4. MongoDB & Mongoose.js

Завантажуємо і встановлюємо MongoDB. Встановлюємо mongoose: npm i mongoose. Роботу з бд я виведемо в файл libs/mongoose.js.

```
var mongoose = require('mongoose');
var log = require('./log')(module);
mongoose.connect('mongodb://localhost/test1');
var db = mongoose.connection;
db.on('error', function (err) {
    log.error('connection error:', err.message);
});
db.once('open', function callback () {
    log.info("Connected to DB!");
});
var Schema = mongoose.Schema; // Schemas
var Images = new Schema({
    kind: {
        type: String,
        enum: ['thumbnail', 'detail'],
        required: true },
        url: { type: String, required: true }
});
var Article = new Schema({
    title: { type: String, required: true },
    author: { type: String, required: true },
    description: { type: String, required: true },
    images: [Images],
    modified: { type: Date, default: Date.now }
}); // validation
Article.path('title').validate(function (v) {
    return v.length > 5 && v.length < 70;
});
var ArticleModel = mongoose.model('Article', Article);
module.exports.ArticleModel = ArticleModel;
```

В даному файлі виконується підключення до бази, а так же оголошуються схеми об'єктів. Статті будуть містити об'єкти картинок. Різноманітні складні валідації можна описати так само тут.

На даному етапі підключаємо модуль nconf для зберігання шляху до БД в ньому. Так само в конфіг збережемо порт, по якому створюється сервер. Модуль встановлюється командою npm і nconf. Обгорткою буде libs/config.js

```
var nconf = require('nconf');
nconf.argv()
   .env()
   .file({ file: './config.json' });
module.exports = nconf;
```

Створюємо config.json в корінні проекта.

```
{
   "port" : 1337,
   "mongoose": {
      "uri": "mongodb://localhost/test1"
   }
}
```

Зміни mongoose.js (тільки в шапці):

```
var config = require('./config');
mongoose.connect(config.get('mongoose:uri'));
```

Зміни server.js:

```
var config = require('./libs/config');
app.listen(config.get('port'), function(){
      log.info('Express server listening on port ' + config.get('port'));
});
```

Тепер додамо CRUD actions в наші існуючі шляхи.

```
var log = require('./libs/log')(module);
```

```
var ArticleModel = require('./libs/mongoose').ArticleModel;
app.get('/api/articles', function(req, res) {
    return ArticleModel.find(function (err, articles) {
        if (!err) {
            return res.send(articles);
        } else {
            res.statusCode = 500;
            log.error('Internal error(%d): %s', res.statusCode,err.message);
            return res.send({ error: 'Server error' });
        }
    });
});
app.post('/api/articles', function(req, res) {
    var article = new ArticleModel({
        title: req.body.title,
        author: req.body.author,
        description: req.body.description,
        images: req.body.images
    });
    article.save(function (err) {
        if (!err) {
            log.info("article created");
            return res.send({
                status: 'OK',
                article:article
            });
        } else {
            console.log(err);
            if(err.name == 'ValidationError') {
                res.statusCode = 400;
                res.send({ error: 'Validation error' });
            } else {
                res.statusCode = 500;
                res.send({ error: 'Server error' });
```

```
                }
                log.error('Internal error(%d): %s',res.statusCode,err.message);
        }
    });
});
app.get('/api/articles/:id', function(req, res) {
    return ArticleModel.findById(req.params.id, function (err, article) {
        if(!article) {
            res.statusCode = 404;
            return res.send({ error: 'Not found' });
        }
        if (!err) {
            return res.send({ status: 'OK', article:article });
        } else {
            res.statusCode = 500;
            log.error('Internal error(%d): %s',res.statusCode,err.message);
            return res.send({ error: 'Server error' });
        }
    });
});
app.put('/api/articles/:id', function (req, res){
    return ArticleModel.findById(req.params.id, function (err, article) {
        if(!article) {
            res.statusCode = 404;
            return res.send({ error: 'Not found' });
        }
    article.title = req.body.title;
    article.description = req.body.description;
    article.author = req.body.author;
    article.images = req.body.images;
    return article.save(function (err) {
        if (!err) {
            log.info("article updated");
            return res.send({ status: 'OK', article:article });
        } else {
```

```
                if(err.name == 'ValidationError') {
                        res.statusCode = 400;
                        res.send({ error: 'Validation error' });
                    } else {
                        res.statusCode = 500;
                        res.send({ error: 'Server error' });
                    }
                    log.error('Internal error(%d): %s',res.statusCode,err.message);
                }
            });
      });
 });
 app.delete('/api/articles/:id', function (req, res){
      return ArticleModel.findById(req.params.id, function (err, article) {
            if(!article) {
                res.statusCode = 404;
                return res.send({ error: 'Not found' });
            }
            return article.remove(function (err) {
                if (!err) {
                    log.info("article removed");
                    return res.send({ status: 'OK' });
                } else {
                    res.statusCode = 500;
                    log.error('Internal error(%d): %s',res.statusCode,err.message);
                    return res.send({ error: 'Server error' });
                }
            });
      });
 });
```