Plan de trabajo con Codex

Nexus Boards v1

Mapa operativo con prompts y Definition of Done para cada bloque de trabajo. Incluye preparación de carpetas y GitHub, scaffolds de API/Frontend, persistencia, integraciones con calendarios, pruebas, CI/CD y documentación.

Generado: 12 Sep 2025, 10:00 CST

0) Preparación local + GitHub

Objetivo

- Dejar lista la estructura base del repo y el entorno de trabajo.

Prompts/órdenes para Codex

- Crea carpetas en `~/Projects/nexus-boards`: ``` nexus-boards/ apps/ api/ web/ docs/ infra/ scripts/ .github/workflows/ ```
- Inicializa git, crea `.gitignore` (Node/Quasar/Playwright/Prisma/SQLite) y rama `main`.
- Genera `README.md` minimal con visión, stack y cómo correr `dev`.
- Crea repo GitHub privado `Adal612Git/nexus-boards`, añade remoto y primer push.
- Agrega `docs/Plan_Maestro.pdf` y enlázalo en README.
- Crea `CODEOWNERS`, `LICENSE (MIT)`, `CONTRIBUTING.md`.

DoD: repo en GitHub con estructura, README y PDF; `.gitignore` correcto; push OK.

1) Meta-setup del monorepo

Prompts

- Crea `package.json` workspaces (apps/api, apps/web) + scripts `dev`, `build`, `test`, `lint`, `format`.
- Añade Prettier + ESLint (TS/Vue) y `editorconfig`.
- Crea `Makefile` con atajos (`make dev`, `make api`, `make web`, `make test`, `make e2e`, `make demo`).
- Agrega `.env.example` raíz con variables comunes.

DoD: `make` muestra objetivos y corre sin errores.

2) Backend scaffold (apps/api)

Prompts

- Inicializa TypeScript y `nodemon`.
- Instala dependencias: express, zod, pino, pino-http, cors, helmet, dotenv, jsonwebtoken, bcrypt, prisma, @prisma/client, supertest, vitest, tsx.
- Estructura `src/` con `app.ts`, `server.ts`, `config/`, `core/http/`, `modules/{auth,projects,cards,calendar,dashboard}`.
- Routers vacíos por endpoint y `openapi/openapi.yaml` skeleton con script `docs:serve`.

DoD: API corre en dev y OpenAPI visible en Swagger/Redoc.

3) Persistencia y migraciones (Prisma + SQLite)

Prompts

- Define `schema.prisma` con `User, Project, Card, CalendarEvent` + enums y los índices críticos.
- Crea `db:migrate` y `db:seed` (usuario/proyecto/12 cards).
- Servicio `reorder` transaccional (stub).

DoD: migraciones y seeds funcionan; datos demo listos.

4) Autenticación y autorización

Prompts

- Implementa `AuthService` con JWT (access 15m, refresh 7d) + rotación y rate■limit en login.
- Endpoints `/login` y `/password-reset` (token 15 min).
- Middleware `auth` con ownership por `ownerId`.

DoD: tests de login/reset y rutas protegidas pasando.

5) Casos de uso núcleo

Prompts

- Proyectos: create/list/delete con ownership.
- Cards: CRUD + archive.
- Reordenar: transacción atómica; 409 si colisión.
- SetDueDate: normaliza TZ; 'all■day' y duración default 1h.
- DashboardMetrics: conteos por estado y vencidos.

DoD: supertest + vitest listos; cobertura dominio ≥85%.

6) Integraciones externas (Google/Outlook)

Prompts

- Interfaz `CalendarAdapter` y adaptadores Google/Outlook (primero mocks).
- Conexión: connect, pick calendar, refresh, disconnect.
- Sync una vía: create/update/delete con backoff y estados `Synced | Pending | Error`; usa `X■Idempotency■Key`.
- Simuladores de 429/503/401 para pruebas.

DoD: pruebas con mocks pasando; limpieza al borrar fecha verificada.

7) Frontend scaffold (Quasar/Vue/Tailwind)

Prompts

- Quasar + TS; instala tailwindcss, pinia, axios, vue-draggable-next, vue3-carousel, @axe-core/playwright.
- Páginas: Login, Dashboard, Boards, Calendar, Settings.
- Componentes Kanban/Calendar/Dashboard; Axios con interceptor de tokens.
- DnD accesible (teclado) + estados de UI y slideshow local.
- Offline básico: `pendingOps[]` y reconciliación.

DoD: `quasar dev` levanta UI navegable; a11y básica OK.

8) Observabilidad y seguridad

Prompts

- Pino con `requestId`; sanitiza PII en logs.
- Helmet + CORS; validaciones Zod; `errorHandler` uniforme.
- Script de demo para correlación de logs.

DoD: logs estructurados sin PII; 422 y headers de seguridad correctos.

9) Pruebas y calidad

Prompts

- Vitest + Supertest en API; Vitest + Vue Test Utils en Web.
- Playwright (E2E): flujo login→proyecto→cards→DnD→fecha→sync.
- Integrar axe para a11y en E2E.
- Badges de cobertura en README.

DoD: CI local pasa; cobertura dominio ≥85%, UI ≥70%.

10) CI/CD y empaques

Prompts

- Workflow `.github/workflows/ci.yml`: lint, typecheck, unit, integration, e2e.
- Docker opcional (`infra/api.Dockerfile`, `infra/web.Dockerfile`) y `docker-compose.yml` de demo.
- Publicar OpenAPI y colección en artefactos de CI.
- `make demo` para levantar todo con seeds.

DoD: CI verde y artefactos generados.

11) Documentación y artefactos

Prompts

- README completo: instalación, variables, scripts, enlaces a OpenAPI y Plan PDF.
- `docs/traceability.md` (matriz requisito→test) y `docs/qa-checklist.md` (20 pasos).
- `CHANGELOG.md` y `VERSION` (0.1.0).

DoD: docs legibles y checklist ejecutable.

12) Issues NB-### (seed de backlog)

Prompts

- NB■001 Auth: Login (BDD + CA)
- NB■002 Auth: Reset contraseña
- NB■010 Proyectos: Crear/Lista/Eliminar
- NB■020 Cards: CRUD + Archive
- NB■030 Cards: Reordenar (transacción, 409)
- NB■040 Cards: Fecha (all■day/rango)
- NB■050 Calendar: OAuth + pick calendar
- NB■051 Calendar: Sync una vía + reintentos
- NB■060 Dashboard: KPIs + próximos 7 días
- NB■070 Web: DnD accesible + estados UI
- NB■080 Web: Calendario agregado
- NB■090 Observabilidad: Pino + requestId
- NB■100 Seguridad: Helmet/CORS/Zod
- NB■110 QA: E2E Playwright + axe
- NB■120 CI: Workflow GitHub Actions
- NB■130 Docs: README, OpenAPI, Checklist

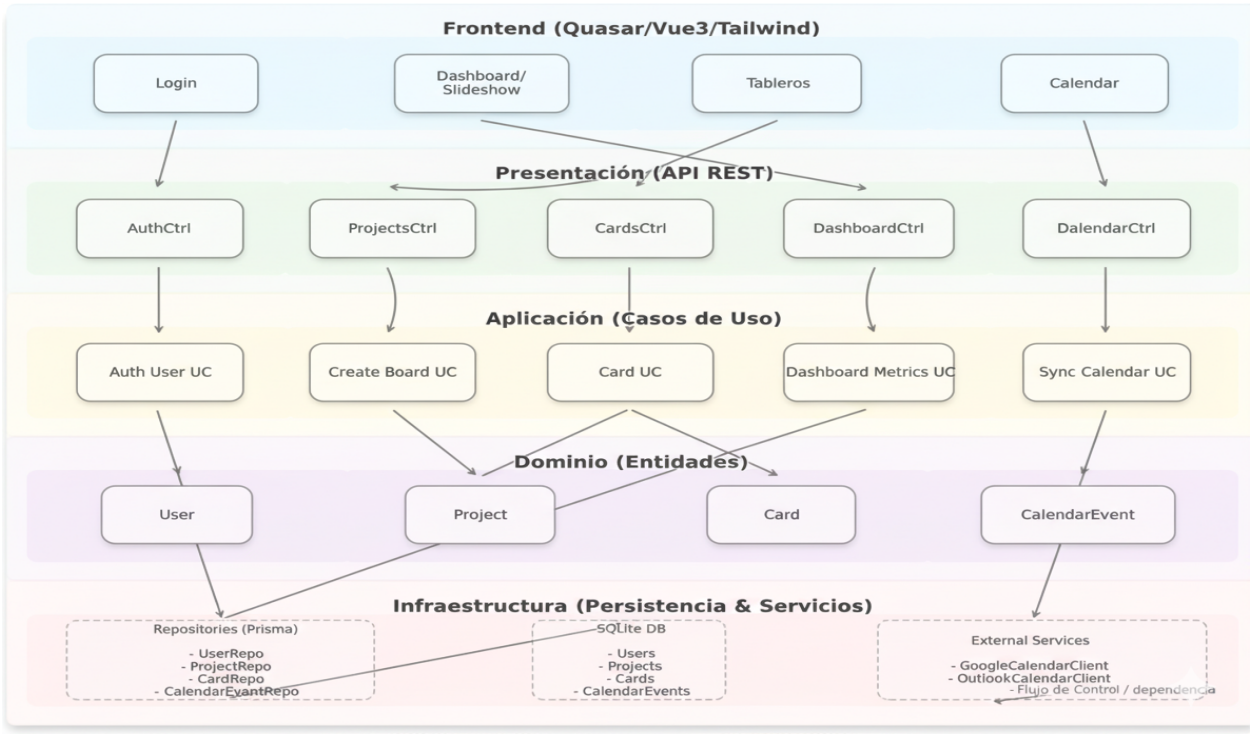
DoD: issues con título, BDD y criterios de aceptación.

13) Scripts de conveniencia

Prompts

- `scripts/dev.sh` (API + Web en paralelo).
- `scripts/seed.sh` (reseeds rápidos).
- `scripts/e2e.sh` (Playwright con BASE_URL).

DoD: `make demo` deja entorno listo y pasa E2E básicos.



Stack Tecnológico — Arquitectura de Proyecto

Frontend — Interfaz de Usuario

Quasar (Vue 3) v2.x — Framework UI	Vue 3 Composition API v3.x — Sintaxis moderna	Tailwind CSS v3.x — Utilidades CSS
Pinia v2.x — Estado global	Axios v1.x — HTTP Client	Vue Draggable Next v2.x — Drag & Drop
Vue3-Carousel v0.1.x — Slideshow	Quasar Iconos — Iconos	

Backend — Lógica y Datos

Node.js LTS — Runtime	Express.js v4.x — API REST	TypeScript v4.x — Tipado estático
Prisma v4.x — ORM type-safe	SQLite v5.x — Base de datos local	JWT + bcrypt (v9.x + v5.x) — Autenticación
Zod v3.x — Validación de datos	dotenv v16.x — Variables de entorno	Pino v8.x — Logging estructurado

Integraciones Externas

Google APIs Node.js Client v100.x — Sincronización con Google Calendar	Microsoft Graph SDK v3.x — Sincronización con Outlook Calendar

Desarrollo y Calidad

VS Code — Editor principal	Git — Control de versiones
Docker (opcional) — Entorno consistente	