TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

----------o0o----------

BÁO CÁO MÔN PROJECT III

Đề tài: Xây dựng CMS với Nodejs

Giáo viên hướng dẫn: Nguyễn Tuấn Dũng

Sinh viên thực hiện: Mạc Văn Thiêm

Hà Nội, 2021

Mục lục

1. Giới thiệu đề tài

1.1. Đề tài

Xây dựng CMS (**C**ontent **M**anagement **S**ystem) trên nền tảng Nodejs.

CMS là chữ viết tắt của **C**ontent **M**anagement **S**ystem. Còn gọi là hệ thống quản trị nội dung nhằm mục đích giúp dễ dàng quản lý, chỉnh sửa nội dung. Nội dung ở đây là text, video, nhạc, hình ảnh, files… CMS là nơi người quản trị Website có thể cập nhật, thay đổi nội dung trên Website. Một hệ thống CMS tốt sẽ cho phép vận hành Website mà không cần sự can thiệp, hỗ trợ từ người lập trình trang web.

Hệ thống CMS giúp tiết kiệm thời gian quản lý, chi phí vận hành và bảo trì nên hiện nay có rất nhiều công ty sử dụng. Không chỉ là công ty mà hiện nay các blog cá nhân cũng ra đời nhiều, giải pháp sử dụng CMS giúp dễ dàng xây dựng website và quản lý nội dung. Bên cạnh đó còn tiết kiệm được chi phí xây dựng website.

Trong project này, chúng tôi xây dựng một CMS cho blog cá nhân.

1.2. Công nghệ sử dụng

1.2.1. Nodejs

Nodejs là một mã nguồn được xây dựng dựa trên nền tảng Javascript V8 Engine, nó được sử dụng để xây dựng các ứng dụng web như các trang video clip, các forum và đặc biệt là trang mạng xã hội phạm vi hẹp. Nodejs là một mã nguồn mở được sử dụng rộng bởi hàng ngàn lập trình viên trên toàn thế giới.

Nodejs có thể chạy trên nhiều nền tảng hệ điều hành khác nhau từ Window cho tới Linux, OS X nên đó cũng là một lợi thế. Nodejs cung cấp các thư viện phong phú ở dạng Javascript Module khác nhau giúp đơn giản hóa việc lập trình và giảm thời gian ở mức thấp nhất.

Các đặc tính của Nodejs:

- Không đồng bộ: Tất cả các API của Nodejs đều không đồng bộ (*none-blocking*).
- Chạy rất nhanh: Nodejs được xây dựng dựa vào nền tảng V8 Javascript Engine nên việc thực thi chương trình rất nhanh.
- Đơn luồng nhưng khả năng mở rộng cao: Node.js sử dụng một mô hình luồng duy nhất với sự kiện lặp.
- Không đệm: Nodejs không đệm bất kỳ một dữ liệu nào và các ứng dụng này chủ yếu là đầu ra dữ liệu.
- Có giấy phép: Nodejs đã được cấp giấy phép bởi MIT License.

Hai Nodejs framework sử dụng phổ biến:

Express

Expressjs là một trong những framework phổ biến dùng để xây dựng API và Website phổ biến nhất của Nodejs. Nó được sử dụng rộng rãi đến mức hầu như mọi dự án Web nào đều bắt đầu bằng việc tích hợp Express. Có rất nhiều lý do để chọn Expressjs:

- Có nhiều tính năng hỗ trợ tất cả những gì bạn cần trong việc xây dựng Web và API
- Quản lý các route dễ dàng
- Cung cấp một nền tảng phát triển cho các API
- Hỗ trợ nhiều thư viện và plugin
- Bảo mật và an toàn hơn so với việc code thuần

SocketIO

SocketIO là một web-socket framework có sẵn cho nhiều ngôn ngữ lập trình.

Trong NodeJS, SocketIO cho phép xây dựng một các ứng dụng realtime như chatbot, tickers, dashboard APIs, và nhiều ứng dụng khác. SocketIO có lợi ích hơn so với NodeJS thông thường.

- Hỗ trợ route URL tùy chỉnh cho web socket
- Tích hợp dễ dàng hơn với Express JS
- Hỗ trợ clustering với Redis

Trong project này, chúng tôi sử dụng Express để xây dựng server cho hệ thống.

1.2.2. MongoDB

MongoDB là một hệ quản trị cơ sở dữ liệu mã nguồn mở thuộc học NoSQL. Nó được thiết kế theo kiểu hướng đối tượng, các bảng trong MongoDB được cấu trúc rất linh hoạt, cho phép các dữ liệu lưu trữ trên bảng không cần tuân theo một cấu trúc nhất định nào cả (điều này rất thích hợp để làm big data).

MongoDB lưu trữ dữ liệu theo hướng tài liệu (document), các dữ liệu được lưu trữ trong document kiểu JSON nên truy vấn sẽ rất nhanh.

Ưu điểm của MongoDB:

- Schema linh hoạt: Do MongoDB sử dụng lưu trữ dữ liệu dưới dạng Document Json nên mỗi một collection sẽ các các kích cỡ và các document khác nhau.
- Cấu trúc đối tượng rõ ràng: Tuy rằng cấu trúc của dữ liệu là linh hoạt nhưng đối tượng của nó được xác định rất rõ ràng. Sử dụng bộ nhớ nội tại, nên truy vấn sẽ rất nhanh.
- MongoDB rất dễ mở rộng.
- Không có các join: Điều này cũng góp phần tạo nên tốc độ truy vấn cực nhanh trên mongoDB.
- MongoDB phù hợp cho các ứng dụng realtime.

Trong project này, chúng tôi sử dụng MongoDB để xây dựng cơ sở dữ liệu cho hệ thống.

1.2.3. Một số công nghệ khác

1. Bootstrap 4
2. AJAX
3. Handlebars.js

1.3. Các chức năng của hệ thống

- Người dùng
    - Đăng nhập/đăng xuất/đăng ký
    - Xem toàn bộ bài viết
    - Xem chi tiết bài viết
    - Bình luận bài viết
    - Tìm kiếm/Tìm kiếm theo thể loại
- Quản trị viên
    - Các chức năng của người dùng
    - Thêm/sửa/xóa bài viết
    - Thêm/sửa/xóa thể loại bài viết
    - Quản lý nội dung bình luận
    - Cập nhật trạng thái các đối tượng

2. Xây dựng Project

Khởi tạo project:

```
$ npm int
```

Tại đây, chúng ta có thể thêm các thông tin của project:

```
{
  "name": "cms_tutorial",
  "version": "1.0.1",
  "description": "Nodejs CMS Application",
  "main": "app.js",
  ▷ Debug
  "scripts": {…
  },
  "author": "thiemmv",
  "license": "ISC",
  "dependencies": {…
  },
  "devDependencies": {…
  }
}
```

File *app.js* là nơi bắt đầu thực hiện chương trình.

Cài đặt và sử dụng package *nodemon* để tự khởi động lại server sau mỗi lần cập nhật code, hỗ trợ cho việc xây dựng hệ thống.

2.1. Xây dựng Server

Express
Cài đặt Express:

```
$ npm install express
```

Gọi package *express*, khởi tạo đối tượng, thiết lập router, cổng lắng nghe (3000) cho hệ thống:

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

app.get('/', function (req, res) {
  res.send('Hello World')
})

app.listen(3000)
```

Khởi chạy chương trình:

```
$ npm start
```

Mở trình duyệt và nhập url: *http://localhost:3000*

Tuy nhiên, do hệ thống cần xây dựng có rất nhiều yêu cầu nên chúng ta cần xây dựng file route riêng cho hệ thống để tránh cho file *app.js* quá dài. Package *express* cung cấp một bộ định tuyến giúp cho chúng ta định danh ra các url và hành động kèm theo nó.
Router trong *express* cung cấp hai phương thức là *get* và *post.* Tuy nhiên như vậy là chưa đủ (*put, patch, delete*), trong phạm vi project này chúng ta cần cung cấp thêm phương thức *delete* để thực hiện gọi xóa các bản ghi. Sử dụng package *method-override* để thực hiện công việc đó.

```
$ npm install method-override
```

Trong project này, chúng ta tạo một folder *routes* có hai file là *admin.js* và *default.js* để định tuyến. *admin.js* để định tuyến các hoạt động của quản trị viên. *default.js* để định tuyến các hoạt động của người dùng nói chung. Ứng với mỗi url bộ định tuyến sẽ gọi đến các phương thức tương ứng trong controller tương ứng để xử lý yêu cầu.

```
const express = require('express');
const router = express.Router();
const adminController = require('../controllers/AdminController');

function isUserAuthenticated(req, res, next) {…
}

router.all('/*', isUserAuthenticated, (req, res, next) => {
    req.app.locals.layout = 'admin';
    next();
})

router.route('/')
    .get(adminController.index);

router.route('/prepare')
    .get(adminController.prepareData);

router.route('/posts')
    .get(adminController.getPosts);

router.route('/posts/create')
    .get(adminController.createPost)
    .post(adminController.storePost);

router.route('/posts/edit/:id')
    .get(adminController.editPost);
```

Controller
Trong controller, chúng ta xây dựng các phương thức để thực hiện các chức yêu cầu của hệ thống.

```
const Post = require('../models/Post').Post;
const Category = require('../models/Category').Category;
const Comment = require('../models/Comment').Comment;
const User = require('../models/User').User;
const { backgroundColor, borderColor } = require('../config/configuration');

module.exports = {

    // Dashboard
    index: async (req, res) => {
        let count_cats = await Category.countDocuments();
        let count_posts = await Post.countDocuments();
        let count_users = await User.countDocuments();
        let count_cmts = await Comment.countDocuments();

        let posts = await Post.find().populate('category').populate('user').limit(5).lean();

        let users = await User.find().limit(5).lean();

        res.render('admin/index', {count_cats: count_cats, count_posts: count_posts, count_users: count_users, count_cmts: count_cmts, posts: post
    },
```

Chúng ta sử dụng thêm một vài package khác như:

- *bcrypt*: mã hóa mật khẩu người dùng.
- *fs*: xử lý file và thư mục trong Nodejs.
- *express-fileupload*: tải file ảnh từ local lên server.
- *express-session*: lưu phiên sử dụng của người dùng, khi họ đăng nhập vào hệ thống.
- *flash*: lưu trữ và hiển thị thông báo trong hệ thống.
- *passport*: cơ chế xác thực người dùng

Trong quá trình xử lý các yêu cầu, controller cần tương tác với cơ sở dữ liệu. Để làm việc này, các controller sẽ tương tác với các model tương ứng.

Model

Sử dụng package *mongoose,* tạo ra các *schema* - nơi ta định nghĩa các trường dữ liệu cho một *document.* Ví dụ dưới đây là *model* cho *collection* category.

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const CategorySchema = new Schema({

    title: {
        type: String,
        required: true
    }

});

module.exports = {Category: mongoose.model('category', CategorySchema)};
```

Ngoài ra, chúng ta sử dụng package *slugify* để tạo đường dẫn cho mỗi đối tượng khi hiển thị trên thanh url. Thay vì hiển thị *id* của đối tượng thì *slug* của chúng sẽ hiện ra. Mục đích tạo đường dẫn thân thiện.

2.2. Xây dựng và sử dụng Cơ sở dữ liệu

Tạo một cơ sở dữ liệu, thiết lập tên của chúng trong file *configuration.js* để server kết nối đến chúng.

Việc ta xây dựng các Model trong phần trên là đã xây dựng các *collections* (bảng) cho cơ sở dữ liệu.

Có thể sử dụng terminal để làm quen với truy vấn trong mongoDB:

CSDL	MySQL	MongoDB
Tạo csdl	CREATE DATABASE test;	use test;
Tạo bảng	CREATE TABLE students (ten_cot - kieu_du_lieu);	db.createCollection('students');
Tạo bản ghi	INSERT INTO studetns ('name', 'gender') VALUES('thanh', 'male');	db.students.insert({ name:'thanh', gender: 'male'});
Cập nhật	UPDATE students SET name = 'thanh update' WHERE id = 1;	db.students.update({ _id: 1 },{$set:{ name: 'thanh update' }});
Xóa bản ghi	DELETE FROM students Where id = 1;	db.students.remove({ _id: 1});
Tìm kiếm all	SELECT * FROM students;	db.students.find({});
Tìm kiếm	SELECT * FROM students WHERE name = 'thanh';	db.students.find({ name: 'thanh' });

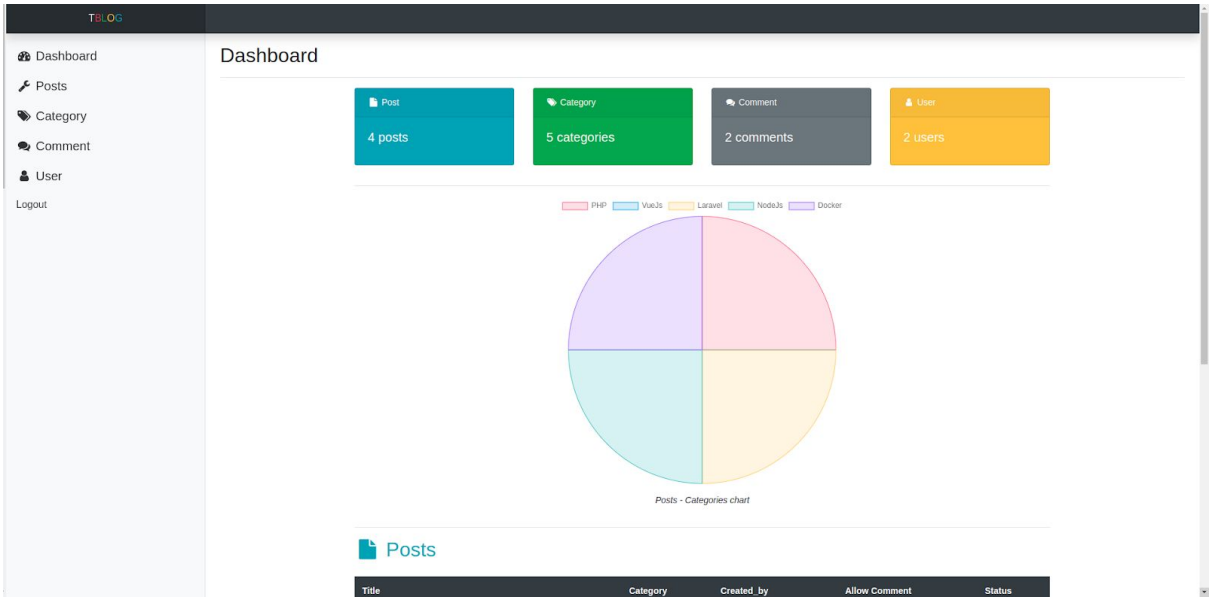
2.3. Giao diện

Toàn bộ giao diện của hệ thống được xây dựng bằng *handlebars.* Handlebar.js là một thư viện *javascript* rất mạnh mẽ hỗ trợ bạn có thể *binding data* vào một *template* để hiển thị ra

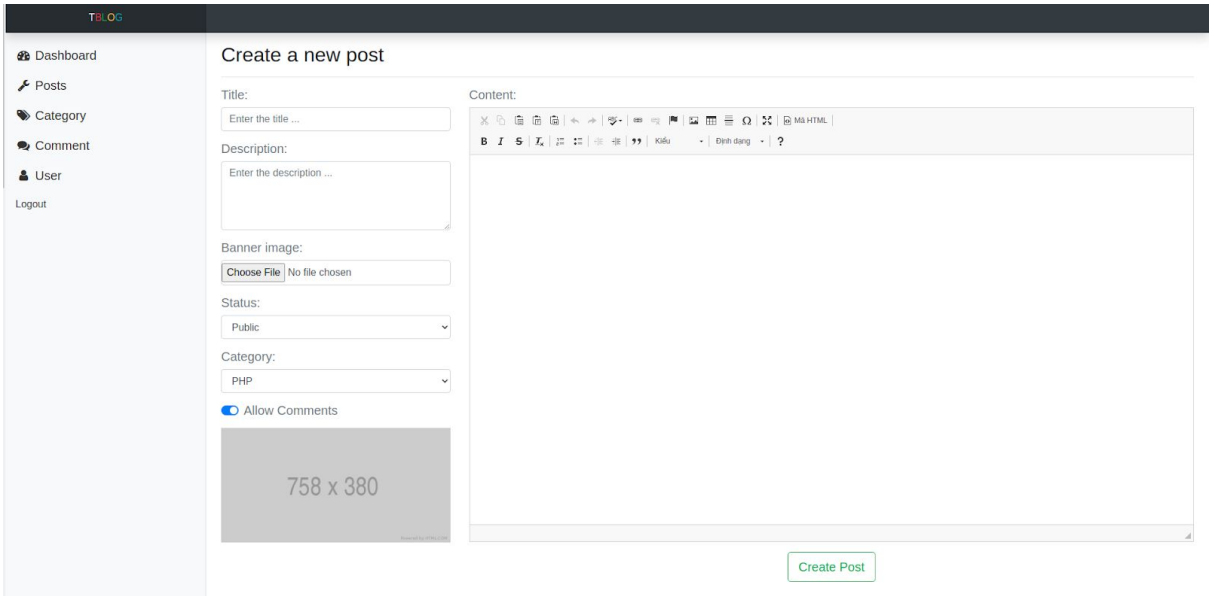
website. Chúng cho phép chia một màn hình thành các phần (*header, navigation, sidebar, footer, message, content, . . .*) chúng ta có thể lắp ghép các phần để tạo được một trang *html* hoàn chỉnh, cũng qua đó chúng ta có thể tái sử dụng lại các phần dùng chung, chỉ thay đổi các phần hiển thị nội dung tương ứng.

Sử dụng framework bootstrap để tùy chỉnh giao diện cho hệ thống.

Sử dụng thư viện *chart.js* để vẽ biểu đồ cho trang quản trị.



ckeditor là một trình soạn thảo mã nguồn mở theo kiểu WYSIWYG của CKSource. Chương trình này có thể tích hợp vào các website mà không cần cài đặt. Sử dụng *ckeditor* để cung cấp cho người dùng một giao diện soạn thảo nội dung của một bài viết.



3. Kết luận

Hệ thống CMS trên được xây dựng trên nền tảng Nodejs có sử dụng một số package phổ biến hiện nay, cơ sở dữ liệu MongoDB để lưu trữ, sử dụng *handlebars* - một thư viện *javascript* rất mạnh mẽ hỗ trợ việc *binding data* vào một *template* để hiển thị ra website. Qua project trên, em đã học hỏi được một công nghệ mới, rất phổ biến và hữu ích. Tuy nhiên, trong bài vẫn còn những thiếu sót, rất mong nhận được sự nhận xét của thầy.

Tài liệu tham khảo

1. https://www.dtmi.net/mapping-giua-sql-va-mongodb/
2. https://www.npmjs.com/
3. https://expressjs.com/en/starter/hello-world.html
4. https://handlebarsjs.com/guide/
5. https://docs.mongodb.com/