santiq / bulletproof-nodejs

Implementation of a bulletproof node.js API https://softwareontheroad.com/ideal-n...

#nodejs #node #javascript #typescript #boilerplate #express #mongodb #mongoose #typedi #agendajs

63 commits | 3 branches | 0 releases | 11 contributors | MIT

Branch: master | New pull request | Create new file | Upload files | Find File | Clone or download

santiq Merge pull request #30 from Kuzmrom7/master ··· Latest commit 9c1cd2a on 5 Aug

.circleci	removed gitkeep	5 months ago
src	Merge pull request #30 from Kuzmrom7/master	last month
tests	Apply eslint and prettier rules	4 months ago
.editorconfig	Bulletproof node.js architecture	5 months ago
.env	Update .env	5 months ago
.eslintrc.js	Add eslint and prettier	4 months ago
.gitignore	Bulletproof node.js architecture	5 months ago
.prettierrc.js	Add eslint and prettier	4 months ago
LICENSE	Create LICENSE	5 months ago
Procfile	Bulletproof node.js architecture	5 months ago
README.md	Add loggin layer	3 months ago
jest.config.js	adds a basic test to let circleci build	5 months ago

nodemon.json	⚔️ Bulletproof node.js architecture	5 months ago
package-lock.json	add express-basic-auth	2 months ago
package.json	Added example for sending mail through mailgun	last month
tsconfig.json	fixed imports with *	last month

README.md

Bulletproof Node.js architecture

This is the example repository from the blog post 'Bulletproof node.js project architecture'

Please read the blog post in order to have a good understanding of the server architecture.

Also, I added lots of comments to the code that are not in the blog post, because they explain the implementation and the reason behind the choices of libraries and some personal opinions and some bad jokes.

The API by itself doesn't do anything fancy, it's just a user CRUD with authentication capabilities. Maybe we can transform this into something useful, a more advanced example, just open an issue and let's discuss the future of the repo.

Development

We use `node` version `10.15.0`

```
nvm install 10.15.0
```

```
nvm use 10.15.0
```

The first time, you will need to run

```
npm install
```

Then just start the server with

```
npm run start
```

It uses nodemon for livereloading :peace-fingers:

API Validation

By using celebrate the req.body schema becomes clary defined at route level, so even frontend devs can read what an API endpoint expects without need to writting a documentation that can get outdated quickly.

```
route.post('/signup',
 celebrate({
   body: Joi.object({
     name: Joi.string().required(),
     email: Joi.string().required(),
     password: Joi.string().required(),
   }),
 }),
 controller.signup)
```

Example error

```
{
 "errors": {
```

```
    "message": "child \"email\" fails because [\"email\" is required]"
  }
 }
```

Read more about celebrate here and the Joi validation API

Roadmap

- [x] API Validation layer (Celebrate+Joi)
- [ ] Unit tests examples
- [ ] Cluster mode
- [x] The logging *'layer'*
- [ ] Add ageda dashboard
- [x] Continuous integration with CircleCI 😍
- [ ] Deploys script and docs for AWS Elastic Beanstalk and Heroku
- [ ] Integration test with newman 😉
- [ ] Instructions on typescript debugging with VSCode

FAQ

Where should I put the FrontEnd code? Is this a good backend for Angular o React or Vue or *whatever* ?

It's not a good idea to have node.js serving static assets a.k.a the frontend

Also, I don't wanna take part in frontend frameworks wars 😅

Just use the frontend framework you like the most *or hate the less* it will work 😄

Don't you think you can add X layer to do Y? Why do you still use express if the Serverless Framework is better and it's more reliable?

I know this is not a perfect architecture but it's the most scalable that I know with less code and headache that I know.

It's meant for small startups or one-developer army projects.

I know if you start moving layers into another technology, you will end up with your business/domain logic into npm packages, your routing layer will be pure AWS Lambda functions and your data layer a combination of DynamoDB, Redis, maybe redshift, and Agolia.

Take a deep breath and go slowly, let the business grow and then scale up your product. You will need a team and talented developers anyway.