**10. 웹 API 만들기**

API(Application Programming Interface) : 다른 애플리케이션에서 현재 프로그램 기능을 사용할 수 있게 허용하는 접점

**3. JWT 토큰으로 인증하기**

JWT : JSON Web Token, JSON 형식의 데이터를 저장하는 토큰

헤더 : 토큰 종류와 해시 알고리즘 정보가 들어있다

페이로드 : 토큰의 내용물이 인코딩 된 부분이다.

시그니처 : 일련의 문자열로, 시그니처를 통해 토큰이 변조되었는지 여부를 확인 할 수 있다.

C:\materK1m\newmimac\test\_kes\nodebird2\_api>npm i jsonwebtoken

* .env

COOKIE\_SECRET=nodebirdsecret

KAKAO\_ID=18905b408ab7fd4ea90ac236b5404f93

JWT\_SECRET=jwtSecret

* routes/middlewares.js

const jwt = require('jsonwebtoken');

exports.isLoggedIn = (req, res, next) => {

if (req.isAuthenticated()) {

next();

} else {

res.status(403).send('로그인 필요');

}

};

exports.isNotLoggedIn = (req, res, next) => {

if (!req.isAuthenticated()) {

next();

} else {

res.redirect('/');

}

};

exports.verifyToken = (req, res, next) => {

try {

req.decoded = jwt.verify(req.headers.authorization, process.env.JWT\_SECRET);

return next();

} catch (error) {

if (error.name === 'TokenExpiredError') { // 유효기간 초과

return res.status(419).json({

code: 419,

message: '토큰이 만료되었습니다',

});

}

return res.status(401).json({

code: 401,

message: '유효하지 않은 토큰입니다',

});

}

};

* routes/v1.js

라우터는 기존 사용자에게 영향을 미치기 때문에 수정을 함부로 하면 안됨

const express = require('express');

const jwt = require('jsonwebtoken');

const { verifyToken } = require('./middlewares');

const { Domain, User, Post, Hashtag } = require('../models');

const router = express.Router();

router.post('/token', async (req, res) => {

const { clientSecret } = req.body;

try {

const domain = await Domain.find({

where: { clientSecret },

include: {

model: User,

attribute: ['nick', 'id'],

},

});

if (!domain) {

return res.status(401).json({

code: 401,

message: '등록되지 않은 도메인입니다. 먼저 도메인을 등록하세요',

});

}

const token = jwt.sign({

id: domain.user.id,

nick: domain.user.nick,

}, process.env.JWT\_SECRET, {

expiresIn: '1m', // 1분

issuer: 'nodebird',

});

return res.json({

code: 200,

message: '토큰이 발급되었습니다',

token,

});

} catch (error) {

console.error(error);

return res.status(500).json({

code: 500,

message: '서버 에러',

});

}

});

router.get('/test', verifyToken, (req, res) => {

res.json(req.decoded);

});

module.exports = router;

**4. 호출서버 만들기**

* app.js

const v1 = require('./routes/v1');

app.use('/v1', v1);

* views/error.pug

h1= message

h2= error.status

pre #{error.stack}

* .env

COOKIE\_SECRET=nodebirdsecret

CLIENT\_SECRET=2ad36296-5493-4e29-8c66-52ae01079dfa

* routes/index.js

const express = require('express');

const axios = require('axios');

const router = express.Router();

router.get('/test', async (req, res, next) => {

try {

if (!req.session.jwt) { // 세션에 토큰이 없으면

const tokenResult = await axios.post('http://localhost:8002/v1/token', {

clientSecret: process.env.CLIENT\_SECRET,

});

if (tokenResult.data && tokenResult.data.code === 200) { // 토큰 발급 성공

req.session.jwt = tokenResult.data.token; // 세션에 토큰 저장

} else { // 토큰 발급 실패

return res.json(tokenResult.data); // 발급 실패 사유 응답

}

}

// 발급받은 토큰 테스트

const result = await axios.get('http://localhost:8002/v1/test', {

headers: { authorization: req.session.jwt },

});

return res.json(result.data);

} catch (error) {

console.error(error);

if (error.response.status === 419) { // 토큰 만료 시

return res.json(error.response.data);

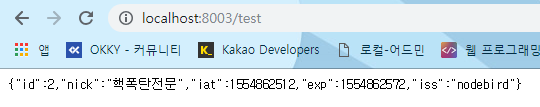
}

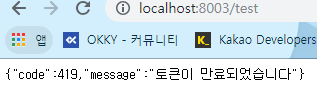
return next(error);

}

});

module.exports = router;





5. SNS API 서버 만들기

* nodebird2-api/routes/v1.js

const express = require('express');

const jwt = require('jsonwebtoken');

const { verifyToken } = require('./middlewares');

const { Domain, User, Post, Hashtag } = require('../models');

const router = express.Router();

router.post('/token', async (req, res) => {

const { clientSecret } = req.body;

try {

const domain = await Domain.find({

where: { clientSecret },

include: {

model: User,

attribute: ['nick', 'id'],

},

});

if (!domain) {

return res.status(401).json({

code: 401,

message: '등록되지 않은 도메인입니다. 먼저 도메인을 등록하세요',

});

}

const token = jwt.sign({

id: domain.user.id,

nick: domain.user.nick,

}, process.env.JWT\_SECRET, {

expiresIn: '1m', // 1분

issuer: 'nodebird',

});

return res.json({

code: 200,

message: '토큰이 발급되었습니다',

token,

});

} catch (error) {

console.error(error);

return res.status(500).json({

code: 500,

message: '서버 에러',

});

}

});

router.get('/test', verifyToken, (req, res) => {

res.json(req.decoded);

});

router.get('/posts/my', verifyToken, (req, res) => {

Post.findAll({ where: { userId: req.decoded.id } })

.then((posts) => {

console.log(posts);

res.json({

code: 200,

payload: posts,

});

})

.catch((error) => {

console.error(error);

return res.status(500).json({

code: 500,

message: '서버 에러',

});

});

});

router.get('/posts/hashtag/:title', verifyToken, async (req, res) => {

try {

const hashtag = await Hashtag.find({ where: { title: req.params.title } });

if (!hashtag) {

return res.status(404).json({

code: 404,

message: '검색 결과가 없습니다',

});

}

const posts = await hashtag.getPosts();

return res.json({

code: 200,

payload: posts,

});

} catch (error) {

console.error(error);

return res.status(500).json({

code: 500,

message: '서버 에러',

});

}

});

module.exports = router;

* nodebird2-call/routes/index.js

const express = require('express');

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

const { User, Domain } = require('../models');

const router = express.Router();

router.get('/', (req, res, next) => {

User.find({

where: { id: req.user && req.user.id },

include: { model: Domain },

})

.then((user) => {

res.render('login', {

user,

loginError: req.flash('loginError'),

domains: user && user.domains,

});

})

.catch((error) => {

next(error);

});

});

router.post('/domain', (req, res, next) => {

Domain.create({

userId: req.user.id,

host: req.body.host,

type: req.body.type,

clientSecret: uuidv4(),

})

.then(() => {

res.redirect('/');

})

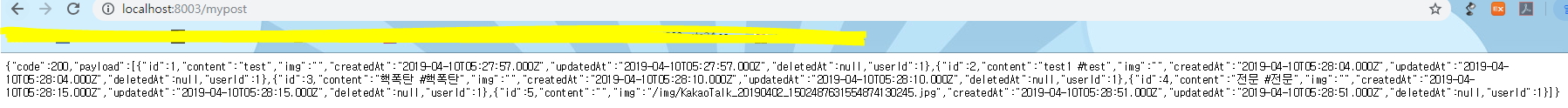
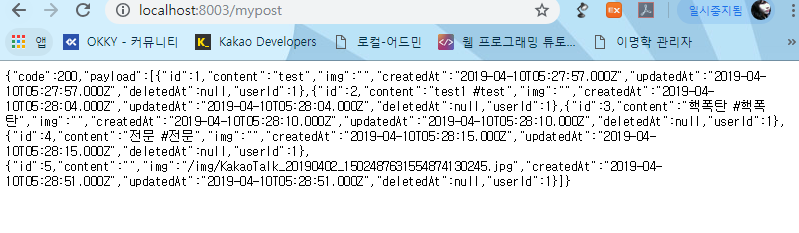
.catch((error) => {

next(error);

});

});

module.exports = router;



**6. 사용량 제한 구현하기**

C:\materK1m\newmimac\test\_kes\nodebird2\_api>npm i express-rate-limit

* nodebird-api/routes/middlewares.js

const jwt = require('jsonwebtoken');

const RateLimit = require('express-rate-limit');

exports.isLoggedIn = (req, res, next) => {

if (req.isAuthenticated()) {

next();

} else {

res.status(403).send('로그인 필요');

}

};

exports.isNotLoggedIn = (req, res, next) => {

if (!req.isAuthenticated()) {

next();

} else {

res.redirect('/');

}

};

exports.verifyToken = (req, res, next) => {

try {

req.decoded = jwt.verify(req.headers.authorization, process.env.JWT\_SECRET);

return next();

} catch (error) {

if (error.name === 'TokenExpiredError') { // 유효기간 초과

return res.status(419).json({

code: 419,

message: '토큰이 만료되었습니다',

});

}

return res.status(401).json({

code: 401,

message: '유효하지 않은 토큰입니다',

});

}

};

exports.apiLimiter = new RateLimit({

windowMs: 60 \* 1000, // 1분

max: 1,

delayMs: 0,

handler(req, res) {

res.status(this.statusCode).json({

code: this.statusCode, // 기본값 429

message: '1분에 한 번만 요청할 수 있습니다.',

});

},

});

exports.deprecated = (req, res) => {

res.status(410).json({

code: 410,

message: '새로운 버전이 나왔습니다. 새로운 버전을 사용하세요.',

});

};

[api 응답목록]

200 : JSON 데이터입니다

401 : 유효하지 않은 토큰입니다.

410 : 새로운 버전이 나왓습니다. 새로운 버전을 사용하세요.

419 : 토큰이 만료되었습니다.

429 : 1분에 한번만 요청할 수 있습니다.

* nodebird2-api/routes/v2.js

const express = require('express');

const jwt = require('jsonwebtoken');

const { verifyToken, apiLimiter } = require('./middlewares');

const { Domain, User, Post, Hashtag } = require('../models');

const router = express.Router();

router.post('/token', apiLimiter, async (req, res) => {

const { clientSecret } = req.body;

try {

const domain = await Domain.find({

where: { clientSecret },

include: {

model: User,

attribute: ['nick', 'id'],

},

});

if (!domain) {

return res.status(401).json({

code: 401,

message: '등록되지 않은 도메인입니다. 먼저 도메인을 등록하세요',

});

}

const token = jwt.sign({

id: domain.user.id,

nick: domain.user.nick,

}, process.env.JWT\_SECRET, {

expiresIn: '30m', // 30분

issuer: 'nodebird',

});

return res.json({

code: 200,

message: '토큰이 발급되었습니다',

token,

});

} catch (error) {

console.error(error);

return res.status(500).json({

code: 500,

message: '서버 에러',

});

}

});

router.get('/test', verifyToken, apiLimiter, (req, res) => {

res.json(req.decoded);

});

router.get('/posts/my', apiLimiter, verifyToken, (req, res) => {

Post.findAll({ where: { userId: req.decoded.id } })

.then((posts) => {

console.log(posts);

res.json({

code: 200,

payload: posts,

});

})

.catch((error) => {

console.error(error);

return res.status(500).json({

code: 500,

message: '서버 에러',

});

});

});

router.get('/posts/hashtag/:title', verifyToken, apiLimiter, async (req, res) => {

try {

const hashtag = await Hashtag.find({ where: { title: req.params.title } });

if (!hashtag) {

return res.status(404).json({

code: 404,

message: '검색 결과가 없습니다',

});

}

const posts = await hashtag.getPosts();

return res.json({

code: 200,

payload: posts,

});

} catch (error) {

console.error(error);

return res.status(500).json({

code: 500,

message: '서버 에러',

});

}

});

module.exports = router;

* nodebird-api/routes/v1.js

const express = require('express');

const jwt = require('jsonwebtoken');

const { verifyToken, deprecated } = require('./middlewares');

const { Domain, User, Post, Hashtag } = require('../models');

const router = express.Router();

router.use(deprecated);

router.post('/token', async (req, res) => {

…

* nodebird2-api/app.js

…

const v1 = require('./routes/v1');

const v2 = require('./routes/v2');

const app = express();

…

app.use('/v1', v1);

app.use('/v2', v2);

app.use('/auth', authRouter);

* nodebird2-call/routes/index.js

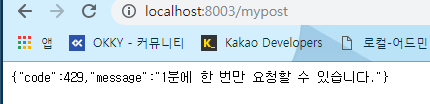
const express = require('express');

const axios = require('axios');

const router = express.Router();

const URL = 'http://localhost:8002/v2';

…



7. CORS 이해하기

* nodebird-call/routes/index.js

…

router.get('/', (req, res) => {

res.render('main', { key: process.env.CLIENT\_SECRET });

});

module.exports = router;

* nodebird2-call/views/main.pug

doctype

html

head

title 프론트 API 요청

body

#result

script.

var xhr = new XMLHttpRequest();

xhr.onreadystatechange = function () {

if (xhr.readyState === xhr.DONE) {

if (xhr.status === 200) {

document.querySelector('#result').textContent = xhr.responseText;

} else {

console.error(xhr.responseText);

}

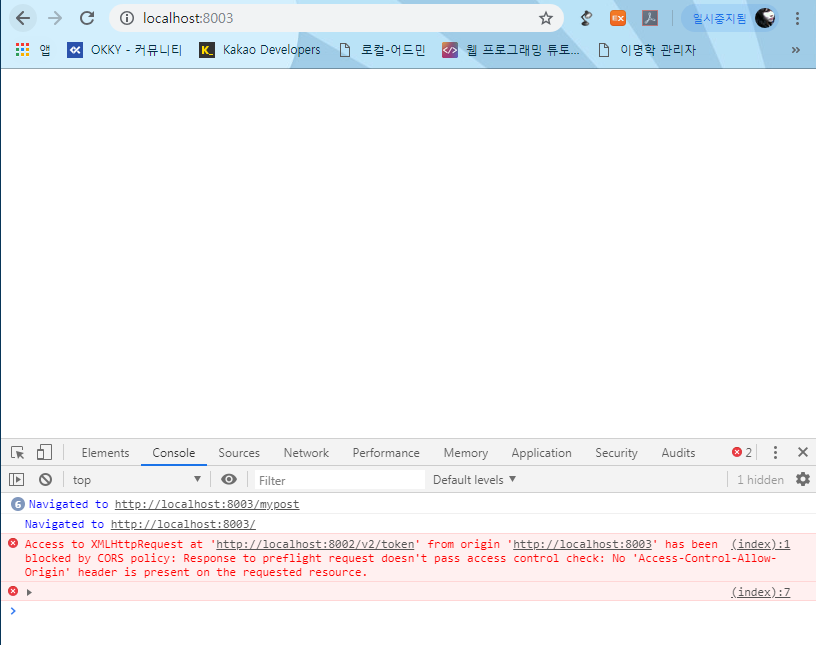
}

};

xhr.open('POST', 'http://localhost:8002/v2/token');

xhr.setRequestHeader('Content-Type', 'application/json');

xhr.send(JSON.stringify({ clientSecret: '#{key}' }));



C:\materK1m\newmimac\test\_kes\nodebird2\_api>npm i cors

* nodebird2\_api/routes/v2.js

const express = require('express');

const jwt = require('jsonwebtoken');

const cors = require('cors');

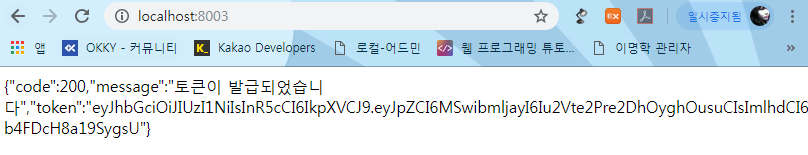
const { verifyToken, apiLimiter } = require('./middlewares');

const { Domain, User, Post, Hashtag } = require('../models');

const router = express.Router();

router.use(cors());

…



* routes/v2.js

호스트와 비밀키가 모두 일치할 때만 CORS를 허용하게 수정

const express = require('express');

const jwt = require('jsonwebtoken');

const cors = require('cors');

const url = require('url');

const { verifyToken, apiLimiter } = require('./middlewares');

const { Domain, User, Post, Hashtag } = require('../models');

const router = express.Router();

router.use(async (req, res, next) => {

const domain = await Domain.find({

where: { host: url.parse(req.get('origin')).host },

});

if (domain) {

cors({ origin: req.get('origin') })(req, res, next);

} else {

next();

}

});

router.post('/token', apiLimiter, async (req, res) => {

