Tutorial: Como usar GridFs do MongoDB para guardar arquivos

Neste tutorial vamos mostrar como faremos para guardar e acessar arquivos usando MongoDB

Instalação

Este tutorial segue os passos de instalação no Ubuntu 14.04, caso esteja em outro sistema operacional, veja a Documentação Oficial.

O primeiro passo é informar o sistema de manutenção de pacotes o endereço do repositório do MongoDB:

```
$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10
$ echo "deb http://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.0 multiverse" |
sudo tee /etc/apt/sources.list.d/mongodb-org-3.0.list
```

Depois é só usar o gerenciador de pacotes para baixar e intalar o Mongo:

```
$ sudo apt-get update
$ sudo apt-get install -y mongodb-org
```

Configuração

Nenhuma configuração especial é necessária para este tutorial.

Usando o GridFS

Vamos supor que queremos guardar um arquivo de áudio na base MongoDB, podemos usar um arquivo de podcast como exemplo por ter uma duração similar a uma transcrição.

Caso precise de um exemplo, baixe o último episódio do *Ponta Solta* usando o seguinte comando:

```
$ wget -O teste.mp3 http://www.terceiraterra.com/audio/ps-20151005_descrenca.mp3
```

Para colocar o arquivo no bando de dados do mongo na base "test", utilizamos o comando PUT do *mongofiles* da seguinte forma:

```
$ mongofiles -d test put teste.mp3
```

A partir deste momento, temos uma cópia do mp3 na base de dados mongo, então não precisamos mais da cópia original e podemos deletá-la se quisermos:

```
$ rm teste.mp3
```

Na interface de shell do mongo podemos ver que ele guardou dois tipos de informações sobre o arquivo de áudio, o *files* e o *chunks* se quisermos os meta dados do arquivo podemos ver em files da seguinte forma:

```
db.fs.files.find()
```

Na resposta virá as seguintes informações:

```
{
    "_id" : ObjectId("562d13c4879fda2760000001"),
    "chunkSize" : 261120,
    "uploadDate" : ISODate("2015-10-25T17:39:18.130Z"),
    "length" : 77852672,
    "md5" : "43636b40941ef40693ca3e987ee0e15c",
    "filename" : "teste.mp3"
}

Obs: o _id e o uploadDate não necessariamente serão os mesmos, mas os outras
informações devem ser precisamente as mostradas acima.
```

O conteúdo do áudio foi guardado nos chunks e não é muito fácil de ler (Como é esperado de um arquivo binário), mas se quisermos saber em quantos pedaços o aquivo foi quebrado para se acomodar na base podemos rodar o seguinte comando no shell do mongo:

```
db.fs.chunks.find({files_id:ObjectId("562d13c4879fda2760000001")}).count()

Obs: caso tenha sido gerado outro _id na inserção, mude o argumento do comando
acima para o id gerado na sua inserção
```

Se quisermos recuperar o arquivo guardado podemos usar de novo o *mongofiles* executando o seguinte comando no terminal:

```
$ mongofiles get_id 'ObjectId("562d13c4879fda2760000001")'
```

Se o comando acima falhar, tente recuperá-lo pelo nome usando o comando abaixo Obs: é conhecido por nós o fato de que diferentes arquivos podem ter mesmo nome, portanto essa não é a maneira mais geral que pretendemos usar para recuperar os arquivos a priori.

```
$ mongofiles get teste.mp3
```

Pronto, o arquivo foi recuperado.

Por fim, para remover o arquivo do BD NoSQL, você pode usar o seguinte comando: ATENÇÃO! O comando abaixo removerá todos os arquivos do BD com o nome teste.mp3

```
$ mongofiles delete teste.mp3
```

Considerações finais

Na aplicação real não utilizaremos estes comandos do terminal, mas sim drivers de mongo com suporte para GridFs para a linguagem de nossa aplicação.