МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра системного анализа и автоматического управления

ПРИМЕНЕНИЕ ФУНКЦИЙ КЭШИРОВАНИЯ КАТАЛОГА И ОБМЕНА СООБЩЕНИЙ МЕЖДУ КОМПОНЕНТАМИ В ВЕБ-РЕСУРСЕ ИНТЕРНЕТ-МАГАЗИНА

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 5 курса 551 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Алтухова Артёма Михайловича

Научный руководитель
к. т. н., доцент _____ И. Н. Фомин

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

Саратов 2026

ВВЕДЕНИЕ

Актуальность темы. Электронная коммерция является одним из главных драйверов развития мирового и локального рынков. Интернет-магазины ежедневно обслуживают сотни тысяч пользователей, что предъявляет колоссальные требования к производительности, отзывчивости и надёжности веб-ресурсов. Задержки в загрузке каталога товаров или сбои при оформлении заказа напрямую ведут к потере клиентов и снижению финансовой прибыли компаний, что делает задачу оптимизации высоконагруженных систем критически важной.

Традиционный подход к разработке веб-приложений, основанный на монолитной архитектуре, сегодня часто не справляется с возросшими нагрузками. Монолиты сложны в масштабировании, поддержке и обновлении. На смену им пришла микросервисная архитектура [1–3], разделяющая единое приложение на независимые компоненты, каждый из которых отвечает за свою бизнес-логику (авторизация, корзина, каталог, заказы). Однако микросервисный подход порождает новые технические вызовы: необходимость быстрого доступа к общим данным и проблему надёжной синхронизации независимых сервисов.

Одним из наиболее эффективных путей решения проблемы скорости отклика является применение технологий кэширования [4, 5]. Интеграция механизмов кэширования как на стороне сервера, так и на стороне клиента позволяет значительно снизить нагрузку на основную базу данных. Сохранение часто запрашиваемой информации, такой как каталог товаров и списки категорий, в быструю промежуточную память гарантирует мгновенную выдачу контента пользователю и фундаментально улучшает пользовательский опыт.

В то же время распределённая природа микросервисов требует надёжных каналов связи между компонентами. Синхронное взаимодействие сервисов через прямые HTTP-запросы часто приводит к эффекту «домино», когда отказ одного узла парализует всю систему. Применение асинхронного обмена сообщениями на базе брокеров (RabbitMQ, Kafka) позволяет реализовать событийно-ориентированную архитектуру и гарантирует надёжную доставку данных даже при временной недоступности отдельных модулей.

Таким образом, разработка веб-ресурса интернет-магазина, объединяющего передовые функции кэширования каталога и асинхронного обмена сообщениями, является высокоактуальной инженерной и научно-прикладной задачей. Её решение позволяет создать масштабируемую, отказоустойчивую и высо-

копроизводительную платформу, отвечающую всем требованиям современной электронной коммерции.

Объектом исследования является процесс проектирования и разработки масштабируемых высоконагруженных веб-ресурсов на базе микросервисной архитектуры.

Предметом исследования являются методы, алгоритмы и программные средства реализации функций кэширования данных и асинхронного обмена сообщениями между микросервисами интернет-магазина.

Целью бакалаврской работы является разработка архитектуры программного комплекса интернет-магазина с применением механизмов кэширования каталога и обмена сообщениями между компонентами для повышения производительности и отказоустойчивости системы.

Поставленная цель определила **следующие задачи**:

1. провести теоретический и сравнительный анализ современных архитектурных паттернов веб-приложений, а также изучить научные подходы к кэшированию и маршрутизации данных;
2. исследовать методы и инструменты программной реализации кэширования и событийно-ориентированного обмена сообщениями в распределённых веб-системах;
3. сформулировать технические и нефункциональные требования к разрабатываемому интернет-магазину и обосновать выбор стека технологий;
4. спроектировать архитектуру системы, структуру реляционной базы данных и смоделировать сценарии взаимодействия компонентов и действий пользователей;
5. разработать программную реализацию бэкенд- и фронтенд-частей веб-ресурса, интегрировав механизмы кэширования каталога и брокер сообщений для связи микросервисов;
6. провести нагрузочное тестирование разработанной системы и выполнить анализ полученных результатов для подтверждения эффективности применённых технологических решений.

Методологическую основу работы составили труды отечественных и зарубежных специалистов в области проектирования распределённых веб-приложений, микросервисной архитектуры и высоконагруженных систем: Э. Таненбаума [1], Н. Н. Нагорного [2], А. Г. Мурлина [3], а также работы, посвящённые техноло-

гиям кэширования — З. И. Хадеевой [4] и М. Х. Томаева [5]. При выполнении работы применялись методы системного анализа для сбора и формализации требований, методы объектно-ориентированного и структурного проектирования (DFD, ER, BPMN, IDEF0) для визуального моделирования бизнес-процессов и архитектуры баз данных, методы программной инженерии для реализации микросервисов, а также методы эмпирического исследования и нагрузочного тестирования для оценки показателей производительности и надёжности системы.

Практическая значимость работы заключается в создании готового контейнеризированного прототипа интернет-магазина, который может быть использован как самостоятельный продукт или как масштабируемая основа для реального коммерческого бизнеса. Применённые подходы по распределению нагрузки и обеспечению асинхронной связи формируют отказоустойчивую базу, защищённую от пиковых нагрузок. Разработанные схемы, диаграммы и исходный код могут служить методическим и практическим руководством для IT-специалистов при проектировании аналогичных микросервисных платформ.

Структура и объём работы. Бакалаврская работа состоит из введения, трёх разделов, заключения, списка использованных источников и приложения с листингами программного кода. Общий объём работы — 65 страниц, включая 17 рисунков и 6 таблиц. Список использованных источников информации содержит 36 наименований.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Теоретические основы технологий кэширования в микросервисных интернет-ресурсах» посвящён формированию теоретико-методологической базы, необходимой для разработки современного высоконагруженного интернет-магазина.

В подразделе 1.1 рассмотрена эволюция архитектурных паттернов веб-приложений: проведено сравнение монолитной и микросервисной архитектур, выявлены их преимущества и недостатки. Показано, что переход к микросервисам обеспечивает гибкость, независимое масштабирование и изоляцию сбоев, однако порождает проблемы синхронизации распределённых данных и сетевых задержек.

В подразделе 1.2 выполнен анализ научных публикаций о методах и инструментах кэширования. Систематизированы подходы к распределению на-

грузки, кэшированию ответов и маршрутизации данных между изолированными узлами системы.

В подразделе 1.3 проведён анализ существующих проектов с реализацией функций кэширования и обмена сообщениями. Рассмотрены архитектурные решения отечественных высоконагруженных сервисов, что позволило сформировать представление о практически применимых подходах.

В подразделе 1.4 представлен сравнительный анализ микросервисных технологий и методов кэширования, на основе которого сделан вывод о целесообразности применения двухуровневого кэширования (на стороне сервера и клиента).

В подразделе 1.5 рассмотрены методы и инструменты обмена сообщениями. Обоснована необходимость использования надёжного брокера сообщений, реализующего асинхронную модель взаимодействия компонентов для обеспечения итоговой согласованности данных.

В подразделе 1.6 определены пользовательские функции кэширования каталога и обмена сообщениями между компонентами, формирующие целостное представление о разрабатываемом продукте.

В подразделе 1.7 на основе проведённого теоретического анализа сформулированы технические задачи разработки веб-ресурса интернет-магазина с функциями кэширования и обмена сообщениями.

Второй раздел «Проектирование и разработка веб-ресурса интернет-магазина» посвящён проектированию архитектуры системы и программной реализации её ключевых компонентов.

В подразделе 2.1 определены технические и нефункциональные требования к разрабатываемому интернет-магазину, включая жёсткие лимиты на время отклика и требования к отказоустойчивости.

В подразделе 2.2 обоснован выбор и спроектирована микросервисная архитектура системы. Определена структура, включающая API-шлюз для маршрутизации клиентских запросов и изолированные сервисы (авторизации, каталога, корзины, заказов и платежей), а также централизованные инструменты оптимизации — Redis и RabbitMQ. Общая схема спроектированной архитектуры представлена на рисунке 1.

В подразделе 2.3 выполнено проектирование компонентов системы, в подразделе 2.4 — проектирование структуры реляционной базы данных с исполь-

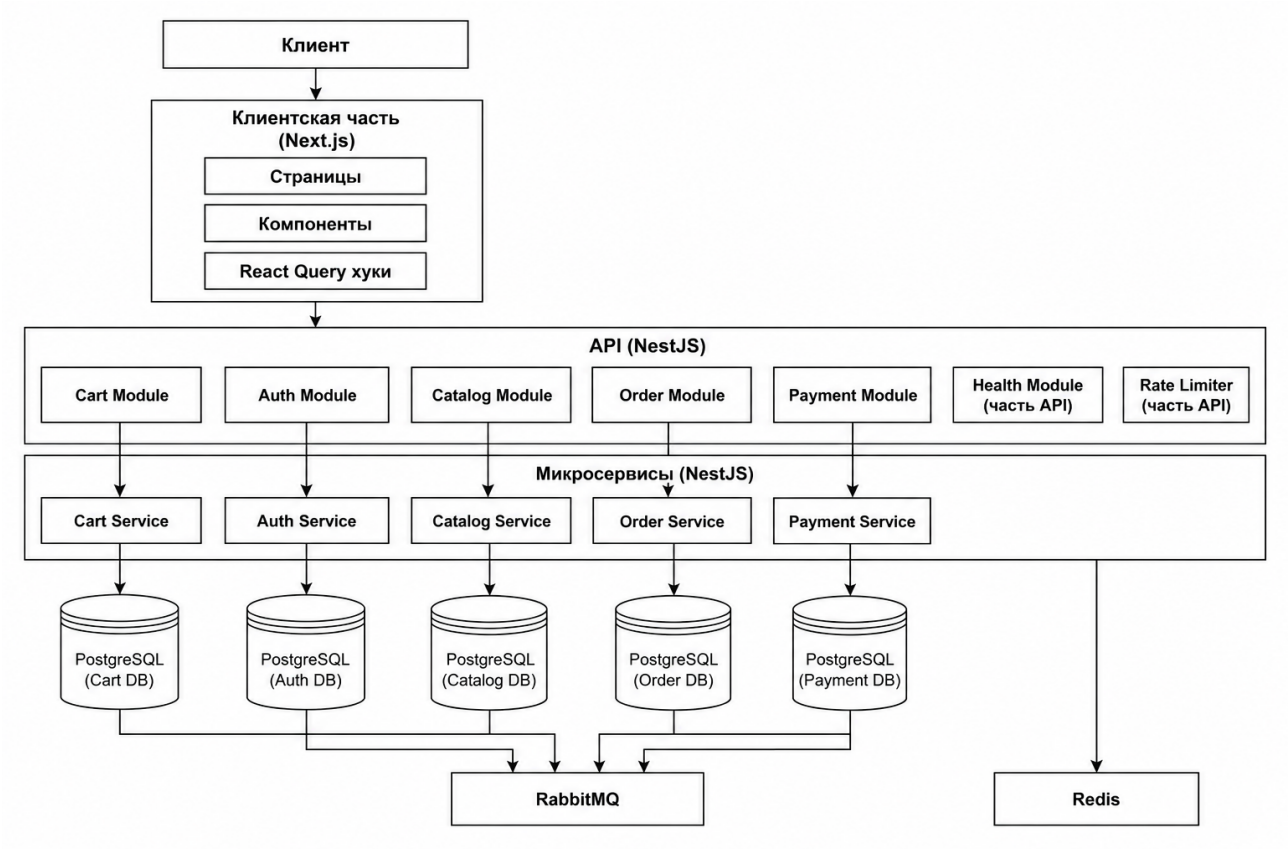

Рисунок 1 – Архитектура проекта

зованием ER-моделирования для каждого независимого сервиса. Логическая ER-диаграмма предметной области приведена на рисунке 2.

В подразделе 2.5 смоделированы сценарии действий пользователей с применением нотаций BPMN и диаграмм прецедентов, что позволило формализовать бизнес-процессы взаимодействия пользователя с системой.

В подразделе 2.6 описаны выбор и настройка среды разработки, обоснован технологический стек: фреймворки NestJS и Next.js, брокер сообщений RabbitMQ, in-memory хранилище Redis, СУБД PostgreSQL и платформа контейнеризации Docker.

В подразделе 2.7 с помощью методологии IDEF0 описаны этапы реализации системы.

Подраздел 2.8 посвящён программной реализации функций кэширования каталога. Реализован механизм двухуровневого кэширования: клиентское кэширование на базе библиотеки React Query со стратегией «Stale-While-Revalidate» и серверное кэширование в in-memory хранилище Redis. Применение этого механизма позволило избежать выполнения сложных SQL-запросов при каждом обращении к каталогу.

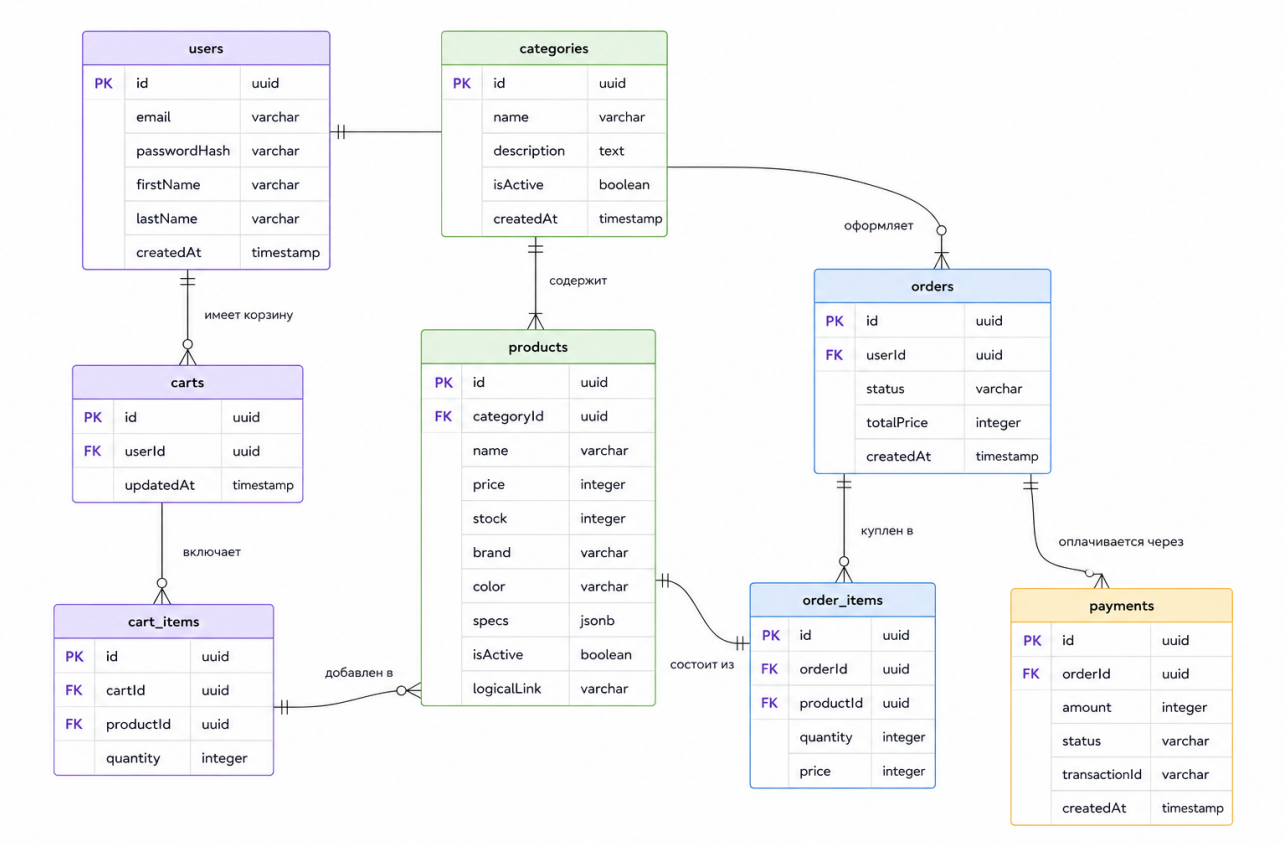

Рисунок 2 – Логическая ER-диаграмма предметной области микросервисной системы

В подразделе 2.9 описана реализация функций обмена сообщениями. На базе брокера RabbitMQ построена событийно-ориентированная архитектура: запуск таких событий, как OrderCreated и PaymentProcessed, позволил вынести ресурсоёмкие транзакции в фоновые процессы и обеспечить слабую связность микросервисов.

Третий раздел «Запуск интернет-магазина и тестирование сервиса» посвящён развёртыванию разработанной системы и оценке её эффективности.

В подразделе 3.1 описан запуск сайта. Развёрнута отказоустойчивая микросервисная инфраструктура, состоящая из 14 Docker-контейнеров. Перенос ресурсоёмких процессов сборки образов на высокопроизводительный сервер позволил сократить время развёртывания с 3–4 часов до 2–3 минут, а использование технологии обратных туннелей обеспечило безопасный публичный доступ к сервису по протоколу HTTPS без приобретения выделенного статического IP-адреса.

В подразделе 3.2 описано нагрузочное тестирование разработанного программного комплекса в условиях, имитирующих пиковую пользовательскую активность (до 1000 запросов в секунду).

В подразделе 3.3 проведён анализ полученных результатов. Сравнение двух режимов работы Catalog API — с кэшированием на базе Redis и без него (с прямыми обращениями к базе данных) — при нагрузке 100, 300, 500 и 1000 запросов в секунду представлено на рисунке 3. Внедрение слоя кэширования позволило удерживать коэффициент попадания в кэш (hit-rate) на уровне 95–96 %, при этом сам кэш каталога потреблял всего 1,2 МБ оперативной памяти. При пиковой нагрузке в 1000 запросов в секунду среднее время отклика сервиса с кэшем составило 1,54 мс против 1968,83 мс без кэша (ускорение более чем в 1200 раз), а максимальная задержка снизилась с 5945,2 мс до 111,4 мс. Кроме того, применение кэша полностью устранило отказы в обслуживании: количество запросов, которые база данных не успевала обработать, при отключённом кэше достигало 18 642, тогда как при включённом кэшировании оставалось равным нулю. Таким образом, система продемонстрировала высокую надёжность, отказоустойчивость и готовность к реальной коммерческой эксплуатации.

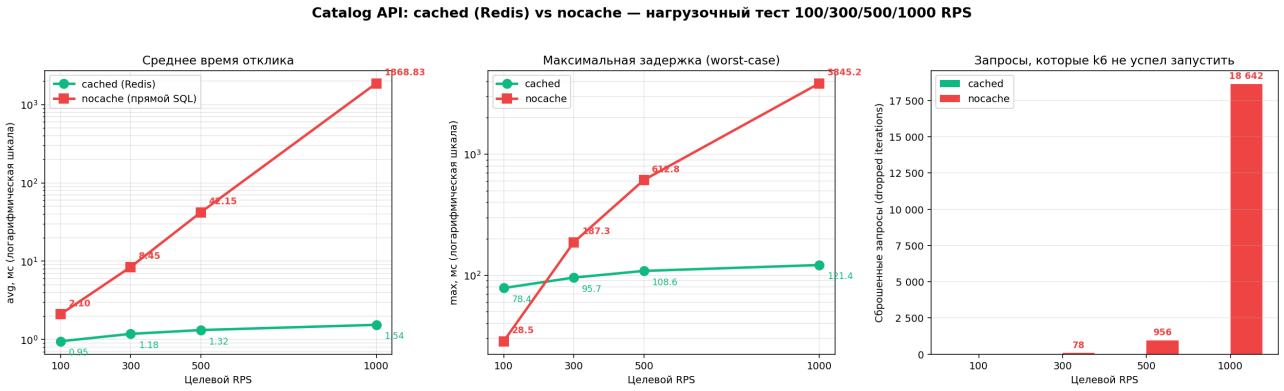

Рисунок 3 – Сравнение производительности Catalog API с кэшированием (Redis) и без него при нагрузке 100/300/500/1000 запросов в секунду

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы была разработана архитектура программного комплекса интернет-магазина с применением механизмов кэширования каталога и обмена сообщениями между компонентами для повышения производительности и отказоустойчивости системы.

Проведён теоретический и сравнительный анализ современных архитектурных паттернов веб-приложений, который показал неоспоримые преимущества микросервисной архитектуры при создании масштабируемых платформ. Исследованы методы и программные инструменты реализации кэширования

и событийно-ориентированного обмена сообщениями. Сформулированы технические и нефункциональные требования и обоснован выбор современного технологического стека: NestJS, Next.js, RabbitMQ, Redis, PostgreSQL и Docker.

Спроектирована комплексная микросервисная архитектура системы с использованием DFD-, ER- и BPMN-моделирования. Выполнена программная реализация бэкенд- и фронтенд-частей веб-ресурса с интеграцией механизмов кэширования каталога (React Query и Redis) и брокера сообщений RabbitMQ, обеспечившего надёжную слабосвязанную передачу системных событий между изолированными микросервисами.

Проведённое нагрузочное тестирование подтвердило, что внедрение механизмов кэширования многократно снизило нагрузку на основную базу данных и сократило время отклика, а событийная архитектура обмена сообщениями обеспечила высокую отказоустойчивость и сохранность данных без потерь. Поставленные в работе задачи решены в полном объёме, цель достигнута, а полученные результаты обладают практической значимостью и могут быть использованы при разработке современных веб-приложений на основе микросервисной архитектуры.

Основные источники информации:

1. Таненбаум, Э. Компьютерные сети / Э. Таненбаум, Д. Уэзеролл. — 5-е рус. изд. — Санкт-Петербург: Питер, 2024.
2. Нагорный, Н. Н. Основные аспекты разработки микросервисного веб-приложения / Н. Н. Нагорный // Системный анализ, управление и обработка информации. — 2023. — DOI: 10.23670/IRJ.2023.133.121.
3. Мурлин, А. Г. Проблема выбора современного архитектурного решения для создания интерактивных приложений и веб-сайтов / А. Г. Мурлин, А. Р. Анисимова, Г. В. Ведерников // Инновационная экономика: перспективы развития и совершенствования. — 2025.
4. Хадеева, З. И. Кэширование в качестве определённой операции в программировании. Кеш в качестве составной части приложений и программ // Перспективное развитие науки, техники и технологий: сборник научных статей 12-й Международной конференции. — 2022. — С. 388–391.
5. Томаев, М. Х. Формализация метода статического кэширования функций / М. Х. Томаев, Р. А. Миронян // Технологии оптимизации пользовательских кодов. — Владикавказ, 2022.

6. Кантелон, М. Node.js в действии / М. Кантелон, А. Янг, Б. Мек. — 2-е изд. — Санкт-Петербург: Питер, 2022. — 432 с.
7. Стоянов, С. React.js. Быстрый старт / С. Стоянов. — Санкт-Петербург: Питер, 2020. — 304 с.
8. Стружкин, Н. П. Базы данных: проектирование / Н. П. Стружкин, В. В. Годин. — Москва: Юрайт, 2023. — 477 с.
9. Давыдовский, М. А. Разработка веб-сервисов / М. А. Давыдовский. — Москва: РУТ (МИИТ), 2020. — 111 с.