

@MongoDB + @javascript = @mongoosejs

Jesse Javier Cogollo Alvarez

Developer by passion

twitter: @jessecogollo

March 11, 2015

Contenido

MongoDB

Javascript

Mongoose

Que es @MongoDB

'MongoDB (from "humongous") is an open-source document database, and the leading NoSQL database. Written in C++.'

<https://www.mongodb.org/>

'MongoDB was not designed in a lab. We built MongoDB from our own experiences building large-scale, high availability, robust systems...' [Eliot Horowitz, CTO and Co-Founder](#)

Que es @MongoDB

'MongoDB (from "humongous") is an open-source document database, and the leading NoSQL database. Written in C++.'

<https://www.mongodb.org/>

'MongoDB was not designed in a lab. We built MongoDB from our own experiences building large-scale, high availability, robust systems...' [Eliot Horowitz, CTO and Co-Founder](#)

NOSQL

En informática, NoSQL (a veces llamado 'no sólo SQL') es una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico del sistema de gestión de bases de datos relacionales (RDBMS) en aspectos importantes, el más destacado que no usan SQL como el principal lenguaje de consultas.

<http://es.wikipedia.org/wiki/NoSQL/>

NOSQL

Las características comunes de las bases de datos NoSQL son:

- No utilizan el modelo relacional.
- Corren bien en clusters.
- Open-source.
- sin esquemas.
- El resultado mas importante del aumento de las bases de datos NoSQL es la **Persistencia Polígloa**.

[http:](http://martinfowler.com/articles/nosqlKeyPoints.html)

[//martinfowler.com/articles/nosqlKeyPoints.html](http://martinfowler.com/articles/nosqlKeyPoints.html)

NOSQL

Las características comunes de las bases de datos NoSQL son:

- No utilizan el modelo relacional.
- Corren bien en clusters.
- Open-source.
- sin esquemas.
- El resultado mas importante del aumento de las bases de datos NoSQL es la **Persistencia Polígloa**.

<http://martinfowler.com/articles/nosqlKeyPoints.html>

NOSQL

Las características comunes de las bases de datos NoSQL son:

- No utilizan el modelo relacional.
- Corren bien en clusters.
- Open-source.
- sin esquemas.
- El resultado mas importante del aumento de las bases de datos NoSQL es la **Persistencia Polígloa**.

<http://martinfowler.com/articles/nosqlKeyPoints.html>

NOSQL

Las características comunes de las bases de datos NoSQL son:

- No utilizan el modelo relacional.
- Corren bien en clusters.
- Open-source.
- sin esquemas.
- El resultado mas importante del aumento de las bases de datos NoSQL es la **Persistencia Polígloa**.

<http://martinfowler.com/articles/nosqlKeyPoints.html>

NOSQL

Las características comunes de las bases de datos NoSQL son:

- No utilizan el modelo relacional.
- Corren bien en clusters.
- Open-source.
- sin esquemas.
- El resultado mas importante del aumento de las bases de datos NoSQL es la **Persistencia Polígloa**.

[http:](http://martinfowler.com/articles/nosqlKeyPoints.html)

[//martinfowler.com/articles/nosqlKeyPoints.html](http://martinfowler.com/articles/nosqlKeyPoints.html)

Persistencia políglota



Caracteristicas

1. Document-Oriented Storage
2. Full Index Support
3. Replication
 - MMS.
4. Auto Sharding
 - Partner with MongoDB.
 - Multiples drivers.
5. Querying
6. Map Reduce
7. GridFS
8. **Other more...**

Caracteristicas

1. Document-Oriented Storage
2. Full Index Support
3. Replication
 - MMS.
4. Auto Sharding
 - Partner with MongoDB.
5. Querying
 - Multiples drivers.
6. Map Reduce
7. GridFS
8. **Other more...**

Caracteristicas

1. Document-Oriented Storage
2. Full Index Support
3. Replication
 - MMS.
4. Auto Sharding
 - Partner with MongoDB.
5. Querying
 - Multiples drivers.
6. Map Reduce
7. GridFS
8. **Other more...**

Caracteristicas

1. Document-Oriented Storage
2. Full Index Support
3. Replication
 - MMS.
 - Partner with MongoDB.
 - Multiples drivers.
4. Auto Sharding
5. Querying
6. Map Reduce
7. GridFS
8. **Other more...**

Caracteristicas

1. Document-Oriented Storage
2. Full Index Support
3. Replication
 - MMS.
4. Auto Sharding
 - Partner with MongoDB.
 - Multiples drivers.
5. Querying
6. Map Reduce
7. GridFS
8. **Other more...**

Caracteristicas

1. Document-Oriented Storage
2. Full Index Support
3. Replication
 - MMS.
4. Auto Sharding
 - Partner with MongoDB.
 - Multiples drivers.
5. Querying
6. Map Reduce
7. GridFS
8. **Other more...**

Caracteristicas

1. Document-Oriented Storage
2. Full Index Support
3. Replication
 - MMS.
4. Auto Sharding
 - Partner with MongoDB.
5. Querying
 - Multiples drivers.
6. Map Reduce
7. GridFS
8. **Other more...**

Caracteristicas

1. Document-Oriented Storage
2. Full Index Support
3. Replication
 - MMS.
4. Auto Sharding
 - Partner with MongoDB.
5. Querying
 - Multiples drivers.
6. Map Reduce
7. GridFS
8. Other more...

Caracteristicas

1. Document-Oriented Storage
2. Full Index Support
3. Replication
 - MMS.
4. Auto Sharding
 - Partner with MongoDB.
 - Multiples drivers.
5. Querying
6. Map Reduce
7. GridFS
8. **Other more...**

Insert Find Update Remove (CRUD)

IFUR

```
db.meetups.insert({ "name": "mongoosejs", "place": "Ruta N" })
```

IFUR

```
db.meetups.find({ "name": "mongoosejs" })
```

IFUR

```
db.meetups.update({ "name": "mongoosejs" },  
{$set: { "description": "Ruta N, piso 0." }})
```

IFUR

```
db.meetups.remove({ "name": "mongoosejs" })
```

Insert Find Update Remove (CRUD)

IFUR

```
db.meetups.insert({ "name": "mongoosejs", "place": "Ruta N" })
```

IFUR

```
db.meetups.find({ "name": "mongoosejs" })
```

IFUR

```
db.meetups.update({ "name": "mongoosejs" },  
{$set: { "description": "Ruta N, piso 0." }})
```

IFUR

```
db.meetups.remove({ "name": "mongoosejs" })
```

Insert Find Update Remove (CRUD)

IFUR

```
db.meetups.insert({ "name": "mongoosejs", "place": "Ruta N" })
```

IFUR

```
db.meetups.find({ "name": "mongoosejs" })
```

IFUR

```
db.meetups.update({ "name": "mongoosejs" },  
  { $set: { "description": "Ruta N, piso 0." } })
```

IFUR

```
db.meetups.remove({ "name": "mongoosejs" })
```

Insert Find Update Remove (CRUD)

IFUR

```
db.meetups.insert({ "name": "mongoosejs", "place": "Ruta N" })
```

IFUR

```
db.meetups.find({ "name": "mongoosejs" })
```

IFUR

```
db.meetups.update({ "name": "mongoosejs" },  
{$set: { "description": "Ruta N, piso 0." }})
```

IFUR

```
db.meetups.remove({ "name": "mongoosejs" })
```


Insert Find Update Remove (CRUD)

IFUR

```
db.meetups.insert({ "name": "mongoosejs", "place": "Ruta N" })
```

IFUR

```
db.meetups.find({ "name": "mongoosejs" })
```

IFUR

```
db.meetups.update({ "name": "mongoosejs" },  
{$set: { "description": "Ruta N, piso 0." }})
```

IFUR

```
db.meetups.remove({ "name": "mongoosejs" })
```

NodeJS - IOJS

NodeJS es una plataforma de javascript construida sobre el "motor" V8 de Chrome. <https://nodejs.org//>

IOJS es un fork de NodeJS. Implementando ES6 y desarrollado bajo un modelo de gobierno abierto. <https://iojs.org//>

NodeJS - IOJS

NodeJS es una plataforma de javascript construida sobre el "motor" V8 de Chrome. <https://nodejs.org//>

IOJS es un fork de NodeJS. Implementando ES6 y desarrollado bajo un modelo de gobierno abierto. <https://iojs.org//>

Mongoose

Un driver de MongoDB con NodeJS

instalación (En el directorio del proyecto.)

```
# npm install mongoose --save
```

Mongoose

Un driver de MongoDB con NodeJS

instalación (En el directorio del proyecto.)

```
# npm install mongoose --save
```

Entendiendo mongoosejs

Esquema

Un esquema es mapeado como una colección en MongoDB y define la forma de los documentos con esa colección.

Schema

```
1 var mongoose = require('mongoose');
2 var schema = mongoose.schema;
3 var userSchema = Schema({
4   firstName:String,
5   lastName:String,
6   telefones:{
7     primary:String,
8     secondary:String
9   },
10  hobbies:Array
11 });
```

Tipos: Number, Date, Buffer, Boolean, mixed and ObjectId.

Entendiendo mongoosejs

modelos

Un modelo es un constructor compilado de nuestra definición de esquema. y representan los documentos que pueden ser guardados y recuperados de la base de datos.

Model

```
1 var User = mongoose.model('User',userSchema);
```

Ya tenemos a User listo para Insertar, encontrar, actualizar y eliminar. Pero además, le podemos crear nuestras propias acciones al schema.

```
1 userSchema.methods.findSimilarLastNames = function (cb
  ) {
2   return this.model('User').find({ lastName: this.
      lastName }, cb);
3 }
```

Metodos estaticos

Statics

Agregar metodos estaticos a un modelo es simple.

```
1 userSchema.statics.findByName = function (name, cb) {  
2   this.find({ firstName: new RegExp(name, 'i') }, cb);  
3 }
```

Indexes

Los índices se pueden definir a nivel de esquema o a nivel de campo.

Campo

```
1 firstName:{type:String, index:true}
```

Esquema

```
1 animalSchema.index({firstName:1});
```

Comunidad

1. Meetup: medellinjs - mongodbmedellin
2. Twitter: @medellinjs - @mongodbmedellin
3. Facebook: /mongodbmedellin
4. gitter (chat): <https://gitter.im/coljs/medellinjs>
5. gitter (chat): <https://gitter.im/MongoDBMedellin/Meetup>

Comunidad

1. Meetup: medellinjs - mongodbmedellin
2. Twitter: @medellinjs - @mongodbmedellin
3. Facebook: /mongodbmedellin
4. gitter (chat): <https://gitter.im/coljs/medellinjs>
5. gitter (chat): <https://gitter.im/MongoDBMedellin/Meetup>

Comunidad

1. Meetup: medellinjs - mongodbmedellin
2. Twitter: @medellinjs - @mongodbmedellin
3. Facebook: /mongodbmedellin
4. gitter (chat): <https://gitter.im/coljs/medellinjs>
5. gitter (chat): <https://gitter.im/MongoDBMedellin/Meetup>

Comunidad

1. Meetup: medellinjs - mongodbmedellin
2. Twitter: @medellinjs - @mongodbmedellin
3. Facebook: /mongodbmedellin
4. gitter (chat): <https://gitter.im/coljs/medellinjs>
5. gitter (chat): <https://gitter.im/MongoDBMedellin/Meetup>

Comunidad

1. Meetup: medellinjs - mongodbmedellin
2. Twitter: @medellinjs - @mongodbmedellin
3. Facebook: /mongodbmedellin
4. gitter (chat): <https://gitter.im/coljs/medellinjs>
5. gitter (chat): <https://gitter.im/MongoDBMedellin/Meetup>

Donde aprender



javascript (nodeJS)

<http://nodeschool.io/>



MongoDB

<https://university.mongodb.com/>

Preguntas

Gracias !!! =)