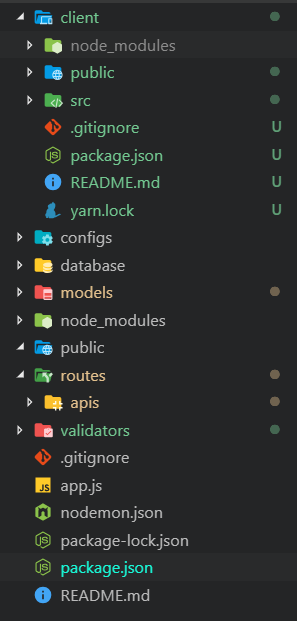
# Các bước xây dựng ứng dụng

* Phân tích database
* Tạo database
* Connect database
* Tạo hình ứng dụng bằng Schema ( mongoose )
* Create-react-app ngay trong project
* 
* Client chính là phần react-app

# Chạy 2 task cmd 1 lúc

* Sử dụng <https://www.npmjs.com/package/concurrently>
* Cài packet dạng global
* Npm I –g npm ( cập nhật npm )
* Vào file package.json của server sửa

"scripts": {

"client": "npm run start --prefix client",

"server": "nodemon --inspect app.js",

"dev": "concurrently \"yarn run server\" \"yarn run client\"",

"install": "npm install --prefix client"

}

Theo đúng thứ tự đó,

--prefix là vào folder đầu tiên tên client ( client là tên create-react-app )

Thiết lập xong npm run dev là OK

# Xử lý 2 task chạy trên 1 PORT

Server: PORT 9999

Client: PORT 3000

Khi dùng axios cần viết <http://localhost:9999> sẽ bị CORS, thay vào đó, ta cần vào package.json của client thêm vào

"proxy": "http://localhost:9999"

Proxy trỏ đến port của express

Khi đó axios.post(‘/data’) = <http://localhost:9999/data>

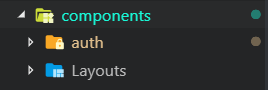
https://www.bravebits.co/ket-hop-create-react-app-voi-expressjs-server-trong-mot-not-nhac/

# Ngăn form submit

Bắt sự kiện onChange ở thẻ form chứ k bắt ở thẻ input

# Xử lý router

Trong file app.js đã wrap bằng thẻ Router



Nhưng khi gọi 1 component trong 1 component khác, ta sẽ không thể truy xuất được đối tượng this.props.history để chuyển hướng website

Xử lý:

import { withRouter } from "react-router-dom";

export default withRouter(Test);

# Gửi token kèm mọi request ( global token )

### Global axios defaults

axios.defaults.baseURL = 'https://api.example.com';

axios.defaults.headers.common['Authorization'] = AUTH\_TOKEN; // dùng cái này

axios.defaults.headers.post['Content-Type'] = 'application/x-www-form-urlencoded';

if (token) { setToken bằng hàm kia } else { delete axios.defaults.headers.common['Authorization'] = AUTH\_TOKEN; }

# jwt-token kết hợp passport

**const** jwt = require("jsonwebtoken");

jwt.sign(

payload,

jwtConfig.secret,

{ expiresIn: jwtConfig.expires },

(err, token) **=>** {

if (err) res.status(500).json({ token: "error create token" });

else {

res.json({

success: true,

token: "Bearer " + token

});

}

}

); *// expires : 1 hour*

Nếu login thành công sẽ tạo token

Kết hợp xác thực passport

Config passport

**const** JwtStrategy = require("passport-jwt").Strategy,

ExtractJwt = require("passport-jwt").ExtractJwt;

**const** User = require("../model/users");

**const** jwtConfig = require('./jwtToken');

**let** opts = {};

opts.secretOrKey = jwtConfig.secret;

opts.jwtFromRequest = ExtractJwt.fromAuthHeaderAsBearerToken();

module.exports = passport **=>** {

passport.use(

new JwtStrategy(opts, **function**(jwt\_payload, done) {

User.findById(jwt\_payload.id, **function**(err, user) {

if (err) {

return done(err, false);

}

if (user) {

return done(null, user);

} else {

return done(null, false);

}

});

})

);

};

Sử dụng config này trong app.js

*//passport midderware*

app.use(passport.initialize());

*//config passport*

require("./config/passportJWT")(passport);

# Chọn DOM và xử lý trên DOM ( init 1 số hiệu ứng )

<https://stackoverflow.com/questions/36339354/how-do-i-get-the-materialize-select-dropdown-to-work-with-react>

# Làm việc với file

## Clinet

* Config giới hạn uploaded

app.**use**(express.**json**({ limit: "2mb", extended: true }));

app.**use**(express.**urlencoded**({ limit: "2mb", extended: true }));

* Config trong axios

export const **registerApi** = *user* => {

const config = { headers: { "Content-Type": "multipart/form-data" } };

return new Promise((*resolve*, *reject*) => {

axios

.**post**("/test", user, config)

.**then**(*res* => **resolve**(res))

.**catch**(*err* => **reject**(err));

});

};

* Luồng dữ liệu gửi lên phải ở dạng formData

**submitAction** = *event* => {

event.**preventDefault**();

let { name, email, pwd, pwd1, sex, selectFile } = this.state;

let fd = new FormData();

fd.**append**("name", name);

fd.**append**("email", email);

fd.**append**("pwd", pwd);

fd.**append**("pwd1", pwd1);

fd.**append**("sex", sex);

fd.**append**("file", selectFile);

this.props.**register**(fd, this.props.history);

};

## Server

* Multer chỉ đóng vai trò đọc dữ liệu từ multipart/form-data và chuyển chúng thành req.file
* Nếu config như trên multer hướng dẫn sẽ không xác minh được đầu vào, thay vào đó ta sẽ dùng sharp để cover file về kích cỡ nhỏ hơn và lưu trực tiếp như sau

Roter điều hướng

const multer = require("multer");

const upload = **multer**({

limits: {

fileSize: 4 \* 1024 \* 1024

}

});

router.**post**("/register", upload.**single**("image"),checkValueRegister, async function(*req*, *res*) {

*// folder upload*

const imagePath = path.**join**(\_\_dirname, "../../public/uploads");

*// call class Resize*

const fileUpload = new Resize(imagePath);

if (!req.file) {

res.**status**(401).**json**({ error: "Please provide an image" });

}

const filename = await fileUpload.**save**(req.file.buffer);

return res.**status**(200).**json**({ name: filename });

});

* Multer chỉ là midderware lấy dữ liệu, nó sẽ đi qua checkValueRegister để lọc và thực hiện lưu bằng class Resize
* Resize:

const sharp = require("sharp");

const uuidv4 = require("uuid/v4");

const path = require("path");

class Resize {

constructor(*folder*) {

this.folder = folder;

}

async **save**(*buffer*) {

const filename = Resize.**filename**();

const filepath = this.**filepath**(filename);

await **sharp**(buffer)

.**resize**(300, 300, {

*// size image 300x300*

fit: sharp.fit.inside,

withoutEnlargement: true

})

.**toFile**(filepath);

return filename;

}

static **filename**() {

*// random file name*

return `${**uuidv4**()}.png`;

}

**filepath**(*filename*) {

return path.**resolve**(`${this.folder}/${filename}`);

}

}

module.exports = Resize;

source: <https://viblo.asia/p/node-express-image-upload-and-resize-Qbq5QMgG5D8>

# Xử lý ngày + ràng buộc dữ liệu giữa các document, xem file mongoose

# Custom css website looklike macos

<https://gist.github.com/spemer/a0e218bbb45433bd611e68446523a00b>

# Add material css

* Làm trong từng cmp

import M from "materialize-css/dist/js/materialize.min.js";

**componentDidMount**() {

M.**AutoInit**();

M.chips = {

autocompleteOptions: {

data: {

Apple: null,

Microsoft: null,

Google: null

},

limit: Infinity,

minLength: 1

}

};

}

Trừ AutoInit là func còn lại đều config ở dạng object

# Xử lý thẻ <Link> dưới dạng button

<Link *to*="/question">

<button

*className*="btn waves-effect waves-light red accent-3

"

*type*="submit"

*name*="action"

>

Hủy bỏ

<i *className*="material-icons right">do\_not\_disturb</i>

</button>

</Link>