Car Hub Event

◇ FRONTEND (React)

🎯 Funzionalità:

- Visualizzazione elenco eventi (locandine, nome, data, luogo con : griglia o lista di Card)
- Filtri per regione/provincia
- Pagina dettagli evento : dettagli, immagine, bottone "Salva"
- Salvataggio evento nei preferiti (localStorage → poi collegamento a DB)

- Login/registrazione utente (opzionale)
- Area personale con lista preferiti
- Dashboard admin con interfaccia UI (opzionale)
- Navigazione tra pagine (React Router)

📦 Componenti React:

- App.js: gestione delle rotte
- Header / Footer
- EventCard: singola card evento
- EventList: griglia di eventi
- EventDetail: dettagli dell'evento
- Filters: filtri per regione/provincia
- FavoriteButton: aggiunge/rimuove dai preferiti
- LoginForm / RegisterForm: autenticazione utente
- FavoritesPage: elenco eventi salvati
- AdminPanel: gestione eventi (solo per admin)
- ProtectedRoute: accesso limitato a utenti autenticati

◇ BACKEND (Node.js + Express + MongoDB e Postman)

🎯 Funzionalità:

- API REST per:
    - Ottenere lista eventi
    - Ottenere dettagli evento
    - Filtrare per zona
    - Gestire preferiti per utente
    - Autenticazione: login/registrazione
    - CRUD eventi (Admin)
  - Middleware per autenticazione (JWT)
- Validazione input (es. Express Validator)
- Connessione al database (MongoDB)

ROADMAP COMPLETA – Car Hub Event

FASE 1 – BACKEND: Setup e API eventi (Mock data)

Obiettivo: creare un server Express funzionante con rotte base per gli eventi.

- Inizializza progetto Node.js: npm init -y
- Installa dipendenze: express, cors, dotenv, nodemon
- Crea struttura cartelle (vedi sezione precedente)
- Crea server.js con rotta /api/events (GET)
- Crea file events.json con dati mock
- Implementa controller per:
  - Ottenere tutti gli eventi
  - Filtrare per regione/provincia
  - Ottenere evento per id

FASE 2 – BACKEND: Aggiunta DB ed eventi dinamici

Obiettivo: spostare i dati dal file a un database.

- Scegli un DB (consigliato: MongoDB)
- Installa mongoose (se usi MongoDB)
- Crea modelli (eventModel.js)
- Connetti il server al DB
- Implementa CRUD eventi:
  - POST /api/events (creazione)
  - PUT /api/events/:id (modifica)
  - DELETE /api/events/:id (elimina)

FASE 3 – BACKEND: Autenticazione utenti

Obiettivo: permettere a utenti di registrarsi e loggarsi.

- Installa bcrypt, jsonwebtoken
- Rotte:
  - POST /api/register
  - POST /api/login (JWT)
- Crea userModel.js
- Middleware auth.js per proteggere le rotte private

FASE 4 – BACKEND: Gestione preferiti

Obiettivo: ogni utente può salvare eventi preferiti.

- Rotte:
  - GET /api/user/favorites
  - POST /api/user/favorites
  - DELETE /api/user/favorites/:id
- Aggiorna userModel con campo favorites: [eventId]
- Proteggi rotte con auth.js

💻 FASE 5 – FRONTEND: Setup base in React

📍 Obiettivo: impostare struttura pagine e componenti.

- Inizializza app React
- Installa react-router-dom, axios
- Crea componenti principali:
  - App.js con router
  - Header, Footer
  - EventList, EventCard, EventDetail
  - Filters, FavoriteButton
  - LoginForm, RegisterForm, FavoritesPage
  - AdminPanel (facoltativo)

🔗 FASE 6 – FRONTEND: Connessione al Backend

📍 Obiettivo: integrare dati reali via API.

- Usa axios per recuperare eventi (/api/events)
- Implementa dettagli evento (/api/events/:id)
- Aggiungi filtri
- Gestisci salvataggio nei preferiti (inizialmente in localStorage, poi DB)
- Login/logout + token JWT

🌓 FASE 7 – EXTRA: Dashboard Admin (facoltativa)

📍 Obiettivo: permettere agli admin di gestire eventi dal frontend.

- Interfaccia AdminPanel
- Form per creare/modificare eventi
- Integrazione con le rotte protette del backend
- Controllo accessi (ruolo utente)