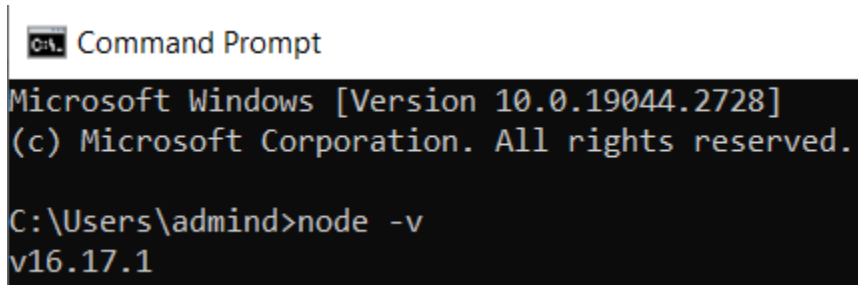


BÀI THỰC HÀNH 2

THIẾT LẬP BACKEND VỚI NODE|EXPRESSJS

Bài 1: Thiết lập môi trường.

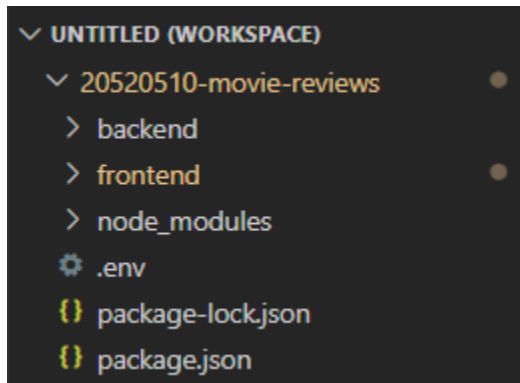
1.1 Tải và cài đặt nodejs – nodejs.org. (Kiểm tra cài đặt thành công với câu lệnh `node -v` trên giao diện dòng lệnh như terminal trên MacOS, Linux và cmd, powershell trên Windows).



```
C:\Users\admin>node -v
v16.17.1
```

1.2 Tải và cài đặt một trong các công cụ soạn thảo và quản lý mã nguồn như: Visual Studio Code, Sublime Text, Notepad++,...

1.3 Khởi tạo cây thư mục chứa mã nguồn của dự án: ví dụ: movie-reviews/backend.



1.4 Khởi tạo dự án với câu lệnh `npm init`.

1.5 Cài đặt một số dependency của dự án như mongodb, express, cors, dotenv.

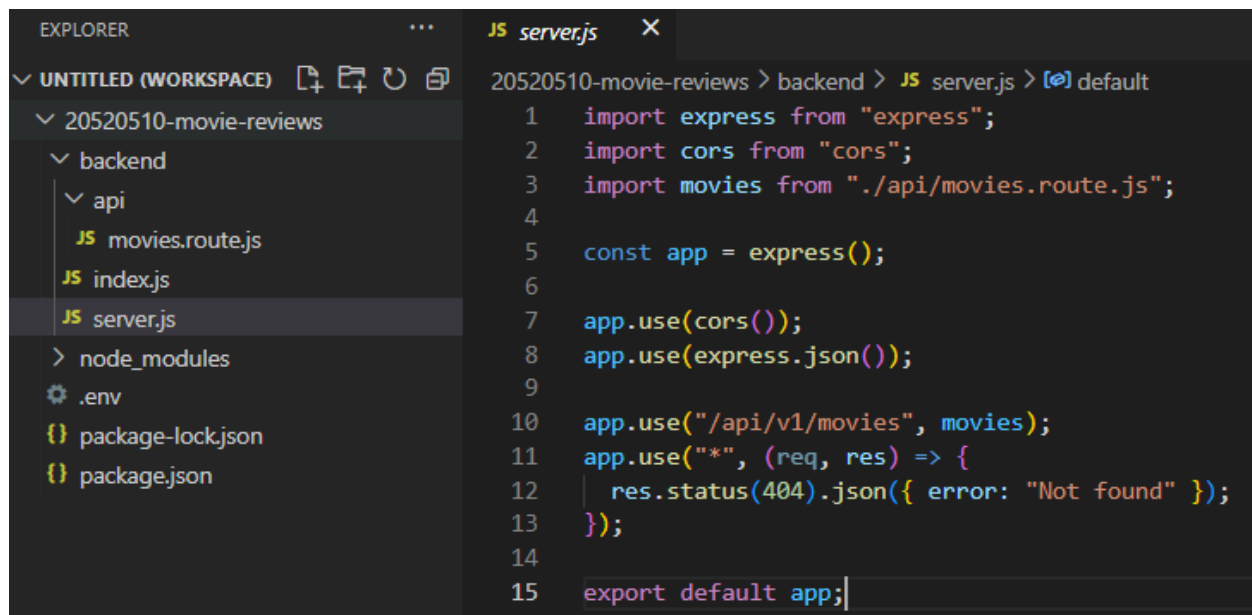
1.6 Cài đặt nodemon – công cụ giúp khởi động lại máy chủ web khi có sự thay đổi về mã nguồn.

```
{} package.json X
20520510-movie-reviews > {} package.json > {} scripts > test
1  {
2    "name": "20520510-movie-reviews",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "type": "module",
7    "scripts": {
8      "test": "echo \"Error: no test specified\" && exit 1",
9      "start": "node ./backend/index.js"
10   },
11   "author": "",
12   "license": "ISC",
13   "dependencies": {
14     "cors": "^2.8.5",
15     "dotenv": "^16.0.3",
16     "express": "^4.18.2",
17     "mongodb": "^5.1.0"
18   },
19   "devDependencies": {
20     "nodemon": "^2.0.21"
21   }
22 }
```

Bài 2:

2.1 Tạo tệp tin server.js là nơi khởi tạo máy chủ web (tệp này nằm trong thư mục backend).

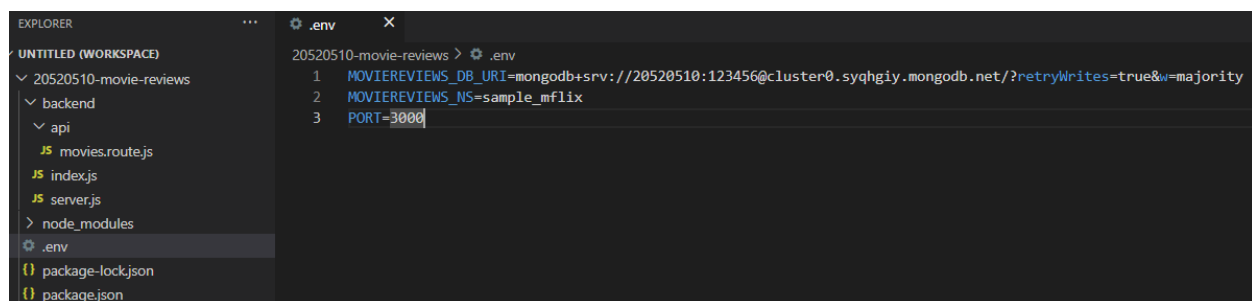
- Trong tệp tin này cần thêm các dependency như express, cors để sử dụng các phương thức (middleware) của chúng.
- Đồng thời, nội dung tệp tin này cũng chứa một số routing cơ bản cho máy chủ web, như xử lý lỗi 404, định tuyến tới /api/v1/movies (xem phần 2.4).



The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays the project structure: '20520510-movie-reviews' with subfolders 'backend' and 'api'. The 'api' folder contains 'movies.route.js', 'index.js', and 'server.js'. The 'server.js' file is selected. The main editor area shows the code for 'server.js' with the following content:

```
1 import express from "express";
2 import cors from "cors";
3 import movies from "../api/movies.route.js";
4
5 const app = express();
6
7 app.use(cors());
8 app.use(express.json());
9
10 app.use("/api/v1/movies", movies);
11 app.use("*", (req, res) => {
12   res.status(404).json({ error: "Not found" });
13 });
14
15 export default app;
```

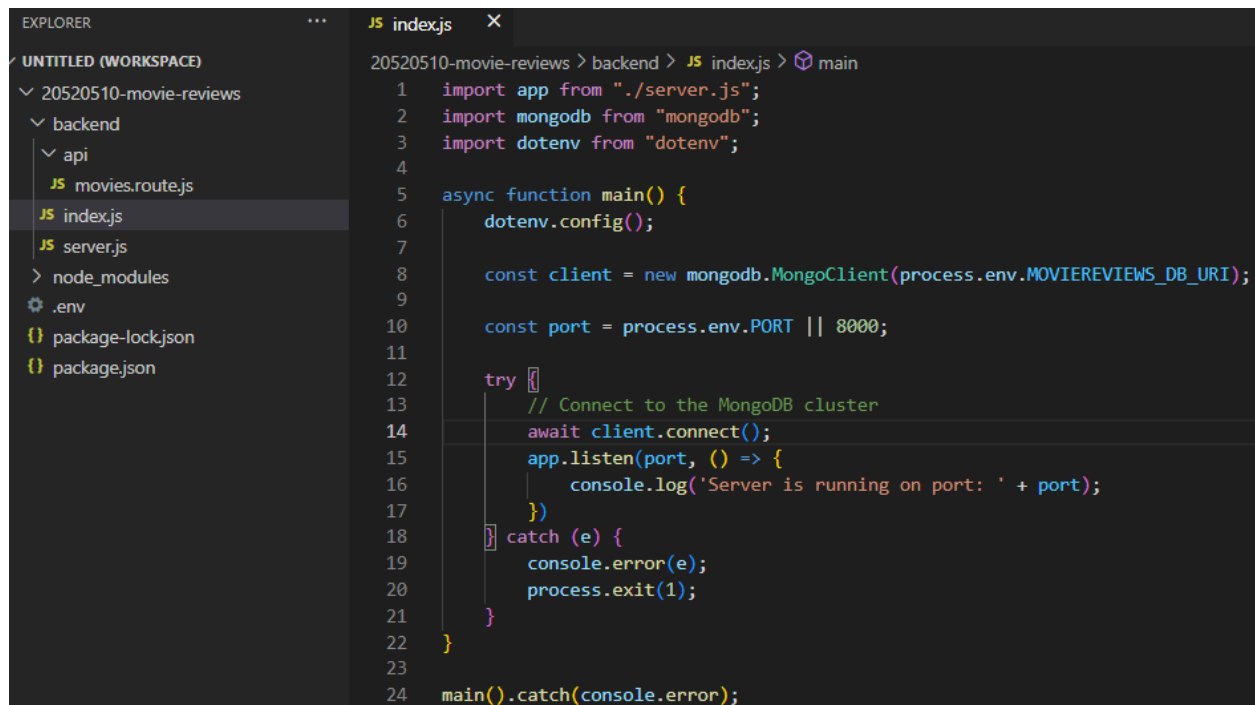
2.2 Tạo tệp tin .env để lưu trữ thông tin biến môi trường phát triển như URL kết nối tới DB trên MongoDB Atlas, PORT dịch vụ web, ví dụ 3000.



The screenshot shows the VS Code interface with the '.env' file selected in the Explorer sidebar. The main editor area displays the content of the '.env' file:

```
1 MOVIEREVIEW_DB_URI=mongodb+srv://20520510:123456@cluster0.syqhgiy.mongodb.net/?retryWrites=true&w=majority
2 MOVIEREVIEW_NS=sample_mflix
3 PORT=3000
```

2.3 Tạo tệp tin index.js để quản lý việc kết nối dữ liệu, khởi tạo đối tượng, và chạy máy chủ.

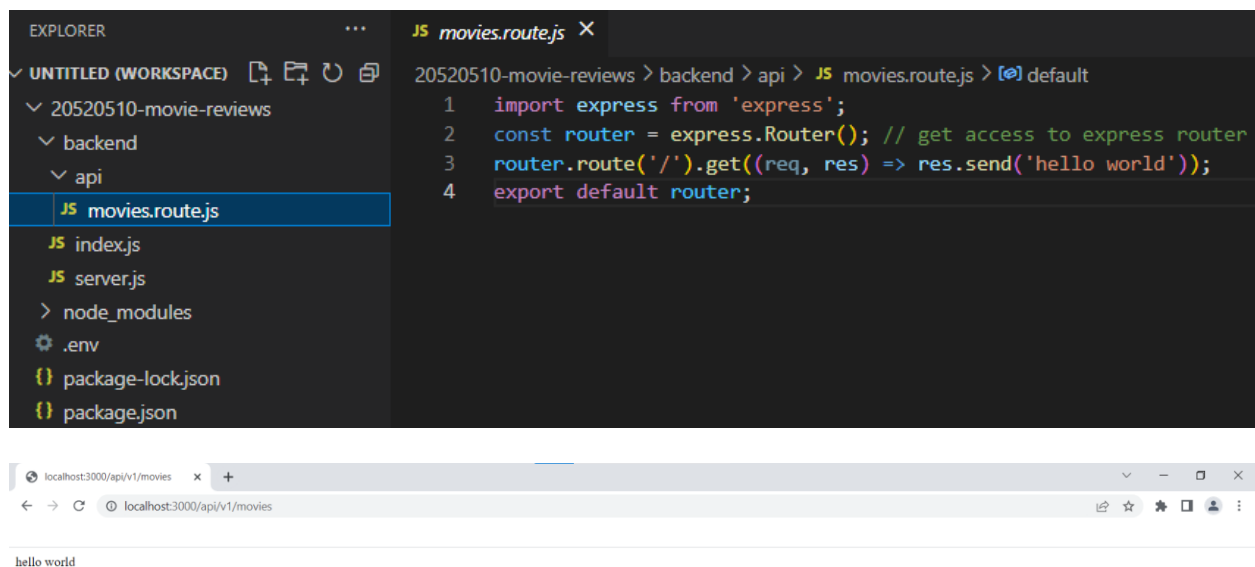


```
20520510-movie-reviews > backend > JS index.js > main
1  import app from "../server.js";
2  import mongodb from "mongodb";
3  import dotenv from "dotenv";
4
5  async function main() {
6      dotenv.config();
7
8      const client = new mongodb.MongoClient(process.env.MOVIEREVIEWS_DB_URI);
9
10     const port = process.env.PORT || 8000;
11
12     try {
13         // Connect to the MongoDB cluster
14         await client.connect();
15         app.listen(port, () => {
16             console.log('Server is running on port: ' + port);
17         })
18     } catch (e) {
19         console.error(e);
20         process.exit(1);
21     }
22 }
23
24 main().catch(console.error);
```

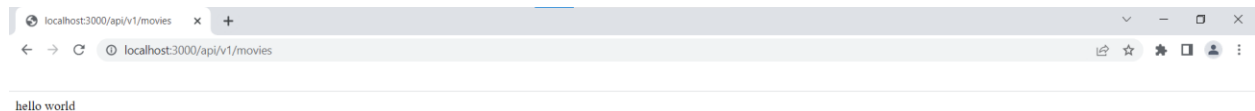
2.4 Tạo thư mục và tệp tin tương ứng trong thư mục backend gồm api/movies.route.js để xử lý các định tuyến liên quan đến ứng dụng minh họa movies về sau.

- Trong nội dung này, hiện tại để một định tuyến duy nhất '/' trả về cho máy khách thông báo 'hello world'.

Ví dụ máy khách truy cập: localhost:3000/api/v1/movies thì sẽ trả về 'hello world'.



```
20520510-movie-reviews > backend > api > JS movies.route.js > default
1  import express from 'express';
2  const router = express.Router(); // get access to express router
3  router.route('/').get((req, res) => res.send('hello world'));
4  export default router;
```

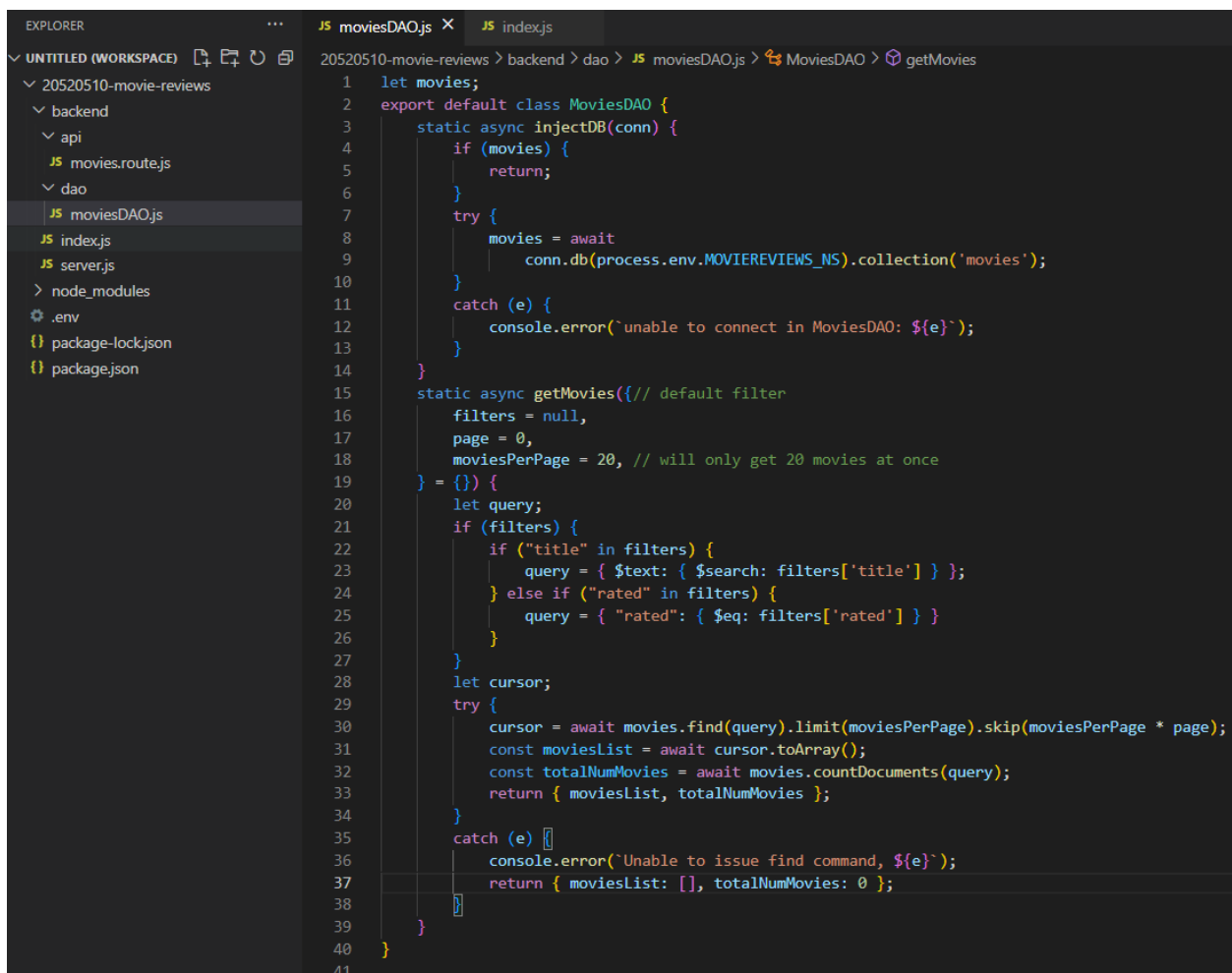


localhost:3000/api/v1/movies

hello world

2.5 Thiết lập công cụ truy xuất dữ liệu cho ứng dụng Movie với DAO – Data Access Object.

- Tạo thư mục dao trong thư mục backend, tạo tệp tin moviesDAO.js trong thư mục này.
- Tệp tin moviesDAO.js hiện tại sẽ bao gồm class MoviesDAO chứa 2 phương thức chính là:
 - injectDB(): dùng để tham chiếu tới dữ liệu collection movies trên sample_mflix.
 - getMovies(): để trả về danh sách các movies và số lượng các movies trả về thông qua 2 tham số: moviesList và totalNumMovies, với bộ lọc mặc định là: không có bộ lọc, bắt đầu từ trang 0, và mỗi trang có 20 phim là tối đa.



```

1  let movies;
2  export default class MoviesDAO {
3    static async injectDB(conn) {
4      if (movies) {
5        return;
6      }
7      try {
8        movies = await
9          conn.db(process.env.MOVIEREVIEWS_NS).collection('movies');
10     }
11     catch (e) {
12       console.error("unable to connect in MoviesDAO: ${e}");
13     }
14   }
15   static async getMovies({ // default filter
16     filters = null,
17     page = 0,
18     moviesPerPage = 20, // will only get 20 movies at once
19   } = {}) {
20     let query;
21     if (filters) {
22       if ("title" in filters) {
23         query = { $text: { $search: filters['title'] } };
24       } else if ("rated" in filters) {
25         query = { "rated": { $eq: filters['rated'] } }
26       }
27     }
28     let cursor;
29     try {
30       cursor = await movies.find(query).limit(moviesPerPage).skip(moviesPerPage * page);
31       const moviesList = await cursor.toArray();
32       const totalNumMovies = await movies.countDocuments(query);
33       return { moviesList, totalNumMovies };
34     }
35     catch (e) {
36       console.error("Unable to issue find command, ${e}");
37       return { moviesList: [], totalNumMovies: 0 };
38     }
39   }
40 }

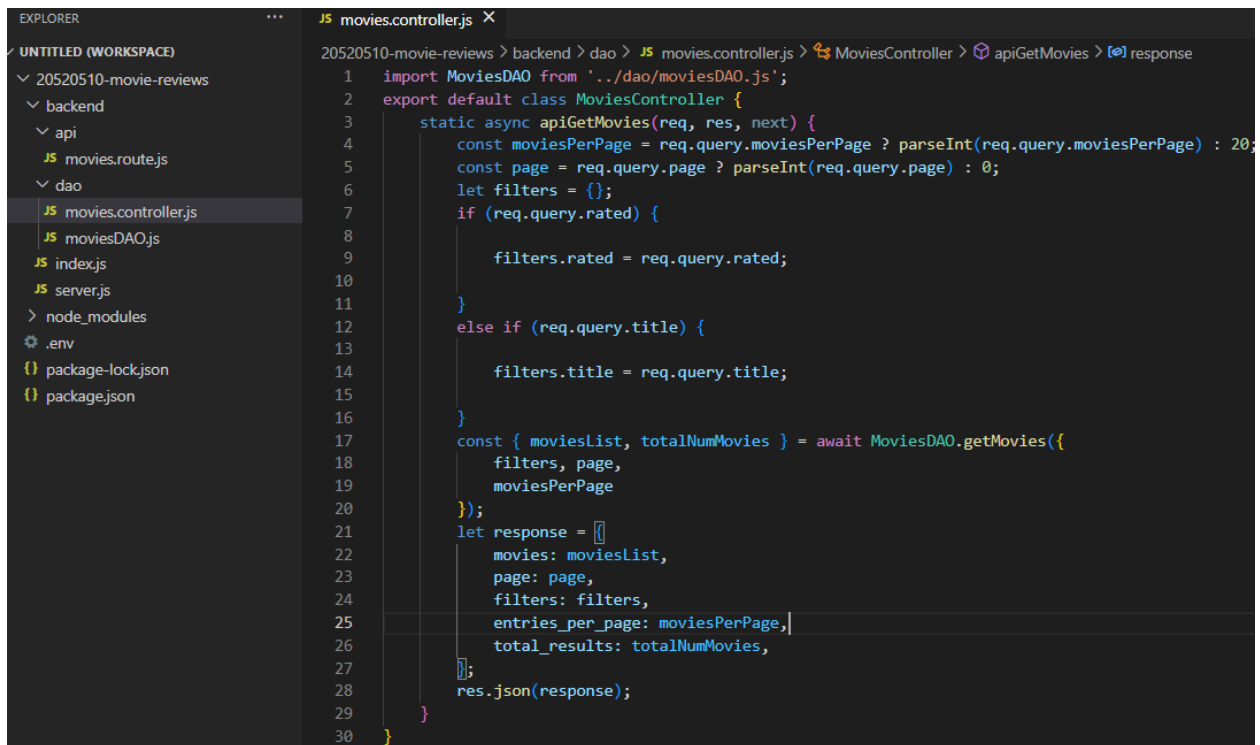
```

- Khởi tạo đối tượng của lớp MoviesDAO trong tệp tin index.js để sử dụng phương thức injectDB(). Phương thức này sẽ được gọi sau khi kết nối tới cơ sở dữ liệu trên MongoAtlas Cloud và trước khi máy chủ được chạy.

```
try {
  // Connect to the MongoDB cluster
  await client.connect();
  await MoviesDAO.injectDB(client);
}
```

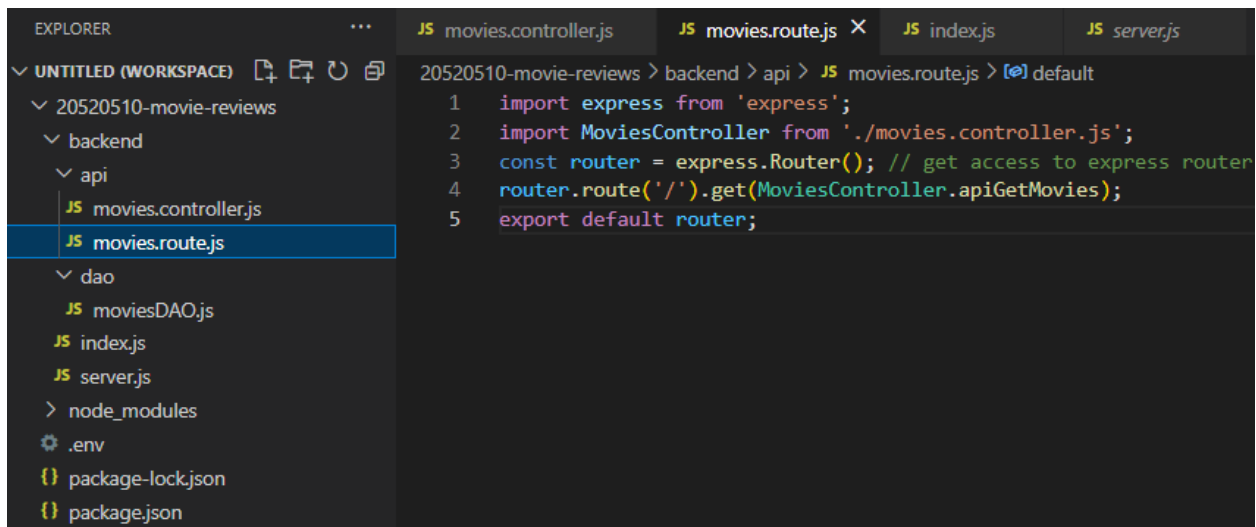
2.6 Thiết lập CONTROLLER cho ứng dụng web để gọi tới DAO. Tạo tệp tin movies.controller.js trong thư mục api để thực hiện tác vụ trung gian, tiếp nhận yêu cầu từ máy khách thông qua api (endpoint) sau đó định tuyến (route) tới function trong dao movies phù hợp.

- Trong class MoviesController tạo function apiGetMovies() trả về chuỗi json để gửi về cho máy khách. Trong function này sẽ có bước gọi tới hàm getMovies() đã định nghĩa trong dao Movie.



```
1 import MoviesDAO from '../dao/moviesDAO.js';
2 export default class MoviesController {
3   static async apiGetMovies(req, res, next) {
4     const moviesPerPage = req.query.moviesPerPage ? parseInt(req.query.moviesPerPage) : 20;
5     const page = req.query.page ? parseInt(req.query.page) : 0;
6     let filters = {};
7     if (req.query.rated) {
8
9       filters.rated = req.query.rated;
10    }
11
12    else if (req.query.title) {
13
14      filters.title = req.query.title;
15    }
16
17    const { moviesList, totalNumMovies } = await MoviesDAO.getMovies({
18      filters, page,
19      moviesPerPage
20    });
21    let response = {
22      movies: moviesList,
23      page: page,
24      filters: filters,
25      entries_per_page: moviesPerPage,
26      total_results: totalNumMovies,
27    };
28    res.json(response);
29  }
30 }
```

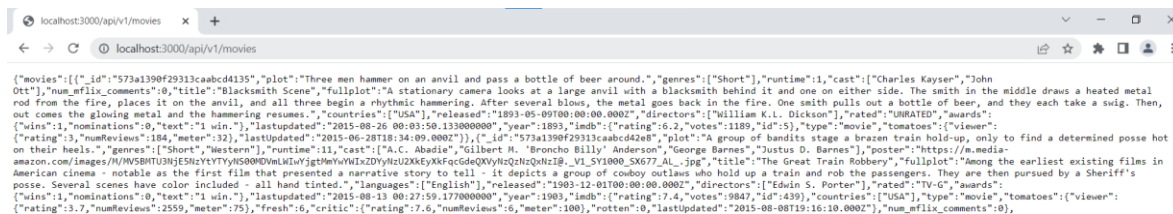
2.7 Đưa Controller vừa tạo ở yêu cầu 2.6 vào định tuyến.



The screenshot shows the VS Code interface. On the left, the Explorer sidebar shows a project structure with folders '20520510-movie-reviews', 'backend', and 'api'. The 'api' folder is expanded, showing 'movies.controller.js' and 'movies.routes.js'. The 'movies.routes.js' file is selected and its content is displayed in the main editor. The code defines an Express router for the '/api/v1/movies/' endpoint, using the 'MoviesController' to handle GET requests.

```
1 import express from 'express';
2 import MoviesController from './movies.controller.js';
3 const router = express.Router(); // get access to express router
4 router.route('/').get(MoviesController.apiGetMovies);
5 export default router;
```

Ví dụ: Khi máy khách gửi một http request như sau: localhost:3000/api/v1/movies/ thì máy chủ web sẽ gọi hàm apiGetMovies trong MoviesController để xử lý yêu cầu này và trả về thông tin trên browser như hình:



Thử nghiệm API trên postman:

