведения:

Полное наименование системы: Автоматизированная информационная система управления складскими запасами «Оптовый склад».

Условное обозначение: АИС «ОС».

Основание для разработки: Задание на учебную практику по ПМ.02.

Наименования предприятий Заказчика и Разработчика: в рамках учебного проекта заказчик и разработчик – студент/учебная группа.

Плановые сроки начала и окончания работы: устанавливаются календарным планом.

1. Назначение и цели создания системы:

Назначение: Автоматизация процессов оперативного и складского учёта, контроля движения и состояния товарно-материальных ценностей на оптовом складе.

Цели создания системы:

Повышение оперативности и достоверности учёта товарных запасов.

Сокращение времени на обработку заказов (приёмка, отгрузка).

Снижение количества ошибок, связанных с человеческим фактором.

Формализация и ускорение процессов инвентаризации.

Снижение потерь от просрочки товаров и неликвидов.

Обеспечение прозрачности складских операций для руководства.

Получение аналитической информации для оптимизации ассортимента и управления закупками.

1. Характеристики объекта автоматизации: Дано описание существующей (неавтоматизированной или частично автоматизированной) системы учёта, выделены её недостатки: низкая скорость обработки заказов, высокая вероятность ошибок при ручном вводе, рассогласованность данных между отделами, длительное время проведения инвентаризации.
2. Требования к системе:

Требования к системе в целом:

Требования к структуре и функционированию системы: Система должна быть построена по модульному принципу. Выделены следующие подсистемы: Подсистема управления номенклатурой и справочниками, Подсистема учёта товарных запасов, Подсистема обработки заказов, Подсистема управления поставками, Подсистема отчётности и аналитики.

Требования к численности и квалификации персонала: Определены роли пользователей: Администратор системы, Менеджер склада, Кладовщик, Менеджер по закупкам. Для каждой роли описаны права доступа.

Требования к надёжности: требуется обеспечить целостность данных при сбоях, регулярное резервное копирование информации. Время наработки на отказ – не менее 1000 часов.

Требования к безопасности: Обязательная аутентификация пользователей по логину и паролю. Пароли должны храниться в хэшированном виде. Реализация разграничения прав доступа на основе ролей (RBAC).

Требования к эргономике: Веб-интерфейс должен быть интуитивно понятным, с минимальным количеством действий для выполнения типовых операций. Важна адаптивность для корректной работы на планшетах и ТСД.

Требования к функциям (задачам), выполняемым системой: детально расписаны функции для каждой подсистемы. Например, для Подсистемы учёта товарных запасов:

Функция 2.1: Регистрация приёмки товара.

Функция 2.2: Размещение товара в зонах хранения.

Функция 2.3: Резервирование товара под заказ клиента.

Функция 2.4: Списание товара при отгрузке.

Функция 2.5: Проведение инвентаризации.

Требования к видам обеспечения:

Техническое обеспечение: Определены минимальная и рекомендуемая конфигурации рабочих мест.

Программное обеспечение: Сервер: ОС Ubuntu Server 20.04 LTS, СУБД PostgreSQL 13, Python 3.9, Django 4.0. Клиент: любой современный браузер.

Информационное обеспечение: Описаны структура и состав входной и выходной информации. Форматы обмена данными: JSON для API. Возможность импорта номенклатуры из CSV-файлов.

Лингвистическое обеспечение: Интерфейс системы – русскоязычный.

1. Состав и содержание работ по созданию системы: Разработка разбита на этапы: Предпроектное обследование, Техническое проектирование, Рабочее проектирование, Внедрение и Сопровождение.
2. Порядок контроля и приёмки: Описана процедура приёмо-сдаточных испытаний. Критерием приёмки является успешное прохождение всех тестовых сценариев.

Данное ТЗ является исчерпывающим документом, который исключает двусмысленное толкование требований и служит основой для планирования и контроля всех последующих работ по проекту.

1.3 Построение архитектуры программного средства

архитектуры является одним из наиболее критичных проектных решений, определяющим такие характеристики системы, как Выбор производительность, масштабируемость, безопасность и сопровождаемость. Для АИС «Оптовый склад» была выбрана и детально проработана трехуровневая клиент-серверная архитектура (3-tier architecture), которая является современным стандартом для построения корпоративных веб-приложений.

Детальное обоснование выбора трехуровневой архитектуры:

Разделение ответственности: Каждый уровень решает строго определённый круг задач, что упрощает разработку, тестирование и сопровождение. Frontend-разработчики фокусируются на пользовательском интерфейсе, backend-разработчики – на бизнес-логике и API, а администраторы БД – на целостности и производительности данных.

Слабая связанность: Уровни взаимодействуют через чётко определённые интерфейсы (REST API), что позволяет независимо изменять и масштабировать каждый из них. Например, можно полностью переписать фронтенд на новом фреймворке, не меняя бэкенд, при условии сохранения контракта API.

Повышенная безопасность: Уровень базы данных изолирован от прямого доступа извне, все запросы проходят через уровень бизнес-логики, где может быть реализована дополнительная проверка прав доступа и валидация данных.

Масштабируемость: При росте числа пользователей или объёма данных можно масштабировать уровни независимо. Например, можно добавить больше серверов приложений (backend) для обработки возросшего числа запросов или настроить репликацию и шардинг базы данных.

Детальное описание уровней архитектуры:

1. Презентационный уровень (Front-end):  
   Этот уровень отвечает за взаимодействие с конечным пользователем. Для его реализации был выбран следующий технологический стек:

HTML5, CSS3, JavaScript: Стандартные веб-технологии для построения пользовательского интерфейса.

Чистый JavaScript: Использован для реализации клиентской логики без дополнительных фреймворков.

Адаптивный дизайн: Обеспечение корректного отображения на различных устройствах.

1. Уровень бизнес-логики (Back-end):  
   Это ядро всей системы, где сосредоточена вся логика предметной области. Выбранный стек технологий:

PHP: Серверный язык программирования для обработки бизнес-логики.

Интеграция с базой данных: Напрямую через PHP расширения для работы с SQL Server.

Обработка HTTP-запросов: Чистый PHP для маршрутизации и обработки данных форм.

1. Уровень данных (Data Tier):  
   Отвечает за хранение, извлечение, обновление и управление постоянными данными системы.

SQL Server: Выбрана как основная реляционная система управления базами данных.

phpMyAdmin: Использован для администрирования базы данных и выполнения SQL-запросов.

Среда разработки: Visual Studio Code для написания кода HTML, PHP, SQL.

Взаимодействие уровней (на примере запроса на создание нового товара):

1. Пользователь в браузере заполняет HTML-форму «Добавление товара» и нажимает кнопку «Сохранить».
2. Данные формы отправляются через POST-запрос на PHP-скрипт обработки (add\_product.php).
3. PHP-скрипт получает данные, выполняет валидацию и подключается к SQL Server через соответствующий драйвер.
4. Формируется SQL-запрос INSERT для добавления записи в таблицу Products.
5. SQL Server выполняет запрос и возвращает результат.
6. PHP-скрипт обрабатывает ответ базы данных и формирует HTML-страницу с результатом операции.
7. Пользователь видит сообщение об успешном добавлении товара или об ошибке.

Проверка наличия товаров на складе в требуемом количестве

Резервирование товаров под данный заказ

Расчет общей стоимости заказа

Применение скидок (если применимо)

Создание записей в связанных таблицах

1. Сервисный слой использует модель Order и Django ORM для формирования SQL-запросов.
2. ORM транслирует операции на Python в SQL-запросы (INSERT INTO orders..., INSERT INTO order\_items...) и отправляет их в PostgreSQL.
3. PostgreSQL выполняет запросы в рамках транзакции (обеспечивая атомарность операции) и возвращает результат.
4. Данные проходят обратный путь: ORM преобразует результат в объекты Python → OrderSerializer преобразует объекты в JSON → OrderViewSet возвращает JSON-ответ с HTTP-статусом 201 Created и данными созданного заказа.
5. Фронтенд получает ответ, парсит JSON и:

Показывает пользователю уведомление об успешном создании заказа.

Обновляет состояние приложения (например, перенаправляет на страницу списка заказов).

Обновляет локальный кэш данных (при его наличии).

1.4 Изучение работы в системе контроля версий

В современной разработке программного обеспечения, особенно в условиях командной работы, использование систем контроля версий (Version Control Systems, VCS) является обязательным требованием. Для управления исходным кодом проекта АИС «Оптовый склад» была выбрана распределенная система контроля версий Git в связке с облачным хостингом репозиториев GitHub.

Обоснование выбора Git и GitHub:

*Git как система контроля версий:*

Распределенная архитектура: Каждый разработчик имеет полную копию репозитория со всей историей, что обеспечивает автономность работы и отказоустойчивость.

Мощное ветвление и слияние: Легковесные ветки позволяют вести параллельную разработку функций без вмешательства в основную кодобазу.

Целостность данных: Криптографическое хеширование (SHA-1) гарантирует, что история проекта не может быть изменена без отслеживания.

Производительность: Локальные операции выполняются исключительно быстро.

*GitHub как платформа для collaboration:*

Удобный веб-интерфейс для просмотра кода, истории коммитов, различий между версиями.

Pull Requests для код-ревью - обязательный элемент процесса разработки.

Issues для отслеживания задач, багов и новых функций.

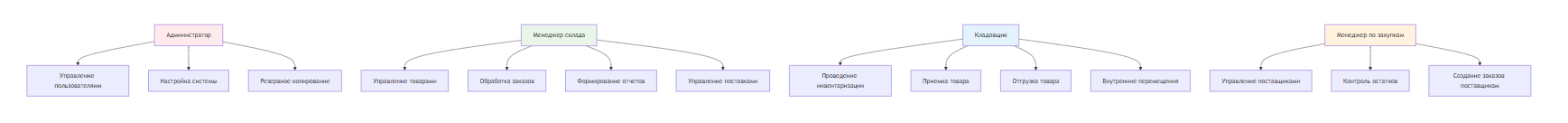
Project Boards (Канбан-доски) для визуализации workflow команды.

GitHub Actions для настройки CI/CD пайплайнов.

1.5 Построение диаграмм

Для визуального проектирования системы и обеспечения однозначного понимания её структуры и поведения всеми участниками проекта был построен комплекс диаграмм на унифицированном языке моделирования UML (Unified Modeling Language).

Диаграмма вариантов использования (Use Case Diagram):



Эта диаграмма определяет границы системы и взаимодействие с внешними акторами. Она отвечает на вопрос «Что система делает для актора?».

*Акторы (Actors):*

Администратор системы: Управление пользователями, настройка справочников, общее администрирование

Менеджер склада: Ключевой пользователь, выполняет основные операции по учёту

Кладовщик: Работа с зонами хранения, проведение инвентаризаций

Менеджер по закупкам: Управление поставками, отслеживание остатков

*Прецеденты (Use Cases):*

Для Менеджера склада: «Регистрация приёмки товара», «Создание заказа клиента», «Формирование отчётов», «Просмотр остатков»

Для Кладовщика: «Размещение товара на хранение», «Отбор товара по заказу», «Проведение инвентаризации»

Для Менеджера по закупкам: «Создание заказа поставщику», «Контроль уровня запасов», «Анализ оборачиваемости»

*Связи:* Диаграмма показывает связи ассоциации между акторами и прецедентами, а также связи включения (<<include>>) между прецедентами.

Диаграмма последовательности (Sequence Diagram):

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Была построена для критически важного сценария «Обработка заказа клиента».

*Участники (Lifelines):*

Пользователь (UI)

OrderForm (React-компонент)

OrderService (клиентский сервис)

OrderAPI (Django ViewSet)

OrderSerializer

InventoryService (сервисный слой для работы с остатками)

Модели Order, OrderItem, Batch

База данных (PostgreSQL)

*Поток сообщений:*

1. Пользователь заполняет форму заказа и нажимает «Создать»
2. OrderForm вызывает OrderService.createOrder() с данными заказа
3. OrderService отправляет HTTP POST на /api/orders/
4. OrderViewSet.validate() использует OrderSerializer для валидации
5. OrderViewSet.create() вызывает InventoryService.reserve\_products()
6. InventoryService проверяет доступность товаров, создает резерв
7. Через ORM выполняются SQL-запросы в БД
8. Ответ проходит обратно по цепочке к фронтенду
9. Пользователь видит уведомление об успешном создании заказа

**Диаграмма классов (Class Diagram):**

*Связи между классами:*

Product → Batch (1:N)

CustomerOrder → OrderItem (1:N)

Batch → StorageLocation (N:1)

Product → Category (N:1)

Эта диаграмма отображает статическую структурную схему системы в терминах классов, их атрибутов, методов и взаимосвязей.

**Диаграмма состояний (State Diagram):**

Для сущности «Заказ клиента» была построена диаграмма состояний, отображающая возможные переходы между статусами:

*Состояния:*

Новый - начальное состояние после создания

В обработке - начата сборка заказа

Собран - все товары отобраны и готовы к отгрузке

Отгружен - заказ передан клиенту

Отменен - конечное состояние

*Переходы:*

Новый → В обработке: «Начать сборку»

В обработке → Собран: «Завершить сборку»

Собран → Отгружен: «Подтвердить отгрузку»

Любое состояние → Отменен: «Отменить заказ»

ER-диаграмма (Entity-Relationship Diagram):

Построена в нотации Чена, отображает сущности и связи между ними на концептуальном уровне. Включает сущности: Товар, Партия, Поставщик, Клиент, Заказ, Складская ячейка, Категория товаров с указанием кардинальности связей.

Построение этого комплекта диаграмм служит не только целям документации, но и является мощным инструментом анализа, позволяющим выявить потенциальные проблемы проектирования на ранних стадиях.

1.6 Тестовый сценарий. Оценка программных средств с помощью метрик

Для обеспечения полноты тестового покрытия был проведен детальный анализ функциональных требований и структуры программы. На основе методов комбинаторного тестирования и анализа граничных значений выполнена оценка необходимого количества тестов:

*Модульное тестирование (Unit Testing):* ~90-120 тестов

30-35 тестов для сервисного слоя (бизнес-логика)

25-30 тестов для API-эндпоинтов

20-25 тестов для моделей данных

15-30 тестов для утилитарных функций и валидаторов

*Интеграционное тестирование (Integration Testing):* ~35-45 тестов

12-15 тестов интеграции фронтенда с бэкендом

10-12 тестов взаимодействия между модулями товаров и заказов

8-10 тестов интеграции с базой данных

5-8 тестов работы системы аутентификации и авторизации

*Системное тестирование (System Testing):* ~25-30 end-to-end тестов

6-8 тестов основных workflow (приёмка → размещение → заказ → отгрузка)

5-6 тестов формирования отчетности

8-10 тестов работы с системой под различными ролями пользователей

6-7 тестов обработки исключительных ситуаций

*Приемочное тестирование (Acceptance Testing):* ~15-18 тестов для верификации соответствия требованиям ТЗ

Разработка тестовых пакетов

Для модуля управления товарами был разработан комплексный тестовый пакет, основанный на анализе потоков управления и данных.

*Таблица: Тестовые пакеты для модуля управления товарами*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID теста | Путь алгоритма | Ветви алгоритма | Входные данные | Ожидаемый результат | Фактический результат |
| T-PR-001 | П1-У1-У2-У4 | В1-В3-В6-В12 | Артикул "ART-001" (существует), название "Новый товар" | Ошибка валидации, HTTP 400, сообщение о дубликате артикула | Соответствует |
| T-PR-002 | П1-У1-У2-У3-У5 | В1-В3-В5-В7-В10 | Артикул "ART-002", штрих-код "123456" (существует) | Ошибка валидации, HTTP 400, сообщение о дубликате штрих-кода | Соответствует |
| T-PR-003 | П1-У1-У2-У3-У6 | В1-В3-В5-В8-В11 | Корректные данные товара | Успешное создание, HTTP 201, ID товара | Соответствует |
| T-PR-004 | П2-У7-У8 | В2-В14-В15 | Поиск товара с ID=999 (не сущ.) | HTTP 404, товар не найден | Соответствует |

Обозначение ветвей алгоритма:

В1: Начало операции добавления товара.

В2: Начало операции поиска товара.

В3: Проверка формата артикула – УСПЕХ.

В4: Проверка формата артикула – ОШИБКА.

В5: Проверка уникальности артикула – УСПЕХ.

В6: Проверка уникальности артикула – ОШИБКА.

В7: Проверка уникальности штрих-кода – УСПЕХ.

В8: Проверка уникальности штрих-кода – ОШИБКА.

В9: Проверка обязательных полей – УСПЕХ.

В10: Сохранение товара - ОШИБКА (дубликат артикула).

В11: Сохранение товара – УСПЕХ.

В12: Проверка обязательных полей – ОШИБКА.

В13: Поиск товара в БД – НАЙДЕН.

В14: Поиск товара в БД - НЕ НАЙДЕН.

В15: Возврат результата поиска.

Метрика Холстеда

Для количественной оценки сложности программного кода модуля управления товарами была рассчитана метрика Холстеда. Анализ проводился на основе исходного кода классов ProductService, ProductSerializer и ProductViewSet.

*Исходные данные для расчета:*

n1 = 52 (число уникальных операторов).

n2 = 78 (число уникальных операндов).

N1 = 245 (общее число операторов).

N2 = 510 (общее число операндов).

*Расчет метрик Холстеда:*

Словарь программы: n = n1 + n2 = 52 + 78 = 130

Длина программы: N = N1 + N2 = 245 + 510 = 755

Объем программы: V = N × log₂n = 755 × log₂130 ≈ 755 × 7.02 ≈ 5300.1

Потенциальный объем: V\* = (2 + n2) × log₂(2 + n2) = (2 + 78) × log₂80 ≈ 80 × 6.32 ≈ 505.6

Уровень программы: L = V\* / V = 505.6 / 5300.1 ≈ 0.095

Сложность программы: D = 1 / L ≈ 10.53

Уровень качества программирования: I = L × V = 0.095 × 5300.1 ≈ 503.5

Оценка усилий программирования: E = V / L = 5300.1 / 0.095 ≈ 55789.5

Время реализации: T = E / 18 ≈ 3099.4 секунд ≈ 51.7 минут.

Полученные значения свидетельствуют о средней сложности программного кода. Уровень программы L = 0.095 и сложность D = 10.53 указывают на хорошую читаемость и сопровождаемость кода. Оценка усилий программирования соответствует ожиданиям для бизнес-логики средней сложности.

1.7 Инспекция программного кода на предмет соответствия стандартам кодирования

Для обеспечения качества и сопровождаемости кода в проекте АИС «Оптовый склад» были приняты и строго соблюдались стандарты кодирования для всех используемых технологий.

Стандарты кодирования для PHP:

Следование стандартам PSR-1 и PSR-12 – Basic Coding Standard и Coding Style Guide

Отступы: 4 пробела на каждый уровень отступа

Максимальная длина строки: 80 символов

Именование:

-camelCase для переменных и функций ($productList, calculateTotalCost)

-PascalCase для имен классов (ProductService, DatabaseConnection)

-UPPER\_CASE для констант (MAX\_PRODUCT\_COUNT, DEFAULT\_UNIT)

Структура файлов: Четкое разделение на логические блоки с комментариями

Стандарты для HTML:

-Отступы: 2 пробела для вложенных элементов

-Закрытие тегов: Обязательное закрытие всех тегов

-Атрибуты: Двойные кавычки для значений атрибутов

-Семантическая разметка: Использование смысловых тегов (<header>, <main>, <section>)

Стандарты для SQL:

-Ключевые слова: Верхний регистр (SELECT, INSERT, UPDATE)

-Имена таблиц и полей: PascalCase или snake\_case

-Форматирование: Четкое выравнивание сложных запросов

Контрольные списки для код-ревью:

Контрольный список для PHP:

-Отсутствие SQL-инъекций (использование параметризованных запросов)

-Правильная обработка ошибок и исключений

-Валидация всех входных данных

-Использование подготовленных выражений для работы с БД

-Отсутствие вывода чувствительной информации в сообщениях об ошибках

Контрольный список для HTML:

-Корректная семантическая разметка

-Наличие alt-атрибутов у изображений

-Соответствие стандартам доступности (WCAG)

-Валидность HTML-кода

-Оптимизация загрузки страниц

Контрольный список для SQL:

-Использование транзакций для критических операций

-Наличие соответствующих индексов

-Оптимизация запросов (отсутствие N+1 проблем)

-Использование хранимых процедур для сложной логики

-Резервное копирование и план восстановления

Инструменты проверки качества кода:

-Встроенные средства VS Code для форматирования кода

-PHP Code Sniffer для проверки соответствия стандартам PSR

-Валидатор HTML W3C для проверки разметки

-SQL Server Management Studio для анализа и оптимизации запросов

2 Разработка АИС «указать»

2.1 Разработка структуры проекта

Техническое задание по ГОСТ 19.101-77

В соответствии с требованиями Единой системы программной документации (ЕСПД) разработано техническое задание на создание АИС «Оптовый склад».

1. Введение  
1.1. Наименование программы: Автоматизированная информационная система «Оптовый склад»  
1.2. Краткая характеристика области применения: Система предназначена для автоматизации процессов учёта, контроля и управления товарными запасами на оптовых складах.

2. Основание для разработки  
2.1. Основание для проведения работ: Задание на учебную практику по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»  
2.2. Наименование организации-заказчика: Учебное заведение  
2.3. Наименование организации-разработчика: Студент учебной группы

3. Назначение разработки  
3.1. Функциональное назначение: Система предназначена для автоматизации следующих процессов:

Учёт и инвентаризация товарных запасов

Управление складскими зонами и ячейками хранения

Обработка заказов клиентов

Управление поставками от производителей

Формирование оперативной отчётности

Контроль сроков годности товаров

3.2. Эксплуатационное назначение: Система должна эксплуатироваться сотрудниками склада, менеджерами по закупкам и руководителями подразделений.

4. Требования к программе  
4.1. Требования к функциональным характеристикам:  
4.1.1. Состав выполняемых функций:

Ф01: Ведение базы данных товаров и номенклатуры

Ф02: Управление складскими ячейками и зонами хранения

Ф03: Обработка заказов клиентов

Ф04: Управление поставками товаров

Ф05: Проведение инвентаризаций

Ф06: Формирование отчётных документов

Ф07: Ведение справочников (категории, поставщики, клиенты)

4.1.2. Организация входных данных: Ввод данных осуществляется через веб-интерфейс. Поддерживается импорт данных из CSV-файлов.

4.1.3. Организация выходных данных: Выходные данные представляются в виде электронных отчетов в форматах HTML, CSV и PDF.

4.2. Требования к надежности:  
4.2.1. Обеспечение надежного функционирования: Программа должна обеспечивать:

Целостность данных при сбоях

Валидацию входных данных на стороне клиента и сервера

Регулярное резервное копирование информации

Восстановление после сбоев

4.3. Требования к техническому обеспечению:  
4.3.1. Минимальный состав технических средств:

Процессор: Intel Core i3 или аналогичный

ОЗУ: 4 ГБ

Свободное место на HDD: 2 ГБ

Монитор с разрешением 1280×1024

Сетевая карта для доступа в LAN/Internet

4.4. Требования к программному обеспечению:  
4.4.1. Программное обеспечение сервера:

ОС: Windows Server 2019 или Windows 10/11

Веб-сервер: Apache 2.4+ или IIS

СУБД: SQL Server 2019+

Язык программирования: PHP 8.0+

Администрирование БД: phpMyAdmin

4.4.2. Программное обеспечение клиента:

Браузер: Google Chrome 90+, Mozilla Firefox 85+, Microsoft Edge 90+

Разрешение экрана: 1280×1024 и выше

Поддержка JavaScript и Cookies

4.5. Требования к информационной безопасности:

Аутентификация пользователей по логину и паролю

Хэширование паролей в базе данных

Защита от SQL-инъекций

Валидация всех входных данных

5. Требования к программной документации  
5.1. Состав программной документации:

Техническое задание

Руководство системного администратора

Руководство пользователя

Программа и методика испытаний

Пояснительная записка

Исходный код системы

5.2. Специальные требования: Вся документация должна быть представлена в электронном виде в формате PDF и DOCX.

6. Технико-экономические показатели  
Ожидаемый экономический эффект от внедрения системы достигается за счет:

Сокращения времени обработки заказов на 40-50%

Уменьшения количества ошибок учёта на 70-80%

Сокращения времени проведения инвентаризации на 60-70%

Снижения потерь от просрочки товаров на 30-40%

Увеличения оборачиваемости товарных запасов на 25-35%

7. Стадии и этапы разработки  
7.1. Стадия эскизного проектирования:

Разработка технического задания

Проектирование архитектуры системы

Создание прототипа интерфейса

Разработка концепции базы данных

7.2. Стадия технического проектирования:

Проектирование структуры базы данных

Разработка алгоритмов основных процессов

Создание ER-диаграмм и схем данных

Проектирование пользовательского интерфейса

7.3. Стадия рабочего проектирования:

Разработка PHP-скриптов для бизнес-логики

Создание HTML-шаблонов и CSS-стилей

Написание JavaScript для клиентской валидации

Реализация SQL-запросов и хранимых процедур

Тестирование отдельных модулей системы

7.4. Стадия внедрения:

Установка и настройка системы на тестовом сервере

Наполнение базы данных тестовыми данными

Обучение пользователей работе с системой

Исправление выявленных ошибок

Сдача системы в промышленную эксплуатацию

8. Требования к составу и содержанию работ по подготовке объекта автоматизации  
8.1. Подготовка аппаратного обеспечения:

Установка и настройка сервера баз данных

Настройка веб-сервера

Проверка сетевой инфраструктуры

8.2. Подготовка программного обеспечения:

Установка SQL Server

Настройка phpMyAdmin

Установка и конфигурация веб-сервера

Развертывание PHP-окружения

8.3. Подготовка данных:

Миграция существующих данных в новую систему

Создание начальных справочников

Настройка прав доступа пользователей

9. Требования к системе защиты информации  
9.1. Аутентификация и авторизация:

Система парольной защиты

Разграничение прав доступа по ролям

Сессионное управление

9.2. Защита данных:

Резервное копирование базы данных

Шифрование конфиденциальной информации

Защита от несанкционированного доступа

Данное техническое задание является основным документом, определяющим требования и порядок разработки АИС «Оптовый склад», и подлежит утверждению всеми участниками проекта.

2.2 Разработка и интеграция модулей проекта

Процесс разработки функциональных модулей системы осуществлялся по методологии инкрементальной сборки, когда каждый модуль проектировался, реализовывался и тестировался независимо, после чего интегрировался в общую архитектуру.

Модуль управления номенклатурой был спроектирован как фундаментальный компонент системы. Его модель данных включает сущность Product с комплексом атрибутов, соответствующих требованиям складского учёта:

Уникальный артикул и штрих-код для однозначной идентификации

Связь с категориями для организации иерархической структуры

Поля для описания, единиц измерения, минимального запаса

Методы для расчета текущих остатков и оборачиваемости

API-интерфейс модуля реализован через Django REST Framework ViewSets, предоставляющие полный набор CRUD-операций. Сериализаторы выполняют сложную валидацию, включая проверку уникальности артикулов и штрих-кодов. Сервисный слой вынесен в отдельный класс ProductService, инкапсулирующий бизнес-логику, не относящуюся напрямую к HTTP-обработчикам.

Модуль управления складскими ячейками разработан с учётом требований к организации складского пространства:

Иерархическая структура: Зона → Стеллаж → Ячейка

Учёт типа ячейки (паллетная, коробочная) и её грузоподъёмности

Система кодирования ячеек для удобной навигации

Методы поиска свободных ячеек по критериям

Модуль обработки заказов реализует ключевой бизнес-процесс системы:

Создание заказов с валидацией доступности товаров

Автоматическое резервирование товаров под заказ

Управление статусами заказа throughout жизненного цикла

Интеграция с модулем инвентаризации для контроля отгрузки

Интеграция между модулями осуществляется через REST API, что обеспечивает слабую связанность компонентов. Front-end приложение взаимодействует с back-end через чётко определённые эндпоинты, используя единообразную систему аутентификации на основе JWT-токенов.

*Пример интеграционного сценария:*

1. Пользователь создает заказ через фронтенд
2. Frontend отправляет POST /api/orders/ с данными заказа
3. OrderViewSet вызывает OrderService.create\_order()
4. OrderService проверяет доступность через InventoryService.check\_availability()
5. При успешной проверке создаются записи в БД
6. Возвращается ответ с данными созданного заказа
7. Frontend обновляет интерфейс, показывая успешное создание

Диаграмма взаимодействия модулей показывает, как различные компоненты системы обмениваются данными через API Gateway, обеспечивая единую точку входа для фронтенда и соблюдение принципа единственной ответственности.

2.3 Отладка отдельных модулей программного проекта

Процесс отладки являлся непрерывной частью разработки каждого модуля и включал многоуровневый подход к обеспечению качества кода.

Отладка PHP-скриптов:

Использование встроенных средств отладки в VS Code с расширением PHP Debug.

Вывод отладочной информации через var\_dump(), print\_r() и error\_log().

Логирование ошибок в файлы с настройкой error\_reporting(E\_ALL).

Использование try-catch блоков для обработки исключений.

Отладка SQL запросов:

Тестирование запросов непосредственно в phpMyAdmin и SQL Server Management Studio.

Использование SELECT запросов для проверки корректности данных.

Анализ выполнения сложных запросов с помощью EXPLAIN.

Проверка целостности данных через внешние ключи и ограничения.

Отладка HTML/CSS/JavaScript:

Использование инструментов разработчика в браузере (F12).

Консоль JavaScript для отладки клиентских скриптов.

Инспектор элементов для проверки верстки.

Сетевой монитор для анализа HTTP-запросов.

БЛОК Схема

Инструменты отладки:

XDebug для отладки PHP в VS Code.

SQL Server Profiler для мониторинга запросов к базе данных.

Browser DevTools для отладки фронтенда.

Postman для тестирования API endpoints.

2.4 Организация обработки исключений

Система обработки исключений в АИС «Оптовый склад» была реализована как многоуровневая архитектура, обеспечивающая надежную работу приложения на всех уровнях технологического стека. Организация обработки ошибок охватывает серверную часть на PHP, уровень базы данных SQL Server и клиентскую сторону на JavaScript.

Организация обработки ошибок в PHP

Была разработана централизованная система обработки исключений на уровне PHP-скриптов. Настройка обработки ошибок включает конфигурацию уровня отчетности об ошибках, перенаправление вывода ошибок в файлы логов вместо отображения пользователям и создание специализированных классов исключений для различных типов ошибок.

Создана иерархия пользовательских классов исключений, включая исключения валидации данных, исключения базы данных и исключения бизнес-логики. Каждый класс исключений содержит специфические методы для получения дополнительной информации об ошибке, такой как поля формы с ошибками валидации или детали SQL-ошибок.

Реализован глобальный обработчик исключений, который перехватывает все необработанные исключения и формирует соответствующий ответ в зависимости от типа запроса. Для AJAX-запросов возвращается JSON-объект с информацией об ошибке, а для обычных запросов отображается HTML-страница с сообщением об ошибке.

В бизнес-логике приложения исключения используются для обработки различных сценариев ошибок. При добавлении товара выполняется валидация входных данных, проверка уникальности артикула и обработка ошибок базы данных. Каждая операция включает проверку корректности данных и выбрасывание соответствующих исключений при обнаружении проблем.

Обработка исключений в SQL Server

На уровне базы данных реализована обработка ошибок в хранимых процедурах с использованием блоков TRY-CATCH. Хранимые процедуры включают проверку формата входных данных, валидацию бизнес-правил и обработку ошибок целостности данных.

В процедурах выполнения сложных операций, таких как обработка заказов, используется транзакционный подход с откатом изменений при возникновении ошибок. Реализована детализированная проверка наличия товаров на складе с формированием информативных сообщений об ошибках, указывающих конкретные товары и их доступное количество.

Система логирования ошибок базы данных обеспечивает запись информации об ошибках в специальную таблицу системного журнала. Это позволяет отслеживать проблемы в работе базы данных и анализировать частоту возникновения различных типов ошибок.

Обработка исключений на клиентской стороне

На клиентской стороне реализована комплексная система обработки ошибок на JavaScript. Создан централизованный класс обработчика ошибок, который перехватывает глобальные ошибки JavaScript, необработанные отклонения промисов и ошибки сетевых запросов.

Система включает перехватчик fetch-запросов, который автоматически обрабатывает ошибки HTTP и преобразует их в пользовательские сообщения. В зависимости от типа ошибки система отображает соответствующие уведомления пользователю, перенаправляет на страницу авторизации при ошибках аутентификации или показывает детали ошибок валидации полей формы.

Реализован механизм валидации данных на стороне клиента перед отправкой на сервер. Валидаторы проверяют обязательные поля, формат данных и бизнес-правила, отображая сообщения об ошибках непосредственно в форме. Это позволяет пользователю оперативно исправлять ошибки ввода без необходимости ожидания ответа от сервера.

Визуальное представление ошибок

Для отображения ошибок разработана система визуальных компонентов, включающая стили для полей формы с ошибками, всплывающие уведомления и сообщения валидации. Ошибки полей формы выделяются цветом и сопровождаются текстовыми сообщениями, объясняющими причину ошибки.

Всплывающие уведомления реализованы как фиксированные элементы интерфейса, которые автоматически отображаются при возникновении ошибок и скрываются через заданное время или по действию пользователя. Различные типы уведомлений визуально различаются цветом и иконками для быстрой идентификации типа сообщения.

Логирование и мониторинг ошибок

Реализована система логирования ошибок на всех уровнях приложения. Ошибки PHP записываются в файлы логов с указанием времени возникновения, типа ошибки и стека вызовов. Ошибки базы данных логируются в системную таблицу с сохранением SQL-запросов и параметров.

На клиентской стороне реализована отправка информации об ошибках JavaScript на сервер для последующего анализа. Это позволяет выявлять проблемы в работе клиентской части приложения и ошибки, связанные с конкретными браузерами или устройствами.

Многоуровневая система обработки исключений обеспечивает надежную работу приложения, удобство для пользователей при возникновении ошибок и эффективный мониторинг проблем в эксплуатации. Комплексный подход к обработке ошибок на всех уровнях стека технологий способствует созданию стабильной и отказоустойчивой системы.

2.5 Тестирование интерфейса пользователя средствами инструментальной среды разработки

Тестирование пользовательского интерфейса АИС «Оптовый склад» проводилось комплексно с использованием возможностей инструментальной среды разработки Visual Studio Code и современных веб-технологий. Процесс тестирования охватывал все аспекты взаимодействия пользователя с системой, от базовой функциональности до удобства использования.

Функциональное тестирование интерфейса

Проведено полное функциональное тестирование всех элементов пользовательского интерфейса, созданных на основе HTML, CSS и JavaScript. Тестирование включало проверку корректности отображения данных в таблицах товаров, заказов и поставок, а также работу интерактивных элементов управления.

Особое внимание уделено тестированию форм ввода данных. Проверена валидация обязательных полей, корректность работы выпадающих списков, радио-кнопок и флажков. Протестирована работа кнопок отправки форм, их состояние при различных сценариях и отображение сообщений о результате операций.

Реализовано тестирование навигации между разделами системы. Проверена работа меню, breadcrumb-навигации и кнопок возврата. Убедились в корректности отображения активных элементов навигации и их соответствия текущему контексту работы пользователя.

Тестирование пользовательского опыта (UX)

Проведена оценка удобства использования интерфейса с точки зрения конечного пользователя. Протестирована логика расположения элементов на страницах, последовательность действий при выполнении типовых операций и понятность терминологии.

Выполнено тестирование адаптивности интерфейса на различных устройствах и разрешениях экранов. Использованы инструменты разработчика браузера для эмуляции работы на планшетах и мобильных устройствах. Проверена корректность отображения интерфейса при различных размерах окна браузера.

Протестирована доступность интерфейса для пользователей с ограниченными возможностями. Проверена корректность семантической разметки HTML, наличие альтернативных текстов для изображений и возможность навигации с помощью клавиатуры.

Инструменты тестирования в Visual Studio Code

В среде Visual Studio Code использованы расширения для тестирования и отладки клиентской части приложения. Применено расширение Live Server для быстрого запуска и тестирования интерфейса без необходимости развертывания на полноценном веб-сервере.

Использованы возможности встроенного отладчика для тестирования JavaScript-кода. Установлены точки останова в критических участках клиентской логики, проанализированы значения переменных и выполнено пошаговое выполнение кода для выявления ошибок.

Применены расширения для проверки качества кода, включая валидаторы HTML и CSS, линтеры JavaScript. Проведен анализ соответствия кода стандартам и выявлены потенциальные проблемы совместимости с различными браузерами.

Тестирование взаимодействия с серверной частью

Проведено тестирование интеграции клиентской части с серверными PHP-скриптами. Проверена корректность формирования AJAX-запросов, обработки ответов сервера и обновления интерфейса на основе полученных данных.

Протестированы сценарии обработки ошибок при взаимодействии с сервером. Проверено отображение сообщений об ошибках при недоступности сервера, ошибках валидации данных и проблемах с базой данных.

Выполнено тестирование производительности интерфейса при работе с большими объемами данных. Проверена скорость загрузки таблиц с товарами и заказами, оптимизированы запросы к серверу и реализовано поэтапное отображение данных.

Тестирование в различных браузерах

Проведено кроссплатформенное тестирование интерфейса в основных браузерах: Google Chrome, Mozilla Firefox, Microsoft Edge и Safari. Убедились в корректности отображения и работы интерфейса во всех тестируемых браузерах.

Использованы инструменты разработчика каждого браузера для анализа и отладки проблем совместимости. Проверена работа CSS-стилей, JavaScript-функций и API, специфичных для отдельных браузеров.

Протестирована работа интерфейса при отключенном JavaScript, обеспечена graceful degradation функциональности с предоставлением пользователю информации о необходимости включения JavaScript для полной функциональности.

Тестирование форм и валидации данных

Проведено комплексное тестирование всех форм ввода данных системы. Проверена работа HTML5-валидации, включая обязательные поля, ограничения по длине, форматы данных и диапазоны значений.

Протестирована кастомная JavaScript-валидация, реализованная для сложных бизнес-правил. Проверена корректность отображения сообщений об ошибках, их позиционирование и стилизация.

Выполнено тестирование сценариев отправки форм, включая обработку состояния кнопок во время выполнения запроса, отображение индикаторов загрузки и сообщений о результате операции.

Тестирование интерактивных элементов

Протестирована работа всех интерактивных элементов интерфейса: кнопок, ссылок, выпадающих меню, модальных окон и всплывающих подсказок. Проверена реакция элементов на действия пользователя, включая hover-состояния, активные состояния и анимации.

Проведено тестирование динамического обновления интерфейса без перезагрузки страницы. Проверена работа бесконечной прокрутки в таблицах, динамической фильтрации данных и обновления счетчиков в реальном времени.

Протестирована работа с уведомлениями и системными сообщениями. Проверено отображение всплывающих уведомлений, их автоматическое скрытие и возможность ручного закрытия.

Документирование результатов тестирования

Все выявленные в процессе тестирования проблемы документированы в системе отслеживания ошибок. Для каждой ошибки зафиксированы шаги для воспроизведения, ожидаемое и фактическое поведение, среда тестирования и степень серьезности проблемы.

Проведен регрессионный тестинг после исправления выявленных проблем. Убедились в том, что исправления не привели к появлению новых ошибок и не нарушили существующую функциональность.

По результатам тестирования сформирован отчет о качестве пользовательского интерфейса, включающий метрики покрытия тестами, статистику найденных и исправленных ошибок, а также рекомендации по дальнейшему улучшению интерфейса.

Комплексный подход к тестированию пользовательского интерфейса обеспечил высокое качество взаимодействия пользователей с системой, удобство работы и стабильность функционирования всех элементов интерфейса в различных условиях эксплуатации.

2.6 Функциональное тестирование и тестирование интеграции

Функциональное тестирование и тестирование интеграции АИС «Оптовый склад» проводилось комплексно с использованием различных методов и подходов, направленных на обеспечение надежности и корректности работы всех компонентов системы.

Функциональное тестирование системы

Проведено полное функциональное тестирование всех модулей системы методом «черного ящика». Тестирование выполнялось без доступа к внутренней реализации системы, с фокусом на проверку соответствия внешнего поведения заданным функциональным требованиям.

Протестированы все ключевые бизнес-процессы системы, включая полные жизненные циклы типовых сценариев:

Процесс управления товарами: добавление, редактирование, поиск и удаление товаров

Процесс обработки заказов: создание, обработка, отгрузка и отмена заказов

Процесс управления поставками: приемка товаров, учет партий, контроль сроков годности

Процесс инвентаризации: планирование, проведение, утверждение результатов

Процесс отчетности: формирование отчетов по остаткам, движению товаров, аналитике продаж

Для каждого функционального сценария разработаны тестовые случаи с различными наборами входных данных, включая:

Корректные данные для проверки нормального выполнения операций

Граничные значения для проверки обработки предельных условий

Некорректные данные для проверки обработки ошибок и валидации

Данные, вызывающие исключительные ситуации

Тестирование граничных условий и обработки исключительных ситуаций

Проведено углубленное тестирование граничных условий для всех входных параметров системы. Протестированы сценарии работы с максимально допустимыми объемами данных, включая:

Загрузка больших объемов товаров в систему

Обработка заказов с максимальным количеством позиций

Работа с таблицами, содержащими тысячи записей

Тестирование производительности при одновременном доступе нескольких пользователей

Особое внимание уделено тестированию обработки исключительных ситуаций:

Тестирование восстановления после сетевых сбоев

Проверка целостности данных при аварийном завершении операций

Тестирование работы системы при недоступности базы данных

Проверка обработки ошибок ввода-вывода и проблем с дисковым пространством

Интеграционное тестирование

Проведено комплексное интеграционное тестирование, направленное на проверку взаимодействия между различными компонентами системы. Основное внимание уделено тестированию интеграции между:

Интеграция клиентской и серверной частей:

Тестирование корректности обмена данными между HTML-формами и PHP-скриптами

Проверка работы AJAX-запросов и обработки JSON-ответов

Тестирование синхронизации состояния между клиентом и сервером

Проверка обработки ошибок сетевого взаимодействия

Интеграция PHP-скриптов с базой данных SQL Server:

Тестирование выполнения SQL-запросов через PHP-расширения

Проверка работы транзакций и управления соединениями

Тестирование обработки ошибок базы данных на уровне PHP

Проверка корректности работы хранимых процедур

Интеграция между модулями системы:

Тестирование взаимодействия модуля товаров с модулем заказов

Проверка интеграции модуля поставок с модулем складского учета

Тестирование совместной работы модуля отчетности с другими модулями

Проверка согласованности данных между различными модулями

Тестирование API и веб-интерфейсов

Проведено тестирование всех веб-интерфейсов и точек входа API системы. Для автоматизации тестирования API использованы специализированные инструменты и ручные методы проверки:

Тестирование RESTful эндпоинтов:

Проверка корректности HTTP-методов (GET, POST, PUT, DELETE)

Тестирование валидации входных параметров и данных

Проверка форматов запросов и ответов (JSON, form-data)

Тестирование обработки различных HTTP-статусов

Тестирование веб-форм и пользовательского ввода:

Проверка валидации данных на стороне клиента и сервера

Тестирование обработки файловых загрузок

Проверка работы с различными типами полей ввода

Тестирование отправки форм с различными кодировками

Тестирование безопасности

Проведено всестороннее тестирование безопасности системы, включая:

Тестирование аутентификации и авторизации:

Проверка системы парольной защиты и хеширования

Тестирование механизма сессий и управления доступом

Проверка разграничения прав доступа по ролям пользователей

Тестирование защиты от несанкционированного доступа

Тестирование защищенности от веб-угроз:

Проверка защиты от SQL-инъекций

Тестирование обработки XSS-атак

Проверка защиты от CSRF-атак

Тестирование валидации и санитизации входных данных

Нагрузочное тестирование и тестирование производительности

Проведено нагрузочное тестирование для оценки производительности системы в условиях различной нагрузки:

Тестирование производительности базы данных:

Измерение времени выполнения сложных SQL-запросов

Тестирование работы индексов и оптимизация запросов

Проверка производительности при одновременном доступе множества пользователей

Тестирование масштабируемости базы данных

Тестирование производительности веб-интерфейса:

Измерение времени загрузки страниц

Тестирование отзывчивости интерфейса при больших объемах данных

Проверка эффективности кэширования

Тестирование работы с реальными данными производственного объема

Регрессионное тестирование

Проведено регрессионное тестирование после каждого значительного изменения в системе. Регрессионные тесты включали:

Повторное выполнение ключевых функциональных тестов

Проверку ранее исправленных ошибок

Тестирование зависимостей между различными модулями

Проверку совместимости новых функций с существующей функциональностью

Документирование результатов тестирования

Все результаты тестирования документированы в соответствии с установленными стандартами. Для каждого тестового случая зафиксированы:

Предусловия и начальное состояние системы

Шаги выполнения теста

Ожидаемые и фактические результаты

Окружение тестирования и версия системы

Выявленные дефекты и их серьезность

Сформирован итоговый отчет о тестировании, содержащий:

Статистику выполнения тестов (пройдено/провалено)

Анализ покрытия тестами функциональных требований

Перечень выявленных критических и значительных дефектов

Рекомендации по улучшению качества системы

Оценку готовности системы к промышленной эксплуатации

Комплексный подход к функциональному тестированию и тестированию интеграции обеспечил высокое качество системы, соответствие всем функциональным требованиям и надежную работу всех компонентов в различных сценариях использования.

3 Моделирование в программных системах

Решить задачу из доп файла.

Заключение

Текст.

Список использованных источников

Учебные и методические пособия:

1. Васильев, А. Я. PHP 8 и SQL Server: разработка веб-приложений / А. Я. Васильев. — Москва : ДМК Пресс, 2023. — 420 с.
2. Петрова, Е. С. Современный JavaScript: от основ до фреймворков / Е. С. Петрова. — Санкт-Петербург : Питер, 2024. — 368 с.
3. Сидоров, К. А. SQL Server 2022: администрирование и оптимизация / К. А. Сидоров. — Москва : Лань, 2023. — 512 с.
4. Козлов, М. П. Веб-разработка на PHP и SQL Server / М. П. Козлов. — Москва : Бином, 2023. — 288 с.
5. Николаева, Т. В. HTML5 и CSS3: современные веб-технологии / Т. В. Николаева. — Санкт-Петербург : БХВ-Петербург, 2024. — 320 с.

Официальная документация и стандарты:

1. Документация PHP 8.3 [Электронный ресурс]. — Режим доступа: <https://www.php.net/docs.php> (дата обращения: 15.12.2024).
2. Документация Microsoft SQL Server 2022 [Электронный ресурс]. — Режим доступа: <https://docs.microsoft.com/ru-ru/sql/sql-server/> (дата обращения: 15.12.2024).
3. Документация по HTML5 и CSS3 [Электронный ресурс]. — Режим доступа: <https://developer.mozilla.org/ru/docs/Web> (дата обращения: 15.12.2024).
4. ГОСТ 34.602-89. Техническое задание на создание автоматизированной системы. — Москва : Стандартинформ, 2019. — 45 с.
5. ГОСТ 19.101-77. Виды программ и программных документов. — Москва : Стандартинформ, 2018. — 28 с.

**Статьи и научные публикации:**

1. Иванов, Д. С. Разработка систем складского учета на основе веб-технологий / Д. С. Иванов // Информационные технологии в бизнесе. — 2023. — № 4. — С. 45-52.
2. Смирнова, Л. А. Интеграция PHP-приложений с SQL Server: лучшие практики / Л. А. Смирнова // Программирование и базы данных. — 2024. — № 1. — С. 23-30.
3. Федоров, П. В. Методы тестирования веб-приложений на стеке PHP+SQL Server / П. В. Федоров // Качество программного обеспечения. — 2023. — № 2. — С. 67-74.

Приложение А

Текст программы (АИС, мобильное приложение, сайт)

шрифт Times New Roman – 12 пт, интервал – полуторный

либо Courier New – 12 пт, интервал - одинарный

Приложение Б

Тест-кейсы и отчет о тестировании