
Inhaltsverzeichnis

1	Einleitung	1
2	Grundlagen	3
2.1	REST-API	3
2.1.1	Allgemeine Definition einer Application Programming Interface (API) . . .	3
2.1.2	Vorteile einer API	4
	Stabilität durch lose Kopplung	4
	Portabilität	4
	Komplexitätsreduktion durch Modularisierung	4
	Softwarewiederverwendung und Integration	4
2.1.3	Nachteile einer API	4
	Interoperabilität	4
	Änderbarkeit	5
2.1.4	Qualitätsmerkmale	5
	Benutzbarkeit	5
	Effizienz	5
	Zuverlässigkeit	5
2.1.5	Grundprinzipien von REST	7
	Eindeutige Identifikation von Ressourcen	7
	Verwendung von Hypermedia	7
	Verwendung von HTTP-Standardmethoden	7
	Unterschiedliche Repräsentationen von Ressourcen	8
	Statuslose Kommunikation	8
2.1.6	HATEOAS	8
2.2	Das Build-Tool Gradle	9
2.2.1	Eigenschaften von Gradle	9
	Declarative Dependency Management	9
	Declarative Builds	10
	Build by Convention	10
	Incremental Builds	10

Gradle Wrapper	10
Plugins	10
2.2.2 Verwaltung von Projekten und Tasks	10
Projekte	11
Tasks	11
2.3 Schachnotationen FEN und SAN	12
2.4 Schachregeln	13
3 Vergleich zwischen Kotlin und dem Google Web Toolkit	15
4 Konzept des Servers	17
4.1 Anforderungen	17
4.1.1 Ressource: Player (Spieler)	19
4.1.2 Ressource: Match (Partie)	19
4.1.3 Ressource: Draw (Zug)	19
4.2 Ressourcenzugriffe mithilfe von Controllern	19
4.2.1 Player Controller	20
4.2.2 Match Controller	21
4.2.3 Draw Controller	21
5 Konzeption des Clients	23
5.1 Anforderungen	23
5.2 Verwendete Bibliotheken/Frameworks	24
5.2.1 RequireJS	24
5.2.2 kotlinx.html	24
5.2.3 kotlinx.serialization	25
5.2.4 kotlinx.coroutines	25
6 Implementation des Servers	27
6.1 Verwendete Bibliotheken/Frameworks	27
6.1.1 Spring	27
6.1.2 SQLite	29
6.1.3 ORMLite	29
7 Implementation des Clients	33
8 Fazit	35
9 Ausblick	37

Literatur	39
Anhang	51
A First chapter of appendix	51
A.1 Parameters	51

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.

Zur Unterstützung der Erläuterungen in den Kapiteln [4.1.1](#) bis [4.1.3](#) bietet die [Abbildung 4.1](#), in Form eines Unified Modeling Language (UML) Klassendiagramms, eine Veranschaulichung.

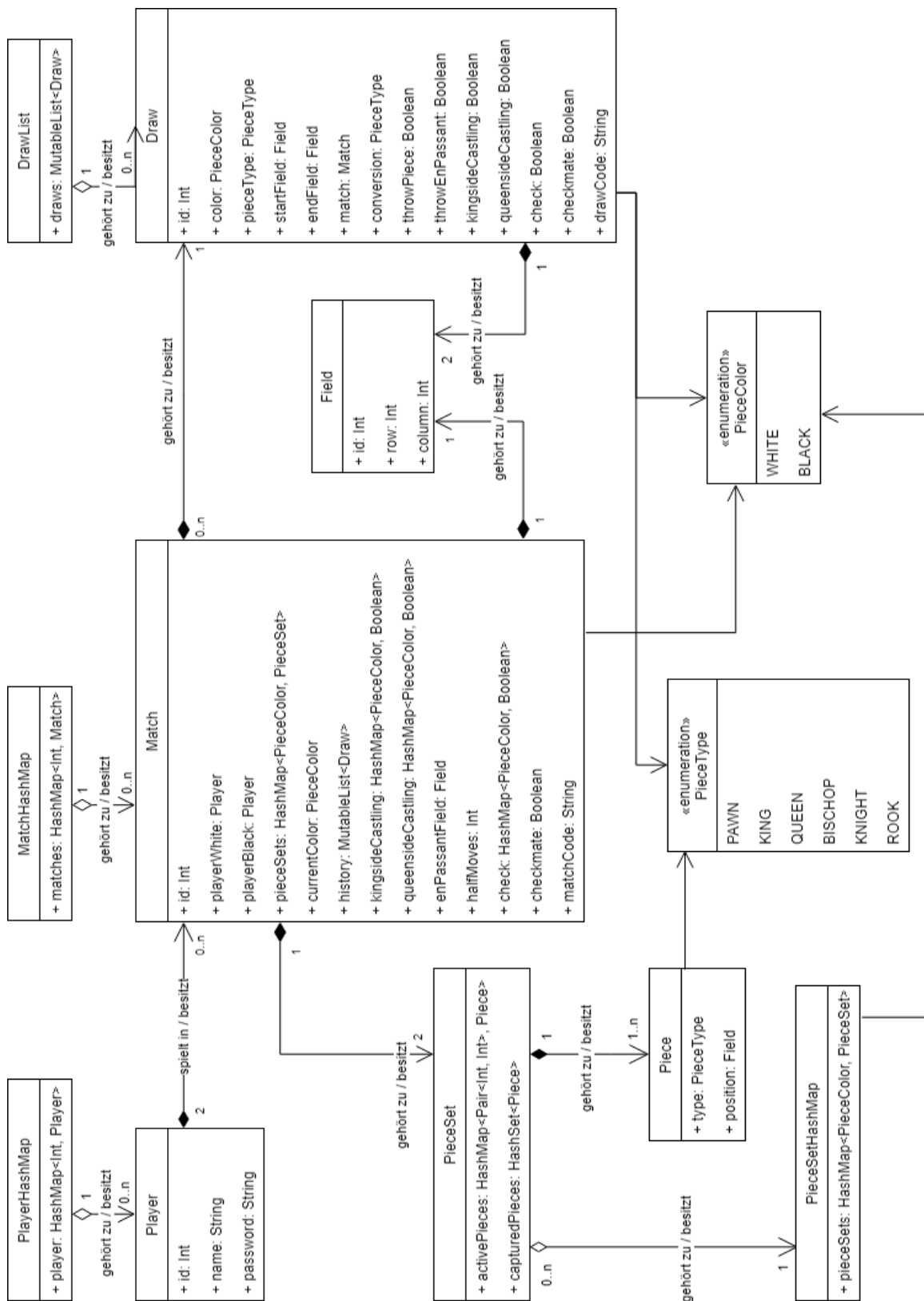


Abbildung 4.1: Klassendiagramm: Modelle des Servers

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 4.2.3 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-urlencoded mitgeschickt werden können. Die gesendeten Anfragen sollen ihr Feedback

¹ siehe Kapitel 2.3

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.

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 Auto-increment 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 Anforderungen¹ aktualisiert werden, wobei dieses zusätzlich als Parameter an den Request mit angehängen werden muss.

Für ein besseres Verständnis bietet die [Abbildung 4.2](#) eine visuelle Verdeutlichung der Einstiegspunkte des Player Controllers.

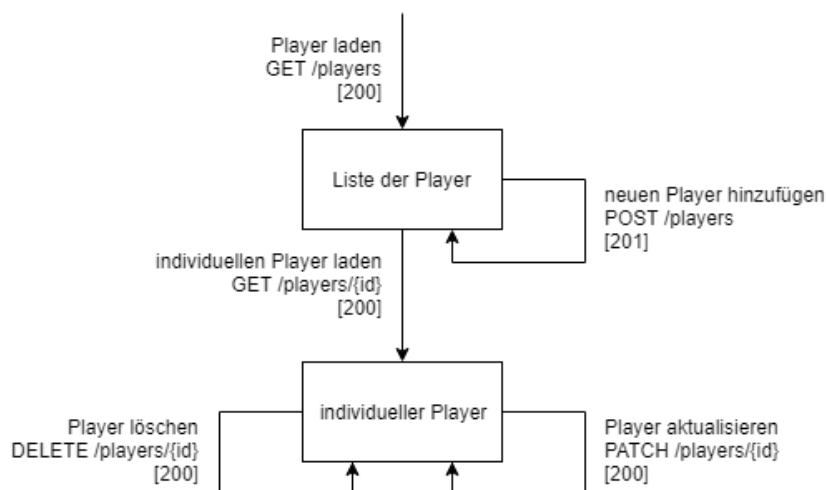


Abbildung 4.2: Player Controller - Übersicht der Einstiegspunkte

¹ siehe [Kapitel 4.1](#)

4.2.2 Match Controller

Die URIs `/match/`, `/match/{id}`, `/match/id/draws` und `/match/id/pieceSets` sollen durch den Match Controller bereitgestellt werden.

Dabei soll die erste URI ebenso wie beim „Player Controller“ Zurverfügungstellung einer Liste aller registrierten Matches und dem anlegen neuer dienen. Das Bereitstellen der Liste soll mittels GET- und das anlegen mittels POST-Request erfolgen. Der GET-Request soll zwei optionale boolesche Parameter bereitstellen, womit das Senden der Historie von Draws bzw. der Figurenstellung bestimmt werden soll. Standardmäßig sollen dabei die Parameter `TRUE` sein. Weiter unten in diesem Kapitel wird beschrieben wie diese Informationen separat geholt werden können. Um ein neues Match zu registrieren, müssen dabei die ID's der beiden Spielteilnehmer mitgeschickt werden. Anhand des Parameternamens soll festgelegt werden welcher Spieler Weiß und welcher Schwarz spielen soll.

Der zweite Einstiegspunkt soll in diesem Controller ausschließlich dazu verwendet werden, um einzelne Matches mithilfe eines GET-Request anzufordern oder mithilfe eines DELETE-Request zu löschen. Für das anfordern eines einzelnen Matches stehen ebenso zwei boolesche, wie beim Anfordern einer Liste, bereit. Wenn ein Nutzer ein Match löscht, sollen ebenfalls alle zugehörigen Draws gelöscht werden. Um eine unrechtmäßige Manipulation der Match-Daten durch einen Nutzer zu verhindern, soll keine Möglichkeit bereitstehen ein Match zu aktualisieren. Die Aktualisierung eines Matches soll ausschließlich durch das hinzufügen von Draws erfolgen.¹

Die letzten beiden Einstiegspunkte sollen 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 bereitgestellt werden. Neben den aktuell auf dem Spielfeld stehenden Figuren sollen dabei auch die geschmissen mitgeschickt werden.

Die [Abbildung 4.3](#) bietet für die zuvor definierten Einstiegspunkte eine grafische Veranschaulichung.

4.2.3 Draw Controller

Auch der Draw Controller stellt wieder zwei Einstiegspunkte an den URIs `/draw/` und `/draw/{id}` bereit.

¹ siehe [Kapitel 4.2.3](#)

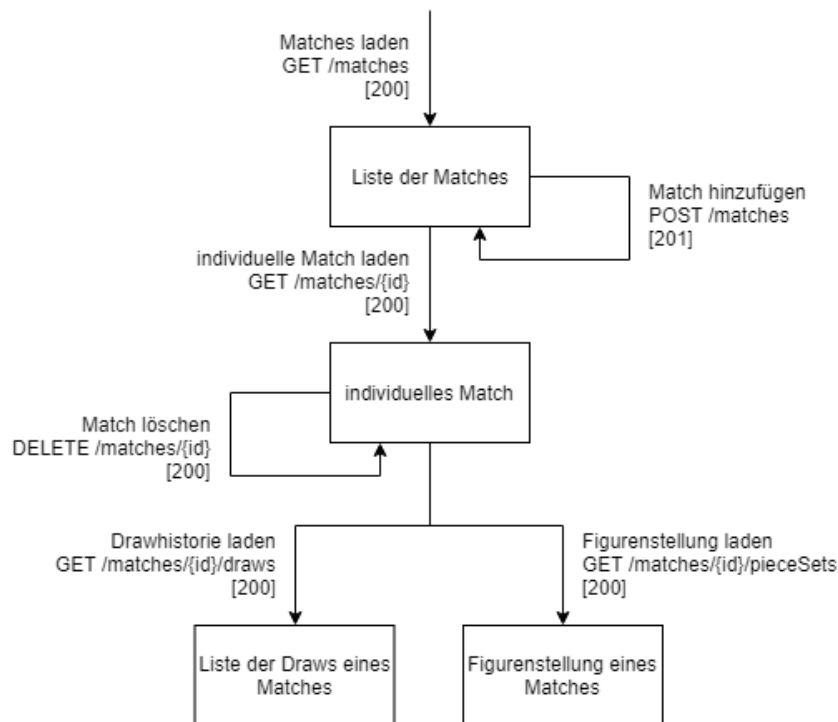


Abbildung 4.3: Match Controller - Übersicht der Einstiegspunkte

Am ersten Einstiegspunkt soll wieder das bereitstellen einer Liste mittels GET- und das hinzufügen mittels POST-Request von Draws zur Verfügung gestellt werden. Für das hinzufügen eines neuen Draws ist die ID des Matches und der Draw-Code in der SAN als Parameter vonnöten. Zusätzlich soll die Möglichkeit bestehen die Zeilen- und die Spaltennummer der Startposition mitzugeben. Wenn diese Informationen nicht mitgegeben werden, so soll der Controller die Startposition kalkulieren. Spalten sollen dabei als Nummern und nicht als Buchstaben mitgegeben werden¹. Nach erfolgreicher Validierung des Draw-Codes soll der Draw dem zugehörigen Match hinzugefügt und die Match-Daten aktualisiert werden.

Über den zweiten Einstiegspunkt soll wieder die Abfrage nach einem einzelnen Draw möglich sein. Eine Möglichkeit zum Löschen oder Aktualisieren des Draws soll nicht bestehen, da sonst der Nutzer wieder eine Möglichkeit hätte das Match zu manipulieren. Das Löschen von Draws soll über die Löschung des dazugehörigen Matches erfolgen².

Wie in den vorherigen Kapiteln bietet die [Abbildung 4.4](#) eine Veranschaulichung.

¹ $A \rightarrow 8; \dots; H \rightarrow 8$

² siehe [Kapitel 4.2.2](#)

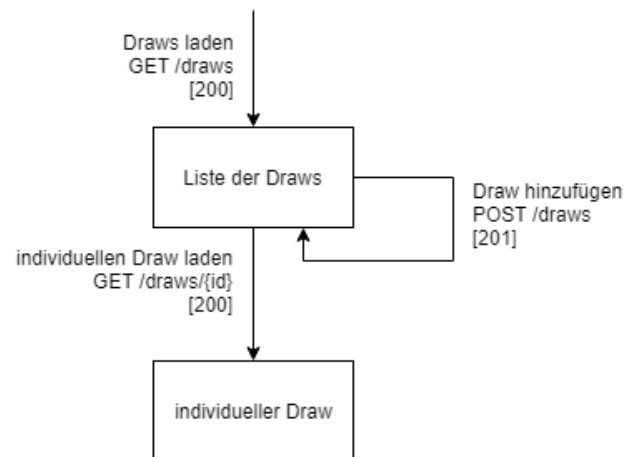


Abbildung 4.4: Draw Controller - Übersicht der Einstiegspunkte

Literatur

- [ANOa] ANONYM: „Require.js. A JavaScript Module Loader“. (), Bd. URL: <http://requirejs.org/> (besucht am 13.04.2018).
- [ANOb] ANONYM: „Why AMD?“ (), Bd. URL: <http://requirejs.org/docs/whyamd.html> (besucht am 13.04.2018) (siehe S. 45).
- [ARD17a] ARD/ZDF-MEDIENKOMMISSION: „ARD/ZDF-Onlinestudie 2017: Neun von zehn Deutschen sind online. Bewegtbild insgesamt stagniert, während Streamingdienste zunehmen - im Vergleich zu klassischem Fernsehen jedoch eine geringe Rolle spielen.“ *Media Perspektiven* (Sep. 2017), Bd. URL: <http://www.ard-zdf-onlinestudie.de/ardzdf-onlinestudie-2017/> (besucht am 22.03.2018).
- [ARD17b] ARD/ZDF-MEDIENKOMMISSION: „Kern-Ergebnisse der ARD/ZDF-Onlinestudie 2017“. *Media Perspektiven* (Sep. 2017), Bd. URL: http://www.ard-zdf-onlinestudie.de/files/2017/Artikel/Kern-Ergebnisse_ARDZDF-Onlinestudie_2017.pdf (besucht am 26.03.2018).
- [Bra17] BRAUN, HERBERT: „Modul.js. Formate und Werkzeuge für JavaScript-Module“. *c't Heft 3/2017* (2017), Bd.: S. 128–133.
- [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] HIPPI, 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).
- [Inc17] INC., STACK EXCHANGE: „Developer Survey 2017“. (2017), Bd. URL: <https://insights.stackoverflow.com/survey/2017#technology> (besucht am 26.03.2018).
- [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. 45).
- [Sha17] SHAFIROV, MAXIM: „Kotlin on Android. Now official“. (Mai 2017), Bd. URL: <https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official/> (besucht am 26.03.2018).
- [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) (siehe S. 45).

Abbildungsverzeichnis

2.1 Startposition eines Schachspiels in der FEN	12
2.2 Beispiel SAN: Bauer zieht von a2 nach a4	12
2.3 Beispiel SAN: Spring zieht von b1 nach c3	12
4.1 Klassendiagramm: Modelle des Servers	18
4.2 Player Controller - Übersicht der Einstiegspunkte	20
4.3 Match Controller - Übersicht der Einstiegspunkte	22
4.4 Draw Controller - Übersicht der Einstiegspunkte	23

Tabellenverzeichnis

2.1	Eigenschaften/Ziele des Qualitätsmerkmals „Benutzbarkeit“ (verändert nach [Spi16, S. 14–23])	6
2.2	Figurenbedeutung in der FEN und SAN (Quelle: [Kre15, Tabelle 2.1])	13

Listings

2.1 Beispiel: Gradle-Task	12
5.1 Beispiel: Moduldefinition mittels Asynchronous Module Definition (AMD) ¹ . . .	24
5.2 Beispiel: Verwendung der Bibliothek kontlinx.html ²	24
5.3 Beispiel: Verwendung der Bibliothek kotlinx.html (Ergebnis)	25
6.1 Einbindung des Spring Framework mithilfe von Gradle	27
6.2 Beispiel: Spring Controller	28
6.3 Beispiel: Spring Application Class	28
6.4 Einbindung der Bibliothek SQLite mithilfe von Gradle	29
6.5 Einbindung der Bibliothek ORMLite mithilfe von Gradle	30
6.6 Beispiel: Persistierung einer Klasse mittels ORMLite ³	30
6.7 Beispiel: Verwendung von ORMLite ⁴	31

1 Quelle: [[ANOb](#)]

2 Quelle: [[Mas](#)]

3 Quelle: [[Wat](#)]

4 verändert nach [[Wat](#)]

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 17. Mai 2018

Felix Dimmel

A First chapter of appendix

A.1 Parameters

