V-Web

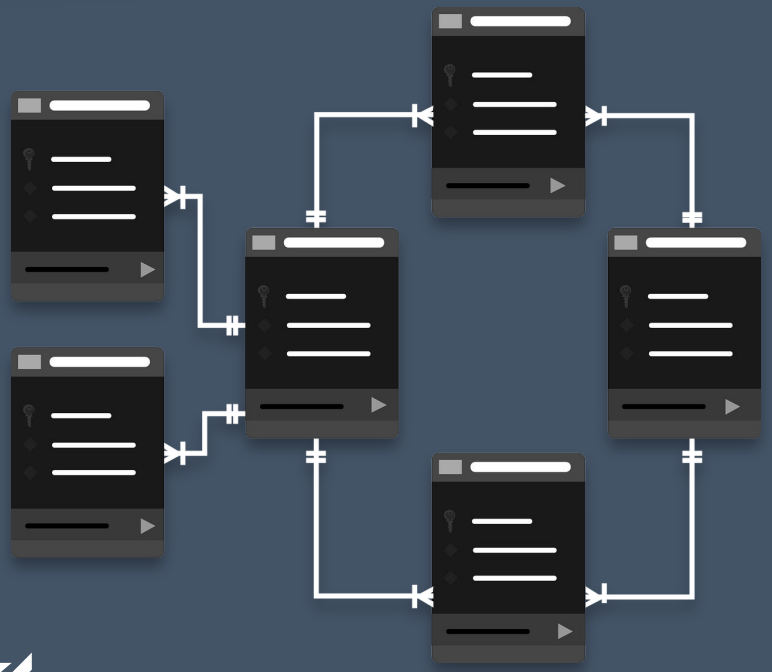
Módulo Backend - Dia 2

mongoDB

Por que um Banco de Dados?

- Rapidez
- Manutenção da Integridade
- Acesso Concorrente
- Acesso Restrito
- Fácil Administração

Modelos de Bancos de Dados



Collection

Document

```
{
    pessoa_ID: "93742",
    nome: "João da Silva",
    doacoes: [100, 10, 55, 13],
    data_nasc: "1984-09-29"
}
```

Document

```
{
    pessoa_ID: "915513",
    nome: "João da Silva Jr.",
    doacoes: [9, 8, 11, 20],
    data_nasc: "2000-01-01"
}
```

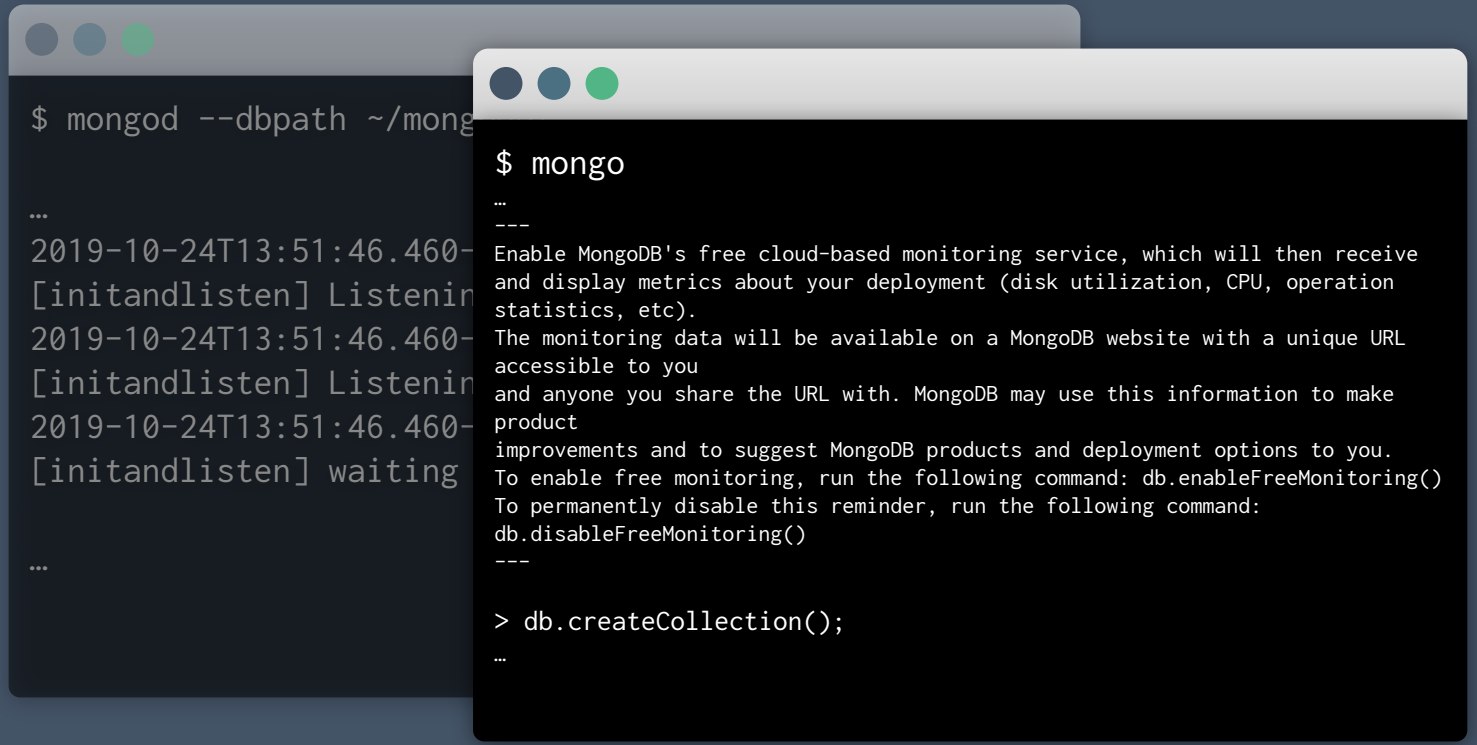
Iniciando o Servidor

```
$ mkdir mongodb
$ mongod --dbpath ~/mongodb/

…
2019-10-24T13:51:46.460-0300 I  NETWORK  [initandlisten] Listening on
/tmp/mongodb-27017.sock
2019-10-24T13:51:46.460-0300 I  NETWORK  [initandlisten] Listening on 127.0.0.1
2019-10-24T13:51:46.460-0300 I  NETWORK  [initandlisten] waiting for connections on port
27017

…
```

Se conectando ao servidor



Operações na database mongo

CRUD

Create **R**ead **U**pdate **D**elete

Operações de Criação/Inserção

```
db.coleção.insertOne({
        <objeto>
    })
```

```
db.coleção.insertMany({
        <objeto 1>,
        <objeto 2>,
            ...
    })
```

(a coleção é criada automaticamente)

Operações de Leitura

```
db.coleção.find({

<query>,

<projeção>

}).limit(<quantidade>)
```

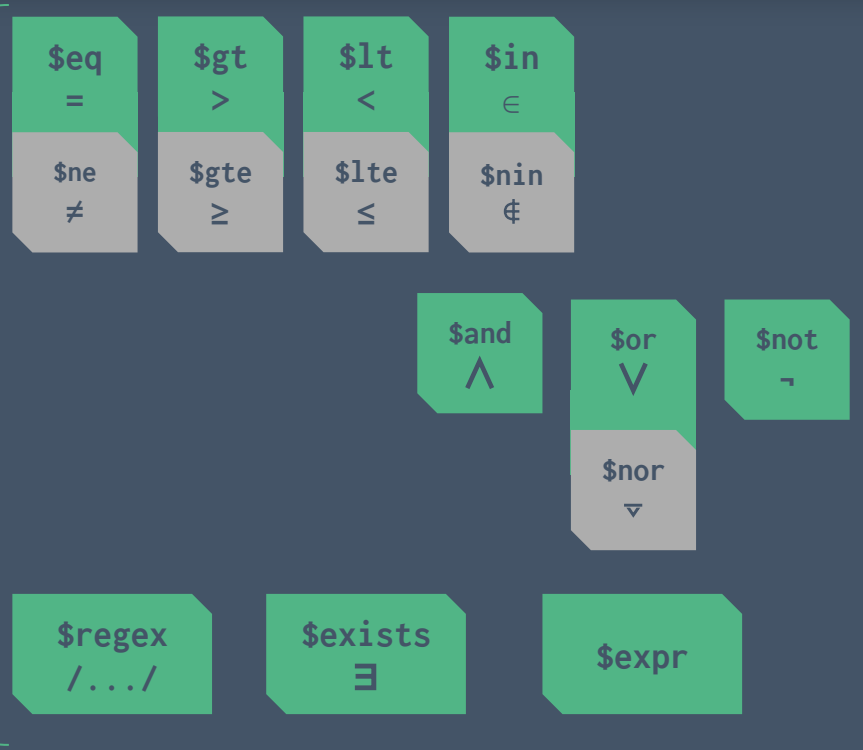
filtros para a pesquisa (objeto JSON contendo subconjunto de propriedades da coleção)

maneira na qual a pesquisa deverá ser retornada (também um objeto JSON)

Filtros para a pesquisa

```
{
    propriedade: { __: __(, __: __, …) }
}
```

operadores é um de



referência completa: https://docs.mongodb.com/manual/reference/operator/query/

Projeção

```
{
    “propriedadeMostrada”: 1,
    “outraPropriedade”: 1
}
```

```
{
    “propriedadeOcultada”: 0
}
```

referência completa: https://docs.mongodb.com/manual/reference/operator/projection/

Operação de Atualização

```
db.coleção.updateOne({
    <query>,
    <alteração>
})
```

```
db.coleção.updateMany({
    <query>,
    <alteração>
})
```

Alteração

```
$set: {propriedade: <valor>}

$unset: {propriedade: qualquerCoisa}

$currentDate: {propriedade: true} ou {propriedade: {$type: “timestamp”} ou
{$type: “date”}}


$push: {vetor: <valor>}

$pop: {vetor: <valor>}
```

referência completa: https://docs.mongodb.com/manual/reference/operator/update/

Operação de Deleção

```
db.coleção.deleteOne({
    <query>
})
```

```
db.coleção.deleteMany({
    <query>
})
```

Mongoose


node
JS

Schemas e Modelos

Schemas são abstrações sobre o sistema de documentos do Mongo de modo a fornecer uma estrutura mais rígida
Um Schema é gerado para cada coleção do mongoDB e gera um formato para os documentos pertencentes a essa coleção

Modelos são construtores de Objetos gerados a partir de Schemas.
Utilizando esses modelos podemos gerar objetos a serem inseridos em nossa database respeitando a restrições do Schema

Se conectando ao Banco de Dados

```
var mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/db', {useNewUrlParser:
true});

var db = mongoose.connection;

db.on('error', console.error.bind(console, 'Erro de conexão:'));
db.once('open', function() {
  // onde vamos trabalhar com a db
});
```

Criando o nosso Schema

```
var dogSchema = new mongoose.Schema({
    nome: String,
    raca: String
    // tipos permitidos:
    // String, Number, Date, Buffer, Boolean,
    // Mixed, ObjectId, Array, Decimal128, Map
});

dogSchema.methods.latir = () => console.log("Au Au");
```

Criando e usando Modelos

```
var Dog = mongoose.model('Dog', dogSchema);

var satan = new Dog({
    nome: "Tremedeira",
    raca: "Chihuahua"
});

satan.latir(); // "Au, Au"

satan.save(); // Agora nosso dog tá no banco de dados
```

Operações CRUD

- `Dog.find()`
- `Dog.updateOne()`
- `Dog.updateMany()`
- `Dog.deleteOne()`
- `Dog.deleteMany()`

Querys

As operações no Mongoose geram Querys, que podemos executar de 2 maneiras:

- `Modelo.op(..., function(erro, resultado){})`
- ```
  let operacao = Modelo.op(...);
  operacao.exec(function(erro, resultado){});
  ```


  ```

Projeto Final

Adote um Bixo Inc.

A empresa requer as seguintes interações:

1. Adicionar Cachorros ao banco de dados de disponíveis para adoção
2. Obter os dados dos Cachorros disponíveis para adoção
3. Remover Cachorros que venham a falecer
4. Adicionar Pessoas disponíveis a adotar
5. Remover Pessoas da lista de adoção
6. Transformar cachorros de disponíveis para adotados, ganhando uma propriedade relativa ao novo dono
7. Alterar os dados de todos os documentos

Lembre-se:

- Siga o padrão HTTP: GET, POST, PUT, DELETE
- É possível testar o seu programa com um REST Client (disponíveis como extensões para Browsers)
- Tente ser stateless (não guarde informações do usuário)
- Busque retornar um formato portátil como JSON
- Procure reutilizar URLs para diferentes tipos de requisições (e.g.: POST em http://localhost/dogs/ ≠ GET em http://localhost/dogs/)