Questo progetto vi guiderà a comprendere le basi del backend pur lavorando solo con tecnologie frontend (HTML, CSS, JavaScript). L'obiettivo è mostrare come il frontend comunichi con un server attraverso API pubbliche, gestendo richieste, risposte, errori e stati.

Cosa costruirete

Una **mini dashboard "Hello Backend (senza backend)"** con 3 aree:

1. **Città → Ora + Meteo**
    - Selezione tra *Rome, London, New York*.
    - Mostra **ora locale** (WorldTimeAPI) e **meteo attuale** (Open-Meteo).
2. **Echo Playground (GET con query)**
    - Invia un testo e visualizza cosa "vede" il server (httpbin).
3. **Post finti (GET/POST)**
    - Crea un **post fake** (JSONPlaceholder) e carica una lista di post.

Obiettivi didattici

- Comprendere **HTTP** (metodi, query string, status code).
- Usare **fetch**, `async/await`, e gestire **loading/success/error**.
- Leggere un **JSON** e mostrare solo i campi utili.
- Ragionare su **CORS** e interfacce di un backend (API).

Cosa dovete fare passo-passo

1. Aprire `index.html` (meglio con **Live Server** su VS Code).
2. Provare subito:
    - Selezionare città → **Aggiorna** → vedere **ora** e **meteo**.
    - Echo → inviare testo → vedere i **parametri ricevuti**.
    - Post → compilare e **Crea post**, poi **Carica lista post**.
3. Leggere `script.js` e **commentare** le parti di `fetch` (dove si costruisce l'URL, dove si controlla `res.ok`, cosa si stampa a schermo).
4. Migliorare la UX: testi di caricamento ("Carico…/Invio…"), messaggi di errore chiari.

User stories minime (da soddisfare)

- Come utente, seleziono una città e vedo **ora** e **meteo**.
- Come utente, invio un testo e vedo **echo** con i parametri ricevuti.
- Come utente, creo un **post** e vedo una **lista** (anche parziale).

Criteri di accettazione

- Ogni sezione mostra **loading**, poi **risultato** o **errore**.
- L’output è “pulito”: solo i campi importanti.
- Il codice ha **commenti** dove avviene la logica di rete.

Estensioni (se finisci in anticipo)

- Spinner animato e **toast** per successi/errori.
- **Dark mode** toggle.
- Cronologia: salva e mostra le **ultime 5 risposte**.
- Card per i post (UI più ricca).
- Selettore con **più città**.

Modalità di consegna

Ogni studente dovrà consegnare una cartella con i file `index.html`, `styles.css`, `script.js` e un breve file `README.txt` che spieghi cosa ha imparato e quali eventuali estensioni ha realizzato. Il progetto potrà essere testato direttamente aprendo l’HTML in un browser, senza necessità di server locale