UNIVERSIDADE FEDERAL DE GOIÁS
SOFTWARE PARA PERSISTÊNCIA DE DADOS
PROF. ELIAS B. FERREIRA
DISCENTES: FERNANDO SEVERINO ALMEIDA (201802762), MICHELLY SILVA LIMA (201802780)

Trabalho Final: Projeto Web - Agenda de Vacinação

Para o desenvolvimento deste projeto final de Software de Persistência de Dados, foram utilizadas algumas ferramentas. Tais ferramentas, assim como o seu uso e o passo-a-passo para executar a aplicação desenvolvida encontram-se no repositório do projeto, disponível no Github no link https://github.com/fernandosev/projetoFinalSPPD.

Entre essas ferramentas, foram utilizados o banco de dados MongoDB e o framework de persistência Mongoose. Neste documento iremos explorá-los um pouco mais para que fique claro como este framework funciona e para que é empregado.

Uma breve descrição sobre MongoDB

De acordo com o Tecnoblog, o MongoDB é um banco de dados orientado a documentos que possui código aberto (open source) e foi projetado para armazenar uma grande escala de dados, além de permitir que se trabalhe de forma eficiente com grandes volumes. Ele é categorizado no banco de dados NoSQL (not only SQL) pois o armazenamento e a recuperação de dados no MongoDB não são feitas no formato de tabelas.

No MongoDB é possível criar vários bancos de dados, já que este funciona como um servidor de banco de dados, onde as informações são armazenadas, mas o processo é mais fluido, independente, com os elementos tendo identificações únicas.

Entre suas vantagens, o MongoDB permite criar vários bancos de dados e várias coleções dentro do principal. Nestas coleções, ficam os documentos com os dados que serão armazenados no banco de dados do Mongo e dentro da coleção pode conter vários documentos. Não é necessário que um documento seja semelhante ao outro.

Nos documentos, é possível armazenar dados aninhados, permitindo criar relações complexas entre eles e armazená-los no mesmo documento, tornando o trabalho e a busca mais eficientes em comparação com o SQL.

Além disso, outras vantagens são:

- Não precisa projetar o esquema do banco de dados ao trabalhar com o mongoDB;
- Fornece grande flexibilidade para os campos nos documentos;
- Trabalha com dados heterogêneos;
- Não requer nenhuma adição ou injeção de SQL;
- Facilmente integrável com o Big Data Hadoop (com diversas versões open source).

Porém, o MongoDB também possui desvantagens. Entre elas estão:

- Utiliza muita memória para armazenar e estocar dados;
- Limite de 16 MB de dados para armazenar nos documentos;
- Limite para aninhar dados nos arquivos BSON em até 100 níveis.

Sobre o Mongoose

Partindo para o Mongoose, ele é uma biblioteca para Node.js que é baseada em **Schemas** para modelar os objetos de uma aplicação para o banco de dados MongoDB.

A instalação do mongoose é feita com o comando:

```
$ npm install --save mongoose
```

Após isso, podemos estabelecer uma conexão com nosso banco de dados usando a função **connect()**, passando como parâmetro a URL de conexão para o MongoDB.

Em nossa aplicação, a conexão está feita da seguinte forma:

```
const mongoose = require("mongoose");
mongoose.connect(
"mongodb+srv://creeper:c1012UFG@cluster0.ybwxo.mongodb.net/vacinacao?re
tryWrites=true&w=majority",
  {
    useNewUrlParser: true,
    useUnifiedTopology: true,
  }
);
```

A partir disso, os objetos da aplicação podem ser modelados para o banco de dados a partir do uso dos **Schemas**. No mongoose, a criação do **Schema** é feita conforme exemplo abaixo, retirado da nossa aplicação:

```
import * as mongoose from "mongoose";
const Schema = mongoose.Schema;

const CalendarSchema = new mongoose.Schema({
  user: { type: Schema.Types.ObjectId, ref: "User" },
  vacine: { type: Schema.Types.ObjectId, ref: "Vacine" },
  date: Date,
  hour: Number,
  dose: Number,
  Observations: String,
  status: String,
});

module.exports = mongoose.model("Calendar", CalendarSchema);
```

Após a definição, utilizamos um module.exports para poder utilizar este Schema como *Model*, fornecido pelo próprio Mongoose, no resto da aplicação.