

Н. М. Острянин¹✉, Е. Ю. Воронкин¹

Разработка веб-сервиса для автоматизации и анализа продаж на основе ABC и XYZ-анализа для малого бизнеса

¹Сибирский государственный университет геосистем и технологий,
г. Новосибирск, Российская Федерация
e-mail: nikitaostryanin2018@gmail.com

Аннотация. В статье рассматривается проблема автоматизации товарного учёта и аналитики продаж на малых предприятиях, лишённых доступа к дорогостоящим ERP-системам. Выявлен «информационный разрыв» между оперативным вводом сделок и формированием управленческой отчётности: традиционные электронные таблицы не обеспечивают ни ссылочной целостности данных, ни автоматического пересчёта аналитических матриц. Предложен архитектурный подход на базе стека React, Node.js и PostgreSQL, реализующий одностраничное клиентское приложение (SPA) с REST API-сервером и реляционным хранилищем. Разработана платформа Dashly, объединяющая модули регистрации транзакций, импорта исторических данных из CSV и динамического расчёта матриц ABC/XYZ. ABC-классификация распределяет номенклатуру по вкладу в совокупную выручку (пропорция 80/15/5 %), XYZ-анализ оценивает стабильность спроса через коэффициент вариации (пороговые значения 10 % и 25 %). Описаны алгоритмы серверной агрегации данных PostgreSQL, типобезопасного доступа через Drizzle ORM и валидации входящих запросов посредством библиотеки Zod. Результаты апробации подтверждают высокую эффективность предложенного решения: пиковая загрузка процессора не превысила 1 %, потребление памяти стабилизировалось на уровне 500 МБ, а время формирования аналитического отчёта сократилось с нескольких часов ручного труда до долей секунды. Разработанные методы могут быть масштабированы для внедрения предиктивной аналитики и мультиканальных интеграций.

Ключевые слова: веб-сервис, автоматизация учёта, ABC-анализ, XYZ-анализ, React, Node.js, PostgreSQL, REST API, управление ассортиментом

N. M. Ostryanin¹✉, E. Yu. Voronkin¹

Development of a web service for sales automation and analysis based on ABC and XYZ analysis for small business

¹Siberian State University of Geosystems and Technologies, Novosibirsk, Russian Federation
e-mail: nikitaostryanin2018@gmail.com

Abstract. The article deals with the problem of automating inventory management and sales analytics in small enterprises that lack access to expensive ERP systems. An 'information gap' was identified between operational transaction entry and management reporting: traditional spreadsheets provide neither referential data integrity nor automatic recalculation of analytical matrices. An architectural approach based on the React, Node.js, and PostgreSQL stack is proposed, implementing a single-page client application (SPA) with a REST API server and a relational database. The Dashly platform was developed, combining modules for transaction registration, historical data import from CSV files, and dynamic calculation of ABC/XYZ matrices. ABC classification distributes the product range by contribution to total revenue (80/15/5% ratio), while XYZ analysis evaluates demand stability through the coefficient of variation (threshold values of 10% and 25%). The algorithms of server-side data aggregation in PostgreSQL, type-safe access through Drizzle ORM, and validation of incoming

requests using the Zod library are described. The test results confirm the high efficiency of the proposed solution: peak CPU load did not exceed 1%, memory consumption stabilized at 500 MB, and the time to generate an analytical report was reduced from several hours of manual labor to fractions of a second. The developed methods can be scaled to implement predictive analytics and multichannel integrations.

Keywords: web service, accounting automation, ABC analysis, XYZ analysis, React, Node.js, PostgreSQL, REST API, assortment management

Введение

Слабая интеграция между системами повседневного учёта и модулями глубокой аналитики является существенным препятствием для эффективного функционирования малых предприятий. Зачастую сотрудники вынуждены фиксировать сделки в децентрализованных электронных таблицах или на бумажных носителях, а формирование управленческой отчётности происходит обособленно. Товарный анализ либо полностью игнорируется, либо требует значительных ручных трудозатрат [1].

Отсутствие единого информационного пространства провоцирует расхождения в данных: предпринимателю крайне сложно определить, какие позиции приносят реальный доход, а какие «замораживают» оборотные средства в избыточных запасах. При расширении ассортимента в базах данных накапливаются дублирующие записи, а вычисление метрик нестабильности спроса (XYZ-анализ) становится слишком ресурсоёмким, поскольку любое изменение первичных данных требует повторного ручного пересчёта [2].

Целью данного исследования является разработка и практическая реализация специализированного веб-сервиса Dashly, адаптированного под потребности малого бизнеса и обеспечивающего единый контур регистрации продаж, ведения справочников и автоматического построения аналитических матриц ABC/XYZ без участия аналитика.

Методы и материалы

Для реализации поставленной цели применялись методы системного анализа предметной области, реляционного проектирования баз данных и объектно-ориентированного программирования. Технологической основой решения выбран стек, широко применяемый в современной веб-разработке: библиотека React [3] для клиентского интерфейса, среда Node.js с фреймворком Express [4] для серверной логики и СУБД PostgreSQL [5] для надёжного хранения данных [6].

Архитектура системы строго подчинена стандартам разработки современных веб-платформ: сочетание одностраничного клиентского приложения (SPA) и серверного REST API [7]. Клиентская часть перехватывает действия пользователя и формирует сетевые запросы к бэкенд-узлу, который выступает единственным доверенным посредником при взаимодействии с базой данных через Drizzle ORM [8].

Фундаментом сервиса служит реляционная схема базы данных. Ключевыми таблицами выступают справочник номенклатуры (products) и реестр транзакций (sales). Связи между сущностями строятся через внешние ключи, что гаранти-

рует ссылочную целостность на уровне СУБД. Drizzle ORM транслирует TypeScript-описания таблиц непосредственно в DDL-команды PostgreSQL, исключая ручное написание миграций [9].

Событийная модель обмена между клиентом и сервером опирается на классический цикл «запрос – ответ» протокола HTTP. Браузер формирует REST-запрос к конкретному эндпоинту – например, POST /api/sales для регистрации продажи или GET /api/analytics/abc-xyz для получения матрицы. Сервер валидирует входящие данные посредством схем Zod [10] и возвращает ответ в формате JSON [11].

ABC-классификация опирается на принцип Парето: товары ранжируются по убыванию вклада в совокупную выручку, затем вычисляется накопительная доля. Группа А формирует первые 80 % результата, В – следующие 15 %, С – оставшиеся 5 % [12]. XYZ-анализ классифицирует позиции по стабильности спроса: для каждого товара рассчитывается коэффициент вариации ($CV = \sigma / \mu \times 100 \%$). При $CV \leq 10 \%$ товар относится к классу X (стабильный спрос), при $10 \% < CV \leq 25 \%$ – к Y (умеренные колебания), при $CV > 25 \%$ – к Z (нерегулярный спрос) [13].

Комбинированная матрица ABC/XYZ формирует девять пересекающихся зон: от стратегически важных AX до проблемных CZ. Вся математика выполняется на стороне сервера, что полностью исключает зависимость от вычислительных возможностей клиентского устройства. Для оценки эффективности предложенного решения применялось нагрузочное тестирование на наборах от 100 до 10 000 записей о продажах.

Сравнение предложенного подхода с существующими альтернативами проведено по критериям стоимости владения, скорости развёртывания, масштабируемости и степени интерактивности (табл. 1).

Таблица 1. Сравнение основных подходов к организации учёта продаж

Технология	Достоинства	Недостатки
Электронные таблицы (Excel)	Нет затрат на сервер, простота освоения	Высокий процент ошибок, медленная работа с большими массивами
Монолитные ERP-системы	Максимальная функциональность и надёжность	Высокая стоимость лицензий и внедрения
Веб-сервис (React / Node.js)	Быстрый старт, интерактивность, масштабируемость	Требует грамотного проектирования API и архитектуры

Результаты

Практическая реализация разработанной архитектуры позволила создать полнофункциональный веб-сервис Dashly, охватывающий весь контур от первичного ввода данных до визуализации аналитической матрицы. При поступлении транзакции серверный модуль Node.js выполняет валидацию входных данных, сохраняет запись в PostgreSQL через Drizzle ORM и инвалидирует клиент-

ский кэш React Query, обеспечивая актуальность аналитических представлений без ручного обновления страницы.

Маршрутизация запросов реализована следующим образом. Обращение к эндпоинту GET /api/analytics/abc инициирует на сервере группирующий SQL-запрос для суммирования выручки по каждому идентификатору товара, сортировку по убыванию финансового вклада, вычисление накопительной доли и разбиение по категориям согласно пропорции 80–15–5. Запрос GET /api/analytics/xyz дополнительно группирует продажи по месячным интервалам, рассчитывает среднее значение и стандартное отклонение для каждого товара и определяет коэффициент вариации. Итоговая матрица объединяет оба фильтра и возвращается клиенту в формате JSON [11].

Функция массового импорта обеспечивает загрузку исторических архивов в формате CSV. Алгоритм осуществляет построчное чтение файла с потоковой обработкой (stream-based parsing), верифицирует названия колонок и проверяет корректность сумм и форматов дат посредством схем Zod. Некорректные строки отклоняются с формированием подробного отчёта об ошибках. Валидные записи сохраняются в PostgreSQL в рамках единой транзакции: если хотя бы одна строка не проходит валидацию, откатывается весь пакет, предотвращая частичную загрузку данных.

Клиентский интерфейс матрицы ABC/XYZ реализован в виде интерактивной сетки 3×3, где каждая ячейка окрашивается в зависимости от статуса: стратегически важные товары AX выделяются зелёным цветом, проблемные CZ – красным. Пользователь взаимодействует с интуитивно понятным графиком, при этом данные запрашиваются через REST API и кэшируются посредством React Query, что соответствует рекомендациям по проектированию отзывчивых интерфейсов.

Комплексное тестирование охватило все подсистемы платформы. Строгой проверке подверглась надёжность защиты REST-маршрутов: неавторизованные запросы к приватным эндпоинтам отклоняются с кодом 401, сессионная кука корректно выдаётся при успешной аутентификации. Результаты основных сценариев тестирования приведены в табл. 2.

Таблица 2. Основные сценарии тестирования платформы Dashly

Проверяемый сценарий	Ожидаемый результат	Фактический результат
Аутентификация и защита маршрутов	Выдача сессионной куки, блокировка без неё	Выполнено
Импорт повреждённого CSV	Валидатор Zod отклоняет файл с сообщением об ошибке	Выполнено
Алгоритм ABC-классификации	Корректное ранжирование по пропорции 80/15/5 %	Выполнено
Алгоритм XYZ-классификации	Верный расчёт коэффициента вариации (CV)	Выполнено

Нагрузочное тестирование продемонстрировало высокую эффективность выбранной архитектуры. Пиковые показатели загрузки центрального процессора Node.js не превысили 1 %, а расход оперативной памяти стабилизировался на уровне 500 МБ. Данный факт свидетельствует о том, что веб-сервис готов к промышленной эксплуатации даже на бюджетных виртуальных серверах (VPS) с минимальными тарифными планами, что критически важно для малого бизнеса с ограниченным ИТ-бюджетом.

Результаты сравнительного анализа эффективности до и после внедрения платформы Dashly приведены в табл. 3.

Таблица 3. Сравнительный анализ показателей до и после внедрения веб-сервиса

Параметр оценки	До внедрения Dashly	После внедрения Dashly
Формирование ABC/XYZ-отчётов	Многочасовой ручной расчёт в таблицах	Доли секунды в браузере (автоматически)
Загрузка исторических данных	Долгое ручное копирование строк	Моментальный импорт CSV с валидацией
Вероятность ошибки ввода	Критически высокая	Сведена к минимуму (валидация Zod)
Трудозатраты аналитика	Высокие	Нулевые на этапе расчёта матриц
Перспективы масштабирования	Практически отсутствуют	Неограниченные благодаря модульной архитектуре

Обсуждение

Полученные в ходе тестирования данные подтверждают состоятельность выбранной архитектуры. Сравнение предложенного метода с монолитными ERP-системами и электронными таблицами выявило существенное преимущество в части доступности и скорости развёртывания. Отказ от тяжёловесных монолитных систем в пользу легковесного REST API с реляционной СУБД снизил порог входа для малого бизнеса, сохранив при этом возможности промышленных аналитических систем [14, 15].

Проблема обработки некорректных данных при импорте CSV была решена внедрением строгой валидации на уровне схем Zod и транзакционной модели PostgreSQL, что соответствует лучшим практикам разработки надёжных серверных приложений [16]. Переход на потоковую обработку CSV-файлов (stream-based parsing) позволил полностью исключить блокировку основного потока выполнения Node.js при загрузке объёмных архивов.

Основным ограничением предложенного метода является необходимость первоначальной настройки серверной инфраструктуры (VPS, PostgreSQL, переменные окружения). Однако затраты на развёртываниекратно ниже стоимости лицензий и поддержки корпоративных ERP-решений, а модульная архитектура

обеспечивает горизонтальное масштабирование без переработки ядра системы [17].

Заключение

В рамках проведённого исследования разработана, протестирована и введена в эксплуатацию платформа Dashly – специализированный веб-сервис для автоматизации учёта продаж и расчёта аналитических матриц ABC/XYZ для малого бизнеса. Успешно решена проблема «информационного разрыва» между оперативным вводом транзакций и формированием управленческой аналитики. Разработанный серверный модуль на Node.js обеспечил сквозную автоматизацию расчёта классификационных матриц, полностью исключив ручной труд аналитика.

Результаты нагрузочного тестирования подтверждают многократное повышение эффективности: время формирования ABC/XYZ-отчёта сокращено с нескольких часов до долей секунды, пиковая нагрузка CPU не превысила 1 %, что позволяет эксплуатировать сервис на бюджетном VPS. Реализованная валидация входящих данных через Zod и транзакционная модель PostgreSQL обеспечивают целостность данных при массовом импорте CSV-архивов.

Дальнейшие исследования будут направлены на расширение пула аналитических моделей: внедрение RFM-анализа для сегментации клиентской базы, создание механизма автоматических уведомлений при критическом смещении товара между группами матрицы, а также интеграцию алгоритмов машинного обучения для предиктивной аналитики спроса.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. СТО СМК СГУГиТ 8.5–39–2025. Государственная итоговая аттестация выпускников СГУГиТ. Структура и правила оформления. – Новосибирск : СГУГиТ, 2025. – 67 с.
2. Панина У. Е., Попова Е. В. ABC/XYZ-анализ как способ анализа товарооборота розничного предприятия. – URL: <https://cyberleninka.ru/article/n/abc-xyz-analiz-kak-sposob-analiza-tovarooborota-rozничного-predpriyatiya> (дата обращения: 25.05.2026).
3. Документация React. – URL: <https://react.dev/> (дата обращения: 15.04.2026).
4. Документация Express.js. – URL: <https://expressjs.com/> (дата обращения: 15.04.2026).
5. Руководство по PostgreSQL. – URL: <https://www.postgresql.org/docs/> (дата обращения: 15.04.2026).
6. Фрэнкс Б. Укрощение больших данных: как извлекать знания из массивов информации с помощью глубокой аналитики. – Москва : Манн, Иванов и Фербер, 2014.
7. Филдинг Р. Архитектурные стили и дизайн сетевых программных архитектур / Р. Филдинг // Information and Computer Science: сайт. – URL: https://ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (дата обращения: 15.04.2026). – Текст : электронный.
8. Документация Drizzle ORM. – URL: <https://orm.drizzle.team/> (дата обращения: 15.04.2026).
9. Документация TypeScript. – URL: <https://www.typescriptlang.org/> (дата обращения: 15.04.2026).
10. Документация Zod. – URL: <https://zod.dev/> (дата обращения: 15.04.2026).

11. Брей Т. Формат обмена данными JavaScript Object Notation (JSON). RFC 8259 / Т. Брей // RFC Editor : сайт. – URL: <https://www.rfc-editor.org/rfc/rfc8259> (дата обращения: 15.04.2026).
12. Бузмакова Е. Е. Методы классификации запасов предприятия // КиберЛенинка. – URL: <https://cyberleninka.ru/article/n/metody-klassifikatsii-zapasov-predpriyatiya> (дата обращения: 25.05.2026).
13. Деревянко А. А., Рябова Т. В. Сущность и направления использования ABC-анализа // КиберЛенинка. – URL: <https://cyberleninka.ru/article/n/suschnost-i-napravleniya-ispolzovaniya-avs-analiza> (дата обращения: 25.05.2026).
14. Фаулер М. Шаблоны корпоративных приложений / М. Фаулер. – Москва : Вильямс, 2016.
15. Ричардсон К. Микросервисы. Паттерны разработки и рефакторинга / К. Ричардсон. – Санкт-Петербург : Питер, 2019.
16. Киченко Л. П., Шалимов В. Совершенствование системы управления запасами на предприятиях оптовой торговли // КиберЛенинка. – URL: <https://cyberleninka.ru/article/n/sovershenstvovanie-sistemy-upravleniya-zapasami-na-predpriyatiyah-optovoy-torgovli> (дата обращения: 25.05.2026).
17. Клеппман М. Высоконагруженные приложения. Программирование, масштабирование, поддержка / М. Клеппман. – Санкт-Петербург : Питер, 2018. – 640 с.
18. Durski K., Murlewski J., Makowski D., Sakowicz B. Warehouse Management System in Ruby on Rails Framework on Cloud Computing Architecture // Радиоэлектроника и информатика. – 2011. – № 4. – URL: <https://cyberleninka.ru/article/n/warehouse-management-system-in-ruby-on-rails-framework-on-cloud-computing-architecture> (дата обращения: 25.05.2026).
19. Документация Node.js. – URL: <https://nodejs.org/ru/docs/> (дата обращения: 15.04.2026).
20. Официальная документация Tailwind CSS. – URL: <https://tailwindcss.com/docs> (дата обращения: 15.04.2026).

© Н. М. Острянин, Е. Ю. Воронкин, 2026