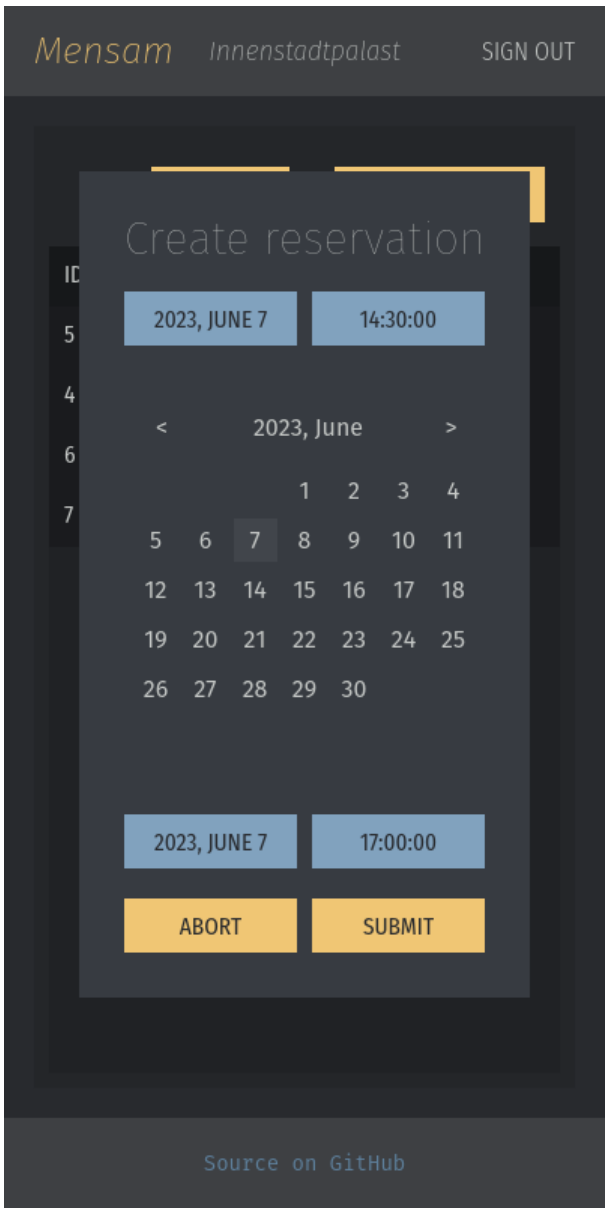
Announcement: Mensam

Over the last few months I was developing a desk-booking web application.



GitHub Repository

https://github.com/jumper149/mensam

Hosted instance

- https://mens.am
- OpenAPI definition: https://mens.am/openapi

Motivation

At work we are using hot desks. This does make a lot of sense, because many of my coworkers work remotely most of the time.

Still, sometimes there are events at the office, that many people want to attend.

At some point we had more employees than desks in the office, so we needed a reservation system. For the time being we are using the Microsoft Office 365 calendar which sort of works fine, but there are a few issues. For example you cannot get an overview of booked/available desks.

I started working on Mensam as a side project because there was no good solution and I wanted to create one.

Technical stuff

The backend server is written in Haskell and the frontend is using Elm. Everything is glued together with Nix and I'm hosting mens.am on a NixOS machine.

Backend

I started out with copying some Haskell code from my homepage. This already layed the foundation in some aspects.

- mtl-style "effects"
- JSON configuration file
- warp web server
- servant routing

I also needed some persistent storage and after some experimentation I settled on SQLite using the selda library.

The next step was sticking everything together. Servant and selda were the most influential frameworks during this step. I decided to write all of the API documentation with servant-openapi3. Selda was quite good by just not getting in my way.

Frontend

This was a part, that I struggled to get started with. First I wrote a "quick" (this took way too long) TUI client with brick and servant-client as a proof of concept.

And then I decided on Elm to write the HTML/CSS/JavaScript frontend.

I tried to generate the API calls from servant or from the OpenAPI specification, but unfortunately that didn't go too well. I am using a servant checkout very close to master and the libraries that I found unfortunately just weren't able to handle multiple responses using `UVerb`. The code generation from the OpenAPI spec sort of worked, but I wasn't very happy with the output. HTTP stati were completely ignored, which I want to use to convey different kinds of errors. A failed login should be disambiguated from a network error and the backend might want to attach information to an error. Tagged unions were ugly as well.

NOTE	I think that codegen from OpenAPI is probably be the best approach to generate API

> calls. I just didn't spend the time to support my needs with the code generator.

So for now I wrote the API calls by hand.

I am also using elm-ui, which is a library, that adds an abstraction around HTML and CSS. Centering `div`s is hard, but with elm-ui it becomes quite simple.

Overall the design decisions here were quite pragmatic. I just wanted to get started with the frontend and I don't think it would take much work to rewrite it in another framework.

I would have loved to write Haskell for the frontend as well and share data types, but unfortunately the GHC JavaScript backend is too new and there is no framework available to actually use it. GHCJS on the other hand is a little old and libraries would be out of date.

Current development progress

I am very close to finishing my initial goal of a *minimal viable product*.

MVP Goals from Readme

> **Setup**
>
> As the operator you simply visit an instance of Mensam, such as mens.am.
>
> First you will have to create a *User* account for yourself. Now you create a new *Space* for your location. Within this *Space* you can now add *Desks*.
>
> **Join a Space**
>
> After creating a *User* account you can simply join a *Space* with a single click. Depending on the setup of the *Space*, it is listed publically or only accessible with a specific link.
>
> **Book a desk**
>
> After joining a *Space* you can create a *Reservation* for a *Desk*. You will be able to see other *Reservations*, so you can plan around them. Mensam will deny overlapping reservations.

Future

At this point I really want to talk to people.

- If you are interested in using this, please let me know what kind of features you would like.
- If you are a Haskell/Elm developer and want to look into a real-world application, feel free to talk to me.
- If there are any bad decisions I made, give me advice. We all keep learning.

I'm also keeping a todo list in the git tree.

One of the major goals is improving the UI/UX, which is functional at this point, but still missing some useful information.

Another important feature is to support database migrations. I already have an idea how to do this, but it still needs to be implemented.

You can expect the first release in the coming month.

I'll be at Zurihac 2023. Feel free to talk to me!

If you want to get in touch online just send me an email or use the GitHub issues.