Roman Ermishin

Software Developer

Contacts

- **GitHub:** @EternalRival
- **E-Mail:** erdevelopment8@gmail.com
- **Discord:** Eternal Rival#0309
- **Telegram:** @Eternal_Rival

Technical Skills

- Git
- TypeScript / React
- Webpack / Vite / NextJS (Pages Router)
- TanstackRouter / React Router
- Zustand / RTK
- TanstackQuery / RTKQ
- RESTful Api / GraphQL
- CSS modules / SCSS / TailwindCSS / MUI
- Yup / Zod
- Jest / Vitest / Testing Library

Experience

- **gem-puzzle**
  - source / deploy
  - Study Project
  - A classic 15 puzzle game made with JS classes and canvas. SPA. Implemented selection of playing field size and saving records between game sessions.
  - JS, Webpack, Sass
- **songbird**
  - Study project
  - source / deploy
  - Quiz on TV show soundtracks. MPA. Implemented custom audio player, logic of consecutive questions change, getting points for answers, saving the last received

result.
    - JS, Webpack, Sass
- **online-store**
    - Study project (team)
    - source / deploy
    - Online Keyboard Store. SPA. For this project, I used self-written devtools scripts to parsing and converting product data from a real online store. I also implemented a kind of InMemory database of goods and Api Service for working with it (sorting, filtering, cart mutations, etc.). In addition, I was responsible for customizing project configs (webpack, eslint) and creating/ maintaining basic class components that were used throughout the project.
    - TS, Webpack, Sass
- **what do you meme**
    - Study project (team)
    - source
    - Fullstack application card game "what's the meme" written in Angular + NestJS. My responsibility was to implement the backend of our application. This included connecting and working with the DB, providing REST Api for frontend, authorization, registration with password encryption (hashing), storing and reading/editing/deleting user data in the DB, storing static files on the server and accessing them via REST Api. Compression and delivery of requested files as zip-archives. In addition to working with the database, I implemented work with chat (receiving and sending messages with division into rooms), work with lobbies (creating/deleting lobbies with an interface for configuring game parameters, providing lists of lobbies by WebSocket request) and gameplay itself (launching the game, manual and automatic change of game phases, listening and processing different actions of players at each game phase).
    - TS, NestJS, TypeORM, PostgreSQL, JWT, Swagger, Socket.IO
- **virtual-keyboard**
    - Study project
    - source / deploy
    - A simple web application with a virtual keyboard and text field.
    - JS, Webpack, Sass
- **rsschool-tools**
    - Pet project
    - source / deploy
    - Application for RSSchool students/activists. Consists of an app to calculate expected points for completed assignments, an app to find deployments of other students and an app to filter unique url's in the list.
    - TS, React, NextJS (Pages Router), TailwindCSS, SWR, Zod,
- **rsschool-gratitude-summary**

    - Pet project
    - source / demo
    - An application in the form of a Chrome browser extension for RSSchool students/activists. The application requests and displays a list of sent/received gratitudes (honor system). Implemented narrowing search and customizable sorting. All actions are performed on the client, as I don't have direct access to backend information
    - TS, Vue3, Vite, BeerCSS
- **home-library-service**
    - Study project
    - source
    - Backend application for storing data on artists, albums and songs with the ability to add them to your favorites. Implemented running the application and database in Docker containers and interaction between them. Implemented authorization using JWT. Implemented Endpoints for CRUD operations for requests via REST Api.
    - TS, NestJS, Docker, Docker Compose, TypeORM, PostgreSQL, JWT, Swagger
- **graphiql-app**
    - Study project (Team)
    - source
    - Playground/IDE for graphql requests. The application allows you to select any GraphQL Api, send requests to it (including those with headers and variables) and display responses. I was responsible for the main page of the app. I implemented the main features of the application: interface for selecting/changing Api, query and output documentation of the selected Api, interface for typing GraphQL query, dynamically displayed interface for customizing headers and variables of GraphQL query, interface for displaying query results. Also participated in customizing project configs.
    - TS, React, Vite, Vitest, Testing Library, MSW, Sass, CSS Modules, React Router, Redux/RTK/RTKQ, Yup, GraphQL, React Hook Form, Firebase
- **e-commerce-app**
    - Study project
    - source / deploy
    - Standard frontend application online store using CommerceTools service (GraphQL Api is used) as backend. Implemented authorization, registration, product list page with filtering and sorting, detailed product information page, current user profile page (password and personal data can be changed). This project was originally developed by RSSchool curators as a team project, but I am doing it alone, in my free time, in order to expand my technology stack. Among the planned features are editing shipping/billing addresses, working with shopping

cart, etc.. The project is not finished. The project development is temporarily suspended for personal reasons.
- TS, React, NextJS (Pages Router), Vitest, TailwindCSS, Material UI, Zustand, Tanstack Query, Zod, Axios, GraphQL, React Hook Form, Swiper, Feature Sliced Design, CommerceTools

Education

- RSSchool JSFE2022Q3 (Certificate)
- RSSchool NodeJS2023Q2 (Certificate)
- RSSchool React2023Q4 (Certificate)