Utilizando o Mongoose

EC021 - Tópicos Avançados II
Sistemas Distribuídos

Prof. Adauto Mendes | E-mail: adauto.mendes@inatel.br

O Mongoose

- O Mongoose é uma biblioteca NodeJS que funciona como ODM (object data modeling).
- O Mongoose é uma biblioteca que fornece um ambiente de modelagem rigoroso para seus dados, reforçando a estrutura conforme necessário e mantendo a flexibilidade que torna o MongoDB poderoso.
- Isso que dizer que tanto o MongoDB como o NodeJS não possuem uma rígida tipagem de dados. Com o Mongoose é possível definir tipos para os campos dos objetos que serão persistidos no MongoDB.

Instalando o Mongoose

- Para instalar o Mongoose na nossa aplicação devemos executar o comando abaixo:

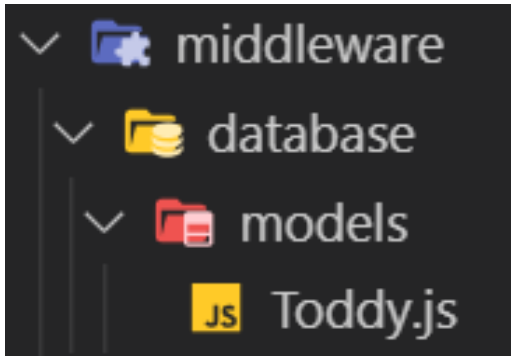
```
npm install mongoose
```

- Feito isto, adicione a importação da biblioteca no arquivo de entrada da aplicação (index.js):

```
const mongoose = require('mongoose');
```

Criando a Model

- Para que o Mongoose entenda a estrutura dos dados que serão persistidos é necessário criar um arquivo que modele estes dados. Criaremos um arquivo Toddy.js no caminho database/models:



Criando a Model

- Importe a biblioteca do Mongoose e também o módulo Schema do Mongoose:

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
```

- Feito isto vamos modelar nossa estrutura:

Criando a Model

- Nosso modelo ficará com a estrutura abaixo:

```
const Toddy = new Schema(
    {
        lote: { type: String, default: null },
        conteudo: { type: Number, default: null },
        validade: { type: String, default: null }
    },
    { timestamps: true }
)
```

Criando a Model

- Após modelado, exportaremos o modelo para podermos utilizar em outras partes do código:

```
module.exports = mongoose.model('Toddy', Toddy);
```

Realizando Conexão

- Para realizar a conexão com o MongoDB via Mongoose, devemos primeiro criar um objeto com as diretivas de conexão, além de criar uma constante com a URL do MongoDB:

```
const DB_URL = 'mongodb://localhost:27017';

const DB_SETTINGS = {
    useCreateIndex: true,
    useNewUrlParser: true,
    useFindAndModify: false,
    useUnifiedTopology: true,
    user: '',
    pass: '',
    dbName: 'ec021'
}
```

Realizando Conexão

- Com as diretivas configuradas configure a conexão no arquivo de entrada da aplicação (index.js):

```
mongoose.connect(DB_URL, DB_SETTINGS, (err) => {
    if (err) {
        console.log(`Erro na conexão com o MongoDB: ${DB.DB_ADDRESS}`);
        console.log(`${err.message}`);
    }
    else {
        console.log(`Servidor conectado ao MongoDB: ${DB.DB_ADDRESS}`);
    }
});
```

- É possível chamar esta função no callback da função *listen()* do Express.

Executando Operações

- No arquivo dao.js teremos que refatorar o código para utilizado o Mongoose. Para isto teremos que importar a model criada:

```
const Toddy = require('./database/models/Toddy');
```

- Feito isto podemos implementar as nossas funções de CRUD...

Executando Operações

- Para inserir uma informação no MongoDB utilizaremos a função *create()* do nosso model:

```
inserir: async function (toddy) {
    let result = await Toddy.create(toddy);
    return result;
}
```

Executando Operações

- No index.js devemos alterar a função *inserir()*:

```
async function inserir(req, res, next) {
    /* Recebendo os dados da requisição */
    let toddy = {
        lote: req.body.lote,
        conteudo: req.body.conteudo,
        validade: req.body.validade
    }

    let result = await dao.inserir(toddy);

    return res.json(result);
}
```

Testando

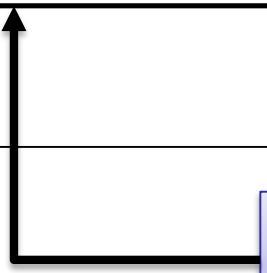
- Enviando o body abaixo via Postman:

```
{
    "lote": "ABC",
    "conteudo": 123,
    "validade": "15/02/2015"
}
```

Testando

- Devemos receber isto:

```
{
    "lote": "ABC",
    "conteudo": 123,
    "validade": "15/02/2015",
    "_id": "5d777f075138885040a7f310",
    "createdAt": "2019-09-10T10:46:31.287Z",
    "updatedAt": "2019-09-10T10:46:31.287Z",
    "__v": 0
}
```



Campos criados a partir da definição do **`timestamps: true`** na model

Sua vez!

- Utilize os métodos abaixo para refatorar as demais funções da sua aplicação:
  - atualizar: *findByIdAndUpdate*
  - listar: *find*
  - buscarPorId: *findById*
  - buscarVencidos: *find*
  - buscarLotes: *find*
  - buscarPorLote: *find*
  - excluir: *findByIdAndDelete*