Inhaltsverzeichnis

KAPITEL 5

Konzeption des Clients

In diesem Kapitel soll die Konzeption für den Schachclient näher erläutert werden, welches als Grundbaustein für die Implementierung dienen soll.

5.1 Anforderungen

Als grundlegende Anforderung an den Client ist die Verwaltung von Playern und Matches zu sehen. Dabei soll über die Startseite eine Möglichkeit bestehen auf die jeweiligen Verwaltungsseiten und wieder zurück zu gelangen. Auf den Unterseiten sollen dabei auf der linken Seite eine Übersichtsliste zu den bisher angelegten Playern oder Matches angezeigt werden. Auf der Rechten Seite soll ein Formular zur Erstellung der Objekte bereitstehen. Innerhalb der Liste soll für jedes Element eine Toolbar verfügbar sein, über welche Aktionen angesteuert werden können.

Für einen Player muss dabei die Möglichkeit bestehen gelöscht oder bearbeitet werden zu können. Im Falle einer Bearbeitung soll das Formular zur Erstellung ausgetauscht werden. Sofern erfolgreich der Player aktualisiert oder die Bearbeitung abgebrochen wird, wird das Formular wieder zurückgetauscht.

Bei einem Match soll neben dem Löschen die Möglichkeit bestehen es zu starten. Sobald ein Match gestartet wird soll eine Weiterleitung zu einer neue Seite erfolgen. Auf dieser sollen sich auf der linken Seite ein Schachbrett und auf der rechten Seite Statusinformationen zum gestarteten Match befinden, wie zum Beispiel eine Liste aller bisher gespielten Züge. Die sich auf dem Schachbrett befindlichen Figuren sollen dabei mittels Drag&Drop bewegt werden können. Für eine vereinfachte Spielweise sollen Figuren ihre möglichen Spielzüge via *Mouseover* anzeigen.

Des weiteren soll der Client registrieren sobald ein Spieler Schach gesetzt wurde und dies in den Statusinformationen anzeigen. Sobald ein Spieler im Schach steht, sollen nur noch Züge möglich sein, um die Situation zu beenden. Sollte keine Möglichkeit bestehen aus dem Schach zu gelangen, soll eine Meldung mit „Schachmatt“ erscheinen. Anschließend sollen keine Züge mehr durchführbar sein.

Einbindung der KI definieren

5.2 Verwendete Bibliotheken/Frameworks

5.2.1 RequireJS

RequireJS ist ein für JavaScript entwickelte Bibliothek zum laden von Modulen. Dabei ist es für Nutzung innerhalb des Browsers optimiert, kann aber auch für andere Umgebungen genutzt werden. Ziel von solchen Bibliotheken wie RequireJs soll sein den eigenen Code zu beschleunigen und die Qualität zu steigern.[1]

Für die Modulbeschreibung innerhalb von JavaScript stehen mehrere Formate bereit. RequireJS setzt dabei auf das Format Asynchronous Module Definition (AMD). Das Listing 5.1 stellt ein einfaches Beispiel für die Definition eines AMD-Moduls bereit. In dem Zeitschriftartikel [Bra17] können zu den einzelnen Modul-Formaten und deren Einsatzmöglichkeiten nähere Informationen nachgelesen werden.

```
1 define(['jquery'] , function ($) {
2   return function () {};
3 });
```

Listing 5.1: Beispiel: Moduldefinition mittels Asynchronous Module Definition (AMD)[2]

5.2.2 kotlinx.html

Die Kotlinx.html ist eine offiziell von JetBrains entwickelte Bibliothek, welche eine Domain-specific language (DSL) für das Erstellen bzw. Ergänzen von HTML-Code bereitstellt. Sie kann dabei für die Java Virtual Machine (JVM) oder JavaScript Plattform verwendet werden.

Diese Bibliothek ermöglicht es den sämtlichen HTML-Code in Kotlin-Code auszulagern. Das bringt einen großen Vorteil mit, denn das Erstellen des Codes wir durch eine statisch Typisierung abgesichert. Dadurch können mögliche Fehler im HTML-Code bereits während der Übersetzung des Quellcodes erkannt werden. Vergessene oder gar falsch verschachtelte HTML-Tags werden dadurch vermieden. Ein Beispiel für die Verwendung stellt das Listing 5.2 dar, welches den HTML-Code aus Listing 5.3 generiert.

```
1 import kotlinx.html.*
2 import kotlinx.html.dom.*
```

1 siehe [ANOa]
2 Quelle: [ANOb]

```
3
4 val myDiv = document.create.div {
5   p { +"text inside" }
6 }
```

Listing 5.2: Beispiel: Verwendung der Bibliothek kontlinx.html[1]

```
1 <div>
2   <p>
3     text inside
4   </p>
5 </div>
```

Listing 5.3: Beispiel: Verwendung der Bibliothek kotlinx.html (Ergebnis)

5.2.3 kotlinx.serialization

5.3 Informationsermittlung für den Datenaustausch zwischen Client und Server

1 Quelle: [Mas]

Literatur

[ANOa] ANONYM: „Require.js. A JavaScript Module Loader“. (), Bd. URL: http://requirejs.org/ (besucht am 13. 04. 2018) (siehe S. 16).

[ANOb] ANONYM: „Why AMD?“ (), Bd. URL: http://requirejs.org/docs/whyamd.html (besucht am 13. 04. 2018) (siehe S. 16, 27).

[Bra17] BRAUN, HERBERT: „Modul.js. Formate und Werkzeuge für JavaScript-Module“. *c't Heft 3/2017* (2017), Bd.: S. 128–133 (siehe S. 16).

[Mas] MASHKOV, SERGEY: „DOM trees“. (), Bd. URL: https://github.com/Kotlin/kotlinx.html/wiki/DOM-trees (besucht am 13. 04. 2018) (siehe S. 17, 27).

[Wat] WATSON, GRAY: „OrmLite - Lightweight Object Relational Mapping (ORM) Java Package“. (), Bd. URL: http://ormlite.com/ (besucht am 10. 04. 2018) (siehe S. 27).

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Quelle: [Wat]
1 verändert nach [Wat]
1 Quelle: [ANOb]
2 Quelle: [Mas]

Acknowledgments

I thank ?? and ?? for giving me the opportunity to write this bachelor/master/phd thesis at ??, and for their professional advise.

I thank in particular the ?? team who readily/willingly provided information at any time and ??.

I would also like to than all people who supported me in writing this thesis.

Erklärung der Selbstständigkeit

Hiermit versichere ich, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die Zitate deutlich kenntlich gemacht zu haben.

Dresden, den 14. April 2018 Felix Dimmel

A First chapter of appendix

A.1 Parameters