**Marco Portfolio — Tech Doc (PM edition, YC-top-1% concise)**

**TL;DR**

A **Next.js (App Router) frontend** that renders a portfolio **without backend** by reading JSON files from /public/data.  
All pages are assembled by a small **data loader** that normalizes the JSONs to one UI model.

**Core idea:**

* proyectos.json = canon of projects.
* Per-area tables enrich each project:
  + Leadership → resultado-proyecto.json (for tags) + liderazgo\*.json
  + Development → dev-tech-stack.json (by desarrolloId) + desarrollo\*.json
  + Design → diseno-highlight\*.json (by disenoId) + diseno\*.json
* Sections and carousels derive what to show only from those files.

**1) Scope & goals**

* **Single repo, single Next app** (frontend/).
* **No DB / backend at runtime.** Content is static JSON under /public/data.
* Project pages show: hero, purpose, beginning, results, story, media, per-role sections, extra content.
* Home shows: dynamic counters, passion cards with project counts, and three “preview” carousels.

**Non-goals (for now)**

* Auth, dashboards, APIs, mutations.
* SSR data fetching from a server DB (kept as future option).

**2) Data model (JSON files)**

Location: frontend/public/data/

| **File** | **Purpose** | **Keys that matter** |
| --- | --- | --- |
| **proyectos.json** | Base catalog of projects | `{ id, title, descripcionBreve, backgroundUrl, logoUrl, slogan, areas: ["liderazgo" |
| **resultado-proyecto.json** (or resultados.json) | Metrics used as *leadership tags* (and “The result” panel) | { id, proyectoId, orden, valor, descripcion } |
| **liderazgo.json** + liderazgo-highlight\*.json | Leadership role details | liderazgo.json rows keyed by proyectoId; highlights join by liderazgo\_id |
| **desarrollo.json** + desarrollo-highlight\*.json + **dev-tech-stack.json** | Dev role details & tags | desarrollo.json rows keyed by proyectoId; highlights join by desarrollo\_id; **tags come from dev-tech-stack rows with desarrolloId** |
| **diseno.json** + *diseno-highlight.json*\* | Design role details & tags | diseno.json rows keyed by proyectoId (also holds **previewUrl**); highlights join by **disenoId** and give 2 design tags |
| **contenido-adicional.json** | Extra content carousel per project | { id, proyectoId, imagenUrl, titulo, descripcion, enlace } → normalized to { imageUrl, title, description, href } |

The loader is tolerant to naming variations: proyecto\_id/proyectoId/project\_id, imagen\_url/image\_url/imagenUrl, diseñoId/disenoId, etc.

A diagram of a computer

AI-generated content may be incorrect.

**3) UI data model (normalized)**

The loader in src/lib/projects-loader.ts returns an array of:

type Area = "liderazgo" | "desarrollo" | "diseno";

interface UIResultado { id: string; order?: number; label: string; description?: string; count?: number; }

interface UISubSection {

videoUrl?: string;

queHice?: string;

previewUrl?: string; // from diseno.json

figmaUrl?: string;

githubUrl?: string;

projectUrl?: string;

highlights?: string[];

techStack?: string[]; // dev: from dev-tech-stack by desarrolloId; design: from diseno-highlights by disenoId

}

interface UIContenido { id: string; imageUrl?: string; title?: string; description?: string; href?: string; }

interface UIProject {

id: string; title: string;

descripcionBreve?: string; backgroundUrl?: string; logoUrl?: string;

area?: Area;

liderazgo?: UISubSection;

desarrollo?: UISubSection;

diseno?: UISubSection;

resultados?: UIResultado[];

contenidoAdicional?: UIContenido[];

tags?: string[];

}

**4) App structure**

frontend/

public/

data/ ← all JSON files here (served at /data/\*)

src/

app/

page.tsx ← home (hero, counters, carousels, passion cards)

section/page.tsx ← lists by area (?type=leader|dev|designer ⇒ maps to "liderazgo|desarrollo|diseno")

proyectos/[id]/page.tsx ← project page

components/

molecules/

PassionCard.tsx, ProjectCard.tsx, TrustItem.tsx, ...

organisms/

HeroCarousel.tsx

SectionProjectsGrid.tsx ← lists projects by area from JSON

LeaderCarousel.tsx ← preview handpicked IDs (1,3,5)

DevCarousel.tsx ← preview IDs (2,4,6)

DesignCarousel.tsx ← preview IDs (7,8,9); shows previewUrl + design tags

ProjectHero.tsx, ProjectDetails.tsx

LeaderRoleSection.tsx, DevRolSection.tsx, DesignRolSection.tsx

ContentCarousel.tsx, ContentCard.tsx

lib/

projects-loader.ts ← the single source of truth for joining JSONs

next.config.ts

package.json

**5) Rendering logic (what shows where)**

**Home**

* **Hero counters**: load /data/proyectos.json, classify by area (title/area/tags), count per area.
* **PassionSection**: same counts → “X projects”.
* **Leader/Dev/Design Carousels**:
  + Picks **specific IDs** (leader: 1,3,5 · dev: 2,4,6 · design: 7,8,9).
  + Build tags:
    - **Leader**: top 3 resultado-proyecto by orden → shows “+count label” if count is numeric.
    - **Dev**: 3 items from dev-tech-stack filtered by desarrolloId.
    - **Design**: 2 from diseno-highlight filtered by disenoId.
  + The 4th card is a “See all projects” link to /section?type=….

**Section (listing)**

* SectionProjectsGrid:
  + Loads proyectos.json.
  + Filters projects by areas array (must include "liderazgo" | "desarrollo" | "diseno").
  + Attaches **area-specific tags** (same rules as above).
  + Uses slogan as primary line and descripcionBreve as secondary where appropriate.

**Project page**

* Uses getProjects() (the loader) to get a fully hydrated UIProject.
* **Hero**: background, logo, **brand + typing animated slogan**, social links.
* **Details**:
  + Cards: “What is {brand}?”, “The purpose”, “The beginning”.
  + “The result” grid → from resultados (animated counters).
  + “The story” paragraphs from historiaBreve.
  + Optional second image.
* **Role sections**:
  + Leader: video + “qué hice” + highlights.
  + Dev: video/links + techStack (from dev-tech-stack via desarrolloId) + highlights.
  + Design: **previewUrl** (from diseno.json), figma, tools=techStack (mapped from diseno-highlight), highlights.
* **ContentCarousel**: items from contenido-adicional.json (supports imagenUrl/imageUrl aliases).

**6) Tech stack**

* **Next.js 15 (App Router)**, React 18, TypeScript
* **TailwindCSS** for styling
* **Framer Motion** for micro-animations
* **Lucide** icons
* **pnpm** as package manager
* **Vercel** hosting

**next.config.ts**

* images.remotePatterns allow GitHub RAW + jsDelivr.
* Safe redirects (/projectos/:id → /proyectos/:id, etc.).
* Lint/TS build errors ignored for velocity (can re-enable later).

**7) Architecture**

**Pure-frontend, file-backed content.**  
Pros: instant deploys, zero infra, works offline for content editing.  
Trade-offs: need redeploy to publish content changes; caching considerations.

**Data layer**: projects-loader.ts as a “joiner”:

* Loads JSON (with duplicate key tolerance).
* Groups/joins by IDs.
* Outputs stable UI shape used by all components.

You can swap the JSON with an API later by changing only this loader.

**8) How to add/edit content**

1. Create/edit files in frontend/public/data/.
2. **Add a new project** to proyectos.json:
3. {
4. "id": 10,
5. "title": "NewThing",
6. "slogan": "Short promise",
7. "descripcionBreve": "Concise description",
8. "backgroundUrl": "https://…",
9. "logoUrl": "https://…",
10. "areas": ["liderazgo", "desarrollo"] // ← IMPORTANT
11. }
12. Leadership tags: add rows into resultado-proyecto.json with proyectoId: 10 and orden 1–N.
13. Dev tags: add rows to dev-tech-stack.json with **desarrolloId** (the id of the row in desarrollo.json for this project).
14. Design tags: add rows to diseno-highlight.json with **disenoId** (the id of the row in diseno.json).
15. Design preview image: set previewUrl in diseno.json for this project.
16. Extra content: add to contenido-adicional.json with proyectoId: 10.

Tip: If a project doesn’t show in a section, 90% of the time it’s because areas doesn’t include the expected key.

**9) Running locally**

cd frontend

pnpm install

pnpm dev

# → http://localhost:3000

* Node ≥ 18 recommended.
* JSONs live at http://localhost:3000/data/\*.json.
* We disable cache in dev for JSON (no-store) so changes reflect on refresh.

**10) Deploying to Vercel**

**Critical settings**

* Project → **Root Directory**: frontend
* Framework Preset: **Next.js**
* Build Command: pnpm build
* Install Command: pnpm install --frozen-lockfile
* Output Directory: **leave empty** (Next manages it)
* Do **not** commit .next/ or .vercel/

**Common pitfall:** 404 at /data/\*.json  
→ Means Vercel did not build the **frontend** or used “Other” preset. Fix root directory and preset.

**11) Configuration & env**

* Social links (optional fallbacks used in code):
  + NEXT\_PUBLIC\_GITHUB\_URL
  + NEXT\_PUBLIC\_LINKEDIN\_URL
  + NEXT\_PUBLIC\_RESUME\_URL

**12) Quality bar & UX notes**

* **Animations** are subtle (slide/blur/scale) and gated by viewport to avoid jank.
* **Accessibility**: alt text, aria labels, keyboard focus states, readable contrast overlays.
* **Image domains** whitelisted to avoid Next/Image runtime errors.

**13) Extension roadmap (nice-to-haves)**

* **Move JSON to a headless CMS** (or a tiny API) and reuse the same loader shape.
* **Admin preview mode** guarded by a simple token, writing to draft JSON.
* **Search & filters** in the section pages (by tag / stack / year).
* **Unit tests** for the loader’s normalization.

**14) Troubleshooting**

* **New project not rendering in a section** → check areas includes the area; check tag joins use the correct IDs (desarrolloId/disenoId vs proyectoId).
* **Design preview not showing** → diseno.json must have previewUrl for that project’s diseno row.
* **ContentCarousel empty** → ensure contenido-adicional.json uses imagenUrl/imageUrl and proyectoId matches the project.
* **404 on /data/** → Vercel preset/root misconfigured (see §10).
* **Hydration warning <li> inside <li>** → keep TrustItem as <motion.li className="list-none"> and ensure the parent list maps to <ul> → <TrustItem/> directly (no nested <li> wrappers).

**15) Ownership & change velocity**

* **Single owner edits JSON → push → Vercel redeploy = live.**
* All code paths for data are centralized in projects-loader.ts. If you change the JSON schema, you only touch this file.