ISBN 978-5-901722-96-1

УДК 004.4:656.1

РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ УПРАВЛЕНИЯ ЗАКАЗАМИ НА ГРУЗОПЕРЕВОЗКИ НА ОСНОВЕ СТЕКА NODE.JS, REACT И POSTGRESQL

Аль-Гурайри Зайд Саади Салман[1], студент
e-mail: zaid.saadi789@gmail.com
Кремлёва Эльмира Шамильевна[1], кандидат технических наук, доцент кафедры вычислительных систем
e-mail: e-smile29.04@mail.ru

[1] Казанский национальный исследовательский технический университет имени А.Н. Туполева, Казань, Россия

Аннотация. В статье рассматривается разработка информационной системы управления заказами на грузоперевозки, ориентированной на малые и средние транспортные компании. Система автоматизирует полный жизненный цикл заказа: от регистрации заявки до подтверждения доставки и выставления счёта. Проведён сравнительный анализ существующих TMS-решений, обоснован выбор технологий и спроектирована трёхуровневая архитектура на базе Node.js, Express, TypeScript, React, Vite и PostgreSQL. Реализованы ролевая модель доступа, REST API и клиентские интерфейсы для четырёх категорий пользователей.

Ключевые слова: грузоперевозки, транспортная логистика, информационная система, TMS, REST API, Node.js, PostgreSQL, ролевая модель доступа.

DEVELOPMENT OF A FREIGHT ORDER MANAGEMENT INFORMATION SYSTEM BASED ON THE NODE.JS, REACT AND POSTGRESQL STACK

Zaid Saadi Salman Al-Gurairi[1], Bachelor’s Degree Student
e-mail: zaid.saadi789@gmail.com
Elmira Sh. Kremleva[1], Candidate of Technical Sciences, Associate Professor of the Department of Computing Systems
e-mail: e-smile29.04@mail.ru

[1] Kazan National Research Technical University named after A. N. Tupolev, Kazan, Russia

Abstract. The paper presents the development of an information system for freight order management aimed at small and medium-sized transport companies. The system automates the full order lifecycle: from request registration to delivery confirmation and invoicing. A comparative analysis of existing TMS solutions is performed, technology choices are justified, and a three-tier architecture based on Node.js, Express, TypeScript, React, Vite, and PostgreSQL is designed. A role-based access control model, a REST API, and client interfaces for four user categories are implemented.

Keywords: freight transportation, transport logistics, information system, TMS, REST API, Node.js, PostgreSQL, role-based access control.

Транспортно-логистическая отрасль обеспечивает движение товаров между производителями и потребителями и является одним из ключевых секторов экономики. Рост объёмов грузоперевозок и усиление конкуренции формируют устойчивый спрос на цифровизацию операционных процессов транспортных компаний [1, 2]. Вместе с тем значительная часть предприятий малого и среднего бизнеса по-прежнему ведёт учёт заказов средствами электронной почты, телефона и электронных таблиц, что порождает задержки в обработке заявок, ошибки при назначении водителей, отсутствие прозрачности для клиентов и сложности при формировании отчётности.

Промышленные TMS-системы (SAP Transportation Management, Oracle Transportation Management, Blue Yonder TMS, Manhattan TMS) ориентированы преимущественно на крупные предприятия, требуют значительных инвестиций и долгих сроков внедрения [3]. Это обусловливает актуальность разработки доступного специализированного решения, покрывающего полный жизненный цикл заказа и соответствующего потребностям средних компаний.

Целью исследования является проектирование и разработка информационной системы управления заказами на грузоперевозки, автоматизирующей полный цикл обработки заказа — от подачи клиентом заявки до подтверждения доставки и выставления счёта.

Анализ существующих решений

Проведённый анализ показал, что SAP TM обеспечивает глубокую интеграцию с ERP-контуром, однако характеризуется высокой стоимостью лицензий и требованием к квалификации специалистов. Oracle TM обладает высокой масштабируемостью, но отличается сложностью внедрения. Blue Yonder TMS и Manhattan TMS предоставляют развитые средства оптимизации маршрутов и расширенную аналитику, однако для компаний среднего масштаба избыточны по функционалу и требованиям к инфраструктуре [3, 4]. Результаты сравнительного анализа приведены в таблице 1.

Таблица 1. Сравнительный анализ TMS-систем

Система	Основные возможности	Преимущества	Недостатки
SAP TM	Планирование перевозок, управление заказами	Глубокая интеграция с SAP ERP, высокая функциональность	Высокая стоимость, требует квалифицированных специалистов
Oracle TM	Управление логистическими цепями	Высокая масштабируемость, интеграция с Oracle suite	Сложность внедрения, высокая цена лицензий
Blue Yonder TMS	Оптимизация маршрутов, аналитика	Современные алгоритмы оптимизации	Сложна для небольших компаний
Manhattan TMS	Управление перевозками, отслеживание	Гибкость настройки	Высокая стоимость, длительное внедрение
Разрабатываемая ИС	Полный цикл: заявка → доставка → счёт	Доступность, специализация под СМБ, открытый стек	Меньший функционал для крупных предприятий

В отличие от перечисленных решений, разрабатываемая система ориентирована на низкую стоимость внедрения, простоту эксплуатации и быстрый запуск за счёт использования современного открытого стека технологий.

Обоснование выбора технологий

Для обоснованного выбора серверной платформы проведено сравнение трёх распространённых технологий: Node.js с Express, Python с FastAPI и Java с Spring Boot. Результаты сравнения приведены в таблице 2.

Таблица 2. Сравнение серверных технологий

Критерий	Node.js + Express + TypeScript	Python + FastAPI	Java + Spring Boot
Модель обработки запросов	Асинхронная, event loop	Асинхронная (ASGI)	Многопоточная
Производительность при I/O	Высокая	Высокая	Высокая
Простота разработки	Высокая (единый язык TS)	Высокая	Средняя
Размер экосистемы	Очень широкая (npm)	Широкая (PyPI)	Широкая (Maven)
Единый язык с frontend	Да (TypeScript)	Нет	Нет

На основании проведённого сравнения выбрана связка Node.js, Express и TypeScript. Node.js использует асинхронную событийно-ориентированную модель, эффективно обрабатывающую большое число параллельных I/O-операций [5]. Применение TypeScript на стороне сервера и клиента обеспечивает единый типизированный контракт данных и снижает число ошибок на этапе компиляции.

Клиентская часть реализована на React с использованием Vite в качестве сборщика. Компонентная архитектура React и наличие развитой экосистемы позволяют реализовать ролевые пользовательские интерфейсы и оперативно адаптировать их под изменяющиеся требования [6].

В качестве системы управления базами данных выбрана PostgreSQL версии 16 — реляционная СУБД с полной поддержкой ACID-транзакций, внешних ключей и развитыми средствами индексирования [7]. Принято решение отказаться от объектно-реляционного отображения (ORM) в пользу параметризованных SQL-запросов через библиотеку node-postgres, что обеспечивает полный контроль над структурой запросов, предсказуемость производительности и защиту от SQL-инъекций.

Архитектура системы

Система построена по классической трёхуровневой архитектуре с чётким разделением обязанностей между уровнями [8]. Уровень представления реализован в виде одностраничного приложения на React. Уровень бизнес-логики реализован как REST API на Node.js и Express и построен по схеме «контроллеры — сервисы — репозитории». Контроллеры принимают HTTP-запросы и формируют ответы; сервисный слой инкапсулирует бизнес-правила и управляет переходами состояний заказа; репозиторный слой отвечает за выполнение SQL-запросов и преобразование строк базы данных в объекты переноса данных (DTO). Уровень данных реализован на PostgreSQL. Состав уровней архитектуры представлен в таблице 3.

Таблица 3. Компоненты трёхуровневой архитектуры системы

Уровень	Компонент	Технология	Основные функции
Представление	Одностраничное приложение	React, TypeScript, Vite	Ролевые пользовательские интерфейсы, централизованный API-слой
Бизнес-логика	REST API	Node.js, Express, TypeScript	Аутентификация, авторизация, бизнес-правила, транзакции
Данные	Реляционная СУБД	PostgreSQL 16	Хранение данных, внешние ключи, индексы, транзакции

Для операций, затрагивающих несколько таблиц одновременно (например, назначение водителя и запись в журнал аудита), применяются транзакции СУБД, что гарантирует атомарность изменений. На уровне API реализована поддержка идемпотентности для ключевых изменяющих состояние запросов, что снижает риск возникновения логических дубликатов при повторной отправке запроса клиентом.

Безопасность и ролевая модель

Аутентификация пользователей выполняется на основе JSON Web Token с разделением на access- и refresh-токены; жизненный цикл сеансов ограничен по времени. Авторизация реализована через ролевую модель (RBAC) с четырьмя ролями: Client (клиент), Employee (диспетчер), Driver (водитель) и Admin (администратор). Проверка роли выполняется в middleware-слое на каждом защищённом эндпоинте, что исключает возможность обхода контроля доступа на уровне маршрутов. Журнал аудита фиксирует привилегированные действия пользователей и переходы статусов заказов, что обеспечивает прослеживаемость операций и соответствует требованиям внутреннего контроля.

Функциональные возможности

Система поддерживает полный жизненный цикл заказа, описываемый конечным автоматом со следующей последовательностью состояний: new → accepted → assigned → in_transit → delivered. Из большинства состояний допустим переход в терминальное состояние cancelled. Переходы между состояниями защищены проверками прав роли: клиент инициирует заявку, диспетчер подтверждает её и назначает водителя с транспортным средством, водитель обновляет статус доставки и загружает подтверждение доставки (POD), диспетчер формирует и выставляет счёт.

Результаты и обсуждение

В результате исследования разработан функционирующий прототип информационной системы, покрывающий полный цикл обработки заказа на грузоперевозку. Сборка и развёртывание автоматизированы средствами GitHub Actions: на каждое изменение в основной ветке выполняются статический анализ, компиляция, прогон автоматических тестов и развёртывание на целевую инфраструктуру [9]. Практическая значимость полученного решения заключается в применимости в транспортных подразделениях малого и среднего бизнеса, где требуется быстрое внедрение без значительных инфраструктурных затрат. Перспективы дальнейшего развития связаны с расширением аналитических панелей, внедрением механизмов оптимизации маршрутов, повышением покрытия автоматизированными тестами и оптимизацией запросов к базе данных на высоконагруженных эндпоинтах.

Заключение

Разработанная информационная система управления заказами на грузоперевозки является эффективным инструментом автоматизации операционных процессов для транспортных компаний среднего масштаба. Применение современного стека технологий (Node.js, Express, TypeScript, React, PostgreSQL), трёхуровневой архитектуры и механизмов обеспечения безопасности позволяет сочетать доступность внедрения, гибкость развития и необходимый уровень промышленной надёжности.

Список литературы:

1. Гаджинский А. М. Логистика : учебник. — 20-е изд. — Москва : Дашков и К°, 2020. — 484 с.
2. Сергеев В. И. Логистика: информационные системы и технологии. — Москва : Юрайт, 2022. — 479 с.
3. Иванова Т. Н., Громова А. К. Анализ TMS-систем на российском рынке // Логистика. — 2022. — № 3. — С. 22–28.
4. Blokdyk G. Transportation Management System TMS: A Complete Guide. — 5STARCooks, 2020. — 302 p.
5. Simpson K. You Don’t Know JS: Async & Performance. — Sebastopol : O’Reilly Media, 2015. — 296 p.
6. Wieruch R. The Road to React. — Self-published, 2022. — 350 p.
7. Momjian B. PostgreSQL: Introduction and Concepts. — Boston : Addison-Wesley, 2001. — 440 p.
8. Fowler M. Patterns of Enterprise Application Architecture. — Boston : Addison-Wesley, 2003. — 533 p.
9. Humble J., Farley D. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. — Boston : Addison-Wesley, 2010. — 512 p.
10. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : Doctoral dissertation. — Irvine : University of California, 2000. — URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm (дата обращения: 21.04.2026).
11. Козлов А. С. Проектирование реляционных баз данных для систем управления транспортными заказами // Информационные технологии. — 2019. — № 6. — С. 334–341.
12. Мирошниченко М. А., Стукач О. В. Управление транспортными потоками с использованием информационных технологий // Вестник науки Сибири. — 2020. — № 2 (37). — С. 45–52.