Erstellen einer Vertretungsplan-Webseite

Kamal Abdellatif
Tim Gerlach

21. April 2016

Fachbetreuer:
Herr König

Inhaltsverzeichnis

1 Einleitung

Der Vertretungsplan hat schon seit längerer Zeit eine Online-Präsenz, welche intensiv von den Schülern dieses Gymnasiums genutzt wird. Doch nach Nachfragen unter den Schülern erkannten wir, dass eine Überarbeitung des Vertretungsplans notwendig geworden war. Vor allem der hochfrequenten Nutzung durch Mobilgeräte ist die aktuelle Version der Webseite nicht gewachsen. Daher nahmen wir uns im Rahmen des Informatikprojektes der zehnten Klasse eine Neugestaltung vor. Dabei setzen wir unsere Schwerpunkte auf Optimierung für mobile Endgeräte, Benutzerfreundlichkeit für sowohl Schüler als auch Lehrer und innovatives, responsives Design.

2 Theoretische Grundlagen

Wir verwenden für den Aufbau einer funktonierenden Kommunikation viele verschiedene Arten von Code, Dateiformaten und Übertragungsprotokollen. Ein kleiner Teil des Datenaustauschs ist zur Veranschaulichung hier erklärt.

Zunächst liegen die Daten als XML-Code, also durch sogenannte Tags gegliederter Text vor. Ein wichtiges Prinzip beim Generieren der Website ist das Umwandeln dieser in ein für den Server benutzbares Format. Wir haben uns dazu entschieden, für diesen Vorgang `Regular Expressions`, kurz `regexp` zu benutzen: Ein Platzhalterstring wird auf eine Quelle „gematcht“, d.h. alle Strings, die auf ein entsprechendes Muster passen, werden mit einer Schlüsselbezeichnung zurückgegeben. So wird der reguläre Ausdruck `/^[\t]*\$/` auf alle nur mit Leerzeichen und Tabs gefüllten Zeilen passen. Damit lassen sich verschachtelte Klassenangaben wie bsp. `10A/ 10ABCET1` auftrennen: die Ethikgruppe 1 der Klassen 10a, 10b und 10c ist hier gemeint.

Der XML-Code wird so Schritt für Schritt in eine JSON-Datei (`JavaScript Object Notation`) umgewandelt. Dieses ist ein weit verbreitetes Speicherformat, welches Daten in Schlüsseln und Werten dartellt, die durch Listen und Gruppen strukturiert sind.

Die ursprüngliche Quelle für das XML-Dokument liegt hier bei einem Stundenplanverwaltungsprogramm der Schule. Die dort generierten Dateien werden über die `/upload` Webseite hochgeladen. Dafür werden jQuery und AJaX verwendet. jQuery ist eine beliebte JavaScript Bibliothek, die dem Anwender viel Arbeit wie das manuelle Registrieren von EventListenern, die unter anderem für das Abfangen von Mausbewegungen oder Tastendrücken sorgen können, erspart. AJaX steht für „Asynchronous JavaScript and XML“und bedeutet, dass über JavaScript die Seite mit dem Server kommunizieren kann, ohne eine komplette HTML-Seite als Rahmen zu übermitteln. Damit können sämtliche Dateien problemlos auf den Server übertragen werden.

Nach dem Hochladen und dem Konvertieren sind die Dateien bereit, vom Client abgerufen zu werden. Dafür müssen sie in einen von einem Browser interpretierbaren Code umgewandelt werden, was in fast allen Fällen HTML, also `Hypertext Markup Language` ist. Dafür benutzen wir den Template-Parser von Pyramid, einem Serverframework. Es werden in einem HTML-Dokument Angaben gemacht, die vom Server mit entsprechenden Informationen ersetzt werden. Dabei wird das zuvor generierte JSON-Dokument verwendet.

Zusammen mit Stildateien und Scriptcode werden die Informationen an den Client übertragen. Dafür muss dieser zunächst ein Passwort angeben. Noch auf der Clientseite wird daraus eine Prüfsumme erstellt, aus der das ursprüngliche Passwort mit aktuellen technischen Möglichkeiten nicht ableitbar ist. Diese wird auch als `Hashsum` bezeichnet, was so viel wie „zerhackter Wert “bedeutet. Ein solcher wird mithilfe eines Requests an den Server übertragen, der die Prüfsumme mit einer Liste vergleicht und eine positive oder negative Rückmeldung an den Client sendet.

Vor dem Auslesen des Passworts ist der Kommunikationsweg zwischen Server und Client sicher, doch ein weiteres Risiko bleibt bestehen: Wer den Hashwert abfängt, kann sich als Client ausgeben und diesen direkt übertragen. Die einzige Lösung dagegen ist, den kompletten Datenaustausch End-to-End zu verschlüsseln. Eine Lösung hierfür ist SSL, welches durch ein etwas anderes Protokoll, `https` statt `http`, genau dies gewährleistet.

3 Lösungsidee

3.1 Wahl eines Webframeworks

Für die Umsetzung des Projektes war ein Webframework unabdingbar, welches einem Anforderungen wie Sicherheit, ein voll funktionsfähiger Webserver, Template-Rendering, Cookie-Policies und vielem mehr abnimmt. Jedoch waren viele Frameworks viel zu umfangreich, und erforderten viel überflüssigen Aufwand. Daher bat sich ein Microframework an. Auf Grund von unseren Vorkenntnissen und der schnellen Produktion entschieden wir uns für ein `Python`-basiertes Microframework. Schlussendlich fiel die Wahl auf das `Pyramid Web Framework`[1].

3.2 Struktur der Applikation

Die Applikation ist in drei Teile gegliedert: die `Python`-Scripts, den `Pyramid Web Server` und die eigentliche Seite. Sowohl die Scripts als auch der Webserver sind serverseitig, während die Seite beim Client agiert.
Die Seite muss eine Nutzeroberfläche anbieten, die die Daten vom Server dem Nutzer visualisiert, und die vom Nutzer eingegebenen Daten dem Server zustellt. Der `Pyramid Web Server` muss für sicheren den Datenaustausch zwischen den `Python`-Scripts und der Seite via sorgen. An höhster stelle stehen die `Python`-Scripts, welche die vom Nutzer eingegebenen Daten bearbeiten, speichern, und allen Nutzern zurückgeben müssen.

3.3 Server

3.3.1 Funktionen

Der Server der Webapplikation muss mehrere Aufgaben bewältigen. Ein Webserver muss `HTTP` bedienen können, um dem Browser des Nutzers eine `HTML`-Oberfläche bieten zu können. Dazu müssen Vetretungsplan-Daten auf dem Server gepeichert, abgerufen, und bearbeitet werden. Weiterhin muss der Server in der Lage sein, vom Nutzer hochgeladene Dateien zu empfangen, konvertieren und zu seinen gepeicherten Daten hinzuzufügen. Damit nur autorisierte Nutzer Daten bearbeiten können, ist eine sichere Authenifizierung über eine Passworteingabe des Nutzers erforderlich. Diese erforderliche Web-Sicherheit beinhält Hashwerte und die Verwendung von `SSL`.

[1] siehe `http://docs.pylonsproject.org/projects/pyramid/en/latest/index.html`

3.3.2 Struktur

Der Server baut auf den schon von Pyramid gegebenen Strukturen auf. In dem Webframework sind schon viele Teile eines Servers gekapselt, sodass wir uns nicht direkt damit außeinandersetzen mussten. Dies umfasst Authentifikation, also das Erkennen und Speichern von Nutzern, Autorisierung, die Einschränkung aller Nutzer auf ihre Erlaubnisbereiche. Weiterhin nimmt uns das Framework das gesamte Kommunikationsprotokolle `HTTP` ab. Somit läuft die gesamte Kommunikation mit dem Client über Pyramid ab. Dies umfasst das Senden von Daten via `GET`-Requests, das Empfangen von Daten durch `POST`-Requests und die Kommunikation via `AJaX`.
Am Back End des Servers befinden sich die Python-Scripts. Diese haben Zugriff auf die auf dem Server gespeicherten Daten, und steuern das Framework. Die gesamte applikationsspezifische Programmlogik wird innerhalb dieser Scripts ausgeführt. Die Scripts bekommen Nutzerdaten von Pyramid in Form Requests (Anfragen) vom Nutzer. Diese verarbeiten diese je nach Art des Requests, bearbeiten gegebenenfalls die gespeicherten Daten auf dem Server, und geben dem Framework alle Informationen für eine Antwort an den Client. Meist bestehen diese Informationen aus einzelnen Variablen.
Um dem Nutzer jedoch eine Antwort zu senden, müssen diese Daten noch in ein dem Nutzer verständlichen Format wie `HTML` gebracht werden. Diese Aufgabe wird von sogenannten Renderern erledigt. Renderer setzen aus einer Template (Vorlage), und den speziefischen Daten eine vollständige Antwort zusammen. Für jeden Datensatz von den Python-Scripts, welcher an den Client gesendet werden soll, gibt es eine bestimmte Template. So können zum Beispiel mit Leichtigkeit große, komplexe HTML-Seiten generieren, indem man unveränderliche Bestandteile in Templates auslagert, und nur die veränderlichen Daten der Scripts vom Renderer einsetzen zu lassen. Handelt es sich nur um unveränderlichen Daten in einer Template, wenn es sich beispielsweise um ein reines `CSS`-Stylesheet handelt, so ist kein Renderer von nöten. Dies wird durch sogenannte Static Assets realisiert, welche ganze Dateien unbearbeitet dem Nutzer zur Verfügung stellen.

3.3.3 Web-Sicherheit

3.3.4 Bearbeiten von Daten

Ablaufprotokoll

Konvertieren der Daten

3.3.5 Abrufen von Daten

3.3.6 Fehlerbehandlung

3.4 Client

Der Client ist in diesem Fall ein Browser eines Besuchers der Seite. Das Gerät des Benutzers kann sowohl ein normaler PC als auch ein Smartphone oder ein Tablet sein. Aus Gründen der Sicherheit und Privatsphäre kann ein Server nicht vor der eigentlichen Kommunikation zwischen diesen Gerätetypen unterscheiden. Während es theoretisch möglich ist, nach dem Herstellen einer Verbindung über `AJaX` Daten nachzuladen ist es praktischer, wenn die Seite eigenständig bestimmen kann, um welchen Gerätetyp es sich handelt. Dementsprechend ist eine Reaktion und Anpassung des Seitenlayouts nach dem Übertragen der Daten durch das Ausführen von Code durch den Client notwendig.

Außerdem muss die Seite ohne Serverunterstützung sämtliche Aufgaben durchführen können, für die eine erneute Datenübertragung schlichtweg zu Zeitaufwändig wäre. Beispiele hierfür sind das Filtern nach bestimmten Lehrern bzw. Klassen, das Anzeigen von Menüs, das Wechseln zwischen verschiedenen Tagen etc.

Um eine Seite dynamisch und unabhängig zu machen, werden Stil- und Skriptsprachen verwendet. Die gebräuchlichen sind `CSS` und `JavaScript`.

3.5 Design

3.5.1 Tabellendarstellung

3.5.2 Slides

3.6 Interaktivität

3.6.1 Filter

3.6.2 Cookies

3.6.3 Menü

4 Benutzung