1. Front end: sign up user

Dùng Reactive Form, FormGroup, FormBuilder, Validation để kiểm tra thông tin tạo người dùng

Dùng auth.service để chuyển thông tin người dùng đến back end

2. Back end: tạo dữ liệu người dùng

Tạo model user

Dùng Joi để kiểm tra dữ liệu tạo người dùng nếu sai:

const { error, value } = Joi.validate(req.body, schema);

    if (error && error.details) {

      return res.status(HttpStatus.BAD\_REQUEST).json({ msg: error.details });

Tạo luật helper: lowercase/uppercase đưa dữ liệu cần tạo về dạng chuẩn

Kiểm tra dữ liệu cần tạo đã có chưa:

const userEmail = await User.findOne({

      email: Helpers.lowerCase(req.body.email)

    });

    if (userEmail) {

      return res

        .status(HttpStatus.CONFLICT)

        .json({ message: "Email đã tồn tại!" });

    }

    const userName = await User.findOne({

      userName: Helpers.firstUpper(req.body.username)

    });

if (userName) {

      return res

        .status(HttpStatus.CONFLICT)

        .json({ message: "Username đã tồn tại!" });

    }

Nếu không tồn tại thì hash password:

return await bcrypt.hash(req.body.password, 10, (err, hash) => {

      if (err) {

        return res

          .status(HttpStatus.BAD\_REQUEST)

          .json({ message: "Lỗi hash password không thành công!" });

      }

Tạo dữ liệu để nạp vào database mongodb

const account = {

        username: Helpers.firstUpper(value.username),

        email: Helpers.lowerCase(value.email),

        password: hash

      };

Nạp dữ liệu đã tạo vào database và theo dõi thành công hay không ?

User.create(account)

        .then(user => {

          const token = jwt.sign({ data: account }, dbConfig.keysecret, {

            expiresIn: "1h"

          });

          res.cookie("auth", token);

          res

            .status(HttpStatus.CREATED)

            .json({ message: "Thêm thành công!", user, token });

        })

        .catch(err => {

          res

            .status(HttpStatus.INTERNAL\_SERVER\_ERROR)

            .json({ message: "Lỗi server!" });

        });

3.Front end: kiểm tra đăng nhập

Nếu đăng nhập hay mới tạo tài khoản thì vào trang chính, không cho vào trang login và tạo tài khoản

Nếu không có token thì đẩy về trang login hoặc tạo tài khoản: tạo token.service để lấy, tạo, xóa token và dùng auth.guard để kiểm tra token

canActivate(

    next: ActivatedRouteSnapshot,

    state: RouterStateSnapshot

  ): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {

    const token = this.tokenService.GetToken();

    if (token) {

      return true;

    } else {

      this.router.navigate(['/']);

      return false;

    }

  }

Mỗi lần req Backend thì phải thêm vào headers 1 chuỗi token mang tên authorization bằng HttpInterceptor

export class TokenInterceptorService implements HttpInterceptor {

  constructor(private tokenService: TokenService) {}

  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {

    const headersConfig = {

      'Content-Type': 'application/json',

      Accept: 'application/json'

    };

    const token = this.tokenService.GetToken();

    if (token) {

      headersConfig['authorization'] = `beader ${token}`;

    }

    const \_req = req.clone({ setHeaders: headersConfig });

    return next.handle(\_req);

  }

}

(Khai báo trong module để nạp thư viện HTTP Intercepter)

      provide: HTTP\_INTERCEPTORS,

      useClass: TokenInterceptorService,

      multi: true

4.Back end: Kiểm tra http headers đã được thêm thông tin do Intercepter thực hiện không ?

KIỂM TRA http headers: console.log(req.headers)

Kiểm tra đăng nhập

const jwt = require("jsonwebtoken");

const HttpStatus = require("http-status-codes");

const dbConfig = require("../config/link");

module.exports = {

  VerifyToken: (req, res, next) => {

    if (!req.headers.authorization) {

      return res

        .status(HttpStatus.UNAUTHORIZED)

        .json({ message: "Không có chứng nhận đăng nhập" });

    }

    const token = req.cookies.auth || req.headers.authorization.split(" ")[1];

    console.log(token);

    if (!token) {

      return res

        .status(HttpStatus.FORBIDDEN)

        .json({ message: "Không có thông tin đăng nhập" });

    }

    return jwt.verify(token, dbConfig.keysecret, (err, decoded) => {

      if (err) {

        if (err.expiredAt < new Date()) {

          return res.status(HttpStatus.INTERNAL\_SERVER\_ERROR).json({

            message:

              "Tài khoản hết thời gian đăng nhập, vui lòng đăng nhập lại!",

            token: null

          });

        }

        next();

      }

      req.user = decoded.data;

      next();

    });

  }

};

Và thực hiện kiểm tra từ router xem http request có quyền vượt qua middleware hay không ?