Guía de desarrollador · presentations-designer

presentations-designer: guía técnica

Cómo está construido el agente que diseña, verifica, exporta y publica las presentaciones comerciales de Campuslands, y cómo se mantiene.

3 nov 2026 | Guía de desarrollador | Claude Code | 25 min de lectura

Descripción general

El **Estudio de Presentaciones** (`presentations-designer`) es el agente que construye las presentaciones comerciales de Campuslands Full Service. Recibe los alcances de un cliente, propone un plan lámina por lámina, y una vez aprobado construye el deck en **HTML y CSS puros**, lo verifica con una herramienta y a la vista, lo exporta a **PDF** y lo publica en Vercel con un link para compartir con el cliente.

Esta guía explica cómo funciona por dentro: las reglas de marca que gobiernan el diseño, la estructura del repositorio, el flujo de construcción de diez pasos, el verificador automático, la exportación a PDF, el registro obligatorio en el portal y los puntos donde conviene tener cuidado. Está pensada para quien mantiene el agente o necesita entender por qué un deck se ve como se ve.

10
láminas como máximo por presentación, salvo pedido textual del usuario

8
reglas supremas de diseño, con prioridad sobre cualquier otra

2
entregas obligatorias: PDF exportado y link desplegado

1056 × 594
píxeles de cada lámina, igual al tamaño del PDF (11 × 6,19 in)

LO ESENCIAL

La marca es siempre **Campuslands**: fondo blanco, tipografía Poppins y logo a color. El cliente aporta **su logo y su contenido**, nunca su paleta. Y nada se entrega sin pasar `verificar_deck.py` y una revisión a la vista de cada lámina.

Qué hace y qué no hace

El agente hace	El agente no hace
Construye decks HTML/CSS autocontenidos a partir de una plantilla que ya cumple las reglas	Inventar cifras, nombres o datos del cliente: si faltan, los pide con un cuestionario estándar
Presenta un plan por lámina y espera la confirmación del usuario antes de diseñar	Construir antes de que el plan esté aprobado
Verifica con herramienta y a la vista, y corrige hasta cumplir	Entregar algo que no haya pasado ambas verificaciones
Exporta el PDF, registra el deck en dos lugares, hace commit y push y comparte el link	Teñir las láminas con los colores del cliente o usar tipografías distintas de Poppins
Señala al usuario cuando una petición contradice el Brandbook	Superar las 10 láminas sin un pedido textual

El usuario cura, dirige y aprueba; el agente hace todo el trabajo de construcción, verificación, despliegue y mantenimiento de la wiki.

Portadas y clientes

Las portadas de esta galería salen de la primera página del PDF de cada deck en `presentaciones/decks/`, y solo se muestran las del diseño vigente (sistema v4, desde Globant Dojo, 29 de septiembre de 2026), en orden cronológico; los logos, de `resources/companies-logos/`, recortados y convertidos a WebP (con sus colores originales). Las imágenes se descargan bajo demanda: solo cuando el carrusel se acerca a la pantalla y, a partir de ahí, la activa y sus vecinas; en el PDF se cargan todas. Ambos se regeneran con `guides/_src/recursos_web.py` y la lista que se muestra está en `PORTADAS` y `LOGOS` de `guides/_src/build.py`.



Globant · Modelo de desarrollo con dojo



Financiera Comultrasan · Asistente Conversacional con IA Generativa



Zona Franca Santander · Ecosistema Campuslands en el Hub Tecnológico



Alcaldía de Girón · Módulo de Cobranza con Inteligencia Artificial



Palmera Junior · Plataforma segura y vigente con Laravel 13

Alcaldía de Girón

Avicampo

Chico Soluciones Logísticas

ColBeef

Compumax

Fundación Cardiovascular

Financiera Comultrasan

GasPaís

Green Metal

Marval

Mchaileh Inmobiliaria

Miami Aqua Tours

Multinal

Palmera Junior

Unidrogas

Cómo corre: Claude Code y `CLAUDE.md`

El agente corre en **Claude Code** y se gobierna con `CLAUDE.md`, el archivo que lee al inicio de cada sesión y que se co-evoluciona con el usuario. Fue reconceptualizado el 2026-10-02 a partir del Brandbook de Campuslands: donde algo de la wiki lo contradiga, **gana** `CLAUDE.md`.

Sus secciones, en orden:

1. **Mi rol:** agente de presentaciones HTML/CSS que despliega en Vercel y entrega un link.
2. **Reglas supremas de diseño:** las ocho reglas que mandan sobre todo lo demás.
3. **Decisiones fundacionales:** stack, entrega, quién exporta, flujo, idioma y marca.
4. **Arquitectura del repositorio** y reglas de carpetas.
5. **Operaciones:** el flujo de construcción, los requisitos del `<head>` y el lint de la wiki.
6. **Portal web:** diseño, roles, cotizador y demás reglas del portal.
7. **Estándar de calidad** y **visor interactivo obligatorio.**
8. **Convenciones de la bitácora.**

LA FUENTE NORMATIVA ES EL BRANDBOOK

El Brandbook (`resources/campuslands-resources/Campuslands_Brandbook2023V2._compressed (1).pdf`, 27 páginas) se sigue estrictamente; `wiki/marca-campuslands.md` lo resume con citas de página. El orden de prioridad ante un conflicto es: 1) Brandbook, 2) reglas dadas por el usuario, 3) convenciones del agente.

Estructura del repositorio

```
presentations-designer/
  CLAUDE.md                       # reglas, operaciones y portal
  herramientas/                   # verificación automática
    verificar_deck.py             # mide fondos, logos, fuentes, geometría, contraste, PDF
    elegir_logo.py                # elige logo por contraste con el fondo
    igualar_logos.py              # factor --k-cliente: igual peso visual de ambos logos
    colores_marca.py              # colores del cliente para las franjas del portal
    verificar_cotizador.py        # compara el cotizador del portal con el del CRM
  presentaciones/                 # REPO GIT PROPIO: lo que despliega Vercel
    decks/
      assets/                     # fuentes, favicon, logos de Campuslands, assets/logos/
      _plantilla-campuslands/     # plantilla base: se copia para cada deck nuevo
      <slug-cliente>/             # index.html, styles.css, script.js, assets/
    web/                          # portal Next.js 16 + TypeScript
      src/data/decks.ts           # catálogo: todo deck nuevo se registra aquí
  recursos/                       # fuentes inmutables (solo lectura)
    logos-campuslands/ · fonts/ · capturas de ColBeef/ · plantilla-base.pdf
  wiki/                           # marca, sistema de diseño, verificación, despliegue, bitácora
```

DÓNDE VIVE EL AGENTE Y SUS RECURSOS COMPARTIDOS

El agente está en `agents/claude-agents/presentations-designer/`. Los logos, el isotipo, el favicon, el `Brief Fullservice.pdf` y el Brandbook de Campuslands ya no se copian dentro de cada agente: están en `resources/campuslands-resources/`, en la raíz del repositorio (desde la carpeta del agente: `../../../resources/campuslands-resources/`). Los logos de las empresas cliente (Alcaldía de Girón, Avicampo, ColBeef, FCV, Marval, Multinal, Unidrogas, etc.) están en `resources/companies-logos/<empresa>/`, y de ahí sale la copia «sin procesar» que se guarda en `presentaciones/decks/assets/logos/`.

Hay reglas de carpetas que conviene respetar:

- **`wiki/`, `herramientas/` y `presentaciones/` son del agente; `recursos/` es solo lectura.**
- **Cada presentación es autocontenida** en `presentaciones/decks/<slug>/`: lo propio en su `assets/`; fuentes y logos compartidos en `decks/assets/`, referenciados como `../assets/…`.

- **Web y decks no se mezclan.** La tecnología nueva (Next.js, TypeScript, base de datos) es solo del portal; los decks siguen siendo HTML/CSS/JS estático. `web/public/` es una copia generada de `decks/` (con `npm run sync:decks`) que nunca se edita ni se versiona.
- **El slug es también la URL pública** (`/<slug>`): corto y limpio, como `fcv` o `marval`. Si cambia, se usa `git mv` y se actualizan portal y wiki en el mismo cambio.

Dos repositorios Git

`presentaciones/` tiene su **propio historial y remote** (`Presentaciones`). El repositorio raíz (`FullService-Agents`) versiona la wiki, las herramientas y `CLAUDE.md`, además de las demás carpetas de agentes. Por eso:

- Los archivos del deck, el portal y los logos se commitean **dentro de** `presentaciones/` con `git add .`.
- Los cambios solo de `wiki/*.md`, `herramientas/` o `CLAUDE.md` se commitean en el repo raíz con `git add <archivos puntuales>`**, nunca** `git add .`, para no arrastrar cambios ajenos.

El flujo de construcción



1. Recibir alcances e información

El agente recibe métricas, dolores operativos, soluciones, módulos y datos de cotización. Aplica la regla de **no invención**: si falta información estratégica, se detiene y envía el cuestionario estándar (datos generales, línea base y dolores, arquitectura y alcance, ROI y sector, condiciones comerciales y logo del cliente).

EL TOTAL DE INVERSIÓN SALE DEL XLSX, NO DEL PDF

Cuando la fuente es un XLSX del agente de cotizaciones, el PDF de alcance generado puede mostrar un «Total General» que no es la cifra comercial correcta. Se verifica siempre la celda `Y122` del XLSX (equivalente a `Y1 ÷ 0.6`) o se recalcula, y si hay discrepancia se confirma con el usuario. El desglose por módulo es material interno: **no se muestra precio por módulo** en el deck salvo pedido explícito.

2. Plan por lámina (obligatorio)

Antes de escribir una línea de HTML, el agente presenta, **lámina por lámina** (máximo 10): el contenido con sus cifras y fuentes, el arquetipo de diseño, qué bloques van en tarjeta de acento, el factor `--k-cliente` del logo y la **razón de ubicación** de los bloques principales. **No construye hasta que el usuario confirme.**

3. Borrador completo

Elige un slug corto, **copia** `presentaciones/decks/_plantilla-campuslands/` (nunca un deck de otro cliente) y construye el borrador: contenido, logo del cliente recortado al contenido exacto, `<title>` con la razón social y favicon con el isotipo de Campuslands. La plantilla trae siete láminas de ejemplo que ya cumplen las reglas y pasan el verificador.

4. Verificación

Se corre `verificar_deck.py` hasta obtener **APROBADO** y luego se revisa cada PNG a la vista (ver la sección «Verificación»).

5. Ajustes y entrega

El usuario revisa el borrador completo; los ajustes se aplican sobre el HTML y **se vuelve a verificar tras cada uno**, porque un cambio de texto puede romper el reparto de una lámina. Al final se exporta el PDF, se registra el deck, se hace commit y push y se comparte el link.

Las ocho reglas supremas

#	Regla	Qué significa	Cómo se comprueba
1	Tema claro obligatorio	Fondo `#FFFFFF` en toda lámina y en el visor. Navy, violeta, dorado, celeste, verde y arena solo como tarjetas, franjas e íconos. **Sin bordes ni líneas:** todo se separa por sombras	El verificador rechaza cualquier otro fondo y los bordes sólidos finos

#	Regla	Qué significa	Cómo se comprueba
2	Co-branding con igual peso	«Campuslands × Cliente» solo en portada, cierre y barra del visor; Campuslands primero, «×» centrada, áreas de logo iguales	Medición de áreas (±6 %) y posición de la «×» (±2 px)
3	Logo a color	Sobre blanco, siempre el logo de color. Nunca se recolorea, rota, recorta ni redistribuye	`elegir_logo.py` y verificador
4	Poppins en todo	Regular 400, Medium 500, SemiBold 600 y Black 900. Roboto Mono, Playfair, Montserrat, Arial y otras, prohibidas	El verificador rechaza otras fuentes y fuentes de respaldo en el PDF
5	Nada plano	Alternar arquetipos entre láminas contiguas y usar el lenguaje gráfico de la marca; decoraciones `.fx` e íconos con significado	Revisión a la vista
6	Máximo 10 láminas	Solo se supera con pedido textual (`--max-slides N`)	El verificador falla con más de 10
7	Verificación visual rigurosa	Script y vista en cada build y cada ajuste	Protocolo en `wiki/verificacion.md`
8	Cada objeto con razón	Se debe poder explicar por qué cada bloque está ahí, a ese tamaño y con ese color	Se escribe en el plan y en la bitácora; lo que no tenga razón se quita

PROHIBIDO EL RÓTULO «CONFIDENCIAL»

Ni como badge, ni como óvalo, ni como texto: el verificador falla si aparece. Tampoco hay indicadores de página («NN / TT») ni filas de puntos; solo vale el numeral grande y difuminado `.ghost` .

El sistema de diseño

Paleta



Blanco
`#FFFFFF`
Único fondo de lámina y de la página del visor

Navy
`#000087`
Títulos y tarjetas de acento; nunca fondo

Violeta
`#5E3AE2`
Énfasis, números, el símbolo `</`, tarjetas de acento

Dorado
`#F4B422`
Íconos, barras y marco neón; no es color de texto sobre blanco

Celeste
`#2CAAFF`
Íconos y detalles

Verde
`#00AA80`
Éxito, chevrones e íconos

Arena
`#E4E4DB`
Paneles de apoyo; nunca fondo de lámina

Tinta
`#373435`
Color del texto de cuerpo sobre blanco

Pulsa un color para copiar su código HEX.

Los pares de texto permitidos están verificados con contraste WCAG (4,5 : 1 para texto normal, 3 : 1 para texto grande). Por ejemplo, sobre blanco se usan títulos navy, acentos violeta y cuerpo en tinta; sobre violeta el texto es blanco, porque el dorado sobre violeta no alcanza el contraste.

Tipografía

Elemento	Fuente	Tamaño orientativo
Título de portada o cierre	Poppins Black	38–44 pt
Título de sección	Poppins Black, con `</` y una palabra en Black Italic de acento	26–29 pt
Número grande o métrica	Poppins Black	36–78 pt
Etiqueta	Poppins Black, mayúsculas, violeta	7,5–9 pt

Elemento	Fuente	Tamaño orientativo
Cuerpo, listas, pies y migas	Poppins 400 (énfasis 500 o 600)	9–12 pt; **mínimo 9 px**

Nutmeg, la tipografía de destacados de la marca, es comercial y no está licenciada en el repositorio, por lo que cae a Poppins Black. El agente avisa al usuario.

La lámina

Cada lámina mide **1056 × 594 px** (11 × 6,1875 in a 96 dpi), el mismo tamaño que la página del PDF. La estructura es una raíz `.sl` con `data-bg="white"`, el contenido (el título abre la lámina) y un pie `.ft` con las migas `| campuslands | propuesta | cliente | sección |` abajo a la izquierda y el lema «Exploramos · Despegamos · Conquistamos» a la derecha. Las **láminas de contenido no llevan logos**; el co-branding vive solo en portada, cierre y barra del visor.

Arquetipos y lenguaje gráfico

La plantilla incluye siete arquetipos (portada, afirmación, métricas, proceso en escalera, antes y después, módulos y cierre). Otros válidos son línea de tiempo, diagrama de capas, tabla de planes, mapa, comparador de opciones y galería de logos. **No se repite el mismo arquetipo en láminas contiguas**: como el fondo nunca cambia, el ritmo sale de alternar arquetipos y de dónde se usan las tarjetas de acento.

Recurso	Para qué sirve
`</` **antes del título**	Prefijo de título de sección, en violeta sobre blanco
`{ etiqueta }`, `<= … =>`	Etiquetas y subtítulos
Decoraciones `.fx`	Anillos, puntos, rayas, cruces y chevrones, discretos y fuera del texto, en cada tarjeta importante
Íconos `.ic`	Un ícono con significado por elemento, de trazo uniforme
Marco neón dorado	**Un** dato o tarjeta clave por lámina; nunca en banners ni franjas
Fondo de portada y cierre (`.deco-cover` **)**	Anillos concéntricos, resplandor y dos hojas laterales, de baja saturación

UNA TRAMPA DEL VERIFICADOR

Una clase llamada `deco` hace que `verificar_deck.py` trate el elemento como decoración y **omita sus mediciones** (contraste, desborde, superposición). Las decoraciones de los marcos usan `.fx`, nunca `deco`.

Logos y co-branding

Dos logos a la misma altura no se ven del mismo tamaño si sus proporciones difieren: el logo vigente de Campuslands es 4,31 : 1, y uno de proporción 5,09 : 1 se vería cerca de un 18 % más ancho. Por eso la regla es **áreas de caja recortada iguales**, y `igualar_logos.py` calcula el factor `k`:

CÓDIGO

```
alto_cliente = alto_Campuslands × √(ratio_Campuslands / ratio_cliente)
```

Ese `k` se asigna a la variable CSS `--k-cliente`. Además, los logos se dibujan solo con `height` y `width:auto` (nunca ambos), para no deformarlos, y el del cliente debe estar **recortado a su contenido** (por canal alfa).

Lista de comprobación del logo del cliente: PNG o SVG con transparencia real; recortado al contenido exacto; copia **sin procesar** en `presentaciones/decks/assets/logos/` (para el portal) y copia recortada en el `assets/` del deck; probado sobre blanco; áreas iguales dentro del ±6 %; y confirmado mirando que ninguno de los dos domina. Si el logo no contrasta con el blanco, se pide otra versión: el fondo no se cambia.

El visor interactivo

Toda presentación web se entrega en un visor de **una lámina a la vez**, con la barra superior (co-branding en tamaño grande, mínimo 40 px de alto para el logo de Campuslands), la lámina al centro con sombra, y la barra inferior con anterior, avance, progreso, reproducción automática (5 segundos) y siguiente. Ambas barras miden lo mismo (`--bar-h`, 76 px) y se separan por sombras, nunca por líneas.

`script.js` resuelve varias cosas que no son obvias:

- **Escalado con `zoom`**, no con `transform:scale()`, para que el texto y las mediciones se mantengan coherentes.
- **Navegación** con flechas del teclado (con vuelta circular) y con deslizamiento táctil en celular.
- **`preloadAllFonts()`:** las `@font-face` de láminas ocultas no se descargan solas, y sin esto el PDF caería a fuentes de respaldo.

- **Sombras seguras para PDF:** al imprimir, las sombras con desenfoque se convierten en sombras vectoriales escalonadas, porque algunos visores de PDF de celular pintan rectángulos grises alrededor de las tarjetas.
- En celular en vertical aparece un aviso descartable para girar el equipo.

Herramientas

Herramienta	Comando	Qué hace
Verificador	`python3 herramientas/verificar_deck.py presentaciones/decks/<slug> --pdf <ruta>.pdf --png-dir <carpeta>`	Mide el deck y exporta PDF y un PNG por lámina. Sale con código 1 si hay errores
Igualar logos	`python3 herramientas/igualar_logos.py <logo-cliente.png>`	Imprime el factor `--k-cliente` y las medidas resultantes
Elegir logo	`python3 herramientas/elegir_logo.py "#FFFFFF"` (o `--tabla`)	Elige la versión del logo por contraste con un fondo
Colores de marca	`python3 herramientas/colores_marca.py <logo.png>`	Extrae 1–3 colores del cliente para el campo `colors` del portal
Cotizador	`python3 herramientas/verificar_cotizador.py`	Compara la lógica del cotizador del portal con la del CRM; debe decir «idéntica»

El verificador requiere Chromium y `pip install pymupdf`. En Windows el comando es `python` en lugar de `python3`.

Exportación a PDF

La exportación la corre el agente con Chrome en modo headless. Dos piezas hacen que el PDF salga bien:

En el CSS (`styles.css`), el tamaño de página debe ser el de una lámina y los márgenes cero:

CÓDIGO

```
@page { size: 11in 6.1875in; margin: 0; }
```

En el comando, se necesitan rutas absolutas, `--no-margins` y un presupuesto de tiempo virtual para que se carguen las fuentes:

CÓDIGO

```
chrome.exe --headless --disable-gpu --no-margins --virtual-time-budget=8000 --print-to-pdf="<ruta absoluta>.pdf" "file:///<ruta>/index.html"
```

El PDF debe tener tantas páginas como láminas, ninguna fuente de respaldo (Liberation, Georgia, Times, Arial) y ningún texto desbordado de una lámina a la siguiente. Si el panel del navegador no puede tomar capturas, la verificación visual se hace leyendo el PDF con PyMuPDF (`page.get_pixmap(dpi=150)` por lámina) en lugar de insistir con la captura.

Verificación

La verificación tiene dos mitades que no se sustituyen: la herramienta **mide**, la vista **juzga**.

Lo que mide `verificar_deck.py`

Grupo	Qué comprueba
Estructura	Número de láminas, `<title>` con la razón social y favicon con el isotipo
Fondos	Toda lámina y el visor en blanco; sin resplandor celeste ni borde dorado en banners
Logos	Peso visual igual, orden, «×» centrada, sin deformación, sin logos en láminas de contenido, sin rótulo «Confidencial»
Bordes y sombras	Sin bordes ni líneas en láminas ni en el visor; barras de igual altura
Tipografía	Solo Poppins; texto mínimo de 9 px

Grupo	Qué comprueba
Geometría	Nada fuera de la lámina, ni sobre el pie, ni recortado; sin tarjetas superpuestas; hueco vertical de 20 % como máximo
Contraste	WCAG 4,5 : 1 (3 : 1 en texto grande)
PDF	Páginas iguales a láminas y sin fuentes de respaldo

Los **avisos** (⚠) se resuelven o se justifican por escrito en la bitácora; los **errores** bloquean.

La revisión a la vista

Se abre cada PNG y se mira, en este orden: logos (¿del mismo peso visual?, ¿la «×» al medio?), jerarquía (¿qué se lee primero?), espacios (¿hay franjas muertas o elementos pegados?), color, tipografía, decoración e íconos (¿cada uno significa algo?), y dinamismo (¿la lámina cambia de arquetipo respecto a sus vecinas?). También se mira el visor en pantalla, porque tiene sus propios logos.

En el plan y luego en la bitácora se deja, por lámina, una línea por bloque relevante con su **razón de ubicación**; por ejemplo, «escalera de cuatro peldaños descendente: refleja avance en el tiempo».

Registro y publicación

Todo deck nuevo se registra en **dos lugares**, sin excepción, o el portal no lo muestra:

1. **`wiki/index.md`:** se lista o actualiza el deck.
2. **`presentaciones/web/src/data/decks.ts`:** un objeto nuevo al inicio del arreglo `DECKS`, con los campos `slug`, `client`, `company`, `title`, `desc`, `category` (`ia`, `software`, `demos` o `institucional`), `categoryLabel`, `slides`, `investment`, `path`, `pdf`, `logo`, **`colors`**, `date` (`YYYY-MM-DD`) y `keywords`.

Además se copia el logo del cliente **sin procesar** a `presentaciones/decks/assets/logos/` y se obtienen sus colores de marca con `colores_marca.py`. El campo `colors` es obligatorio: sin él, el portal falla al compilar.

EL CANDADO DEL AGENTE

Lo que registra el agente queda **bloqueado** en el portal: la fecha (`date`), el link de la presentación (`path` , `https://fullservicepresentations.vercel.app/<slug>/`) y el PDF (`pdf`) salen con candado y no se pueden editar desde «Editar proyecto». Por eso deben registrarse **correctos y completos desde el primer momento**: fecha real de entrega, PDF exportado y verificado, presentación ya desplegada. Si hay que cambiarlos, lo hace el agente editando `decks.ts` .

Requisitos del `<head>`

- **Favicon:** el isotipo de Campuslands sin texto, en `assets/favicon.png` .
- `<title>` **:** la razón social completa del cliente, con sufijo legal (por ejemplo, `Compumax Computer S.A.S.`), sin descripción de la propuesta. Si el usuario no la dio, el agente la pide; no la adivina.

Commit, push y link

Dentro de `presentaciones/` : `git add .` , `git commit -m "feat: <Nombre Comercial de la Empresa>"` y `git push` . Al terminar, el agente comparte el link `https://fullservicepresentations.vercel.app/<slug>/index.html` y **avisa de que la web tarda aproximadamente 2 minutos en reflejar el cambio**. Si no pudo confirmar que el despliegue ya está publicado, **lo dice**. Los permisos y el detalle del despliegue están en la sección siguiente.

El repositorio aparte y su publicación

`presentaciones/` no es un submódulo: el repositorio raíz solo la registra como un puntero (sin `.gitmodules`), así que un `git clone` del raíz la deja **vacía**. Se clona aparte, en esa ruta exacta; si no, el agente no encuentra `decks/_plantilla-campuslands/` y sus comandos de `git` resuelven al repositorio equivocado.

RUTA DENTRO DEL REPOSITORIO 1 `agents/claude-agents/presentations-designer`

PASO 1 · CLONAR EL REPOSITORIO 1

1 **`fullservice-agents`** Repositorio 1

Las instrucciones, la wiki y las herramientas del agente, junto con los demás agentes.

ASÍ QUEDA LA CARPETA

```
presentations-designer/
├─ … archivos del agente
└─ presentaciones/
   └─ (vacía)
```

La carpeta `presentaciones/` llega **vacía**: Git no trae el contenido del repositorio 2.

↓

PASO 2 · CLONAR EL REPOSITORIO 2 DENTRO DE ESA CARPETA

2 **`Presentaciones`** Repositorio 2 · aparte

Las presentaciones ya hechas y el portal web. Es lo que publica Vercel.

ASÍ QUEDA LA CARPETA

```
presentations-designer/
├─ … archivos del agente
└─ presentaciones/
   ├─ decks/
   └─ web/
```

Su contenido queda **dentro** de `presentaciones/`, con ese nombre exacto.

COMANDO DEL PASO 2

```
git clone https://github.com/GustavoAdolfoNavarroPacheco/Presentaciones presentaciones
```

Instalación correcta

TERMINAL

```
git clone https://github.com/GustavoAdolfoNavarroPacheco/fullservice-agents
cd fullservice-agents/agents/claude-agents/presentations-designer
git clone https://github.com/GustavoAdolfoNavarroPacheco/Presentaciones presentaciones
git -C presentaciones remote -v          # origin ... Presentaciones
git -C presentaciones status -sb         # ## main...origin/main
```

En Windows, activa antes `git config --global core.longpaths true` y clona en una ruta corta (`C:\dev`): el repositorio raíz tiene archivos con nombres muy largos. La versión para usuarios no técnicos, con un mensaje que se le da a la IA, está en Instalación.

Permisos

`Presentaciones` es **privado**: sin acceso falla incluso el clon («Repository not found»). El acceso lo concede Gustavo Navarro agregando a la persona como **colaborador con permiso de escritura**; el inicio de sesión lo hace cada quien en el navegador (Git Credential Manager o `gh auth login`). El agente no pide, guarda ni recibe contraseñas ni tokens.

Qué hace el agente

Estas reglas están en la sección «Repositorio aparte y publicación» de su `CLAUDE.md`:

Situación	Qué hace el agente
Al empezar, `presentaciones/` no existe, está vacía o no es su propio repositorio	Guía el clon en la ruta exacta. No crea otra carpeta ni copia decks a mano
Va a hacer el primer `git push` de la sesión	Recuerda que necesita permiso de escritura de Gustavo en `Presentaciones`
El `push` falla con `403` o *Permission denied*	Se detiene, explica que falta el permiso y a quién pedirlo. No usa *forks*, otros remotos ni otras credenciales
Termina un `push`	Informa que la web tarda aproximadamente 2 minutos, comparte el link y no afirma que está publicado hasta confirmarlo

Situación	Qué hace el agente
Le preguntan cómo funciona la web	Explica la zona privada (`/`), los links públicos (`/<slug>`) y el registro en `decks.ts`

Qué ocurre después del push

Vercel está conectado al repositorio `Presentaciones` (rama `main`). Cada push dispara un build: `npm run sync:decks` copia `decks/` a `web/public/`, Next.js compila el portal y el despliegue queda en línea. Todo el proceso tarda **aproximadamente 2 minutos**.



Animación ilustrativa de los pasos; el tiempo real es de aproximadamente 2 min.

- Durante ese tiempo el link puede dar 404 o mostrar la versión anterior; la caché del navegador se salta con `Ctrl + F5`.
- Si pasan más de cinco minutos, se revisa el último despliegue en el panel de Vercel: las causas habituales son un `decks.ts` con un campo obligatorio ausente (como `colors`) o un error de TypeScript.

El portal web

El portal (`presentaciones/web`) es una aplicación **Next.js 16 con TypeScript** que muestra el catálogo de presentaciones. No es una lámina: tiene su propio diseño, documentado en `CLAUDE.md`. Lo esencial para quien mantiene el agente:

Tema	Cómo funciona
Acceso	Solo `/` (el catálogo) es privado; las presentaciones `/<slug>/` y sus PDF son **públicas**, para compartirlas con clientes
Datos	Usuarios, proyectos creados desde el portal, cotizaciones y PDF subidos viven en **Neon** (Postgres); la sesión va en cookie `httpOnly` firmada

Tema	Cómo funciona
Roles	**Propietario** (todo), **administrador** (gestiona proyectos y usuarios), **usuario** (crea y edita proyectos, cotiza) y **visitante** (solo ve). El rol se lee de la base en cada petición
Cotizador	Copia fiel de la fase de cotización del CRM. La lógica no se toca; si el CRM cambia, se vuelve a copiar y se corre `verificar_cotizador.py` y `npm test`
Proyectos del agente	Los registrados en `decks.ts` no se pueden eliminar desde el portal y tienen candado en fecha, link y PDF
Variables de entorno	`DATABASE_URL` (Neon, conexión *pooled*) y `AUTH_SECRET` (32 caracteres aleatorios o más); nunca se versionan

Puesta en marcha local: `npm install`, `cp .env.example .env.local`, `npm run db:migrate`, `npm run user:create -- correo "Nombre"` y `npm run dev` (puerto 3000), todo dentro de `presentaciones/web`. En Vercel: framework Next.js, *Root Directory* `web`, «Include source files outside of Root Directory» activado y Node 22.18 o superior.

Comandos «!»

Los comandos son órdenes cortas del usuario (`!optimize`, `!notime`…) que el agente debe ejecutar de forma obligatoria, siempre dentro de sus reglas duras. Hay un solo catálogo para todos los agentes y de él salen las instrucciones de cada uno, las bitácoras y las tablas de las guías.

Dónde vive cada cosa

Qué	Dónde
Catálogo, regla común y qué comandos tiene cada agente	`resources/comandos/comandos.py` (la fuente única; no se edita a mano nada de lo que genera)
Catálogo legible, matriz por agente y decisiones	`resources/comandos/catalogo.md`
Instrucciones del agente	sección «Comandos (`!`)» de `CLAUDE.md`, entre las marcas `<!-- comandos:inicio -->` y `<!-- comandos:fin -->`

Qué	Dónde
Tabla de esta guía	se genera del mismo catálogo al compilar las guías

La regla común

1. Un comando es una palabra precedida de `!` al inicio de un mensaje o de una línea; el texto que sigue es su objeto.
2. Es obligatorio: el agente no pide permiso ni lo reinterpreta. Solo lo limita una **regla dura** (veracidad, marca, confidencialidad, puertas de aprobación y de verificación); entonces hace lo que la regla permite y dice qué no pudo.
3. Se combinan. Si dos modos chocan, el orden es: reglas duras > `!strict-mode` y `!no-assumptions` > `!think-deep`, `!critical-mode` y `!double-check` > `!notime`.
4. Rige la tarea en curso, no las siguientes.
5. La primera línea de la respuesta nombra los comandos recibidos.
6. Un comando desconocido no se adivina: se propone el más parecido y se pregunta. `!help` lista los disponibles.
7. Si un comando cambió un entregable o la wiki, se anota en `wiki/log.md`.

Cómo añadir o cambiar un comando

1. Edita `CMDS` (definición), `AGENTES` (a quién aplica y su nota) o `ALIAS` en `resources/comandos/comandos.py`.
2. Corre `python -X utf8 resources/comandos/comandos.py`: reescribe la sección de cada agente, el catálogo y, la primera vez, añade la entrada a su bitácora.
3. Recompila las guías (`python guides/_src/build.py`).
4. `python resources/comandos/comandos.py --verificar` falla si algo quedó desfasado.

Comandos de este agente

General

Comando	Qué hace aquí
`!analyze`	Analiza la solicitud, el problema, los documentos o la situación **antes de actuar**: qué se pide, qué hay, qué falta y qué riesgos ve. Entrega hallazgos; no ejecuta ni modifica nada. **Aquí:** Analiza los alcances y datos del cliente: qué mensaje, qué cifras, qué falta; no diseña todavía.
`!research`	Investiga a fondo el tema, cruza fuentes y entrega lo relevante con **cada dato citado a su fuente**. Sin fuente verificable no es un dato: se marca `[PENDIENTE]`.

Comando	Qué hace aquí
`!plan`	Entrega un plan de trabajo estructurado (pasos, orden, insumos, entregables, supuestos y puntos de decisión) y **se detiene a esperar aprobación** antes de ejecutarlo. **Aquí:** Es el **plan lámina por lámina** (máx. 10): pregunta que responde cada una, contenido y diseño. Espera aprobación antes de construir.
`!execute`	Ejecuta ya lo solicitado o el siguiente paso aprobado del flujo, sin volver a proponer ni a preguntar lo ya decidido. No se salta las puertas de aprobación ni de verificación. **Aquí:** Construye el borrador del plan **ya aprobado**. Sin plan aprobado no construye: lo pide.
`!review`	Revisa el trabajo indicado y entrega oportunidades de mejora priorizadas (qué cambiaría y por qué). No lo modifica salvo que se le pida.
`!audit`	Auditoría integral de punta a punta: contradicciones entre documentos, cifras que no cuadran, reglas incumplidas y datos sin fuente. Entrega hallazgos con evidencia y severidad. **Aquí:** Auditoría de la presentación completa: marca, cifras, logos, láminas fuera de las reglas, coherencia entre láminas.
`!validate`	Comprueba, requisito por requisito, que el resultado cumple lo establecido (reglas duras, insumos, formato). Veredicto «cumple / no cumple» por ítem, con evidencia. **Aquí:** Corre `verificar_deck.py` y la revisión a la vista de cada lámina contra las reglas supremas del Brandbook.
`!optimize`	Optimiza lo que se esté desarrollando o ya existe (proceso, solución o resultado): mejora claridad, calidad, consistencia, peso o rendimiento **sin cambiar su alcance ni sus datos**. **Aquí:** Pule la presentación en curso (ritmo, texto, jerarquía, peso) sin cambiar el plan aprobado ni las reglas de marca.
`!compare`	Compara alternativas con criterios explícitos y recomienda la más conveniente, con el porqué.
`!summarize`	Resumen claro y estructurado (lo esencial, decisiones y pendientes) del trabajo, de un documento o de la conversación.
`!document`	Documenta lo hecho: actualiza la wiki y la bitácora del agente, o genera la documentación técnica, funcional o ejecutiva que se pida.
`!prioritize`	Clasifica actividades, hallazgos o elementos por impacto, importancia y urgencia, y entrega el orden recomendado.
`!help`	Lista los comandos disponibles en este agente, agrupados, con su función.

Modos

Comando	Qué hace aquí
`!notime`	**Modo prisa:** apresura el desarrollo optimizándolo: agrupa preguntas en un solo mensaje, paraleliza, reutiliza lo ya validado, omite explicaciones y pasos opcionales y responde solo con lo esencial. **Nunca** se salta verificaciones obligatorias, puertas de aprobación ni la veracidad: acelera el camino, no recorta la calidad. **Aquí:** Agrupa preguntas, construye por bloques y reduce explicaciones. **No** se salta el plan aprobado ni la verificación automática y a la vista.
`!think-deep`	Análisis más profundo antes de responder: considera alternativas, casos límite y consecuencias, y muestra solo el razonamiento que importa.
`!critical-mode`	Modo crítico: cuestiona decisiones, supuestos y su propio trabajo; busca errores y contradicciones antes de entregar y los dice.
`!no-assumptions`	No supone: ante un dato faltante o ambiguo pregunta (todas las dudas juntas) en lugar de rellenar o inferir.
`!double-check`	Segunda verificación independiente de su propio resultado (releer la fuente, recalcular, comparar) y corrección de lo que encuentre; reporta qué verificó.
`!strict-mode`	Sigue al pie de la letra los requisitos e instrucciones dados: sin iniciativas, adiciones ni interpretaciones creativas.

Técnico y seguridad

Comando	Qué hace aquí
`!debug`	Identifica la causa de un error de software (script, conversión, verificación) y lo corrige; vuelve a correr la verificación para comprobarlo. **Aquí:** Corrige el fallo encontrado en scripts, plantilla o visor y repite la verificación.
`!troubleshoot`	Investiga un problema técnico y determina su **causa raíz**, con evidencia. Diagnostica; no cambia nada. **Aquí:** Averigua por qué falla una verificación, la exportación a PDF o la publicación.
`!security-check`	Revisa seguridad y confidencialidad: datos de otros clientes mezclados, credenciales, datos personales, información que no debe salir del proyecto. **Aquí:** Revisa que no se mezclen datos de otros clientes, que el portal limite el acceso y que las presentaciones confidenciales no queden públicas.

Comando	Qué hace aquí
`!implement`	Desarrolla la funcionalidad o solución solicitada (código, plantilla, script). **Aquí:** Desarrolla la herramienta, plantilla o mejora del portal pedida, verificada con las herramientas del repositorio.
`!refactor`	Mejora la estructura del código sin alterar su comportamiento esperado; lo demuestra con la verificación antes y después. **Aquí:** Reordena código del visor, scripts o plantilla sin cambiar lo que producen; lo demuestra con `verificar_deck.py`.
`!code-review`	Evalúa calidad, organización y mantenibilidad del código; entrega observaciones priorizadas. **Aquí:** Revisa el código de la plantilla, el visor, los scripts y el portal (`presentaciones/web`).
`!performance`	Analiza y optimiza el rendimiento: tiempos de carga, peso de archivos, fluidez. **Aquí:** Mide y reduce el peso y el tiempo de carga de la presentación (imágenes, fuentes, scripts) y del portal.
`!deploy-check`	Verifica que el proyecto esté listo para publicarse: requisitos, exportación y enlaces comprobados. **Aquí:** Comprueba antes de publicar: requisitos de `<head>` , visor interactivo, PDF exportado, registro en el portal y enlace que abre.
`!rollback`	Gestiona la reversión controlada de cambios a un estado anterior conocido, sin perder lo nuevo (queda respaldado). **Aquí:** Vuelve la presentación a una versión anterior con git, conservando la nueva en una rama o copia.

UX/UI y diseño

Comando	Qué hace aquí
`!ui-review`	Audita visualmente una interfaz y detecta problemas de diseño. **Aquí:** Revisa a la vista láminas y visor: jerarquía, alineación, espaciado, legibilidad.
`!ux-review`	Evalúa la experiencia de usuario y los flujos de navegación. **Aquí:** Evalúa el recorrido de quien ve la presentación: navegación del visor, ritmo, claridad del mensaje.

Comando	Qué hace aquí
`!redesign`	Propone y desarrolla un rediseño visual, dentro de las reglas de marca. **Aquí:** Rediseña una lámina o sección **dentro del Brandbook v3** (fondo blanco, Poppins, colores Campuslands); nunca cambia las reglas de marca.
`!design-system`	Construye, documenta o estandariza el sistema de diseño. **Aquí:** Documenta o ajusta el sistema de diseño v3 en `wiki/sistema-diseno.md` y la plantilla base.
`!visual-audit`	Examina proporciones, distribución, tipografía y consistencia visual del entregable, a la vista y página por página. **Aquí:** Examina cada lámina a la vista: proporciones, distribución, tipografía, logos (mismo peso visual) y consistencia.
`!responsive-check`	Evalúa cómo se adapta la interfaz a distintos tamaños y dispositivos. **Aquí:** Comprueba el visor y el portal en distintos anchos de pantalla.
`!accessibility`	Evalúa y mejora la accesibilidad: contraste, tamaños, teclado, textos alternativos. **Aquí:** Revisa contraste, tamaños de texto, navegación por teclado y textos alternativos del visor y las láminas.
`!microinteractions`	Diseña o implementa microinteracciones y animaciones sutiles, sin romper las reglas de marca. **Aquí:** Propone o implementa transiciones y animaciones del visor que respeten las reglas de marca y no distraigan.
`!consistency-check`	Detecta inconsistencias entre secciones, componentes, cifras, nombres y estilos del mismo entregable. **Aquí:** Revisa que nombres, cifras, logos y estilos sean idénticos en todas las láminas y en el PDF.
`!componentize`	Identifica elementos repetidos y los convierte en componentes reutilizables. **Aquí:** Convierte bloques repetidos de láminas en componentes de la plantilla base reutilizables.

Gestión de proyecto

Comando	Qué hace aquí
`!breakdown`	Divide un proyecto o solicitud compleja en actividades específicas, cada una con su entregable y su orden.

Comando	Qué hace aquí
`!track-progress`	Informa el progreso y los pendientes (hecho, en curso, bloqueado, `[PENDIENTE]`) a partir de la bitácora y del trabajo actual.
`!requirements`	Extrae y estructura requerimientos funcionales y no funcionales de las fuentes, cada uno con su referencia de origen. **Aquí:** Extrae de los alcances los datos que la presentación necesita (6 bloques) y señala lo que falta.
`!generate-report`	Genera un informe ejecutivo (decisiones, resultado, pendientes) o técnico (método, evidencias, verificaciones) del trabajo.
`!final-delivery`	Prepara y valida la entrega final: corre todas las verificaciones obligatorias, completa el checklist de cierre (wiki y bitácora) y entrega solo si todo pasa. **Aquí:** Verificación completa, PDF, registro en el portal y publicación con link; solo entrega si la publicación está confirmada.
`!agent-handoff`	Prepara el traspaso al siguiente agente del flujo comercial: resumen, entregables con su ruta, decisiones tomadas y pendientes. **Aquí:** Prepara el traspaso a **Desarrollador de Cotizaciones** (`quote-developer`) con la presentación aprobada, sus datos y decisiones.

IA avanzada

Comando	Qué hace aquí
`!second-opinion`	Un subagente independiente evalúa el resultado **sin ver tu razonamiento**; se contrasta con el tuyo y se explican las diferencias.
`!cross-review`	Otro especialista (subagente con otro rol) revisa el resultado y devuelve observaciones; se aplican las válidas y se informa de las descartadas.

Multiagente

Comando	Qué hace aquí
`!orchestrate`	Reparte partes **independientes** del trabajo entre varios subagentes en paralelo y las integra en un solo resultado verificado. **Aquí:** Reparte láminas o verificaciones independientes entre subagentes; la integración y la verificación final las hace el agente principal.

Comando	Qué hace aquí
`!team-review`	Varios revisores con roles distintos evalúan por separado el mismo resultado y se consolida su evaluación. **Aquí:** Revisores con rol de **diseño/marca**, **contenido y veracidad** y **técnico (HTML/visor)**, cada uno por separado.
`!consensus`	Consolida las evaluaciones disponibles (de subagentes o de revisiones previas) en una sola recomendación, con las discrepancias a la vista.

Mantenimiento

La bitácora `wiki/log.md` es append-only. Cada entrada empieza con `## [YYYY-MM-DD] <tipo> | <descripción>` y el tipo es `setup`, `build`, `ajuste`, `deploy`, `lint` o `marca`. En el lint periódico se revisan contradicciones, información desactualizada, páginas huérfanas y convenciones que ya no se usan.

Quiero...	Qué tocar
Incorporar una regla de marca nueva	`CLAUDE.md` y `wiki/marca-campuslands.md`, en el mismo cambio; si es medible, también `verificar_deck.py`
Cambiar la plantilla base	`presentaciones/decks/_plantilla-campuslands/` (y comprobar que sigue en APROBADO)
Cambiar los componentes visuales	El CSS de la plantilla y `wiki/sistema-diseno.md`
Agregar una comprobación	`herramientas/verificar_deck.py` y la tabla de `wiki/verificacion.md`
Registrar un deck nuevo	`wiki/index.md`, `decks.ts`, logo en `decks/assets/logos/` y `colors`
Cambiar el slug de un deck	`git mv` de la carpeta y actualizar `decks.ts` y la wiki en el mismo cambio

Problemas frecuentes

Síntoma	Causa probable	Qué hacer
El verificador marca logos de distinto peso	Se igualó la altura, no el área	Correr `igualar_logos.py` con el logo recortado y aplicar `--k-cliente`
El PDF usa fuentes de respaldo	Faltan `--virtual-time-budget` o `preloadAllFonts()`	Reexportar con el presupuesto de tiempo y confirmar `script.js`
Una decoración no se mide	La clase se llama `deco`	Renombrar a `.fx` y marcarla con `aria-hidden`
El portal falla al compilar	El deck nuevo no tiene `colors`	Obtener los colores con `colores_marca.py` y registrarlos
El total de inversión no coincide con el XLSX	Se tomó el total del PDF de alcance	Usar la celda `Y122` del XLSX o `Y1 ÷ 0.6`
El logo del cliente no contrasta con el blanco	Es una versión clara o blanca	Pedir la versión a color u oscura; no cambiar el fondo
Aparece un aviso de hueco vertical	Composición con mucho aire	Redistribuir o justificar el aviso por escrito en la bitácora

Cosas que no están alineadas

Al leer la wiki conviene saber que algunas páginas conservan reglas del sistema anterior y contradicen `CLAUDE.md`, que es el que manda:

- **`wiki/sistema-diseno.md`** (sección 2) dice que las tarjetas blancas se separan con un borde fino, mientras que la regla vigente, y la propia sección 3 del mismo archivo, prohíben bordes en favor de sombras.
- **`wiki/plantilla-base.md`** menciona un indicador `NN / TT` a la derecha del pie, que la regla vigente prohíbe.
- **`wiki/estandar-v3-laminas.md`** describe el rediseño de un deck concreto con Playfair, Montserrat y fondo gris, tipografías y fondo hoy prohibidos: es historia, no norma.
- **`wiki/despliegue.md`** conserva rutas de Windows antiguas y un CSS de impresión con fondo oscuro de un sistema previo.

- `CLAUDE.md` (paso 3 del flujo) y `wiki/flujo-trabajo.md` aún hablan de fondo «arena», cuando la regla vigente es blanco.

Referencia rápida de archivos

Archivo	Para qué lo abres
`CLAUDE.md`	Reglas supremas, flujo de construcción, portal
`wiki/marca-campuslands.md`	Normativa de marca con citas del Brandbook
`wiki/sistema-diseno.md`	Tokens, componentes, arquetipos y anti-patrones
`wiki/verificacion.md`	Protocolo de verificación y falsos positivos conocidos
`wiki/plantilla-base.md`	Biblioteca narrativa de hasta 10 láminas
`wiki/despliegue.md`	Exportación a PDF, repos Git y despliegue
`wiki/temas-por-cliente.md`	Qué aporta el cliente (logo y contenido) y qué no
`presentaciones/README.md`	Puesta en marcha del portal