DMarlon Studio

Manual de usuario

Manual de DMarlon Studio

Guía completa del IDE visual para crear aplicaciones de Python

Versión 1.0.18 · 2026-11-05

dmarlonstudio.com

Contenido

1 **Introducción e instalación**
Qué es DMarlon Studio · Requisitos · Instalación · Ejemplos incluidos · Dónde se guarda cada cosa

2 **Primeros pasos**
La ventana principal · Tema e idioma · Su primera ventana en cinco pasos

3 **El Diseñador**
Proyectos · La pantalla del Diseñador · Componentes · Mover, alinear y editar · Posición de la ventana · Eventos y código · Python generado y exportación · Ejecutar, detener y ver la salida · Conexiones de datos

4 **IA en el Diseñador**
Chat con Claude Code · Pedir a la IA · Presupuesto de IA · GitHub

5 **El Editor**
Abrir una carpeta · Explorador y pestañas · Buscar y reemplazar · Terminal · Control de código (git) · Claude implementa · Chat con la IA

6 **Bases de datos**
Conexiones · Tablas y consultas

7 **Cliente de API**
Peticiones · Colecciones, variables e historial

8 **FTP**
Servidores y archivos

9 **Diagramas**
Diagramas de flujo y de base de datos

10 **Ajustes**
Conectar con Claude (MCP) · Inteligencia artificial · Idioma, apariencia y versión

11 **Atajos de teclado**
Diseñador · Editor

12 **Solución de problemas**
No encuentra Python o le falta tkinter · Falta Flask · La app no abre (sin firmar) · AttributeError: 'Form1' object has no attribute 'root' · No se encontró Claude Code

13 **Hoja de ruta**
Destinos · Proyectos más completos

Capítulo 1

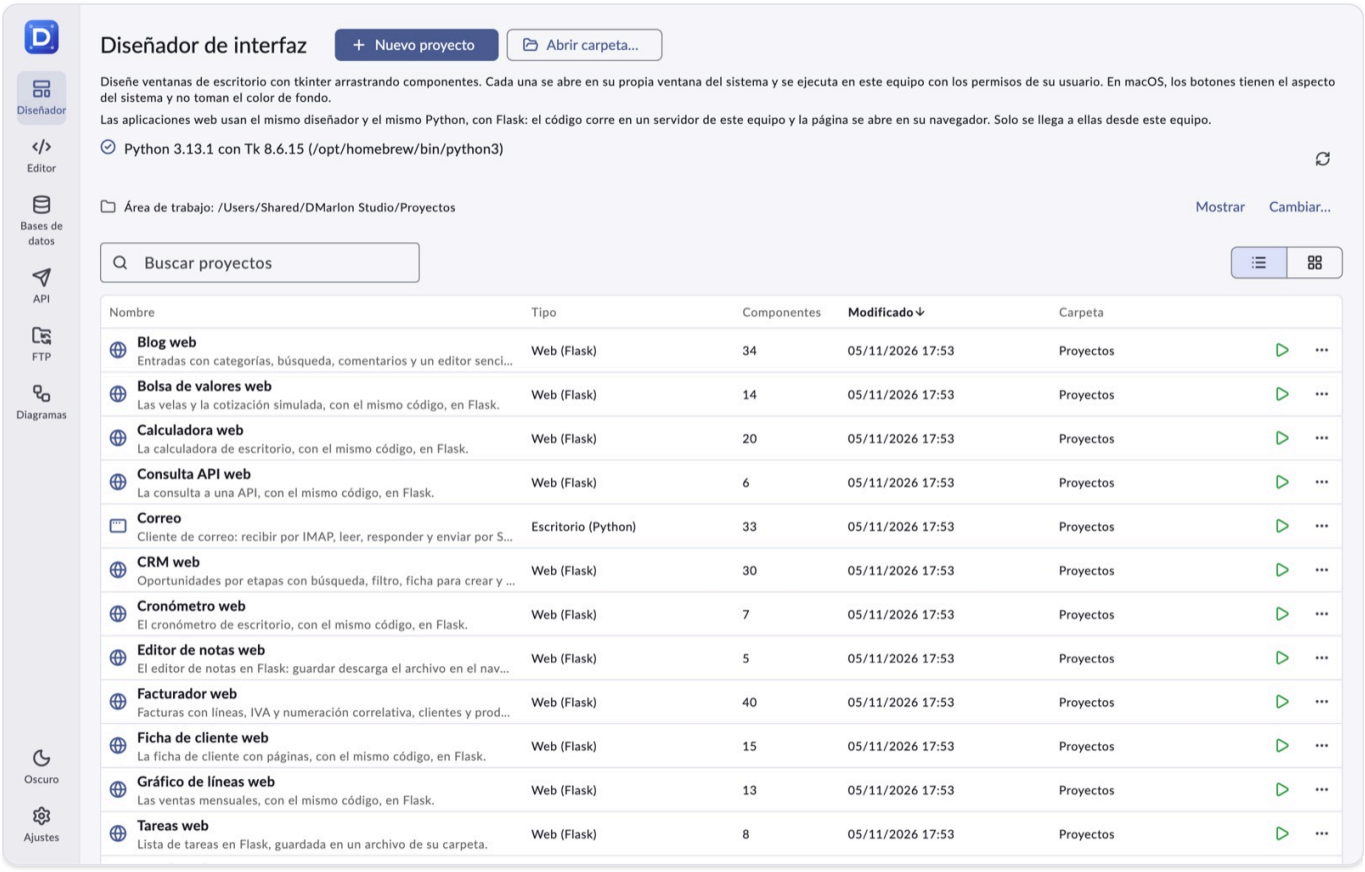
Introducción e instalación

Qué es DMarlon Studio, qué necesita y dónde guarda cada cosa.

Qué es DMarlon Studio

DMarlon Studio es un IDE visual de escritorio. Las ventanas se diseñan arrastrando componentes a un formulario, sus propiedades se ajustan en un inspector y el código de cada evento se escribe en Python. Con **F9** la ventana se ejecuta en su equipo.

Hoy genera dos tipos de aplicación con el mismo diseñador: ventanas de escritorio con **tkinter** y aplicaciones web con **Flask**. Alrededor del diseñador hay un editor de código al estilo de VS Code con git y terminal, un gestor de bases de datos, un cliente de API, un cliente FTP, un editor de diagramas y asistentes de inteligencia artificial (Claude por defecto).



El Diseñador: paleta arriba, árbol e inspector a la izquierda y el formulario en el centro.

Requisitos

Para…	Necesita
Usar el estudio	macOS 10.15 o posterior, o Windows 10/11 de 64 bits.
Ejecutar ventanas de escritorio	Python 3 con tkinter. En macOS: `brew install python-tk` o el instalador de python.org. En Windows: el instalador de python.org con la opción **tcl/tk and IDLE** marcada.
Ejecutar aplicaciones web	Flask en ese mismo Python: `python3 -m pip install flask` (en Windows, `py -3 -m pip install flask`).
Control de versiones en el Editor	git instalado en el equipo.
Chat con Claude Code en el Diseñador	Claude Code instalado y con la sesión iniciada (opcional).
Asistentes de IA	Una clave de API del proveedor elegido en **Ajustes** (opcional).

> El estudio busca Python por su cuenta: en el PATH, en las carpetas de Homebrew y en la instalación de python.org, y se queda con el que trae la versión de Tk más reciente. El resultado se ve en la lista de proyectos del Diseñador.

Instalación

Descargue el zip de su sistema desde la página de descargas de dmarlonstudio.com.

1. **macOS**: descomprima `DMarlonStudio-macOS.zip` y mueva **DMarlon Studio** a la carpeta Aplicaciones. La primera vez, abra la app con clic derecho → **Abrir**: no está firmada.
2. **Windows**: descomprima `DMarlonStudio-Windows.zip` completo donde quiera (no en Archivos de programa) y ejecute `DMarlonStudio.exe`. Si aparece «Windows protegió su PC», pulse **Más información** → **Ejecutar de todas formas**.

Ejemplos incluidos

Todos guardan sus datos en la carpeta del proyecto y funcionan sin configurar nada. Los marcados con (web) también existen como aplicación de Flask.

Ejemplo	Qué enseña
Calculadora (web)	Botones, eventos y estado
Conversor de unidades (web)	Listas desplegables y cálculo al escribir
Tareas (web)	Lista con altas, bajas y guardado
Cronómetro (web)	Temporizador, menú, barra de progreso y de estado
Ficha de cliente (web)	Control de páginas, listas y validación
Editor de notas (web)	Cinta de opciones y diálogos Abrir y Guardar
Consulta API (web)	Integración API en segundo plano y tabla de datos
Bloc de notas	Menú clásico, abrir, guardar, deshacer y barra de estado con línea y columna
Agenda	Contactos y citas en SQLite, búsqueda y páginas
Conversor de moneda	Tipos de cambio reales (BCE) con Integración API y reserva sin red
La serpiente	Juego en un lienzo con temporizador y récord guardado
Reproductor de música	Lista de reproducción, volumen y progreso (mejor con `pygame`)
Ventas mensuales (web)	Gráfico de líneas, área o barras y modo en vivo
Bolsa de valores (web)	Velas, media móvil y volumen con datos simulados en vivo
Ventanas y diálogos	Mensajes, preguntas, pedir datos, otra ventana que devuelve un resultado, un diálogo modal y ventanas no modales
Asistente de textos (web)	Claude para resumir, traducir, clasificar y extraer datos, y un chat con memoria
Informes (web)	Una tabla de ventas exportada a PDF y a Excel
Visor 3D	Figuras en 3D dibujadas en un lienzo: giro con el ratón, animación, luz y modo alambre
Correo	Recibir por IMAP y enviar por SMTP con adjuntos, sin congelar la ventana
CRM web, **Facturador web**, **Blog web**, **Tienda web**	Aplicaciones web completas con SQLite
Contactos de Odoo / **ERPSqlite** (web)	Conectarse a un Odoo por XML-RPC

Dónde se guarda cada cosa

Qué	Dónde
Proyectos del Diseñador	Una carpeta por proyecto en `~/Documents/DMarlon Studio/Proyectos` (se puede cambiar en el Diseñador). Se pueden versionar con git.
Ajustes, conexiones, peticiones de API, historial y diagramas	`studio.sqlite`, en la carpeta de datos de la aplicación: `~/Library/Application Support/com.dmarlonstudio.studio` en macOS y `%APPDATA%\DMarlon Studio\DMarlon Studio` en Windows.
Proyectos borrados	La carpeta `.papelera` dentro del área de trabajo: nada se borra del todo.

Cada carpeta de proyecto contiene `project.dmsproj` (el diseño en JSON), el código de la ventana (`Form1.py`, con el nombre del formulario), un `.gitignore` y, al ejecutar, `build/` con el Python generado.

Capítulo 2

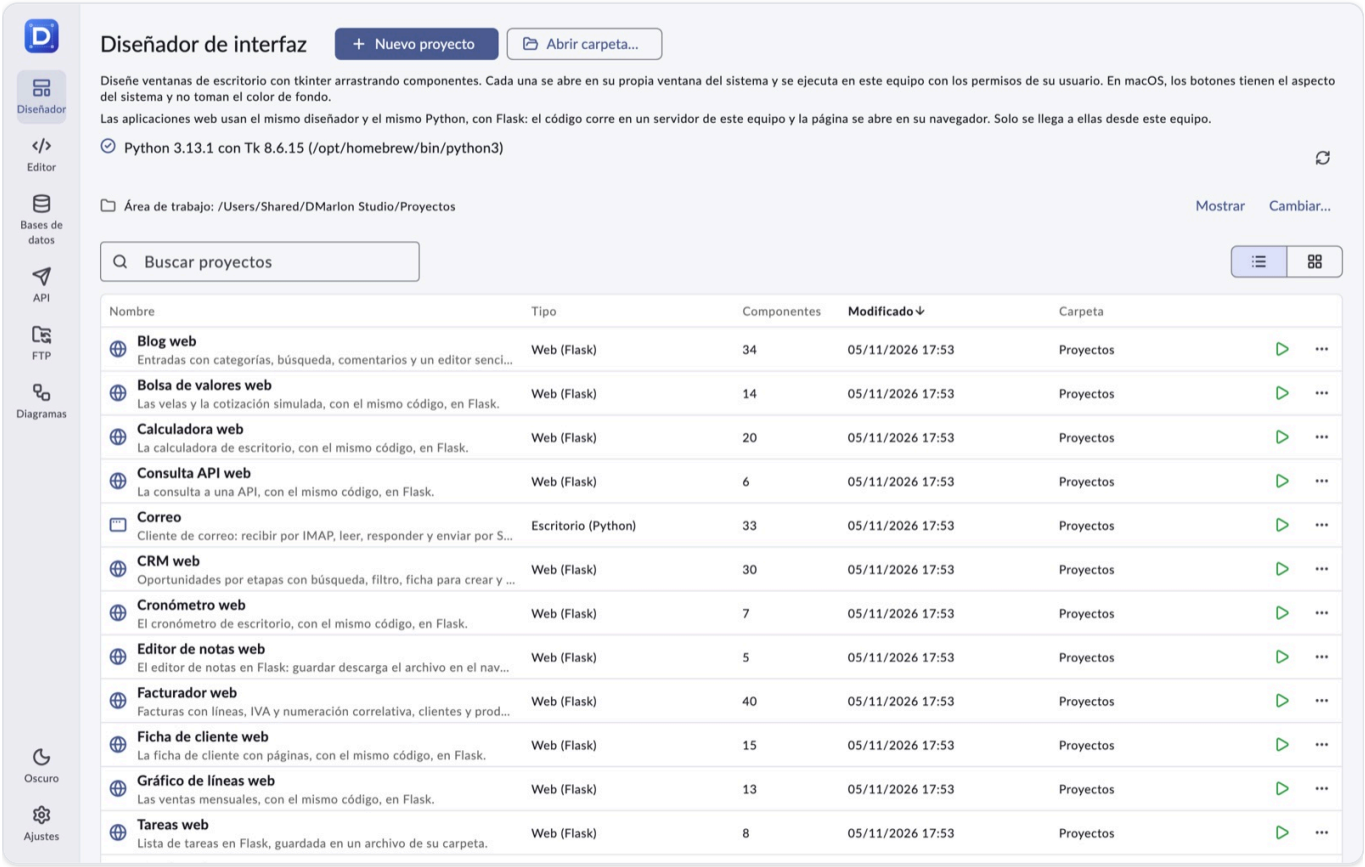
Primeros pasos

La ventana principal, los módulos, el tema y el idioma.

La ventana principal

A la izquierda está la barra de módulos: **Diseñador**, **Editor**, **Bases de datos**, **API**, **FTP**, **Diagramas** y, abajo, el botón de modo **Oscuro**/**Claro** y **Ajustes**.

Cada módulo se crea la primera vez que se abre y conserva su estado: puede pasar del Diseñador al Editor y volver sin perder lo que no haya guardado.



La lista de proyectos, con los ejemplos que trae el estudio.

Tema e idioma

- El botón **Oscuro**/**Claro** de la barra cambia el tema al momento. En **Ajustes → Apariencia** también puede elegir **Sistema** para seguir al del equipo.
- **Ajustes → Idioma** cambia entre **Español** e **English** sin cerrar nada. La primera vez se usa el idioma del sistema.

Su primera ventana en cinco pasos

1. En **Diseñador**, pulse **Nuevo proyecto**, escriba un nombre y elija **Escritorio (Python)**.
2. Arrastre una **Etiqueta**, un **Campo de texto** y un **Botón** desde la paleta al formulario.
3. Seleccione el botón y cambie su **Texto** en el inspector, por ejemplo a «Saludar».
4. Haga doble clic en el botón: se crea el método `Button1_click` y se abre el código. Escriba `self.Label1.config(text="Hola, " + self.Entry1.get())`.
5. Pulse **F9**. La ventana se abre en su propia ventana del sistema.

Capítulo 3

El Diseñador

Proyectos, componentes, inspector, eventos, código generado y ejecución.

Proyectos

La lista de proyectos muestra los del área de trabajo y los abiertos desde otras carpetas. Arriba se ve el Python encontrado (versión de Python y de Tk) y el botón para **Volver a buscar Python**.

Se ve como una **lista** con columnas (**Nombre**, **Tipo**, **Componentes**, **Modificado** y **Carpeta**) que se ordenan con un clic en su título, o en **Tarjetas**. Tiene **Buscar proyectos** y va por páginas (**Por página**: 10, 15, 25 o 50). Doble clic abre un proyecto. El botón derecho sobre un proyecto ofrece **Abrir**, **Ejecutar**, **Mostrar la carpeta**, **GitHub…**, **Presupuesto de IA**, **Duplicar** y **Eliminar**; sobre el fondo, **Agregar nuevo** › **Proyecto de escritorio (Python)**, **Proyecto web (Flask)** o **Abrir carpeta…**.

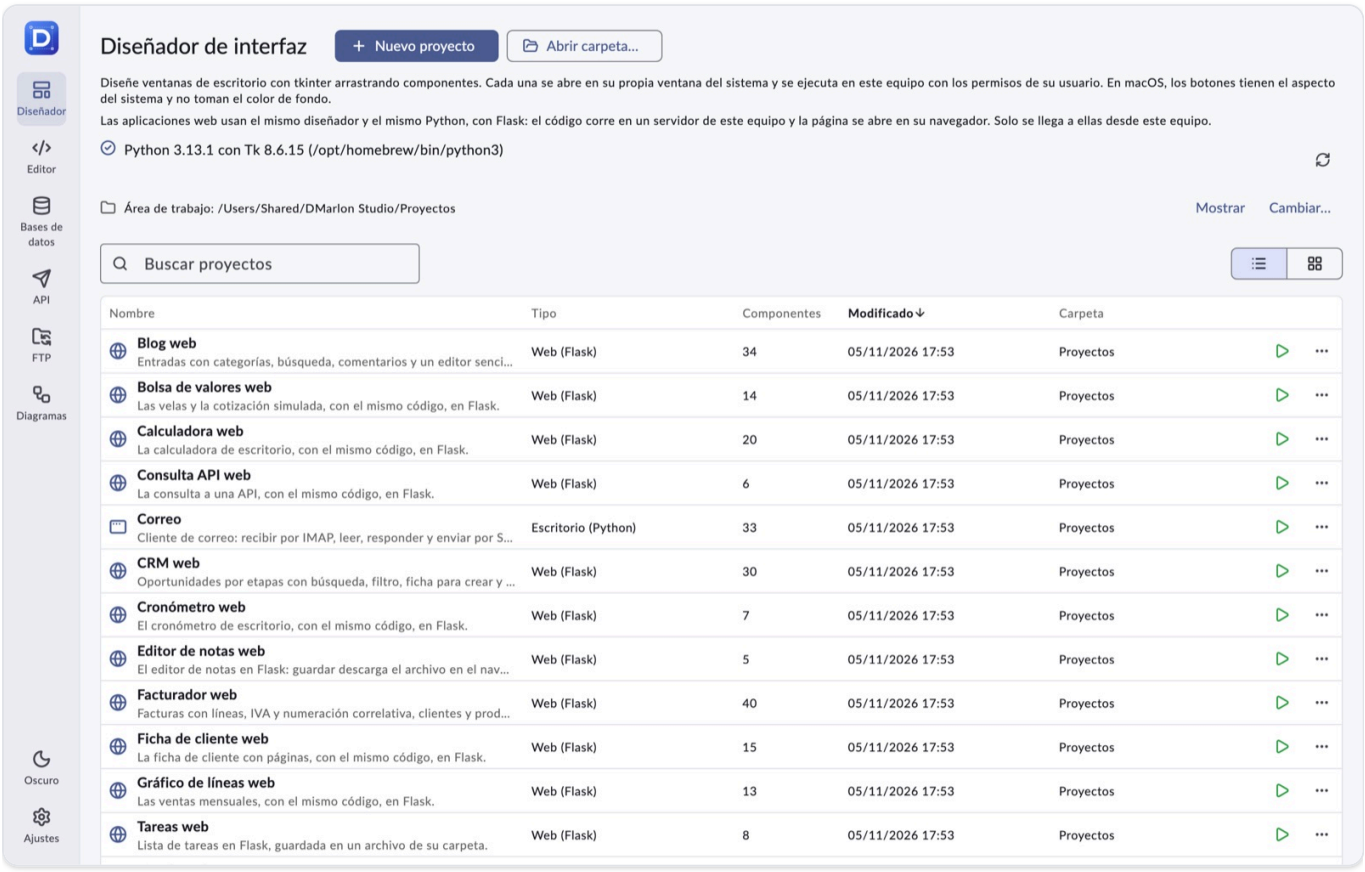
- **Nuevo proyecto**: pide el nombre y el destino, **Escritorio (Python)** o **Web (Flask)**. El destino no se puede cambiar después.
- **Abrir carpeta…**: abre un proyecto que esté en cualquier otra carpeta; queda en la lista como reciente.
- **Área de trabajo**: **Mostrar** la abre en el Finder o el Explorador y **Cambiar…** elige otra carpeta.
- En cada tarjeta: **Ejecutar**, **Mostrar la carpeta** y, en **Más acciones**, **GitHub…**, **Presupuesto de IA**, **Duplicar** y **Eliminar** (lleva el proyecto a la papelera del área de trabajo).
- **Pestañas**: cada proyecto abierto tiene su pestaña arriba, con un punto si hay cambios sin guardar. Puede tener varios abiertos a la vez; la pestaña **Proyectos** vuelve a la lista sin cerrar nada. Se cierran con la ✕ o con clic central, y preguntan si guardar.

> ⓘ El estudio copia en el área de trabajo los proyectos de ejemplo (los nuevos aparecen también en áreas de trabajo ya creadas). Vea la lista completa en **Ejemplos incluidos**.

La pantalla del Diseñador

- **Barra de herramientas**: volver a los proyectos, nombre de la aplicación (clic para renombrarla), **Guardar**, **Deshacer**, **Ejecutar (F9)** en verde, **Detener** en rojo, **Abrir en el navegador** (web), **Mostrar la carpeta del proyecto**, **Exportar el archivo .py**, **GitHub**, **Presupuesto de IA**, **Pedir a la IA**, **Chat con Claude Code**, alineación y centrado, **Ajustar a la rejilla** y **Ver la salida**.
- **Consola de salida**: abajo; se agranda o se reduce arrastrando su borde superior, se minimiza con la flecha y **Copiar la salida** copia todos los mensajes.
- **Paleta**: una barra de iconos; al pasar el ratón sale el nombre del componente. Se arrastra al formulario o se añade con doble clic.

- **Árbol**: la ventana y sus componentes, con los que están dentro de un marco o de una página sangrados.
- **Inspector**: **Propiedades** y **Eventos** del componente seleccionado o, si no hay ninguno, de la ventana.
- **Centro**: pestañas **Diseño**, **Código** y **Python generado**. **F12** pasa del diseño al código y vuelta.



La calculadora de ejemplo en el Diseñador.

Componentes

Todos los componentes tienen **Nombre**, posición y tamaño. Casi todos admiten **Color de fondo**, **Color del texto** y fuente (**Fuente**, **Tamaño de fuente**, **Negrita**, **Cursiva**), y los eventos de ratón (**Clic**, **Doble clic**, **Clic derecho**, **Entrar el ratón**, **Salir el ratón**). Los de escritura añaden los de teclado (**Recibir el foco**, **Perder el foco**, **Tecla pulsada**, **Tecla soltada**, **Tecla Intro**).

Componente	Propiedades propias	Eventos principales
Etiqueta (`Label`)	Texto, Alineación, Justificación, Relieve, Ajuste de línea	Ratón
Botón (`Button`)	Texto, Estado, Relieve	**Al pulsar** (`Button1_click`)
Campo de texto (`Entry`)	Texto inicial, Justificación, Ocultar con (p. ej. `*` para contraseñas), Estado, Tabla enlazada y Campo enlazado	Teclado y ratón

Componente	Propiedades propias	Eventos principales
Contraseña (`Password`)	Texto de ayuda, Botón para ver la contraseña, Longitud máxima, Estado	Teclado y ratón
Imagen (`Image`)	Archivo (en la carpeta `images` del proyecto), Ajuste (tamaño original, entera sin deformar, rellenar recortando, estirar), Borde	Ratón
Texto largo (`Text`)	Texto inicial (varias líneas), Ajuste de línea, Estado	Teclado y ratón
Casilla (`Checkbutton`)	Texto, Estado, Marcada	**Al pulsar**
Opción (`Radiobutton`)	Texto, Estado, Grupo, Valor, Marcada	**Al pulsar**
Lista (`Listbox`)	Elementos (uno por línea), Selección (browse, single, multiple, extended)	**Al seleccionar**
Lista desplegable (`Combobox`)	Elementos (uno por línea), Texto inicial, Justificación, Estado (normal, readonly, disabled)	**Al seleccionar**
Deslizador (`Scale`)	Título, Desde, Hasta, Paso, Valor, Orientación, Mostrar el valor	**Al cambiar** (recibe el valor)
Marco (`Frame`)	Relieve, Borde. Contiene otros componentes	Ratón
Lienzo (`Canvas`)	Color de fondo. Para dibujar desde el código	Ratón, **Arrastrar**, **Soltar el botón**, **Mover el ratón**
Tabla de datos (`Treeview`)	Fuente de datos, Consulta SQL, Columnas (campo|título|ancho), Límite, Orden, Cargar al abrir	**Al seleccionar**
Barra de progreso (`Progressbar`)	Valor, Máximo, Modo (determinate o indeterminate), Orientación	Ratón
Control de páginas (`PageControl`)	Páginas (una por línea), Página activa. Cada página contiene sus componentes	**Al cambiar de página**
Barra de estado (`StatusBar`)	Texto. Ocupa siempre el borde inferior de la ventana	Ratón

Componente	Propiedades propias	Eventos principales
Cinta de opciones (`Ribbon`)	Pestañas, grupos y botones (con sangría; «Botón | icono»), Pestaña activa	**Al pulsar un botón**, **Al cambiar de pestaña**
Menú (`Menu`)	Opciones (una por línea; sangría = submenú; `-` = separador). No visual: se ve arriba de la ventana	**Al elegir una opción** (recibe la opción)
Temporizador (`Timer`)	Intervalo (ms), Activo. No visual	**Cada intervalo**
Integración API (`RestClient`)	URL base, Autenticación (ninguna, token Bearer, básica o clave en una cabecera; el secreto, mejor en una variable de entorno), Cabeceras, Tiempo máximo. No visual	**Al terminar una llamada**, **Si falla una llamada**
Icono de bandeja (`TrayIcon`)	Texto al pasar el ratón, Imagen, Menú (una opción por línea). No visual	**Al hacer clic en el icono**, **Al elegir una opción del menú**
Diálogo Abrir (`OpenDialog`)	Título, Tipos («Texto|*.txt»), Extensión por defecto, Carpeta inicial, Elegir varios archivos. No visual	**Al elegir un archivo**
Diálogo Guardar (`SaveDialog`)	Título, Tipos, Extensión por defecto, Carpeta inicial, Nombre del archivo, Preguntar antes de sobrescribir. No visual	**Al elegir un archivo**
Asistente IA (`AiAssistant`)	Modelo de Claude, Instrucciones, Esfuerzo, Longitud máxima. No visual. Desde el código: `ask`, `chat` (con memoria; `reset` empieza otra), `summarize`, `translate`, `classify` (una etiqueta de una lista), `extract` (datos en un diccionario) y sus variantes `*_async`, que no congelan la ventana. Usa el SDK oficial de Anthropic (`pip install anthropic`); al ejecutar desde el estudio recibe la clave de **Ajustes** por la variable `ANTHROPIC_API_KEY`	**Al llegar la respuesta**, **Si falla la petición**
Informe PDF (`PdfReport`)	Título del documento, Nombre del archivo, Orientación, Autor, Números de página. No visual. Desde el código: `title`, `heading`, `text`, `table` (con totales y cabecera repetida en cada página), `image` (JPEG), `line`, `page_break`; luego `save`, `save_dialog`, `open` o `download` (web). Sin bibliotecas que instalar	—
Informe Excel (`ExcelReport`)	Nombre del archivo, Primera hoja. No visual. Desde el código: `sheet`, `header`, `write_row`, `write_rows` (texto, números, fechas y casillas con su tipo; cabecera fija con filtro); luego `save`, `save_dialog`, `open` o `download` (web)	—

Componente	Propiedades propias	Eventos principales
Gráfico (`Chart`)	Tipo (líneas, área, barras o velas), Título, Títulos de los ejes, Leyenda, Rejilla, Decimales, Prefijo y Sufijo de los valores, Puntos como máximo. Desde el código: `set_series` , `add_point` , `set_candles` , `add_candle` , `clear`	**Al hacer clic en un punto**
Conexión de datos (`Connection`)	Conexión (una de las guardadas en **Bases de datos**). No visual	—

Menú, Barra de progreso, Barra de estado, Temporizador y Control de páginas llegan con la versión 1.0.5; Contraseña, Imagen, Cinta de opciones, Integración API, Icono de bandeja y los diálogos Abrir y Guardar, con la 1.0.6; el Gráfico, con la 1.0.8. En una aplicación web, los componentes sin equivalente lo indican en el inspector.

Mover, alinear y editar

- Clic selecciona; arrastrar mueve; los tiradores cambian el tamaño. Con **Ajustar a la rejilla** activado todo encaja en la rejilla.
- Las flechas mueven la selección un paso de rejilla; con Alt (Opción), un píxel; con Mayús, cambian el tamaño.
- Con dos o más componentes seleccionados se activan **Alinear a la izquierda**, **Centrar en horizontal**, **Alinear a la derecha**, **Alinear arriba**, **Centrar en vertical**, **Alinear abajo**, **Igualar el ancho** e **Igualar el alto** (respecto al primero seleccionado).
- Con uno o más seleccionados, **Centrar en la ventana (horizontal)** y **Centrar en la ventana (vertical)** los centran como un bloque en la ventana o en su marco.
- El botón derecho ofrece **Cortar**, **Copiar**, **Pegar**, **Eliminar**, **Traer al frente** y **Enviar al fondo**.
- **Copiar entre proyectos**: lo copiado va al portapapeles del sistema, con sus hijos y **el código de sus eventos**. Se pega en otro proyecto (o tras reiniciar el estudio): los componentes toman nombres libres y sus métodos se renombran con ellos, sin pisar ninguno que ya exista. Los métodos auxiliares que llamen sus eventos se copian a mano.
- **Deshacer** (⌘Z / Ctrl+Z) vuelve atrás cada cambio del diseño. Si el código se cambió a mano después de un paso, ese paso restaura el diseño pero no el código, y se avisa.

Posición de la ventana

La propiedad **Posición** de la ventana decide dónde se abre: **Centrada en la pantalla** (la de los proyectos nuevos), **Por defecto** (la decide el sistema) o **Manual**, con **Izquierda** y **Arriba**.

Eventos y código

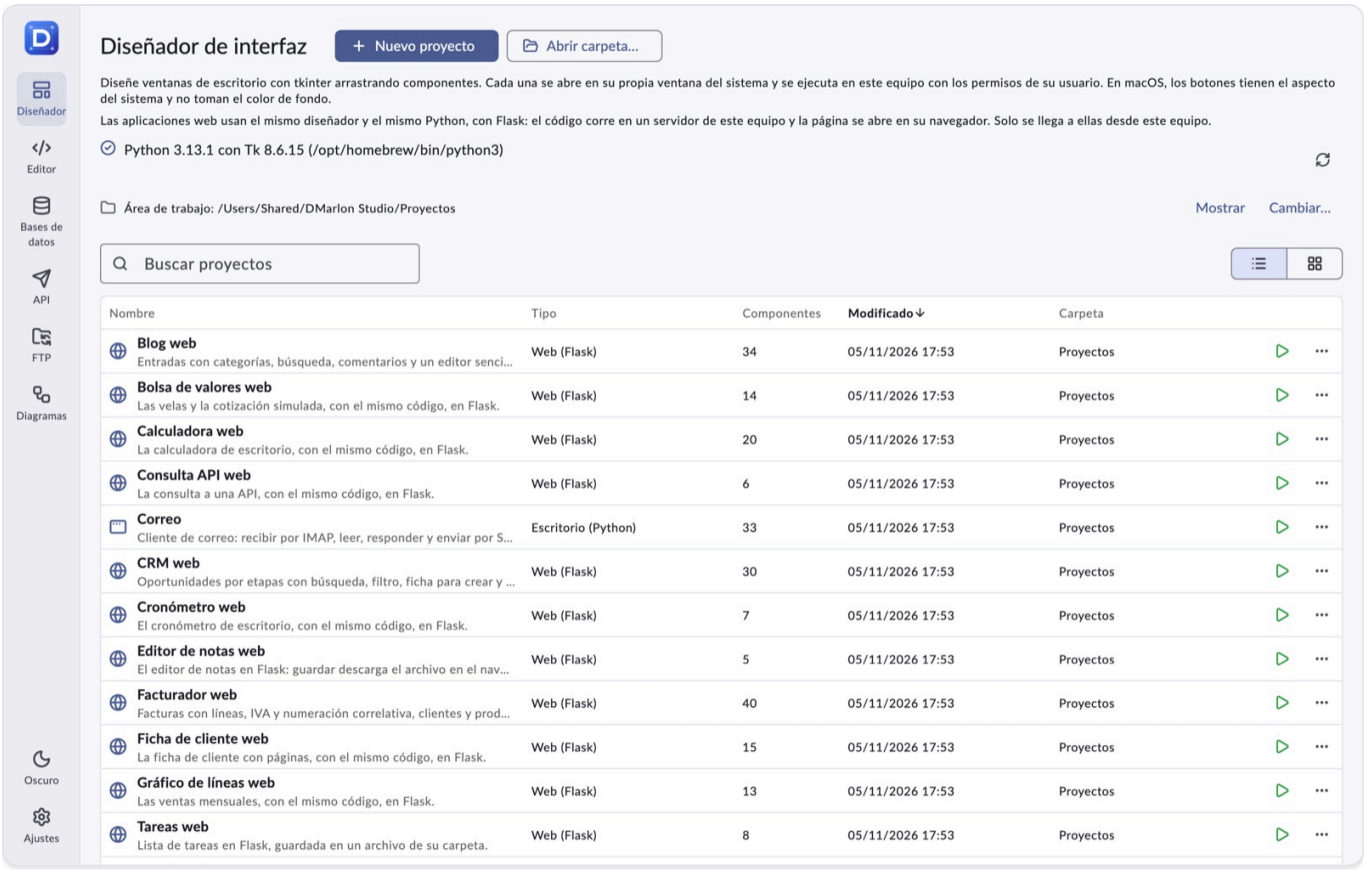
El código de la ventana es una clase de Python con el nombre del formulario que hereda de `FormDesign`, la clase que genera el estudio. `self.root` es la ventana y cada componente es `self.<Nombre>`.

- Doble clic en un componente abre su evento principal (`Button1_click`, `Listbox1_select`…) y lo crea si no existe.
- En la pestaña **Eventos** del inspector se escribe o se elige el método de cada evento; **Ir al código** salta a él.
- Renombrar un componente renombra sus métodos de serie y las referencias del código.
- La ventana tiene sus eventos: **Al abrir** (`form_create`), **Al cerrar**, **Tecla pulsada** y **Al cambiar de tamaño**, y sus propiedades **Título**, **Color de fondo**, **Redimensionable** y **Siempre encima**.

```
class Form1(FormDesign):
    def form_create(self):
        self.total = 0
        self.StatusBar1.config(text="Listo")

    def Button1_click(self):
        self.total += 1
        self.Label1.config(text=f"Pulsado {self.total} veces")
```

⚠ Ponga lo que deba ocurrir al arrancar en el evento **Al abrir** (`form_create`). Si define su propio `__init__`, llame primero a `super().__init__()`.

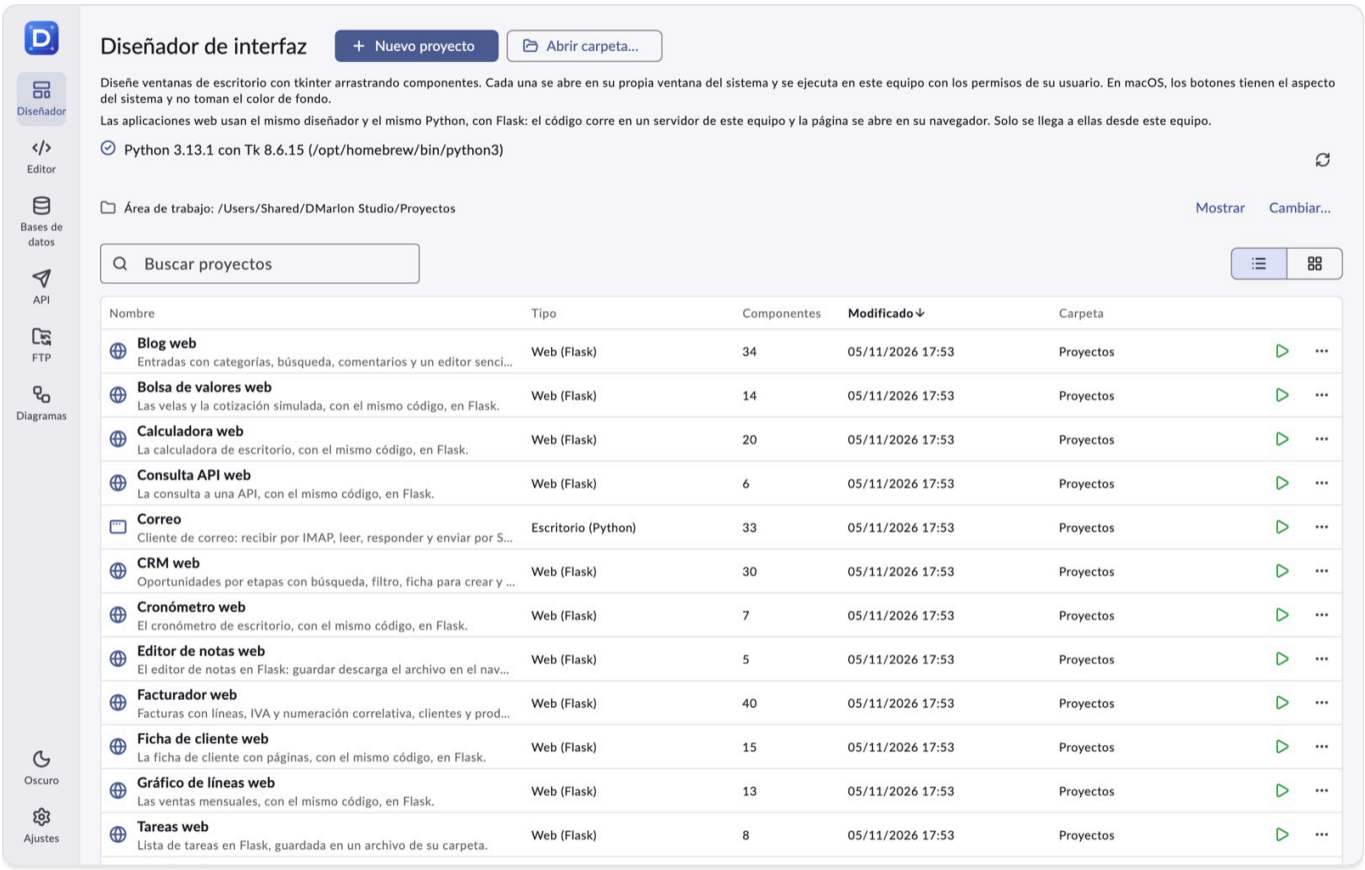


La pestaña Código: la clase de la ventana con sus manejadores.

Python generado y exportación

La pestaña **Python generado** muestra, solo para leer, el archivo completo que se ejecuta: el diseño convertido en código más su clase. Al ejecutar se escribe en `build/<proyecto>.py`, dentro de la carpeta del proyecto.

Exportar el archivo .py guarda ese archivo donde quiera. Funciona con cualquier Python 3 con tkinter (o con Flask, si es web) sin el estudio.



El Python generado: no se edita, se rehace desde el diseño y el código.

Ejecutar, detener y ver la salida

- **Ejecutar (F9)** guarda el proyecto y lanza la ventana en su propia ventana del sistema, con los permisos de su usuario.
- **Detener** cierra la ventana o el servidor en marcha.
- **Ver la salida** abre el panel con lo que la aplicación escribe (`print`, errores de Python) y el Python usado.
- En una aplicación web, el estudio arranca el servidor de Flask en este equipo y abre el navegador; **Abrir en el navegador** vuelve a abrirla. Solo se llega a ella desde este equipo.

Conexiones de datos

1. Guarde la base de datos en el módulo **Bases de datos** (SQLite, PostgreSQL o MySQL/MariaDB).
2. Añada una **Conexión de datos** al formulario y elija esa conexión en su propiedad **Conexión**.
3. Añada una **Tabla de datos**, elija la conexión en **Fuente de datos** y escriba la **Consulta SQL** y las **Columnas** (`campo|título|ancho`).
4. Para editar la fila elegida, enlace un **Campo de texto** con **Tabla enlazada** y **Campo enlazado**.

Desde el código, `self.Connection1.query(sql, params)` devuelve filas y `self.Connection1.execute(sql, params)` modifica datos (siempre con parámetros `?`, nunca uniendo texto). `self.Treeview1_refresh()` vuelve a cargar la tabla.

> Las contraseñas no se escriben en el archivo generado: llegan al proceso en variables de entorno `DMSTUDIO_DB_…`. Un archivo exportado las lee del entorno.

Capítulo 4

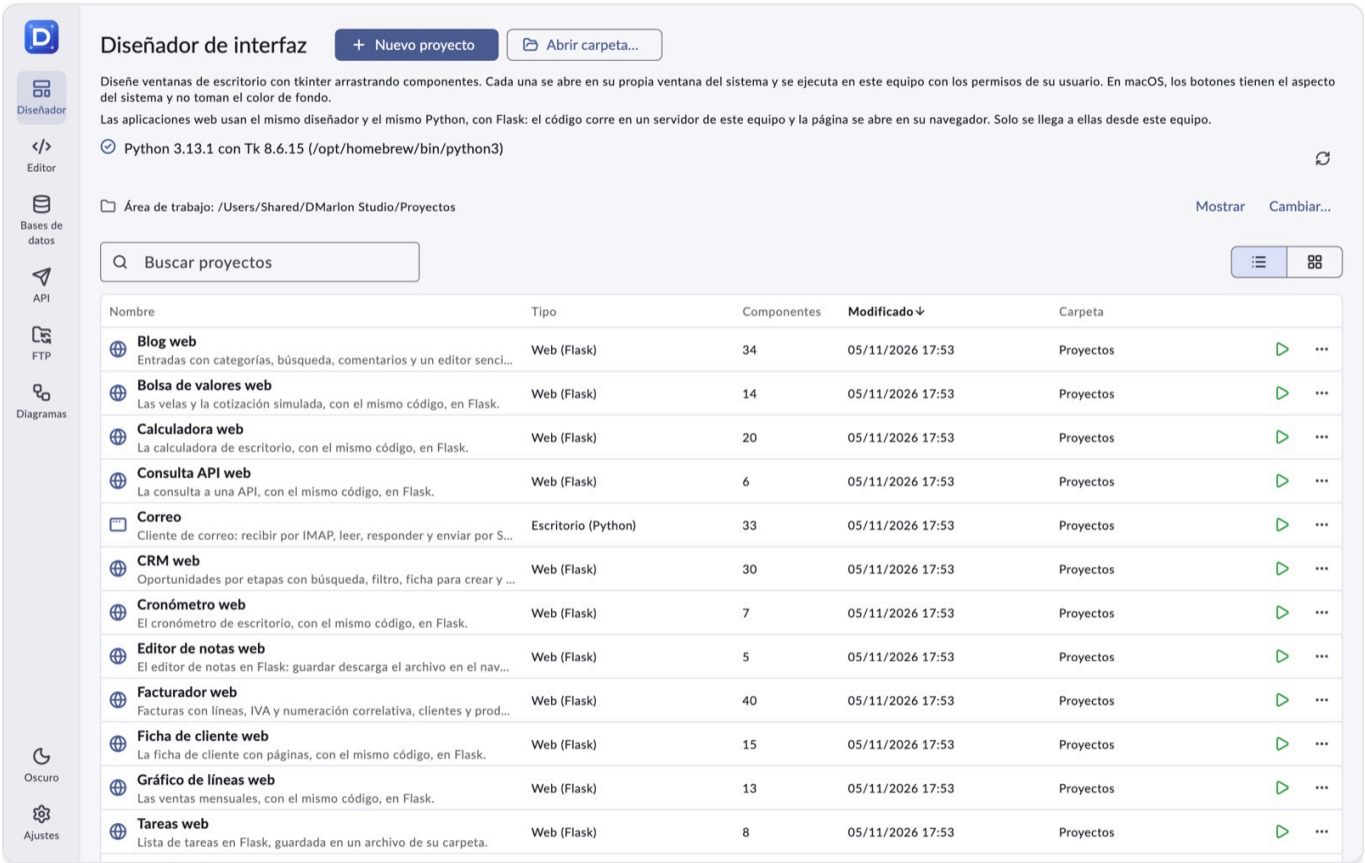
IA en el Diseñador

Chat con Claude Code y «Pedir a la IA».

Chat con Claude Code

El botón **Chat con Claude Code** (el robot de la barra) despliega un panel a la derecha. Describa el cambio y pulse Intro (Mayús+Intro hace un salto de línea).

- Usa la Claude Code instalada en su equipo y su sesión: no necesita clave de API.
- Antes de cada mensaje el estudio guarda el proyecto. Claude Code lee y edita el diseño (`project.dmsproj`) y el código de la ventana, sin terminal.
- Al terminar, el diseño y el código nuevos se cargan en el Diseñador como un solo paso que se puede deshacer.
- La conversación sigue entre mensajes; la papelera empieza una nueva.
- El clip 📎 adjunta imágenes (un boceto, una captura) o archivos de texto: Claude Code los lee sin que entren en el proyecto.



El panel de Claude Code, a la derecha del formulario.

Pedir a la IA

Pedir a la IA usa el proveedor configurado en **Ajustes** (Claude por defecto) y tiene tres modos:

- **Cambiar**: modifica la ventana actual («Añade un campo de teléfono y alinea los botones»).
- **Ventana nueva**: diseña una ventana desde cero; puede **Adjuntar un boceto** (una imagen).
- **Código de** un componente: con un componente seleccionado, escribe el método de su evento principal.

La respuesta llega como **Propuesta de la IA**: se revisa antes de aplicarla y, una vez aplicada, se deshace en un paso.

Presupuesto de IA

El botón de la cartera (**Presupuesto de IA**) muestra lo gastado en Claude en el proyecto: por herramienta (Claude Code, Claude implementa, Pedir a la IA, chat), por modelo y por día, con las últimas llamadas y sus tokens.

- Se puede poner un **Tope (US$)**, **Mensual** o **Total**. Avisa al 80 % y, con **Bloquear al agotarse**, la IA deja de responder en ese proyecto hasta que se suba.
- Los costes usan la tarifa de la API de Anthropic. Con una suscripción de Claude, el de Claude Code es su equivalente en API, no lo que se cobra.

GitHub

- El botón **GitHub** del Diseñador conecta su cuenta con un token (se guarda solo en este equipo).
- **Publicar en GitHub** crea el repositorio (privado o público), hace el primer commit y lo sube.
- Ya publicado: lista los cambios, pide el **Mensaje del commit** y **Confirmar y subir**; también **Traer cambios** y **Abrir en GitHub**.

Capítulo 5

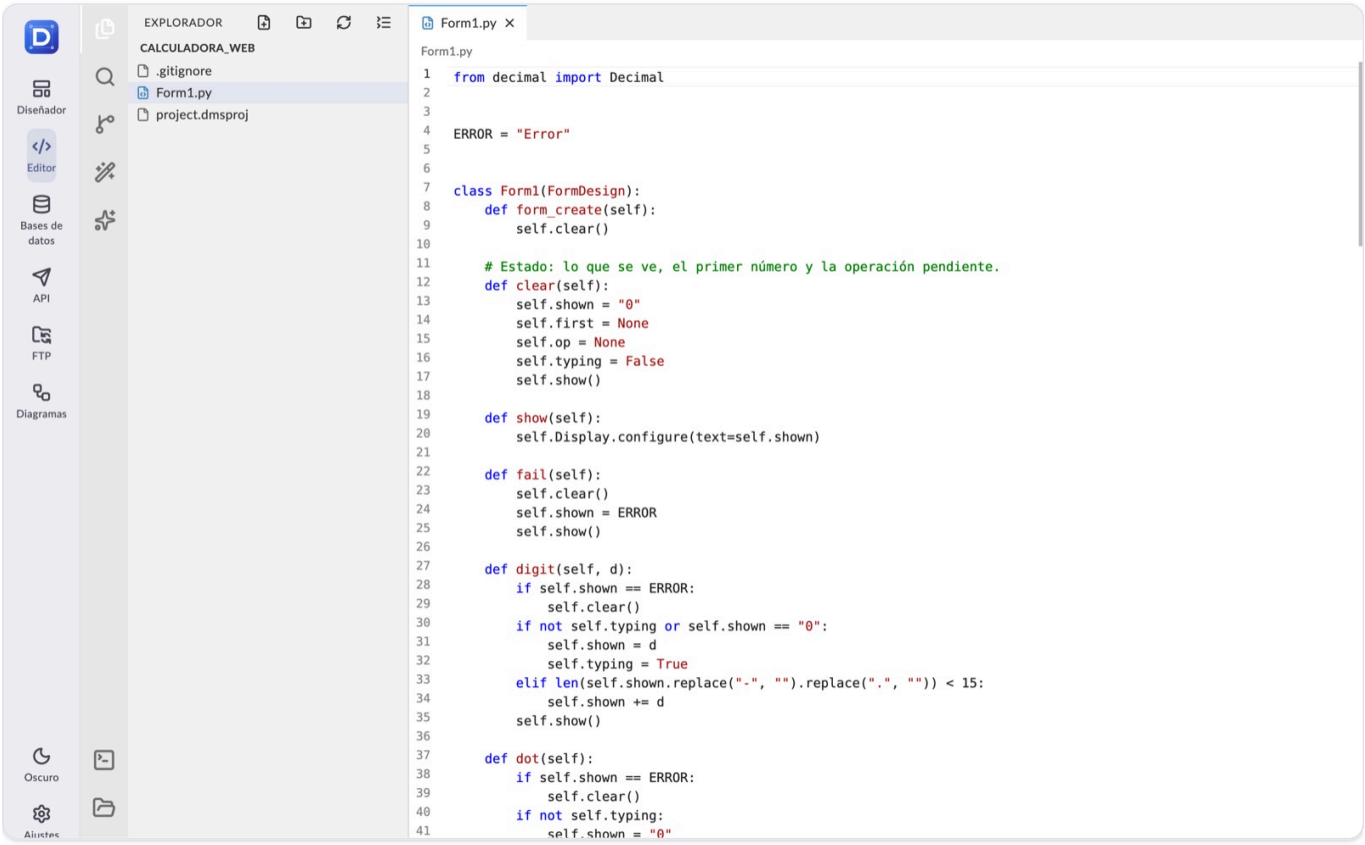
El Editor

Editor de código con explorador, búsqueda, terminal, git e IA.

Abrir una carpeta

Al entrar sin carpeta se ofrecen **Abrir carpeta…**, **Abrir DMarlon Studio** (la carpeta de documentos del estudio), **Clonar repositorio…** y la lista de **Recientes**.

Si hay un proyecto abierto en el Diseñador, el Editor muestra su código y su Python generado como **Proyecto abierto en el Diseñador de interfaz**: lo que escriba aquí es el mismo texto que en la pestaña Código.



El Editor: barra de actividad, explorador y pestañas.

Explorador y pestañas

- La barra de actividad tiene **Explorador**, **Buscar**, **Control de código**, **Claude implementa** y **Chat con la IA**, y abajo la terminal y **Abrir carpeta**.
- El explorador permite **Nuevo archivo**, **Nueva carpeta**, **Renombrar**, **Eliminar**, **Copiar la ruta**, **Abrir en la terminal** y **Contraer todo**.

- Las pestañas tienen resaltado de sintaxis para decenas de lenguajes y números de línea. ⌘S / Ctrl+S guarda; al cerrar algo sin guardar se pregunta.
- Lo que no guarde se conserva al cambiar de módulo.

Buscar y reemplazar

- **Buscar en el archivo** con **Reemplazar** y **Reemplazar todo**.
- El panel **Buscar** recorre todos los archivos de la carpeta, con **Distinguir mayúsculas** y **Expresión regular**; un clic abre el archivo en la línea.

Terminal

⌘J / Ctrl+J muestra u oculta la terminal, que ejecuta órdenes en la carpeta abierta (`git status`, `python3 app.py`, `npm install` …). Las flechas arriba y abajo recorren el historial y **Detener** corta la orden en marcha.

Control de código (git)

- **Cambios**: archivos modificados con su diff; escriba el **Mensaje del commit** y use **Confirmar todo** o **Confirmar y subir**. También **Corregir el último commit** y **Deshacer el último commit**.
- **Subir (push)**, **Traer (pull)** y **Buscar cambios en el remoto (fetch)**, con lo pendiente en cada sentido.
- **Historial** de la rama o de **Todas las ramas**: ver el commit, **Revertir este commit**, **Aplicar en la rama actual (cherry-pick)**, crear una rama o una etiqueta desde él y **Volver la rama a este commit…**.
- **Ramas** locales y remotas: crear, cambiar, **Fusionar**, renombrar y eliminar (también del remoto).
- **Guardar los cambios aparte (stash)** para cambiar de rama y recuperarlos después.
- **Remotos…**, **Inicializar repositorio**, **Clonar repositorio…** y **Publicar en GitHub…** (público o privado) con su **Cuenta de GitHub** (un token con permiso «repo»).

Claude implementa

Claude implementa pide un cambio a Claude a través de la API de Anthropic (Claude Opus 5.5 por defecto). Claude explora la carpeta abierta (lista, lee y busca archivos) y propone cambios en uno o varios archivos.

- Nada se escribe hasta que usted lo aplica: la lista **Cambios propuestos** muestra cada archivo con sus líneas añadidas y quitadas; un clic abre el diff con **Aplicar** y **Descartar**.

- **Aplicar todo** escribe todos los cambios; si un archivo cambió en el disco mientras Claude trabajaba, se avisa en lugar de pisarlo.
- Necesita la clave de Anthropic en **Ajustes → Inteligencia artificial**.
- El clip 📎 adjunta imágenes y archivos de texto o código a la petición; el botón de la cartera abre el **Presupuesto de IA** de la carpeta.

Chat con la IA

El **Chat con la IA** responde sobre el archivo abierto con el proveedor de **Ajustes**. Si propone el archivo completo, **Aplicar al archivo** lo pone en la pestaña (sin guardar) y **Copiar el código** lo copia.

En los tres chats Intro envía, Mayús+Intro salta de línea y el clip 📎 adjunta imágenes (PNG, JPG, GIF o WebP de hasta 5 MB) y archivos de texto (hasta 300 KB).

Capítulo 6

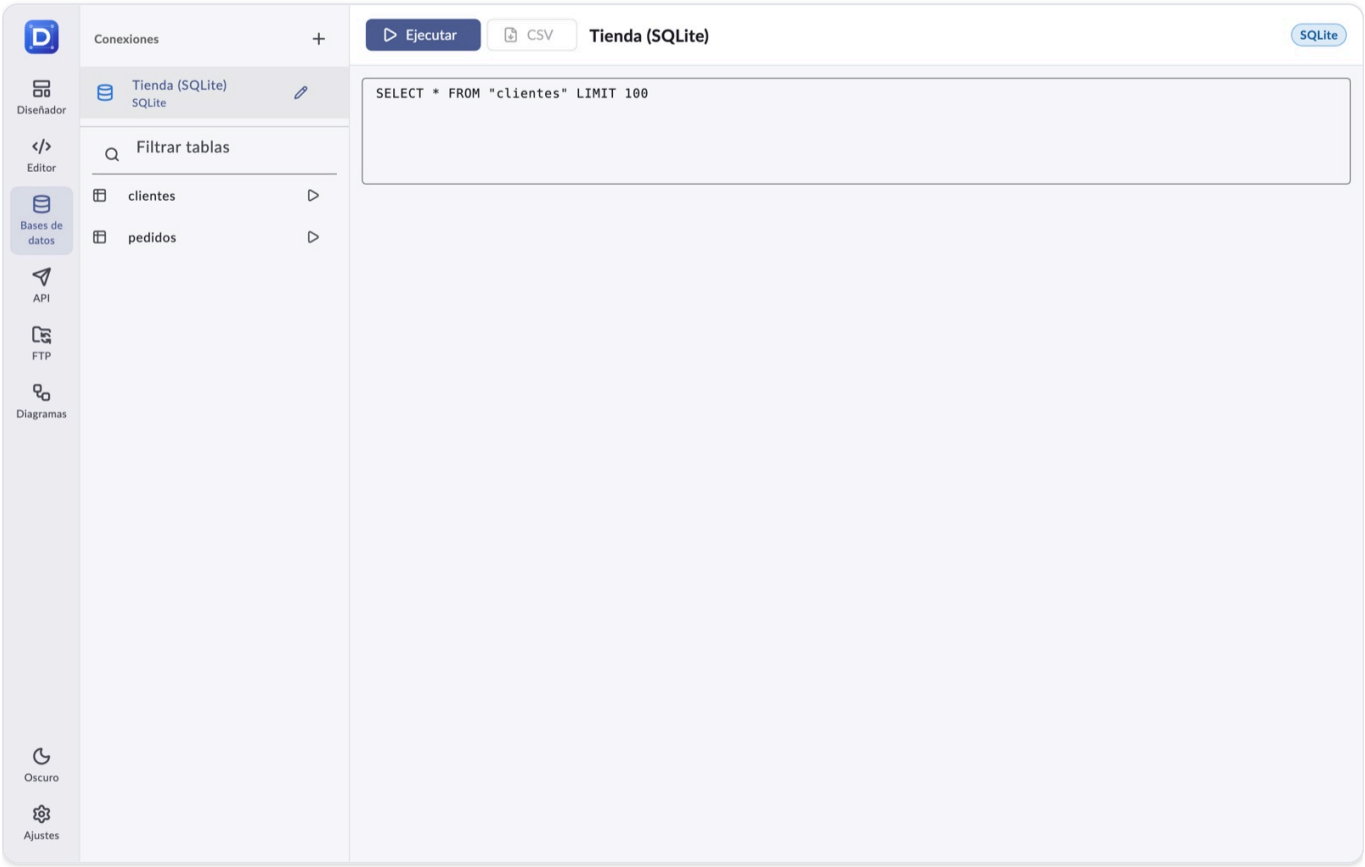
Bases de datos

Conexiones, tablas, consultas SQL y exportación a CSV.

Conexiones

Con **+** se añade una conexión de tipo SQLite, PostgreSQL, MySQL / MariaDB o MongoDB, con servidor, puerto, base de datos, usuario, contraseña y **Conexión cifrada (SSL)** según el tipo. **Probar conexión** comprueba que se llega.

Las conexiones guardadas aquí son las que pueden usar las ventanas del Diseñador con el componente **Conexión de datos**.



Una base SQLite con sus tablas y el editor de consultas.

Tablas y consultas

- La lista de tablas se puede **Filtrar tablas**; cada tabla abre una consulta de ejemplo.
- **Ejecutar** lanza la consulta del editor y muestra el resultado en una rejilla.
- Antes de ejecutar algo que cambia datos se pide confirmación.

- **CSV** guarda el resultado completo en un archivo.

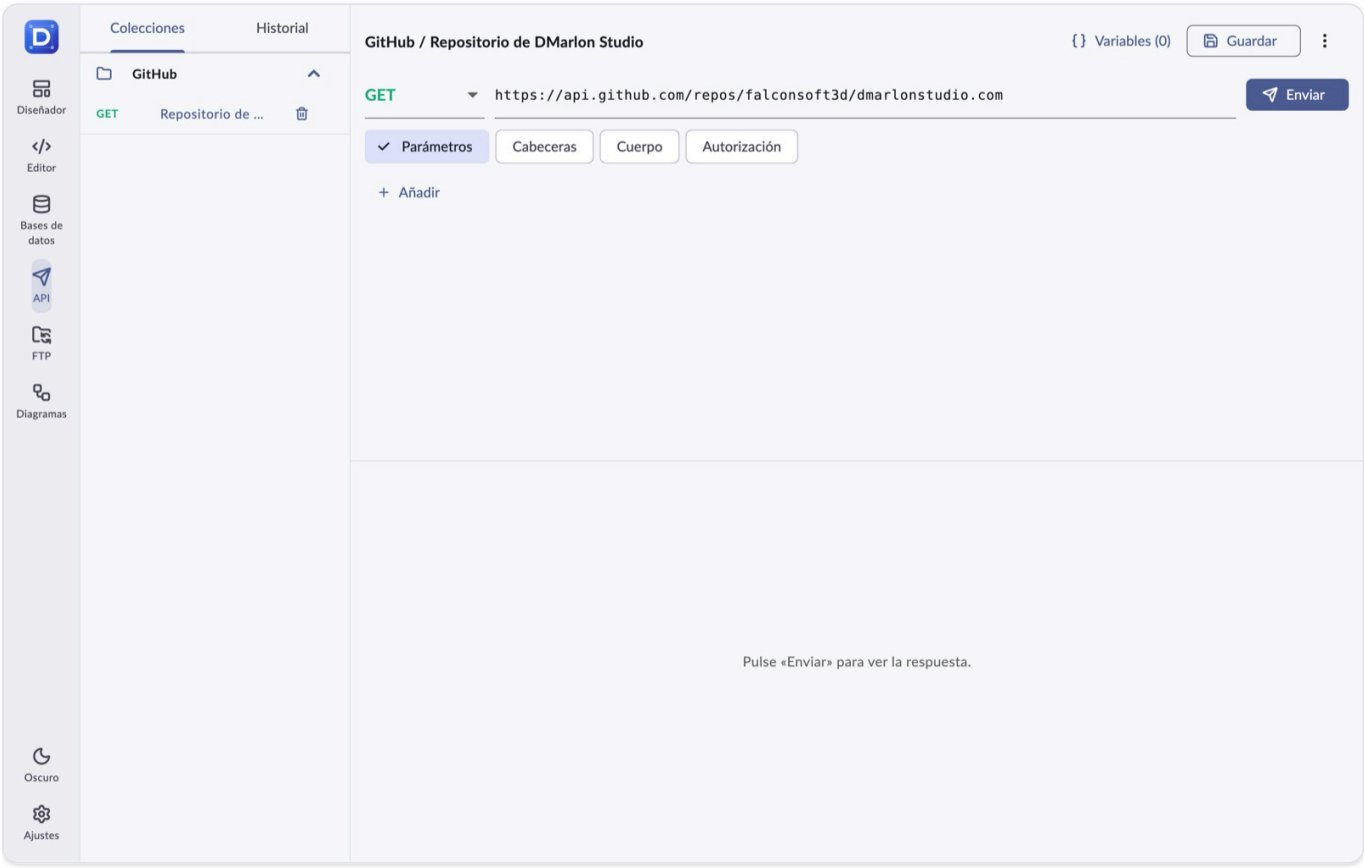
Capítulo 7

Cliente de API

Peticiones HTTP con colecciones, variables e historial.

Peticiones

- Elija el método (GET, POST, PUT, PATCH, DELETE…), escriba la URL y pulse **Enviar**.
- Pestañas **Parámetros**, **Cabeceras**, **Cuerpo** (JSON, formulario o texto) y **Autorización** (ninguna, Bearer o Basic).
- La respuesta muestra el código, el tiempo, las cabeceras y el cuerpo.
- **Copiar como curl** deja la petición lista para la terminal.



Una petición guardada en la colección GitHub.

Colecciones, variables e historial

- **Guardar** guarda la petición en una **Colección**; **Guardar como nueva…** hace una copia.
- **Variables de entorno…** define valores como `base_url` o `token`, que se usan en la URL, las cabeceras y el cuerpo como `{{base_url}}`.

- El **Historial** guarda las 100 últimas peticiones enviadas; **Borrar historial** lo vacía.

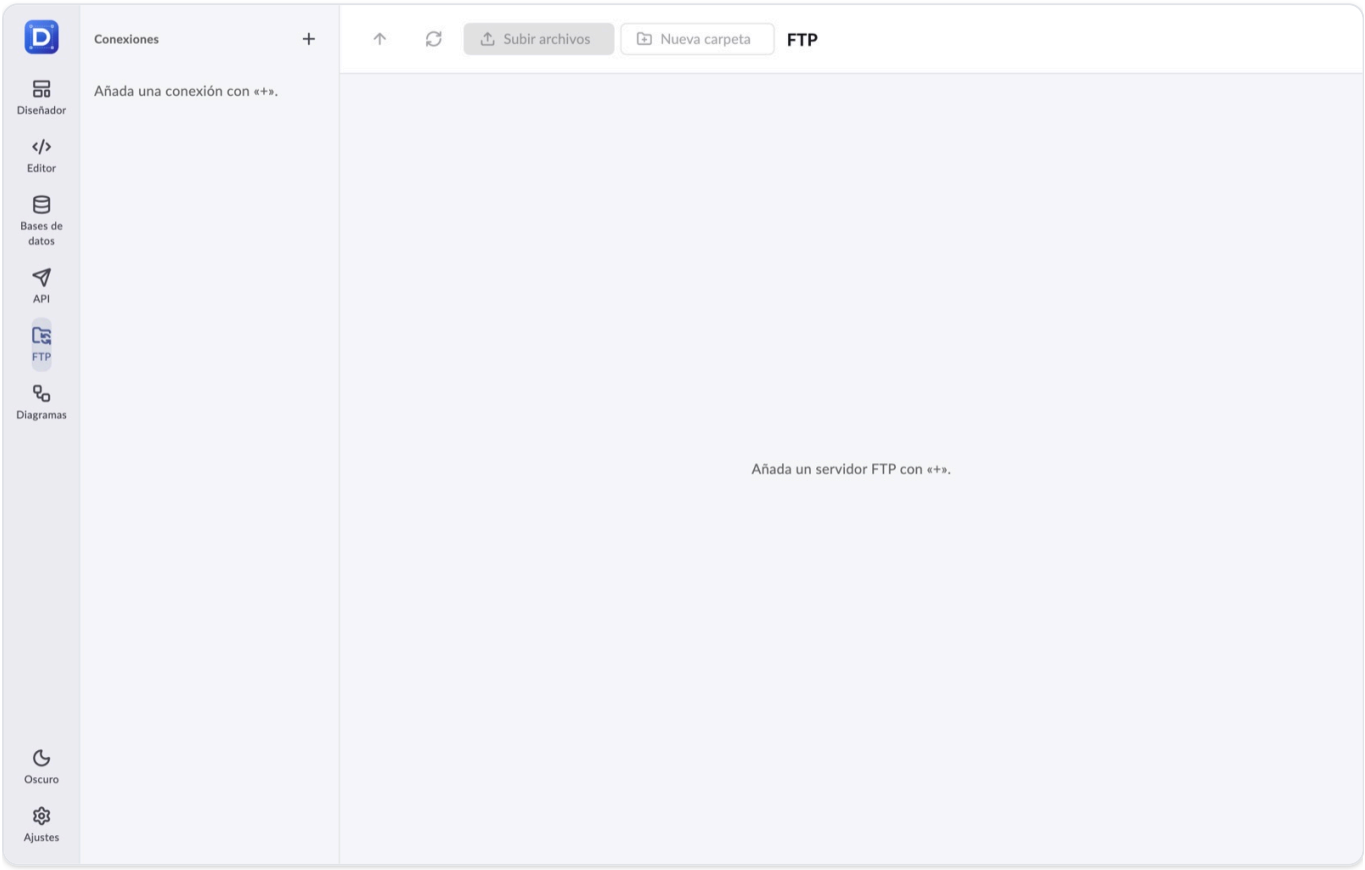
Capítulo 8

FTP

Servidores FTP, FTPES y FTPS.

Servidores y archivos

- Con **+** se añade un servidor con su **Seguridad**: **FTP (sin cifrar)**, **FTPES (TLS explícito)** o **FTPS (TLS implícito)**.
- Elija un servidor para recorrer sus carpetas; **Carpeta superior** sube un nivel.
- **Subir archivos**, **Descargar**, **Nueva carpeta**, **Renombrar…** y **Eliminar** (las carpetas, con su contenido).



El cliente FTP, listo para añadir un servidor.

Capítulo 9

Diagramas

Diagramas de flujo y de bases de datos, con IA y PDF.

Diagramas de flujo y de base de datos

- **De flujo**: figuras **Inicio / fin**, **Proceso**, **Decisión**, **Entrada / salida** y **Nota**. Pulse → junto a una figura para conectarla con otra; las conexiones pueden llevar etiqueta (p. ej. «Sí»).
- **De base de datos**: tablas con sus columnas, **Clave primaria** y claves ajenas, y las relaciones entre tablas. **SQL del diagrama** muestra el SQL que lo crea.
- **Desde una base de datos** dibuja el diagrama de una conexión guardada (opcionalmente, solo las tablas que empiezan por un prefijo).
- **Con IA** crea el diagrama a partir de una descripción y **Cambiar con IA** lo modifica.
- **Ordenar** recoloca las figuras y **Exportar a PDF** lo guarda para imprimir o compartir.



Un diagrama de flujo para dar de alta un cliente.

Capítulo 10

Ajustes

Idioma, apariencia e inteligencia artificial.

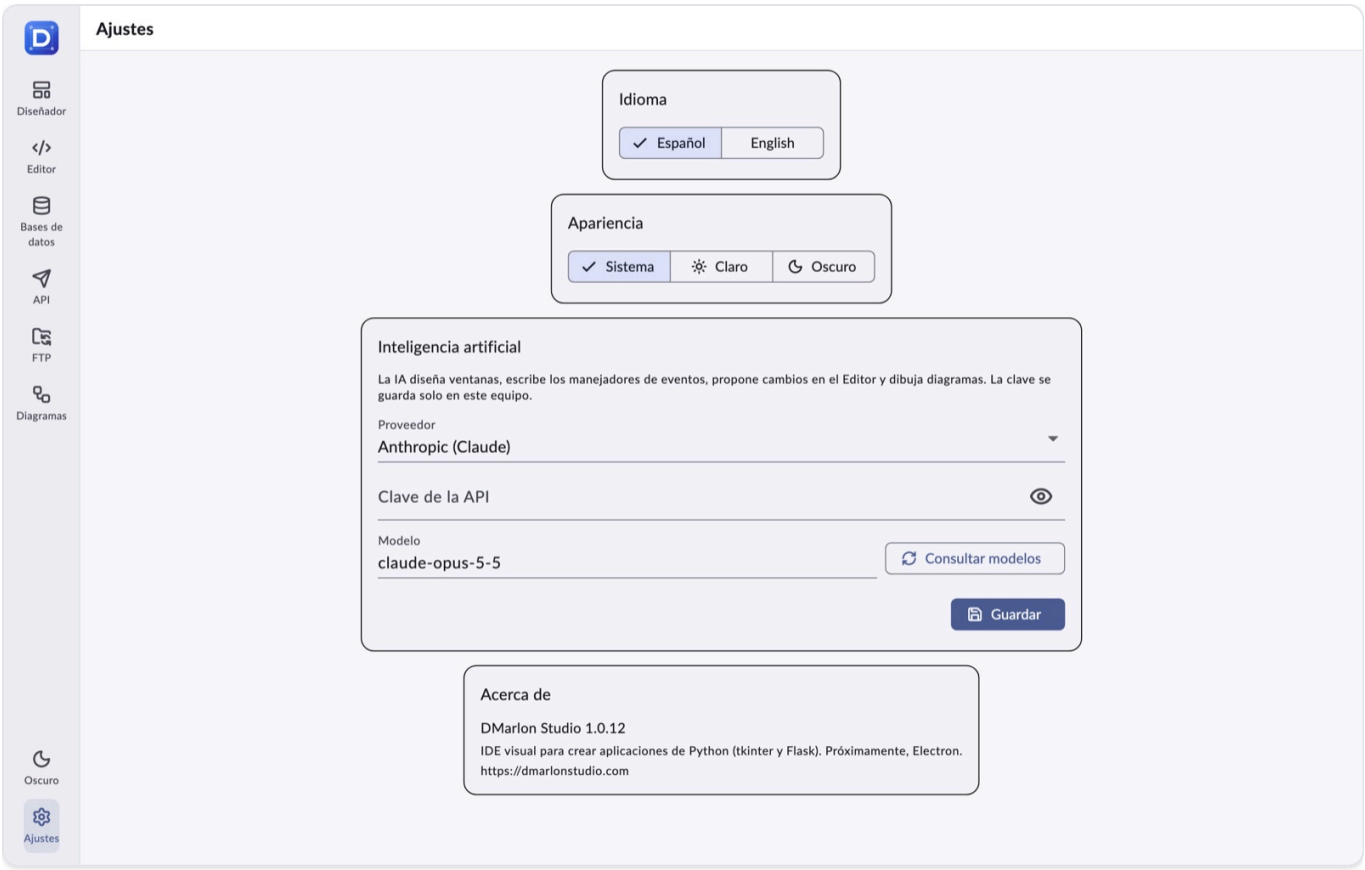
Conectar con Claude (MCP)

- En **Ajustes → Conectar con Claude (MCP)** se activa el **Servidor MCP**: mientras DMarlon Studio está abierto, Claude (Claude Code u otro cliente MCP) puede trabajar con sus proyectos.
- Herramientas: `list_projects`, `get_project`, `list_components`, `create_project`, `add_component`, `update_component`, `remove_component`, `update_code`, `generate_python`, `run_project`, `get_output` y `stop_project`.
- Solo responde en este equipo (`127.0.0.1`) y con el **Token**. **Añadir a Claude Code** lo da de alta con un clic; también se puede copiar la orden `claude mcp add` que se muestra.
- Si un proyecto está abierto en el Diseñador sin cambios pendientes, se recarga solo con lo que haga Claude; con cambios sin guardar, Claude recibe un aviso para que primero se guarden.

Inteligencia artificial

Elija el **Proveedor**: Anthropic (Claude), OpenAI, Gemini, xAI, DeepSeek, Mistral, Groq, Cohere, OpenRouter, Perplexity, Ollama o una API personalizada compatible. Escriba la **Clave de la API** y el **Modelo**; **Consultar modelos** trae los de su cuenta. Claude (`claude-opus-5-5`) es el proveedor por defecto.

La clave se guarda solo en este equipo. Cada proveedor recuerda su clave y su modelo al cambiar de uno a otro.



Ajustes: idioma, apariencia, inteligencia artificial y versión.

Idioma, apariencia y versión

- **Idioma**: Español o English, al momento.
- **Apariencia**: Sistema, Claro u Oscuro.
- **Acerca de** muestra la versión instalada.

Capítulo 11

Atajos de teclado

Todos los atajos del Diseñador y del Editor.

Diseñador

Atajo	Acción
F9	Ejecutar
F12	Pasar del diseño al código y vuelta
⌘S / Ctrl+S	Guardar el proyecto
⌘Z / Ctrl+Z	Deshacer
⌘C, ⌘X, ⌘V (Ctrl en Windows)	Copiar, cortar y pegar componentes
Supr / Retroceso	Eliminar la selección
Flechas	Mover un paso de rejilla (con Alt, un píxel)
Mayús + flechas	Cambiar el tamaño
Doble clic en la paleta	Añadir el componente
Doble clic en un componente	Abrir su evento principal
Intro en un chat	Enviar (Mayús+Intro: salto de línea)

Editor

Atajo	Acción
⌘S / Ctrl+S	Guardar el archivo
⌘J / Ctrl+J	Mostrar u ocultar la terminal
↑ / ↓ en la terminal	Recorrer el historial de órdenes
Intro en un chat	Enviar (Mayús+Intro: salto de línea)

Capítulo 12

Solución de problemas

Los avisos más habituales y cómo resolverlos.

No encuentra Python o le falta tkinter

- **No se encontró Python**: instálelo (python.org o Homebrew) y pulse **Volver a buscar Python** en la lista de proyectos.
- **Se encontró … pero no trae tkinter**: en macOS, `brew install python-tk`; en Linux, `sudo apt install python3-tk`; en Windows, reinstale Python marcando **tcl/tk and IDLE**.
- **Tk 8.5 es antiguo y no respeta los colores del diseño**: es el Python que trae macOS. Instale uno de python.org o Homebrew.

Falta Flask

Las aplicaciones web necesitan Flask en el mismo Python. El aviso **Falta Flask** muestra la orden exacta, por ejemplo `python3 -m pip install flask`.

La app no abre (sin firmar)

- **macOS**: la primera vez, clic derecho sobre DMarlon Studio → **Abrir** → **Abrir**.
- **Windows**: en «Windows protegió su PC», **Más información** → **Ejecutar de todas formas**.

AttributeError: 'Form1' object has no attribute 'root'

Pasaba cuando el código de la ventana definía su propio `__init__` sin llamar a `super().__init__()`: ese método sustituía al que crea la ventana. Desde la versión 1.0.5 la ventana y los componentes se crean antes de su `__init__`, y el evento **Al abrir** se ejecuta después.

```
class Form1(FormDesign):
    def __init__(self):
        super().__init__()   # primero
        self.current = "0"

    # Mejor aún: el evento Al abrir
    def form_create(self):
        self.current = "0"
```

No se encontró Claude Code

Instale Claude Code desde claude.com/claude-code y ejecute `claude` una vez en una terminal para iniciar sesión. El estudio lo busca en el PATH y en `~/.local/bin`.

Capítulo 13

Hoja de ruta

Lo que viene después.

Destinos

Al crear un proyecto se elige el destino; no se puede cambiar después:

Destino	Estado
Python + tkinter (escritorio)	Disponible
Python + Flask (web)	Disponible

Proyectos más completos

- Varios formularios por proyecto, con un nodo **Proyecto** en el árbol.
- El icono de la aplicación, definido en el proyecto.
- Compilar ejecutables (PyInstaller para Python).