Vladimir Logachev

Haskell, Elm, Functional programming

Remote

Mail: mailto:logachev.dev@ya.ru

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 prefer not to rely on intuition (which, I believe, is usually based on previous experiences and tends to fail in unprecedented situations), but instead, read books well in advance.

I like to see functional programming deliver value to both companies and individual specialists, so I devote a lot of time and attention not only to code but also to people. I'm a big fan of meetups and reading groups, which I run at my workplaces from time to time, and also I consider pair programming and pair testing to be very effective practices.

In addition to technical experience, I have entrepreneurial experience, so on my personal projects, I act without a need for instructions, come up with ideas, fill in the missing pieces myself, produce an immediate result, and apply the polishing later.

That's why I prefer functional languages that implement strict static typing, such as Haskell and Elm. My preference is based on the idea that once we have strict static typing and ADTs, we can move very quickly without fear of making design mistakes because they become easy to solve later. (And we can move even more quickly with the help of code generation and mono repository).

As a person, I love the win-win approach, I don't play a victim, I don't seek approval and I do publically respect everyone who helps others or has knowledge. So, the vision of my technical career is that FP and strong teamwork can make everyone involved rich.

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

Other

Databases: PostgreSQL, Redis, Clickhouse, MongoDB

Infrastructure and tooling: Docker, GitHub Actions

APIs: GraphQL

Misc: GitHub Projects, Figma, Miro

Commercial experience

Wolf, private trading platform

Principal engineer and founder 08/2021 — 01/2022

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: Nginx, Docker, GitHub Actions

Project management: GitHub Projects

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

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

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

Scala

Libraries: cats-core, cats-effect, fs2, scala-parser-combinators, scalatest, scalacheck, specs2, scodec, akka, akka-http, akka-stream, scala-parser-combinators

Commercial experience

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

Showcase projects and assessments

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

TypeScript

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

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

Network: Socket.io, Apollo

Testing: Jest, Mocha, Jasmine

Styling: SCSS, Emotion

Commercial experience

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

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

Notable contributions

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