Sprintdoc 1614 & 1616
Reimplementation einer Webapplikation für Schulen

Pascal Ernst
70302367
Ostfalia Hochschule für angewandte Wissenschaften

Mai 2016

Die Beschreibung der Sprints 1614 und 1616 wurden zusammengefasst da sich die Zeitpunkte der Fertigstellung derer Aufgaben (Implementation in 1614 und Kundenbesprechung in 1616) überlappten.

1 Implementation der SegmentPerformers im Backend

Elternsprechtagswahlen-Events haben `Performer`, in dem Fall der Schulen sind das die Lehrer, die den einzelnen Segmenten zugewiesen sind (z.B. der Lehrer `Hans-Peter Reinhardt` ist dem Segment `Donnerstag` zugewiesen). Bisher waren die Zuweisungen etwas unhandlich zu benutzen. Das kann Ich mit der Interaktivität, die Ich mit ReactJS gewinne, verbessern. Ich habe ein neues Interface in Absprache mit dem Kunden designed. Es ist etwas komplexer, wodurch Ich dafür länger brauchen werde als geplant.

2 API Überarbeitung

Bisher waren die Daten, die der Server bei Anfragen auf die API-Route gesendet hat, nicht einheitlich. Manche gaben nur eine Nachricht zurück, andere die Daten eines oder mehrerer Objekte. Um das zu vereinheitlichen, habe Ich nach einem passenden Standard umgesehen, wie der Aufbau einer API für JSON aussehen kann. Der wahrscheinlich meisverbreitete Standard ist JSON-API, eine umfangreiche Definition mit vielen zusätzlich gesendeten Informationen, wie zum Beispiel normalisierten Daten und Indizes davon.

Da die Daten meiner Clients eher leichtgewichtig sind, habe Ich mich für eine andere Variante entschieden. Jede Resource wird in einen key mit Ihrem Namen gepackt (z.B. `bs/users`) und Daten dazu in einem separaten `meta`-key gepackt. Dies ist flexibel und einfach ohne viel Code oder eine Library umzusetzen.

Ich habe den bisherigen API-Code refactored, sodass Ich dieses Datenformat bei allen Endpoints zurückgesendet bekomme.

3 Updates der Software

Da die Library-Versionen inzwischen zwei Monate alt wahren, habe Ich diese aktualisiert. Innerhalb dieser Zeit gab es einige Libraries, die `breaking changes` eingeführt hatten, was einige Probleme verursacht hat. Unter anderem wurde NodeJS von Version 5 auf 6 geupdated, wodurch der `SASS`-Compiler anfangs nicht lief weil er diese Version nicht unterstützte. Meine einzige Möglichkeit war, NodeJS wieder herunterzusetzen bis die neue Version des SASS-Compilers veröffentlicht wurde.

Hier zeigt sich wieder, dass die Aufteilung des Servers und der Clients in Systemumgebungsunabhängige Docker-Container den Arbeitsaufwand im späteren Verlauf der Entwicklung verbessern könnte. Durch den etwas komplexeren Zusammenhang zwischen dem Server und den einzelnen Clients ist das aber Arbeit für eine ganze Woche, die bei mir nicht in den Zeitplan passt.

4 Besprechung mit dem Kunden

Ich habe mit dem Kunden über das alte Programm Babesk gesprochen und bin gemeinsam mit Ihm durch den Prozess gegangen. Es gab unerwartet viele Verbesserungsvorschläge und Fragen nach neuen Features. Sehr erfreulich war, dass Ich einige dieser Verbesserungen und Features bereits eingeplant oder eingebaut hatte. Durch meine gewonnene Erfahrung mit dem alten Programm konnte Ich einige Schwachstellen in der Persistenzschicht entfernen und gleichzeitig damit die Benutzerfreundlichkeit verbessern.

Ich habe auch die bereits funktionierenden Teile des neuen Systems vorgeführt, die ebenfalls erstaunlich positiv aufgenommen worden sind.

Der Termin der grundlegenden Fertigstellung wurde auf den 16. September gesetzt. Von da aus kann Ich ein funktionsfähiges System vorzeigen, bei dem sich der Kunde neue Features und Support dafür von mir kaufen kann.