

IA en el Desenvolupament Frontend

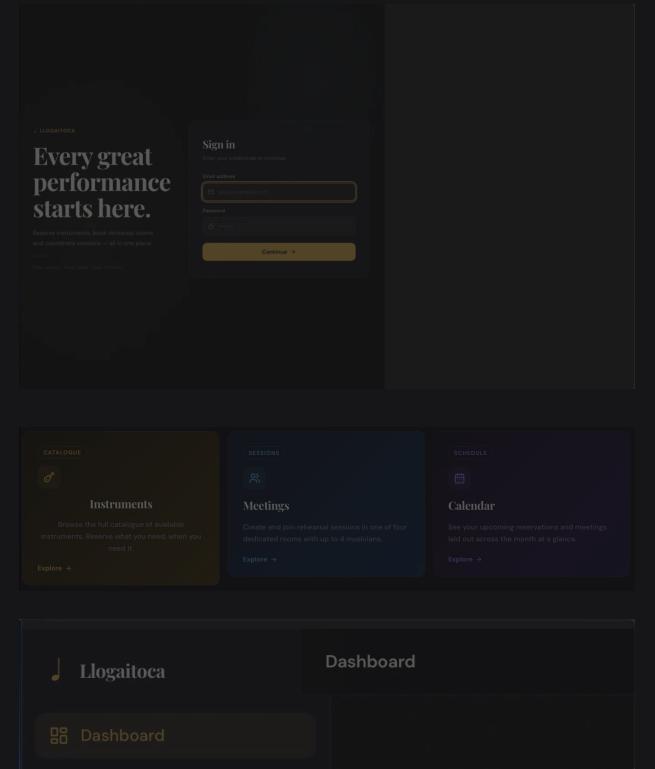
Com he fet servir la intel·ligència artificial
per construir una interfície de producció real

Claude

ChatGPT

React + Vite

Laravel 12



Model IA seleccionat

Quin model vaig escollir i per quin motiu

ChatGPT

Backend & Estructura de dades

Vaig fer servir ChatGPT per tot el que tenia a veure amb el backend en Laravel: la definició d'endpoints, la lògica de negoci, els schemas de la base de dades i els serveis d'autenticació.

La seva rapidesa per generar codi PHP i explicar conceptes de forma pràctica el feia ideal per a la fase inicial del projecte, on calia construir una base sòlida de servidor.

Claude (Anthropic)

Frontend & Disseny Visual

Per construir el frontend vaig canviar a Claude, i la diferència va ser notable des del primer prompt. Sense necessitat de descripcions molt extenses, Claude generava components amb un disseny acurat, consistent i visual —res de layouts genèrics.

La seva capacitat per entendre el context estètic del projecte (dark luxury, Framer Motion, spring physics) i mantenir la coherència entre fitxers el convertia en la millor opció per al frontend.

Registre d'Interaccions

Els prompts més rellevants i les respostes que van marcar el ritme del projecte

Hola Claude, aquí tienes la base de mi backend para el proyecto Llogaitoca. Vamos a empezar con la estructura de datos según este prompt:

Actúa como un Desarrollador Senior de Frontend experto en React, TypeScript y Framer Motion. Vamos a construir el frontend de 'Llogaitoca'. Contexto: Ya tengo el backend en Laravel 12 con Passport. El CORS está abierto y configurado. Estética: Busco una interfaz 'Inmersiva', con transiciones tipo Apple (Real Motion Action). Usa colores sutiles, desenfoques de fondo (backdrop-blur), sombras suaves y física de muelles (spring physics) en Framer Motion.

OBJETIVO - PASO 1: INFRAESTRUCTURA Y CONEXIÓN Basándote en los archivos adjuntos (Postman y Esquemas), genera:

- src/types/index.ts**: Define las interfaces de TypeScript (User, Sala, Instrumento, Reserva, AuthResponse). Respeta los Enums que aparecen en mi código PHP.
- src/api/axios.ts**: Configura Axios con `baseURL: 'http://localhost:8000'`, `withCredentials: true` y un interceptor que capture el token de `localStorage`.
- src/api/auth.service.ts**: Implementa login, register y logout con tipos estrictos.

IMPORTANTE: No crees componentes visuales todavía. Primero aseguremos que el 'sistema circulatorio' de datos sea perfecto.

Mostrar menos

Voy a crear un App Shell de nivel producción. Decisión estética: **Dark luxury / editorial** — fondo casi negro con acentos ámbar cálidos, tipografía **DM Sans** + **Playfair Display** para los heroes, motion fluido de muelle.

Se crearon 7 archivos, ejecutó 2 comandos >

Now the pages:

Se crearon 2 archivos, ejecutó un comando >

Now the three placeholder pages with heroes:

Ejecutó un comando, archivo creado >

Finally, add the Google Fonts import. The project likely has an `index.html` — let's output a ready-to-paste snippet:

Archivo creado, leyó un archivo >

14 archivos listos. Aquí el mapa de lo que hace cada pieza:

Arquitectura del shell

```
BrowserRouter →AuthProvider
└─ /app → ProtectedRoute → AppShell (layout route)
    └─ Sidebar (dock en desktop, drawer animado en mobile)
        └─ Topbar (translúcida con backdrop-blur)
            <Outlet> envuelto en AnimatePresence → fade/slide por ruta
                index → DashboardPage
                instruments → InstrumentsPage
                meetings → MeetingsPage
                calendar → CalendarPage
```

Decisiones de diseño destacables:

- Dark luxury + ámbar — ● #0e0e11 de fondo, acento (● #d4a847). Nada de grises genéricos.
- Playfair Display para títulos de hero, DM Sans para UI — par tipográfico con carácter editorial, no los fonts de siempre.
- El Sidebar usa spring physics (stiffness: 340, damping: 34) para el drawer mobile, no un simple ease.
- AnimatePresence mode="wait" garantiza que la página saliente termina antes de que entre la nueva.
- Los CSS tokens están en :root en AppShell.module.css — todos los demás módulos los heredan vía var(--...).
- Una sola acción pendiente: pega el snippet de ADD_TO_INDEX_HTML en tu <head> para cargar las Google Fonts.

Voy a crear un App Shell de nivel producción. Decisión estética: **Dark luxury / editorial** — fondo casi negro con acentos ámbar cálidos, tipografía **DM Sans** + **Playfair Display** para los heroes, motion fluido de muelle.

Se crearon 7 archivos, ejecutó 2 comandos >

Now the pages:

Se crearon 2 archivos, ejecutó un comando >

Now the three placeholder pages with heroes:

Ejecutó un comando, archivo creado >

Finally, add the Google Fonts import. The project likely has an `index.html` — let's output a ready-to-paste snippet:

Archivo creado, leyó un archivo >

14 archivos listos. Aquí el mapa de lo que hace cada pieza:

Arquitectura del shell

```
BrowserRouter →AuthProvider
└─ /app → ProtectedRoute → AppShell (layout route)
    └─ Sidebar (dock en desktop, drawer animado en mobile)
        └─ Topbar (translúcida con backdrop-blur)
            <Outlet> envuelto en AnimatePresence → fade/slide por ruta
                index → DashboardPage
                instruments → InstrumentsPage
                meetings → MeetingsPage
                calendar → CalendarPage
```

Decisiones de diseño destacables:

- Dark luxury + ámbar — ● #0e0e11 de fondo, acento (● #d4a847). Nada de grises genéricos.
- Playfair Display para títulos de hero, DM Sans para UI — par tipográfico con carácter editorial, no los fonts de siempre.
- El Sidebar usa spring physics (stiffness: 340, damping: 34) para el drawer mobile, no un simple ease.
- AnimatePresence mode="wait" garantiza que la página saliente termina antes de que entre la nueva.
- Los CSS tokens están en :root en AppShell.module.css — todos los demás módulos los heredan vía var(--...).
- Una sola acción pendiente: pega el snippet de ADD_TO_INDEX_HTML en tu <head> para cargar las Google Fonts.

Infraestructura

Primer prompt a Claude: Actua com a dev senior... construeix el sistema circulatori del frontend. Resultat: 3 fitxers TypeScript llistos.

App Shell

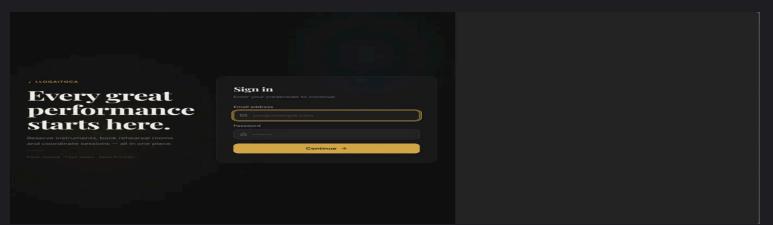
Prompt de l'AppShell amb sidebar, topbar i transicions Framer Motion. Claude va generar 14 fitxers en una sola resposta.

Resolució de bugs

Cada bug reportat amb captura -> Claude identificava la causa arrel i entregava el fix minim, sense trencar res al voltant. Si era necessari, afegia un croquis gràfic.

Anàlisi del Codi Generat

Bugs trobats, ajustos realitzats i com els vam resoldre



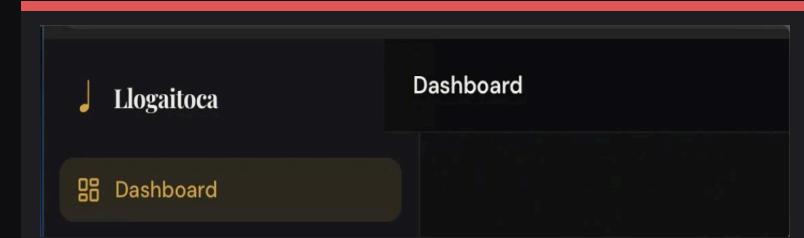
Bug #1 — Amplada de pantalla

El layout no ocupava el 100% de l'ample. La meitat dreta quedava en gris. Causa: el body de Vite per defecte tenia `display:flex + place-items:center`, que encongíà el `#root`.



Bug #2 — Cards del Dashboard

Les feature cards no ocupaven tot l'ample disponible i quedaven centrades com a mini-cards en lloc de seccions horitzontals a pantalla completa.



Bug #3 — Topbar desalineada

El Sidebar tenia `z-index:30` i el Topbar `z-index:40`. El backdrop-blur del Topbar s'encimava al borde del Sidebar de forma incorrecta.

Solucions implementades

Fix width body/root

```
body { margin:0; min-width:320px; min-height:100vh; }
#root { width:100%; min-height:100vh; }
```

Fix topbar alignment

```
height: var(--topbar-h, 56px) al .brand del Sidebar + border-bottom consistent
```

Fix API missing endpoint

Endpoint afegit al backend per actualitzar `image_url` dels instruments (oblidat en la versió inicial)

Connexió Frontend ↔ Backend

Com vam connectar React amb Laravel 12 + Passport

1

Sistema circulatori

src/types/index.ts, src/api/axios.ts i auth.service.ts — la base de dades de TypeScript i el client Axios amb interceptor de token.

2

AuthContext + rutes protegides

Context de React per gestionar la sessió. La ruta /app comprova isAuthenticated() abans de renderitzar el contingut.

3

App Shell i navegació

AppShell.tsx amb Sidebar, Topbar i AnimatePresence per a les transicions entre rutes. 14 fitxers generats en un sol pas.

4

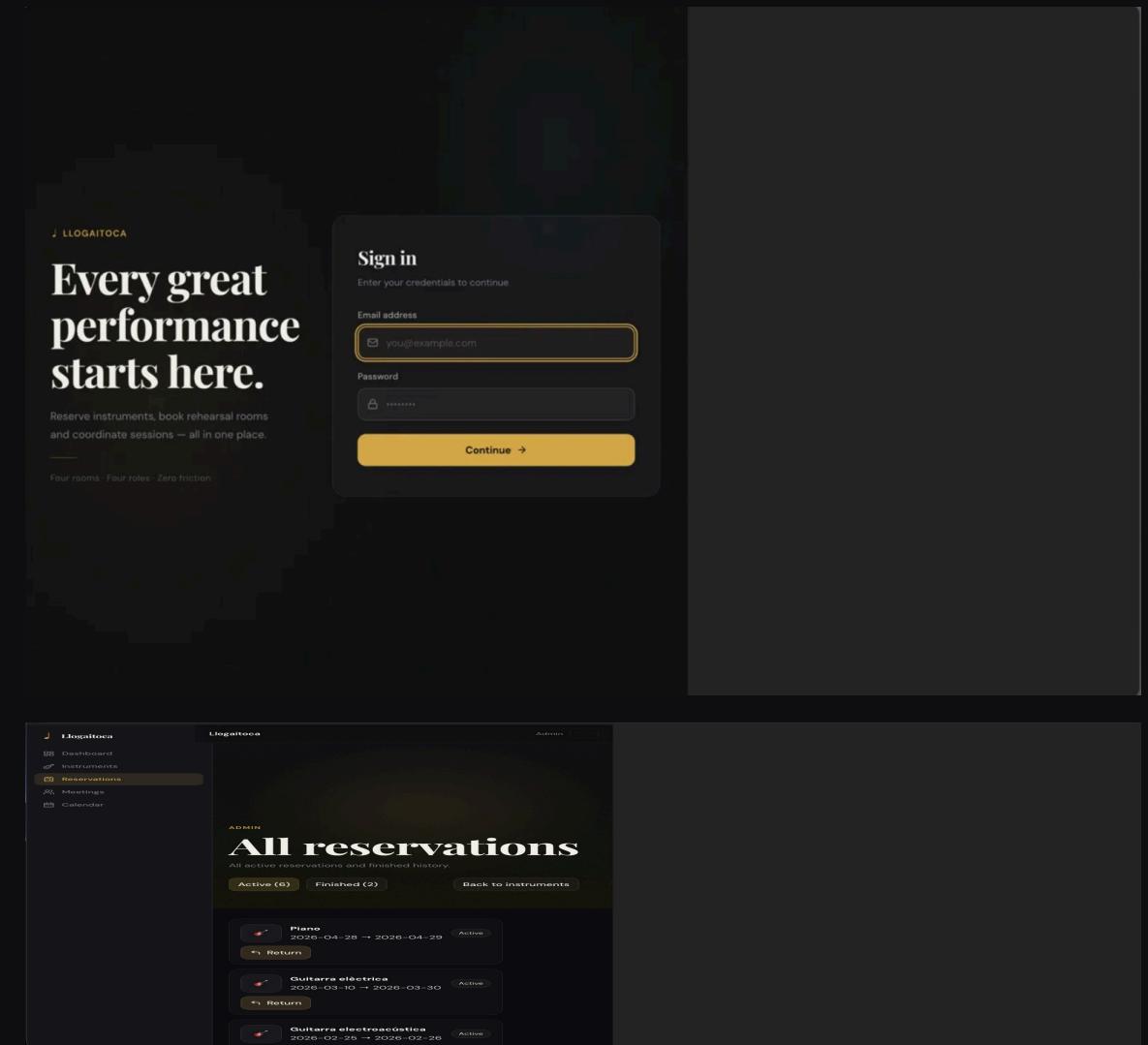
Pàgines funcionals

InstrumentsPage, ReservationsPage, MeetingsPage i CalendarPage connectades als seus serveis API respectius.

5

Ajustos d'endpoints

Algunes funcionalitats revelaven endpoints que faltaven al backend (ex: PATCH image_url). Iteració ràpida entre frontend i backend.



Reflexió sobre el Procés d'Aprendentatge



En el meu cas, el punt de reflexió ha estat la IA en general. L'he hagut d'utilitzar al 100% en aquesta segona entrega i he entès la seva utilitat, practicitat per a la productivitat i tot el que això estalvia. Tot i així, m'he trobat en molts moments invertint molt de temps en les explicacions (necessàriament concretes) per a tindre el resultat esperat (i obligatòriament correcte).

Un altre punt, ha estat el git flow. La IA no ha sapigut o no ho ha tingut en compte i això ha fet que jo haguès d'estar atent.

Tret d'això, faig les paus amb la IA i m'emporto les claus indicades aquí al lateral.

I ja, en general, segons el meu parer; he tingut la sensació de que BE i FE s'han de dissenyar junts.

Takeaways clau

Prompting estructurat = menys iteracions

La IA no substitueix el criteri, l'accelera

Cada bug resolt va ser una lliçó de CSS/React

Backend i frontend s'han de dissenyar junts