БИБЛИОТЕКА MONGOOSE

УСТАНОВКА

Установка:

```
$ npm install mongoose
```

Подключение:

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/test');
```

Использование:

```
const db = mongoose.connection;
db.on('error', console.error.bind(console, 'connection error:'));
db.once('open', function() {
  // we're connected!
});
```

OBJECT-DOCUMENT MAPPER

ORM - Object-Relational Mapping (объектно-реляционное отображение) технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных»

ODM – Object-Document Mapper (объектно-документное отображение)

СХЕМА

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
const BlogSchema = new Schema({
    title: {
      type: String,
      required: [true, 'Укажите заголовок статьи']
    },
    body: {
      type: String,
      required: [true, 'Укажите содержимое статьи']
    },
    date: {
      type: Date,
      default: Date.now,
      required: [true, 'Укажите дату публикации']
    }
  });
mongoose.model('blog', BlogSchema);
```

СХЕМА

```
{
    "_id" : ObjectId("591c92ab0b99e824b8895463"),
    "title" : "Наша первая",
    "body" : "Лучшая статья",
    "date" : ISODate("2017-05-17T00:00:00.000+0000"),
    "__v" : NumberInt(0)
}
{
    "_id" : ObjectId("591c92d50b99e824b8895464"),
    "title" : "Вторая",
    "body" : "У меня ДР",
    "date" : ISODate("2017-01-30T00:00:00.000+0000"),
    "__v" : NumberInt(0)
}
```

ТИПЫ СХЕМ

- **String** — любая строка в кодировке UTF-8;
- **Number** — Mongoose не поддерживает длинные числа или числа с двойной точностью, но допускает расширение для их поддержки с помощью плагинов Mongoose. Поддерживаемого по умолчанию типа достаточно в большинстве случаев;
- **Date** — обычно возвращается из MongoDB в виде объекта ISODate;
- **Boolean** — true или false;
- **Buffer** — для двоичной информации, например изображений;
- **Mixed** — любой тип данных;
- **Array** — может быть или массивом данных соответствующего типа, или массивом вложенных поддокументов;
- **ObjectId** — для уникального ID в пути, отличном от _id. Обычно используется для ссылок на пути _id в других документах.

ПОИСК ЗАПИСЕЙ

```
const Person = mongoose.model('Person', yourSchema);

Person.findOne({ 'name.last': 'Ghost' }, 'name occupation',
  function (err, person) {
    if (err) return handleError(err);
    console.log(person.name.first, person.name.last,person.occupation)
})

Person.find({ 'name.last': 'Ghost' }, 'name occupation',
  function (err, docs) {}
)

Person.find({ 'name.last': 'Ghost' }, 'name occupation')
  .exec(function (err, docs) {})
```

СОЗДАНИЕ ЗАПИСЕЙ

```
const Person = mongoose.model('Person', yourSchema);

const subject = new Person({name: 'Marina'});

// the first way
subject.save(function(err) {
      if (err) return handleError(err);
})

// the second way
Person.create({name: 'Marina'}, function(err, subject) {
      if (err) return handkeError(err);
})
```

РЕДАКТИРОВАНИЕ ЗАПИСЕЙ

```
const query = {name: 'Marina'};


Model.update(query, {name: 'Sveta'}, options, callback);
Model.update(query, {$set: {name: 'Sveta'}}, options, callback);
Person.findOne({name: 'Sveta'}, function (err, person) {
  if (err) return handleError(err);
  person.name = 'Sveta';
  person.save();
})
```

УДАЛЕНИЕ ЗАПИСЕЙ

```
Model.remove({name: 'Sveta'}, function (err, person) {
  if (err) return handleError(err);
})
```