Server

Als Server wird eine virtuelle Maschine (VM) der Hochschule RheinMain (HSRM) verwendet. Auf dieser VM läuft das Betriebssystem Debian 11. Die Hardware wird von der HSRM zur Verfügung gestellt und von ihnen verwaltet. Auf dem Server werden Materialien für das LMS gespeichert und es läuft darauf die Datenbank. Der Server ist aufgrund von Sicherheitsvorkehrungen von der Hochschule nur über SSH erreichbar. Deswegen wurde für jeden Entwickler ein SSH-Keypair erstellt, welches auf dem Server und auf dem Rechner des Entwicklers hinterlegt ist.

Im Überordner „material“ werden die Dateien je nach Datentyp im Unterordner gespeichert (z.B. PDFs im Ordner „pdfs“, PNGs im Ordner „imgs“, usw.).

Mittels Shell-Skript wird jede Nacht eine Sicherung der Datenbank angelegt. Dabei werden alle Tabellen und Daten in einer SQL-Datei abgespeichert, sodass im Krisenfall die neueste SQL-Datei ausgeführt werden kann, wodurch der DB-Stand des Vortages wiederhergestellt werden kann. Diese Dateien bleiben 30 Tage vorhanden, sodass eine Backup-Datei nach 30 Tagen automatisch gelöscht wird.

Serverseitige Software

Aufgrund der Vorkenntnisse der Entwickler, wurde für das Backend die JavaScript-Laufzeitumgebung Node.js gewählt. Für Alternativen, wie bspw. Django in Python oder Spring Boot in Java hätte sich vorher eingelesen werden müssen, was zusätzliche Zeit in Anspruch genommen hätte. Neben diesem Hauptgrund bietet Node.js viele vorentwickelte Pakete an, welche dank des Paketmanagers npm einfach installiert und genutzt werden können.

Als Datenbank wurde sich für relationale Datenbank mit MySQL als DB-Verwaltungssystem (DBMS) entschieden. Für die Ausmaße dieses Projektes machen die Unterschiede der DBMS keinen Unterschied, weswegen das populärste Produkt gewählt wurde. Es wurde eine relationale DB und keine dokumentorientierte DB genommen, weil relationale DBs weniger komplex sind und eine höhere Konsistenz gegenüber dokumentorientieren DBs vorweisen.

Für das Seiten-Routing wird ExpressJS verwendet. Es vereinfacht Weiterleitungen und HTTP-Abfragen in Node.js. Mit dem Paket „express-session“ ist es möglich einem Nutzer eine Session im Browser zu geben, sodass Anmeldungen und Registrierungen durchgeführt werden können. Um Formulare vernünftig auslesen und weiterverarbeiten zu können werden die Pakete „formidable“ und „body-parser“ angewendet.

Für eines der Kernthemen, der PDF-Erzeugung, wird das Paket „pdf-lib“ eingesetzt. Damit können ein oder mehrere Bilder in eine PDF konvertiert werden. Zusätzlich gibt es Einstellungen für das Layout der zu erzeugenden PDF.

Mithilfe der Template-Sprache Embedded JavaScript (EJS) ist es möglich Daten aus der Datenbank serverseitig auf der Webseite anzuzeigen. Die Daten werden von Node.js an EJS übergeben und vor dem Laden der Webseite an die passenden Stellen des HTML gesetzt. Dementsprechend ist es möglich mithilfe von EJS Bedingungen oder Schleifen in den HTML-Code einzubauen.

Damit auf die Datenbank und die Materialien zugegriffen werden kann, muss auf den Server über SSH zugegriffen werden. Dieser Zugriff geschieht mit den Paketen „ssh2“ für die SSH-Verbindung und „mysql2“ für den Zugriff auf die Daten aus der Datenbank. Außerdem werden die Credentials (individuelle Zugangsdaten zum Server und der DB) bei jedem Entwickler lokal in einer „.env“-Datei gespeichert. Diese Credentials sind mit dem Paket „dotenv“ zugänglich.

Alle Pakete wurden aufgrund ihrer Popularität und Einfachheit in ihrer Anwendung gewählt. Dadurch konnten sich die Entwickler vergleichsweise schnell in die Techniken einlesen und hatten eine große Online-Community für Unterstützung und Nachfragen bei Fehlern oder Problemen.

Dateiorganisation

webserver

└ public

| └ assets

| | └ imgs

| | └ pdfs

| | └ videos

| └ css

| └ fonts

| └ script

└ server.js

└ .env

└ ssh-key

└ views

└ partials

node\_modules

package-lock.json

package.json

Um eine vernünftige Organisation des Projektes gewährleisten zu können, wurde eine vorab festgelegte Ordnung der Datei-Struktur eingehalten. In dem obigen Bild wurden die einzelnen Dateien mit Ausnahme von relevanten Dateien, die im Folgenden beschrieben werden, weggelassen. Dafür gibt es auf oberster Ebene die Dateien „package-lock.json“ und „package.json“, sowie den Ordner „node\_modules“, mit denen die von npm installierten Pakete verwaltet und dokumentiert werden. Alle neu erstellten Dateien und Programme befinden sich im Ordner „webserver“. Darin befindet sich der Ordner „public“, welcher alle Nicht-HTML-Dateien enthält, die für den Nutzer sichtbar sind. Im Unterordner „assets“ von „public“ befinden sich weitere Unterordner für die verschieden verwendeten Medien „imgs“, „pdfs“ und „videos“. In den Unterordnern „css“, „fonts“ und „script“ von „public“ liegen die Stylings, Schriftarten und clientseitigen Skripte der Webanwendung. Neben dem Ordner „public“ befinden sich im Ordner „webserver“ die serverseitigen Skripte, beginnend mit der Datei „server.js“, welche zum Start des Servers ausgeführt werden muss. In der Datei „.env“ befinden sich die Umgebungsvariablen (Environment Variables), welche für den Zugang zum Datei- und Datenbankserver benötigt werden. Auch wird der im Ordner „ssh-key“ gespeicherte Private Key des Nutzers für den Zugang zum Server verwendet. Im Ordner „views“ liegen die EJS-Seiten, welche als Template-Sprache zusätzlich zu HTML verwendet wird. Darin gibt es noch einen Unterordner „partials“ für EJS-Dateien, die keine eigenständige Seite darstellen, sondern nur von anderen EJS-Dateien eingebunden werden.

Neben dieser Ordnerstruktur wurde darauf geachtet, dass keine Monolithen entstehen, also der komplette Code innerhalb einer Datei entwickelt wird. Damit soll eine einfache Versionierung ermöglicht werden, sodass beispielsweise das Mergen mittels Git einfacher durchgeführt werden kann. Dementsprechend gibt es beim clientseitigen Styling über CSS und Skripting über JavaScript eine eigene Datei für jede EJS-Seite, sofern es notwendig ist. Trotzdem gibt es eine globale Skripting und Styling-Datei für Styles und Skripte, die auf mehreren oder allen Seiten angewendet wird, wie beispielsweise im Header bzw. der Suchleiste. Des Weiteren wurde der serverseitige Code auf mehrere Dateien ausgelagert, sodass in der Startdatei „server.js“ nur das Routing, die Nutzer-Session und die Datenbank-Befehle stattfinden. Zusätzliche Logik, wie die Datenbankanfragen oder die PDF-Generierung werden in separaten Dateien durchgeführt. Auch wurden redundante Elemente wie der Header der Webseite als „partial“ einmal entwickelt und anschließend von allen Dateien eingebunden.

Webanwendung starten

Um die Webanwendung zu starten, müssen folgende Schritte durchgeführt werden:

1. SSH-Keypair erstellen
   1. Public-Key auf dem Server in der Datei „./root/.ssh/authorized\_keys“ anfügen
   2. Private Key im Ordner „webserver/ssh-key/“ mit dem Namen „id\_rsa“ ablegen
2. Die Datei „.env“ mit den Umgebungsvariablen im Ordner „webserver/“ ablegen
3. Über VPN mit dem Hochschulnetz verbinden oder im Hochschulnetz sein
4. Node.js installieren
5. Den Befehl „npm install“ in der Konsole ausführen

Diese Schritte müssen einmalig zur Einrichtung durchgeführt werden. Anschließend kann der Server jedes Mal mit Schritt 6 gestartet werden.

1. Den Befehl „cd webserver; node server.js“ ausführen