Inhaltsverzeichnis

KAPITEL 4

Konzept des Servers

Inhalt dieses Kapitels soll die Planung sein, welche für die Umsetzung des RESTful Schachservers benötigt wird. Dabei dient der erste Abschnitt für eine Erläuterung der Anforderungen, welche der Server mitbringen bzw. erfüllen soll. Im zweiten Abschnitt werden die verwendeten Bibliotheken bzw. Frameworks in den Punkten Zweck, Einrichtung und Verwendung näher erläutert. Der letzte Abschnitt dieses Kapitels befasst sich anschließend damit, wie der Zugriff auf einzelne Ressourcen des REST-Server erfolgen soll. Dabei werden alle möglichen Request-Methoden für die jeweiligen Ressourcen näher beleuchtet.

4.1 Anforderungen

Die Grundanforderungen an den RESTful Schachserver sollen in erster Linie die Bereitstellung aller benötigten Ressourcen sein. Dabei sollen Elemente erstellt, ggf. bearbeitet und gelöscht werden können. Zusätzlich soll die Möglichkeit bestehen, einzelne oder alle gespeicherten Elemente einer Ressource abzufragen. Beim erstellen eines neuen Ressourcenelements soll dieses in einer SQLite Datenbank gespeichert und die ID automatisch durch SQLite generiert werden.

Um ein Schachspiel abzubilden bedarf es dabei der Ressourcen Player (Spieler), Match (Partie) und Draw (Zug), welche in den nachfolgenden Unterabschnitten 4.1.1 bis 4.1.3 näher betrachtet werden.

Als abschließende Anforderung ist noch die Fehlerresistenz zu erwähnen. Denn die im Rahmen dieser Arbeit entstandene Praktikumsaufgabe

Verweis auf Praktikumsaufgabe im Anhang

soll durch zukünftige Studenten absolviert werden, wobei der Server als Grundlage dienen soll.

4.1.1 Ressource: Player (Spieler)

Neben der ID, welche schon im Abschnitt 4.1 erwähnt und durch die SQLite Datenbank generiert werden soll, besitzt der Player noch Informationen über seinen Name und sein Passwort.

Nach dem anlegen eines neuen Players soll eine Änderung des Namens nicht gestattet sein, die des Passwortes hingegen schon.

4.1.2 Ressource: Match (Partie)

Neben der ID besitzt ein Match Informationen über die beiden Spielteilnehmer und deren Figurenstellung auf dem Schachbrett. Zusätzlich wird registriert welcher der beiden Player als nächstes seinen nächsten Zug tätigen muss, welche Möglichkeiten zum rochieren bestehen, ob ein Schlag „en passant“ möglich ist und wenn ja auf welches Feld gezogen werden muss und wie viele Halbzüge gespielt wurden. Der Wert der Halbzüge wird zurückgesetzt sobald eine Bauernfigur gezogen oder eine beliebige Figur geschmissen wurde. Zusätzlich kann über ein Match ermittelt werden ob ein Spieler im Schach steht oder ob das Spiel schon bis zum Schachmatt gespielt wurde. All diese Informationen werden zusätzlich noch als String in der Forsyth-Edwards-Notation (FEN)[1] gespeichert.

4.1.3 Ressource: Draw (Zug)

Die Ressource Draw speichert zusätzlich zur ID die Farbe des Spielers, die Art der Spielfigur, Start- und Endfeld des Zuges, ob eine Figur geschlagen wurde, wenn ja ob durch en passant und ob seitens der Dame oder des König rochiert wurde. Die Informationen werden ähnlich zum Match als String gespeichert, aber in diesem Fall in der Standard Algebraic Notation (SAN)[1].

4.2 Ressourcenzugriffe mithilfe von Controllern

Die einzelnen Zugriffe auf die Ressourcen werden in den Kapiteln 4.2.1 bis **??** nach ihrer Art bzw. deren Aufgaben in einzelne Controller unterteilt, um eine gute Übersicht zu wahren. Für alle Einstiegspunkte der Representational State Transfer (REST)-API soll die Request-Methode „OPTIONS“ bereitstehen, über welchen ermittelt werden kann welche Methoden für den jeweiligen Einstiegspunkt zur Verfügung stehen.

Etwaige Requestparameter sollen in dem Format JavaScript Object Notation (JSON) oder x-www-form-urlencode mitgeschickt werden können. Die gesendeten Anfragen sollen ihr Feedback je nach Wunsch, via Content Negotiation, entweder in JSON oder in Extensible Markup Language (XML) zurücksenden. Dabei sollen drei Strategien bereitgestellt werden, entweder mittels Suffix, einem URL-Parameter oder dem Hypertext Transfer Protocol (HTTP) Accept-Header.

1 siehe Kapitel 2.3

4.2.1 Player Controller

Der Player Controller soll zwei Einstiegspunkt an den Uniform Ressource Indetifiers (URIs) `/player/` und `/player/{id}` zur Verfügung stellen. Der Parameter `{id}` dient dabei als Platzhalter für die ID eines Players, welche wiederum als Zahl dargestellt wird.

Am ersten Einstiegspunkt soll eine Liste aller Spieler über einen GET-Request bereitgestellt werden können. Des weiteren soll an diesem die Möglichkeit bestehen einen neuen Player mithilfe eines POST-Requests zu erzeugen. Dabei muss als Parameter der Name und das Passwort des Players mitgegeben werden. Die SQLite-Datenbank soll die ID dabei automatisch mittels Autoincrement erzeugen. Bei erfolgreicher Erstellung des Players soll dieser zurückgegeben werden, ansonsten `NULL`.

Am zweiten Einstiegspunkt soll ein einzelner existierender Player über einen GET-Request bereitgestellt, über einen DELETE-Request gelöscht und über einen PATCH-Request aktualisiert werden können. Nur das Passwort darf dabei laut Anforderungen[1] aktualisiert werden, wobei dieses zusätzlich als Parameter an den Request mit angehangen werden muss.

4.2.2 Match Controller

4.2.3 Draw Controller

4.2.4 Game Controller

Der Game Controller soll dazu dienen große Match bezogene Daten separat zu ermitteln. Dabei soll mittels GET-Request an der URI `/match/id/draw` eine Liste aller Draws und über die URI `/match/id/pieceSets` eine Map mit allen Figuren der beiden Spielteilnehmer breitgestellt werden. Neben den aktuell auf dem Spielfeld stehenden Figuren sollen dabei auch die geschmissen mitgeschickt werden.

1 siehe 4.1

Literatur

[ANO] ANONYM: „Why AMD?“ (), Bd. URL: http://requirejs.org/docs/whyamd.html (besucht am 13. 04. 2018) (siehe S. 37).

[Cos17] COSMINA, JULIANA, ROB HARROP, CHRIS SCHAEFER und CLARENCE HO: *Pro Spring 5. An In-Depth Guide to the Spring Framework and Its Tools*. English. 5th. Apress, 11. Nov. 2017.

[Fie00] FIELDING, ROY THOMAS: „Architectural Styles and the Design of Network-based Software Architectures“. phd. University of California, Irvine, 2000. URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm (besucht am 07. 05. 2018).

[Fie08] FIELDING, ROY THOMAS: „REST APIs must be hypertext-driven“. (20. Okt. 2008), Bd. URL: http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven (besucht am 14. 05. 2018).

[Fie] FIELDING, ROY THOMAS u. a.: *Hypertext Transfer Protocol – HTTP/1.1*. URL: https://tools.ietf.org/html/rfc2616 (besucht am 12. 05. 2018).

[Har] HARIRI, HADI, EDOARDO VACCHI und SÉBASTIEN DELEUZE: „Creating a RESTful Web Service with Spring Boot“. (), Bd. URL: https://kotlinlang.org/docs/tutorials/spring-boot-restful.html (besucht am 04. 04. 2018).

[Hip] HIPP, WYRICK & COMPANY INC.: „About SQLite“. (), Bd. URL: https://www.sqlite.org/about.html (besucht am 09. 04. 2018).

[Inc] INC., PIVOTAL SOFTWARE: „Building a RESTful Web Service“. (), Bd. URL: https://spring.io/guides/gs/rest-service/ (besucht am 04. 04. 2018).

[Kre15] KRETZSCHMAR, CHRISTOPH: „Demonstration eines RESTful Webservices am Beispiel eines Schachservers“. Bachelor. Hochschule für Technik und Wirtschaft Dresden, 2015.

[Lim] LIMITED, TUTORIALS POINT INDIA PRIVATE: „SQLite - Java“. (), Bd. URL: https://www.tutorialspoint.com/sqlite/sqlite_java.htm (besucht am 09. 04. 2018).

[Los08] LOSSA, GÜNTER: *Schach lernen. Ein Leitfaden für Anfänger des königlichen Spiels; Der entscheidene Zug zum zwingenden Mattangriff*. German. Joachim Beyer Verlag, 2008.

[Mas] MASHKOV, SERGEY: „DOM trees“. (), Bd. URL: https://github.com/Kotlin/kotlinx.html/wiki/DOM-trees (besucht am 13. 04. 2018) (siehe S. 37).

[Spi16] SPICHALE, KAI: *API-Design. Praxishandbuch für Java- und Webservice-Entwickler*. German. 1st. dpunkt.verlag GmbH, Dez. 2016.

[Var15] VARANASI, BALAJI u. a.: *Introducing Gradle*. English. 1st. Apress, 23. Dez. 2015.

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

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Quelle: [ANO]
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 16. Mai 2018 Felix Dimmel

A First chapter of appendix

A.1 Parameters