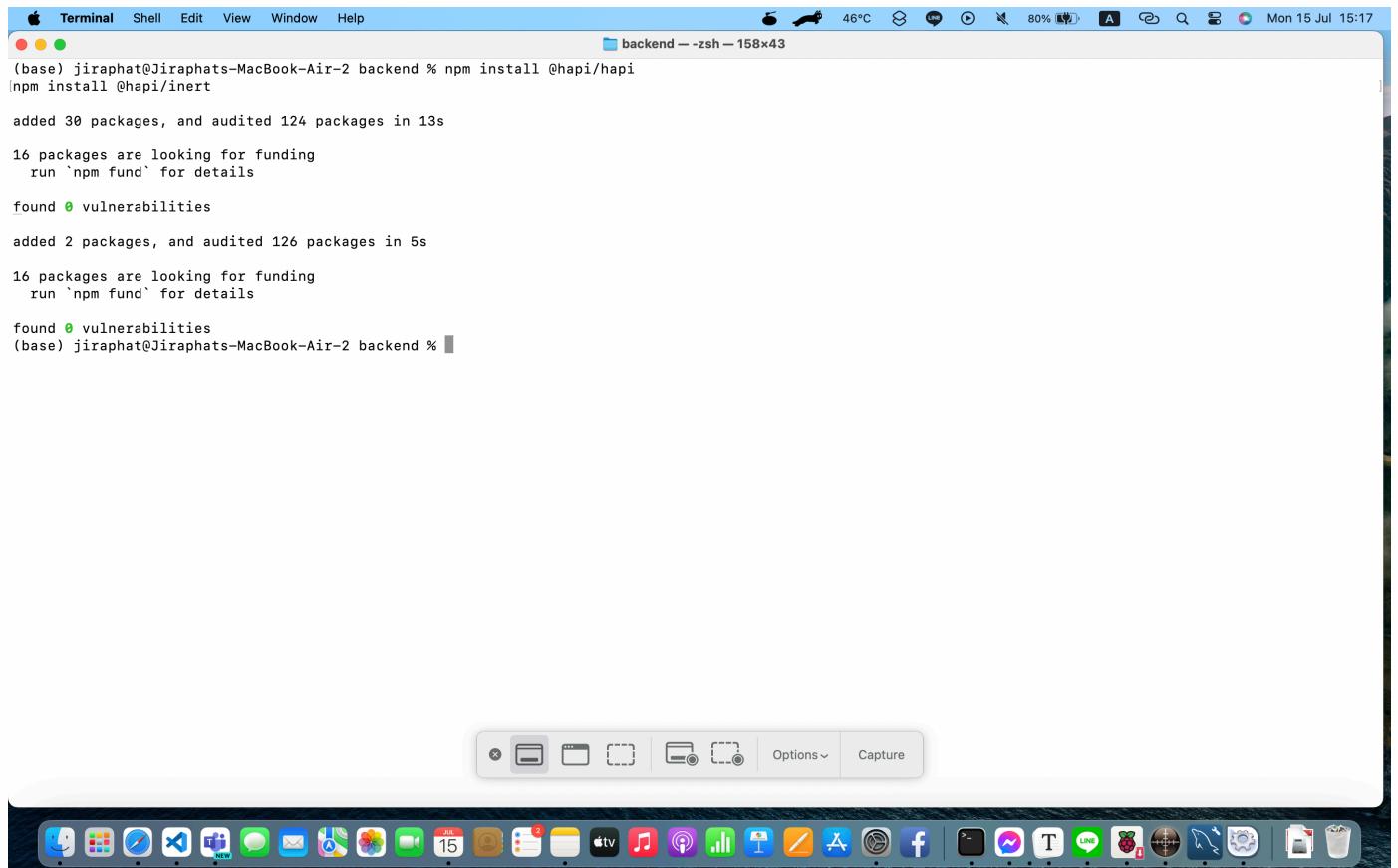


# Lab3 ResfulAPI Server with Nodejs

ติดตั้ง hapi API package

```
npm install @hapi/hapi
npm install @hapi/inert
```



```
Terminal Shell Edit View Window Help
(base) jiraphat@Jiraphats-MacBook-Air-2 backend % npm install @hapi/hapi
|npm install @hapi/inert

added 30 packages, and audited 124 packages in 13s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

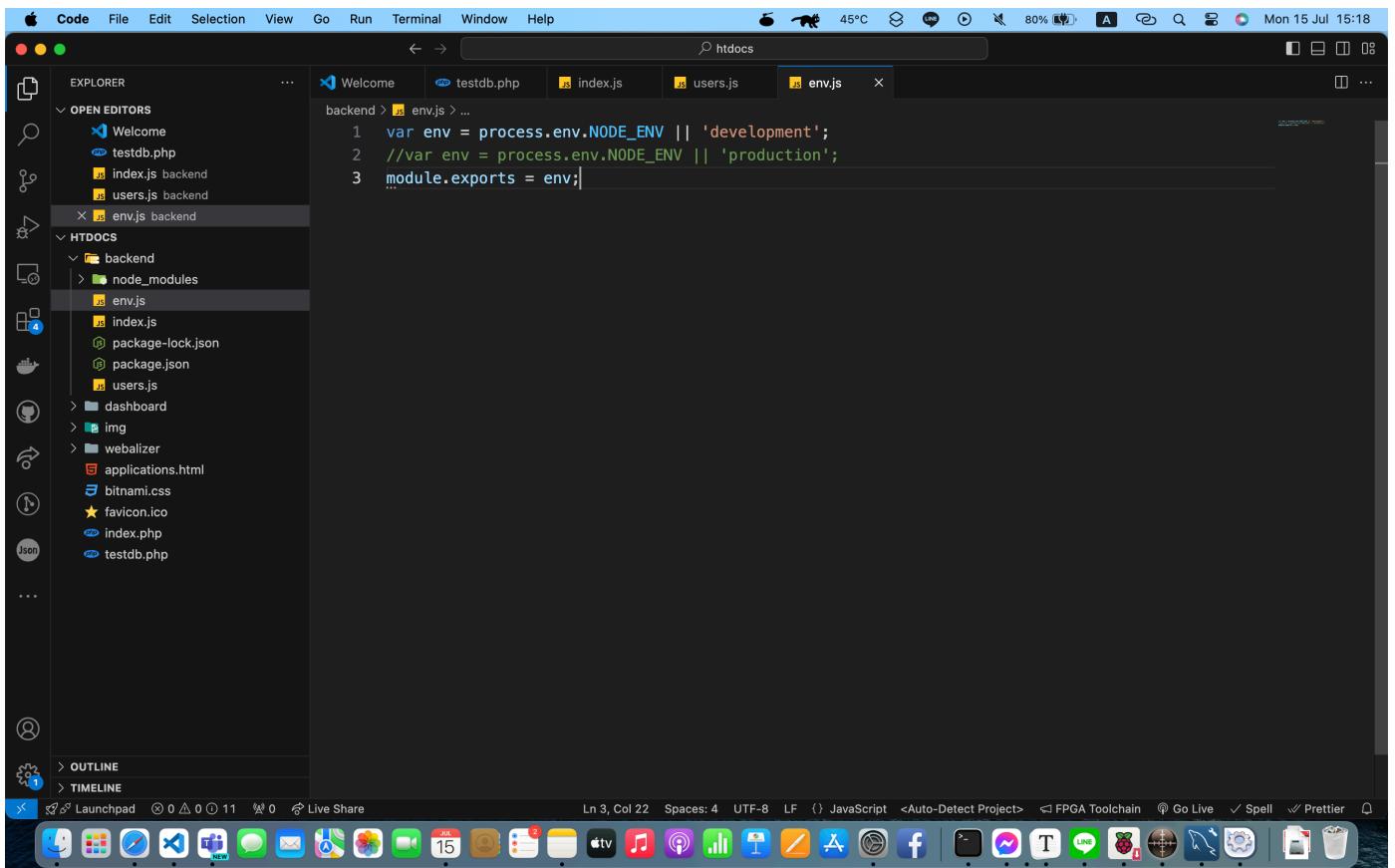
added 2 packages, and audited 126 packages in 5s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
(base) jiraphat@Jiraphats-MacBook-Air-2 backend %
```

สร้างไฟล์ env.js เพื่อเก็บ config ว่าทำงานอยู่ในสภาพแวดล้อมใด

```
var env = process.env.NODE_ENV || 'development';
//var env = process.env.NODE_ENV || 'production';
module.exports = env;
```



ปรับปรุง index.js

```
const hapi = require('@hapi/hapi');
var users = require('./users');
const env = require('./env.js');
const Movies = require('./repository/movie');

const express = require('express');
const app = express();

//-----
const web_port = 3010;
const api_port = 3001;

//----- express -----
app.get('/', (req, res) => {
  res.send('<h1> Hello World main_server </h1>');
})

app.get('/users', function (req, res) {
  res.json(users.findAll());
});

app.get('/user/:id', function (req, res) {
  var id = req.params.id;
  res.json(users.findById(id));
});
```

```

app.listen(web_port, () => {
  console.log('Start web server at port ' + web_port);
})

//----- hapi -----
console.log('Running Environment: ' + env);

const init = async () => {

  const server = hapi.Server({
    port: api_port,
    host: '0.0.0.0',
    routes: {
      cors: true
    }
  });
  //-----

  await server.register(require('@hapi/inert'));

  server.route({
    method: "GET",
    path: "/",
    handler: () => {
      return '<h3> Welcome to API Back-end Ver. 1.0.0</h3>';
    }
  });

  //API: http://localhost:3001/api/movie/all
  server.route({
    method: 'GET',
    path: '/api/movie/all',
    config: {
      cors: {
        origin: ['*'],
        additionalHeaders: ['cache-control', 'x-requested-width']
      }
    },
    handler: async function (request, reply) {
      //var param = request.query;
      //const category_code = param.category_code;

      try {

        const responsedata = await Movies.MovieRepo.getMovieList();
        if (responsedata.error) {
          return responsedata.errorMessage;
        } else {
          return responsedata;
        }
      } catch (err) {

```

```

        server.log(["error", "home"], err);
        return err;
    }

})

server.route({
    method: 'GET',
    path: '/api/movie/search',
    config: {
        cors: {
            origin: ['*'],
            additionalHeaders: ['cache-control', 'x-requested-width']
        }
    },
    handler: async function (request, reply) {
        var param = request.query;
        const search_text = param.search_text;
        //const title = param.title;

        try {

            const responddata = await Movies.MovieRepo.getMovieSearch(search_text);
            if (responddata.error) {
                return responddata.errorMessage;
            } else {
                return responddata;
            }
        } catch (err) {
            server.log(["error", "home"], err);
            return err;
        }
    }
});

server.route({
    method: 'POST',
    path: '/api/movie/insert',
    config: {
        payload: {
            multipart: true,
        },
        cors: {
            origin: ['*'],
            additionalHeaders: ['cache-control', 'x-requested-width']
        }
    },
    handler: async function (request, reply) {

        const {
            title,
            genre,

```

```

        director,
        release_year
    } = request.payload;

    //const title = request.payload.title;
    //const genre = request.payload.genre;

    try {

        const respondedata = await Movies.MovieRepo.postMovie(title, genre,
director,release_year);
        if (respondedata.error) {
            return respondedata.errorMessage;
        } else {
            return respondedata;
        }
    } catch (err) {
        server.log(["error", "home"], err);
        return err;
    }

}
});

await server.start();
console.log('API Server running on %s', server.info.uri);

//-----
};

process.on('unhandledRejection', (err) => {

    console.log(err);
    process.exit(1);
});

init();

```

สร้างไฟล์ dbconfig.js เพื่อเก็บ config ใช้สำหรับติดต่อกับ db server (MySQL)

```

var dbconfig = {
    development: {
        //connectionLimit : 10,
        host      : 'localhost',
        port      : '3306',
        user      : 'root',
        password  : '',
        database  : 'moviedb'
    },
    production: {

```

```

    //connectionLimit : 10,
    host      : 'localhost',
    port      : '3306',
    user      : 'root',
    password  : '',
    database  : 'moviedb'
  }
};

module.exports = dbconfig;

```

ติดตั้ง MySQL for Nodejs

```
npm install mysql
```

```

Terminal Shell Edit View Window Help
(base) jiraphat@Jiraphats-MacBook-Air-2 backend % npm install mysql
added 12 packages, and audited 138 packages in 3s
16 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
(base) jiraphat@Jiraphats-MacBook-Air-2 backend %

```

สร้าง folder repository

```
mkdir repository
```

สร้างไฟล์ชื่อ movie.js และใส่ code ตามนี้

```

var mysql = require('mysql');
const env = require('../env.js');
const config = require('../dbconfig.js')[env];

/*
async function getMovieList() {

  var Query;

```

```

var pool = mysql.createPool(config);

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

    //Query = `SELECT * FROM movies WHERE warehouse_status = 1 ORDER BY CONVERT(
warehouse_name USING tis620 ) ASC `;
    Query = `SELECT * FROM movies`;

    pool.query(Query, function (error, results, fields) {
        if (error) throw error;

        if (results.length > 0) {
            pool.end();
            return resolve({
                statusCode: 200,
                resultCode: 1,
                data: results,
            });
        } else {
            pool.end();
            return resolve({
                statusCode: 404,
                resultCode: 11,
                message: 'No movie found',
            });
        }
    });

});

}

*/



async function getMovieList() {

    var Query;
    var pool = mysql.createPool(config);

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

        //Query = `SELECT * FROM movies WHERE warehouse_status = 1 ORDER BY CONVERT(
warehouse_name USING tis620 ) ASC `;
        Query = `SELECT * FROM movies`;

        pool.query(Query, function (error, results, fields) {
            if (error) throw error;

            if (results.length > 0) {
                pool.end();
                return resolve(results);
            } else {
                pool.end();
                return resolve({

```

```

        statusCode: 404,
        returnCode: 11,
        message: 'No movie found',
    });
}

});

};

}

async function getMovieSearch(search_text) {

var Query;
var pool = mysql.createPool(config);

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

Query = `SELECT * FROM movies WHERE title LIKE '%${search_text}%'`;

pool.query(Query, function (error, results, fields) {
if (error) throw error;

if (results.length > 0) {
pool.end();
return resolve({
statusCode: 200,
returnCode: 1,
data: results,
});
} else {
pool.end();
return resolve({
statusCode: 404,
returnCode: 11,
message: 'No movie found',
});
}
});

});

}

async function postMovie(p_title,p_genre,p_director,p_release_year) {

var Query;
var pool = mysql.createPool(config);

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

```

```

//Query = `SELECT * FROM movies WHERE title LIKE '%${search_text}%`;

var post = {
    title: p_title,
    genre: p_genre,
    director: p_director,
    release_year: p_release_year
};

console.log('post is: ', post);

Query = 'INSERT INTO movies SET ?';
pool.query(Query, post, function (error, results, fields) {
//pool.query(Query, function (error, results, fields) {

    if (error) throw error;

    if (results.length > 0) {
        pool.end();
        return resolve({
            statusCode: 200,
            returnCode: 1,
            message: 'Movie list was inserted',
        });
    }
});

});

}

module.exports.MovieRepo = {
    getMovieList: getMovieList,
    getMovieSearch: getMovieSearch,
    postMovie: postMovie,
};

}

```

ทดสอบการทำงาน

```

cd ..
nodemon index.js

```

โดยใช้ postman

<http://localhost:3001>

Postman

File Edit View Window Help

Home Workspaces API Network Explore

Search Postman

My Workspace

New Import

Overview GET test insert d ● GET http://api.air ● GET http://api.air ● SystemD GET test system ●

Save + ⚡ No Environment

Collections

APIs

Environments

Mock Servers

Monitors

Flows

History

SystemD / test system

GET http://localhost:3001

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description	...	

Body Cookies Headers (9) Test Results

Pretty Raw Preview Visualize HTML

1 `<h3> Welcome to API Back-end Ver. 1.0.0</h3>`

Status: 200 OK Time: 11 ms Size: 338 B Save Response

Online Find and Replace Console Cookies Capture requests Runner Trash



<http://localhost:3001/api/movie/all>

Postman

File Edit View Window Help

Home Workspaces API Network Explore

Search Postman

My Workspace

New Import

Overview GET test insert d ● GET http://api.air ● GET http://api.air ● SystemD GET test system ●

Save + ⚡ No Environment

Collections

APIs

Environments

Mock Servers

Monitors

Flows

History

SystemD / test system

GET http://localhost:3001/api/movie/all

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description	...	

Body Cookies Headers (9) Test Results

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "title": "Avengers: Endgame",
4     "genre": "superhero",
5     "director": "Anthony Russo, Joe Russo",
6     "release_year": 2019
7   },
8   {
9     "title": "Fight Club",
10    "genre": "drama",
11    "director": "David Fincher",
12    "release_year": 1999
13  },
14  {
15    "title": "Forrest Gump",
16    "genre": "drama",
17    "director": "Robert Zemeckis",
```

Status: 200 OK Time: 16 ms Size: 2.28 KB Save Response

Online Find and Replace Console Cookies Capture requests Runner Trash



[http://localhost:3001/api/movie/search?search\\_text=jok](http://localhost:3001/api/movie/search?search_text=jok)

Postman

File Edit View Window Help

Home Workspaces API Network Explore

Search Postman

My Workspace

New Import Overview GET test insert d ● GET http://api.air ● GET http://api.air ● SystemD GET test system ●

Collections + New Collection SystemD GET test system test

APIs Environments Mock Servers Monitors Flows History

SystemD / test system

Set as variable ...

GET http://localhost:3001/api/movie/search?search\_text=jok

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> search_text	jok			
Key	Value	Description		

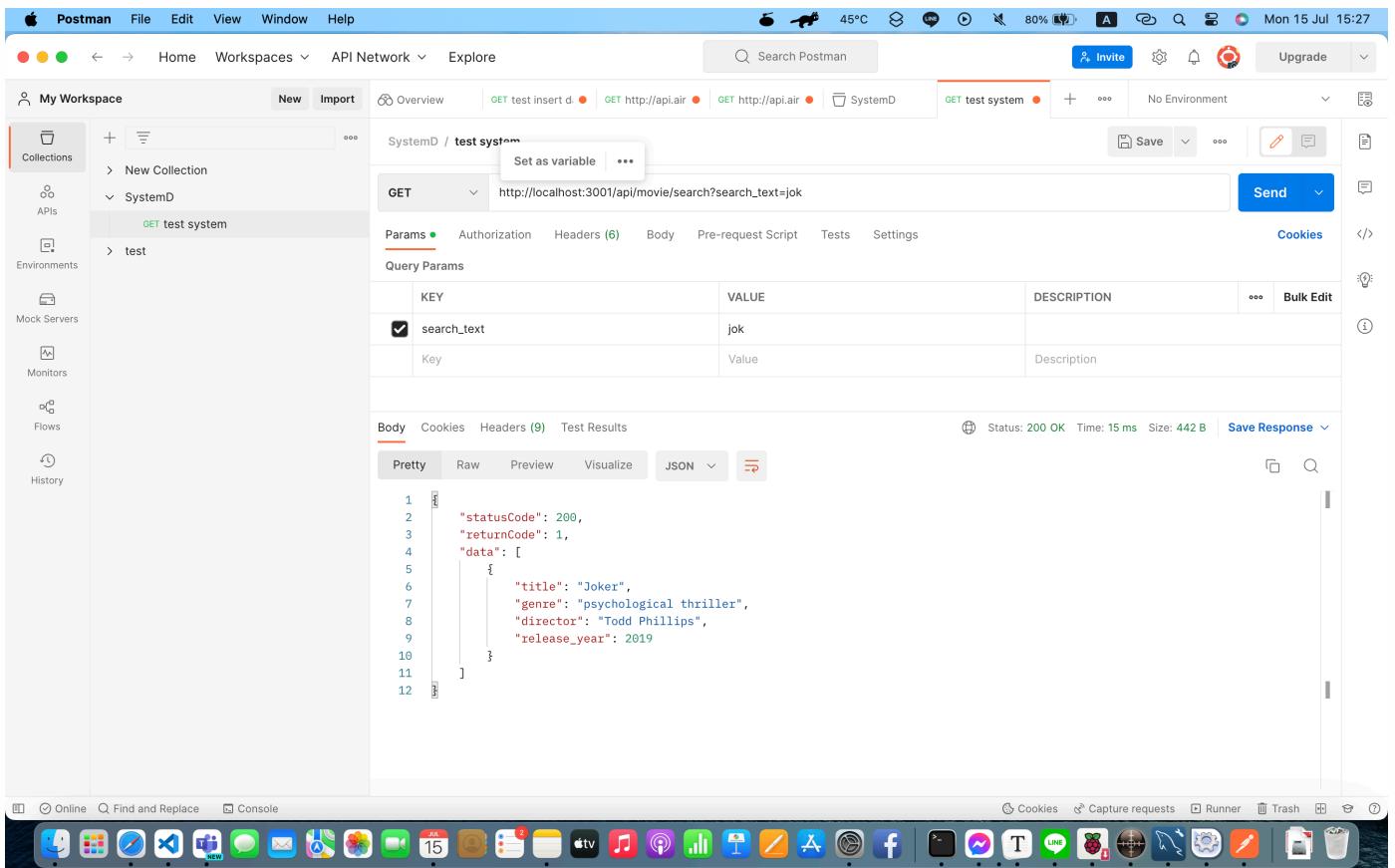
Body Cookies Headers (9) Test Results

Pretty Raw Preview Visualize JSON

```
1 "statusCode": 200,
2 "returnCode": 1,
3 "data": [
4   {
5     "title": "Joker",
6     "genre": "psychological thriller",
7     "director": "Todd Phillips",
8     "release_year": 2019
9   }
10 ]
11 ]
12 ]
```

Status: 200 OK Time: 15 ms Size: 442 B Save Response

Online Find and Replace Console Cookies Capture requests Runner Trash ?



ทดสอบเพิ่มเติมเปลี่ยนชื่อหนัง

[http://localhost:3001/api/movie/search?search\\_text=Fight](http://localhost:3001/api/movie/search?search_text=Fight)

Postman

File Edit View Window Help

Home Workspaces API Network Explore

Search Postman

My Workspace

New Import Overview GET test insert d ● GET http://api.air ● GET http://api.air ● SystemD GET test system ●

Collections + New Collection SystemD GET test system test

APIs Environments Mock Servers Monitors Flows History

SystemD / test system

GET http://localhost:3001/api/movie/search?search\_text=Fight

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> search_text	Fight			
Key	Value	Description		

Body Cookies Headers (9) Test Results

Pretty Raw Preview Visualize JSON

```
1 "statusCode": 200,
2 "returnCode": 1,
3 "data": [
4   {
5     "title": "Fight Club",
6     "genre": "drama",
7     "director": "David Fincher",
8     "release_year": 1999
9   }
10 ]
11 ]
12 ]
```

Status: 200 OK Time: 9 ms Size: 430 B Save Response

Online Find and Replace Console Cookies Capture requests Runner Trash ?

