SÃO PAULO UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE
SCC0219 - Introduction to Web Development

Nome: Michelle Wingter da Silva

nUSP: 10783243

Exercises 7: NoSQL

1. What is the CAP theorem and how is it related to NoSQL systems? Which CAP requirement MongoDB does not implement?
(2 pts)

O Teorema de CAP (ou Teorema de Brewer) é o teorema que afirma que é impossível que o armazenamento de dados distribuído forneça simultaneamente mais de duas dentre três características: Consistência, Disponibilidade, Partição. Ou seja, se por exemplo houver uma partição, é preciso escolher entre consistência e disponibilidade.

Os bancos de dados NoSQL são classificados com base nas duas características de CAP suportadas:

- Banco de dados CP: oferece consistência e tolerância de partição à custa da disponibilidade. Quando uma partição ocorre entre dois nós, o sistema deve tornar indisponível o nó não consistente até que a partição seja resolvida;
- Banco de dados AP: oferece disponibilidade e tolerância de partição à custa da consistência. Quando uma partição ocorre, todos os nós permanecem disponíveis, mas aqueles na extremidade incorreta de uma partição podem retornar à uma versão mais antiga que os outros.
- Banco de dados da CA: fornece consistência e disponibilidade em todos os nós. No entanto, isto não é possível se houver uma partição entre dois nós no sistema e, portanto, não pode oferecer tolerância a falhas.

O MongoDB, em relação ao teorema da CAP, é um repositório de dados CP, ou seja, resolve partições de rede mantendo a consistência e comprometendo a disponibilidade. Dessa forma, o requerimento do CAP que o MongoDB não implementa é Disponibilidade.

2. What are the different main types of NoSQL databases? Include a small description of each type.
(2 pts)

Há 4 tipos básicos de bancos de dados NoSQL:

- NoSQL orientado a documento: é baseada em uma coleção de documentos, sendo um documento um objeto que contém um código único com um conjunto único com um conjunto de informações, podendo ser strings, documentos aninhados ou listas. Alguns exemplos são o MongoDB e o CouchBase.
- NoSQL Key-Value: indexa os dados a uma chave. Ao se armazenar os dados, sua forma de busca se dá por uma base similar a um dicionário, onde estes possuem uma chave. Alguns exemplos são: Oracle e Azure.

- NoSQL representado por Grafos: este modelo de armazenamento utiliza três componentes básicos: um grafo para representar um dado, ligações para representar a associação entre os grafos, e os atributos dos nós e relacionamentos. É um modelo muito usado onde exijam dados fortemente ligados. Alguns exemplos são: GraphBase e InfiniteGraph.
- NoSQL Modelo Colunar: é uma tabela que possui várias famílias de colunas, e dentro destas famílias, colunas onde estão as propriedades. Os valores das propriedades das colunas são semelhantes ao modelo Key-Value. São bancos de dados indicados para mídias sociais e problemas que envolvem consultas complexas.

3. **Create a web service API using the course Creating APIs with Node.js (from the Node.js Module). You should implement all the services and features used in the course up to lesson 30, with the exceptions of Custom Validations (Lesson 23). You should use only MongoDB validations. Features from lessons 31 to 40 (for instance, Password encryption (lesson 31), Welcome emails (lesson 32), etc.) are not to be implemented.**
   Tip: In this course (lesson 13), it shows how to set up an online version of MongoDB. However, the site it uses (https://cloud.mongodb.com) changed how things are done. You should instead install MongoDB locally and connect using the Compass Tool (included with MongoDB) or the Studio 3T Tool (used in the course). The MongoDB server is `localhost:27017`.
   (3 pts)

4. **To the server created in the last exercise, you should add Pet Shop services. Services have an implementation similar to Products. They have the following operations:**
   - **`get`**
   - **`getBySlug`**
   - **`getById`**
   - **`post`**
   - **`put`**

   In addition, you should implement a new service, called `getPartnerHours`, that given part of the name of a partner and a list of hours returns a list with title and partner name of the services offered in, at least, one of these hours. For instance, if the following Services exist:

```
{title: "Tosa radical", slug: "tosa-radical", description:
"Tosa bem curta", partner: "Luis Braga", price: 120.45, hours:
[10, 11, 14, 15, 16]}

{title: "Banho relaxante", slug: "banho-relaxante",
description: "Banho quente demorado", partner: "José Braga
Silva", price: 50.0, hours: [9, 11, 15, 17]}
```

   And a `getPartnerHours` request has the following JSON data:

```
{"name": "braga", "hours": [11, 16]}
```

The following JSON result should be returned:

```
[{"title": "Tosa radical", "partner":"Luis Braga"}, {"title":
"Banho relaxante", "partner":"José Braga Silva"} ]
```

The Service schema:

```
const schema = new Schema({
    title: {
        type: String,
        required: [true, 'O título é obrigatório'],
        trim: true
    },
    slug: {
        type: String,
        required: [true, 'O slug é obrigatório'],
        trim: true,
        index: true,
        unique: true
    },
    description: {
        type: String,
        required: [true, 'A descrição é obrigatória']
    },
    partner: {
        type:String,
        Required: [true, 'O nome do profissional que presta o
serviço é obrigatório']
    },
    price: {
        type: Number,
        required: [true, 'O preço é obrigatório']
    },
    hours: [{
        type: String,
        required: true
    }]
});
```

(3 pts)

Good Work!