Vladimir Logachev

🏴‍☠️ Creating products.
🔋 Engineering every day.
🧰 FP, Haskell, Elm, Scala, whatever.

Location: Remote
Mail: mailto:vladimir@logachev.dev
Telegram: https://t.me/vladimirlogachev
GitHub: https://github.com/vladimirlogachev
Twitter: https://twitter.com/logachev_dev
LinkedIn: https://www.linkedin.com/in/vladimirlogachev
Website: https://logachev.dev
Download cv: https://logachev.dev/cv_vladimir_logachev.pdf

About me

In programming, I thrive on a proactive and strategic approach that goes beyond relying solely on intuition, especially in unprecedented situations. I firmly believe in the importance of continuous learning and staying ahead of the curve. That's why I not only invest time in reading books well in advance but also actively engage in prototyping and experimenting with new technologies, ensuring that I am well-prepared to leverage them when the situation calls for it.

I am deeply passionate about functional programming and its ability to bring value to both companies and individual specialists. I not only focus on writing high-quality code but also invest significant time and attention in fostering strong relationships within the team. I actively organize meetups and reading groups at my workplaces, as well as promote collaborative practices like pair programming and pair testing, which I find to be highly effective.

Beyond my technical expertise, I bring entrepreneurial experience to the table. When working on personal projects, I take initiative, generate ideas, fill in any missing pieces, and deliver immediate results. I believe in taking ownership and refining the details later.

This is why I gravitate towards functional languages with strict static typing, such as Haskell, Elm, and Scala. I value the assurance provided by strict typing and algebraic data types (ADTs), enabling rapid development without the fear of design mistakes, as they can be easily resolved later. Additionally, leveraging code generation and utilizing a mono repository further accelerates our pace.

As an individual, I embrace a win-win mindset, avoiding a victim mentality and the need for constant approval. I openly respect and appreciate those who contribute to the growth and knowledge of others. My vision for my technical career is to create wealth and success for all involved through a combination of functional programming and strong teamwork.

Haskell

Libraries: aeson, Decimal, fmt, generic-random, haskell-to-elm, hspec-golden, hspec, http-conduit, lens, mtl, mu-graphql, nonempty-containers, parsec, persistent, postgresql-typed, QuickCheck, relude, retry, servant-server, time, transformers, wai

Elm

Libraries: elm-all-set, elm-charts, elm-crypto-string, elm-css, elm-dropbox, elm-graphql, elm-ordering, elm-review, elm-round, elm-ui, elm-units, random, remotedata, svg, test

Concepts: Browser API interop (Websockets, LocalStorage), Ports, Tasks

Scala

Libraries: zio, zio-http, zio-test, cats, cats-effect, tapir, circe, chimney, fs2, testcontainers, scalatest, scalacheck, specs2, scodec, akka, akka-http, akka-stream, finagle, flyway, enumeratum, scala-parser-combinators

TypeScript

Functional programming: fp-ts, io-ts, rxjs, sanctuary-js, ramda

Frameworks and related: React, Angular, Next.js, Redux, Redux-saga

Network: Socket.io, Apollo

Testing: Jest, Mocha, Jasmine

Styling: SCSS, Emotion

Other

Databases: PostgreSQL, Redis, Clickhouse, MySQL, MongoDB

Infrastructure and tooling: Kubernetes, Docker, GitHub Actions

APIs: GraphQL

Misc: GitHub Projects, Figma

Commercial experience

Swift Invention

https://www.swiftinvention.com

Backend Team Lead 08/2022 — 05/2023

I maintained two projects as a team lead, including implementing new features, writing tests, and fixing bugs.

Also, I mentored new developers to get on board with Scala (mostly, from a Java background), and run a weekly Scala book reading club.

Backend: Scala, zio, zio-http, zio-test, tapir, circe, chimney, enumeratum, flyway, testcontainers, finagle, scalatest, MySQL, Redis, Docker

Wolf, private trading platform

Principal engineer and founder 08/2021 — 07/2022

My role:

• make technical decisions to produce an MVP as soon as possible without sacrificing the reliability or maintainability

• design and implement features as a full-stack developer, solely and via pair programming, and write unit tests

• design the trading algorithm

• describe tasks and manage the project

• review pull requests

• mentor new developers

Technical details:

• 9+ KLOC of Haskell code and 9+ KLOC of Elm code

• Backend type declarations serve as a contract, and Frontend types and JSON codecs are generated from the backend type declarations

• Postgres queries are type-checked against a real DB at compile time

• Docker containers are built with Github Actions and pushed to GitHub container registry

• Every algorithm can be run in a real environment or simulated (with heavy use of Haskell parallelism and concurrency)

• The project has a strong focus on the absence of partial functions

• In general the whole system is reliable, stable, and behaves correctly, despite the fact that the domain and external APIs have lots of edge cases, and are sensitive even to small deviations from valid values and ranges.

• Tested with different kinds of tests: unit tests, snapshot tests, property-based tests

• The entry threshold for the Haskell codebase is relatively low – it doesn't require an understanding of advanced concepts for everyday work

• The project has documentation, and the development tasks are well-described and tagged in GitHub Projects

All these points lead to ease of development – refactors are fast and do not cause regressions, creating tasks and updating their status does not require much effort, no unnecessary/routine actions are required from the developer. Relatively little time is spent on fixing technical issues – therefore, there is a lot of time left for substantive things and implementing new features.

Although the source code of this project isn't publicly available, I can demonstrate and discuss it during the technical interview.

Backend: Haskell, servant-server, postgresql-typed, haskell-to-elm, mtl, Decimal, relude, hspec, QuickCheck

Frontend: Elm, elm-ui, elm-charts, remotedata, elm-review, elm-test

Infrastructure: Docker, Nginx, GitHub Actions

Project management: GitHub Projects

Fourier Labs

https://fourierlabs.io

Software Engineer 03/2022 — 04/2022

Backend: Haskell, Plutus

Infrastructure: Nix

Blockchain: Cardano, Ethereum

Pamir

Frontend developer 05/2020 — 12/2020

Developed a web application, which utilizes server-side rendering and covered it with unit tests. Packaged everything in Docker and set up CI.

I also mentored the second frontend developer who joined the team later.

Frontend: TypeScript, React, Next.js, GraphQL, Apollo, FP-TS, Emotion, Jest

Infrastructure: Nginx, Docker, GitHub Actions

Eldis

https://eldis.ru

Software engineer 10/2019 — 12/2019

I developed a declarative decoder for the internal binary document format, covered it with tests, including property-based testing.

Backend: Scala, scodec, cats, fs2, decline, specs2, scalacheck

Neolab-Nsk

Fullstack developer 01/2019 — 09/2019

I implemented new functionality in existing web applications, fixed defects, and developed new applications, and microservices, covering them with unit tests and integration tests.

Frontend: TypeScript, React, Redux, Saga, RxJS, FP-TS

Backend: TypeScript, Node, Redux, Saga, RxJS, Redis, Lua, Mongo, PostgreSQL, Clickhouse, Docker

SocialSweet Inc

https://sweet.io

Frontend developer 08/2018 — 01/2019

Sweet's product is a loyalty platform, social network, and online store.

I performed tasks related to business logic at the front end and was engaged in covering the existing code with unit tests and tuning them, thanks to which the tests were launched using CI pipeline, and the defects associated with an unsuccessful merging of Git branches in a huge codebase really began to be prevented.

Frontend: TypeScript, Angular, RxJS

Allmax

https://savl.com

Frontend developer 11/2017 — 08/2018

I worked in the Savl project — this is a mobile application, a wallet with support for 6 cryptocurrencies.

I was responsible for the data layer in the mobile application. I applied everything that I learned from books about functional programming and software design, and also completely covered the business logic with tests, as a result of which the application became fault-tolerant and modular, that is, it stopped crashing due to exceptions or unexpected behavior of external services, and allowed to enable and disable support for individual cryptocurrencies at any time.

Also inside the company, I made several presentations on functional programming.

Frontend: JavaScript, Flow, React Native, Redux, Saga, Ramda, Sanctuary, Socket.io

Showcase projects and assessments

servant-to-elm example

https://github.com/VladimirLogachev/servant-to-elm-example

An example full-stack web application, built in a typesafe functional way. What's cool there is that servant-to-elm does the job of generating types and decoders/encoders from Haskell types and Servant definition to Elm, which not only catches regressions in the compile-time but also provides ready (and highly configurable) Elm functions to fetch necessary data from the server.

Elm, Haskell, Servant, SQLite

Transitive Closure (assessment)

https://github.com/VladimirLogachev/transitive_closure

A function that accepts a list of object ids and returns those objects and all objects which they refer to (directly or indirectly) from some Repository with a monadic interface. The code is pretty abstract, but still well-tested (including tests for cases like very large referencing graphs and cyclic references).

Scala, Cats, ScalaTest

Web crawler microservice (assessment)

https://github.com/VladimirLogachev/crawler

A microservice that accepts a list of page URLs, and returns a list of page titles. It takes into account situations like bad URLs, duplicate urls, redirects, concurrency, and backpressure.

Scala, Akka HTTP

Notable contributions

higherkindness/mu-graphql-example-elm

https://github.com/higherkindness/mu-graphql-example-elm

An example of how to implement both frontend and backend in a schema-first, typesafe, and functional way (for the mu-haskell library, demonstrating its GraphQL capabilities). I rebuilt its Elm frontend and made minor changes to the Haskell backend (and also discovered a couple of bugs).

Elm, Haskell, GraphQL

FP Specialty

https://fpspecialty.github.io

From 2019 to 2021 I have maintained a functional programming reading group for people of all functional programming skills.

Reading group

Russian translation of the Mostly Adequate Guide to Functional Programming in JavaScript

https://github.com/MostlyAdequate/mostly-adequate-guide-ru

The book introduces the reader to the functional programming paradigm and describes a functional approach to developing JavaScript applications. The translation was initiated by Maxim Filippov and stopped at 60%. Then I and Sakayama joined the translation, refactored every chapter translated before us, and then finished the translation.

JavaScript, Ramda

Education and courses

Mastering Haskell Programming

https://www.udemy.com/certificate/UC-DRMAMOQ5

Packt, 2019

Functional Programming in Haskell, part 2 (certificate with honors)

https://stepik.org/cert/207739

Stepik, Computer Science Center, 2019

Functional Programming in Haskell, part 1 (certificate with honors)

https://stepik.org/cert/196007

Stepik, Computer Science Center, 2019

Computer Science Summer School, Theory of Programming Languages

Novosibirsk State University, 2019

Maintenance of computer equipment and computer networks

Novosibirsk Aviation Technical College, 2004 — 2008