David Layer-Reiss

Evaluation of a possible backend framework

After considering the ideas posted in the „Framework für User Interface finden“ the goal is to find suitable framework that is easy to learn and also the used programming language syntax should be similar and therefore allowing a steeper learning curve for everyone participating in the development.

After comparing different programming leagues I come to the conclusion that a NodeJS based framework would probably suit the needs from above best. We probably also have to use JavaScript in our frontend (see React, Vue.js, etc.) or mostly „syntax-similar“ languages like Dart (Flutter).

Beside allowing a steeper learning curve an JS based backend also enables a very good cross platform compatibility and is running on all major unix systems used for providing web services. With NodeJS is also easy to manage dependencies with build in package managers like NPM. The underlying concept of NodeJS als guarantees that almost all dependencies that we might use are also supported on a wide range of platforms. A well supported and widely used module for interacting with mysql based database servers also exists and allows to proxy mysql commands secure and fast to our backend database.

As we have chosen NodeJS as our JavaScript runtime environment we can select a framework for a wide range of different use cases. I suggest we use a framework that is easy to use and also supports JSON REST API features natively. This allows as to easily implement the technical features that we specified in our concept paper.

List of different JS frameworks:

https://koajs.com/
https://expressjs.com/de/
https://hapi.dev/

…. to be continued…

Update: After evaluation of the different projects last week we decided to use: express.js. Its simple structure allowing use to build up a backend from ground without spending too much time learning any given backend structure or figuring out how to use any given boilerplate structure. Express.js allows simple binding of so called „routes“. This routing allows use to specify the structure of our API endpoints. Native JavaScript JSON helpers are also natively supported.

Example Express.js code snippet (source: https://expressjs.com/de/starter/hello-world.html)

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.listen(port, () => {
  console.log(`Example app listening at http://localhost:${port}`)
})
```