Marius Backes und Tobias Meiser

Ausarbeitung – HTW Quiz

HTW Saarland – Wahlfach Webanwendungen

Inhalt

[Inhalt 1](#_Toc523153993)

[Motivation 2](#_Toc523153994)

[Konzeption 3](#_Toc523153995)

[Benutzeroberfläche 3](#_Toc523153996)

[Implementierung 4](#_Toc523153997)

[Struktur der Anwendung 4](#_Toc523153998)

[Datenbank 5](#_Toc523153999)

[Backend 6](#_Toc523154000)

[Frontend 7](#_Toc523154001)

[Testen der Webanwendung 8](#_Toc523154002)

# Motivation

Digitale Werkzeuge gewinnen immer mehr an Bedeutung, wenn es darum geht Wissen zu erwerben oder zu vertiefen. Um Vorlesungen interaktiver gestalten zu können und einen Überblick über den Kenntnisstand der Studenten zu bekommen können Professoren verschiedene Methoden anwenden. Ein beliebtes Werkzeug dafür ist die Webanwendung *Kahoot!*. Kahoot! ist eine spielbasierte Lernplattform, welche von Lehrern, Schülern und Professoren verwendet wird.

Laut einer Studie von Lea®n Inc. aus dem Jahr 2017 ist Kahoot! das 5. beliebteste digitale Werkzeug, welches zu Lernzwecken verwendet wird. Es führt die Kategorie „Bewertung“ vor Quizlet und SchoolCity an. [Quelle: <https://marketbrief.edweek.org/marketplace-k-12/40-popular-ed-tech-tools-k-12-identified-new-analysis/> ]

Die Anwendung kann verwendet werden um Quizspiele mit bestimmten Fragen zu erstellen. So kann ein Professor ein Spiel erstellen und während der Vorlesung das Spiel starten. Die Studenten können dann an diesem Spiel teilnehmen und die gestellten Fragen beantworten. Durch das korrekte Beantworten von Fragen können Spieler Punkte abhängig von ihrer Antwortzeit verdienen. Beim Abschluss des Spiels werden die drei besten Spieler angezeigt und jeder Spieler sieht seine eigene Platzierung. Diese Methode soll dazu dienen Studenten zum Lernen zu motivieren, da sie in den einzelnen Spielen gegeneinander antreten und dabei gleichzeitig ihr Wissen vertiefen können.

Das Problem hierbei ist, dass ein solches Spiel nur gespielt werden kann, wenn das Spiel vom Ersteller gestartet wird und man sich dann mittels des Spielcodes, der beim Erstellen generiert wird, einwählt. Zudem kann nur der Ersteller eines Spieles die Fragen dieses Spiels bearbeiten. Des Weiteren werden erworbene Punkte nur beim Abschluss des Spiels angezeigt und können nicht durch die Anwendung persistiert werden.

Aus diesem Grund soll eine Anwendung erstellt werden, welche dazu verwendet werden kann Fragenkataloge zu erstellen, welche zu jedem Zeitpunkt spielbar sein sollen und welche von mehreren Personen bearbeitet werden sollen. So können sich Studenten zu Lerngruppen zusammenschließen und gemeinsam Fragenkataloge erstellen. Zudem sollen erworbene Punkte gespeichert werden, sodass alle Spieler jederzeit ihre Highscores anschauen und mit ihren Kommilitonen vergleichen können.

# Konzeption

## Benutzeroberfläche

Die Anwendung wurde stark durch bereits vorhandene Anwendungen wie Kahoot! und Quizduell inspiriert. Aus diesem Grund wurde die Benutzeroberfläche für das Beantworten der Fragen, beziehungsweise dem Spielen, diesen etablierten Anwendungen nachempfunden. Die Anwendung soll möglichst einfach und intuitiv gestaltet sein.

**Startseite**

Die Startseite soll alle eigenen Fragenkataloge sowie Fragenkataloge zu denen man beiträgt anzeigen. Fragenkataloge sollen ausgewählt werden und gespielt werden können. Die nachfolgende Abbildung zeigt einen ersten Konzeptentwurf der Startseite der Anwendung.

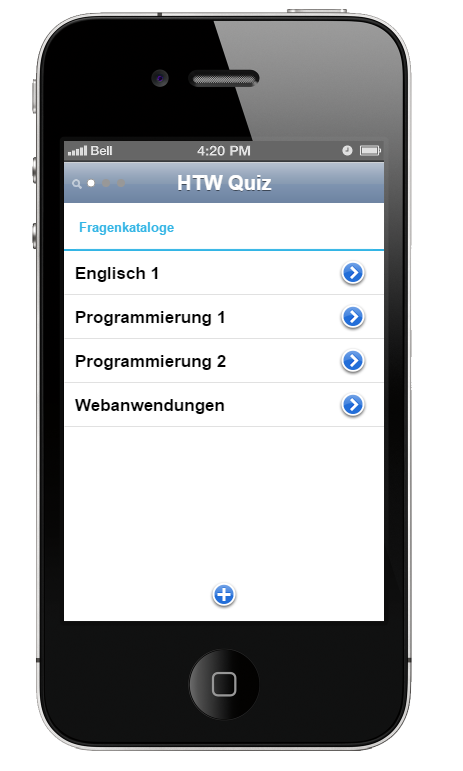


Abbildung: Mockup der Startseite

Die Auflistung enthält alle Fragenkataloge und jeder Fragenkatalog kann über den Pfeilknopf gestartet werden.

**Fragenkataloge spielen**

Die Webanwendung unterstützt Multiple-Choice-Fragen sowie Fragen, bei denen die Antwort vom Nutzer selbst eingegeben werden muss. Aus diesem Grund gibt es auch zwei verschiedene Ansichten für das Beantworten einer Frage.



Abbildung: Mockups der Benutzeroberfläche zum Beantworten der Fragen

Beim Beantworten der Multiple-Choice-Fragen müssen die verschiedenen Antwortmöglichkeiten gut sichtbar angezeigt werden und beispielsweise durch einen Button auswählbar sein. Beim Beantworten von Fragen, welche keine Antwortmöglichkeiten anbieten, muss ein Textfeld vorhanden sein, welches für die Eingabe der Antwort verwendet werden kann. Die vorangegangene Abbildung zeigt Mockups, welche zeigen wie dies umgesetzt werden kann.

# Implementierung

## Struktur der Anwendung

Die Anwendung unterteilt sich grundsätzlich in 2 Bereiche:

Das Backend, welches aus einer Datenbank, einer Loopback API und einem Node.js-Server besteht.

Das Frontend, welches Vue.js nutzt um die einzelnen Seiten der Anwendung anzuzeigen und mittels Axios die Loopback API ansteuert.

Der zentrale Ordner der Webanwendung lautet ***htw-quiz***. Die im weiteren Verlauf dieses Kapitels verwendeten Pfade sind relative Pfade und beziehen sich immer auf den zentralen Ordner.

Der zentrale Ordner der Webanwendung enthält 4 Unterordner. Die jeweiligen Ordner für das Backend (*api*) sowie das Frontend (*app*), welche im weiteren Verlauf des Kapitels näher erläutert werden. Zudem beinhaltet der zentrale Ordner die Unterordner *docs* und *test*.

Der Ordner *docs* enthält alle Dokumente, welche die Webanwendung, sowie dessen Planung, Erschaffung und Handhabung betreffen.

Im Ordner test befinden sich Skripte um das Backend der Webanwendung automatisiert zu testen. Das Testen der Webanwendung wird in einem separaten Kapitel erläutert.

## Datenbank

Als Datenbank wurde eine MySQL-Datenbank verwendet. Folgende Abbildung zeigt das Datenbankmodell mit Tabellen und Relationen.

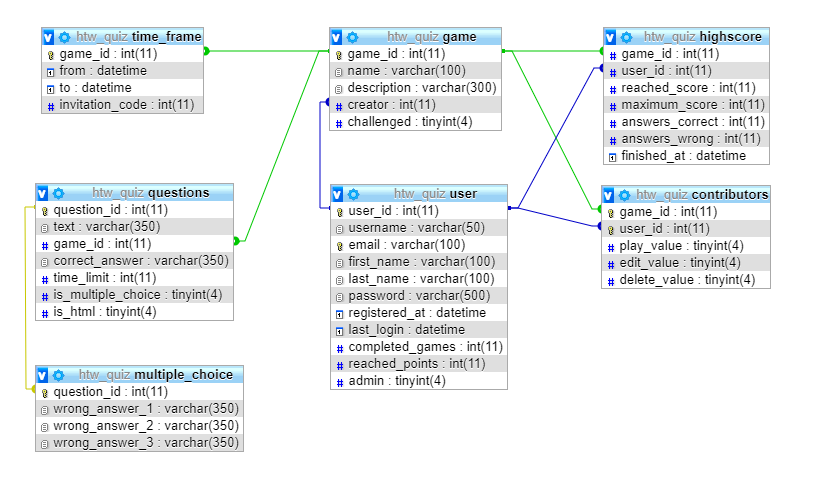


Abbildung: Datenbankmodell

Jeder Benutzer der Anwendung wird mit verschiedenen spezifischen Attributen in der Datenbanktabelle **user** persistiert. Die Passwörter der Nutzer werden als Hash in der Datenbank gespeichert. Es wird die Anzahl der gespielten Spiele des Nutzers und seine summierte erreichte Punktzahl gespeichert.

Die Fragenkataloge werden in der Tabelle **game** gespeichert. Jeder Fragenkatalog kann beliebig viele Fragen enthalten und optional einen Zeitrahmen erhalten, in welchem der Katalog gespielt werden kann. Dieser Zeitrahmen wird zusammen mit einem Invitation\_code in der Tabelle **time\_frame** gespeichert. Dieser Code wird zur Teilnahme an dem Spiel benötigt.

Die Tabelle **questions** enthält alle Fragen. Eine Frage beinhaltet ihren Text, ein Zeitlimit und eine korrekte Antwort. Zusätzlich kann die Frage im HTML-Format gespeichert werden und es kann gewählt werden ob die Frage eine Multiple-Choice-Frage sein soll. Ist die Frage eine Multiple-Choice-Frage, so wird in der Tabelle **multiple-choice** die ID dieser Frage und die restlichen 3 falschen Antwortmöglichkeiten gespeichert.

Die erzielten Punkte der Nutzer zu jedem ihrer getätigten Spiele wird in der Tabelle **highscore** persistiert.

Die Tabelle **contributers** ist eine Zuordnungstabelle. Sie dient dem Zweck einzelnen Spielen, Benutzer zuzuordnen, welche diese Spiele spielen oder bearbeiten dürfen. Ein Datensatz dieser Tabelle beinhaltet die Spiel-ID sowie die ID des jeweiligen Benutzers und enthält drei Booleanwerte, welche die jeweiligen Rechte bestimmen.

## Backend

Der Pfad des zentralen Ordners in welchem sich der Source-Code des Backends befindet lautet *api/server*. Die nachfolgende Abbildung zeigt die Ordnerstruktur des Backends.

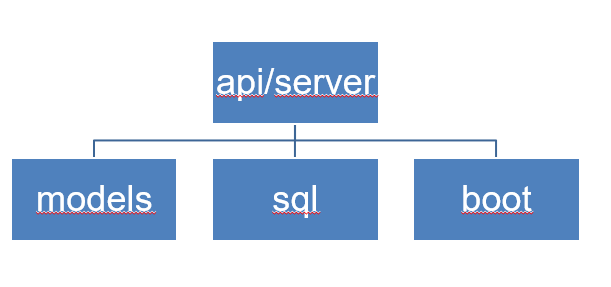


Abbildung: Ordnerstruktur des Backends

Im zentralen Ordner des Backends befinden sich 3 Unterordner.

Die sich im *boot*-Ordner befindliche Datei startet den loopback Server.

Im Ordner *sql* befinden sich die Skripte zum Anlegen der Datenbank sowie des dazugehörigen Datenbankusers.

Der Ordner *models* enthält die Abbildungen der Datenbanktabellen sowie deren Eigenschaften. Ein Model stellt dabei jeweils eine Datenbanktabelle dar und besteht aus einer .json sowie einer .js Datei. Die json-Datei beinhaltet dabei die Struktur und die Attribute der Datenbanktabellen. In den jeweiligen js Dateien für jedes Model werden die Daten in der Datenbank gespeichert oder von dort geladen um sie an den Client zurückzugeben.

Zusätzlich zu diesen drei Ordnern befinden sich noch einige andere Dateien im zentralen Backend-Ordner:

Die Datei *component-config.json* enthält Konfigurationen für den Loopback-Component-Explorer.

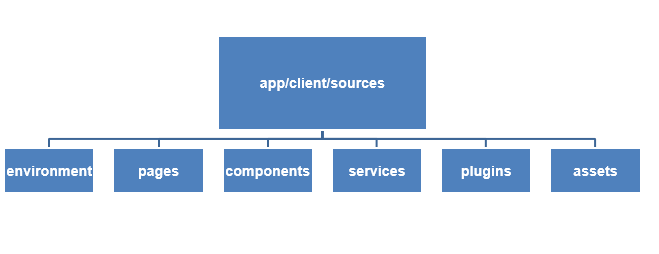
Die Datei *config.json* enthält Konfigurationen für den Loopback-Server wie beispielsweise den Port.

Die Datei *datasources.json* enthält Informationen über Host und Port der Datenbank sowie die benötigten Logindaten zur Datenbank.

Die zentrale Datei, welche den Server startet: ***server.js.***

## Frontend

Der Pfad des zentralen Ordners in welchem sich der Source-Code des Frontends befindet lautet *app/client/src*. Die nachfolgende Abbildung zeigt die Ordnerstruktur des Frontends.



Der Ordner environment enthält die Datei environment.js. Diese Datei exportiert ein Objekt, welches die URL zum Server beinhaltet, um diese von den Service-Methoden nutzen zu können und damit eine Kommunikation zum Backend zu ermöglichen.

Der Ordner *pages* enthält alle Vue Dateien, welche dazu dienen eine Ansicht im Browser generieren.

Im Ordner *components* befinden sich Komponenten, welche einen wiederverwendbaren Code ermöglichen oder aufwendige Passagen auslagern.

Der Ordner *services* enthält Dateien welche wiederum Methoden beinhalten um mit dem Backend zu kommunizieren.

Als Material Design Framework wird vuetifyjs verwendet und die Referenz dazu ist im Ordner *plugins* eingebunden. Diese Referenz wird von der Vue CLI beim Erstellen eines vuetify Projekts automatisch erstellt.

Im Ordner *assets* befinden sich alle Grafikdateien die ins Projekt eingebunden werden.

Der zentrale Ordner enthält neben diesen Unterordnern noch einige weitere Dateien:

Die Datei *router.js*, welche das Routerobjekt exportiert. Der Router enthält Name, Pfad und die entsprechende Komponente der einzelnen Pages von Vue.

Die zentrale Datei, welche die Webanwendung startet: ***App.vue***.

# Testen der Webanwendung

Um das Testen der Webanwendung möglichst effizient zu tätigen, wurden automatisierte Tests geschrieben. Diese Tests befinden sich im Ordner *test/src* und können nach Installation aller benötigten Abhängigkeiten mit dem Kommando **npm test** gestartet werden.

Die automatisierten Tests testen dabei alle API-Requests. Dabei ist jedoch zu beachten, dass der Server gestartet und die Datenbank komplett installiert sein muss.

Nachdem das Backend getestet wurde, wurde das Frontend getestet. Das Frontend wurde manuell von den Entwicklern getestet und auf Fehler untersucht.