Das ist ein Bingo!

Dynamische Singlepage-Website - Web-Engineering 1 HSZG

Maximilian Lehmann

23. April 2018

Inhaltsverzeichnis

1 Einleitung

Im Folgenden werde ich mein Projekt “Das ist ein Bingo!“ näher erläutern. In der Modulvorlesung Web-Engineering 1 wurden wir vor die Aufgabe gestellt, eine dynamische Singlepage-Application für den Webbrowser unseres eigenen Designs zu entwickeln. Ich habe mich für die Umsetzung eines Spiels entschieden, das sogenannte Wort-Bingo.

Wort-Bingo bezeichnet ein Spiel, bei dem jeder Spieler vor sich einen Spielzettel hat mit zum Beispiel 6 mal 6 Quadraten aufgedruckt, jedes Quadrat gefüllt mit einem Wort. Ziel des Spieles ist, während eines Gespräches, Vortrages oder Ähnlichem ein vorgekommenes Wort auf seinem Zettel ab zu-haken. Derjenige Spieler welcher zuerst eine waagerechte oder senkrechte Reihe “voll“ hat, gewinnt das Spiel. Die im Spiel verwendeten Wörter passen meist thematisch zum Kontext, in dem das Spiel stattfindet. Zum Beispiel können IT-Fachbegriffe verwendet werden, während eine entsprechende Vorlesung besucht wird.

Als Herausforderung habe ich mir einige Besondere Applikationsmerkmale als Ziel gestellt.:

- Der Benutzer soll innerhalb der Anwendung bestehende Wortlisten verwalten und erweitern können.
- Jedes Spiel stellt einen Spielraum dar. Spielräume können beliebig oft eröffnet und im nach-hinein betreten werden können.
- Während eines Spiels erhalten die Nutzer Feedback über de Fortschritt ihrer Mitspieler.

Zur Entwicklung habe ich mich auf verschiedene Technologien verlassen. Namentlich für den Hauptentwurf der Seite HTML5, CSS3 und Javascript. Ich habe mit Absicht auf jQuery verzichtet um Einerseits die Seite etwas schlanker zu halten, aber hauptsächlich als Programmierübung in puren Javascript. Des weiteren GitHub, als zentrale Ablage für den Code und die Versionierung, NodeJS für die Serverkommunikation und MaterializeCSS als Framework, um das Design der Website erheblich zu erleichtern.

2 Aufbau der Singlepage

Die Applikation ist in sogenannte Ansichten unterteilt, 4 an der Zahl. Anfangs wurde das Aussehen der Website statisch mit Hilfe vom MaterializeCSS Framework vorgearbeitet. Anschließend wurden nach und nach die benötigten Ansichten und natürlich die eigentlichen dynamischen Funktionalitäten implementiert.

HTML5 entsprechend besteht die Applikation aus einem *header*, *main* und *footer* -Part. In *main* wird je nach Ansicht der entsprechende Inhalt generiert, *footer* stellt Kontextspezifisch die Navigation zur Verfügung und bietet dem Benutzer Feedback, wie viele andere Nutzer und Spielräume auf dem Server vorhanden sind.

2.1 Startansicht

Test.

2.2 Konfigurationsansicht

2.3 Raumansicht

2.4 Spielansicht

3 Verwendete Technologien

3.1 Unterstützende Software

3.1.1 Atom

Als IDE/Editor für mein Projekt habe ich Atom gewählt. Besonders hilfreich war die leichte Erweiterbarkeit durch Plugins, zum Beispiel das Live-Server Package, welches erheblich beim Testen vom Serverside-Code geholfen hat, ebenso Syntax-Highlighting und Code-Completion.

3.1.2 Firefox Developer Toolbox

Die integrierte Developer Toolbox vom Firefox-Browser war enorm Arbeits-erleichternd beim Entwickeln des Frontends, insbesondere das Inspektions-Werkzeug beim Design von HTML und CSS und die Konsole beim Debugging von Clientside-Javascript.

3.1.3 GitHub

Da ich öfters zwischen Desktop-Computer und Laptop gewechselt habe, war die Zentrale Code-Ablage von GitHub wirklich sehr hilfreich. Und ebenso natürlich die Versionskontrolle, auch wenn ich diese bei so einem kleinen Ein-Mann Projekt nicht gebraucht habe.

3.2 Frontend Software

3.2.1 MaterializeCSS

3.3 Backend Software

3.3.1 NodeJS

4 Abschlusswort