Modellierung eines Forum in MongoDB

- Modellierung eines Forum in MongoDB
- Auslegung unserer `JSON Documents`
  - Fälle
    - Was brauchen wir in den jeweiligen Fällen?
      - Ein Nutzer meldet sich an:
      - Ein Nutzer will alle Subforen haben:
      - Ein Nutzer will alle Topics in einem Subforum haben:
      - Ein Nutzer will alle Posts eines Topics haben:
      - Ein Nutzer will alle Posts eines Autors haben:
  - Dokumente
    - User
    - Subforum
    - TopicInSubforum
    - Topic
    - Post
    - Gesamtbild
- Anlegen der Daten in der Datenbank
- Wie kommen wir an unsere Daten?
  - Einloggen:
  - List aller Subforen:
  - Alle Topics innerhalb eines Subforum anzeigen lassen:
  - Öffnen eines Topics und anzeigen der Posts:
  - Profil und Posts eines Nutzers:

Wir modellieren eine Forum Software. Das Forum besteht aus mehreren Sub-Foren, die wiederum mehrere Topics enthalten können. Jedes Topic kann mehrere Posts enthalten.

Mit MongoDB können wir die Struktur in 2 unterschiedliche Art und Weise realisieren:

- Nested Documents (Wir schreiben z.B. für jedes Subforum nur ein Dokument, dieses enthält dann alle Topics, mit allen Posts)
- Verlinkung über DocumentIDs. (Ziemlich Analog zu relationalen Datenbanken. Wir machen für jede Entität ein eigenes Dokument)

Man kann auch ein Hybrid aus beiden Machen, sodass wir die Joins minimieren (Nested Documents) aber nicht unbedingt immer alles holen müssen (Relational). Es bietet sich z.B. an, die Posts eines Topics direkt mit in das Topic zu schreiben, da wir eigentlich nie die Posts einzelnd holen wollen. Dann sollten wir auch die Posts jedes Authors in sein Dokument mit reinschreiben.

Die Topics eines Subforums werden wir aber per DocumentID und in das Subforum verlinken. Außerdem werden wir den Titel des Forums mitschreiben. Das erlaubt es uns alle wichtigen Daten zu bekommen um eine Liste aller Topics im Subforum anzuzeigen, ohne das wir direkt alle Posts mitgeschickt bekommen, was zu einer großen Response führt. Wir brauchen ja nicht alle Posts aus allen Topics des Subforums. Sondern brauchen diese nur, wenn wir uns das Topic angucken wollen. Genau diesen Ansatz werden wir hier verfolgen.

Auslegung unserer JSON Documents

Fälle

Wir haben mehrere Fälle in denen wir Daten aus der Datenbank brauchen:

- Ein Nutzer meldet sich an
- Ein Nutzer will alle Subforen haben
- Ein Nutzer will alle Topics in einem Subforum haben
- Ein Nutzer will alle Posts eines Topics haben
- Ein Nutzer will alle Posts eines Autors haben

Was brauchen wir in den jeweiligen Fällen?

Ein Nutzer meldet sich an:

Wir brauchen seinen Nutzernamen und sein Passwort um die Logindaten zu überprüfen. Wir holen also nur diese Felder.

Ein Nutzer will alle Subforen haben:

Wir brauchen alle Subforen und Grundlegende Informationen zu den Topics in jedem Subforum. Das erlaubt es uns eine Abfrage zu sparen.

Ein Nutzer will alle Topics in einem Subforum haben:

Die Topics haben wir bereits mit der Abfrage der Subforum bekommen und können diese ohne neue Abfrage darstellen.

Ein Nutzer will alle Posts eines Topics haben:

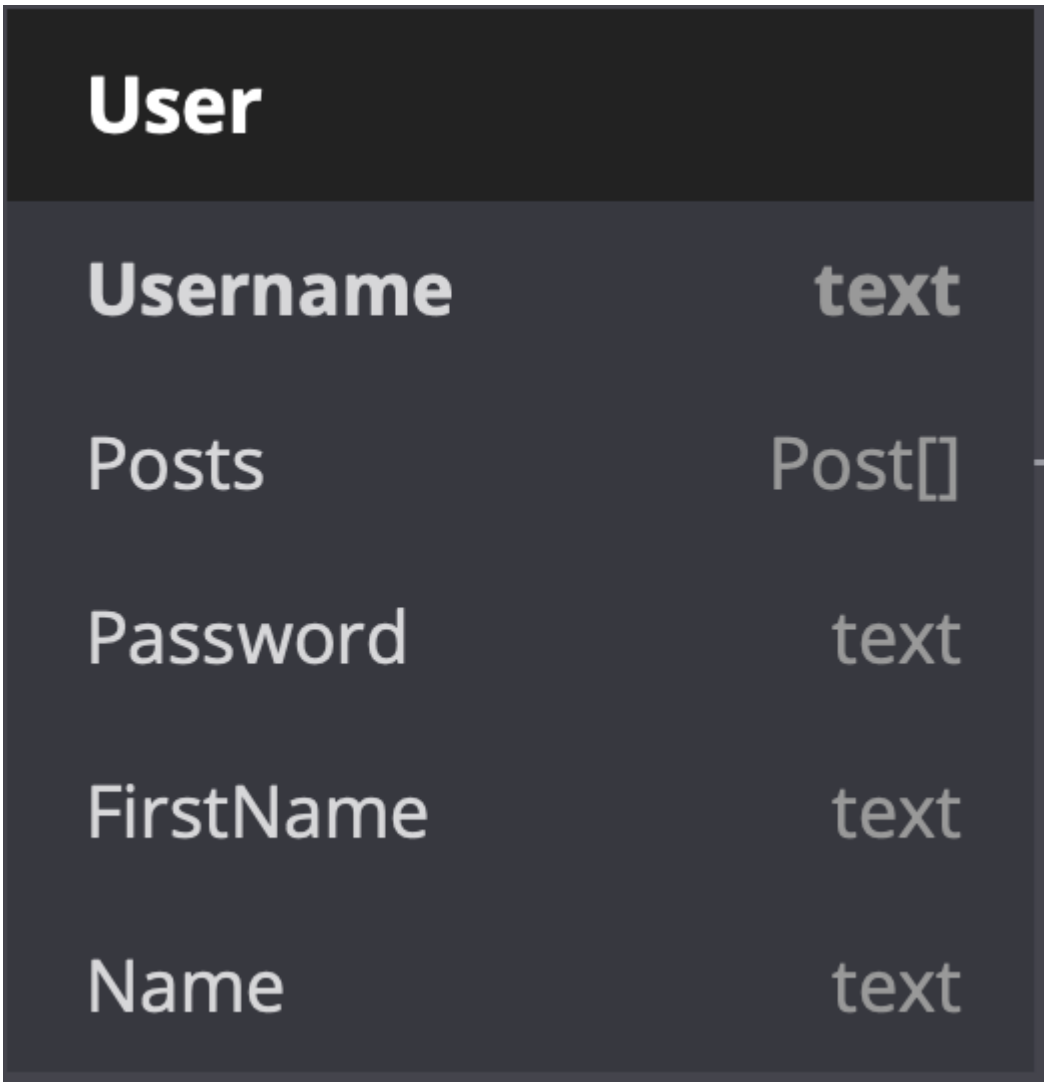
Hierfür nutzen wir die TopicID die wir in der Subforum Anfrage bereits bekommen haben um ein Topic Dokument mit mehr Daten zu bekommen. In dieses Dokument schreiben wir auch ein Array an Posts des Topics. Das erlaubt es uns direkt alle Posts anzuzeigen, ohne erneut Anfragen oder sogar ein aufwendingen JOIN zu machen.

Ein Nutzer will alle Posts eines Autors haben:

Dies ist sehr ähnlich der Topics. Wir schreiben in jeden User seine erstellten Posts in ein Array. Dadurch haben wir direkt zugriff auf die Posts. Wenn wir die Posts nicht benötigen, z.B. beim Login, dann lassen wir das Feld bei der Abfrage einfach aus.

Dokumente

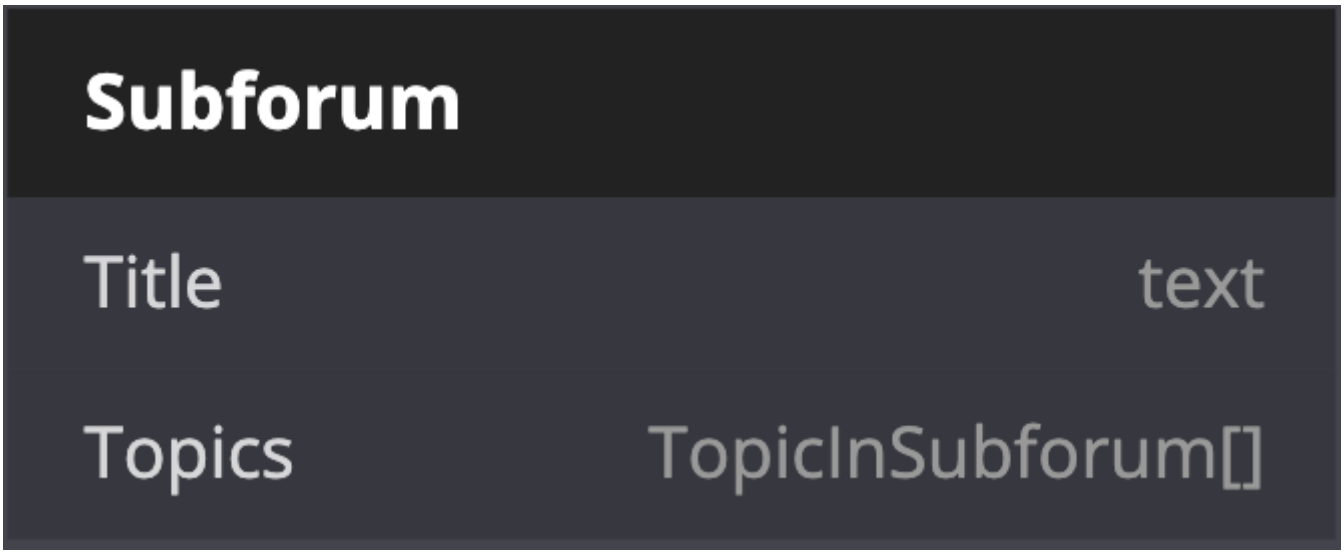
User



Als JSON:

```
{
  username: string,
  password: string,
  firstName: string,
  name: string,
  posts: Post[]
}
```

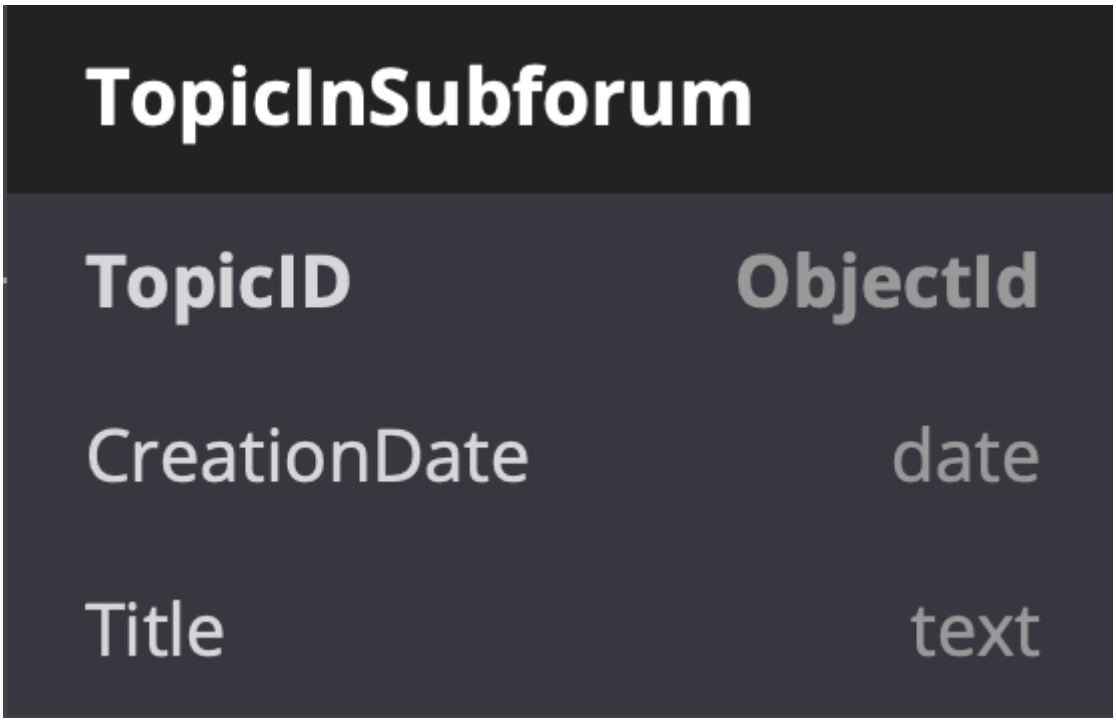
Subforum



Als JSON:

```
{
  title: string,
  topics: TopicInSubforum[]
}
```

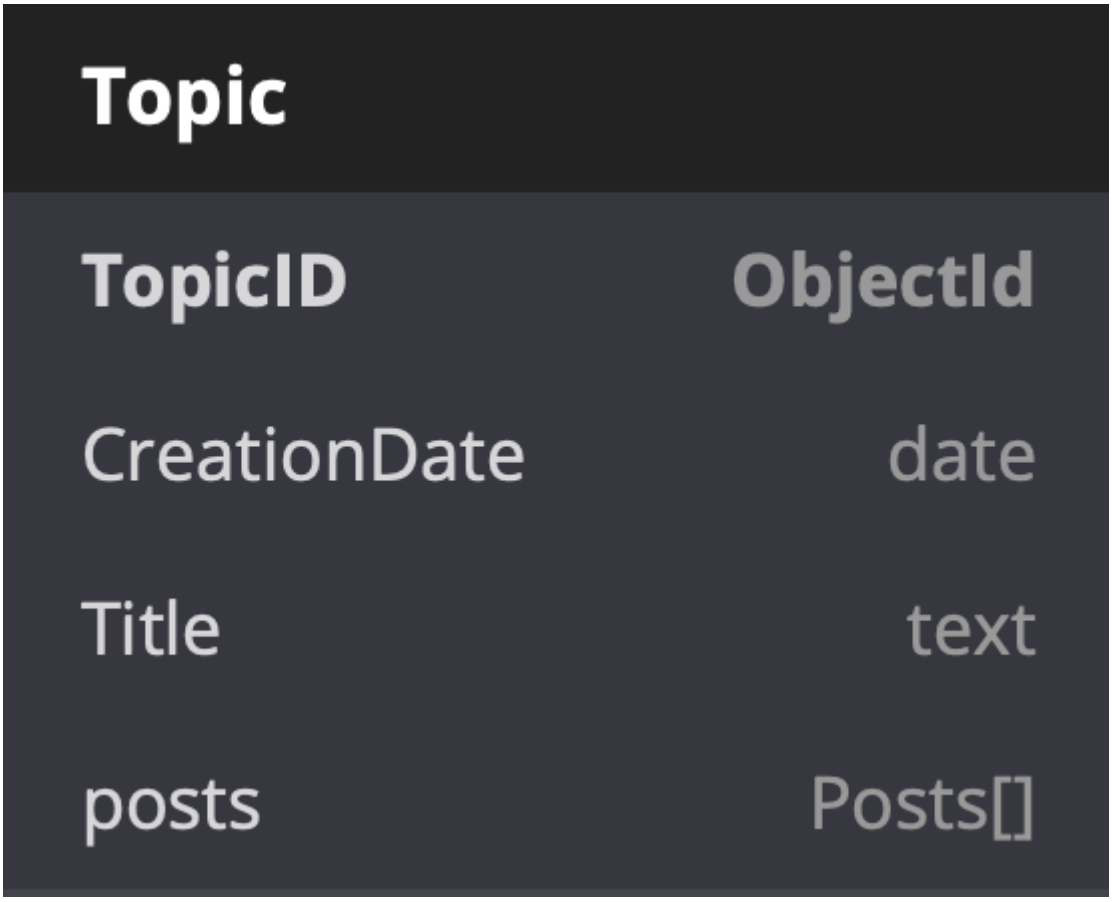
TopicInSubforum



Als JSON:

```
{
  topicID: number,
  title: string,
  creationDate: Date
}
```

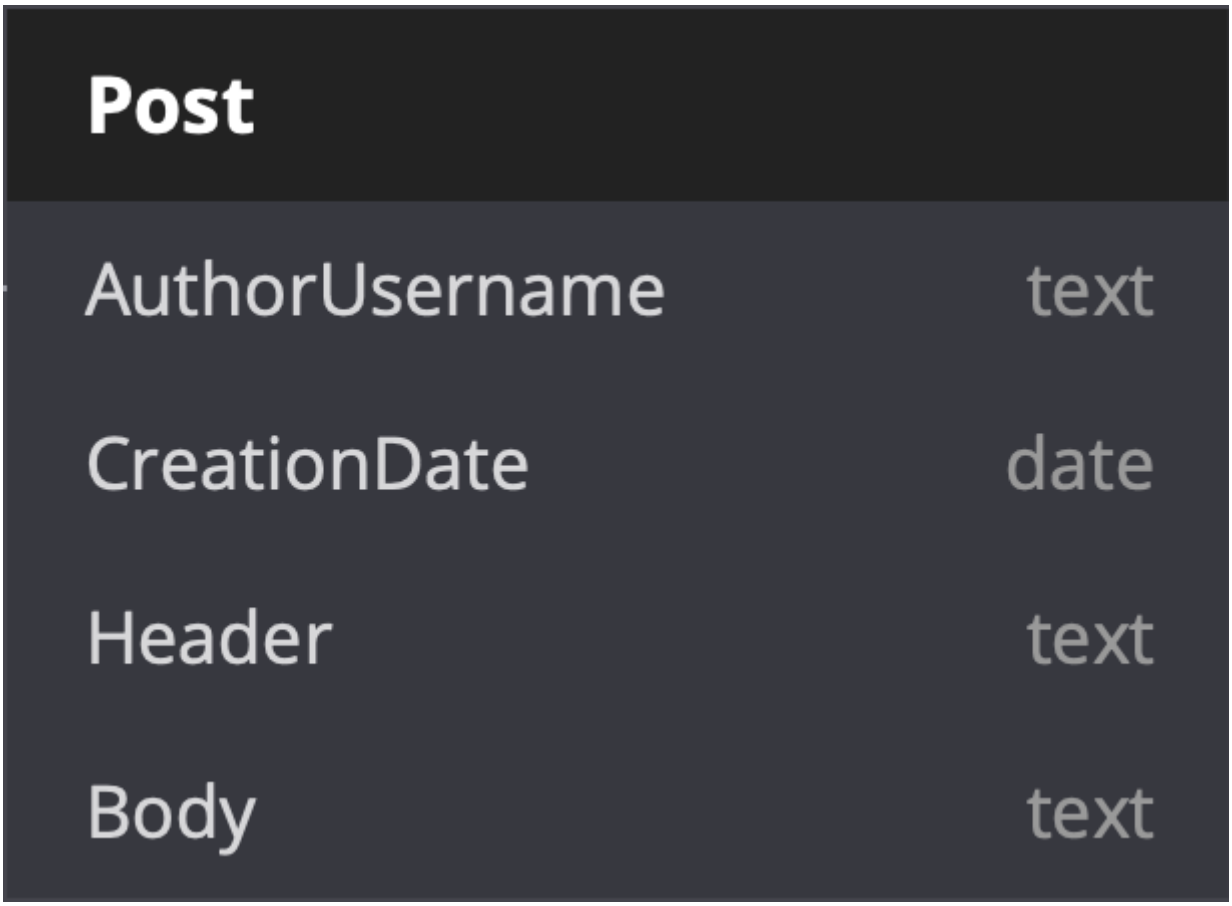
Topic



Als JSON:

```
{
  topicID: number,
  creationDate: Date,
  title: string,
  posts: Post[]
}
```

Post

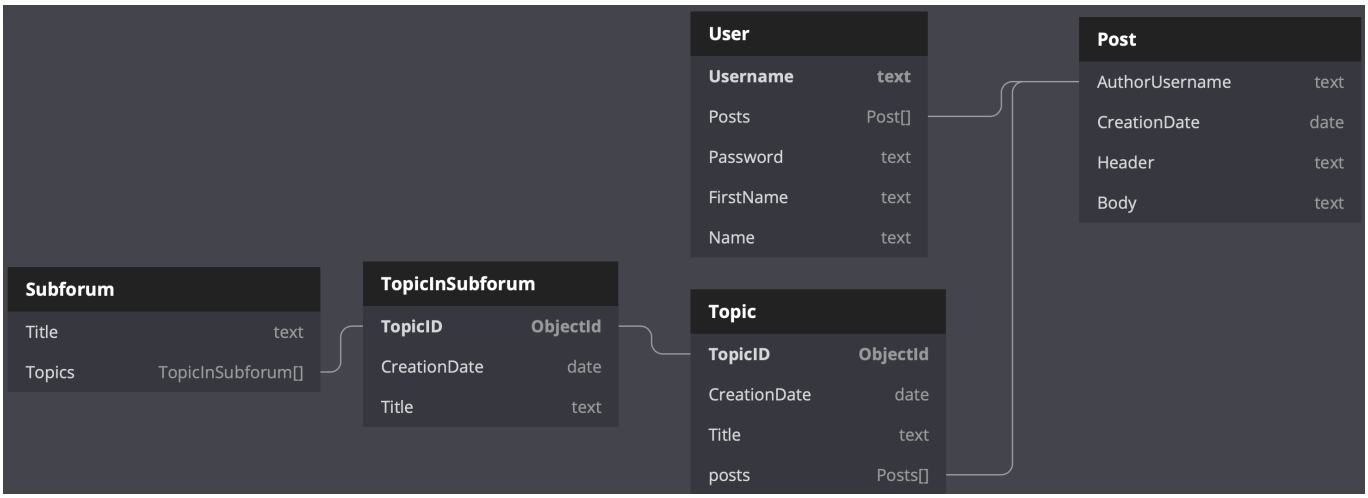


Als JSON:

```
{
  postID: number,
  creationDate: Date,
  authorUsername: string,
  header: string,
  body: string
}
```

Gesamtbild

Das Ganze als Gesamtbild:



Anlegen der Daten in der Datenbank

Um nun mit dem Modell zu arbeiten legen wir uns einige Beispieldaten in der Datenbank an. Wir fangen zunächst damit an, 2 Nutzer und 2 Subforen anzulegen. Die Daten sind im JSON Ordner.

Als aller erstes erstellen wir uns eine Datenbank in der wir alle Collections abspeichern werden:

```
use Forum;
```

Zunächst legen wir uns für die Nutzer eine Collection an:

```
db.createCollection("Users");
```

Und eine Collection für die Subforen:

```
db.createCollection("Subforums");
```

Jetzt legen wir uns die ersten beiden Nutzer an:

```
db.Users.insertMany(
  [
    ...
  ]
)
```

Analog dazu die Subforen:

```
db.Subforums.insertMany([
  ....
])
```

Jetzt erstellen wir unser erstes Topic. Dazu ist jetzt zu beachten, dass wir einmal ein neues `Topic` erstellen. Dies kommt in die Collection `Topics` und wir das Topic Array in dem jeweiligen Subforum mit updaten müssen.

Die TopicID brauchen wir nicht selber zu generieren, dies macht MongoDB automatisch als `_id` Feld.

Das Datum würden wir beim erstellen im Programm Code generieren und mitschicken. Hier ist es in den Daten fest vorgegeben.

```

```

```
db.createCollection("Topics")
db.Topics.insertOne({
  ...
})
```

Jetzt noch das Subforum mit den gleichen Daten, aber ohne das `posts` Array, updaten. Hierbei sollten wir die `DocumentID` des erstellten Topics mitschreiben, damit wir diese später für die Abfragen zur Verfügung haben.

```
db.Subforums.updateOne(
  { title: 'Subforum 1' },
  { $set: {
    topics: [{
        topicId: ObjectId(<objectID>)d
        title: "Topic 1",
        creationDate: "2020-10-01T10:27:01.975Z"
    }]
  }}
)
```

Damit haben wir unser Topic erfolgreich erstellt.

Jetzt füllen wir noch unser Topic mit 3 Posts. Dazu schreiben wir die `posts` einmal in das Posts Array des Topics und einmal in das `posts` Array des Nutzers.

Als erstes updaten wir unser Topic:

```
db.Topics.updateOne(
  { _id: ObjectId(<objectID>) },
  { $set: {
    posts: [
      ...
    ]
  }}
)
```

Und noch unseren Nutzer:

```
db.Users.updateOne(
  { username: 'username 1' },
  { $set: {
    posts: [
      ...
    ]
  }}
)
```

Wie kommen wir an unsere Daten?

Gehen wir also nochmal die User-Story durch:
Hinweis wir können an alle querries `.pretty()` dranhängen um das Ergebnis lesbarer zu machen.

Einloggen:

Wir melden uns mit den `username` an und brauchen `username` und `password`

```
db.Users.findOne({
  username: <username>
}, {
  username: 1,
  password: 1
})
```

List aller Subforen:

Wir holen uns alle Subforen:

```
db.Subforums.find()
```

Alle Topics innerhalb eines Subforum anzeigen lassen:

Die notwendigen Daten hierfür haben wir bereits in dem `topcis` Array des jeweiligen Subforum.

Öffnen eines Topics und anzeigen der Posts:

Wir holen uns aus der `Topics` Collection das jeweilige Topic. In dem Dokument bekommen wir dann die Posts über das `posts` Array:

```
db.Topics.findOne(
  { _id:  ObjectId(<objectID>) }
)
```

Profil und Posts eines Nutzers:

Wir brauchen `username`, `firstName`, `name` und `posts` eines Nutzers:

```
db.Users.findOne({
  username: <username>
}, {
  username: 1,
```

```
    firstName: 1,
    name: 1,
    posts: 1,
  })
```