**Після дослідження я маю два кандидати для реалізації адмін-панелі у екосистемі Express.js, які мені найбільше імпонують:**

# Express-Admin:

Можливості:

* Легка і проста у використанні адміністративна панель, спеціально розроблена для Express.js.
* CRUD-операції для моделей даних з настроюваними поданнями та формами.
* Контроль доступу на основі ролей з аутентифікацією JWT.
* Підтримує різні джерела даних, такі як PostgreSQL, SQLite та MySQL.
* Має відкритий вихідний код і наче більш-менш активно підтримується.

Переваги:

* Проста, але насичена документація.
* Тісно інтегрований з Express.js для безшовної інтеграції.
* Простий і легкий з акцентом на CRUD-функціональність.
* Легке встановлення та мінімальна конфігурація.
* Безкоштовний.

Недоліки:

* Менш настроюється порівняно з іншим варіантом.
* Обмежений набір функцій у порівнянні з більш комплексними рішеннями.

# AdminJS:

Можливості:

* Автоматичний інтерфейс адміністратора з функцією CRUD для великої кількості ORM та ODM (TypeORM, Prisma, Mongoose, Sequelize тощо).
* Багато можливостей для кастомізації (віджети, ролі та дозволи для контролю доступу, валідації форм тощо)
* Відкритий вихідний код з безкоштовною версією для спільноти.
* Підтримка TypeScript.

Переваги:

* Докладна документація та база прикладів використання.
* Простота у налаштуванні та використанні.
* Різноманіття налаштувань.
* Потужні можливості та багата екосистема.
* Безкоштовний.

Мінуси:

* Може бути занадто «overkill» рішенням для простих проектів.
* Вимагає додаткової конфігурації для розширених функцій.

**Щодо рішення для роботи із медіа-файлами я знайшов ось ці:**

# Multer:

* Легка інтеграція в Express.js.
* Підтримує зберігання на диску, в пам'яті або хмарних сервісах.
* Масштабованість залежить від обраного варіанту зберігання.
* Вартість залежить від обраного рішення для зберігання даних (наприклад, локальне сховище або стороннє хмарне сховище).
* Насамперед орієнтований на обробку завантаження файлів.
* Добре задокументований з хорошою спільнотою.
* Я мав досвід роботи з ним

# Express-fileupload:

* Простий і легкий модуль для обробки завантаження файлів на Express.js.
* Пропонує базові функції, такі як ліміт завантаження, дозволені розширення та збереження файлів.
* Легко інтегрується з Express.js.
* Простий у використанні та розумінні.
* Добре підходить для простих проектів з базовими потребами в роботі з файлами.
* Менше налаштовується порівняно з іншими рішеннями.
* Обмежені можливості порівняно з більш повними бібліотеками.

**Для збереження медіа є наступні рішення:**

# Cloudinary:

* Відносно проста інтеграція за допомогою Cloudinary SDK для Node.js.
* Хмарне сховище з функціями динамічної маніпуляції.
* Висока масштабованість, оскільки це хмарне рішення.
* Платний. Тариф на основі використання, включаючи сховище, пропускну здатність і операції з перетворення зображень.
* Пропонує потужні функції, такі як перетворення зображень і відео, автоматична оптимізація форматів та адаптивна доставка.
* Добре задокументований і має активну підтримку спільноти.

# AWS S3:

* Потребує створення облікового запису AWS, але інтеграція спрощується за допомогою AWS SDK для JavaScript.
* Масштабоване хмарне сховище з моделлю pay-as-you-go.
* Високомасштабоване і призначене для обробки великих обсягів даних.
* Плата за зберігання та передачу даних на основі тарифів, що залежать від обсягу даних.
* Надає надійне та масштабоване рішення для зберігання даних, але не має вбудованих функцій маніпулювання медіа.
* Обширна документація та велика підтримка спільноти.

# Firebase Storage:

* Доступний та зрозумілий Firebase SDK для JavaScript, а також легка інтеграція для використання у backend.
* Хмарне сховище з легкою інтеграцією з іншими сервісами Firebase.
* Масштабованість та тісна інтеграція з іншими сервісами Firebase.
* Firebase пропонує безкоштовний рівень з тарифікацією на основі використання для більших обсягів.
* Інтегрований з іншими службами Firebase, пропонуючи оновлення в режимі реального часу, контроль доступу та легку інтеграцію з Firebase Authentication.
* Вичерпна документація та велика спільнота.

# GridFS:

* Система зберігання файлів, вбудована в MongoDB для ефективного зберігання великих файлів.
* Зберігає файли у вигляді фрагментів і метаданих, що дозволяє ефективно шукати та керувати ними.
* Простий у налаштуванні та використанні.
* Масштабований та ефективний для зберігання великих файлів.
* Тісно інтегрований з MongoDB.
* Обмежується MongoDB як бекендом для зберігання даних.
* Може не підходити для дуже великих наборів даних.
* Погано інтегрується з NestJS

**Висновок такий:**

Для використання готової адмін-панелі я б використовував AdminJS, тому що Express-Admin, на мою думку, зовсім для малих проєктів. Також AdminJS використовує React та дозволяє кастомізувати себе, що є великим плюсом. Для обробки запитів з файлами я б використовував multer, а для обробки медіа-файлів - gm, sharp виявляється має якісь вразливості, в продакшн проєктах я б його не використовував. Для зберігання медіа я б використовував скоріше Firebase:Storage, на другому місці AWS S3. Все через їх підтримку та доступність. Інші сервіси по факту базуються на AWS, то чому б не використовувать оригінал?

На дослідження витратив близько 2 годин.