Server

Stack

- node
- express
- jade
- sequelize

Begründungen

Sequelize wurde verwendet, um Serverseitig eine Persistenz implementieren zu können. So kann in der Entwicklungsumgebung eine sqlite und auf Heroku eine postgres Datenbank verwendet werden.

Client

Stack

- backbone
- requirejs

Begründungen

Requirejs wurde verwendet, dass auch Clientseitig eine Modularisierung stattfindet. Collections werden so zwingend als Singleton erzeugt und es kann zu keinen "doppelten" Collections kommen. Views sind typischerweise keine Singletons.

Mit requirejs können Templates auch einfacher in separaten Dateien abgelegt werden, und bei bedarf gelesen werden.

Für den produktiven Einsatz würde man die Applikation mit j.js compilieren. Somit ergibt sich dann nur noch ein einziges .js File, welches der Browser laden müsste.

Sonstiges

Die API verfügt über ein Setup. Die URL */api/reset* löscht alle aktuellen Datensätze und */api/setup* erzeugt neue Beispieldatensätze.