Hochschule Aalen

Projektbericht
Studiengang : Informatik

Visualisierung von MongoDB Datenbanken

von

Max Winter

80559

Betreuender Professor: Prof. Dr. Gregor Grambow

Einreichungsdatum : 01. Dezember 2016

Eidesstattliche Erklärung

Hiermit erkläre ich, **Max Winter**, dass ich die vorliegenden Angaben in dieser Arbeit wahrheitsgetreu und selbständig verfasst habe.

Weiterhin versichere ich, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben, dass alle Ausführungen, die anderen Schriften wörtlich oder sinngemäß entnommen wurden, kenntlich gemacht sind und dass die Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.

Ort, Datum Unterschrift (Student)

Kurzfassung

Ziel der Kurzfassung ist es, einen (eiligen) Leser zu informieren, so dass dieser entscheiden kann, ob der Bericht für ihn hilfreich ist oder nicht (neudeutsch: Management Summary). Die Kurzfassung gibt daher eine kurze Darstellung

- des in der Arbeit angegangenen Problems
- der verwendeten Methode(n)
- des in der Arbeit erzielten Fortschritts.

Dabei sollte nicht auf die Struktur der Arbeit eingegangen werden, also Kapitel 2 etc. denn die Kurzfassung soll ja gerade das Wichtigste der Arbeit vermitteln, ohne dass diese gelesen werden muss. Eine Kapitelbezogene Darstellung sollte sich in Kapitel 1 unter Vorgehen befinden.

Länge: Maximal 1 Seite.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Abkürzungsverzeichnis

1. Einleitung

1.1. Vorangegangene Arbeit

1.2. Motivation

MongoDB hat sich in den letzten Jahren zu einem der wichtigsten Datenbanksysteme entwickelt, da aufgrund immer größerer werdenden Datenmengen die Vorteile von NoSQL-Datenbanken für immer mehr Anwendungen überwiegen. [7] Da MongoDB keine relationale Datenbank ist, können herkömmliche Visualisierungs- und Analysetools, die für SQL entwickelt wurden, nicht verwendet werden. Für MongoDB gibt es zwar ein paar visualisierungstools wie Beispielsweise MongoDB Charts und MongoDB Compass, die meisten davon visualisieren aber die Daten in der Datenbank, und nicht die Struktur und das Schema der Dokumente in der Datenbank. [1] Ziel dieses Projekts ist es deshalb, ein Visualisierungstool für MongoDB Datenbanken zu entwickeln, welches die Dokumente der Datenbanken analysiert und auswertet. Diese Schemas werden dann in Form von verschachtelten Tabellen visualisiert. Das erleichtert es den Entwicklern der Datenbanken, die Strukturen und Abhängigkeiten in ihren Datenbanken zu verstehen und zu verbessern.

1.3. Problemstellung und -abgrenzung

Um Datenbanken analysieren zu können, muss es möglich sein, sich mit diesen zu verbinden. Dies erfordert einerseits eine Nutzeroberfläche, über die der Nutzer die Verbindungsdaten eingeben kann. Andererseits erfordert dies die Verbindung mit der Datenbank selbst über eine geeignete MongoDB Schnittstelle. Die Dokumente der Datenbank müssen anschließend in möglichst kurzer Zeit analysiert werden, um aus ihnen Schemas zu extrahieren. Diese Schemas müssen dann in einem Frontend übersichtlich und visuell ansprechend angezeigt werden. Dafür werden hauptsächlich 2 Ansichten benötigt: Einerseits wird eine Übersicht über alle Collections benötigt, in welcher für jede Collection das meistverwendete Schema angezeigt werden soll. Andererseits wird für jede Collection eine Detailansicht benötigt, welche alle Schema-Variationen in einer Collection sowie weitere Details und Daten anzeigt.

Die Problemstellung dient dazu, das zu lösende Problem klar zu definieren und abzugrenzen. Der Praktikant soll ein klares Verständnis des zu lösenden Problems haben. Insbesondere soll auch verhindert werden, dass zu viele Probleme gleichzeitig angegangen werden. Eine Negativabgrenzung verhindert, dass beim Leser später nicht erfüllte Erwartungen geweckt werden.

1.4. Ziel der Arbeit

Mit dem Ziel der Arbeit wird der angestrebte Lösungsumfang festgelegt. An diesem Ziel wird entschieden, ob das Praktikum erfolgreich absolviert wurde oder nicht.

1.5. Vorgehen

Nachdem mit Problemstellung und Ziel gewissermaßen Anfangs- und Endpunkt des Praktikums beschrieben sind, wird hier der zur Erreichung des Ziels eingeschlagene Weg vorgestellt. Dazu werden typischerweise die folgenden Kapitel und ihr Beitrag zur Erreichung des Ziels der Arbeit kurz beschrieben. Die folgenden Kapitel sind ein – möglicher – Aufbau, Abweichungen können durchaus notwendig sein. Zur Darstellung des Vorgehens ist eine grafische Darstellung sinnvoll, bei der die einzelnen Lösungsschritte und ihr Zusammenhang dargestellt werden. Ein Beispiel hierfür findet sich in Abbildung **??**.

2. Grundlagen

2.1. JSON

JavaScript Object Notation (JSON) ist ein Format zum Datenaustausch. JSON basiert auf der Syntax von JavaScript Objekten, ist jedoch unabhängig von der Programmiersprache einsetzbar. Vorteile von JSON sind die einfache Lesbarkeit für Menschen und das einfache parsen und generieren für Maschinen. [6]

Object ist ein Set aus Key/Value Paaren.

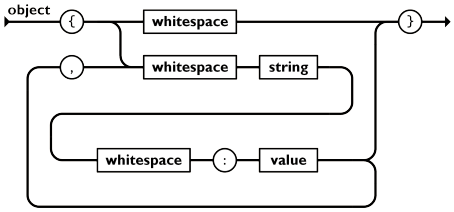

Abbildung 2.1.: JSON Object

Array ist eine geordnete Liste von Values.

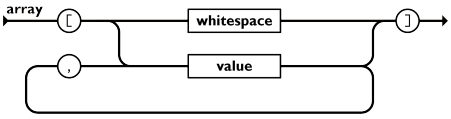

Abbildung 2.2.: JSON Array

Value kann ein String, eine Zahl, ein Boolean, ein Objekt, ein Array oder null sein. Dabei können beliebig viele Values ineinander verschachtelt sein.

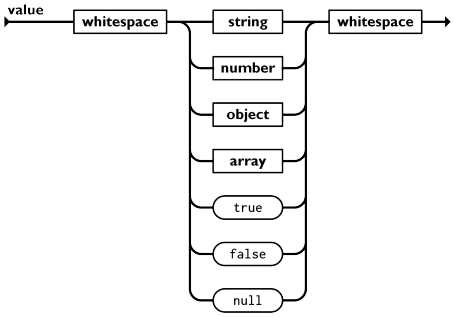

Abbildung 2.3.: JSON Value

String ist eine Sequenz aus Unicode Buchstaben.

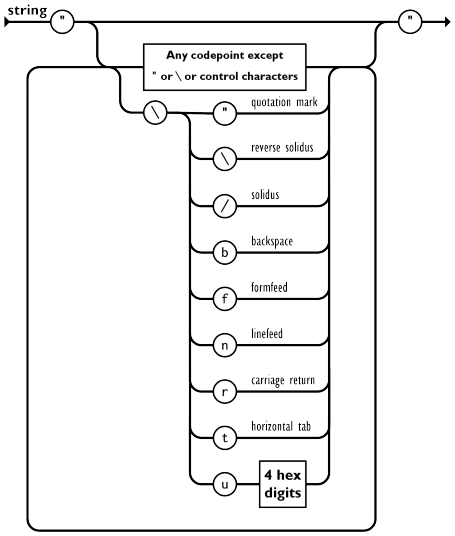

Abbildung 2.4.: JSON String

Number ist eine Zahl. Number kann sowohl eine Gleitkommazahl als auch eine Ganzzahl sein, und kann positiv sowie negativ sein.

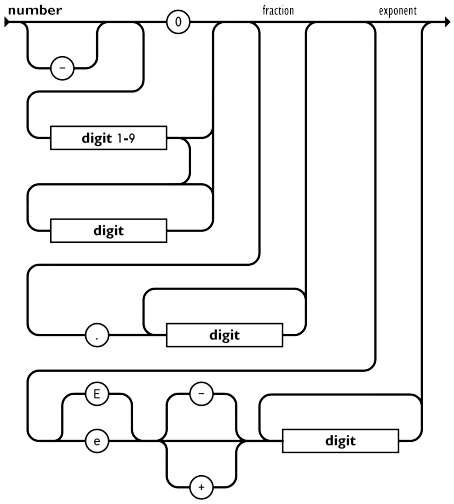

Abbildung 2.5.: JSON Number

2.2. HTTP

Hypertext Transfer Protocol (HTTP) ist ein Protokoll, das benutzt wird, um über das Internet zu kommunizieren. Hauptsächlich wird HTTP für die Kommunikation zwischen einem Webbrowser und einem Webserver benutzt. In HTTP wird mittels Nachrichten kommuniziert. Es wird erst ein Request vom Client abgesetzt, der daraufhin von dem Server mit einer Response beantwortet wird. Diese Nachrichten bestehen aus einem Header und einem Body. Der Body enthält die Nachricht selbst. Der Header enthält Meta-Daten über einen Request, wie beispielsweise die angefragten Ressourcen und Datentypen des Body.

2.3. SQL

Die Structured Query Language (SQL) ist eine Datenbanksprache für relationale Datenbanken. In SQL werden die Daten in Tabellen organisiert, die ein festes Schema definieren. Man kann SQL in 2 Teile aufteilen: Die Data Definition Language (DDL) definiert den Aufbau des Schemas. Mit ihr können Datenbankobjekte erzeugt und gelöscht werden. Zur DDL gehören unter anderem die Befehle CREATE, ALTER, DROP und TRUNCATE. Mit der Data Manipulation Language (DML) können Daten manipuliert, also eingefügt, geändert, gelesen und gelöscht werden. Die Befehle hierfür lauten SELECT, INSERT, UPDATE, DELETE, MERGE und noch weitere. [**schicker:datenbanken**]

2.3.1. Normalformen

2.4. MongoDB

MongoDB ist eine Dokument-Orientierte Datenbank. In Dokument-Orientierten Datenbanken wird das aus SQL bekannte Konzept von Reihen durch Dokumente ersetzt. Dokumente haben im Gegensatz zu Reihen in Tabellen kein fixes Schema, dass sie erfüllen müssen, sondern sind sehr flexibel. Grundsätzlich bestehen Dokumente aus Key - Value Paaren, welche in einer JSON-Ähnlichen Struktur gespeichert werden. Dokumente können, wie auch in JSON, ineinander verschachtelt sein, was Hierarchische Strukturen und dadurch Denormalisierung ermöglicht. Die Dokumente werden in Collections Organisiert. Eine Collection entspricht in SQL einer Tabelle ohne das fixe Schema. Diese Collections befinden sich wiederum in Datenbanken. Eine MongoDB Instanz kann mehrere voneinander unabhängige Datenbanken beinhalten. Man kann sich mit der MongoShell mit einer MongoDB Instanz verbinden um mit der Mongo Query Language oder mit JavaScript die

Instanz administrieren und Daten manipulieren. [3]

2.5. REST API

Ein Application Programming Interface (API) ist eine Schnittstelle, über die von außen mit einem Programm interagiert werden kann. Das Grundprinzip von Representational State Transfer (REST) ist ein Backend, welches Resourcen beinhaltet. Ein Client kann über die gängigen HTTP Operationen mit diesen Resourcen interagieren. Jede Resource hat eine global eindeutige ID und wird meist durch JSON oder Extensible Markup Language (XML) repräsentiert. In Java wird ein RESTful Webservice für gewöhnlich mit **jaxrs!** (**jaxrs!**) umgesetzt. [8]

2.6. Python

Python ist eine Objektorientierte High-Level Programmiersprache, die besonderen Wert auf Lesbarkeit legt. Variablen in Python werden dynamisch getyped. Das bedeutet, dass eine Variable in Python keinen festen Typ hat, sondern dieser dynamisch über den ihr zugewiesenen Wert bestimmt wird. Python ist keine Compilierte, sondern eine interpretierte Programmiersprache. Diese Eigenschaften machen Python zu der idealen Sprache für das Schreiben von Skripten, sowie für Anwendungen, die schnell und simpel entwickelt werden sollen. PIP ist ein in Python integrierter Paketmanager, der das installieren von Packages erleichtert. Der Python Interpreter ist in C geschrieben, weshalb man die Funktionalität des Interpreters durch C-Programme erweitern kann, um beispielsweise laufzeitkritische Funktionen in C auszuführen. [9]

2.7. Flask

Flask ist ein minimalistisches Open-Source Web Framework. Flask kommt mit ein paar Kernpaketen, welche für Web Apps benötigt werden. Alles weitere muss der Nutzer selbst über PIP installieren: Durch diesen minimalistischen Ansatz kann man Flask sehr flexibel einsetzen. Flask ist beispielsweise flexibel mit SQL oder NoSQL Datenbanken, aber auch ohne Datenbanken einsetzbar. [5]

2.8. JavaScript

JavaScript ist eine High-Level Programmiersprache, die just-in-time kompiliert wird. JavaScript ist besonders bekannt als Skriptsprache für Webseiten. Die Pakete in JavaScript werden mittels dem Paketmanager npm verwaltet.

2.9. React

React ist eine JavaScript Bibliothek zum bauen Von Benutzeroberflächen. React ist Komponentenbasiert. Das bedeutet, das React aus einzelnen Komponenten bestehen, die ihren eigenen State haben, und die zusammengesetzt ein komplexes User Interface (UI) bilden. [2]

2.9.1. Redux Store

Redux ist ein State Container für JacaScript Apps, der es ermöglicht, States nicht mehr in den Komponenten, sondern in einem Zentralen Container zu speichern. Dadurch können beispielsweise States wiederhergestellt werden, nachdem eine Komponente geschlossen und wieder geöffnet wurde. Außerdem erleichtert Redux es, State Änderungen in Komponenten nachzuvollziehen. [4]

2.9.2. Axios

Axios ist ein HTTP-Client für JavaScript. Mithilfe von Axios Können HTTP-Requests versendet und die Responses von diesen verarbeitet werden.

3. Problemanalyse

Die Analyse des zu lösenden Problems ist Grundlage für jedes ingenieurmäßige Vorgehen. Daher soll in diesem Kapitel das zu lösenden Problem auf Basis des im Grundlagenkapitel aufbereiteten Wissens analysiert werden. Hierzu ist insbesondere notwendig zu klären, wie sich das Gesamtproblem in Teilprobleme zerlegen lässt und welche Abhängigkeiten zwischen diesen bestehen.

Bei Software-Projekten befindet sich an dieser Stelle typischerweise die Anforderungsanalyse des Rational Unified Process (RUP).

Anforderungen:

- modular
- erweiterbar

4. Lösungskonzept

Auf der Basis der im vorangegangenen Kapitel erstellten Problemanalyse und der im Grundlagenkapitel aufgearbeiteten theoretischen Kenntnisse wird ein Lösungskonzept erarbeitet.

Bei Software-Projekten entspricht dieses Kapitel typischerweise der Analyse & Design-Phase des RUP. Typische Ergebnisse dieser Phase sind Klassendiagramme etc.

4.1. Verwendete Technologien

- Wahl des Backend Frameworks und der Sprache
  - Probleme mit MongoDB Client in Java/Spring
  - Python und Flask sind lightweight
- Wahl des Frontend Frameworks und der Sprache
  - bestehendes Projekt mit diesem Framework
  - Verbreitung von React
  - Modularität dank React
  - Warum Web?

4.2. Bestehende Visualisierungstools

4.3. Analyse der MongoDB Datenbank

4.4. Planung des Frontends

5. Implementierung

In diesem Kapitel wird die konkrete Implementierung des im Kapitel 4 entwickelten Lösungskonzepts beschrieben. Hierbei wird auf die konkret verwendeten Entwicklungswerkzeuge etc. Bezug genommen.

Bei Software-Projekten besteht dieses Kapitel typischerweise aus den Phasen Implementierung & Test im RUP.

Zum Beispiel kann man hier auch ein kleines Listing einfügen.

```
1  #include<stdio.h>
2
3  int main() {
4      // Kommentar
5      int answer = 20 << 1;
6      answer += 2;
7      printf("Hallöchen Welt!\n");
8      printf("Die Antwort ist: %d\n", answer);
9      return 0;
10 }
```

Quelltext 5.1: Überschrift des Quelltexts

Manchmal hilft auch eine kleine Tabelle:

Messwert a	**Messwert b**
9	5
1	4
1	3

Tabelle 5.1.: Überschrift der Tabelle

Details siehe Tabelle 5.1.

5.1. Backend

5.1.1. Analyse der MongoDB Datenbank

5.1.2. REST API

5.2. Frontend

5.2.1.

6. Inbetriebnahme

Aufgabe des Kapitels Inbetriebnahme ist es, die Überführung der in Kapitel 5 entwickelte Lösung in das betriebliche Umfeld aufzuzeigen. Dabei wird beispielsweise die Inbetriebnahme eines Programms beschrieben oder die Integration eines erstellten Programmodules dargestellt.

Bei der Software-Erstellung entspricht dieses Kapitel der Auslieferungsphase (Deployment) im RUP.

7. Evaluierung

Aufgabe des Kapitels Evaluierung ist es, in wie weit die Ziele der Arbeit erreicht wurden. Es sollen also die erreichten Arbeitsergebnisse mit den Zielen verglichen werden. Ergebnis der Evaluierung kann auch sein, das bestimmte Ziele nicht erreicht werden konnten, wobei die Ursachen hierfür auch außerhalb des Verantwortungsbereichs des Praktikanten liegen können.

8. Zusammenfassung und Ausblick

8.1. Erreichte Ergebnisse

Die Zusammenfassung dient dazu, die wesentlichen Ergebnisse des Praktikums und vor allem die entwickelte Problemlösung und den erreichten Fortschritt darzustellen. (Sie haben Ihr Ziel erreicht und dies nachgewiesen).

8.2. Ausblick

Im Ausblick werden Ideen für die Weiterentwicklung der erstellten Lösung aufgezeigt. Der Ausblick sollte daher zeigen, dass die Ergebnisse der Arbeit nicht nur für die in der Arbeit identifizierten Problemstellungen verwendbar sind, sondern darüber hinaus erweitert sowie auf andere Probleme übertragen werden können.

8.2.1. Erweiterbarkeit der Ergebnisse

Hier kann man was über die Erweiterbarkeit der Ergebnisse sagen.

8.2.2. Übertragbarkeit der Ergebnisse

Und hier etwas über deren Übertragbarkeit.

Literatur

[1] Ralf Abueg. *Visualization Solutions For MongoDB*. 2020. URL: https://www.knowi.com/blog/visualization-solutions-for-mongodb/ (besucht am 30. 01. 2023).

[2] Alex Banks und Eve Porcello. *Learning React: Modern Patterns for Developing React Apps*. O'Reilly Media, 2020.

[3] Shannon Bradshaw, Eoin Brazil und Kristina Chodorow. *MongoDB: the definitive guide: powerful and scalable data storage*. O'Reilly Media, 2019.

[4] Soham De Roy. *What is Redux? Store, Actions, and Reducers Explained for Beginners*. 2022. URL: https://www.freecodecamp.org/news/what-is-redux-store-actions-reducers-explained/ (besucht am 20. 01. 2023).

[5] Miguel Grinberg. *Flask web development: developing web applications with python*. Ö'Reilly Media, Inc.", 2018.

[6] *Introducing JSON*. URL: https://www.json.org/json-en.html (besucht am 11. 08. 2022).

[7] *MongoDB Systemeigenschaften*. 2023. URL: https://db-engines.com/de/system/MongoDB (besucht am 30. 01. 2023).

[8] Marcus Schießer und Martin Schmollinger. *Workshop Java EE 7*. 2., aktualisierte und erweiterte Auflage. Heidelberg: dpunkt.verlag, 2015, S. 1–13. ISBN: 978-3-86490-195-9.

[9] Guido Van Rossum und Fred L Drake Jr. *Python tutorial*. Bd. 620. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 2020.

A. Anhang A

B. Anhang B