Marienschule Opladen

Facharbeit Informatik

Black Jack Online Spiel

Jonas Zellerhoff

supervised by
Matthias Kühnert

Opladen, 2021/22

Inhaltsverzeichnis

1 Einleitung

1.1 Was genau?

Zunächst einmal stellte sich mir natürlich die Frage, was ich überhaupt programmieren möchte. Zu dieser Frage gab es für mich schnell eine sehr klare Antwort: Ein Spiel, und zwar Black Jack. Dies schoss mir sofort in den Kopf, da ich erstens sehr gerne alles mögliche spiele, deshalb ein Spiel, und zweitens sehr gerne alles mögliche mit Karten mache und Black Jack ein nicht so leicht zu programmierendes Programm, aber ein leicht zu verstehendes Spiel ist.

1.2 In welcher Sprache?

Als nächstes stellte sich mir dann die Frage, in welcher Programmiersprache ich das Spiel programmieren möchte, dazu musste ich mir dann natürlich erst einmal paar Gedanken machen, was mein Spiel alles können soll, um dann die jeweiligen Vorteile der Programmiersprachen abzuwägen und zu entscheiden, welche am geeignetsten ist. Da ich mein Spiel als Online Server Spiel programmieren möchte, bot sich dafür JavaScript an. Da ich im Unterricht bisher nur Java kennengelernt hatte, musste ich also erst einmal die Grundlagen von JavaScript verstehen, bevor ich mich ans Eingemachte: die BackEnd (Server Programmierung) begeben konnte. Als ich mir dann die Grundlagen von JavaScript angeeignet hatte, habe ich mir einen Videokurs zum Thema Node.js und express.js angesehen, um in die Server Programmierung einzusteigen.

2 Hauptteil

2.1 Struktur

Im Folgenden werde ich kurz meine Projekt-Struktur darstellen:

2.1.1 Server

Der Server wird in der Klasse index.js erstellt, dabei verwende ich express.js und socket.io. Der Server wird aufgerufen von der Klasse client.js, die den Client in gewisser Weise wiederspiegelt und ebenfalls auf die Eingaben vom Client hört und dann die dazu passende Funktion ausführt.

2.1.2 HTML-CSS

In meinem Programm gibt es vier verschiedene HTML Seiten:

=> 1. : index.html
Die erste ist dafür da, dass wenn man auf den Server draufgeht, dass man zunächst einmal seinen Display-Namen und den Room-Namen eingeben kann. Dabei kann man seinen Namen beliebig wählen und den Raum zunächst einmal auch, möchte man allerdings dem gleichen Raum von einem Freund joinen, muss man den gleichen Namen des Raums eingeben, wobei Groß- und Kleinschreibung nicht zu beachten ist, da sowohl Username, als auch Roomname zu kleinen Buchstaben umgewandelt werden.

=> 2. : server.html
Auf dieser Seite läuft der Server und man wird von der index.html Seite direkt auf diese geleitet. Hier hat man im unteren Drittel die Möglichkeit, mit den anderen, die im gleichen Raum sind, zu chatten. Auf der linken Seite ist die sogenannte sidebar, wo der Room-Name und die Users, die in dem Raum sind, angezeigt werden. In der Mitte ist ein großes Spielfeld, auf dem es nach Klicken des Start-Game Buttons möglich ist, mit bis zu vier Spielern das Spiel Black Jack zu spielen. Auf der rechten Seite sind sechs Buttons, der eine führt zum Black Jack Spiel gegen einen Bot und der andere zu den Regeln. Die restlichen vier Buttons sind dafür da, sich als Player 1/ 2/ 3/ 4 auszugeben, sind Player 1 und Player 2 vergeben, ist es möglich auf dem Server das Spiel zu starten.

=> 3. rules.html
Hier werden ausschließlich die Regeln für das Spiel Black Jack angezeigt. Ist man sich noch nicht ganz sicher, wie das Spiel funktioniert, kann man sich mit ein paar Spielen gegen den Bot beweisen und das Spiel erlernen.

=> 4. : blackJackBot.html
Zu guter Letzt bleibt noch das Spiel gegen den Bot / Carddealer übrig. Hier kann man nach dem Drücken des Start-Game Buttons Erfahrung im Spiel gegen den Carddealer / Kartengeber sammeln, um danach für eine oder mehrere Partien Black Jack gegen seine Freunde gewappnet zu sein.

2.1.3 JavaScript

Wie oben schon mal erwähnt, habe ich zwei JavaScript Dateien, die für den Server verantwortlich sind: index.js und client.js

=> utils package
Im utils package sind zwei kleine aber feine Dateien, die es ermöglichen, die Nachrichten richtig zu formatieren und anzugeben (message.js) und die dabei helfen, herauszufinden wenn User dem Raum beitreten oder ihn verlassen, aber auch herauszufinden, wer gerade im angegebenen Raum ist (users.js).

=> blackjack.js
In dieser Datei habe ich in Plain JavaScript die Mechanik des Spiels programmiert, ein Array mit dem Namen der Karte, dem Wert der Karte und ein Teil der img-src der Karte von der deckofcardsapi. Die Karte wird mit Math.random erstellt und es wird dann der jeweilige Teil des documents mit Hilfe des querySelectors getrackt und dann um den richtigen Teil der Karte zum Beispiel ergänzt:

```
1       let card = Math.round(Math.random() * 51)
2           console.log(cards[card].word)
3
4           player1.innerHTML += `<img src="${imgSource}${cards[
                card].img}" width="80px" height="111.15px"/>`
```

Eine leicht abgewandelte Version dieser Datei findet sich ebenfalls in client.js wieder, da dort die Spielmechanik ja auch an den Server übertragen werden muss und es anstatt einen Spieler vier Spieler gibt.

2.2 Probleme

Im Folgenden werde ich kurz auf zwei Probleme, die ich hatte, eingehen und erläutern, wie ich sie gelöst habe.

2.2.1 Infinite Loop / Infinite TimeOut

Bei der Mechanik für den Kartengeber habe ich beim Spiel eine while Schleife verwendet, um die Karten zu erzeugen, dabei bin ich dann auf das Problem gestoßen, dass dadurch, dass der Computer so schnell ist, einfach keine Spannung entsteht, da alle Karten des Kartengebers fast gleichzeitig angezeigt werden. Somit dachte ich mir dann ganz einfach, dass ich einfach nur die von JavaScript eingebaute Funktion setTimeOut verwenden muss, dabei entstand jedoch dann das Problem, dass sich dann alles aufgehangen hat, da die Timeouts so schnell hintereinder gestartet wurden und somit so viele Timeouts gleichzeitig gestartet wurden, dass man nicht mehr aus dem Timeout rauskam. Somit habe ich dann eine Variable i benutzt, die nach jedem Timeout um 1 addiert wird, und dann immer mit 1000 Mal genommen wird, somit

habe ich dann die gewünschte Pause von einer Sekunde erreichen können, da die Timeouts im Prinzip auch noch gleichzeitig gestartet werden, aber unterschiedlich lang dauern und sie sich somit nicht immer weiter aufstocken.

2.2.2 Different Cards when clicking (On the Server)

Als ich dann die Mechanik vom Spiel zu Client.js übertragen und verändert hatte, habe ich es ausprobiert und es sah zunächst einmal ganz gut aus. Als ich dann jedoch in einem zweiten Tab dem Server beigetreten bin, habe ich gemerkt, dass bei der Erstellung der Karte ein Fehler vorliegt. Da die Karte mit einer zufälligen Nummer erstellt wurde, war diese zufällige Nummer in den 2 Tabs, die zwei User darstellen sollen, anders, da die Karte auf beiden Tabs zufällig erstellt wurde und es somit sehr selten zu einer Übereinstimmung der Karte führte. Dieses Problem habe ich dahingehend gelöst, dass ich die Karte nicht mehr außerhalb der Server Methode (socket.emit) erstellt habe, sondern sie dann innerhalb dieser Methode erstellt habe. Dann habe ich diese erstellte Karte als Parameter den nächsten Funktionen zugewiesen, sodass auf allen Geräten jetzt immer die gleiche Karte angezeigt wird.

2.2.3 Different Cards at the Carddealer

Das gerade beschriebene Problem ereignete sich logischer Weise auch beim Kartengeber. Hier musste ich jedoch eine kleine andere Lösung finden, da hier die Karten nicht beim Klick erzeugt werden sollen, sondern automatisch, wenn er am Zug ist. Diese automatische Kartenzuweisung habe ich dann so gelöst, dass ich in der socket.emit Methode 6 Karten erstellt habe, die ich dann ebenfalls als Parameter weiter gegeben habe. Ich habe 6 Karten erstellt, da dies die maximale Anzahl an Karten war, die ich in mehreren 100 Versuchen benötigt habe, um den Score 17 zu erreichen. Jetzt wird nicht mehr mit einer while Schleife jede Karte angezeigt, sondern mit verschieden if-Bedingungen.

3 Schluss

Zum Schluss kann ich sagen, dass es mir extrem viel Spaß gemacht hat, meine Arbeitsergebnisse Woche für Woche vor Augen zu führen und zu sehen, wie viel ich geschafft habe. Während dieser Facharbeit habe ich sehr viel gelernt, ich habe mich in JavaScript hineingearbeitet und mir macht es jetzt richtig Spaß, zwischen zwei Programmiersprachen hin und her wechseln zu können, je nachdem auf was ich gerade ich Lust habe. Außerdem habe ich gelernt und auch verstanden, wie man einen Server mit Hilfe von express.js und socket.io programmiert.

4 Anhang

4.1 Mein Projekt

I. GitHub: https://github.com/JonPro0/BlackJack-Server-Game
II. Veröffentlicht mit Heroku: https://blackjackzelle.herokuapp.com/

4.2 Quellen

III. JavaScript Tutorial: https://www.youtube.com/playlist?list=PLNmsVeXQZj7qOfMI2ZNk-LXUAiXKrwDIi
IV. Nodejs/Expressjs Tutorial: https://www.udemy.com/course/the-complete-nodejs-developer-course-2/
V. Verschiedenste HTML/CSS Tutorials
V.I. https://www.youtube.com/watch?v=dOgFkZiVC8w
V.II. https://www.youtube.com/watch?v=D-h8L5hgW-w
V.III. https://www.w3schools.com/howto/howto_css_animate_buttons.asp
V.IV. https://www.sliderrevolution.com/resources/css-button-hover-effects/

Viel Spaß beim Ausprobieren meines eigenen

Black Jack Online Spiels