

Maxim Eliseev

Senior Backend Engineer

Python · Go · Rust

7 years of experience · Open to offers

Senior Backend Engineer with 7 years of commercial experience building backend systems in Python and Go. I specialise in distributed-systems architecture, real-time data processing and geospatial computation. I have worked on a fleet-monitoring system for 90,000+ vehicles and an agricultural platform, owning architecture decisions, performance optimisation and service reliability. I also run an independent open-source project in Rust.

Experience

Senior Backend Engineer Python / Go — Magnifio

2023 — 2026

- Fixed a critical performance problem: field-processing calculations exceeded the 40-minute Celery task limit; after optimisation, runtime dropped to roughly 5 minutes
- Found O(n²) quadratic complexity in pairwise polygon-intersection checks; used the STRtree spatial index and geometry pre-filtering to cut expensive operations
- Designed and shipped a Go microservice to production for heavy geospatial computations, extracting them from Django/Celery into a standalone backend component
- Developed new Django-backend modules, tuned PostgreSQL/PostGIS, set up CI/CD and Docker; implemented field and crop-rotation analytics

Backend-developer Python — GISDEV

2019 — 2023

- Built the backend of a fleet-monitoring system for 90,000+ vehicles and streaming GPS-telemetry processing services in Python, aiohttp and asyncio (clients include Gazprom Neft)
- Implemented live coordinate and status updates in the browser over WebSockets for real-time map tracking
- Optimised spatial queries in PostgreSQL/PostGIS, built background tasks on Celery/Redis, maintained and refactored the Django monolith

Independent project

Backend-developer `Rust` — Konstruct

`Nov 2025 — present`

Independent side project. An open-source messenger with post-quantum encryption that knows only what delivering a message requires: an account is a keypair — no phone number, no email.

- Implemented the secure-messaging protocol in Rust: X3DH, Double Ratchet and post-quantum key agreement (PQXDH, ML-KEM); used ChaCha20-Poly1305 and hybrid Ed25519/ML-DSA-65 signatures
- Split the backend into identity, messaging, signaling and media services; message delivery over a bidirectional gRPC stream on QUIC with automatic HTTP/2 fallback
- Published the source code and shipped iOS (TestFlight) and Android (alpha) clients

github.com/konstruct-msg · protocol specification

Key skills

PYTHON	Django · FastAPI · asyncio · aiohttp · Celery
GO	microservices, geo computations
RUST	Tokio · gRPC · async
DATA	PostgreSQL · PostGIS · Redis
ARCHITECTURE	microservices · distributed systems · system design
PROTOCOLS	REST · WebSockets · gRPC · QUIC
INFRA	Docker · CI/CD · Linux · Git

Contact

TELEGRAM	@maxeliseyev	EMAIL	maxim.elis@icloud.com
LINKEDIN	in/maximeliseyev	GITHUB	maxeliseyev