1 Einleitung

Unsere eLearning-Plattform ist wie eine "Single-Page-Webanwendung", kurz SPA aufgebaut. Hier besteht die Webanwendung im Gegensatz zum klassischen Website-Design aus einem einzelnen HTML-Seite. Der Inhalt der Seite wird dann dynamisch zur Laufzeit nachgeladen. Dies hat den Vorteil, dass Informationen über mehrere Seitenwechsel hinweg (innerhalb der SPA) erhalten bleiben. Außerdem entfallen Ladezeiten zwischen den üblichen Seitenwechseln. Dies hat folgende Gründe:

- Zum Laden einer anderen Seite müssen nur die dafür notwendigen Informationen vom Server geladen werden. Der zugehörige HTML-Code wird dann clientseitig generiert.
- Das Laden der Informationen kann im Hintergrund geschehen. Dadurch geht der eigentliche Seitenwechsel sehr schnell.
- Style-Informationen, Side-Bars und Navigationsleisten innerhalb der Website müssen beim Seitenwechsel nicht neu geladen und erzeugt werden.

1.1 Lines-Of-Code

Sprache	Reiner Code	Mit Kommentaren
Java	9271	11247
JavaScript	4735	5970
HTML	834	903
CSS	5047	5246
Gesamt	19887	23366

2 Gesamtkonzeption

2.1 Client-Seitig

2.1.1 JavaScript

Die eLearning-Plattform ist, wie oben erwähnt, als SPA aufgebaut. Deshalb enthält die Clientseite sehr viel Logik. Der Server dient jetzt nur noch als eine Art "verlängerter Arm" der Datenbank. Er antwortet nur auf Daten-Anfragen des Clients und schickt die Daten dann zurück. Dies wird durch clientseitige AJAX-Calls aus dem JavaScript-Code realisiert. Leider müssen anfallende Sicherheits- und Zugangsprüfungen jetzt doppelt anfallen. Zum einen muss der JavaScript-Code dafür sorgen, dass der Benutzer keine ungültigen Aktionen tätigt bzw. dass er entsprechende Buttons (Erstellen, Löschen) erst gar nicht angezeigt bekommt. Zum anderen muss der Server ebenfalls alle Prüfungen durchführen um

Hackern den Zugang zu verwehren. Im Großen und Ganzen besteht der Server also nur aus diversen Zugangsprüfungen und der Weiterleitung von Daten. Der Client, genauergesagt der JavaScript-Code stellt die Logik der eLearning-Plattform dar. Er ist für die korrekte Darstellung, sowie für die Interaktion mit dem Benutzer verantwortlich.

2.1.2 HTML

Das HTML-Dokument enthält nur eine Art Grundgerüst der Website mit nicht veränderlichen Komponenten und es enthält unsichtbare HTML-Templates, die vom JavaScript-Code "geklont" werden um sie wo anders weiterzuverwenden.

2.2 Server-Seitig

2.2.1 Servlets

Die Servlets des Servers antworten eingehenden Anfragen ausschließlich mit JSON-Objekten. Es wird also serverseitig kein HTML-Code erzeugt. JSON-Objekte sind im Grunde genommen eine Liste von Key-Value-Paaren wobei Values entweder Strings oder Arrays (Liste von String-Values) sind. Das Ganze Objekt wird dann als reiner String übertragen und im JavaScript wieder als Objekt interpretiert. So lässt sich die Client-Server-Kommunikation problemlos realisieren.

2.2.2 Datenbank

Der nächste Punkt ist die persistente Datenspeicherung. Hier wird zum einen die SQL-Datenbank "MySQL" verwendet. Diese ist kostenlos und somit fiel die Wahl nicht schwer. Diese Datenbank verwaltet den Großteil aller Daten im System. Ein Datenbank-Manager stellt hierzu eine Schnittstelle bereit, die es den Servlets einen einfachen Zugriff auf die Daten erlaubt.

Der zweite Teil unseres Datenbank-Systems bildet die Graphdatenbank "Neo4J". Diese ist besonders gut geeignet für die Darstellung von Beziehungen in Graphen. Deshalb wurde sie von uns verwendet, um die Beziehungen zwischen Karteikarten zu speichern. Sie bietet einfache Funktionen als eine SQL-Datenbank um durch den Graph zu wandern, um Kindknoten zu suchen oder um Breiten und Tiefensuche zu betreiben. Dies sollte vor allem bei einer großen Anzahl an Karteikarten einen Performance-Vorteil bringen.