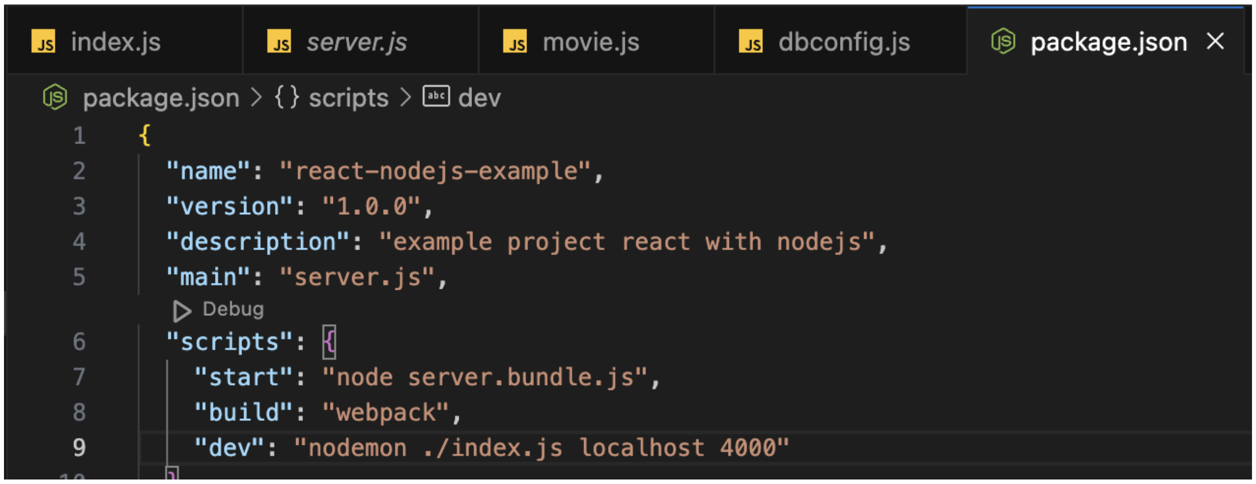
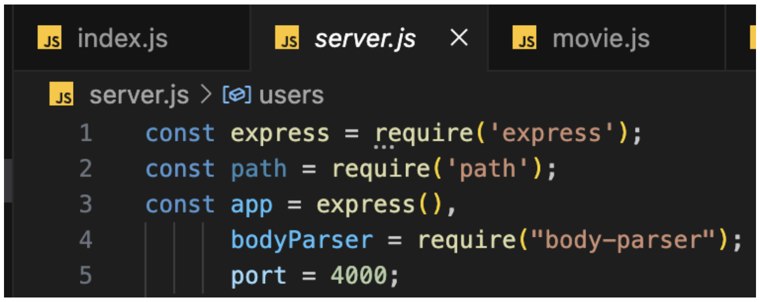
ENGSE203 LAB4

เปลี่ยน Port ของ Backend เป็น 4000

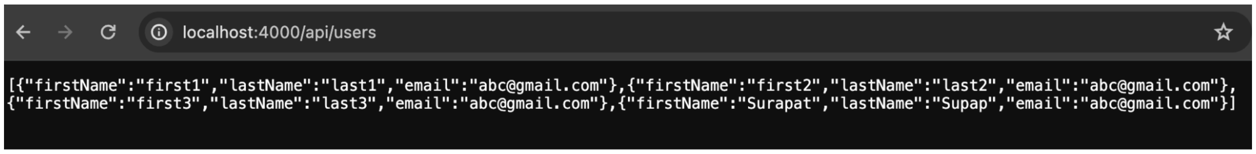


```json
{
  "name": "react-nodejs-example",
  "version": "1.0.0",
  "description": "example project react with nodejs",
  "main": "server.js",
  "scripts": {
    "start": "node server.bundle.js",
    "build": "webpack",
    "dev": "nodemon ./index.js localhost 4000"
```



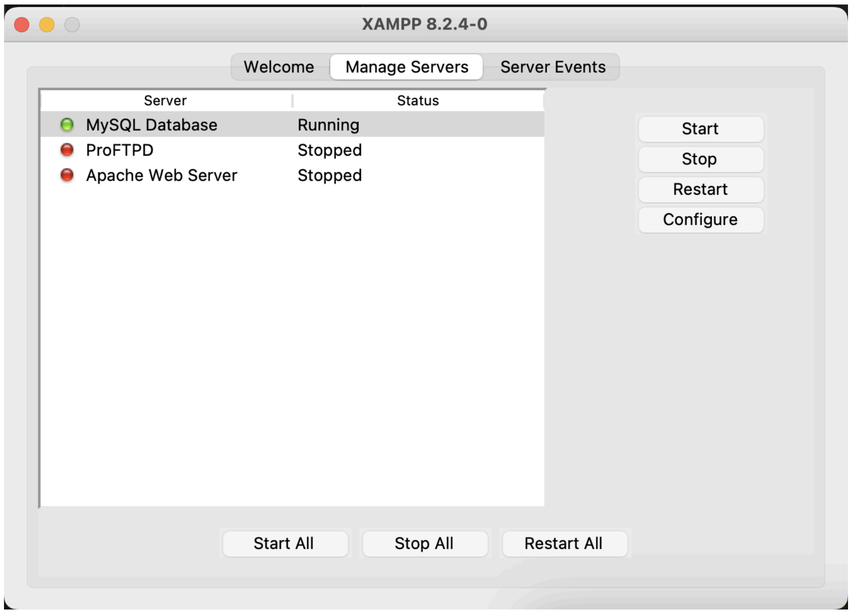
```javascript
const express = require('express');
const path = require('path');
const app = express(),
      bodyParser = require("body-parser");
      port = 4000;
```

ทดสอบ รัน Port 4000



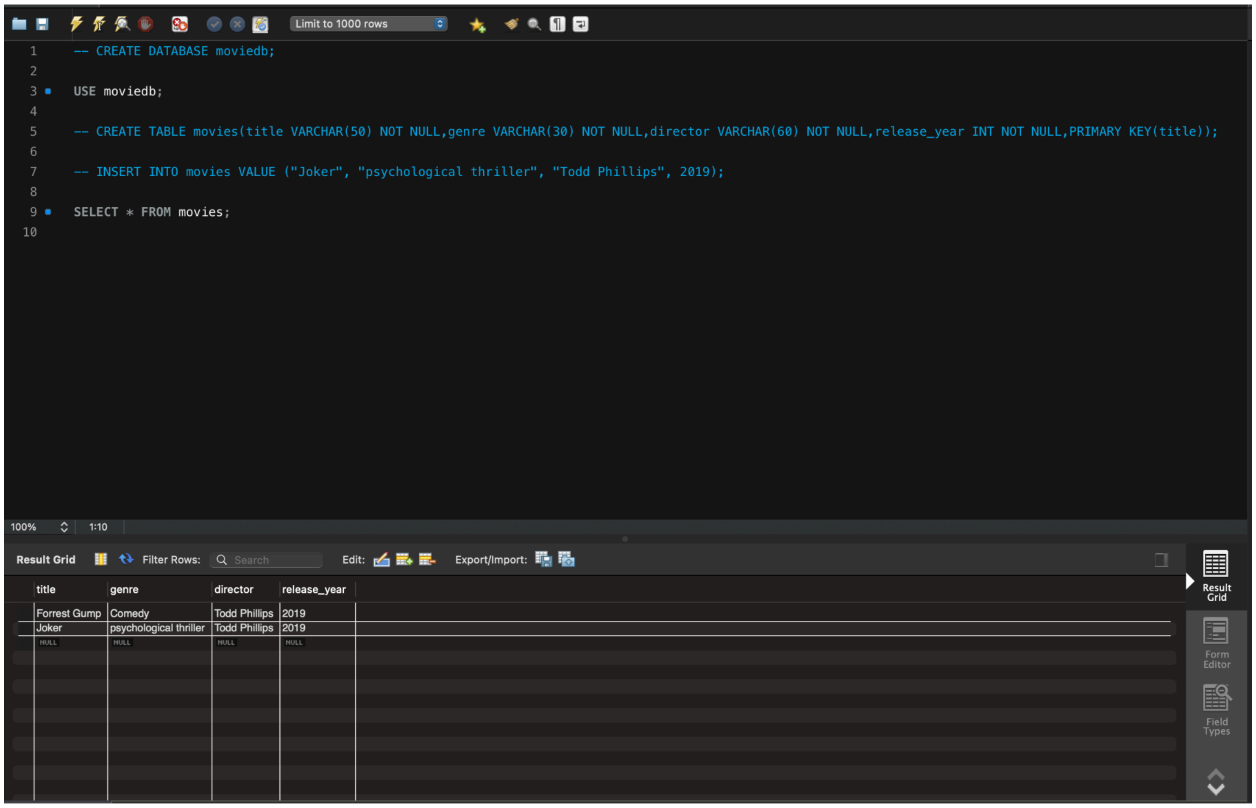
```
[{"firstName":"first1","lastName":"last1","email":"abc@gmail.com"},{"firstName":"first2","lastName":"last2","email":"abc@gmail.com"},
{"firstName":"first3","lastName":"last3","email":"abc@gmail.com"},{"firstName":"Surapat","lastName":"Supap","email":"abc@gmail.com"}]
```

เปิด MySQL ใน Xammp



Server	Status
MySQL Database	Running
ProFTPD	Stopped
Apache Web Server	Stopped

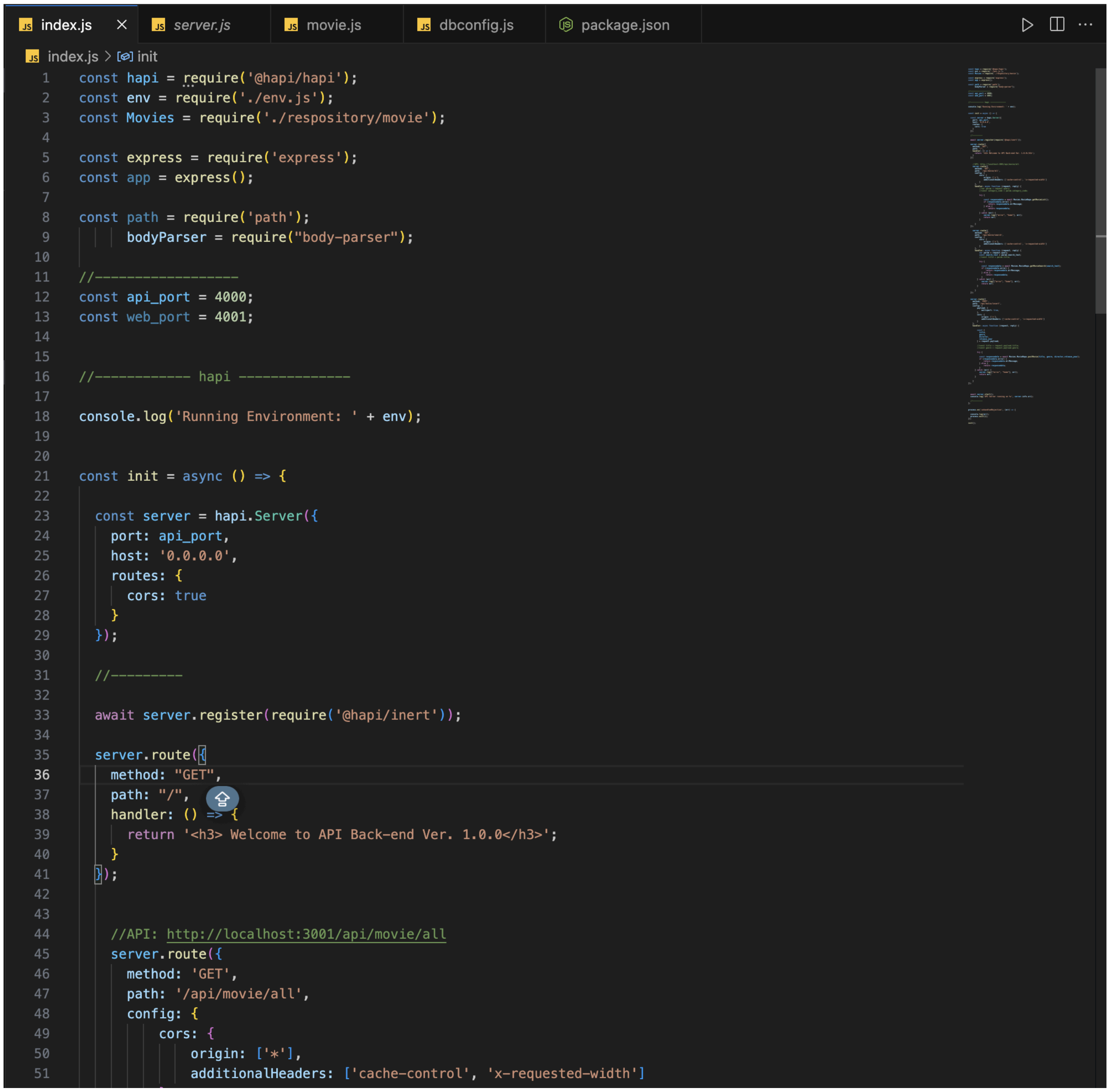
สร้าง database , เพิ่มตาราและเพิ่มข้อมูลลงในตาราง



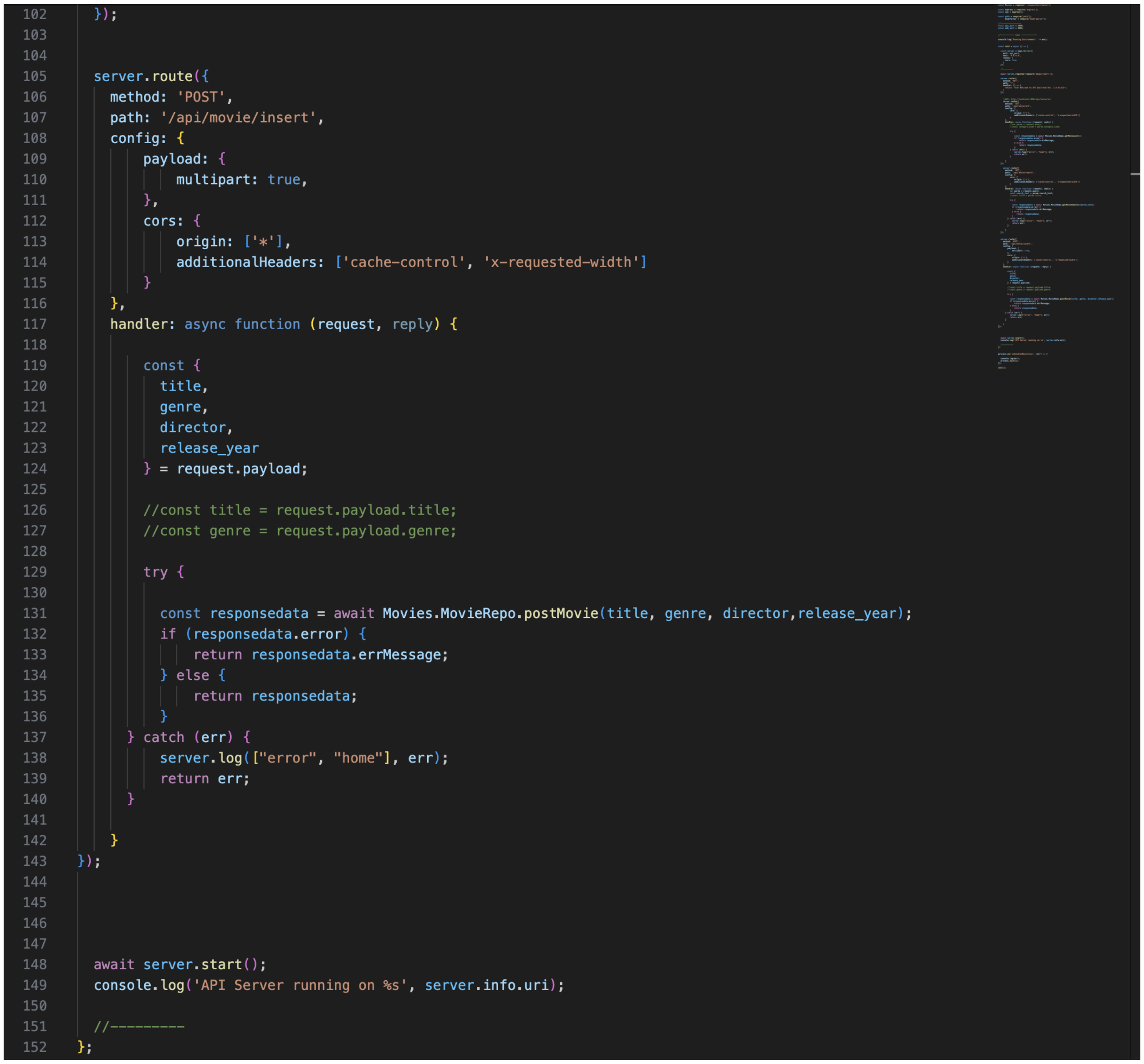
```
-- CREATE DATABASE moviedb;

USE moviedb;

-- CREATE TABLE movies(title VARCHAR(50) NOT NULL,genre VARCHAR(30) NOT NULL,director VARCHAR(60) NOT NULL,release_year INT NOT NULL,PRIMARY KEY(title));

-- INSERT INTO movies VALUE ("Joker", "psychological thriller", "Todd Phillips", 2019);

SELECT * FROM movies;
```

title	genre	director	release_year
Forrest Gump	Comedy	Todd Phillips	2019
Joker	psychological thriller	Todd Phillips	2019
NULL	NULL	NULL	NULL

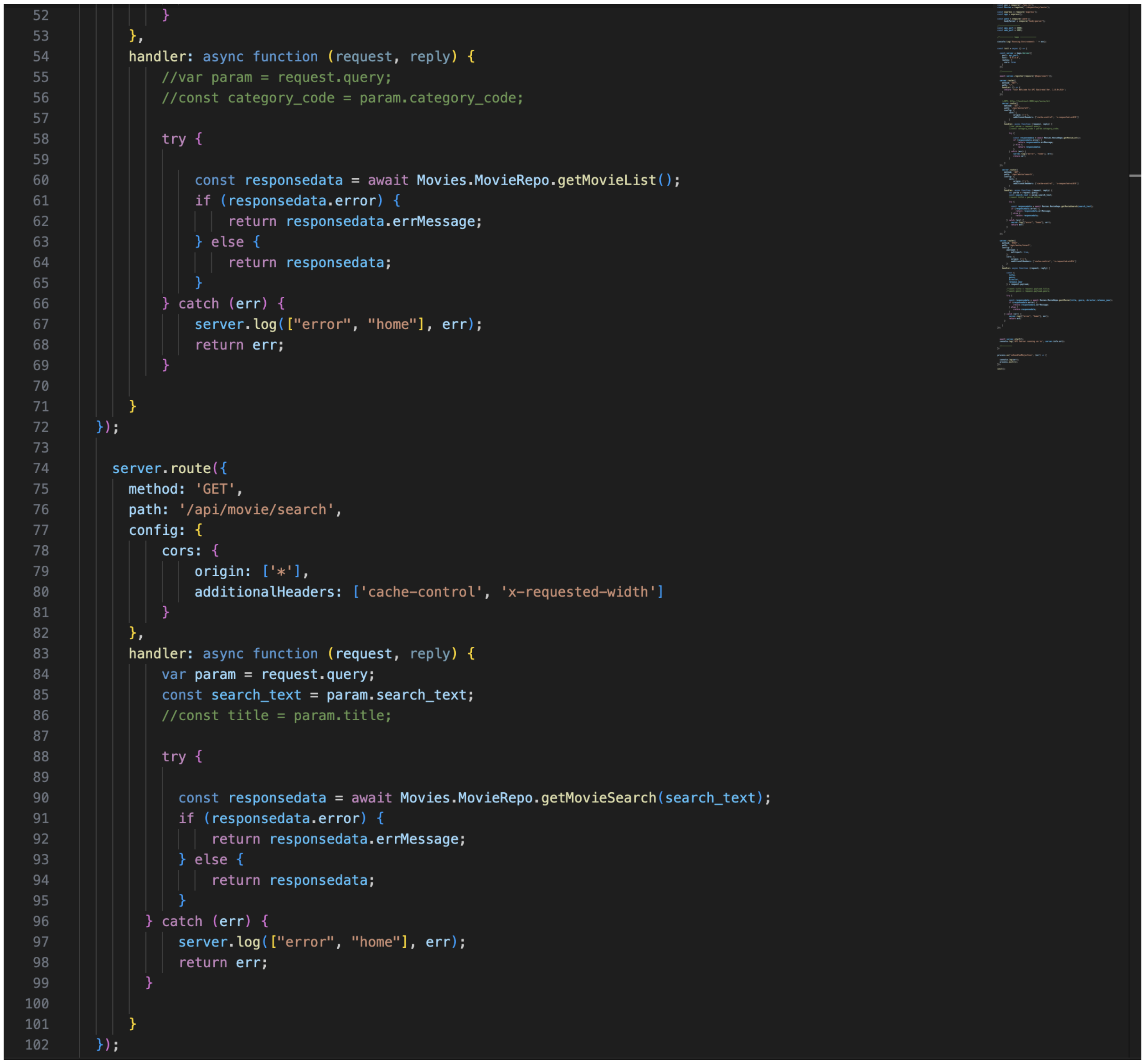
สร้าง index.js



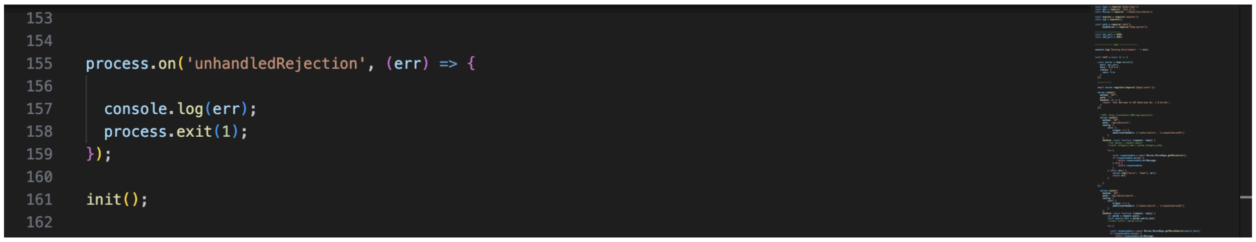
```
const hapi = require('@hapi/hapi');
const env = require('./env.js');
const Movies = require('./respository/movie');

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

const path = require('path');
    bodyParser = require("body-parser");

//-----------------
const api_port = 4000;
const web_port = 4001;


//----------- 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']
```



```
  });


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

    }
});




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

  //---------
};
```

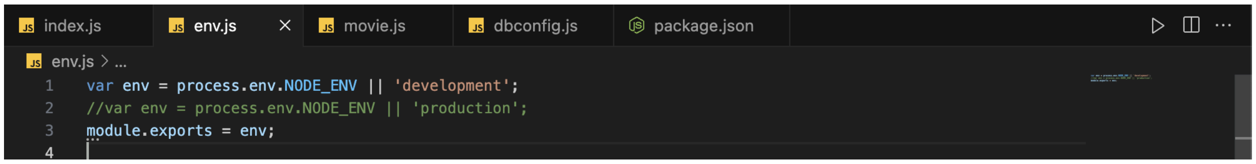


```
            }
        },
        handler: async function (request, reply) {
            //var param = request.query;
            //const category_code = param.category_code;

            try {

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

        }
    });
```



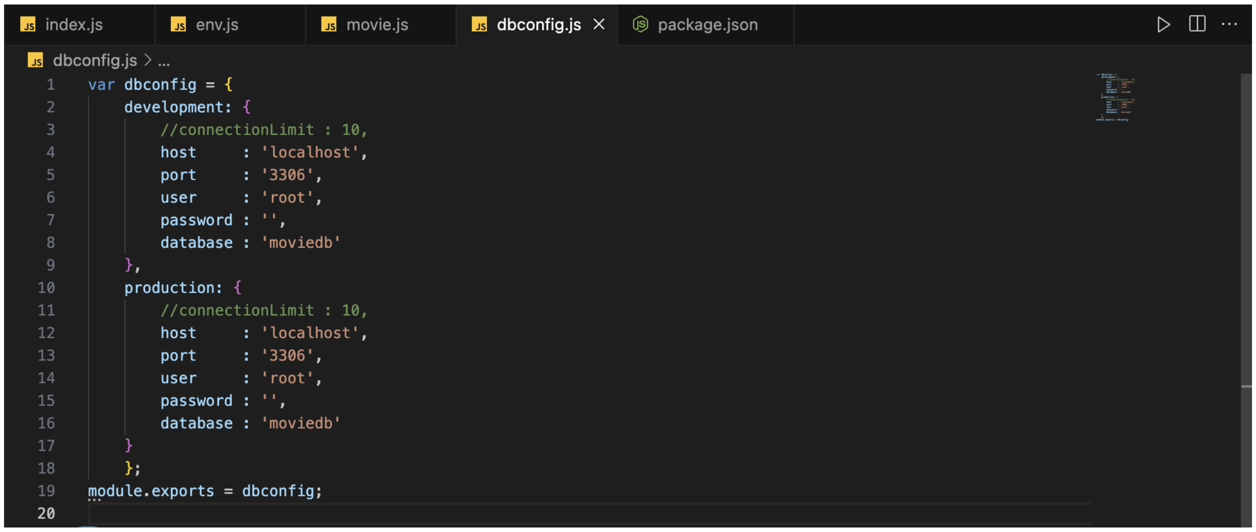
```


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

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

init();

```

สร้าง env.js



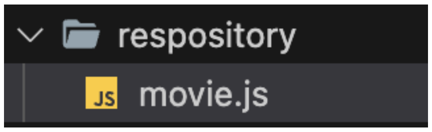
```
var env = process.env.NODE_ENV || 'development';
//var env = process.env.NODE_ENV || 'production';
module.exports = env;

```

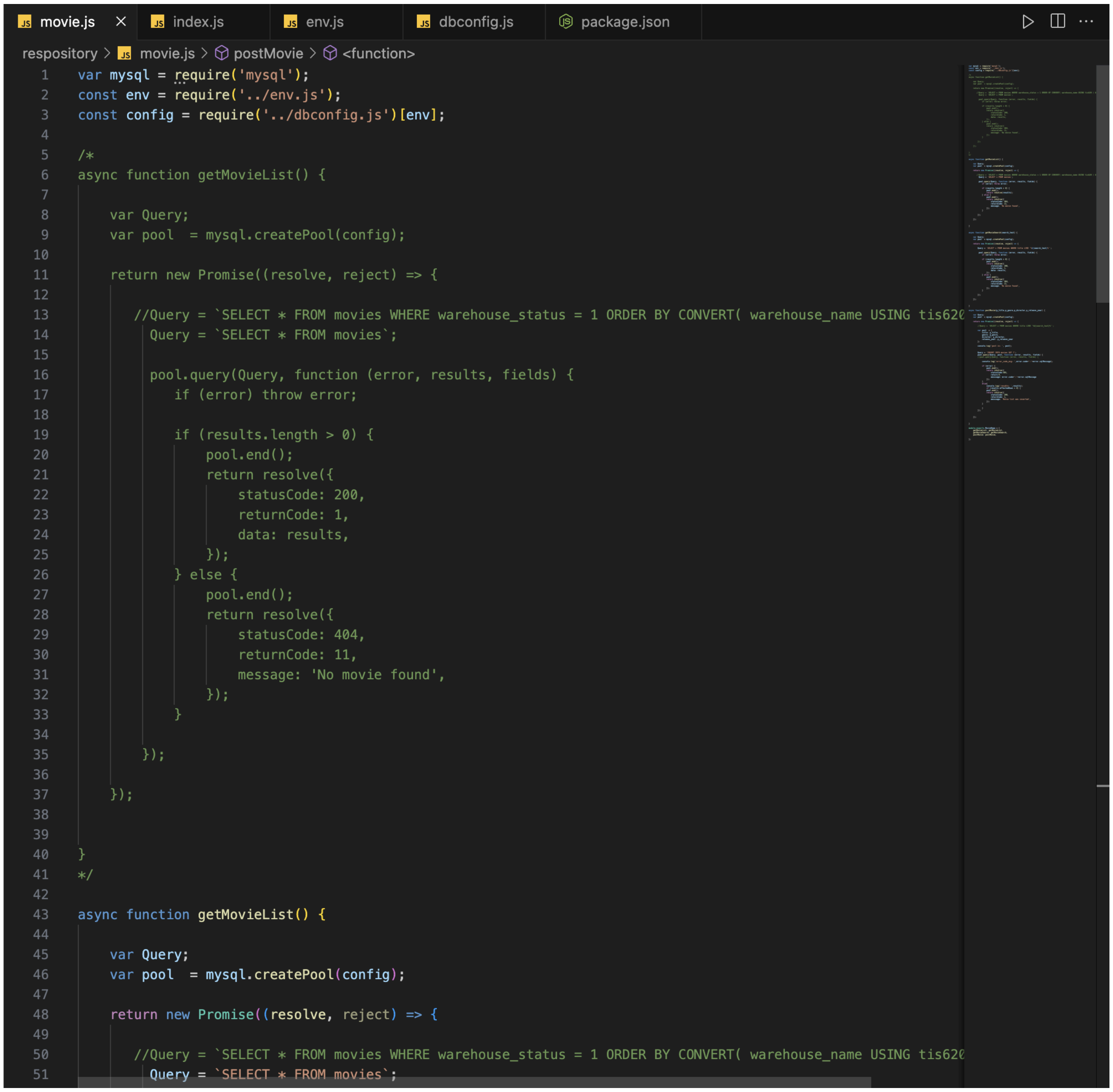
สร้าง dbconfig.js



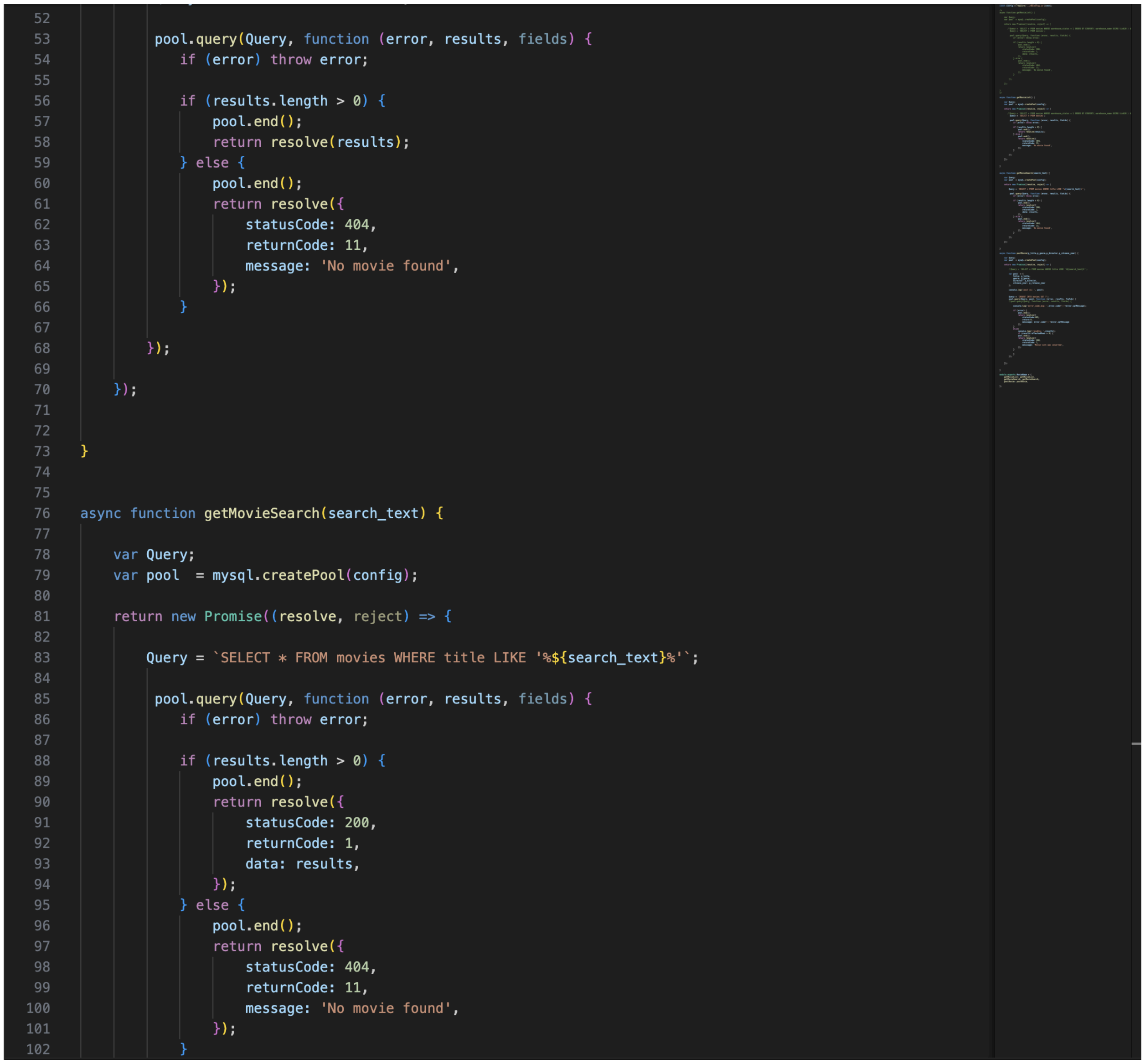
```
var dbconfig = {
    development: {
        //connectionLimit : 10,
        host     : 'localhost',
        port     : '3306',
        user     : 'root',
        password : '',
        database : 'moviedb'
    },
    production: {
        //connectionLimit : 10,
        host     : 'localhost',
 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

```

สร้างโฟลเดอร์ respository และสร้างไฟล์ movie.js

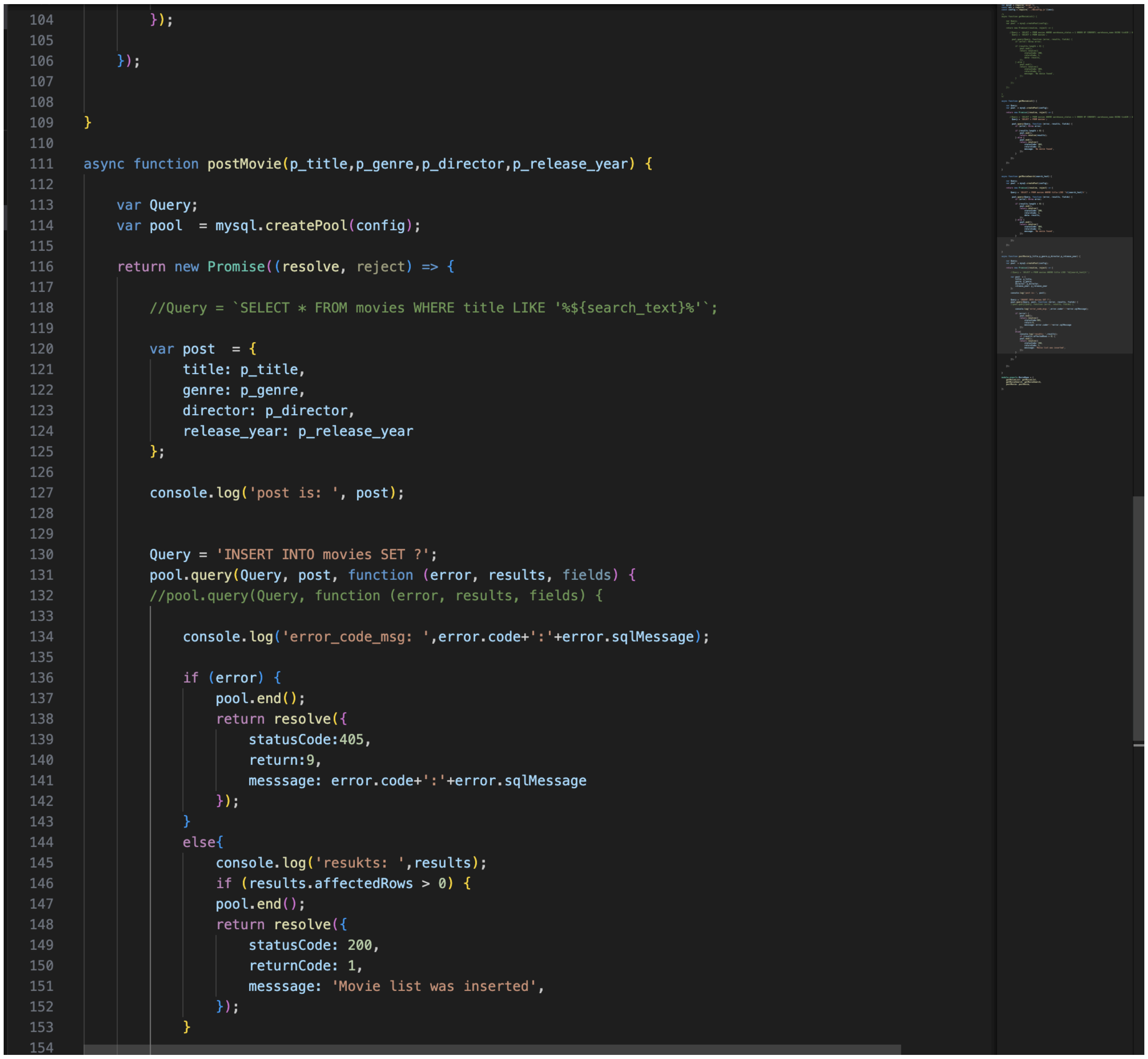




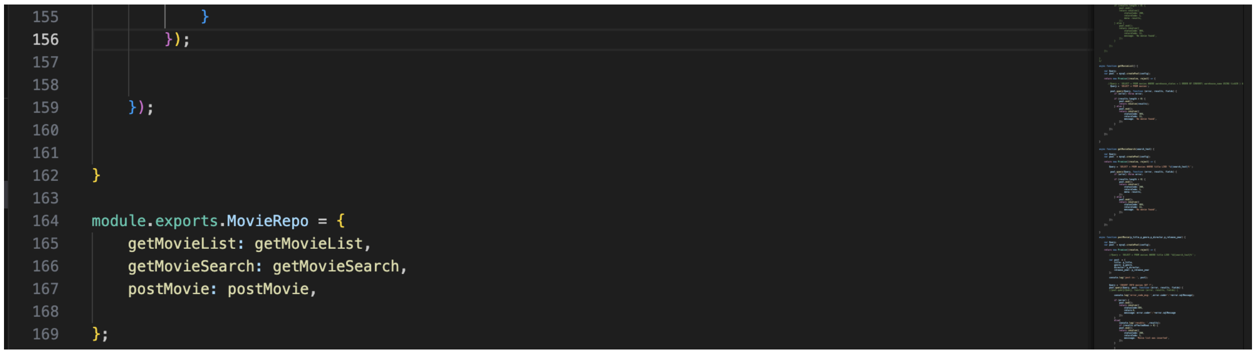
```
var mysql = require('mysql');
const env = require('../env.js');
const config = require('../dbconfig.js')[env];

/*
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
         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,
                    returnCode: 1,
                    data: results,
                });
            } else {
                pool.end();
                return resolve({
                    statusCode: 404,
                    returnCode: 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
         Query = `SELECT * FROM movies`;
```



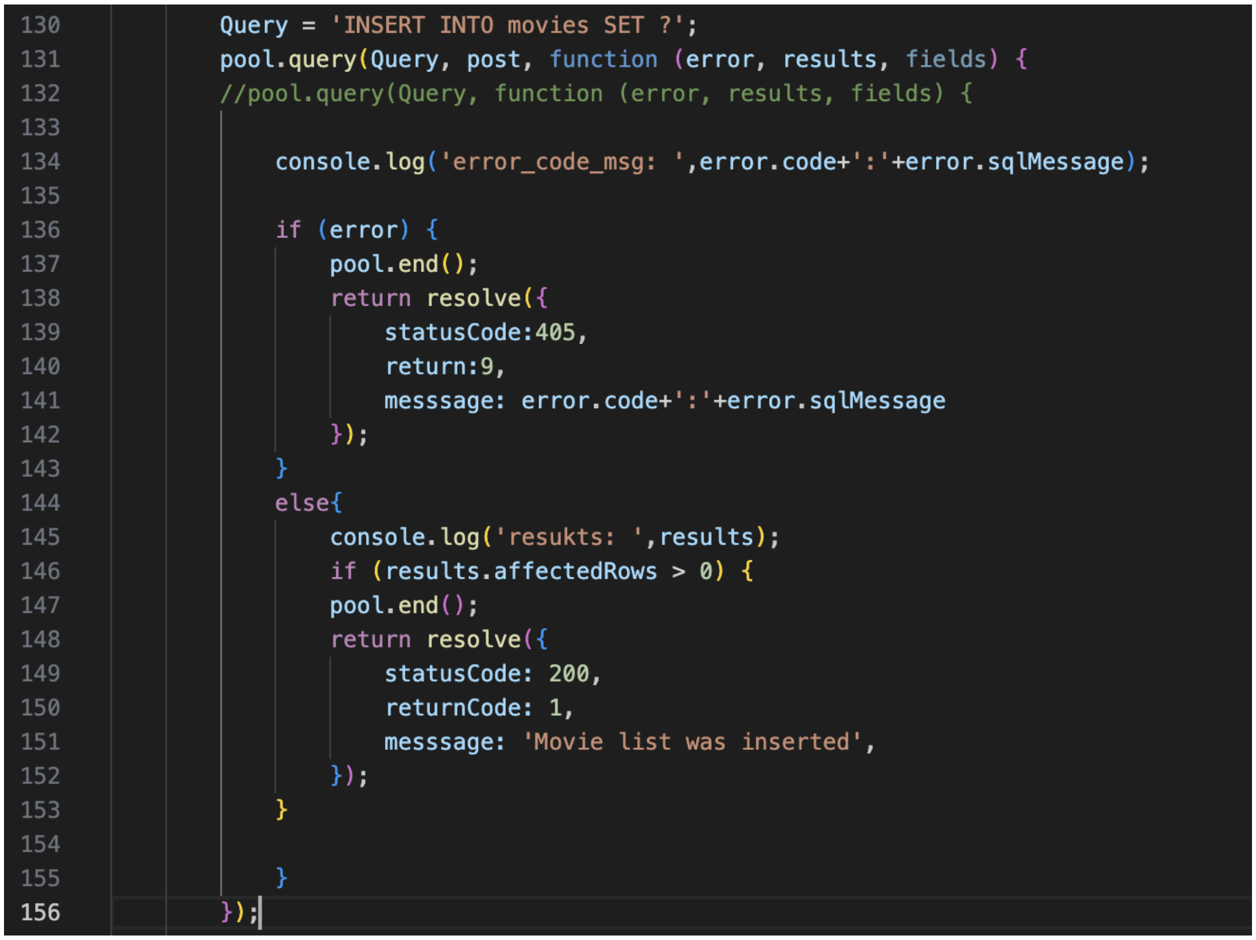
```
         pool.query(Query, function (error, results, fields) {
            if (error) throw error;

            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) {

            console.log('error_code_msg: ',error.code+':'+error.sqlMessage);

            if (error) {
                pool.end();
                return resolve({
                    statusCode:405,
                    return:9,
                    messsage: error.code+':'+error.sqlMessage
                });
            }
            else{
                console.log('resukts: ',results);
                if (results.affectedRows > 0) {
                pool.end();
                return resolve({
                    statusCode: 200,
                    returnCode: 1,
                    messsage: 'Movie list was inserted',
                });
            }

```



```
            }
        });


    });


}

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

};
```



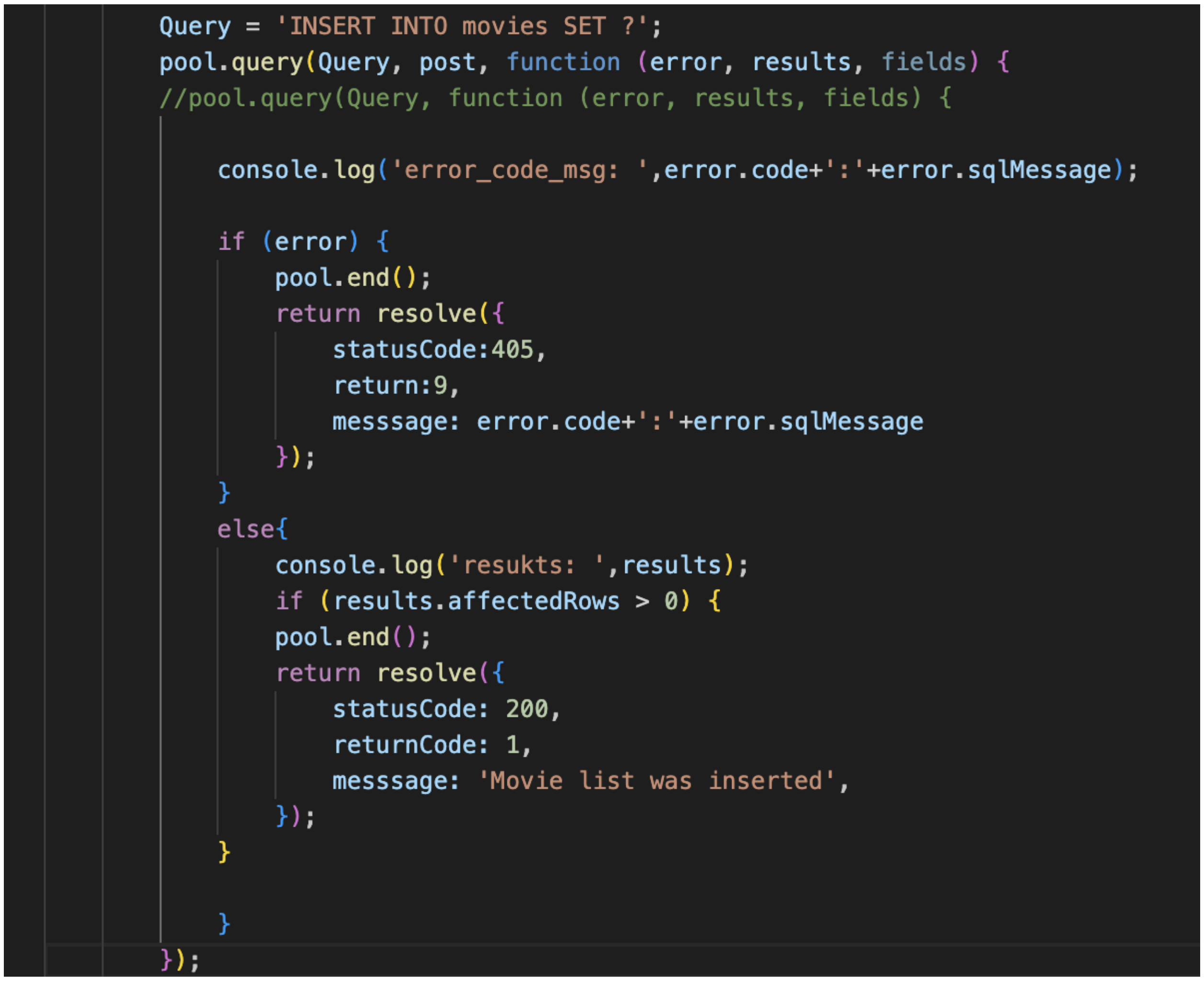
```
        Query = 'INSERT INTO movies SET ?';
        pool.query(Query, post, function (error, results, fields) {
        //pool.query(Query, function (error, results, fields) {

            console.log('error_code_msg: ',error.code+':'+error.sqlMessage);

            if (error) {
                pool.end();
                return resolve({
                    statusCode:405,
                    return:9,
                    messsage: error.code+':'+error.sqlMessage
                });
            }
            else{
                console.log('resukts: ',results);
                if (results.affectedRows > 0) {
                pool.end();
                return resolve({
                    statusCode: 200,
                    returnCode: 1,
                    messsage: 'Movie list was inserted',
                });
            }

            }
        });
```

แก้ไข การทำงาน ในบรรทัด 132 ถึง 142

Before

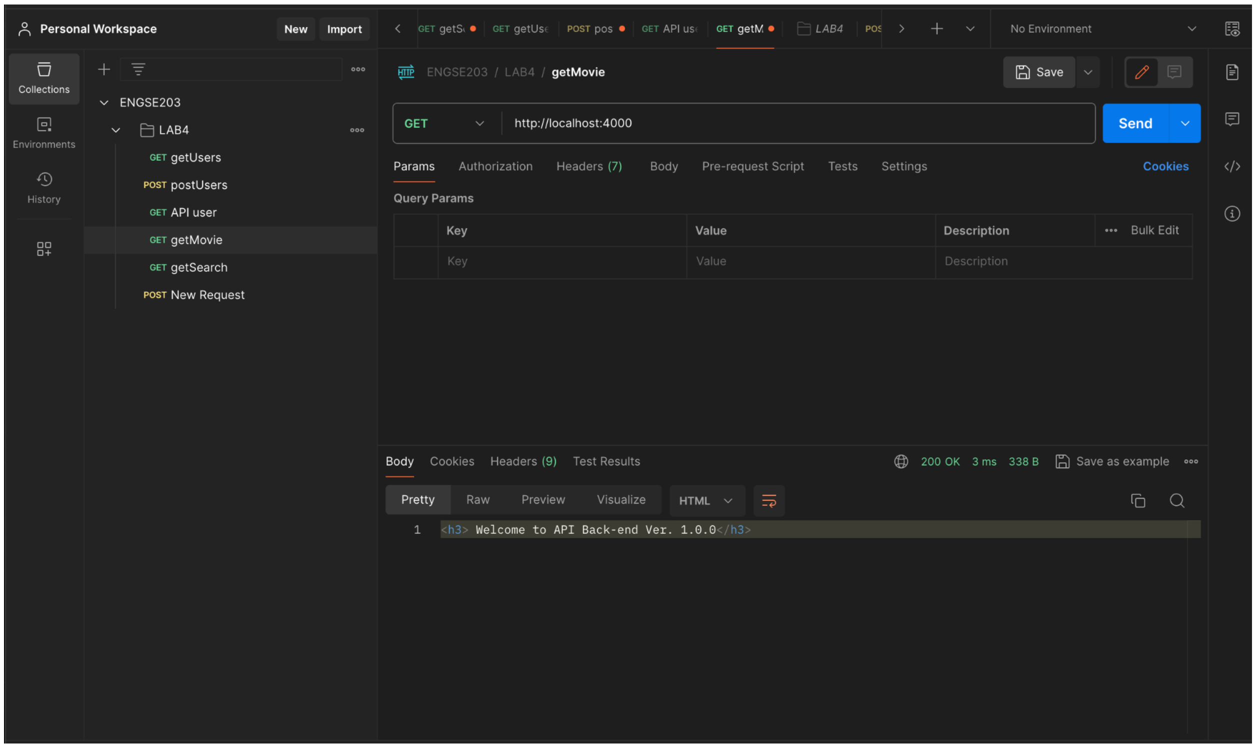
```
 if (error) throw error;

           if (results.length > 0) {

               pool.end();

               return resolve({

                   statusCode: 200,

                   returnCode: 1,

                   messsage: 'Movie list was inserted',

               });

           }

       });

   });

}
```

After



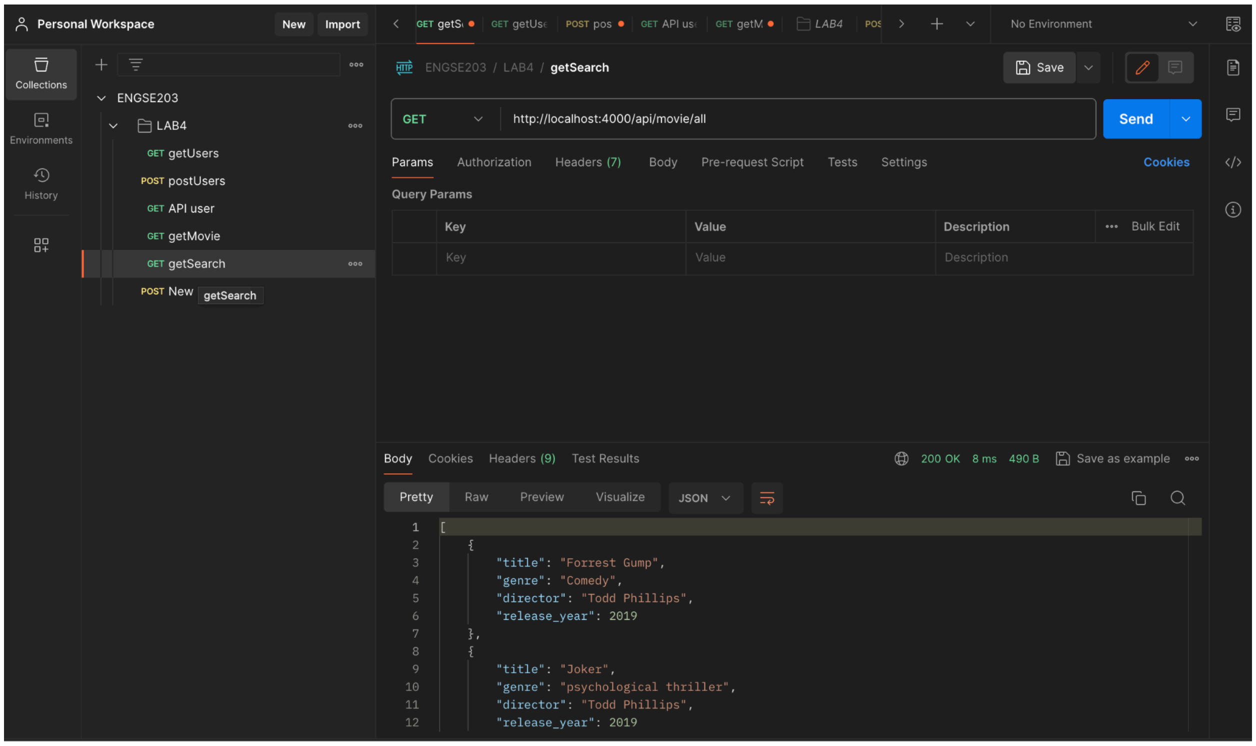
```
        Query = 'INSERT INTO movies SET ?';
        pool.query(Query, post, function (error, results, fields) {
        //pool.query(Query, function (error, results, fields) {

            console.log('error_code_msg: ',error.code+':'+error.sqlMessage);

            if (error) {
                pool.end();
                return resolve({
                    statusCode:405,
                    return:9,
                    messsage: error.code+':'+error.sqlMessage
                });
            }
            else{
                console.log('resukts: ',results);
                if (results.affectedRows > 0) {
                pool.end();
                return resolve({
                    statusCode: 200,
                    returnCode: 1,
                    messsage: 'Movie list was inserted',
                });
            }

            }
        });
```

ทดสอบการทำงานโดยใช้ Postman



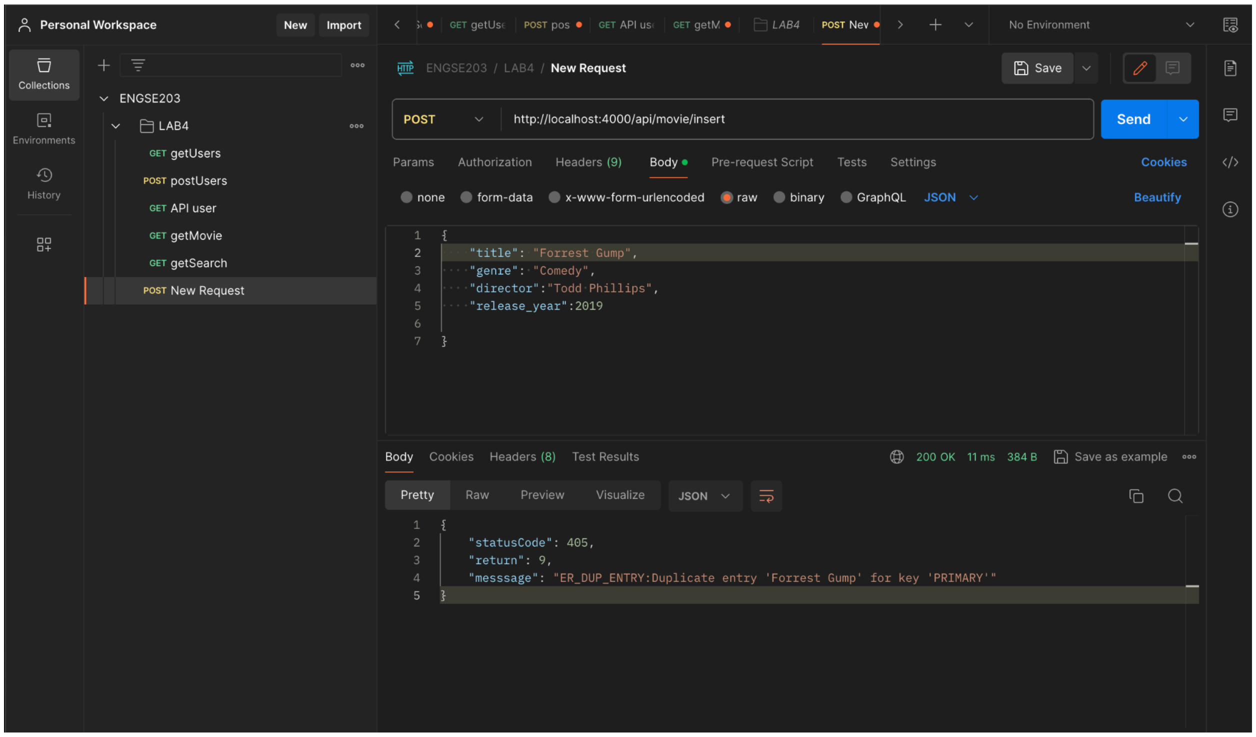
GET http://localhost:4000

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



GET http://localhost:4000/api/movie/all

```
[
    {
        "title": "Forrest Gump",
        "genre": "Comedy",
        "director": "Todd Phillips",
        "release_year": 2019
    },
    {
        "title": "Joker",
        "genre": "psychological thriller",
        "director": "Todd Phillips",
        "release_year": 2019
```

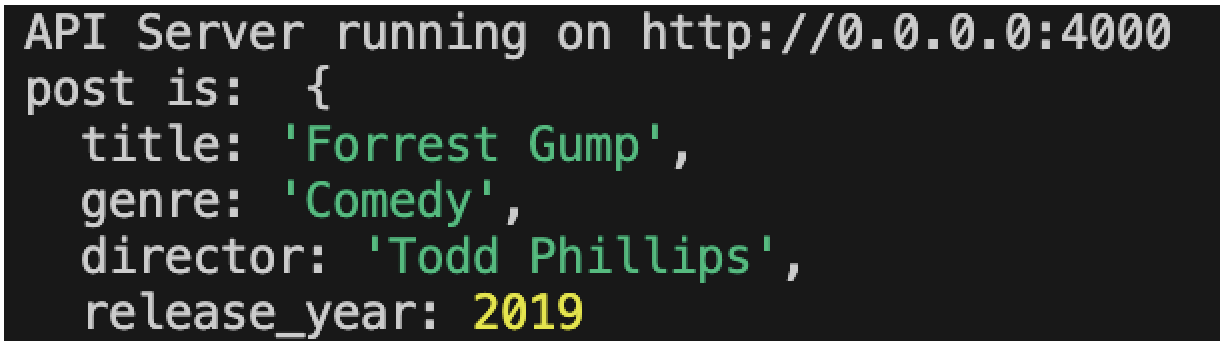


POST http://localhost:4000/api/movie/insert

```
{
····"title":·"Forrest·Gump",
····"genre":·"Comedy",
····"director":"Todd·Phillips",
····"release_year":2019

}
```

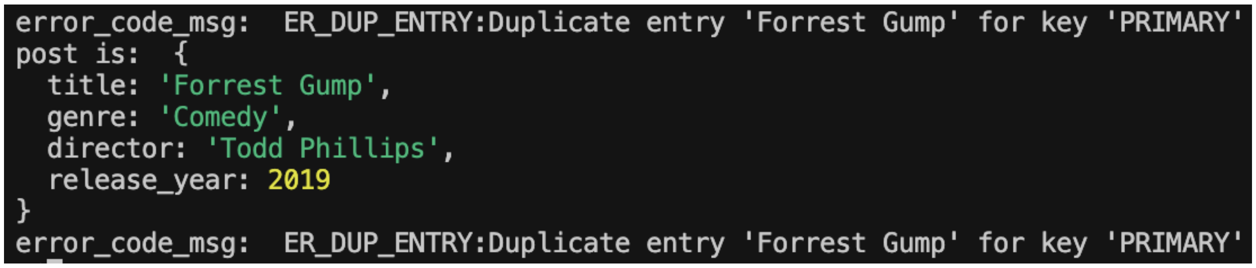
200 OK 11 ms 384 B

```
{
    "statusCode": 405,
    "return": 9,
    "messsage": "ER_DUP_ENTRY:Duplicate entry 'Forrest Gump' for key 'PRIMARY'"
}
```



```
API Server running on http://0.0.0.0:4000
post is:  {
  title: 'Forrest Gump',
  genre: 'Comedy',
  director: 'Todd Phillips',
  release_year: 2019
```

กรณีข้อมูลซ้ำกัน



```
error_code_msg:  ER_DUP_ENTRY:Duplicate entry 'Forrest Gump' for key 'PRIMARY'
post is:  {
  title: 'Forrest Gump',
  genre: 'Comedy',
  director: 'Todd Phillips',
  release_year: 2019
}
error_code_msg:  ER_DUP_ENTRY:Duplicate entry 'Forrest Gump' for key 'PRIMARY'
```

สุรพัศ สุภาพ 66543210033-7