**Cal Blay Pro App**

**1. Què fa l'app?**

L'aplicació **Cal Blay Pro** permet als responsables d'esdeveniments consultar en temps real tota la informació relacionada amb cadascun d'ells: des d'un cercador simple fins al detall de personal, incidències i documents associats.

* **Cerca d'esdeveniments**: Per nom o per rang de dates (dia o període), amb control d'errors de dates.
* **Llista d'esdeveniments**: Agrupats per data i total de pax (participants), amb botó de tornada fix.
* **Vista de detall**: En clicar un esdeveniment, mostra pestanyes per:
  + **Full d'encàrrec** (attachments de Google Calendar que comencen per "full enc").
  + **Personal**: Llegeix un CSV publicat en Google Sheets, filtra per nom d'esdeveniment, agrupa per departament, i mostra nom, hora entrada, hora sortida, hores totals (format H:MMh), icones de responsable (🎓) i carnet (🚚).
  + **Incidències**: Llegeix un altre CSV, filtra per esdeveniment i llista missatges d'incidència.
  + **Fitxes tècniques**: Resta d'adjunts de Calendar.
  + **Pressupost** i **Contracte**: Accés a carpetes de Google Drive protegides per OAuth; si no s'ha fet login, mostra una pantalla de GoogleLogin i botó de cancel·lar.

**2. Com funciona per dins?**

* **React + Tailwind** per un UI responsiu i modular.
* **XLSX** (SheetJS) per parsejar CSVs des de Google Sheets.
* **Google Calendar API** per obtenir events amb fetch i paràmetres dinàmics.
* **Google Drive API + OAuth** per carregar documents de carpetes concretes, filtrant pel nom de l'esdeveniment.
* **React Hooks** (useState, useEffect, useCallback) per gestionar estats, efectes i dependències.
* **LoginPage** amb @react-oauth/google per la finestra de login només al tab de pressupost/contracte.

**3. Què queda per fer?**

* **Arreglar parseig CSV**: l'ús de XLSX.read(..., {type:'string'}) sobre CSV fa errors HTML (could not find <table>). Cal:
  + Canviar a XLSX.utils.csv\_to\_sheet() o usar papaparse.
* **Optimitzar React Hooks**:
  + Envoltar isMatch amb useCallback per satisfer react-hooks/exhaustive-deps.
* **Depuració de warnings**:
  + Eliminar imports no usats.
  + Afegir eslint-disable-next-line si és necessari.
* **Millores d'estil**:
  + Actualitzar index.css amb una tipografia més professional (Inter, Montserrat).
  + Ajustar paleta de colors i modes (dark/light).
  + Refinar disseny de botons i pestanyes perquè siguin més visuals i coherent en mòbil.
* **Experiència d'usuari**:
  + Spinner o loader durant les crides fetch.
  + Missatges d'error amigables quan fallen les API.
* **Tests i QA**:
  + Proves unitàries per utilitats com isMatch.
  + Tests d'integració per assegurar que cada tab carrega contingut.
* **Desplegament**:
  + Configurar entorn de staging.
  + Pipeline CI/CD per validació i linter abans de producció.

**4. Següents passos**

1. **Arreglar CSV parsing** i donar formats correctes a les dades.
2. **Revisar i tancar warnings** d'ESLint i React Hooks.
3. **Polir l'estil global** (fonts, colors, responsive).
4. **Afegir feedback UX** (loaders, errors, animacions suaus).
5. **Escriure tests** bàsics i provar en diferents dispositius.
6. **Desplegar a staging** i fer una ronda d'usuaris pilots.
7. **Posada en producció** i manteniment: monitoratge d'errors i autoactualització de dades.

Amb aquest full de ruta tens una visió clara del que ja tenim implementat, què falta polir i com avançar en les pròximes fases. Si vols exemples de codi, ajuda amb algun pas concret o prioritzar tasques, digues-m'ho!