МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра системного анализа и
автоматического управления

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ СИСТЕМЫ ЭЛЕКТРОННОЙ КОММЕРЦИИ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 5 курса 551 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Быкова Данилы Сергеевича

Научный руководитель
доцент, к. т. н. _____ Д. Ю. Петров

Заведующий кафедрой
к. ф.-м. н., доцент _____ И. Е. Тананко

Саратов 2026

ВВЕДЕНИЕ

Актуальность темы. По данным Ассоциации компаний интернет-торговли, в 2025 году объём интернет-торговли в России вырос на 28 % и достиг 11,5 трлн руб., а доля онлайн-продаж в розничном товарообороте поднялась до 18,8 % [3]. Доля отечественных интернет-магазинов и маркетплейсов составила 96,2 % этого объёма; трансграничная торговля занимает оставшиеся 3,8 % [3]. Внутри рынка усиливается переход от классических интернет-магазинов к маркетплейсам, объединяющим на одной площадке предложения многих продавцов [2].

Заметный сегмент рынка занимают цифровые товары — программные лицензии, пополнения игровых сервисов, подписки, медиаконтент. Цифровой товар не требует складского хранения и физической доставки: после оплаты покупателю выдаётся файл или ключ. Это порождает обратную задачу — выдать файл только тому покупателю, который оформил и оплатил заказ, и не допустить повторного или несанкционированного скачивания.

Гаврилов Л. П. разделяет программные средства для электронной коммерции на готовые системы управления интернет-магазином и заказные разработки, выбор между которыми определяется требованиями к функциональности и стоимостью владения [1]. Готовые платформы ориентированы на типовой сценарий со складским учётом, корзиной и физической доставкой; маркетплейс цифровых товаров с несколькими независимыми продавцами и выдачей файла поддерживается в них через платные модули или доработку исходного кода. Это делает обоснованной заказную разработку для рассматриваемого сегмента.

Цель бакалаврской работы — разработать программное обеспечение системы электронной коммерции, реализующей функции маркетплейса цифровых товаров.

Поставленная цель определила **следующие задачи**:

1. проанализировать предметную область: принципы работы и классификацию систем электронной коммерции, существующие программные решения, архитектурные подходы к построению веб-систем;
2. сформировать функциональные и нефункциональные требования к программному продукту;
3. спроектировать архитектуру системы, модель данных и структуру

классов серверной части;

4. выбрать технологический стек для серверной и клиентской частей;
5. реализовать серверный API, витрину и административные панели;
6. протестировать ключевые сценарии работы и развернуть демонстрационный стенд.

Методологические основы работы составили исследования и руководства в области электронной коммерции и проектирования информационных систем: труды Гаврилова Л. П. по построению систем электронной коммерции [1], работа Плаксиной В. С. по маркетплейсам и тенденциям онлайн-торговли [2], учебное пособие Ананченко И. В., Войтюк Т. Е., Марченко Е. В. по архитектуре информационных систем [4] и руководство Рочева К. В. по их анализу и проектированию [5], монография Мартина Р. по уровневой архитектуре программного обеспечения [6], руководство Буча Г., Рамбо Дж. и Якобсона И. по языку UML [7], статья Галигузовой Е. В. и Илларионовой Ю. Е. о сравнении протоколов GraphQL и REST [8], работа Куликова С. С. по тестированию программного обеспечения [9]. Качество продукта оценивалось по характеристикам, заданным ГОСТ Р ИСО/МЭК 25010-2015 [10].

Практическая значимость бакалаврской работы. Разработанное программное обеспечение воспроизводит полный цикл продажи цифрового товара — от каталога до выдачи файла покупателю — и пригодно как основа для запуска маркетплейса цифровых товаров либо как учебный пример проектирования веб-системы с разделением серверной и клиентской частей. Демонстрационный стенд развёртывается из исходного кода набором контейнеров Docker и может быть использован при изучении дисциплин, связанных с проектированием веб-приложений.

Структура и объём работы. Бакалаврская работа состоит из введения, шести разделов, заключения, списка использованных источников информации и шести приложений. Общий объём работы — 92 страницы, из них 76 страниц — основное содержание, включая 31 рисунок и 4 таблицы, USB-флеш-накопитель с исходным кодом в качестве приложения, список использованных источников информации — 31 наименование.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел — «Анализ предметной области» формирует кон-

текст для последующего проектирования.

В подразделе 1.1 рассмотрены основные принципы функционирования систем электронной коммерции и их классификация по составу участников сделки (B2B, B2C, C2C, B2G) по работе Гаврилова Л. П. [1]. Введены понятия программной системы электронной коммерции, витрины, маркетплейса и цифрового товара.

В подразделе 1.2 разобраны существующие программные решения для построения интернет-магазинов и маркетплейсов. Обзор показал, что готовые платформы базовой поставкой поддерживают типовой сценарий: один продавец, складской учёт, физическая доставка. Сценарий маркетплейса цифровых товаров с несколькими продавцами и выдачей файла либо требует доработки исходного кода, либо реализуется через платные модули. По итогам обзора обоснован выбор заказной разработки.

В подразделе 1.3 рассмотрены архитектурные подходы к построению веб-систем: монолитная, клиент-серверная, многоуровневая и headless-архитектуры. Сопоставление подходов опирается на работы Ананченко И. В. и соавторов [4], Рочева К. В. [5] и монографию Мартина Р. [6]: уровневое деление с зависимостями, направленными к бизнес-логике, позволяет менять СУБД или формат API без переписывания бизнес-слоя.

В подразделе 1.4 рассмотрены способы организации программного интерфейса между клиентом и сервером: архитектурный стиль REST и язык запросов GraphQL. Сопоставление выполнено по работе Галигузовой Е. В. и Илларионовой Ю. Е. [8]: для типовых операций чтения каталога удобнее REST за счёт простоты маршрутов и кэширования; для разнородных запросов с переменным набором полей выгоднее GraphQL за счёт одной точки входа и контроля над составом ответа. Сделан вывод о том, что выбор протокола определяется характером запросов клиента.

В подразделе 1.5 выбраны нотации представления требований и проектных решений — диаграммы UML (вариантов использования, классов, последовательности, развёртывания) по Бучу Г. и соавторам [7] и модель «сущность–связь» для проектирования базы данных.

В подразделе 1.6 рассмотрены характеристики качества программного обеспечения по ГОСТ Р ИСО/МЭК 25010-2015 [10]. Для тестирования DigitalMarket из восьми характеристик выбраны функциональная пригод-

ность, производительность и надёжность.

В выводах по разделу зафиксированы исходные позиции для проектирования: целесообразность заказной разработки, выбор клиент-серверной и headless-архитектур с уровневым делением серверной части, выбор нотаций представления и характеристик качества для последующей проверки.

Второй раздел — «Формирование требований к программному продукту» переводит результаты анализа предметной области в перечень требований к разрабатываемой системе. Разрабатываемой системе присвоено название DigitalMarket.

В подразделе 2.1 описаны функциональные требования через роли пользователей и доступные им действия. Выделено пять ролей: гость, покупатель, продавец — владелец магазина, сотрудник магазина и администратор площадки. Для каждой роли перечислен набор сценариев. Покупателю доступны просмотр каталога, оформление заказа, скачивание оплаченного файла, ведение профиля, публикация отзывов и вопросов. Продавцу-владельцу магазина доступны ведение каталога магазина, обработка заказов магазина, просмотр аналитики продаж и управление сотрудниками. Администратору площадки доступны управление учётными записями, категориями и типами товаров, настройками витрины и модерация обратной связи. Полный состав сценариев представлен на диаграмме вариантов использования в нотации UML [7].

В подразделе 2.2 сформулированы нефункциональные требования. Они сгруппированы по характеристикам качества из ГОСТ Р ИСО/МЭК 25010-2015 [10]: функциональная пригодность (полнота поддержки сценариев, разграничение прав по ролям), производительность (время отклика страниц витрины и API), надёжность (целостность данных при оформлении заказа, восстановление после сбоев), сопровождаемость (модульность серверной части, наличие тестов) и переносимость (контейнеризация развёртывания). Каждое требование сопровождено критерием приёмки.

В выводах по разделу зафиксирован перечень требований, на основании которого далее проектируются архитектура и модель данных.

Третий раздел — «Проектирование архитектуры и модели данных» содержит проектные решения для DigitalMarket.

В подразделе 3.1 описана архитектура программной системы. Серверная часть разделена на уровни (контроллеры, репозитории, доменные серви-

сы, данные) согласно принципу направленных к бизнес-логике зависимостей по Мартину Р. [6]: такая структура позволяет менять СУБД или формат API без переписывания бизнес-логики. Выбран headless-подход: один серверный API обслуживает три самостоятельных клиентских приложения — покупательскую витрину и две административные панели (на протоколах REST и GraphQL). Архитектура иллюстрирована соответствующей диаграммой.

В подразделе 3.2 спроектирована модель данных в виде ER-диаграммы. Центральное проектное решение — разделение заказа на родительский и дочерние: родительский заказ соответствует чеку покупателя, дочерние — частям заказа, принадлежащим магазинам разных продавцов. Это решение поддерживает сценарий маркетплейса с несколькими продавцами в одной покупке и упрощает обработку заказа продавцом, которому видны только дочерние заказы его магазина.

В подразделе 3.3 спроектирована структура классов серверной части. Каждой сущности модели данных сопоставлены три класса: модель (объектно-реляционное отображение), репозиторий (доступ к данным) и контроллер (приём HTTP-запросов). Бизнес-логика, не сводимая к чтению или записи, вынесена в доменные сервисы (расчёт стоимости, выдача файла, рассылка уведомлений).

В подразделе 3.4 обоснован выбор технологического стека. Серверная часть реализуется на языке PHP 8.1 с использованием программного каркаса Laravel 9; токены доступа выдаются библиотекой Laravel Sanctum, роли и права — библиотекой Spatie Permission, GraphQL-точка входа — библиотекой Lighthouse. Клиентские приложения реализуются на TypeScript и Next.js (React); для REST-витрины и REST-панели применяется TanStack Query, для GraphQL-панели — Apollo Client; оформление — Tailwind CSS. В качестве СУБД выбран MySQL, в качестве кэша и хранилища сессий — Redis, развёртывание организовано через Docker Compose.

В выводах по разделу зафиксирован набор проектных решений, передаваемых на реализацию.

Четвёртый раздел — «Реализация программного продукта» описывает воплощение проектных решений в код.

В подразделе 4.1 разобрана реализация серверной части. Маршруты REST организованы по сущностям (товары, заказы, магазины, пользователи)

и сгруппированы по уровню доступа: публичные маршруты доступны без токена, защищённые — по токену Sanctum с проверкой прав через Spatie Permission. Типы и резолверы GraphQL зарегистрированы средствами Lighthouse: схема описывает сущности и связи, резолверы переиспользуют те же репозитории, что и REST-контроллеры. Каталог отдаётся с постраничной выдачей, файл цифрового товара — по подписанной ссылке, выдаваемой только после подтверждения оплаты. В приложении приведены коды контроллера пользователей, репозиториев заказа и оформления, а также события OrderCreated.

В подразделе 4.2 разобрана реализация клиентских приложений — покупательской витрины и двух административных панелей — на Next.js. Витрина и REST-панель обращаются к серверу через TanStack Query, GraphQL-панель — через Apollo Client; общее оформление построено на Tailwind CSS. Все три приложения изолированы друг от друга и развёртываются как отдельные сервисы.

В подразделе 4.3 разобран центральный бизнес-процесс системы — оформление заказа покупателем — в нотации BPMN 2.0. Описаны выбор товаров и формирование корзины на витрине, расчёт стоимости с учётом налога и доставки на серверном API, разделение оформленного заказа на родительский и дочерние, создание доступа к цифровому файлу и публикация события OrderCreated. Диаграммы последовательности конкретизируют процесс на уровне взаимодействия контроллеров, репозиториев и базы данных.

В подразделе 4.4 проведено сопоставление двух протоколов доступа к API — REST и GraphQL — по характеру запросов витрины и панели. Подтверждён вывод подраздела 1.4: для витрины с типовым чтением каталога и выдачей файлов проще REST, а для административной панели со сложными страницами, объединяющими разнородные сущности, удобнее GraphQL за счёт одной точки входа и контроля над составом ответа.

В выводах по разделу подведён итог реализации и обозначены сценарии, передаваемые в тестирование.

Пятый раздел — «Тестирование и развёртывание программного обеспечения» описывает проверку соответствия реализованного продукта требованиям.

В подразделе 5.1 описано функциональное тестирование, проведённое методом «чёрного ящика» [9] по сценариям, построенным от функциональ-

ных требований подраздела 2.1. Тестировались регистрация и вход, просмотр каталога и карточки товара, оформление заказа покупателем, получение файла цифрового товара, обработка заказа продавцом, ведение каталога и пользователей администратором. Каждому требованию сопоставлен сценарий, который проходит соответствующую функцию в браузере от начала до конца; наблюдаемый результат сравнивается с ожидаемым. Перечень сценариев и результат их прохождения сведены в таблицу.

В подразделе 5.2 описано тестирование производительности — замеры времени отклика страниц витрины и серверного API при наполненном каталоге. Замеры выполнялись из браузерного инструмента разработчика; усреднённые значения сопоставлены с нефункциональными требованиями подраздела 2.2.

В подразделе 5.3 описано развёртывание демонстрационного стенда: семь контейнеров Docker Compose — серверный API, три клиентских приложения, СУБД MySQL, кэш Redis и контейнер запуска фоновых задач. Приведена диаграмма развёртывания и порядок запуска стенда из исходного кода.

В выводах по разделу зафиксирован результат тестирования и состав развёрнутого стенда.

Шестой раздел — «Обзор программного продукта» содержит обзор готовых пользовательских интерфейсов по снимкам экранов демонстрационного стенда.

В подразделе 6.1 показан интерфейс витрины: главная страница с каталогом цифровых товаров, карточка цифрового товара, страница оформления заказа и страница созданного заказа со ссылкой на файл. Снимки экрана подтверждают полноту реализации сценариев покупателя.

В подразделе 6.2 показан интерфейс административных панелей (REST и GraphQL): страницы управления товарами, заказами, магазинами, категориями, типами и пользователями. Сравнение панелей подтвердило содержательное совпадение наполнения при различии реализации обращения к API.

В выводах по разделу подведён итог обзора и зафиксировано соответствие реализованных интерфейсов сформированным требованиям.

ЗАКЛЮЧЕНИЕ

В результате выполнения работы разработано программное обеспечение системы электронной коммерции — маркетплейса цифровых товаров DigitalMarket.

Анализ предметной области показал, что готовые платформы электронной коммерции не покрывают базовой поставкой три требования маркетплейса цифровых товаров: выдачу файла оплатившему покупателю, разделение заказа по магазинам и разграничение прав пяти ролей. По итогам обзора архитектурных подходов выбрана клиент-серверная архитектура с уровневым делением серверной части и headless-подход, при котором один серверный API обслуживает три витрины.

На основе требований спроектированы уровневая архитектура серверной части, модель данных с разделением заказа на родительский и дочерние и структура классов из контроллеров, репозиториев и моделей. Выбран технологический стек: серверная часть на PHP 8.1 и Laravel 9, клиентские приложения на TypeScript и Next.js, СУБД MySQL, кэш Redis, развёртывание через Docker Compose.

Реализованы серверный API, витрина и две административные панели. Сопоставление панелей на REST и GraphQL показало, что выбор протокола зависит от задачи: для витрины с типовым чтением каталога и отдачей файлов удобнее REST, для административной панели со сложными страницами — GraphQL.

Ключевые сценарии работы протестированы в браузере, расхождений между панелями не выявлено; демонстрационный стенд развёрнут набором из семи контейнеров Docker Compose.

Дальнейшее развитие системы связано с реализацией модулей, не вошедших в текущую версию: подключения внешних платёжных шлюзов для онлайн-оплаты, мобильного клиента и расширенной аналитики продаж. Принятая архитектура с разделением серверной части и витрин позволяет добавлять эти модули, не затрагивая реализованный бизнес-слой.

Поставленные во введении задачи выполнены, цель работы достигнута.

Основные источники информации:

1. Гаврилов Л. П. Электронная коммерция: учебник и практикум для вузов. 3-е изд., доп. М.: Юрайт, 2019. 477 с.
2. Плаксина В. С. Электронная торговля: особенности и тенденции развития [Электронный ресурс] // КиберЛенинка. URL: `https://cyberleninka.ru/article/n/elektronnaya-torgovlya-osobennosti-i-tendentsii-razvitiya` (дата обращения: 21.01.2026).

3. Анализ состояния интернет-торговли в России по итогам 2025 года [Электронный ресурс] / Ассоциация компаний интернет-торговли. URL: https://akit.ru/news/analiz-sostoyaniya-internet-torgovli-v-rossii-po-itogam-2025-goda (дата обращения: 23.01.2026).
4. Ананченко И. В., Войтюк Т. Е., Марченко Е. В. Архитектура информационных систем: учебное пособие. СПб.: Университет ИТМО, 2024. 57 с.
5. Рочев К. В. Информационные технологии. Анализ и проектирование информационных систем. Ухта: УГТУ, 2019. 128 с.
6. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. СПб.: Питер, 2018. 352 с.
7. Буч Г., Рамбо Дж., Якобсон И. Язык UML. Руководство пользователя. 2-е изд. М.: ДМК Пресс, 2007. 496 с.
8. Галигузова Е. В., Илларионова Ю. Е. Язык запросов GraphQL как замена REST API. Сравнение GraphQL и REST API [Электронный ресурс] // КиберЛенинка. URL: https://cyberleninka.ru/article/n/yazyk-zaprosov-graphql-kak-zamena-rest-api-sravnenie-graphql-i-rest-api (дата обращения: 27.02.2026).
9. Куликов С. С. Тестирование программного обеспечения. Базовый курс. Минск: Четыре четверти, 2020.
10. ГОСТ Р ИСО/МЭК 25010-2015. Информационные технологии. Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Модели качества систем и программных продуктов. М.: Стандартинформ, 2015. 36 с.