



Міністерство освіти і науки України
Національний технічний університет України „КПІ імені Ігоря
Сікорського ”

Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Звіт до комп'ютерного практикуму №5

З дисципліни «Основи Back-end технологій»

Прийняв:

Викладач

пос. Зубко Р. А.

«08» квітня 2024 р.

Виконала:

Студентка 3 курсу, гр. ІМ-11

Бащак Ярина

2024 р.

Лабораторна робота №5.

NodeJS. Робота з БД MongoDB. Додаток, що реалізує CRUD операції в БД.

Завдання.

- Створити додаток, що реалізує CRUD операції з БД – додавання, читання, редагування та видалення записів БД.
- Забезпечити роутінг запитів та виведення результатів запитів на WEB-сторінку.
- Додати новий роут для виведення інформації у вигляді json-файлу.

Хід роботи

1. Створюємо базу даних MongoDB на віддаленому сервері MongoDB Atlas. Підключаємось до неї через відповідні налаштування у файлі server.js:

```
const express = require('express')
const mongoose = require('mongoose')
const { engine } = require('express-handlebars')
const postRoutes = require('./routes/postRoutes')
const moment = require('moment')
const app = express()
const port = 3000

app.engine('hbs', engine({
  extname: '.hbs',
  defaultLayout: 'main',
  runtimeOptions: {
    allowProtoPropertiesByDefault: true,
    allowProtoMethodsByDefault: true
  },
  helpers: {
    formatDate: function (date, format) {
      return moment(date).format(format)
    }
  }
}))
app.set('view engine', 'hbs')
app.set('views', './views')

app.use(express.json())
app.use(express.urlencoded({ extended: true }))
app.use('/posts', postRoutes)

mongoose
  .connect(
    'mongodb+srv://xxx:xxx@cluster0.srk1crx.mongodb.net/?retryWrites=true&w=majority&appName=cluster0'
  )
  .then(() => console.log('Successfully connected to MongoDB'))
  .catch(err => console.error('Connection error', err))
```

```
app.listen(port, () => {
  console.log(`Server is running on port ${port}`)
})
```

2. Створюємо модель посту, CRUD операції над ним та всі необхідні роути:

Post.js

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

const postSchema = new mongoose.Schema({
  title: String,
  author: String,
  text: String,
  createdAt: { type: Date, default: Date.now },
  updatedAt: { type: Date, default: Date.now }
});

const Post = mongoose.model('Post', postSchema);

module.exports = Post;
```

postController.js

```
const Post = require('../models/Post')

// Отримання всіх постів і відображення у вигляді списку
exports.getAllPosts = async (req, res) => {
  try {
    const posts = await Post.find({})
    res.render('postsList', { posts: posts })
  } catch (err) {
    res.status(500).send({ message: err.message })
  }
}

// Відображення сторінки для додавання нового посту
exports.addPostPage = (req, res) => {
  res.render('addPost')
}

// Створення нового посту
exports.createPost = async (req, res) => {
  const { title, author, text } = req.body
  try {
    await Post.create({ title, author, text })
    res.redirect('/posts')
  } catch (err) {
    res.status(500).send({ message: err.message })
  }
}

// Відображення сторінки для редагування посту
exports.editPostPage = async (req, res) => {
  try {
    const post = await Post.findById(req.params.id)
```

```

    res.render('editPost', { post })
  } catch (err) {
    res.status(404).send({ message: 'Post not found' })
  }
}

// Оновлення посту
exports.updatePost = async (req, res) => {
  const { title, author, text } = req.body
  try {
    const post = await Post.findById(req.params.id);

    if (!post) {
      return res.status(404).send({ message: "Post not found" });
    }

    post.title = title;
    post.author = author;
    post.text = text;
    post.updatedAt = new Date();

    await post.save();
    res.redirect('/posts')
  } catch (err) {
    res.status(500).send({ message: err.message })
  }
}

// Видалення посту
exports.deletePost = async (req, res) => {
  try {
    await Post.findOneAndDelete({ _id: req.params.id })
    res.redirect('/posts')
  } catch (err) {
    res.status(500).send({ message: err.message })
  }
}

// Отримання всіх постів у json форматі
exports.getPostsJson = async (req, res) => {
  try {
    const posts = await Post.find({}).lean();
    res.json(posts);
  } catch (err) {
    res.status(500).send({ message: err.message });
  }
};

```

postRouters.js

```

const express = require('express');
const router = express.Router();
const postController = require('../controllers/postController');

router.get('/', postController.getAllPosts);
router.get('/add', postController.addPostPage);
router.get('/edit/:id', postController.editPostPage);
router.post('/add', postController.createPost);
router.post('/edit/:id', postController.updatePost);
router.get('/delete/:id', postController.deletePost);

```

```
router.get('/json', postController.getPostsJson);

module.exports = router;
```

3. Створюємо відповідні структури web-сторінок, використовуючи hbs.

postsList.hbs

```
<h1>Posts</h1>
<ul>
  {{#each posts}}
    <li>
      <strong>Title:</strong> {{this.title}}<br>
      <strong>Author:</strong> {{this.author}}<br>
      <strong>Text:</strong> {{this.text}}<br>
      <strong>Created At:</strong> {{formatDate this.createdAt "MMMM Do YYYY,
h:mm:ss a"}}<br>
      <strong>Last Updated:</strong> {{formatDate this.updatedAt "MMMM Do YYYY,
h:mm:ss a"}}<br>
      <a href="/posts/edit/{{this._id}}">Edit</a> |
      <a href="/posts/delete/{{this._id}}" onclick="return confirm('Are you
sure?');">Delete</a>
    </li>
  {{/each}}
</ul>
<a href="/posts/add">Add New Post</a>
```

addPost.hbs

```
<h1>Add New Post</h1>
<form action="/posts/add" method="POST">
  <div>
    <label for="title">Title:</label>
    <input type="text" id="title" name="title" required />
  </div>
  <div>
    <label for="author">Author:</label>
    <input type="text" id="author" name="author" required />
  </div>
  <div>
    <label for="text">Text:</label>
    <textarea id="text" name="text" required></textarea>
  </div>
  <button type="submit">Add Post</button>
</form>
<a href="/posts">Back to Posts</a>
```

editPost.hbs

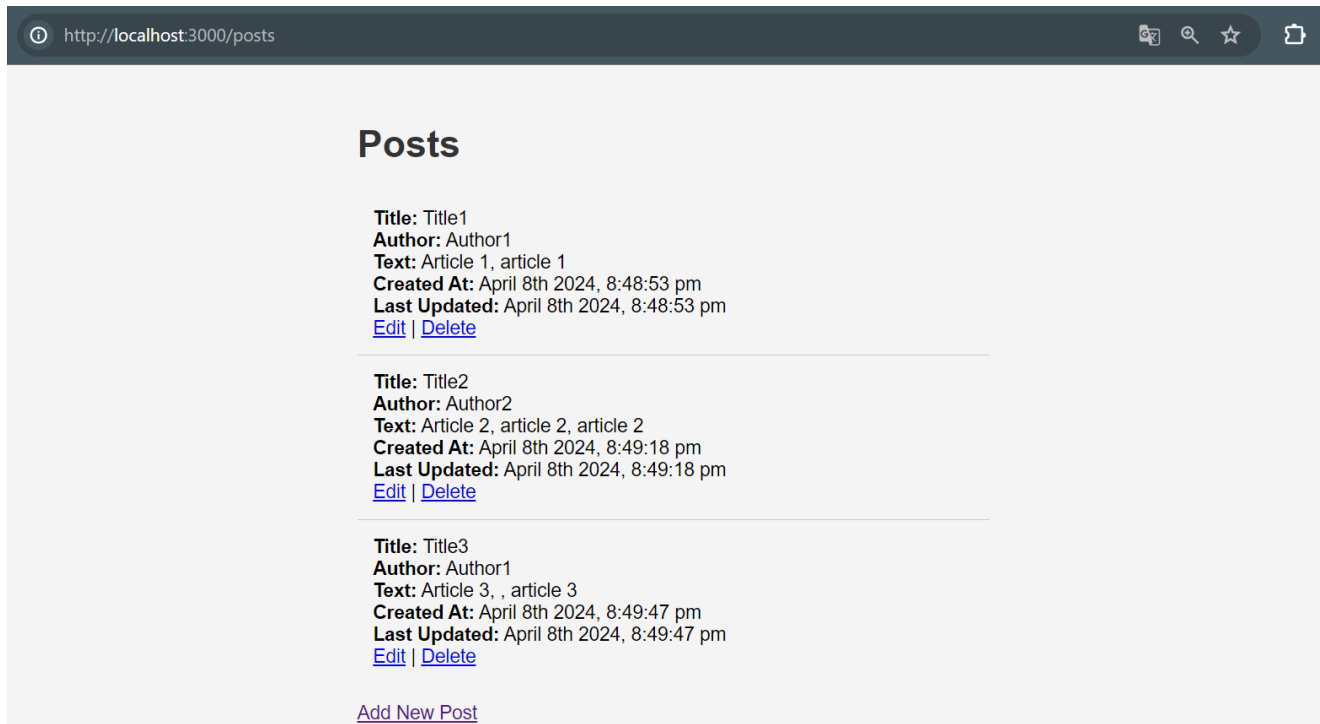
```
<h1>Edit Post</h1>
<form action="/posts/edit/{{post._id}}" method="POST">
  <div>
    <label for="title">Title:</label>
    <input type="text" id="title" name="title" value="{{post.title}}" required>
  </div>
  <div>
    <label for="author">Author:</label>
```

```

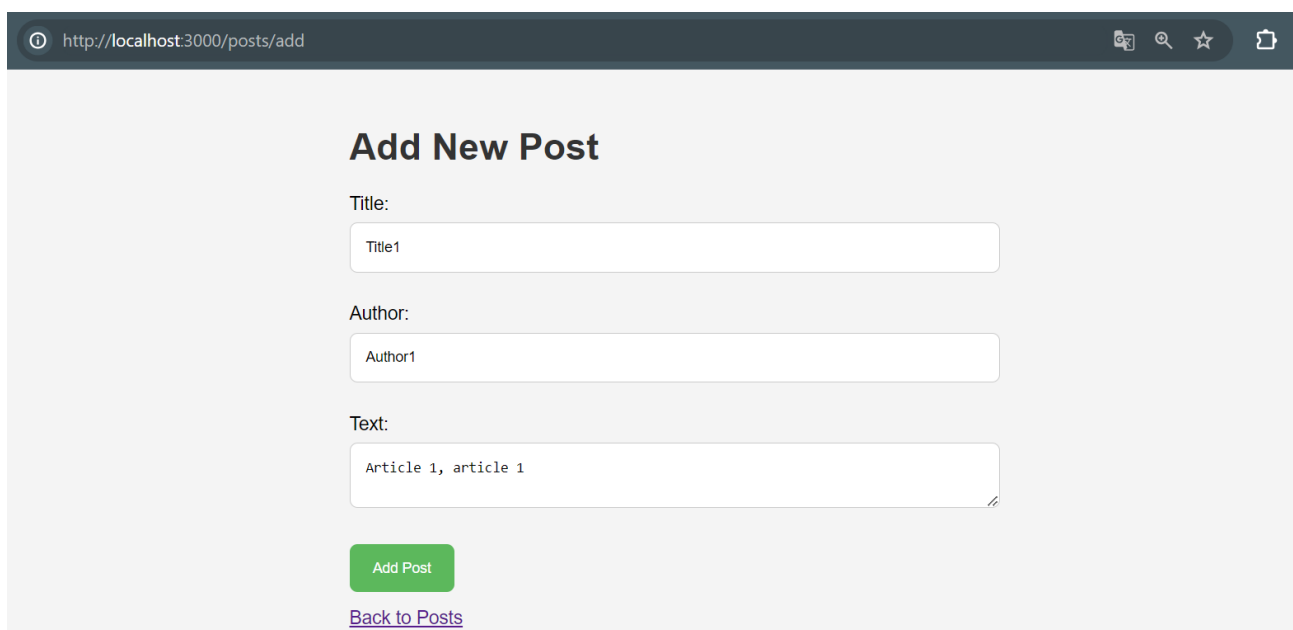
        <input type="text" id="author" name="author" value="{{post.author}}"
required>
    </div>
    <div>
        <label for="text">Text:</label>
        <textarea id="text" name="text" required>{{post.text}}</textarea>
    </div>
    <button type="submit">Update Post</button>
</form>
<a href="/posts">Back to Posts</a>

```

4. Результати



Мал.1 Головна сторінка



Мал.2 Сторінка створення поста

ⓘ http://localhost:3000/posts/edit/66142e1eb480116c7788da16

Edit Post

Title:

Author:

Text:

[Back to Posts](#)

Мал.3 Сторінка редагування поста

ⓘ http://localhost:3000/posts

Повідомлення з localhost:3000

Are you sure?

P

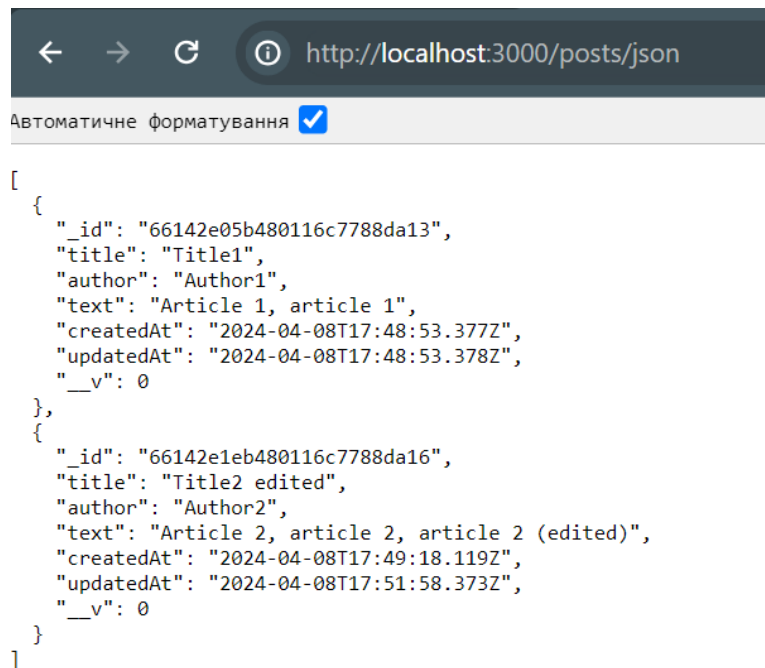
Title: Article 1, article 1
Author: Author1
Text: Article 1, article 1
Created At: April 8th 2024, 8:48:53 pm
Last Updated: April 8th 2024, 8:48:53 pm
[Edit](#) | [Delete](#)

Title: Title2 edited
Author: Author2
Text: Article 2, article 2, article 2 (edited)
Created At: April 8th 2024, 8:49:18 pm
Last Updated: April 8th 2024, 8:51:58 pm
[Edit](#) | [Delete](#)

Title: Title3
Author: Author1
Text: Article 3, , article 3
Created At: April 8th 2024, 8:49:47 pm
Last Updated: April 8th 2024, 8:49:47 pm
[Edit](#) | [Delete](#)

[Add New Post](#)

Мал.4 Видалення поста



```
[
  {
    "_id": "66142e05b480116c7788da13",
    "title": "Title1",
    "author": "Author1",
    "text": "Article 1, article 1",
    "createdAt": "2024-04-08T17:48:53.377Z",
    "updatedAt": "2024-04-08T17:48:53.378Z",
    "__v": 0
  },
  {
    "_id": "66142e1eb480116c7788da16",
    "title": "Title2 edited",
    "author": "Author2",
    "text": "Article 2, article 2, article 2 (edited)",
    "createdAt": "2024-04-08T17:49:18.119Z",
    "updatedAt": "2024-04-08T17:51:58.373Z",
    "__v": 0
  }
]
```

Мал.5 Інформація про пости в json-форматі

Висновок: при виконанні даної роботи ми ознайомились із процесом створення бази даних на віддаленому сервері MongoDB Atlas, впровадження перших моделей, контролерів та роутів. В контролері ми реалізувати всі необхідні CRUD операції. Також було розроблено web-сторінки для тестування роботи контролера через користувацький інтерфейс, де передача даних була реалізована через шаблонізацію. Завдання комп'ютерного практикуму було виконано, а всі результати, у вигляді скріншотів, представлені у звіті.