Como configurar a conexão de uma aplicação NodeJS com MongoDB

O projeto usado como base para fazer a adaptação do código é o projeto MMORPG baseado na série de TV Game Of Thrones, o código é o disponibilizado na atual aula 90 do curso.

As dependências do projeto estão da seguinte maneira:

```
"dependencies": {
  "body-parser": "^1.18.2",
  "consign": "^0.1.2",
  "ejs": "^2.5.2",
  "express": "^4.16.2",
  "express-validator": "^2.20.8",
  "mongodb": "^3.0.2"
 }
```

E os arquivos que vamos modificar para configurar a nova conexão são:

/app/models/UsuariosDAO.js
/config/dbConnection.js

O arquivo dbConnection que antes utilizava uma instância da classe do Mongo para criar uma conexão com o banco de dados agora passa a utilizar o método connect com alguns parâmetros de configuração, abaixo segue o novo código:

```
var mongo = require("mongodb").MongoClient;
var assert = require("assert");

const url = "mongodb://localhost:27017";
const dbName = "got";

var connMongoDB = function(dados) {
	mongo.connect(url, function(err, client) {
		assert.equal(null, err);
		console.log("Connected successfully to server");
		const db = client.db(dbName);
		query(db, dados);
		client.close();
	});
};
```

```
function query(db, dados) {
	var collection = db.collection(dados.collection);
	switch (dados.operacao) {
		case "inserir":
			collection.insertOne(dados.usuario, dados.callback);
			break;
		default:
			break;
	}
}

module.exports = function() {
	return connMongoDB;
};
```

Vamos às explicações, no começo do código há a importação do MongoClient que tem que ser usado na nova versão, e logo abaixo vem a importação do assert, que foi usado para verificar erros na conexão com o banco de dados.

Na sequência temos as variáveis de credenciais do banco, primeiro a url para conexão e em seguida o nome do banco de dados que queremos conectar.

Chegamos à função de conexão, a variável connMongoDB continua a mesma, o que mudou foi seu escopo e os parâmetros recebidos, agora recebemos na conexão um objeto como parâmetro, com os dados para a conexão e manipulação dos dados, mas não se preocupe, será explicado mais a frente sobre os parâmetros, dentro da função usamos a variável importada no começo para chamar o método connect passando alguns dados para a conexão, e é no callback dessa função que executamos as operações no banco de dados, dentro da função de callback a primeira coisa que verificamos é se não houve erros na conexão, se não houve chegamos na declaração da variável db que guarda a referência do banco de dados que queremos executar as operações, em seguida chamamos a função query que é quem efetivará as operações, vamos falar dela logo abaixo, e por último para não serem geradas múltiplas conexões usamos o comando client.close() para finalizar a conexão.

A função query foi criada para que fosse evitada a repetição de código, pois do contrário teríamos que criar toda a configuração de conexão para cada operação que fosse realizada, com a função query podemos apenas inserir um case no switch para cada operação e dentro do case indicar o que exatamente queremos fazer, no exemplo de código acima há o case inserir e dentro dele temos o comando collection.insertOne(dados.usuario, dados.callback), esse comando irá inserir um documento na collection usuarios e em seguida renderizar uma página, vou explicar na sequência quais são os parâmetros passados para essa função.

O arquivo UsuariosDAO.js que antes chamava a conexão dentro de cada função indicando a operação a ser feita e o callback agora apenas cria um objeto de dados e os passa para a conexão, o novo código ficou da seguinte maneira:

```
function UsuariosDAO(connection) {
	this._connection = connection;
}

UsuariosDAO.prototype.inserirUsuario = function(usuario, res) {
	var dados = {
		operacao: "inserir",
		usuario: usuario,
		collection: "usuarios",
		callback: function(err, result) {
			res.send("olá Marilene");
		}
	};
	this._connection(dados);
};

module.exports = function() {
	return UsuariosDAO;
};
```

Como vocês podem ver dentro da função inserirUsuario criamos um objeto dados contendo algumas configurações para realizar a operação no banco de dados, esses índices dentro do objeto devem ser padrão para qualquer operação a ser realizada, são eles:

operacao = Deve receber uma string que coincida com o case da função query

usuario = É um índice variável, dependendo do case você pode passar outros dados, por exemplo, passar o índice noticia para adicionar noticias, basta ajustar no case qual dado usar

collection = Deve receber uma string indicando qual a collection que será manipulada no banco

callback = Deve receber uma função para ser executada como resposta da execução da operação no banco de dados

E por último chamamos a conexão passando esse objeto.

Usando esse novo modelo, que já foi testado, as operações no banco de dados ocorrem normalmente.

Caso tenham alguma dúvida, nos avisem para irmos evoluindo sobre o assunto ;)

Atenciosamente,