Fakultät Wirtschaft

Projektdokumentation

Modul: Neue Datenbankkonzepte

im Studiengang Wirtschaftsinformatik

Semester: 4

Prüfungsleistung zum Thema

„MongoDB / Bilder-Management-System“

Vorgelegt von:

Gruppe 5:

- Thomas Wegele
- Emily Springer
- Maximilian Schmutterer
- Janis Franzen

Betreuer: Marvin Scharle

Link zum Github-Repository: https://github.com/thethomss/Datenbankprojekt-WWI21

Branch: „main“

1 Inhalt

1 Inhalt I
1 Projektvorstellung 3
2 Anforderungen 3
3 Technische Umsetzung 3
4 Fazit 5

Abbildungsverzeichnis

Abbildung 1 - Aufbau Datenbank 3
Abbildung 2 – m:n-Beziehung Images und Tags 4
Abbildung 3 - Aufbau Image-File 4
Abbildung 4 - Aufbau Tag-File 5
Abbildung 5 - Einfache Funktionserweiterung 5

1 Projektvorstellung

Unser Team hat sich zum Ziel gesetzt, ein leistungsstarkes und effektives Bilder-Management-System zu entwickeln. Hierbei können Bilder in der Datenbank gespeichert werden und mit Tags und Beschreibungen versehen werden.

2 Anforderungen

Für unsere Anwendung benötigen wir eine Datenbank, die in der Lage ist, flexibel mit verschiedenen Arten von Daten, wie beispielsweise Bildern und Tags, umzugehen. MongoDB ermöglicht es uns, Daten flexibel und effizient zu speichern, ohne uns an ein starres Tabellenmodell halten zu müsse. Das Speichern von Dokumenten erfolgt in Collections – ohne ein striktes Schema. Um eine hohe Leistung bei der Verarbeitung von großen Datenmengen zu erreichen, ist es wichtig, dass das Bilder-Management-System schnell auf Daten zugreifen und sie verarbeiten kann. Auch hier punktet MongoDB durch seine optimierte Verarbeitung großer Datenmengen. Zudem benötigt die Anwendung eine einfach zu bedienende Abfragesprache, um effektiv mit der Datenbank zu interagieren. MongoDB bietet eine Abfragesprache an, die sehr ähnlich zu JavaScript ist. Das erleichtert Entwicklern das schnelle Schreiben und Verstehen von Abfragen.

3 Technische Umsetzung

Diese API verwendet das Express-Framework für Node.js und die Mongoose-Bibliothek für die Interaktion mit MongoDB. Letztere wird mittels Docker-Container zur Verfügung gestellt, um die Datenbank in einer isolierten und konsistenten Umgebung auszuführen.

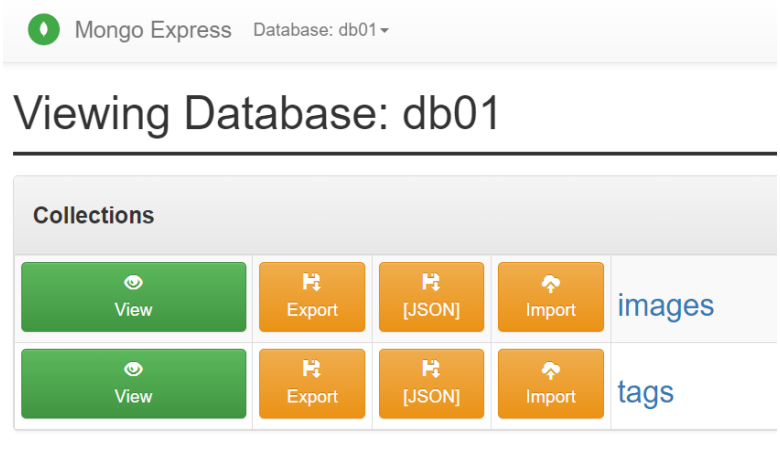

Abbildung 1 - Aufbau Datenbank

Für das Hochladen von Bildern wird „Multer“ verwendet, das eine Middleware zum Verarbeiten von Datei-Uploads bereitstellt. Die hochgeladenen Bilder werden auf dem Server im Verzeichnis "upload/images" gespeichert.

Die MongoDB-Datenbank enthält zwei Sammlungen: "images" und "tags". Die "images"-Sammlung enthält Dokumente, die Informationen zu Bildern wie den Dateinamen, die Beschreibung, die Tags und das Erstellungsdatum enthalten. Die "tags"-Sammlung enthält Dokumente, die nur den Namen des Tags enthalten.

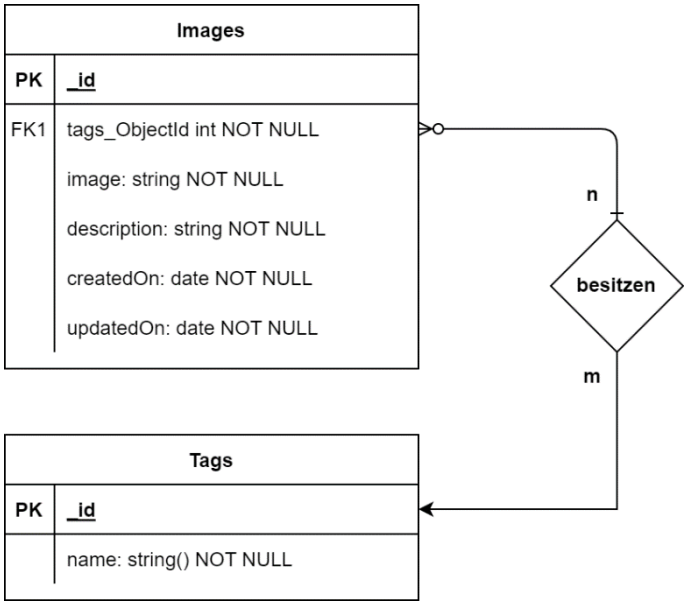

Abbildung 2 – m:n-Beziehung Images und Tags

In der "images"-Sammlung wird das Feld "tags" verwendet, um auf die "tags"-Sammlung zu verweisen, indem die IDs der zugehörigen Tag-Dokumente als Array von ObjectIds gespeichert werden.

```
{
    _id: ObjectId('63ffa35e86ca032029a74543'),
    image: 'http://localhost:3000/profile/file-image_1678265730394.png',
    tags: [
        ObjectId('6407ce585500758ac0cdcb9c'),
        ObjectId('6407ce585500758ac0cdcb9e')
    ],
    description: 'Update Test',
    createdOn: 1677697886849,
    updatedOn: ISODate('2023-03-08T08:55:30.469Z')
}
```

Abbildung 3 - Aufbau Image-File

Um die Beziehungen zwischen Bildern und Tags zu verwalten, verwendet die API eine Kombination aus der "tags"-Sammlung und der "populate()" -Methode von Mongoose, um die Tag-Informationen beim Abrufen von Bildern abzurufen.

Die API ermöglicht auch die Erstellung von neuen Tags, wenn beim Hochladen eines Bildes neue Tags angegeben werden, die noch nicht in der "tags"-Sammlung vorhanden sind.

```
{
    _id: ObjectId('63ffb22475532add2650e04c'),
    name: 'Nature',
    __v: 0
```

Abbildung 4 - Aufbau Tag-File

4 Fazit

MongoDB ermöglicht es uns, neue Funktionen wie Likes, Kommentare, Anzahl der Viewer oder andere Elemente mühelos zu implementieren, ohne dass eine Anpassung der Datenstruktur erforderlich ist. Dies erleichtert die Skalierbarkeit der Anwendung erheblich und erhöht ihre Flexibilität, da Änderungen an der Datenbank ohne die üblichen Einschränkungen eines relationalen Datenbanksystems vorgenommen werden können. Ein exemplarisches Anwendungsbeispiel für diese Vorgehensweise besteht in der Erweiterung der Funktionalität um eine Like- oder Kommentarfunktion. In diesem Kontext erfolgt die Speicherung von Likes in der User Collection anhand der entsprechenden User-IDs, sowie von Kommentaren in einer eigenen Comment Collection unter Verwendung der jeweiligen Kommentar-IDs. Auf diese Weise wird eine effiziente und differenzierte Organisation der relevanten Daten innerhalb der MongoDB-Datenbank gewährleistet (siehe Abbildung 5).

```
{
    _id: ObjectId('6408f831a55840c36aeee0ed'),
    image: 'http://localhost:3000/profile/file-image_1678309425046.png',
    description: 'Das ist ein Testupload',
    tags: [
        ObjectId('6408b017294b8657f69e3046'),
        ObjectId('6408b017294b8657f69e3048'),
        ObjectId('6408b017294b8657f69e304a')
    ],
    likes: [
        ObjectId('6408b01725488657f69e3046'),
        ObjectId('64088548294b8657f69e3048'),
        ObjectId('6408c85486948657f69e304a')
    ],
    comments: [
        ObjectId('6408b01784594557f69e3046'),
        ObjectId('6408b017294b8658454e3048'),
        ObjectId('6408b017294b4849649e304a')
    ],
    createdOn: ISODate('2023-03-08T21:03:45.100Z'),
    updatedOn: ISODate('2023-03-08T21:03:45.100Z')
}
```

Abbildung 5 - Einfache Funktionserweiterung