Varianti – “TaskManager Plus”

> Queste attività sono pensate per chi desidera rendere il proprio progetto **più completo, professionale e realistico**.
> Puoi sceglierne **una o più**, in base al tempo e al tuo livello.

1. Dashboard Front-End (HTML o React)

Obiettivo: creare un’interfaccia utente che si connette alla tua API online.

Cosa fare

Realizza una piccola dashboard che permetta di:

- registrarsi e fare login,
- visualizzare i task personali,
- crearne di nuovi o eliminarli,
- caricare un’immagine o allegato.

Come farlo

- Se vuoi un approccio semplice: usa **HTML** + **CSS** + **JavaScript** con `fetch()` per chiamare le API (`/api/auth/login`, `/api/tasks`, ecc.);
- Se vuoi una versione moderna: usa **React** con `useEffect` e `useState`.

Consigli

- Salva il token JWT nel `localStorage`;
- Mostra messaggi chiari se il token è scaduto o mancante;
- Crea una UI pulita e leggibile, anche minimale.

Bonus extra

- Modalità Dark/Light;
- Filtri per stato dei task (completati, in corso);
- Anteprima delle immagini caricate.

📘 2. Documentazione API con Swagger

Obiettivo: imparare a documentare le API come in un progetto reale.

Cosa fare

Integra **Swagger** nel tuo backend per generare una pagina interattiva (tipo Postman nel browser).

Come farlo

1. Installa:

   ```
   npm install swagger-ui-express yamljs
   ```

2. Crea un file `docs/openapi.yaml` con la descrizione delle tue rotte principali:
   - `/api/auth/register`
   - `/api/auth/login`
   - `/api/tasks`
3. Monta la documentazione su una rotta pubblica, ad esempio:

   ```
   app.use('/api/docs', swaggerUi.serve, swaggerUi.setup(spec));
   ```

Cosa impari

- Scrivere documentazione leggibile per altri sviluppatori;
- Mostrare come usare le tue API anche senza Postman;
- Abituarti a standard come **OpenAPI** (usato in ambienti professionali).

3👩‍💼Dashboard Admin (ruoli e permessi)

Obiettivo: aggiungere un livello di **autorizzazione avanzata**.

Cosa fare

1. Introduci un campo `role` nel modello `User` (es. `"user"` o `"admin"`).
2. Aggiungi un middleware `isAdmin` che verifica:

   ```
   if (req.user.role !== 'admin') return res.status(403).json({ error:
   'Accesso negato' });
   ```

3. Crea rotte admin protette:
   - `GET /api/admin/users` → lista di tutti gli utenti;
   - `GET /api/admin/tasks` → tutti i task esistenti;
   - `DELETE /api/admin/users/:id` → rimuovere un utente.

- Crea una dashboard visiva (HTML o React) che consenta all’admin di gestire utenti e task. Usa il token JWT di un admin per accedere a queste rotte.

Cosa impari

- Differenziare i ruoli (utente vs amministratore);
- Applicare controlli di accesso granulari;

- Simulare logiche aziendali reali.

🔄4. Refresh Token e Password Reset via Email

Obiettivo: rendere il sistema di autenticazione più robusto e realistico.

Cosa fare

1. **Refresh Token**
    - Genera due token:
        - `accessToken` (breve durata, es. 15 min),
        - `refreshToken` (più lungo, es. 7 giorni).
    - Crea una rotta `POST /api/auth/refresh` che accetta un refresh token valido e restituisce un nuovo access token.
2. **Password Reset**
    - Aggiungi un campo `resetToken` temporaneo al modello `User`;
    - Crea due rotte:
        - `POST /api/auth/request-reset` (invia email con link di reset);
        - `POST /api/auth/reset-password` (aggiorna password se il token è valido).
    - Usa un servizio email gratuito (es. **Mailtrap**, **SendGrid**, **Nodemailer**).

Cosa impari

- Gestire la **scadenza e rinnovo dei token JWT**;
- Comprendere come funzionano i flussi di recupero password;
- Aggiungere funzionalità di sicurezza reali come nei servizi professionali.

Variante “Pro Plus”: unisci tutto in un mini ecosistema

> Solo per i più curiosi o chi vuole usare il progetto nel portfolio.

Idea:
Crea un mini sistema composto da:

- Un backend completo (API Express con JWT e upload);
- Un frontend React che consuma l’API;
- Un account admin con dashboard riservata;
- Un database MongoDB Atlas condiviso;
- Documentazione con Swagger ospitata online.

Risultato finale:
Un’applicazione *full-stack reale*, con autenticazione, CRUD, ruoli e documentazione, pronta da mostrare in colloqui o come portfolio.

Consigli per affrontarle

- Parti **da una sola variante**, non da tutte.
- Lavora in piccoli passi: ogni feature completata → testala su Postman.
- Usa Git in modo disciplinato:
    - `feat(frontend-dashboard): login UI`
    - `feat(admin): route GET /users`
- Scrivi README aggiornato con le nuove feature.

Obiettivo finale del modulo avanzato

Al termine di queste estensioni, dovrete:

- Avere una **API pubblica documentata e sicura**;
- Capire come **frontend e backend comunicano via token JWT**;
- Essere in grado di **aggiungere ruoli e logiche di business**;
- Sapere **come migliorare e mantenere** un progetto reale nel tempo.