Вячеслав Джагаев

Backend-инженер (NestJS / TypeScript) · Углубляюсь в Go

Волгоград, Россия · Открыт к переезду (EC) · slava.dzhagaev@proton.me · vdzhagaev.tech · github.com/vdzhagaev

SUMMARY

Backend-инженер с **3+ годами коммерческого опыта**, **единственный архитектор** продовой платформы на NestJS/MongoDB, построенной на **чистой архитектуре** - доменные сущности без знания о персистентности, узкие read/write-порты вместо монолитного репозитория, и гибридная система доставки событий для eventual consistency между агрегатами. Умею диагностировать продовые инциденты до первопричины и доставлять фикс в реальных условиях (особенности легаси-схем, лимиты сторонних API, хранилища с одним writer'ом). Активно развиваюсь в **Go** для self-hosted API-first сервисов и углубляюсь в **Rust** через pet-проекты. В поиске backend-роли в **EC** с поддержкой релокации.

НАВЫКИ

CORE

TypeScript, NestJS, Node.js, Go, SQL, Rust, Python

BACKEND

REST API, Чистая архитектура, DDD, Domain events, Event sourcing / outbox,

CQRS-подобные read-модели, BullMQ / Redis, Playwright/CDP-автоматизация, WebSockets, FastAPI

FRONTEND

Angular, React, RxJS, Svelte

ДАННЫЕ

MongoDB, PostgreSQL, SQLite, Миграции

ИНФРА

Docker, docker-compose, Bitbucket Pipelines, Монорепозиторий, Linux, Git

ОПЫТ

Full-Stack инженер, Comrade Digital Marketing Agency

Март 2023 - настоящее время

Удалённо

Платформа для link-building / мониторинга бэклинков (NestJS + MongoDB, pnpm-монорепозиторий) **- единственный архитектор и основной автор**

- Спроектировал **чистую архитектуру** с нуля: доменные сущности без знания о персистентности, инварианты закреплены в конструкторах и покрыты юнит-тестами без базы данных, плюс узкие read/write-порты на домен вместо одного толстого

репозитория - дашборды и отчёты получают денормализованные, query-оптимизированные данные, не затрагивая write-side инварианты.

- Построил **гибридную систему доставки событий** (change-stream watch + polling fallback с lease-based захватом) для eventual consistency между агрегатами, сохраняя каждое событие как outbox для воспроизведения инцидента из журнала событий, а не восстановления по логам.
- Диагностировал регрессию bulk-записи (40с на ~1150 строк) как **отсутствующий индекс**, а не архитектурное узкое место - до того как команда согласилась на многодневный рефакторинг с разделением очередей; реальный фикс сократил время записи до <1с.
- Переработал bulk-запись скрейпинга на 30 тыс. строк, которая молча зависала на 7-21 минуту без ретраев, в **чанкованные идемпотентные upsert-батчи** с таймаутами на чанк и ретраем на уровне очереди.
- Реализовал **FIFO-семафор**, ограничивающий доступ к общему пулу browser context'ов для 14 параллельных скрейпинг-воркеров, вместе со случайными паузами взаимодействия, чтобы повторные запросы не читались антибот-системами как бот.
- Поддерживаю три отдельных сервиса с персистентной сессией **браузерной автоматизации** вместо одного универсального скрейпера - каждый со своим headful Chrome-сайдкаром (Xvfb + noVNC), в который оператор логинится один раз; сессия держится живой через адаптивный auth-health prober и постоянную имитацию активности мыши/скролла - защита именно от тихого разлогина или флага «неактивной» сессии, а не от объёма запросов.
- Спроектировал **AI-augmented pipeline** для извлечения данных: правила по каждой площадке лежат в data-driven каталоге, а не хардкодятся, и питают составной, cache-friendly промпт, который один раз на шаблон страницы находит стабильный CSS-селектор - дальше переиспользуется без единого нового вызова AI, подтверждено на практике: 5/5 полей из кэша на втором аккаунте.

Инструмент редакторского workflow (NestJS + MongoDB) **- ведущий контрибьютор, автор-основатель**

- Нашёл первопричину двух продовых инцидентов в fire-and-forget доставке вебхуков (гонка данных и молчаливая потеря сообщения), затем спроектировал **transactional outbox** - атомарный статус-флаг, lease-based polling fallback, backoff с джиттером, dead-lettering.
- Нашёл путь bulk-approve, который обходил новый outbox целиком, молча пропуская уведомления - исправил транзакционным снапшоттингом в том же query-пути вместо дублирования outbox-логики.
- Реализовал соответствующую фичу на Angular (server-side row model гриды) end-to-end поверх этого бэкенда.

Общая frontend-платформа (Angular) **- контрибьютор в отдельных модулях**

- Построил real-time **WebSocket-слой** для набора функций совместного редактирования - от первой реализации до последующей миграции и выравнивания event-схемы с бэкендом.
- Адаптировал существующий **undo/redo-движок** под второй домен данных через

generic adapter-интерфейс, не трогая ядро движка - включая обработку TTL soft-delete edge case, который иначе мог тихо ломать восстановление.

Мониторинг-инфраструктура (Python + FastAPI)

- Построил прослойку синхронизации, которая сверяла внутреннюю базу клиентов с мониторами в Uptime Kuma и создавала, обновляла или удаляла мониторы по дифу - каждый с нужным прокси, каналами уведомлений и проверками под конкретного клиента, заменив ручное ведение мониторов.

Также поддерживал легаси-инструмент на Laravel (PHP), мигрировал операторскую панель с server-rendered HTML на Vite + React SPA, докеризировал сервисы и вёл Bitbucket Pipelines CI/CD по всему монорепозиторию.

Разработчик ПО (IT), Волгоградский государственный архитектурно-строительный университет

`Август 2020 - Январь 2024`
Волгоград, Россия

- Поддерживал **платформу дистанционного обучения** университета для студентов и сотрудников.
- Разрабатывал внутренние инструменты для упрощения административных процессов.
- Поддерживал инфраструктуру **онлайн-приёмной кампании** на протяжении нескольких приёмных циклов.

ИЗБРАННЫЕ ПРОЕКТЫ

watchlight `GO · В РАЗРАБОТКЕ` `github.com/vdzhagaev/watchlight`

Сервис мониторинга аптайма, написанный соло. Разделил scheduler на три канала с разной семантикой backpressure (ограниченные буферы job/result, небуферизованная сигнализация) вместо одной общей очереди, подобрал фиксированный пул воркеров и сериализовал все записи в SQLite через единственного consumer'a - из-за ограничения «один writer даже в WAL». Graceful shutdown вытекает работающие задачи через ступенчатую последовательность отмены, прежде чем жёсткий таймаут форсирует завершение. Миграции, метрики и нагрузочное тестирование - в процессе.

pokeroll `TAURI · RUST · БЕТА` `github.com/vdzhagaev/pokeroll`

Кроссплатформенный десктоп-виджет (Tauri 2, SvelteKit + Rust), который выбирает случайного покемона из PokéAPI и рендерит карточку с сохранённым локальным прогрессом. Выпущен как публичная бета со сборками под Windows/macOS/Linux - самый законченный проект на сегодняшний день.

ОБРАЗОВАНИЕ

Магистратура - Информационные системы и технологии (09.04.02)

Волгоградский государственный архитектурно-строительный университет · 2023-2026

Бакалавриат - Информационные системы и технологии (09.03.02)

Волгоградский государственный архитектурно-строительный университет · 2019-2023

ЯЗЫКИ

Русский родной

Английский A2 · Элементарный