

Daniil Vdovin

Backend / Fullstack Developer

+7 (982) 611-98-00 daniilvdovin4@gmail.com Telegram: @v_d_v_7 daniilvdovin.com career.habr.com/vdv007
github.com/VDV001 Ekaterinburg, Russia from 150 000 ₺

ABOUT ME

Backend/Fullstack developer focused on Go and TypeScript. 1.5+ years of commercial experience, running several parallel production projects on Clean Architecture + DDD + TDD - from AI CRM and estimation to ERP modules. I write code that others can understand: clear module boundaries, tests, documentation. I read others' code and ask questions before implementing.

AI is part of my daily workflow (Claude Code CLI + custom MCP servers): architecture planning, TDD, parallel verification via subagents, custom skills for the team. I'm growing towards complex backend systems and a lead role: distributed architectures, reliability, performance, mentoring.

Ekaterinburg, Russia 1.5+ years commercial Russian - Native English - B1-B2, technical, actively improving
UGGU, Big Data Specialist (Computer Vision)

TECH STACK

Backend	Go · Python · FastAPI · SQLAlchemy 2 · Alembic · Pydantic · Node.js · NestJS · chi · Gin · gRPC · PostgreSQL · Redis · pgx/v5 · golang-migrate · REST API · WebSocket · SSE · SMTP / IMAP · JWT / OAuth2 · bcrypt · CQRS · Event Sourcing · DDD · Clean Architecture · Repository / Unit of Work · TDD
Frontend	TypeScript · JavaScript · React 19 · Vite · Next.js 16 · App Router · Server Components · Zustand · SWR · TanStack Query · React Hook Form · Zod · Tailwind CSS · shadcn/ui · Three.js · PWA · i18n
DevOps & Infra	Docker · Docker Compose · Docker Swarm · Kubernetes · Helm · kustomize · kind · Terraform / OpenTofu · tfint · GitHub Actions · CI/CD · Nginx · CloudFlare · Prometheus · Grafana · Loki · OpenTelemetry · MinIO / S3 · LocalStack · golangci-lint · Dependabot · CodeQL · govulncheck · gitleaks · SBOM · supply-chain gates · GoReleaser
Testing	testify · gomock · pytest · Vitest · Jest · Playwright · mutation testing · k6 · httptest · race detector · table-driven tests · negative control · fault injection · self-testing gates · проверка на копии живых данных
AI & Tools	Claude Code · Codex · MCP Servers · FastMCP · Claude Code Skills · Anthropic SDK · OpenAI · Groq · Gemini · Ollama · Prompt Engineering · RAG · pgvector · embeddings · semantic search · Subagent-Driven Development · prompt-injection protection · agent reliability gates · AI-driven TDD

EXPERIENCE

Multimedia Video Systems - Fullstack Developer

October 2025 - Present

- **Lead contributor on a client product:** about 2/3 of commits over three months across a team of eight
- End-to-end presale flow **brief** → **spec** → **bill of materials** → **proposal**: document versions and a public link for client approval
- **Normalized equipment catalog** with an MCP server on FastMCP: an item is published only with its key specs filled in, and the catalog has a readiness metric
- Standard solutions and an **equipment selection wizard** by room and subsystem
- Proposal export to **1C (CommerceML 2.05)**
- Claude Code (CLI) as the main AI tool: MCP servers (Serena, Context7, Playwright, Figma), **AI-driven TDD**, custom skills for debugging and code review

Development methodology. Every T-xxx task comes with a spec and tests built on CJM scenarios. Architecture decisions are recorded in ADRs.

Architecture & design. FastAPI backend with api / models / schemas / repositories layers, Alembic migrations. A separate catalog MCP server on FastMCP.

Backend (Python / FastAPI). SQLAlchemy 2, Pydantic 2, JWT + Argon2id, multiprocessing via gunicorn and UvicornWorker. Proposals in three variants: Base, Optimal, Premium.

Frontend (React / TypeScript). React 19, Vite 7, TanStack Query, Zustand, React Hook Form + Zod, Tailwind 4. Equipment selection wizard and the standard solutions window.

AI tools. The brief becomes a spec through Codex, and a human checks the result. The catalog is filled through Claude Code and the MCP server.

DevOps & infrastructure. CI in GitHub Actions, S3-compatible storage (MinIO), PostgreSQL 16.

Testing. pytest, Vitest + RTL, Playwright e2e on CJM scenarios, mutation testing.

Integrations. Proposal export to 1C (CommerceML 2.05), MinIO for files, public links to specs and proposals for the client.

Anvirtekhno - Fullstack Developer

November 2024 - Present

- **1+ year of stable remote work** in a team: regular delivery, **code review approved on first-second attempt**
- Auto-generated TypeScript clients from OpenAPI: **eliminated entire class of errors** in frontend/backend sync
- Built PWA with offline mode and push notifications - **app works without internet and installs as native**
- Connected LLM providers (OpenAI, Ollama) to production with **fallback strategy and token monitoring**
- Built PostgreSQL + MinIO auto-backup system with **encryption and offsite sync**
- Claude Code (CLI) as primary AI tool: MCP servers (Serena, Context7, Playwright, Figma), **AI-driven TDD**, custom skills for debugging and code review

Development Methodologies. API-first approach: OpenAPI spec → auto-generated TypeScript clients. TDD for business logic. RDD for feature planning before implementation. TypeScript types from DB schemas.

Architecture & Design. Clean Architecture on NestJS modules. REST API with auto-documentation from decorators. Event-driven approach for loosely coupled components. Structure ready for microservice extraction.

Backend (NestJS / TypeScript). NestJS as primary stack. JWT auth with refresh tokens. Background jobs and scheduled tasks. LLM providers (OpenAI, Ollama) with fallback strategy and token monitoring. Telegram Bot integration.

Frontend (React / TypeScript). Next.js PWA with offline mode and push notifications. Zustand + SWR for state management. React Hook Form + Zod for validation. Component library in Storybook. i18n and theming.

AI Tools. Claude Code (CLI) for code review, debugging and refactoring. MCP servers: Context7 (docs), Playwright MCP (E2E), Figma MCP (design). TypeScript language server for code intelligence. Custom skills.

DevOps & Infrastructure. Docker + Docker Compose, multi-stage builds. Auto-backups PostgreSQL + MinIO with encryption and offsite sync. CI/CD in GitHub Actions, Dependabot. Observability: Prometheus, Grafana, Loki.

Testing. Jest unit tests for TypeScript/Node.js. E2E with Playwright. Mocking for LLM providers and external services. Tests integrated in CI/CD: PRs don't merge without green matrix.

Integrations. LLM providers (OpenAI, Ollama) with fallback and monitoring. Telegram Bot API. S3/MinIO for storage and backups. Composio Gmail API for notifications.

PROJECTS

Deal Sense - AI for Tender Teams

5725 commits, 5636 of them mine (measured 30 Oct) · v0.334.0 · 5 AI providers

Product-grade AI tool for tender evaluation: go/no-go scoring, commercial proposal generation, and tender document analysis. Go 1.26 backend + React/Vite/TypeScript. Clean Architecture, DDD, pluggable LLM with 5 providers and runtime switching.

Clean Architecture + DDD on Go 1.26: domain invariants in the domain layer, ports in usecase, adapters in infra. Pluggable LLM provider as a domain interface. Modular structure ready to scale. golangci-lint v2 with the modernize analyzer in CI

5 LLM providers (Anthropic Claude / OpenAI / Gemini / Groq / Ollama) with runtime switching from config. Rubric-based tender scoring (go/no-go), proposal generation from templates, tender document processing, two-stage prompt-injection protection

Docker Compose, GitHub Actions CI/CD (build + race tests + linter), multi-stage builds. Local development in OrbStack. Versioning via ldflags

Table-driven tests, go test -race. Coverage of domain logic, LLM response parsing, scoring, and the usecase layer. Tests gate merges in CI

Academic Secretary IS

1640 commits · v0.223.0 · in production

Modular monolith on Go: DDD + Clean Architecture, encapsulated domains. In-memory EventBus with Kafka abstraction. CQRS + Event Sourcing + Repository + Unit of Work. 22 PostgreSQL 17 SQL migrations. Results: Redis cache for hot reads via the Decorator pattern, monitoring from scratch, encrypted auto-backups, 189 releases in 4 months.

Modular monolith on Go: DDD + Clean Architecture, CQRS, Event Sourcing, Repository + Unit of Work. In-memory EventBus with Kafka abstraction. gRPC interfaces. 22 PostgreSQL 17 SQL migrations with versioning and rollback

Gin: middleware chains, correlation IDs. OAuth 2.0 + JWT (full security claims, timing-attack prevention, bcrypt cost 14, rate limiting 5 req/15 min, 6 security headers). WebSocket chats. Predictive analytics. Composio Gmail API + Telegram Bot. 1C ERP integration. i18n ru/en/fr/ar with RTL

Prometheus + Grafana (latency/error rate/throughput). Loki + Promtail (JSON logs). Grafana Tempo + OpenTelemetry (distributed tracing). Telegram alerts: CPU/RAM/Disk/API errors/backup. Docker + Docker Swarm. GitHub Actions CI/CD. Nginx + CloudFlare. Auto-backups PostgreSQL & MinIO with encryption

testify unit tests (Go), k6 load tests, Playwright E2E. AI-driven TDD: tests before implementation, parallel verification. Documentation: architecture, API reference, deployment, security. GitHub Projects: 40+ Issues. .taskmaster - AI-assisted task management

Floq - AI Sales Assistant

731 commits (measured 30 Oct) · v0.118.0 · 4 AI providers

Full-stack B2B CRM with AI-powered cold outreach, multi-channel sequences (Email + Telegram + phone), and automated lead qualification. Clean Architecture, DDD, 20 PostgreSQL migrations, 51 API endpoints.

Clean Architecture (handler → usecase → repository → domain), DDD with value objects (LeadStatus, Channel, OutboundStatus), port/adaptor interfaces, compile-time interface checks, TxManager for transactions, DTO mapping without json tags in domain layer, 20 SQL migrations (golang-migrate), anti-corruption layers (ProspectReader, LeadCreator)

51 REST API endpoints on Go 1.26 + chi v5, JWT auth (access + refresh tokens), pluggable AI provider (Claude/OpenAI/Groq/Ollama) with runtime switching from DB, SMTP sending via mail.ru/Yandex/Gmail (TLS 465 + STARTTLS 587), IMAP polling for inbound emails (go-imap/v2), Telegram MTPROTO client (gotd/td) for personal account outreach, built-in email verifier (MX/SMTP probe/disposable filter), 2GIS API parser + website email scraper, AI chat assistant with DB context injection, prompt feedback loop, tracking pixel for open/reply tracking, cron-based follow-up system

Docker Compose (PostgreSQL 18 + Redis 8 + Ollama + Backend), GitHub Actions CI/CD (Go build/test + Next.js lint/build), multi-stage Alpine image, OrbStack for local development

130+ unit tests across 16 packages (testify, race detector), coverage: auth (JWT middleware, token generation), AI client (extractJSON, prompt resolution, qualification parsing), email verification, settings DTO mapping, chat system prompt building

EstimatePro

414 commits · v0.15.0 · 145 tests

Collaborative project estimation platform. Modular Monolith + Clean Architecture + DDD on Go 1.26 and Next.js 16 with PostgreSQL 18, Redis 8, MinIO. Telegram bot Esti with LLM parsing and two-stage prompt protection architecture.

Modular Monolith with 7 modules (auth, project, document, estimation, notify, bot, ws). Clean Architecture: domain → usecase → handler. DDD: entities with business logic, sentinel errors, value objects. Adapters in composition root, zero cross-module imports. 8 PostgreSQL migrations via goose

57 REST endpoints (Chi router). JWT auth with token rotation in Redis. OAuth2 Google/GitHub. PERT estimation aggregation. WebSocket real-time. Telegram bot Esti: 4 LLM providers (Claude/OpenAI/Grok/Ollama), 12 intents, inline keyboards, conversation memory, two-stage architecture (classifier + personality formatter). Notifications: in-app + email + Telegram

Docker Compose (PostgreSQL 18, Redis 8, MinIO, Go backend). GitHub Actions CI/CD. CORS allowlist, rate limiting (10 req/min on auth), MaxBodySize middleware, RequireProjectMember middleware for IDOR protection

145 test functions across 25 files. Table-driven tests, function-field mocks. Coverage: all usecases, domain logic, handlers, LLM providers, Telegram client (httpptest), webhook, session flows. TDD for estimation (parser + aggregation)

Lexis

148 commits · v0.17.0 · 4 AI providers

AI-powered language tutor: Go modular monolith (Clean Architecture, DDD) + Next.js 16 frontend. 4 AI providers, SM-2 spaced repetition, JWT auth with token rotation, PostgreSQL, Redis.

Modular monolith (4 modules: auth, tutor, vocabulary, progress), layered domain → usecase → handler → infra. ProviderRegistry as domain interface. Auto-migrations via golang-migrate with embedded SQL. In-process event bus (pub/sub) for cross-module communication. RFC 7807 Problem Detail errors

JWT access/refresh token rotation with reuse detection and user-level blacklist via Redis. 4 AI providers (Claude, GPT-4o, Qwen, Gemini) with SSE streaming. SM-2 spaced repetition for vocabulary. 5 exercise modes (chat, quiz, translate, gap, scramble). Domain-level validation with sentinel errors

Docker Compose (PostgreSQL 18, Redis 8). GitHub Actions CI: golangci-lint, go test -race, Next.js build, Playwright e2e. Ldflags version injection. MaxBodySize, RequireJSON, RateLimit (prefix-based, fail-closed for login) middleware

83 unit tests across 12 packages. Gomock for auth service, hand-written mocks for handler tests. Coverage includes: token rotation, SM-2 algorithm, goal progress, handler CRUD, eventbus concurrency, middleware auth. Playwright e2e for auth flow

kb-engine - Knowledge Base Engine (OSS)

392 commits, 58k lines of Go, 246 test files (measured 30 Oct) · 28 releases, 12 CI jobs

Open-source Go engine: CLI, terminal UI and an embedded React dashboard over a single usecase layer, so the surfaces cannot drift apart. 28 releases with binaries for six platforms and SBOMs. Kubernetes deployment (kustomize + a Helm chart) and infrastructure in Terraform. AGPL-3.0-or-later with a commercial alternative.

Clean Architecture + DDD: a domain free of clocks and randomness, repository interfaces declared in the usecase layer. One rule lives in one place - terminal and web read through a shared query layer, so the views cannot disagree. Two architecture maps with file:line anchors verified by a validator in CI: the engine at 84 nodes and 48 scenarios, the cloud layer at 35 nodes and 8 scenarios

Catalog with integrity audits, link-health checks, and two-way jsonl ↔ xlsx sync plus duplicate protection right on the write path. Four-layer search: exact match, transliteration, edit distance, and a semantic layer over vector embeddings (1,487-entry index, bge-m3 model, service called over HTTP). The call was made by measurement: the semantic layer resolves one query in twelve, so no model was embedded into the binary - four direct dependencies and a 13.5 MB image stayed unchanged

Kubernetes: kustomize (base plus overlays) and a Helm chart, both installed into one kind cluster in CI and compared by the objects in the cluster. Liveness and readiness probes are split deliberately: a corrupted catalog removes the pod from the Service without restarting it. Terraform: an S3 bucket with versioning and encryption, a read-only IAM role, state in the bucket with locking (verified by planting a foreign lock), tfint in CI. A /metrics endpoint in Prometheus format without a client library, plus Prometheus and Grafana with a dashboard as code. GitHub Actions, 12 jobs: race-detector tests, an 80% coverage floor, golangci-lint with modernize, govulncheck, CodeQL, gitleaks across the full history, and a dependency-age quarantine self-tested against a real incident. Verified against LocalStack and MinIO; real AWS and behaviour under load were not tested

189 test files with a coverage floor in CI. Every gate is tested against a planted violation