Auswahl der Frameworks

Für den Aufbau der Webanwendung mussten einige größere Entscheidungen getroffen werden, wie zum Beispiel welches Framework verwendet wird. In diesem Abschnitt möchte ich die Gründe für die Entscheidungen, welche getroffen wurden, erläutern.

Backend FastAPI

Beschränkung der Auswahlgruppe

Für die API standen verschiedene Frameworks zur Verfügung.
Zwar gibt es noch viele weitere kleine Frameworks, aber ich habe mich bei der Entscheidungsfindung auf die Bekannteren beschränkt, da für diese meist eine größere Community existiert, wodurch es eine größere Auswahl an Tutorials und Hilfestellungen gibt.
Somit blieben Django, Flask und FastAPI.

Django

Da Django ein sehr monolithisches Framework ist, was eine Menge Boilerplate Code mitbringt, wurde es schnell verworfen, da der geplante Umfang der App nicht so groß ist, dass dies nötig sein wird.

Flask

Mir war es wichtig ein Framework zu nehmen, in das man sich in kurzer Zeit sinnvoll einarbeiten kann, demnach wurde Flask in Betracht gezogen, da es ein Mikro-Framework ist, das für Leute, die sich mit diesem Framework nicht auskennen, keine allzu steile Lernkurve bietet.

FastAPI

FastAPI hat ähnliche Vorteile wie Flask: es ein schnelles, leicht zu lernendes Framework.
Persönlich fand ich FastAPI etwas lesbarer als Flaks, wenn man nicht weiß, wie gut sich die Personen, die sich den Code später ansehen werden, mit dem Framework auskennen. Außerdem sah der Code meines Erachtens ordentlicher aus als bei Flask.
Deswegen ist es FastAPI geworden.

Frontend Sveltekit

Beschränkung der Auswahlgruppe

Ähnlich wie beim Backend wollte ich ein bekannteres Framework haben, wegen der größeren Community. Demnach kamen sofort React, Angular und Vue ins Visier.

Ich hab mir dann noch ein paar Websites angeschaut, welche die beliebtesten JS Frameworks aufgelistet haben und dabei ist mir Svelte aufgefallen. Svelte hat keine so große Community wie die anderen drei Frameworks, aber einige der Features klangen interessant, deswegen hab ich Svelte in die Auswahlgruppe mit aufgenommen.

React & Angular

Da ich mich komplett in das Framework einarbeiten musste, wollte ich ein simples Framework, das keine allzu steile Lernkurve hat. Deswegen sind React und Angular direkt ausgeschieden.

Vue

Vue.js ist ein JavaScript Framework zum Erstellen von UIs, welches auf HTML, CSS und JS aufbaut. (vgl. Vuejs 2019)
Der Aufbau einer Vue Anwendung ist Komponenten basiert. (vgl. Vuejs 2020)
Da Vue eine relativ einfache Syntax hat und im Allgemeinen als leichter zu erlernen gilt als React und Angular, wurde es in Betracht gezogen.

Svelte

Svelte ist ebenfalls ein JavaScript Framework, das ein ähnliches Konzept wie Vue verwendet. Es ist ein Komponenten basiertes Framework, das auf HTML, CSS und JavaScript aufbaut. (vgl. Svelte contributors 2023)
Bei Svelte wird beim Kompilieren der Code zu reinem JavaScript übersetzt. (vgl. Bardiau 2020)

Svelte verfügt ebenfalls über eine simple Syntax, was dafür sorgt, dass man sich dort leicht einarbeiten kann.

Beide Frameworks haben ähnliche Vorteile, daher war es letztendlich eine rein intuitiv getroffene Entscheidung, welches von den beiden Frameworks ausgewählt wurde.

Svelte selbst verfügt allerdings nicht über alle notwendigen Features (z. B. unterstützt es kein Routing), sodass ich das auf Svelte basierte Meta-Framework Sveltekit gewählt habe.

Datenbank

Der Entwickler von FastAPI arbeitet noch an einer anderen Python Library: SQLModel[1]. FastAPI bietet an einigen Stellen spezielle Unterstützung für SQLModel, weswegen es sich angeboten hat diese Library zu benutzen. Da SQLModel auf SQAlchemy[2] basiert, was wiederum ein SQL Toolkit ist, wurde sich für eine SQL-Datenbank entschieden.
Die Datenbank muss keine sonderlich komplexen Datentypen oder Relationen speichern, von daher reicht eine simple Datenbank wie SQLite für den Entwicklungsprozess aus.
Die Datenbank unterstützt Asynchronität, um den Vorteil der Asynchronität der API optimal nutzen zu können.

Bei Bedarf kann die SQLite-Datenbank ohne größeren Aufwand gegen eine andere asynchrone, SQL-basierte Datenbank ausgetauscht werden.

Alembic

"Alembic is a lightweight database migration tool for usage with the SQLAlchemy Database Toolkit for Python." (Bayer 2010)
Ich habe mich für Alembic entschieden, da es eines der bekanntesten Tools für SQLAlchemy ist, und man sich dort relativ einfach einarbeiten kann. Da das mit FastAPI verwendete SQLModel auf SQLAlchemy basiert, hat sich dies angeboten.

---

1. https://sqlmodel.tiangolo.com/↩
2. https://www.sqlalchemy.org/↩