Приватний заклад вищої освіти
Одеський технологічний університет «ШАГ»
Кафедра інформаційних технологій та фундаментальної підготовки

випускна кваліфікаційна робота бакалавра

«ІНФОРМАЦІЙНА СИСТЕМА КЕРУВАННЯ ЕЛЕКТРОННИМИ ДОКУМЕНТАМИ ТА ПУБЛІКАЦІЯМИ»

на здобуття бакалаврського рівня вищої освіти
зі спеціальності «122 Комп'ютерні науки»

Виконавці проєкту:

№	П.І.Б.	Група
1	Омельницький М. М.	КН-П-192
2	Желіковський А. В.	КН-П-192
3	Мануйлов М. О.	КН-П-192
4	Горіщак К.С.	КН-Д-192
5	Амікішиєв Е. Н.	КН-А-191

Автор звіту:

_____ Желіковський Артем Володимирович

Одеса – 2023

АНОТАЦІЯ

УДК 004.9
Жел-50

Желіковский А. В. Інформаційна система керування електронними документами та публікаціями: Автореферат випускної кваліфікаційної роботи на здобуття бакалаврського рівня вищої освіти зі спеціальності «122 Комп'ютерні науки». - Одеса: ОТУШ, 2023 - 22 с.

Кваліфікаційна робота присвячена розробці та реалізації інформаційної системи керування електронними документами та публікаціями. В цей час, з поширенням електронних засобів комунікації та збільшенням обсягу електронних документів і публікацій, ефективне управління цими ресурсами стає важливим завданням.

У рамках цього дослідження було проведено аналіз поточного стану керування електронними документами та публікаціями, виявлені основні проблеми та недоліки наявних систем. На основі цього аналізу були сформульовані вимоги до інформаційної системи, яка повинна бути розроблена.

Розроблено прототип інформаційної системи, що базується на сучасних технологіях керування електронними документами та публікаціями. Ця система надає можливості зберігання, організації та керування електронними документами та публікаціями в уніфікованому середовищі.

Застосування розробленої інформаційної системи може сприяти поліпшенню ефективності роботи з електронними документами та публікаціями, зменшенню часу на пошук та обробку інформації, а також забезпечити більшу надійність та безпеку даних.

У результаті експериментального використання прототипу системи було підтверджено його функціональну придатність та ефективність. Розроблена система може бути застосована в організаціях, які працюють з великим обсягом електронних документів та публікацій, забезпечуючи зручний і безпечний доступ до цих ресурсів.

ЗМІСТ

ВСТУП

Сучасний світ знаходиться в епоху цифрової трансформації, де інформаційні технології відіграють вирішальну роль у розвитку суспільства, бізнесу та науки. За останні десятиліття значний прогрес досягнуто в галузі електронного документообігу та публікацій, що призвело до зростання обсягу електронних документів та публікацій, а також до необхідності ефективного керування цими ресурсами.

Інформаційні системи керування електронними документами та публікаціями стають невіддільною складовою сучасного управління об'єктами інформаційної діяльності. Ці системи дозволяють збирати, зберігати, обробляти та розповсюджувати електронні документи та публікації в ефективний і безпечний спосіб. Вони забезпечують централізований доступ до інформації, сприяють її швидкому пошуку та обміну, а також забезпечують збереження та контроль доступу до цінних даних.

Метою даної кваліфікаційної роботи є розробка та аналіз інформаційної системи керування електронними документами та публікаціями, зокрема в контексті використання в наукових організаціях та освітніх установах. Робота спрямована на вивчення сучасних підходів та технологій управління електронними ресурсами, а також розробку пропозицій щодо вдосконалення наявних систем та процесів.

У процесі дослідження будуть розглянуті основні принципи побудови інформаційної системи керування електронними документами та публікаціями, включаючи збір, зберігання, обробку, пошук та розповсюдження інформації. Будуть досліджені поточні тенденції у цій галузі, а також проблеми, які виникають при впровадженні та використанні інформаційних систем у різних сферах діяльності.

Очікується, що результати цієї роботи допоможуть покращити управління електронними документами та публікаціями, сприятимуть підвищенню ефективності роботи організацій та підприємств, а також сприятимуть подальшому розвитку сфери електронного документообігу та публікацій.

В даному звіті будуть представлені результати дослідження, аналіз отриманих даних, а також рекомендації щодо покращення наявних інформаційних систем керування електронними документами та публікаціями. Також будуть висунуті пропозиції щодо подальших напрямів досліджень та розвитку даної галузі.

ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЄКТУ

Загальний опис

Система управління комерційним контентом призначена для ефективного управління, організації та публікації комерційного контенту в інтернет-проєктах, таких як вебсайти, електронні магазини, блоги і т.д.

Основне призначення системи управління публікаціями полягає в тому, щоб забезпечити власникам проєктів і вебадміністраторам зручні інструменти для створення, редагування, організації, публікації та керування комерційним контентом. Вона допомагає забезпечити актуальність і якість контенту, а також знизити витрати на розробку та обслуговування вебпроєктів.

Призначення

Інформаційна система керування електронними документами та публікаціями призначена для ефективного зберігання, організації, керування та поширення електронних документів та публікацій. Основними цілями такої системи є:

- Збереження електронних документів та публікацій
- Організація та категоризація документів та публікацій
- Керування життєвим циклом документів та публікацій
- Забезпечення безпеки та контролю доступу
- Публікація та поширення інформації

Вимоги до функціоналу

1. Реєстрація та авторизація користувачів:

    - Можливість реєстрації нових користувачів з обов'язковим заповненням основних персональних даних.
    - Можливість входу в систему для зареєстрованих користувачів за допомогою логіна та пароля або іншого методу авторизації (наприклад, через соціальні мережі).

2. Пошук та навігація:

    - Розташування публікацій за категоріями та жанрами для полегшення пошуку.

- Можливість використовувати різні фільтри для уточнення результатів пошуку (наприклад, за автором, роком видання, видавництвом тощо).
- Результати пошуку повинні бути відображені швидко та ефективно.

3. Відображення публікації:

   - Інформація про книгу, включаючи назву, автора, видавництво, опис, обкладинку, рейтинг, відгуки покупців, ціну тощо.
   - Наявність функції "Додати до кошика" або "Придбати" на сторінці товару.

4. Кошик та оформлення замовлення:

   - Можливість додавання товарів до кошика, видалення пубікацій або зміна їх кількості.
   - Відображення загальної суми замовлення та вартості доставки.
   - Обробка замовлення, включаючи введення адреси доставки, способу оплати та перевірку замовлення перед підтвердженням.

5. Оплата та доставка:

   - Різні способи оплати, такі як кредитні картки, електронні гроші, оплата при отриманні тощо.
   - Різні варіанти доставки, включаючи доставку поштою, кур'єром, самовивіз з магазину тощо.
   - Відстеження стану доставки та повідомлення про її успішне завершення.

6. Огляди та рейтинги:

   - Можливість користувачам залишати відгуки та оцінювати книги.
   - Відображення середньої оцінки книги на її сторінці.

7. Адміністративний функціонал:

   - Можливість додавання та видалення книг, оновлення інформації про них адміністратором або продавцем.
   - Керування замовленнями та користувачами.

8. Мобільна сумісність:

- Адаптивний дизайн, що дозволяє зручно переглядати та здійснювати покупки на сайті з мобільних пристроїв.

Архітектура

ПЗ має бути спроєктоване за клієнт-серверною архітектурою з розподіленими вузлами:

- Backend - застосунок, що дозволяє керувати даними, зберігати їх у базі даних та віддавати дані за допомогою програмних інтерфейсів (API), якими користуються інші вузли.
- Web Application - застосунок, з яким буде працювати користувач через зрозумілий інтерфейс. Використовує Backend частину для передачі та отримання даних.

Усі модулі ПЗ повинні мати засоби тестування функціональності та розгортання на новому пристрої (інсталяції).

Технології

Зберігання вихідних кодів ПЗ повинно здійснюватись за допомогою системи контролю версій (вибір конкретної системи узгоджується групою проєкту). Для кожного з модулів рекомендується (з можливістю часткової зміни) наступні технології створення:

- Backend - JavaScript (Node.js), MongoDB
- Web Application - HTML, CSS, Javascript у складі одного з фреймворків (React.js, Angular.js, Vue.js)

Вихідний код повинен оформлятися відповідно до загальноприйнятих стандартів, на зразок стандартів GNU http://www.gnu.org/prep/standards/, у тому числі з додержанням вимог до коментування та документування коду.

РОЗДІЛ 1. Загальні характеристики системи

Проєкт "Книжковий магазин Historium" включає комплексну розробку та реалізацію інформаційної системи публікацій, найпоширенішими з яких є книги. У рамках цього проєкту використовуються передові технології та методології для забезпечення ефективного та зручного функціонування системи.

Початково, були створені макети та інтерфейс користувача, використовуючи Figma. При розробці інтерфейсу, було звернуто увагу на розумну навігацію та ергономіку, забезпечуючи комфортне користування системою для різних категорій користувачів.

Клієнтська частина системи була реалізована з використанням фреймворку Vue.js. Це дозволило створити модульну та масштабовану архітектуру, забезпечуючи швидкодію та зручність розробки. Компоненти для головної сторінки, каталогу книг, сторінок окремих книг та кошика покупок забезпечують зручну навігацію та взаємодію з системою.

Серверна частина системи була реалізована з використанням Node.js та Express. Цей підхід дозволив створити потужний та масштабований сервер, який забезпечує обробку запитів та взаємодію з базою даних. API було розроблено для отримання списку книг, деталей книг, додавання товарів до кошика та оформлення замовлення.

Для забезпечення надійності та якості системи, була розроблена тестова частина з використанням Mocha та Supertest. Це дозволило автоматизувати процес тестування та перевірки функціональності системи, забезпечуючи високу стабільність та надійність.

Крім того, для зберігання та керування даними системи була використана база даних MongoDB та фреймворк Mongoose. Це забезпечило зручну роботу з даними, створення схеми та виконання запитів до бази даних.

Одним з важливих аспектів розробки проєкту є використання практик DevOps, що дозволяють забезпечити ефективне співробітництво між розробниками та операторами системи.

Для контролю версій використовується система Git, що дозволяє зберігати, відстежувати та керувати кодом проєкту. Завдяки цьому, команда розробників може працювати одночасно над різними функціональностями та вносити зміни безпеки.

Інфраструктура проєкту розгорнута у хмарних сервісах, що надають гнучкість та масштабованість. Використання хмарних рішень дозволяє ефективно керувати ресурсами та забезпечує високу доступність системи.

Для контейнеризації використовується технологія Docker, що дозволяє пакувати додаток та його залежності у віртуальний контейнер. Це спрощує розгортання та масштабування системи, забезпечуючи консистентність середовищ та підвищуючи безпеку та надійність.

Автоматизація процесів розробки та розгортання забезпечується за допомогою інструментів Continuous Integration/Continuous Deployment (CI/CD). Це дозволяє автоматично збирати, тестувати та розгортати нові версії додатка, забезпечуючи швидку зміну та впровадження нового функціоналу.

Усі частини системи були інтегровані в єдину цілісну платформу, що дозволяє зручно та ефективно продавати книжки онлайн. Технологічні рішення та використання передових інструментів забезпечують високу продуктивність та надійність системи проєкту "Книжковий магазин Historium".

РОЗДІЛ 2. Реалізація бази даних та робота з нею

У процесі розробки інформаційної системи вибором для бази даних була MongoDb типу NoSQL. Налаштування та взаємодія здійснювались за допомогою мови JavaScript у середовищі Node.js. Для забезпечення взаємодії з базою даних була використана бібліотека Mongoose. Використовувався інструмент Lucid для розроблення діаграм бази даних. Крім того, використовувалися наступні засоби:

- **Visual Studio Code** - інтегроване середовище розробки, яке надає засоби для написання та запуску програмного коду.
- **MongoDb Atlas** - хмарна платформа для хостингу та управління базами даних MongoDb. Вона надає зручні інструменти для налаштування, масштабування та керування базами даних без необхідності встановлення та налаштування власного сервера.
- **MongoDb Compass** - графічний інтерфейс користувача для взаємодії з базами даних MongoDb.
- **MongoDb Tools** - набір утиліт і командного рядка для роботи з базами даних MongoDb.
- **Jira** - інтегрована платформа для управління проєктами та задачами, що надає засоби для планування, відстеження та співпраці над розробкою програмного забезпечення.

Процес розробки бази даних можна розділити на наступні основні етапи:

- Аналіз вимог та виділення потрібних сутностей в базі даних.
- Створення схем для сутностей за допомогою бібліотеки Mongoose.
- Написання тригерів для виконання автоматичних дій при певних подіях.
- Заміна фактичного видалення на "м'яке" видалення.
- Розроблення діаграми створеної бази даних.

Аналіз вимог та виділення потрібних сутностей в базі даних.

Для виділення потрібних сутностей були проаналізовані різноманітні сайти, спеціалізовані на продажах книжок.

- Інтернет-магазин книг — Книгарня “Є”. URL: https://book-ye.com.ua
- Книжковий інтернет-магазин — Наш Формат.
  URL: https://nashformat.ua/
- Інтернет-магазин книг, подарунків та дитячих товарів — YAKABOO.
  URL: https://www.yakaboo.ua/

Під час цього аналізу було виділено основні сутності, які відображають ключові аспекти діяльності книжкових магазинів. Серед виділених сутностей був прийнятий набір з 19 сутностей, який відповідає потребам і вимогам. Під час цього вибору враховувалися різні фактори, такі як функціональність, ефективність, логічна зв'язаність та масштабованість бази даних. Цей набір включає основні елементи, такі як книги, автори, видавці, замовлення, користувачі та інші, що відображають основні аспекти функціонування книжкового магазину.

Створення схем для сутностей за допомогою бібліотеки Mongoose.

Процес створення схем передбачав опис кожної сутності у вигляді коду, використовуючи синтаксис Mongoose. Схеми складалися з набору полів, які визначали структуру даних для кожної сутності. Кожне поле мало свою назву, тип даних та інші параметри, які визначали поведінку цього поля у базі даних. У загальному було створено 19 схем для різних сутностей системи. Крім визначення структури даних, в схемах також була встановлена валідація для забезпечення коректності даних, що зберігаються у базі даних.
Як приклад схем, що були розроблені, представляються схеми користувача (Схема 1) та продукту (Схема 2).

```
    {
    firstName: {type: String, required: true, minlength: 2, maxlength: 50},
    lastName: {type: String, required: true, minlength: 2, maxlength: 50},
    phoneNumber: {type: String, required: true, unique: true, index: true},
    email: {type: String, required: true, unique: true, index: true},
    password: {type: String, required: true},
    salt: {type: String, required: true},
    role: {type: String, required: true, default: 'user'},
    reviews: [{type: ObjectId, ref: 'Review', required: false}],
    temporaryPassword: {type: String, required: false},
    wishlist: [{type: ObjectId, ref: 'Product', required: false}],
    history: [{type: ObjectId, ref: 'Product', required: false}],
    products: {
          type: [{type: ObjectId, ref: 'Product', required: false}],
          required: false,
          default: []
      },
    cart: {type: ObjectId, ref: 'Cart', required: false},
    birthDate: {type: Number, required: false},
    createdAt: {type: Number},
    updatedAt: {type: Number}
},
{ versionKey: false, timestamps: true, strict: false}
```

Схема 1.

```
 {
  name: {type: String, required: true},
  specificProduct: {type: ObjectId, required: false},
  code: {type: String, required: false, unique: true},
  key: {type: String, required: true, unique: true},
  price: {type: Number, required: true,  min: 0},
  quantity: {type: Number,  required: false, min: 0},
  type: {type: ObjectId, ref: 'ProductType', required: true},
  model: {type: String, required: true},
  sections: [{type: ObjectId, ref: 'Section', required: true}],
  description: {type: String, required: true},
  reviews: [{type: ObjectId, ref: 'Review', required: false}],
  images: [{type: String, required: true}],
  requiresDelivery: {type: Boolean, default: true},
  createdAt: {type: Number},
  updatedAt: {type: Number},
  deletedAt: {type: Number, required: false}
 },
 {versionKey: false, timestamps: true}
```

Схема 2.

Написання тригерів для виконання автоматичних дій при певних подіях.

Під час розробки були написані тригери, що являти собою хуки "pre" та "post" для певних подій в схемі Mongoose. Ці тригери автоматично виконуються перед або після виникнення певних подій. Як приклад, розглянемо тригер оновлення останнього використаного коду товару.

В схемі продукту зазначається, що перед виконанням методу save у моделей, створених за цією схемою, виконуються встановлені методи.

```
productSchema.pre('save', async function (next) {
    await setProductCode(this);
    await increaseProductsQuantity();
    next();
});
```

Код методу setProductCode:

```
const setProductCode = async doc => {
    try {
        await mongoose.connection
            .collection('product_code_counter')
            .updateOne({}, { $inc: { currentCode: 1 } });

        const productCodeCounter = await mongoose.connection
            .collection('product_code_counter')
            .findOne();

        await doc.set('code', productCodeCounter.currentCode);
    } catch (err) {
        throw {
            status: err.status ?? 500,
            message: err.message ?? err
        };
    }
};
```

Цей метод звертається до єдиного документа колекції product_code_counter, який має такий вид:

```
_id: ObjectId('649aeb17db84143f62789661')
currentCode: 116243
```

Заміна фактичного видалення на "м'яке" видалення.

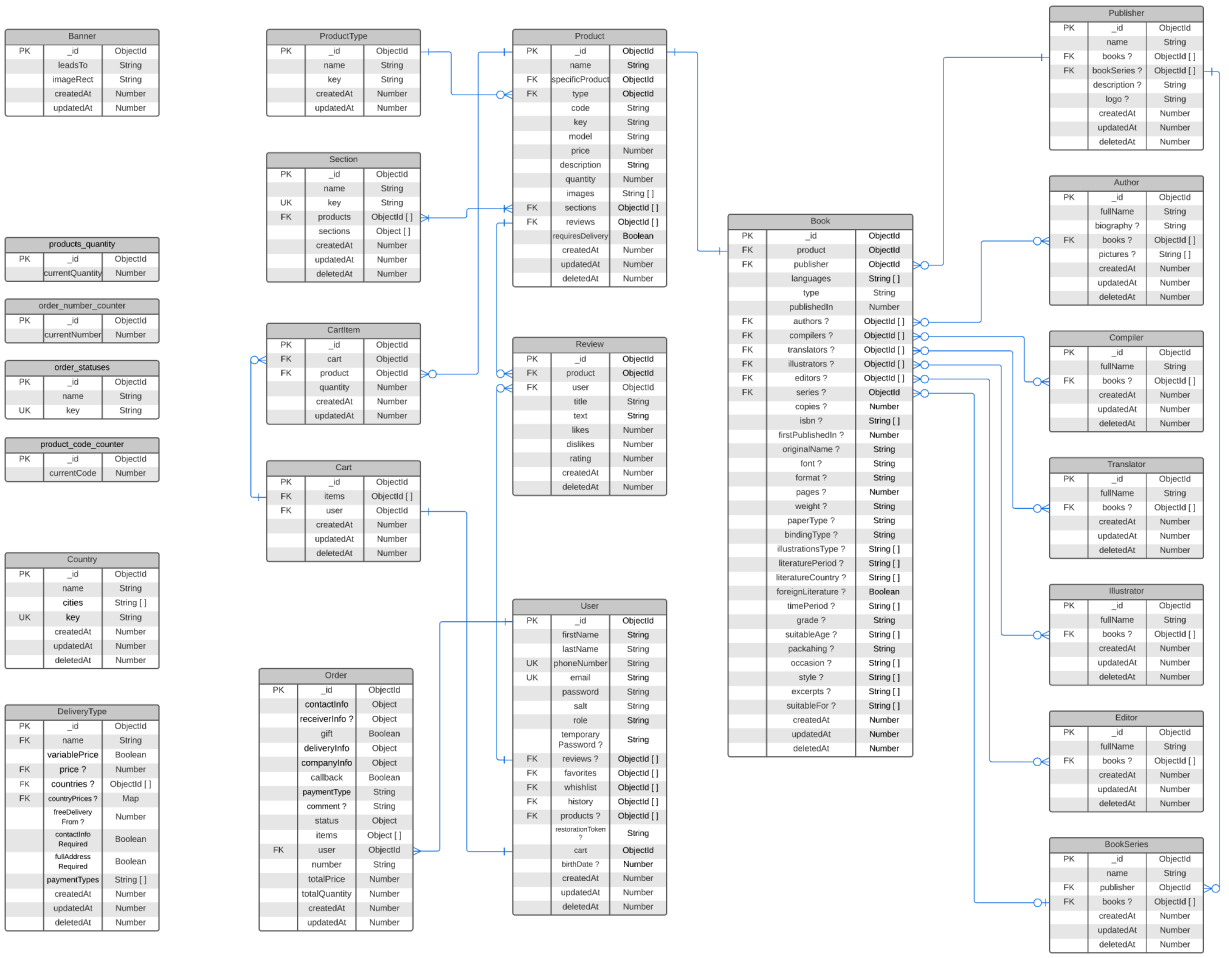
У процесі розробки була реалізована заміна фактичного видалення записів в базі даних на "м'яке" видалення. М'яке видалення дозволяє позначити запис як видалений, зберігаючи його в базі даних, але відмічаючи його як неактивний або архівний. Це забезпечуючи більш гнучке та контрольоване управління даними в базі даних. Для реалізації м'якого видалення було перевизначено метод deleteOne для всіх схем, що вимагають такого підходу. Нижче наведений фрагмент коду, який демонструє цю реалізацію:

```
deleteDocument = async doc => {
    try {
        if (!doc.hasOwnProperty('deletedAt')) {
            doc.set('deletedAt', Date.now());

            await doc.save();
        } else
            throw new Error(
                'Error while uninstalling product. Product already removed.'
            );
    } catch (err) {
        throw {
            status: err.status ?? 500,
            message: err.message ?? err
        };
    }
};
```

Розроблення діаграми створеної бази даних

У процесі розробки проєкту створювалася та вдосконалювалася діаграма бази даних.Окрім колекцій, що відображають сутності самого додатку, також включені допоміжні колекції, які використовуються для реалізації певних функціональностей та зберігання додаткової інформації.. Нижче наведена остаточна версія діаграми, яка відображає структуру створеної бази даних та взаємозв'язки між сутностями.



Ця діаграма відображає структуру створеної бази даних та взаємозв'язки між сутностями, що являє собою важливий аспект в розробці бази даних.

РОЗДІЛ 3. Результати випробувань

Для належного функціонування застосунку, база даних може бути розміщена у двох варіантах. Перший варіант — база даних розгорнута в хмарному сховищі, відокремлено від сервера додатка. Наприклад, може бути використано сервіс MongoDb Atlas. Другий варіант — сервер MongoDb повинен бути встановлений та запущений на тому ж сервері, де розгорнуто додаток. Для забезпечення взаємодії між сервером та базою даних, потрібно вказати рядок підключення до бази даних у змінних оточення на серверній стороні.

У результаті тестування системи було виявлено, що локальне розгортання сервера MongoDb забезпечує кращу продуктивність та швидкість обробки запитів, що дозволяє забезпечити ефективну роботу додатка. Далі наведено приклад різниці часу відповіді при зверненні до бази даних, яка розгорнута в хмарному сховищі (Рисунок 1) та бази даних, яка розгорнута на тому ж сервері, що і додаток (Рисунок 2).

Рисунок 1

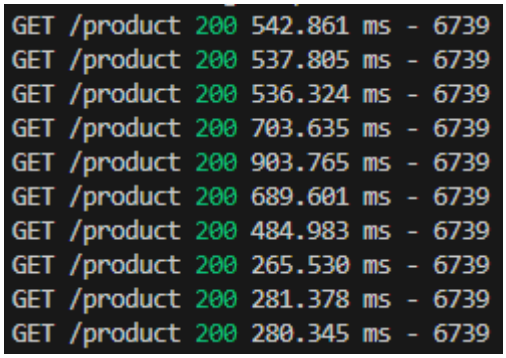
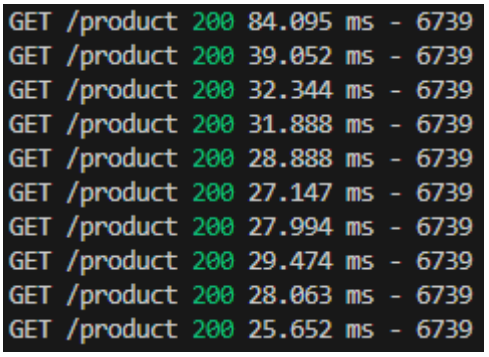


Рисунок 2



При запуску сервера додатка, опціонально виконується заповнення бази даних у разі відсутності даних — "seeding". Для цього виконується перевірка наявності даних у базі та заповнення її за допомогою спеціального методу ImportData.

Виведення у консоль результату спрацьовування методу ImportData

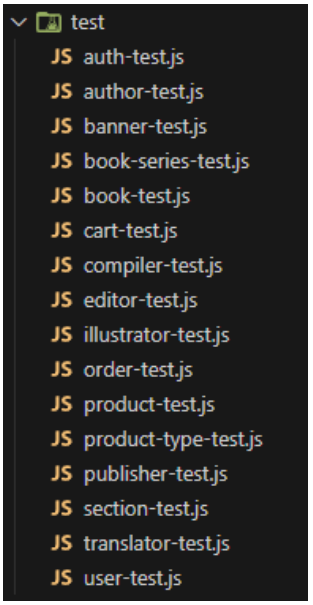
```
Base not found. Initialization in progress.
The 'authors' collection is being initialized
'authors' collection initialization complete
The 'books' collection is being initialized
'books' collection initialization complete
The 'bookseries' collection is being initialized
'bookseries' collection initialization complete
The 'carts' collection is being initialized
'carts' collection initialization complete
The 'compilers' collection is being initialized
'compilers' collection initialization complete
The 'countries' collection is being initialized
'countries' collection initialization complete
The 'deliverytypes' collection is being initialized
'deliverytypes' collection initialization complete
The 'editors' collection is being initialized
'editors' collection initialization complete
The 'illustrators' collection is being initialized
'illustrators' collection initialization complete
The 'order_number_counter' collection is being initialized
'order_number_counter' collection initialization complete
The 'order_statuses' collection is being initialized
'order_statuses' collection initialization complete
The 'products' collection is being initialized
'products' collection initialization complete
The 'products_quantity' collection is being initialized
'products_quantity' collection initialization complete
The 'producttypes' collection is being initialized
'producttypes' collection initialization complete
The 'product_code_counter' collection is being initialized
'product_code_counter' collection initialization complete
```

Дані для ініціалізації бази даних зберігаються у серверній частині додатка та можуть бути оновлені поточними даними бази даних за допомогою команди:

```
mongodump --uri
"mongodb+srv://Admin:vqgrOubr2rcQngBa@cluster0.e036dv9.mongodb.net/?retryWr
ites=true&w=majority" --out "BACKUP_DIR_PATH"
```

РОЗДІЛ 4. Процес та результат тестування серверної частини

Для перевірки функціональності серверної частини додатка було вирішено провести тестування API. Для написання тестів були використані мова JavaScript, фреймворк Node.js і такі бібліотеки, як Mocha, Chai, Supertest та Mongoose. Всі тести розташовані у директорії проєкту і розділені на окремі файли залежно від тестованої функціональності.



Загалом було написано 80 тестів, поділених на 16 файлів, з позитивними сценаріями, які охоплюють тести GET, POST, PATCH та DELETE запитів до API, а також складні тести, наприклад, оформлення замовлення. Як приклад, продемонстровано тестування GET-запиту '/user/'.

```
describe(' "/user/" GET request ', () => {
        it('The token is correct; the role is correct; an array of users should be
returned', async () => {
            const userData = {
                login: 'dobriy.edu@gmail.com',
                password: '41424344'
            };

            userToken += (await request(app).post('/login').send(userData))
                .body.token;

            await request(app)
                .get('/user/')
                .set('Authorization', userToken)
                .then(response => {
                    expect(response.status).to.equal(200);
                    expect(response.header['content-type']).to.include(
                        'application/json'
                    );
                    expect(response.body).to.be.an('array');
                });
        });
    });
```

Для тестування була використана окрема тестова база даних MongoDB. Це зроблено для забезпечення незалежності тестування та роботи сервера. Підключення до тестової бази даних відбувається перед кожнім комплексом тестів:

```
before(async () => {
        await mongoose
            .connect(process.env.TEST_CONNECTION_STRING)
            .catch(err => {
                console.log(`Failed to connect to database: ${err.message}`);
            });
    });
```

Кожен тест має наступну послідовність:

1. Додавання користувача з правами адміністратора в тестову базу даних (якщо це необхідно) та отримання його токена.
2. Додавання додаткових даних в базу даних для встановлення правильних залежностей (якщо це необхідно).
3. Виконання запиту до API з передачею токена користувача та всіх необхідних параметрів.
4. Отримання відповіді та порівняння її з очікуваним результатом.
5. Видалення доданих даних з бази даних.

Такий підхід до тестування дозволяє перевірити різні сценарії взаємодії з API, а також гарантує коректність його роботи у різних умовах.

ВИСНОВКИ

Проведено огляд інформаційних систем керування електронними документами та публікаціями. Виявлено, що суттєве поширення мають системи, пов’язані з комерційною діяльністю. В якості такої системи було обрано та реалізовано інтернет-магазин по продажу книг.

На етапі проєктування бази даних було досліджено декілька наявних інтернет-магазинів на предмет встановлених сутностей, їх актуальність, можливі альтернативи та покращення. Основою дослідження став інтернет-магазин Yakaboo (https://www.yakaboo.ua). В результаті проєктування було виділено 19 сутностей для описування їх у базі даних.

В рамках розробки проєкту було вирішено використовувати базу даних MongoDB для зберігання та управління даними. Вибір цієї бази даних було зумовлено її документо-орієнтованим форматом даних, масштабованістю та відповідності до поставлених вимог.

Для відображення структури створеної бази даних та візуалізації взаємодії між її сутностями було створено діаграму, яка допомагає зрозуміти будову бази даних.

В результаті розробки було створено 24 колекції у базі даних, серед яких 4 відносяться до службових, та 19 схем сутностей, кожна з яких має обмежування валідації, серед яких 16 схем реалізують "м'яке" видалення.

Було створено комплекс тестів для серверної частини, який перевіряє весь її основний функціонал незалежно від її роботи, що гарантує цілісність даних. Тестування забезпечує надійність та працездатність додатка.

Підсумовуючи можна сказати, що впроваджені засоби відповідають технічному завданню і можуть бути доповнені та покращенні у майбутньому. Система створювалась маючи на увазі певний рівень абстракції, щоб залишити можливість її подальшого розширення.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Стратегія розвитку інформаційного суспільства в Україні. Розпорядження Кабінету Міністрів України від 15 травня 2013 р. № 386-р. // Урядовий портал. URL: https://www.kmu.gov.ua/npas/246420577
2. Стратегія кібербезпеки України. Указ Президента України від 14 травня 2021 року // Урядовий кур'єр від 28.08.2021 — № 165. URL: https://zakon.rada.gov.ua/laws/show/447/2021/print
3. Решетило В.П., Федотова Ю.В. Невизначеність та ризик: співвідношення понять та специфіка прийняття рішень // Проблеми системного підходу в економіці, випуск № 3(77)-2, 2020. с. 149. URL: http://psae-jrnl.nau.in.ua/journal/3_77_2_2020_ukr/23.pdf
4. Олександр Смичніков. Параноя або сучасні реалії: як зберегти конфіденційність у мережі. - ФЬЮЧЕР МЕДІА. URL: https://mind.ua/openmind/20183329-paranoya-abo-suchasni-realiyi-yak-zberegti-konfidencijnist-u-merezhi
5. Анісімов А.В. Інформаційні системи та бази даних: Навчальний посібник для студентів факультету комп’ютерних наук та кібернетики. / Анісімов А.В., Кулябко П.П. − Київ. – 2017. – 110 с.
6. Величко О.М. Інтелектуальні інформаційні системи: структура і застосування. Підручник / Величко О.М., Гордієнко Т.Б. - Херсон. - 2022. - 728 с. ISBN: 978-966-289-552-0
7. Булгакова О.С. Інформатика: візуальне програмування. Навчальний посібник (стереотипне видання) / Булгакова О.С., Зосімов В.В., Броницька Н.А., Танкова Н.В. - Херсон. - 2020. - 312 с. ISBN: 978-966-289-039-6
8. Покотилова В.І. Використання інформаційних технологій в теорії прийняття рішень. Навчальний посібник / Покотилова В.І., Фомішина В.М., Лугінін О.Є. - Херсон. - 2019. - 240 с. ISBN: 978-966-289-277-2
9. Величко О.М. Основи системного аналізу і прийняття оптимальних рішень. Підручник / Величко О.М., Гордієнко Т.Б. - Херсон. - 2021. - 672 с.ISBN: 978-966-289-553-7
10. Чайковська М.П. Концептуально-методологічні засади управління маркетинговими ІТ-проєктами в умовах цифрових трансформацій. Монографія / Чайковська М.П. - Херсон. - 2021. - 370 с.ISBN: 978-966-289-537-7