Sprintdoc 1612
Reimplementation einer Webapplikation für Schulen

Pascal Ernst
70302367
Ostfalia Hochschule für angewandte Wissenschaften

März 2016

1 Implementation Elawa im Backend

Bei Elawa handelt es sich um eine Funktionalität, die Elternsprechtagswahlen ermöglichen soll. Dabei sollen im Backend Sprechzeiten mit einem Host (der Lehrer) in einem Raum, einer bestimmten Länge und einem Zeitpunkt erstellt werden können. Diese Sprechzeiten können auch geblockt sein (Pinkelpause") und nur ein Besucher kann sich pro Sprechzeit daran anmelden. Eine solche Elternsprechtagswahl kann sich auf mehrere Tage erstrecken.

1.1 Datenstruktur

Da es am Ende tausende von Sprechzeiten für etwa hundert Lehrer geben wird, muss das Backend eine einfache automatische Generierung der Daten bieten. Dafür teile Ich die Konfiguration in zwei Teile auf. Zum einen definiert der Administrator, welche Hosts für welchen Tag in welchem Raum sind. Dann definiert er die Zeiten für jeden Tag, an denen die Sprechzeiten stattfinden. Am Ende werden die Sprechzeiten (`Sessions`) generiert. Nun fehlt für diese nur noch der Besucher und ob diese Session eine Pause ist.

Somit kann Ich folgende Tabellen für die Datenbank mit dazugehörigen Models in Ruby on Rails definieren:

- `Event` Trennt die Elternsprechtagswahlen voneinander ab.
- `Segments` Tage einer Elternsprechtagswahl.
- `Sessions` Die einzelnen Sprechzeiten.
- `SegmentTimes` Die Zeiteinheiten für die Sprechzeiten.
- `SegmentPerformers` Die Hosts für die Sprechzeiten.

1.2 Design

Die Implementierung der Funktionalität im Client-Browser will Ich dieses mal komplett dynamisch und asynchron implementieren. Damit will Ich erreichen, dass server-seitig nur noch die API bereitgestellt werden muss und der Client die Darstellung der Daten übernimmt.

Damit kann Ich auch mehr Funktionalität auf eine Seite packen und die Bedienung der Elemente vereinfachen und verschnellern.

Für die Events erstelle Ich eine normale Index-Seite mit Tabelle für die einzelnen Events, ähnlich den bisherigen Seiten für Models die ein einfaches CRUD-Interface anbieten. Der Unterschied hier ist dass Ich alle Elemente (inklusive Create Update) mit auf dieser Seite als dynamische Elemente einbinde, was die Benutzbarkeit dieser verbessert.

Die Segmente werde Ich mit in die Übersicht-Seite eines jeweiligen Events packen, von wo auch die CRUD-Funktionen der Segmente möglich ist. Von dort aus sind dann auch die Einstellungen der SegmentTimes und der SegmentPerformers erreichbar.

1.3 Client-seitiger State

Ich habe mich weiter über Redux informiert und den Aufbau der clientseitigen Datenspeicherung verändert, um diesen kompatibler mit den vielen Funktionen zu machen, die das Backend erfüllen soll.

Dabei ist mir aufgefallen, dass die Informationssuche für moderne bleeding-edge Technologien anders ist. Statt offizieller Dokumentation und Tutorials ist man mehr auf "How-To's"von Programmierer-Blogs und aufgenommenen Vorträgen, die über diese Technologie diskutieren, angewiesen.

2 Sprint Aufgabenfertigstellung

Um den Programmaufbau des Clients konsistent zu halten während Ich darauf umstelle, dass allein der Client-Code für die Anzeige zuständig ist, musste Ich auch das gesamte Layout-Rendering der Seite auf ReactJS umstellen. Das alleine hat mich etwa 15 Stunden gekostet, sodass Ich in diesem Sprint nur die Implementierung der Events und der Segments geschafft habe. Die SegmentTimes und SegmentPerformers werden in den Sprint 1614 verschoben.