Zhavlonbek Artykov

Full-stack Developer

DJANGO · POSTGRESQL · CELERY · LINUX

Writing code since 7th grade. I design the architecture, write the backend and keep it running in production myself.

Tashkent · on-site or remote · open to offers

@unique_developer · artykovzh@gmail.com · linkedin.com/in/artykovzh · unique-developer.uz

ABOUT

I built my first websites in 7th grade and never left development since. I started with landing pages at an agency, and now I run mstar.uz, the ticketing platform of the largest event organiser in Tashkent. Django, PostgreSQL, Celery, Redis, WebSocket. One developer, live since 2024.

I own all of it: the data model, the REST API, the role model, payment integrations, the server, deployment, backups and support. On the same codebase I launched a second instance under a different brand, with its own database, its own environment and an end-to-end action log.

Alongside that I earned a Computer Science degree at Zhejiang University of Technology and worked in China: partnership negotiations, sales, interpreting. So I am equally comfortable designing a system and closing a deal with a supplier in Chinese.

FLAGSHIP PROJECT

Live since 2024

mstar.uz

Ticketing platform for the largest event organiser in Tashkent.

Django · Sole developer · Zero to production

- Designed the data model, the REST API and the role model; wrote the entire project from the first line to launch.
- Grew from a landing page into an online sales system wired to the physical box office in real time.
- Deployed and maintain the server: PostgreSQL, Redis, Celery, Nginx, Gunicorn, systemd, SSL, daily backups.
- Still moving it forward: new modules, refactoring, reviewing my own code and running incident post-mortems.

CASE STUDIES

Live since 2024

mstar.uz

Ticketing platform

Sole developer: architecture, code, infrastructure, support

- Problem. The online storefront and the physical box office sell the same hall. When tickets for a popular concert go on sale, hundreds of people pick the same seats at the same second — and no seat may be sold twice.
- Solution. Seat state is split across two layers. Temporary holds live in Redis: an atomic Lua script sets the key with SET NX EX plus an owner token, adds the seat to the owner's set and publishes an event to a channel; the TTL clears abandoned carts by itself. PostgreSQL is the single source of truth for 'sold': an order is finalised inside a transaction with select_for_update. Django Channels push changes over WebSocket to every open seat map. Celery beat clears expired holds every minute, reconciles box-office holds with the database every ten minutes, and settles stuck payments every three.
- Why this way. A hold lives for minutes and is created hundreds of times a second. Keeping it in PostgreSQL means locking rows at peak load and inviting deadlocks. Redis gives atomicity and TTL for free. But Redis cannot be trusted with money, so the sale itself is recorded only in the database and Redis stays a fast layer with periodic reconciliation. The owner token sits in the key's value rather than being a plain busy flag, so somebody else's tab cannot release your hold.
- Result. No double sales: online and box office trade the same hall in real time. The platform has been in production since 2024 and I run it alone — from the data model to the server, deployment and backups.

Stack: Python · Django · DRF · PostgreSQL · Redis · Celery · Django Channels · WebSocket · JWT · Nginx · Gunicorn · React · TypeScript · GitHub Actions

Launched in 2026

Second brand of the platform

An isolated instance for a separate client

Sole developer: instance isolation, audit trail, rollout

- Problem. Another organiser needed the same product: its own brand, its own staff, its own money. Plus an audit requirement — every action must show who changed what.
- Solution. Isolation at the instance level rather than a shared schema: a separate database, its own Gunicorn on its own port, its own Nginx, systemd and PM2 configuration. Everything brand-specific moved into database-backed settings plus a rebranding management command, so rolling out a new brand is a repeatable procedure instead of hand edits across the repository. On top of that, a single action log: UUID key, category, action, severity, target model and id, old and new value in JSONB, IP, user agent, request method and path, indexes for querying and per-period archiving to a file.
- Why this way. Two organisers are two legal entities with separate money. Row-level isolation in a shared database rests on discipline: one forgotten filter in a queryset and another tenant's sales are visible. I deliberately traded deployment convenience for a hard isolation guarantee. One unified log instead of a separate log per module means the audit has one query, one index and one diff format.
- Result. The second brand launched on the same codebase and ran a full sales season; the rollout procedure is documented and repeatable. The client is under NDA — I do not name them, but I am happy to walk through the architecture on a call.

Stack: Django · PostgreSQL · Celery · Redis · JSONB · systemd · Nginx · PM2 · React · Vite · Flutter

HOW I WORK

Data model first	Models and migrations before the API. Reshaping a schema under live money is expensive.
Tests where it hurts	544 tests cluster around money, seats and permissions — not around a coverage percentage.
CI on every push	Docker image builds, code-quality checks, a weekly security scan, dependabot on pip and npm.
Changes ship as commands	Data migrations, rebranding and one-off operations live in 80 management commands, not in a shell on production.
Production is watched	systemd units, logs, daily database backups and a restore procedure that has been tested.
Decisions written next to the code	Deployment checklists, incident post-mortems and security reviews live in the repository, not in my head.

EXPERIENCE

2024 – present

mstar.uz

Full-stack Developer, Technical Support

Tashkent

- Sole developer of the platform: architecture, data model, REST API, role model, payments.
- The whole infrastructure: PostgreSQL, Redis, Celery, Nginx, Gunicorn, systemd, SSL, backups, CI on GitHub Actions.
- Online sales wired to the physical box office; hall state propagates over WebSocket.
- Launched a second instance on the same codebase under a separate brand, with an isolated database and full audit trail.

2025

Partner business development

Marketing and investment

Tashkent

- Invested my own funds and ran promotion and customer acquisition.
- Project closed by mutual decision of the partners.

2024 – 2025

Huadong Holding

Sales Manager, Interpreter

China

- Arranged partnership negotiations between the holding and Uzbek pharmaceutical companies.
- Interpreted negotiations and translated documentation both ways: Chinese and Russian.
- Introduced AI tools into the department's workflow and trained colleagues to use them.

2020 – present

Talks UZ

Co-founder, Developer

Tashkent · project paused

- Media project on culture and events. Built the website, owned the technical side and promotion.
- Covered major concerts and events.
- Project paused, the team went separate ways.

2017 – 2020

MustMedia UZ

Web Developer

Tashkent

- Built landing pages and corporate websites for agency clients.
- Markup, CMS work, deployment to hosting.
- Joined the agency while still at school. The agency closed during the lockdown.

2015 – present

Family business, textile manufacturing

Procurement and Technical Support

Tashkent · part-time

- Work with Chinese suppliers: sourcing raw materials and spare parts, negotiating in Chinese.
- Maintaining computers and production equipment: diagnostics, repair, setup.
- Automating routine processes, including with AI tools.

2015 – present

Freelance, web development

Websites, landing pages, online stores

Tashkent

- Dozens of full-cycle projects: markup, CMS, content, deployment, handover to the client.
- Work lives in private repositories and is not published publicly.

SKILLS

Backend	Python · Django · DRF · REST API · Celery · Django Channels · WebSocket · JWT · OpenAPI / drf-spectacular
Databases	PostgreSQL · Redis · Schema design · Migrations · Indexing · Transactions and locking
Infrastructure	Linux · Nginx · Gunicorn · systemd · Docker · GitHub Actions · SSL and domains · Backups and restore · Servers from scratch
Architecture	API design · Role-based access control · Audit logging · Data isolation · Payment integrations · Background jobs
Frontend	React · TypeScript · Vite · JavaScript · HTML · CSS · Responsive layout
Programming languages	Python · TypeScript · C++ · Java · C#
CMS	WordPress · Joomla · Turnkey websites
AI tools	Process automation · Faster development · 5+ years
Beyond code	Negotiation · Working with Chinese suppliers · Sales · Launching projects from scratch

EDUCATION

2021 – 2025

Zhejiang University of Technology

Computer Science

Hangzhou, China

Core languages on the programme: C++, Java, C#. Studied in Chinese, HSK 4.

LANGUAGES

Russian	Native
Uzbek	Native
English	Fluent
Chinese	HSK 4