**Coloratsk\_1**

https://www.youtube.com/watch?v=m5ribwPpIPw

1. npm init - create a package.json

2. npm i express

3. touch server.js - создание файла server.js

4. Запуск - node server

5. Пользователи находятся в коллекции **system.users (пока не будем их туда записывать)**

6. /models/user.js - модель в которой хранятся функции создания/извлечения пользователя из БД

7. /routes/ - тут храним маршрутизацию для разных страниц. Ее помогает осуществить **express**. Используется **res.render** **().**  Во всех файлах в папке /routes/ должна быть строка **var express = require('express');**

/routes/**users.js -** хранит валидацию при авторизации и регистрации, а также стратегию для **passport** и маршрутизацию на **express.** К каждому роуту назначено свое view (**views/layouts/filename.handlebars** или **views/filename.handlebars**)

Почему стоит использовать mongoose наряду с mongojs:

I moved really fast with mongojs, but then I hit a wall. Joins. Is just not possible to do it with MongoJS for the same reason you can’t do it with MongoDB, you have to use some sort of black magic to do them, and this is when Mongoose excells.

Bottom line. If your project is really simple, no joins, no complicated features, go with MongoJS, is really easy but limited. If you’re trying to save the world with your crazy idea and you need more powers than Superman, spend some time learning Mongoose and use it, it will take you there.

If you are going to use MongoDB as a database with Node.js in your application then you will use Mongoose or MongoJS to perform database related operations like connecting to MongoDB database, performing CRUD operations and many more database operations.

**Mongoose:** is a Node.js library that provides MongoDB object mapping similar to ORM with a familiar interface within Node.js. Mongoose is a great ODM(Object Document Mapping), it means Mongoose translate data in the database to JavaScript objects for use in your application.

Mongoose provides straight-forward, schema-based solutions to model your application data. It include built in type-casting, validation, query building,business logic hooks and more, out of the box

**MongoJS:** is a Node.js module for MongoDB that emulates official MongoDB API as much as possible.It wraps MongoDB native and its very easy to use.

(**http://stackoverflow.com/questions/16380007/why-do-i-need-mongoose-or-mongojs-for-node-js**)

Переменные, которые используются для **express handlebars** в **index.handlebars**

**app.locals**

The app.locals object has properties that are local variables within the application.

**// Global Vars**

**app.use(function (req, res, next) {**

**res.locals.success\_msg = req.flash('success\_msg');**

**res.locals.error\_msg = req.flash('error\_msg');**

**res.locals.error = req.flash('error');**

**res.locals.user = req.user || null;**

**next();**

**})**

**Нужно соблюдать порядок подключения пакетов**

**http://stackoverflow.com/questions/16781294/passport-js-passport-initialize-middleware-not-in-use**

Follow the example to avoid the out-of-order middleware hell that express makes it so easy to enter. Straight from the docs. Note how yours does not match this exactly.

app.configure(function() {

app.use(express.static('public'));

app.use(express.cookieParser());

app.use(express.bodyParser());

app.use(express.session({ secret: 'keyboard cat' }));

app.use(passport.initialize());

app.use(passport.session());

app.use(app.router);

});

Docs

1. cookieParser
2. session
3. passport.initialize
4. passport.session
5. app.router

**глобальные переменные** в app.use(function (req, res, next) {

res.locals.success\_msg = req.flash('success\_msg');

res.locals.error\_msg = req.flash('error\_msg');

res.locals.error = req.flash('error');

res.locals.user = req.user || null;

next();

}); **нужно ставить перед** app.use('/', routes);

Названия методов mongoose и MongoDB могут не совпадать. Напр., в mongoose отстутствует **findAndModify ().** Вместо него используют **findByIdAndUpdate()** или **findOneAndUpdate()**