



BORIS STEPANENKO

FRONTEND / FULLSTACK
DEVELOPER

Work format: Remote, full-time

Timezone: UTC+3
(Europe/Moscow)

● ABOUT

Frontend developer with fullstack project experience on Vue / Nuxt. I design scalable application architecture, work with global state, REST APIs and databases (MongoDB, SQLite), implement dark/light themes and multi-language support (up to three languages in one project). I can run a product end to end: from data structure and API to the final interface. In 2026 I shipped 2 commercial projects, including an international website in 3 languages.

● CONTACTS

• +7 993 676-10-75

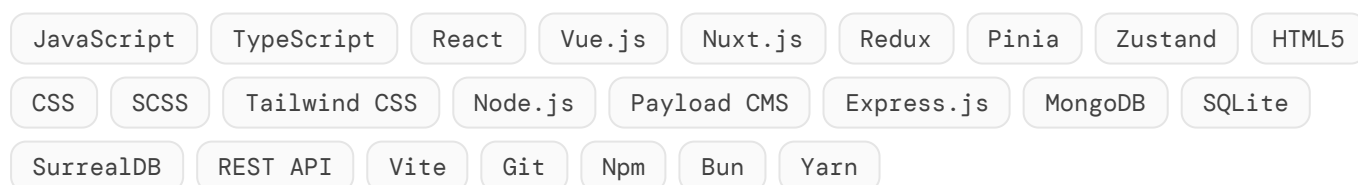
📧 Telegram: @mervik104

📁 GitHub: mervik104

✉ mervik104@gmail.com

🌐 mervik.ru – portfolio site

● TECH STACK



● LANGUAGES

Russian	native
English	A2-B1

● EDUCATION

SELF-EDUCATION

Path into development

MAY 2025 – JANUARY 2026

● EXPERIENCE

| FULLSTACK DEVELOPER

JUNE 2026 – AUGUST 2026

Nekrasovsky Group  nekras.ru

Commercial website for the Nekrasovsky group of companies. Key focus – presenting the business to an international audience: implemented three-language support (Russian, English, Chinese) and dark/light theme switching. Designed a backend on Payload CMS with an admin panel so the client can manage content independently, plus the MongoDB structure and API. Frontend – Nuxt + Tailwind, ISR rendering, SEO optimization for commercial traffic.

| FULLSTACK DEVELOPER

JULY 2026 – JULY 2026

Internetlab (test assignment)  GitHub

Test assignment: a contact form with message sentiment analysis via Google Gemini AI. Backend on FastAPI with layered architecture (API / Service / Repository), Pydantic validation, rate limiting and request logging. Frontend on Nuxt 4 with Pinia and Tailwind CSS 4. Implemented a graceful fallback when the AI model is unavailable, and auto-generated Swagger documentation.

| FULLSTACK DEVELOPER

MAY 2026 – JUNE 2026

Freshcheck  freshcheckastra.ru  GitHub

Website for a civic initiative of Astrakhan residents – an independent project with open-source code. The priority here was speed to launch and openness: backend on Payload CMS + MongoDB. Frontend on Nuxt + Tailwind with dark/light theme, ISR and SEO – so inspection content shows up in search and updates fast.

| FRONTEND DEVELOPER

JUNE 2026 – JUNE 2026

Personal project  mervik.ru  GitHub

Built portfolio site with Nuxt and Tailwind CSS v4. Deployed to GitHub Pages.

Nuxtgram - Personal project  nuxtgram.mervik.ru  GitHub

An Instagram-style social/photo feed app: posts with images, likes/reactions, comments, follows, and profiles. The architecture is fully serverless – no self-hosted monolithic backend, only provider services with clearly separated responsibilities. Frontend is a Nuxt 4 SPA (Vue 3.5, TypeScript 5.9): SSR disabled, hash-based routing, built into a single static HTML file via `nuxt-single-html` – the whole app is deployed as static assets on GitHub Pages, with no CI pipeline yet. Authentication runs entirely on Clerk (@clerk/nuxt) – email/OAuth sign-up and sign-in, session management; for data access Clerk issues a JWT from a custom template with a project-specific issuer and audience. That same JWT doubles as the SurrealDB session token and is verified by a Cloudflare Worker (issuer/audience/subject checks) before any operation – the frontend is never treated as a trust boundary, all secrets and authorization live server-side. Data lives in SurrealDB (a document-graph database) accessed through the typed `surqlize` ORM: tables for users, media, posts, comments, and edges for follows, `post_reactions`, `comment_reactions`, with access enforced by schema-level, server-side permissions (migration `001-infrastructure.surql`), plus real-time LIVE subscriptions that incrementally update the Pinia stores. Media and file handling go through a Cloudflare Worker in front of Backblaze B2: presigned upload/download URLs with a TTL, ownership checks on objects, size limits, and no direct public writes to the bucket. State management is Pinia (auth/post/comment/follows stores), forms use `vee-validate` with Zod schemas, and the UI is built on `@nuxt/ui`, Tailwind CSS 4, `tailwind-variants`, color-mode support, toast notifications, and `@formkit/auto-animate` for animation. The project has unit tests on `bun:test` (mappers, schema, formatting, logger, redirects) and a custom browser logger that batches logs to a local log endpoint. One notable part of the project's evolution: Nuxtgram originally ran on a self-hosted Payload CMS backend (Next.js + MongoDB) as its monolith – that code has since been split out and preserved separately as a legacy repository (`nuxtgram-backend`) purely for reference, while the current codebase has fully moved to the serverless stack described above.