A Haskell Platform for Creating Progressive Web Applications

A midterm progress report

Jakub Zárybnický

January 26, 2019

Contents

1 Introduction

In this report, I will introduce the project to the reader and describe its goals and their current status as of the end of January 2019. In particular, I will focus on what my original task looked like and how I deviated from it, and how I expect to correct these flaws going forward.

I will also describe the work I've performed so far and propose a plan for the remaining part.

2 Project

My original idea for the project was to build a set of libraries and patterns that would ease the development of full-stack Haskell web applications, both their client and server

parts. The work would fill a niche in the language's ecosystem that is in demand, and it would build a basis for further development of tools and libraries.

When I introduced the project to my adviser, I focused in particular on Progressive Web Applications, a rather popular topic in general web development. I think it was a mistake, in retrospect, and that I should have emphasized the 'full-stack' aspect of my original idea, which would expose me to a greater challenge of creating a 'pattern language' to describe different shapes of web applications, which I find more interesting. I have spent some time and effort on this topic, and I hope to incorporate it into the final result of my thesis.

The stated goals of the project at its beginning were:

1. Study the current state of the Haskell ecosystem for creating web applications.
2. Find suitable libraries as a starting point for creating Progressive Web Applications (PWAs), i.e., web applications that can offer the user functionality such as working offline or push notifications.
3. Implement a framework for PWAs. Focus, in particular, on the implementation of components for offline storage, push notifications, and also support tools.
4. Create a set of example PWAs using the created framework.
5. Compare the created framework with existing (e.g. JavaScript) frameworks for PWAs.
6. Summarize the obtained results and discuss the future work.

The goal for this first midterm checkpoint was to complete the first two points at a minimum.

3 Work performed

I would say that I have completed the first two points and made some sizable progress on the third one. While my focus has been on other projects for most of the past half-year, I have continued researching relevant topics for most of the time and I have also made some significant progress on the applications from which I take inspiration for this project.

My work since the start of the project included:

- A cross-sectional analysis of the available frameworks for many languages, to get a taste for the conveniences that developers from other languages are used to.
- An analysis of the web-development tools and libraries available in the Haskell ecosystem.
- I have done further work on the applications upon which I am basing the framework. I have also created one and a half other new applications with a different architecture from the first ones (the second one is only half done).

- I have extracted several patterns from the above applications into a repository of 'snippets'. These snippets will later become the starting point for the framework itself.
- I have researched many topics related to current web development trends (JAM stack, Serverless, Web Components, CRDTs, DevOps topics, best practices in authentication and authorization, and others).

Visible artifacts of my work are in two places:

- A Git repository at `https://github.com/zarybnicky/thesis`
- An almost empty blog at `https://zarybnicky.com`
- One in-progress article at `https://zarybnicky.com/drafts/elements-of-pwa-framework/`

Some artifacts aren't ready to be published yet, but I hope to remedy that during the last week of January or first week of February. These artifacts are:

- A finished and published version of the "Elements of a Web framework" article
- An article on all the Haskell tools and libraries that are available, and a decision about which ones are suitable.
- More publicly available 'snippets' extracted from the existing applications.

I would also like to publish the source code for at least one of the several applications that are the inspiration for this work, but I do not expect to get permission from all stakeholders of the projects.

Regarding the last point of my 'work accomplished' list, the research on current web development trends - I do not expect to go through my entire backlog of notes and bookmarks anytime soon, but I hope to start working through it and writing up my notes into the form of articles. I have started a habit of writing long-form text daily, so I hope these articles will take less than two months to write - unlike the last one.

4 Problems encountered

My main problem during the period from September to December is a lack of available time. In the time before the start of the school year - and also now in January - I've had a much lighter load, and so I was able to work on less urgent projects which include this thesis. The demands for my time are unfortunately irregular, but I now have a rather clear idea about the next steps for the project, which brings me to the next problem–

Until this week, I was rather uncertain about the real scope of the project. That meant, among other things, that there wasn't any visible forward movement in the project, only the constant broadening of scope. I've had to clarify the goals while writing this report and preparing the presentation slides, which means that I will be able to schedule my work better.

My last problem was more of a personal one - my productivity neared zero during October and November, and everything I did took much longer than necessary, which meant that any time I might have had for my thesis went to other, more urgent work.

5 Further plans and schedule

Going over every aspect of the project in the last week while writing the presentation and getting the Git repository into shape meant that I had to confront how large my envisioned scope for this project became. I had to prune out all tasks not directly related to the real goals of the thesis.

Out of several long thinking and writing sessions came several versions of a plan. These plans are tiered, and each includes everything from the previous plan. I call these plans 'realistic' - what I know I can have ready at the end of the project regardless of outside influences; 'optimistic' - what I think is doable but I don't expect to have fully finished by the project deadline; and 'highly optimistic' - my idea of an ideal outcome, I expect to start working on its goals and perhaps have some ready, but not entirely and definitely not as a polished product.

5.1 Realistic

This plan is only composed of the original objectives of this thesis. The result is a set of libraries and scripts for writing a browser application that fulfills all the requirements of a Progressive Web Application as defined on Google's checklist.

This checklist notably includes:

- Pages are responsive on tablets & mobile devices
- All app URLs load while offline
- Metadata provided for Add to Home screen
- Page transitions don't feel like they block on the network
- Each page has a URL
- Pages use the History API
- Site uses cache-first networking
- Site appropriately informs the user when they're offline
- Push notifications (consists of several related requirements)

To achieve this, I need to create:

- A full-featured browser routing library. While there are some existing implementations, they are either incomplete or long abandoned.

- A wrapper around ServiceWorkers, or a template to simplify project creation.
- A push notifications library.
- A library or a script that will render HTML 'shells' of all pages on the site, for fast first load.

Not required by the checklist, but would improve the quality of my work:

- A template of an application, with predefined internal architecture, that uses all of the above libraries
- A utility library for querying and caching data from an API, be it HTTP, WebSocket, GraphQL, or others.

An application using these tools and libraries would consist of a server written independently, and of a browser application. Implementing the communication between them is left to the developer, as is implementing many common conveniences usually provided by a framework.

These goals also include fully documenting the code written, as well as testing and benchmarking it to remove the most obvious bottlenecks.

5.2 Optimistic

In addition to the previous plan, this plan includes a communication channel between the server and client, with a data storage component in the client and a synchronization protocol for use in both real-time applications and fully offline-capable applications.

The result will then also include:

- a library to use in code shared between the server and client - a way to define the shape of the transport channel (and its API for non-Haskell applications)
- a server library, to allow user code to implement the specified protocol
- a client library with a storage component for entities and pending requests

There are multiple sub-goals in this goal. There is no frontend cache library yet, so a library that implements transparent caching would be sufficient here. Designing a synchronization protocol that would enable applications to stay offline for prolonged periods of time and synchronizing whenever they come online would be a great achievement, and perhaps a topic for a bachelor thesis on its own...

An application using these libraries would consist of a server and a client sharing code that contains the definition of their communication channel. I do not yet know how much these libraries would affect the shape/architecture of the server, but the client library would form the core of the client - with the libraries from the previous plan forming the shell.

In an ideal case, designing the synchronization protocol would include a proof of correctness as well. I have seen several ways to achieve this, the most recent one being a project called hs-to-coq.

And, like above, this plan also includes documentation, tests, and benchmarks of the created code.

5.3 Highly optimistic

The nature of this plan's goals is a bit different. Whereas I solve small, well-defined problems in the previous ones, this plan's goal is to design the types to describe all possible shapes of web applications. This is inspired by my research into the 'JAM stack' and related architectures, where most of the client code is executed during build-time or even on demand, and only user interface code remains to be executed in the browser.

The goal here is to design an abstraction that covers, at a minimum, all of these cases:

- Client only. An application that doesn't need to communicate with a server, like a web presentation, or a blog, a set of pre-compiled HTML+JS files.
- Server only. Either just an API, or a plain HTML website with no JavaScript.
- Server and client, with the client rendered during run-time by the server.
- Server and client, with the client rendered during build-time and served separately (e.g. via an S3 bucket).
- Server and client, with the client re-rendered on demand, whenever the data that it shows changes. This is the shape of a 'JAM stack' application.

This also includes projects with multiple clients or servers, which puts even more pressure on the supporting tools.

This is more of exploratory work, so I don't expect to see results immediately. However, I have projects of several of the proposed shapes (a client-only app, a run-time rendered client, and a 'JAM stack' application) in which I could try out any attempts at a design, to see how useful or easy-to-use they are.

6 Conclusion

I am, of course, aiming for the most optimistic plan, as that's that presents both the biggest challenge and offers the greatest benefits. Just to be safe though, I will need to start with the 'realistic' plan, so that I avoid sprinting at the last moment at a time when I should be preparing for my final exam.

I still feel that my expectations are set too high and that I will need to lower them a lot, especially regarding the second and third plan, but I hope that my 'realistic' plan is realistic enough for real-life.