MỤC TIÊU:

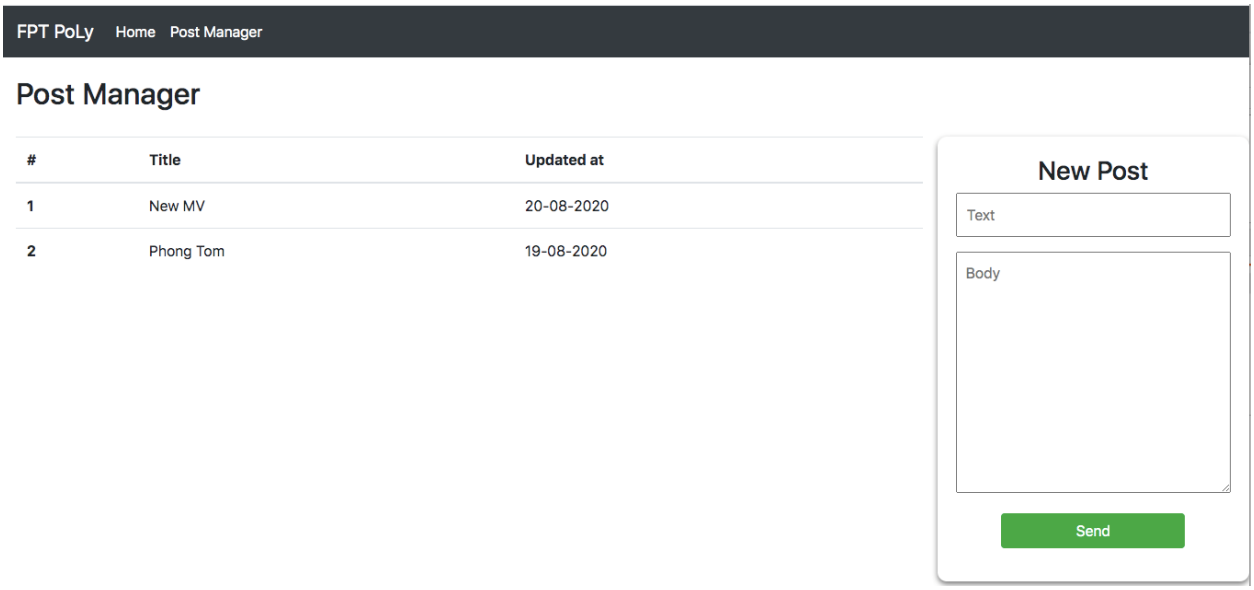
Kết thúc bài thực hành này bạn có khả năng

- ✓ Biết được tại sao phải dùng REST API
- ✓ Hiểu cấu trúc Rest API
- ✓ Tạo được REST API
- ✓ Test Rest API

PHẦN I

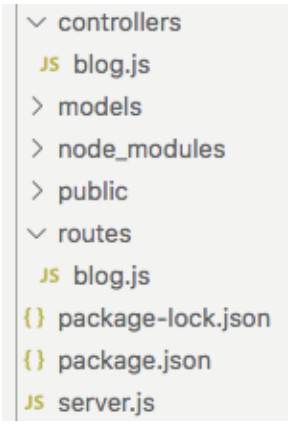
Bài 1

Xây dựng RestFul API cho trang quản lý blog có giao diện như sau



Bước 1: Tổ chức thư mục và phân tích REST API cho ứng dụng

HTTP method	Route	Body	Access
GET	{host}/blog/posts	Không	Public
POST	{host}/blog/post	Có	Admin



Bước 2: Controller blog chứa các hàm API:

```
exports.getPosts = (req, res, next) => {
  res.status(200).json({
    posts: [{ title: 'First Post', content: 'This is the first post!' }]
  });
};
```

```
exports.createPost = (req, res, next) => {
  const title = req.body.title;
  const content = req.body.content;
  res.status(201).json({
    message: 'Post created successfully!',
    post: { id: new Date().toISOString(), title: title, content: content }
  });
};
```

Bước 3: Định nghĩa Route cho blog

```
onst express = require('express');

const blogController = require('../controllers/blog');

const router = express.Router();

// GET /blog/posts
router.get('/posts', blogController.getPosts);

// POST /blog/post
router.post('/post', blogController.createPost);

module.exports = router;
```

Phần server: sử dụng hệ thống route đã xây dựng bước trên

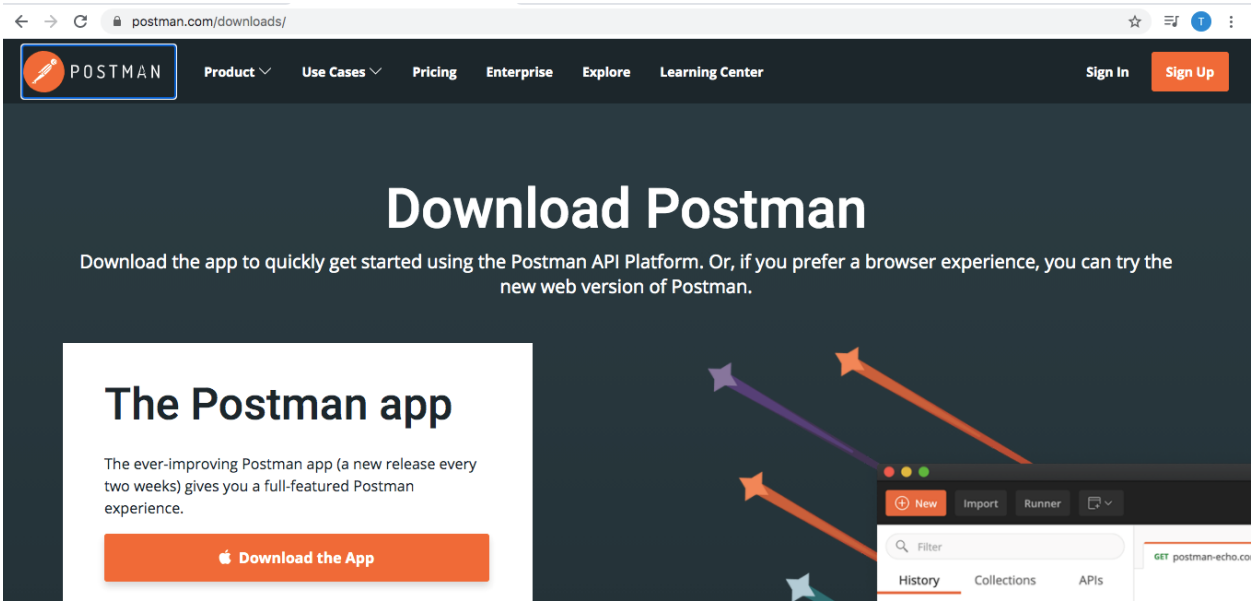
```
const express = require('express');
const bodyParser = require('body-parser');
```

```
const blogRoutes = require('./routes/blog');

const app = express();
app.use(bodyParser.json()); // application/json
const port=3000;
app.use('/blog', blogRoutes);

app.listen(port,()=>{
  console.log(`ứng dụng đang chạy với port: ${port}`);
})
```

Bài 2

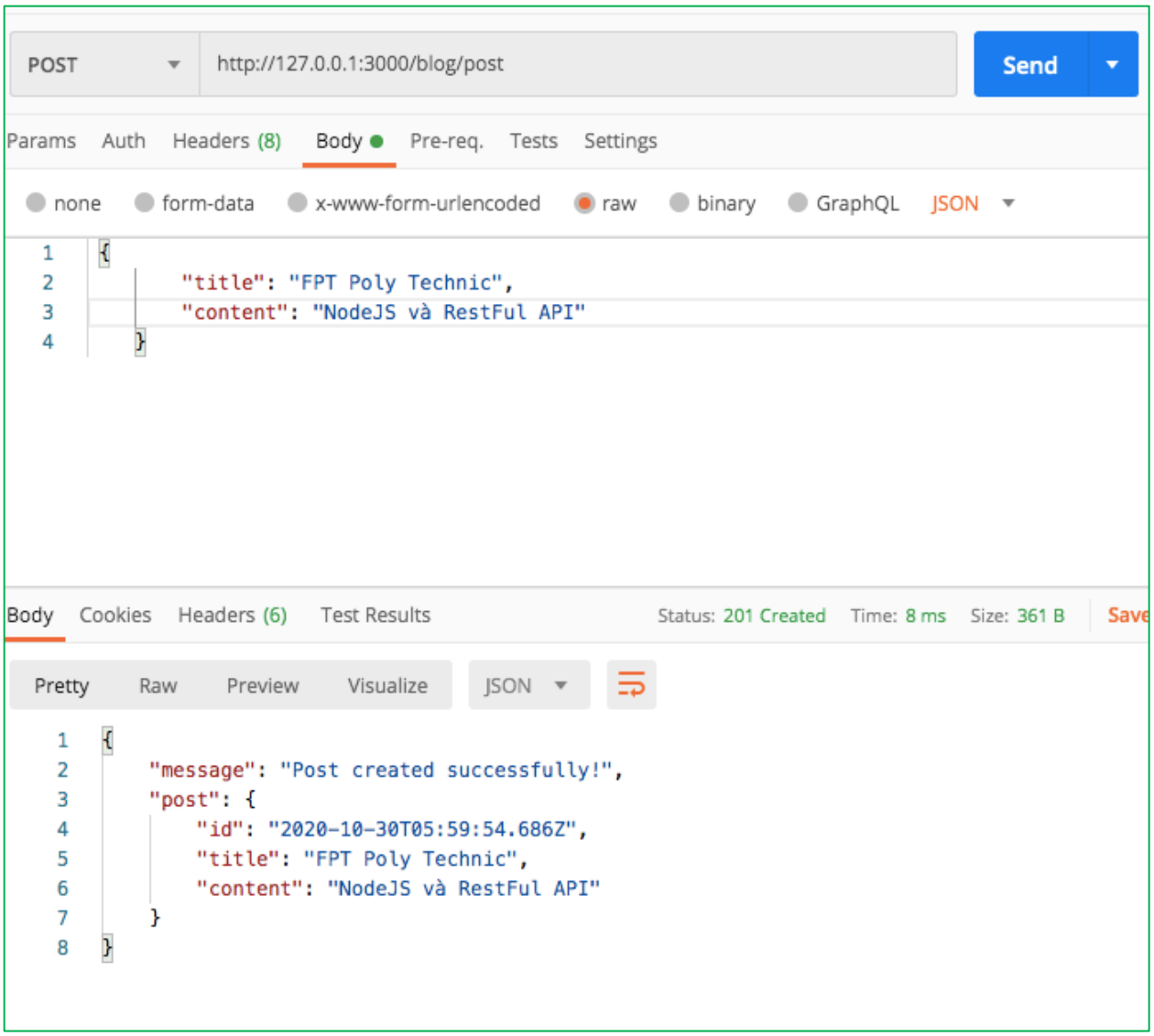
Test REST API ở câu 1 với công cụ Postma

Bước 1: Tải và cài đặt công cụ postman



Bước 2: Test với phương thức GET của API

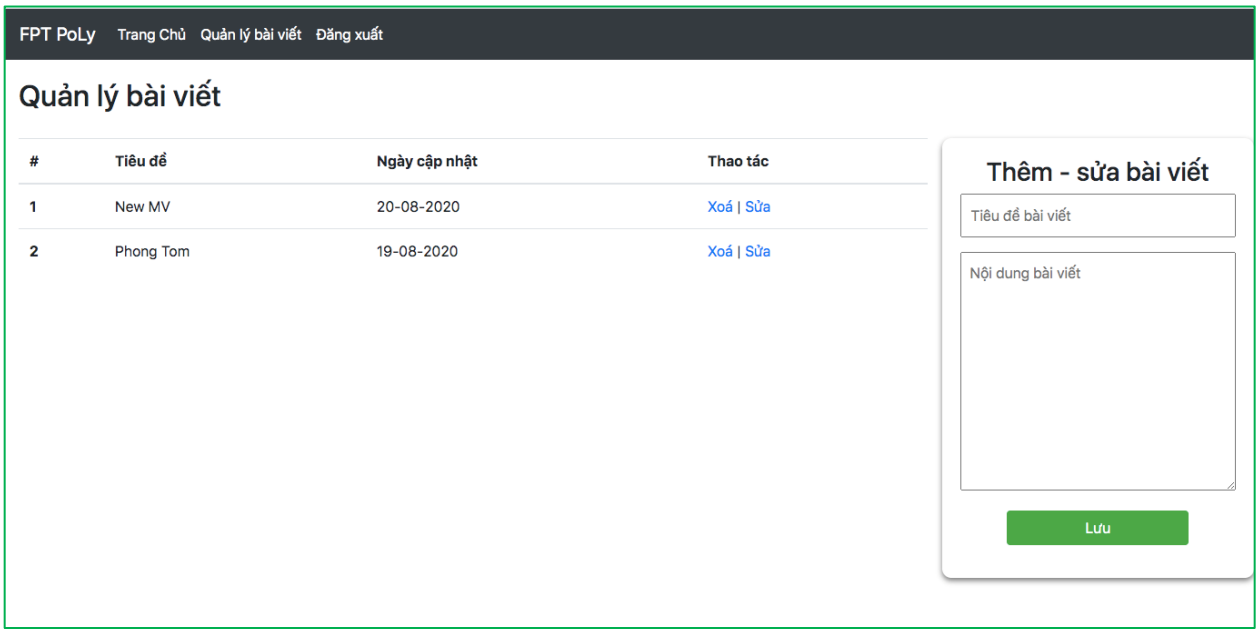
Bước 2: Test với phương thức POST của API


POST
http://127.0.0.1:3000/blog/post
Send
Params Auth Headers (8) Body Pre-req. Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON
{
"title": "FPT Poly Technic",
"content": "NodeJS và RestFul API"
}
Body Cookies Headers (6) Test Results
Status: 201 Created Time: 8 ms Size: 361 B
Save
Pretty Raw Preview Visualize JSON
{
"message": "Post created successfully!",
"post": {
"id": "2020-10-30T05:59:54.686Z",
"title": "FPT Poly Technic",
"content": "NodeJS và RestFul API"
}
}

PHẦN II

Bài 3

Xây dựng RESTFUL API tương ứng cho trang quản lý bài viết (post) của 1 blog có giao diện như sau



Bước 1: Phân tích và tạo cơ sở dữ liệu: sinh viên tự xây dựng

Bước 2: Thiết kế REST API

HTTP method	**Route**	**Body**	**Access**
GET	{host}/blog/posts	Không	Public
GET	{host}/blog/posts/:id	Không	Public
POST	{host}/blog/posts	Có	Admin
PUT	{host}/blog/posts/:id	Có	Admin
DELETE	{host}/blog/posts/:id	Không	Admin

Tương ứng với routing như sau:

Lab05_2 > routes > JS blog.js > ...

```
1   const express = require('express');
2
3   const blogController = require('../controllers/blog');
4
5   const router = express.Router();
6
7   // GET /blog/posts
8   router.get('/posts', blogController.getPosts);
9   router.get('/posts/:postId', blogController.getPostById);
10  // POST /blog/post
11  router.post('/posts', blogController.createPost);
12  //update
13  router.put('/posts/:postId', blogController.deletePost);
14  //delete
15  router.delete('/posts/:postId', blogController.updatePost);
16
17  module.exports = router;
```

Bước 3: Xây dựng REST API

- Tạo model:

Cài đặt sequelize để thực hiện mô hình ORM như sau

Lab05_2 > util > JS database.js > ...

```
1   const Sequelize = require('sequelize');
2
3   const sequelize = new Sequelize('blogDB', 'root', '', {
4     dialect: 'mysql',
5     host: 'localhost'
6   });
7
8   module.exports = sequelize;
```

Lab05_2 > models > JS post.js > ...

```
1   const Sequelize = require('sequelize');
2
3   const sequelize = require('../util/database');
4
5   const Post = sequelize.define('tblPost', {
6     postId: {
7       type: Sequelize.INTEGER,
8       autoIncrement: true,
9       allowNull: false,
10      primaryKey: true
11    },
12    title: Sequelize.STRING,
13    content: {
14      type: Sequelize.STRING,
15      allowNull: false
16    },
17    create_date: {
18      type: Sequelize.DATE,
19      allowNull: false,
20    }
21  },
22  { timestamps: false }
23  );
24  module.exports = Post;
```

Lab05_2 > JS server.js > ...

```
1   const express = require('express');
2   const bodyParser = require('body-parser');
3   const blogRoutes = require('./routes/blog');
4   const sequelize = require('./util/database');
5
6   const app = express();
7   app.use(bodyParser.json()); // application/json
8   const port=3000;
9   app.use('/blog', blogRoutes);
10
11  sequelize
12    .sync()
13    .then(result => {
14      // console.log(result);
15      app.listen(port,()=>{
16        console.log(`ứng dụng đang chạy với port: ${port}`);
17      })
18    })
19    .catch(err => {
20      console.log(err);
21    });
```

Bước 4: Viết API cho các phương thức GET của HTTP

{{URL}}/Blog/posts: Trả về tất cả các bài viết hiện hành

Lab05_2 > controllers > JS blog.js > ...

```
1   const Post = require('../models/post');
2   exports.getPosts = (req, res, next) => {
3     Post.findAll()
4       .then(posts => {
5         res
6           .status(200)
7           .json({ message: 'Fetched posts successfully.', posts: posts });
8       })
9       .catch(err => {
10        if (!err.statusCode) {
11          err.statusCode = 500;
12        }
13        next(err);
14      });
15  };
```

{{URL}}/Blog/posts/:postId: Trả về bài viết có mã là postId

Lab05_2 > controllers > JS blog.js > ...

```
17  exports.getPostById = (req, res, next) => {
18    const postId = req.params.postId;
19    Post.findByPk(postId)
20      .then(post => {
21        if (!post) {
22          const error = new Error('Không tìm thấy bài viết- post.');
23          error.statusCode = 404;
24          throw error;
25        }
26        res.status(200).json({ message: 'Post được tìm thấy.', post: post });
27      })
28      .catch(err => {
29        if (!err.statusCode) {
30          err.statusCode = 500;
31        }
32        next(err);
33      });
34  };
```

Bài 4

Viết các REST API để thay đổi dữ liệu trong database

- POST: Thực hiện thêm một bài viết mới

Lab05_2 > controllers > JS blog.js > ...

```
36  exports.createPost = (req, res, next) => {
37    const title = req.body.title;
38    const content = req.body.content;
39    const post = new Post({ title: title, content: content,create_date: new Date().toISOString() });
40    post
41      .save()
42      .then(result => {
43        res.status(201).json({
44          message: 'Thêm thành công bài viết mới!',
45          post: result
46        });
47      })
48      .catch(err => {
49        if (!err.statusCode) {
50          err.statusCode = 500;
51        }
52        next(err);
53      });
54  };
```

- PUT: Thực hiện cập nhật một bài viết theo mã

Lab05_2 > controllers > JS blog.js > ...

```
56  exports.updatePost = (req, res, next) => {
57    const postId = req.params.postId;
58    const title = req.body.title;
59    const content = req.body.content;
60
61    Post.findByPk(postId)
62      .then(post => {
63        if (!post) {
64          const error = new Error('Không tim thấy bài viết - post.');
65          error.statusCode = 404;
66          throw error;
67        }
68        post.title = title;
69        post.content = content;
70        return post.save();
71      })
72      .then(result => {
73        res.status(200).json({ message: 'Post update thành công!', post: result });
74      })
75      .catch(err => {
76        if (!err.statusCode) {
77          err.statusCode = 500;
78        }
79        next(err);
80      });
81  };
```

- DELETE: Xoá 1 bài post theo mã bài

Lab05_2 > controllers > JS blog.js > ...

```
83  exports.deletePost = (req, res, next) => {
84    const postId = req.params.postId;
85    Post.findByPk(postId)
86      .then(post => {
87        if (!post) {
88          const error = new Error('Không tim thấy bài viết - post.');
89          error.statusCode = 404;
90          throw error;
91        }
92        return post.destroy(postId);
93      })
94      .then(result => {
95        console.log(result);
96        res.status(200).json({ message: 'Đã xoá post.' });
97      })
98      .catch(err => {
99        if (!err.statusCode) {
100         err.statusCode = 500;
101       }
102       next(err);
103     });
104 };
```

Bài 5

Test REST API: sinh viên thực hiện test tất cả các API đã thực hiện ở bài 3,4 với công cụ postman.

*** Yêu cầu nộp bài:

SV nén file (*hoặc share thư mục google drive*) bao gồm các yêu cầu đã thực hiện trên, nộp LMS đúng thời gian quy định của giảng viên. KHÔNG NỘP BÀI COI NHƯ KHÔNG CÓ ĐIỂM.

--- Hết ---