1) Рішення для реалізації адмін-панелі.

Для реалізацій адмін-панелей для Express.js додатків існує багато різних рішень, однак чітко визначеного лідера серед них немає. У різноманітних статтях та рейтингах на цю тему згадується великий перелік шаблонів панелей, від яких очі розбігаються. Багато з даних рішень є платними і всі вони мають приблизно однаковий перелік можливостей та функцій.

Я зосередилася на кількох рішеннях, які були згадані не один раз в різних статтях та відгуках, а саме: ***ForestAdmin***, ***react-admin*** та ***AdminJS(AdminBro)***. Всі вони:

- пропонують створення адмін-панелі, яка може бути легко налаштована та змінена, залежно від побажань користувача;
- пропонують зручний UI з можливостями фільтрації та пошуку;
- пропонують створення дашбордів;
- мають детальну докуменацію;
- пропонують CRUD операції;
- пропонують створення та налаштування власних екшенів та хуків.

Розглянемо кожну з них детальніше:

ForestAdmin не є open-source проєктом. Починати роботу з ним дуже просто: реєструєшся та створюєш власний додаток покроково, кожен крок чітко прописаний. ForestAdmin створює окремі бекенд та фронтенд для адмін-панелі, які зберігаються серверах ForestAdmin. Має власний UI editor. UI збирає дані, про цільовий додаток з його API. Базова версія є безкоштовною, однак платна опція пропонує багато додаткових налаштувань, наприклад, role-based permissions (які, до речі, є безкоштовними в AdminJS).

React-Admin є open-source проєктом, який позиціонується, як бекенд-незалежний. Може діставати дані з будь-якого API: REST, GraphQL чи інше. Зберігається локально. Зручний для зміни UI, однак потребує більше роботи з компонентами, ніж, наприклад, ForestAdmin. Плюсом використання даного варіанту є більша гнучкість, бо розробник напряму працює з кодом, однак з цього випливає мінус: довший час налагодження.

AdminJS є open-source проєктом, який можна легко під’єднати в якості плагіна до express проєкта. На основі моделей бази даних він генерує UI. Підтримує велику кількість ORM(наприклад, sequelize, mongoose, prisma тощо). Пропонує також функціонал інтернаціоналізації та додаткових екстеншенів (наприклад, хешування паролів або завантаження медіа-даних в хмару чи локально). Загалом, це один з найкращих варіантів, плюсами якого є постійна підтримка та оновлення проєкту розробниками, додавання та розробка нових розширень та велике комьюніті. Єдиним мінусом його, на мою думку є, не надто зручний оригінальний UI (в порівнянні з попередніми варіантами).

Отже, з усіх розглянутих варіантів я би обрала для використання у своєму додатку AdminJS, адже по-перше він є open-source, через що має велике комьюніті. Друге, зберігається локально, що дозволяє гнучкіше вправлятися з проєктом. Третє, має багато доступних розширень, які не підтримують інші рішення.

2) Рішення для роботи з медіа-файлами.

Бібліотеку для роботи з медіа потрібно обирати в залежності від того, що саме необхідно зробити з цими файлами. Під певний формат теж необхідно підбирати бібліотеку, яка з ним працює.

Наприклад, для завантаження медіа файлів на сервер можуть використовуватися різні спеціалізовані бібліотеки, такі як ***Formidable***, ***Busboy***, ***Multer***.

Formidable підтримується вже 23 роки та має величезне комьюніті. З мінусів: зберігає файли у тимчасовому каталозі на жорсткому диску.

Busboy – використовує потоки. Підтримується доволі довго.

Multer – Express.js middleware, який всередині використовує Busboy. Зберігає дані в пам’яті або на диску.

Отже, кожен з цих пакетів можна обрати в залежності від проєкта, який створюється і вимог до нього. У випадку розробки Express.js проєкта, я би обрала Multer, адже він є спеціалізованим під Express та всередині використовує швидкий Busboy.

Якщо задача редагувати медіа, тоді необхідно обирати абсолютно інші бібліотеки. Кожен з типів таких файлів може мати багато форматів, тож, скоріш за все, під роботу з кожним окремим типом файлу необхідно підбирати власний пакет.

Для роботи з картинками я би використовувала nodejs бібліотеку ***sharp***. Вона швидка та інтуїтивно зрозуміла, підтримує багато форматів, таких як: JPEG, PNG, WebP, GIF and AVIF. Для інших форматів, необхідно шукати інші рішення.

Для роботи з аудіо та відео добре підходить ***fluent-ffmpeg*** – nodejs модуль, побудований на основі інструмента командного рядка ffmpeg. Використовується купою відомих проєктів і у випадку роботи з аудіо та відео, я би зупинилася саме на ньому.

Текстові файли можна редагувати за допомогою модуля nodejs – fs. Однак, якщо маємо справу з іншим форматом, наприклад, pdf, то необхідно шукати інші бібліотеки.