NEWS · OCTOBER 19, 2026

Construir, no comprar: software a medida en 12 días en lugar de otra licencia SaaS

El hogar de una agente inmobiliaria funcionaba con hojas de cálculo, papel y un puñado de suscripciones -- así que le construí un sistema propio en doce días, con archivo de documentos por IA, passkeys, una bóveda cifrada y apps para iPhone, iPad, Apple TV, Mac y Chrome, y esto es lo que significa para tu aplicación Delphi y tu próxima decisión entre comprar o construir.

By Dr. Holger Flick

Mi esposa es agente inmobiliaria. Durante años, nuestro hogar funcionó como funcionan la mayoría de los hogares -- y, siendo sinceros, la mayoría de las pequeñas empresas: una hoja de cálculo para las pólizas de seguro, una carpeta de PDF que nadie encontraba, una app de calendario, una app de notas, un gestor de contraseñas y una pila creciente de papeles en la encimera de la cocina que todos esquivaban. Cada una de esas herramientas hacía bien su trabajo. Ninguna sabía nada de las demás, y ninguna sabía nada del trabajo de *ella*.

Así que, para su cumpleaños, le construí un solo sistema en su lugar. Se llama **Hearth**. Lee con IA cada

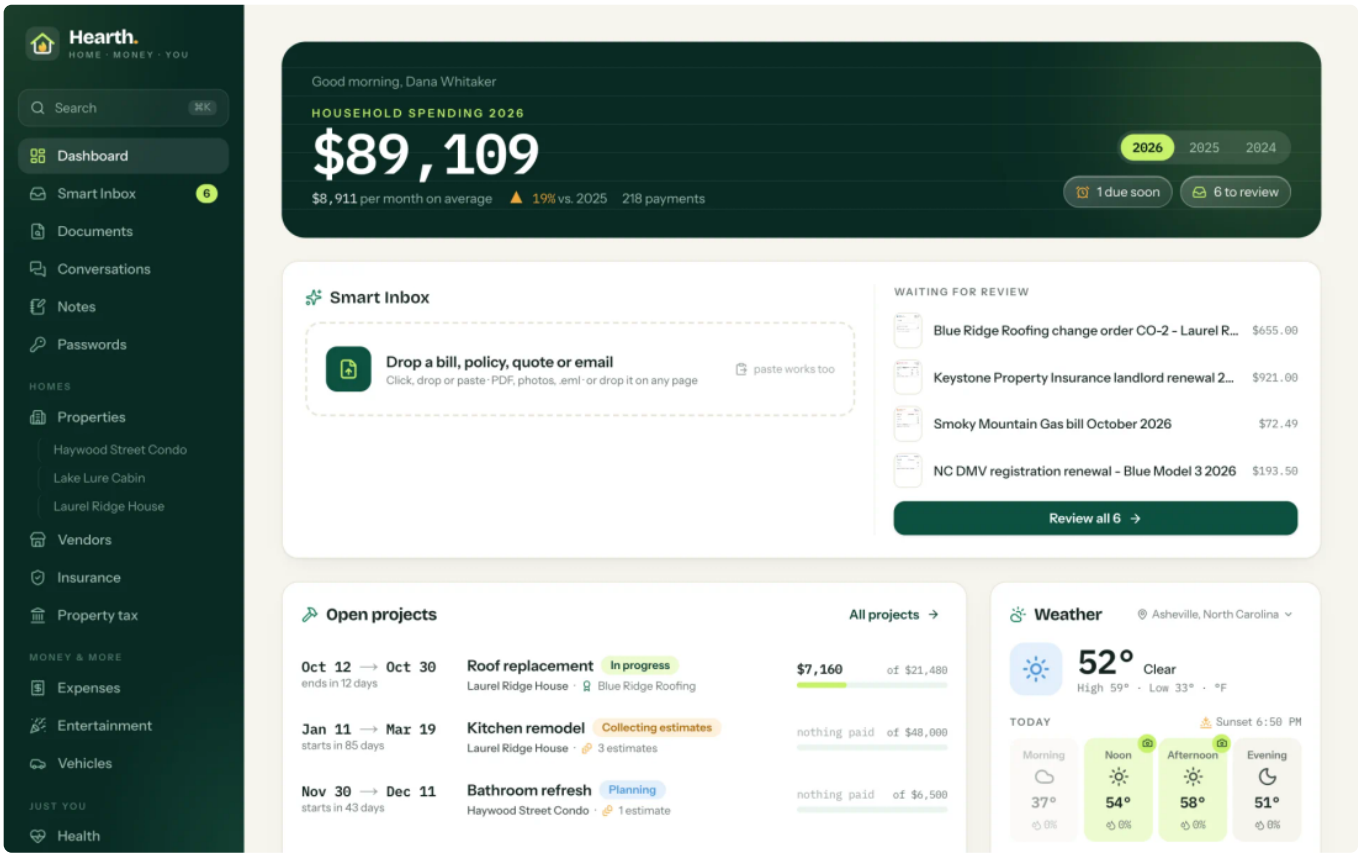
factura, póliza y recibo, lo archiva bajo la casa, el auto o el proyecto correctos, lleva el calendario, las notas y las contraseñas, y lo muestra todo en la web, en su iPhone y su iPad, en el Apple TV de la sala, en el Mac y dentro de Chrome. Desde el primer prompt hasta la última versión publicada pasaron doce días.

Este artículo es el recorrido, y también es un argumento. En mi opinión, la pregunta «¿qué producto deberíamos comprar?» es hoy, muy a menudo, la primera pregunta equivocada. Para cada vez más empresas, la mejor pregunta es «¿cómo sería un software que encaje exactamente con nosotros?» -- porque el precio de la

respuesta se ha desplomado. Te mostraré qué se construyó, cuánto habría llevado sin IA, dónde se sostiene mi argumento y dónde no, y qué significa todo esto si tienes una aplicación de escritorio en Delphi que merece una segunda vida.

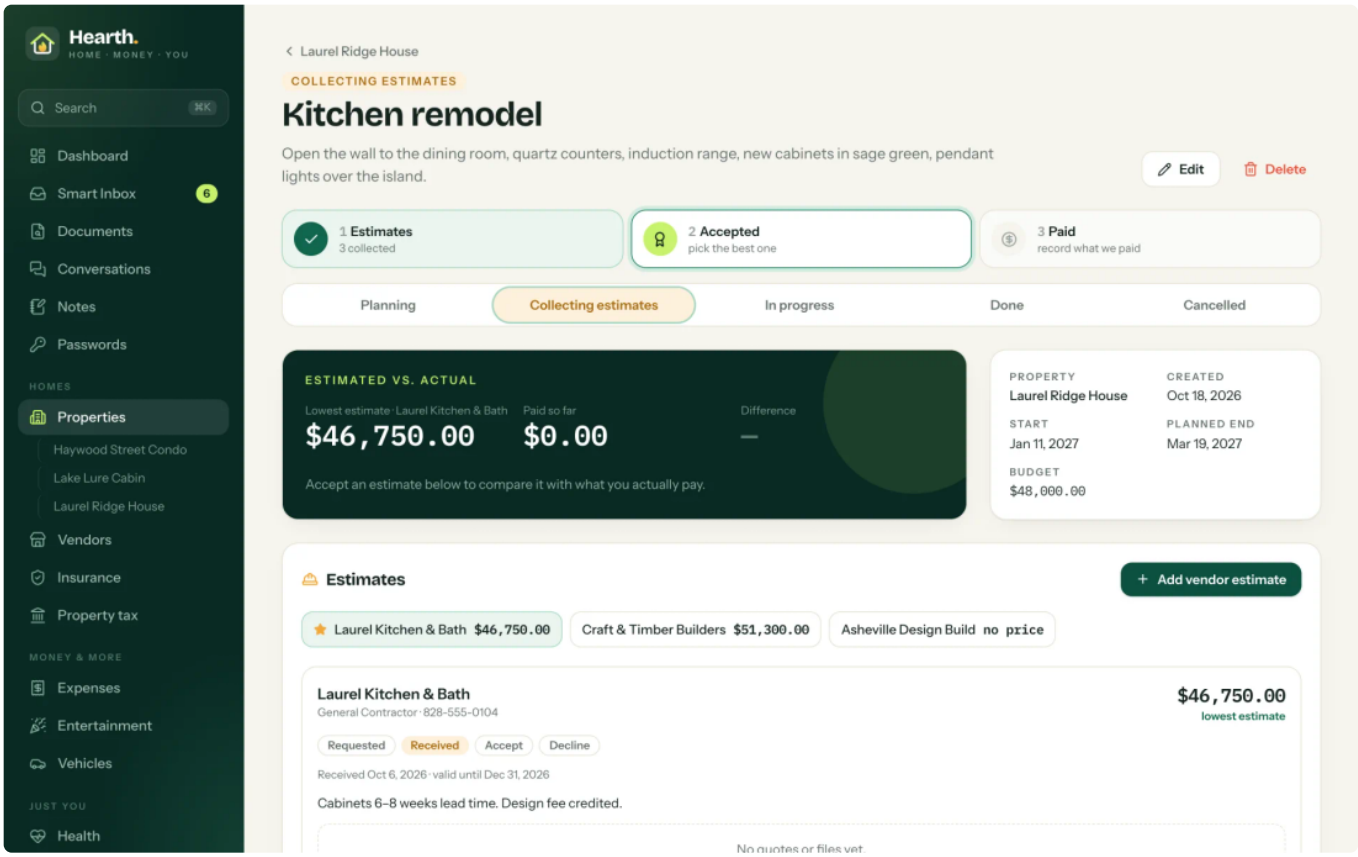
Qué es Hearth, en una pantalla

Hearth gestiona un hogar con varias casas y autos del mismo modo en que una pequeña empresa gestiona sus operaciones. Cada día empieza en el panel principal: el tiempo (con una indicación de los días con buena luz para las fotos de los anuncios), los proyectos abiertos, los gastos del año y todo lo que espera una decisión.



El panel principal de un hogar ficticio de demostración: el tiempo con los días de buena luz para fotos, los proyectos abiertos y los gastos del año.

Detrás del panel están las piezas que antes eran herramientas o carpetas separadas. Las propiedades vienen con su asociación de propietarios (HOA), los recibos del impuesto sobre la propiedad y sus plazos, y los proyectos, cada uno con varios presupuestos de contratistas. Los vehículos vienen con sus contratos de leasing; los seguros, con sus vigencias, sus planes de pago y una línea de tiempo de cada póliza; y el seguro médico, con sus facturas. Los proveedores tienen a su gente y sus números de teléfono, las conversaciones reúnen correos y llamadas grabadas, y los gastos recurrentes se registran solos. Las notas, las contraseñas y un calendario completo cierran el cuadro.

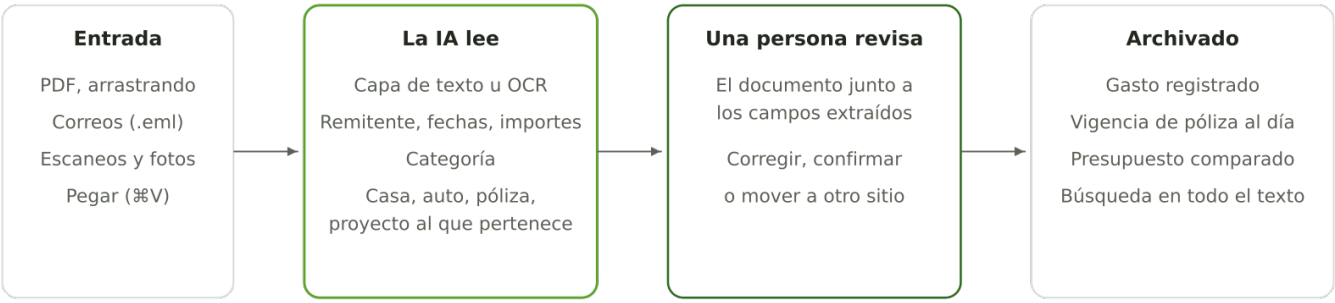


Un proyecto que reúne presupuestos: tres contratistas lado a lado, con el más bajo marcado (datos de demostración).

Nada de esto es exótico. Precisamente de eso se trata. Cada una de estas funciones existe en algún producto comercial. Lo que no existe es el producto que tenga exactamente estas funciones, conectadas exactamente así, para exactamente esta persona.

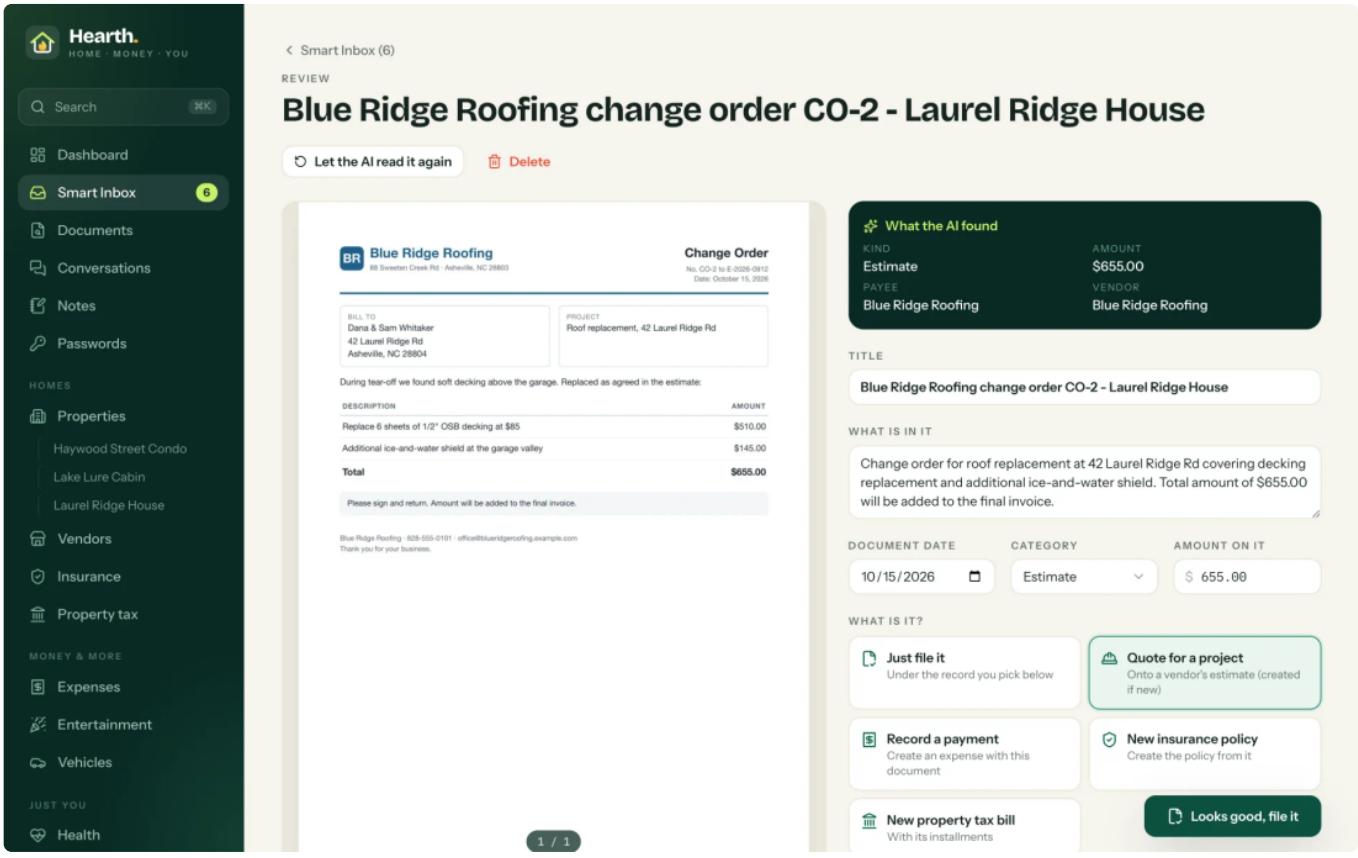
La pila de papel, resuelta: OCR, categorías y archivo

La función que cambió la encimera de la cocina es el flujo de documentos. Un documento llega como PDF, foto, escaneo o correo guardado con sus adjuntos, se suelta sobre la página o se pega con V. Si un PDF trae su propio texto, Hearth lee ese texto; si es un escaneo o una foto, cada página pasa por un modelo de visión que hace de OCR. Después, la IA extrae lo importante: quién lo envía, la fecha, el importe, el vencimiento, el número de póliza o de cuenta. Y luego sugiere dónde va el documento: qué categoría, qué casa o qué auto, qué póliza o qué proyecto.



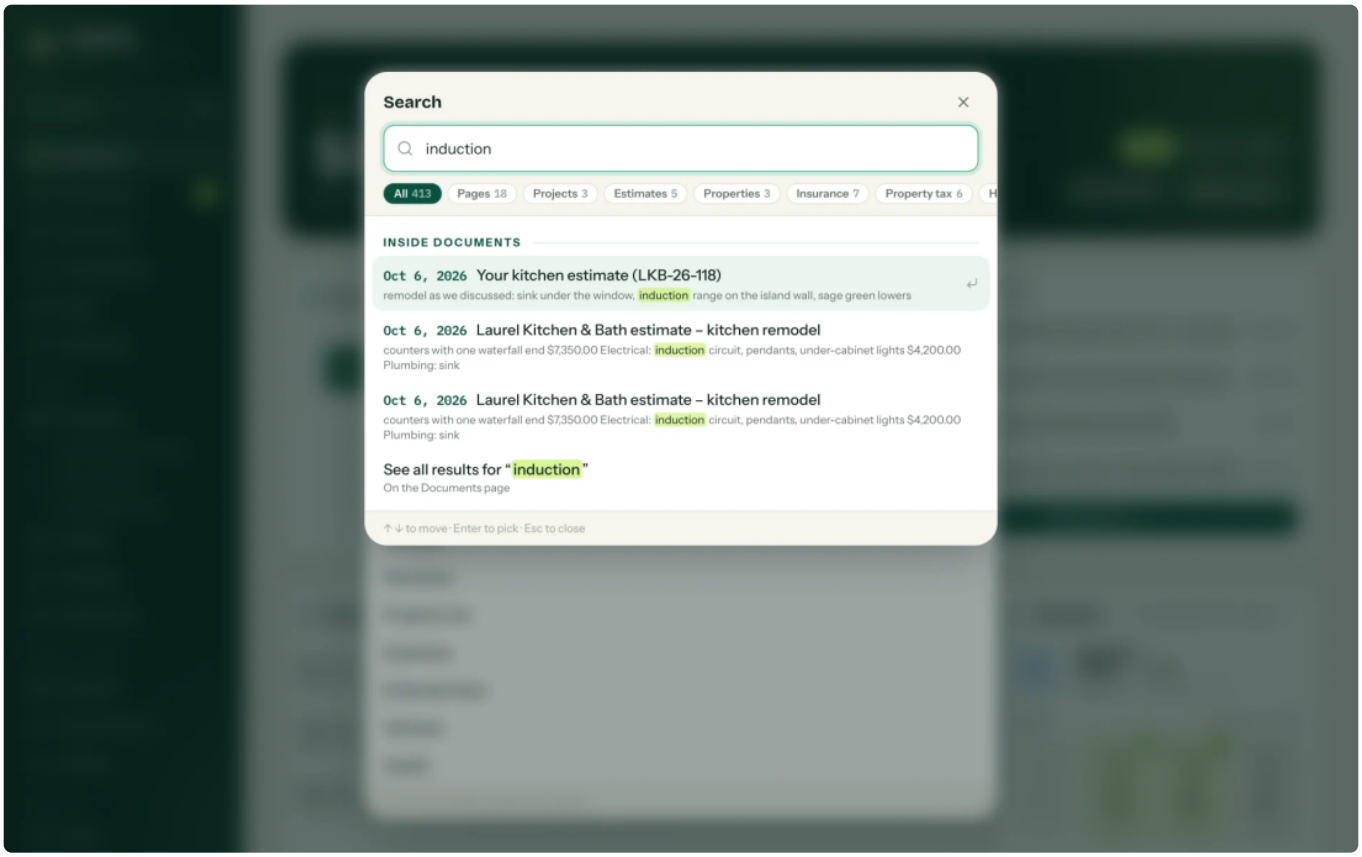
De una pila de papel a un registro archivado y localizable -- con una persona en el circuito

La tercera caja es la que más me importa. La IA nunca decide por su cuenta adónde va algo: todo lo que llega a la Smart Inbox espera hasta que una persona confirme o corrija la sugerencia. (Si sueltas un documento directamente sobre un proyecto o una póliza, ya has dicho adónde pertenece, así que se salta la cola). La pantalla de revisión muestra el documento justo al lado de lo que leyó la IA, de modo que comprobarlo lleva segundos en lugar de minutos.



La Smart Inbox: el documento de un lado; del otro, lo que leyó la IA y dónde sugiere archivarlo.

Una vez archivado, el documento trabaja de verdad. Una factura se convierte en un gasto de la casa correcta, la renovación de un seguro prolonga la póliza correcta y el presupuesto de un contratista aparece junto a los demás presupuestos del mismo proyecto, de modo que la oferta vigente más barata salta a la vista. Cada palabra de cada documento se puede buscar desde cualquier lugar con K, y ese suele ser el momento en que la gente deja de preguntarme para qué querría algo así.

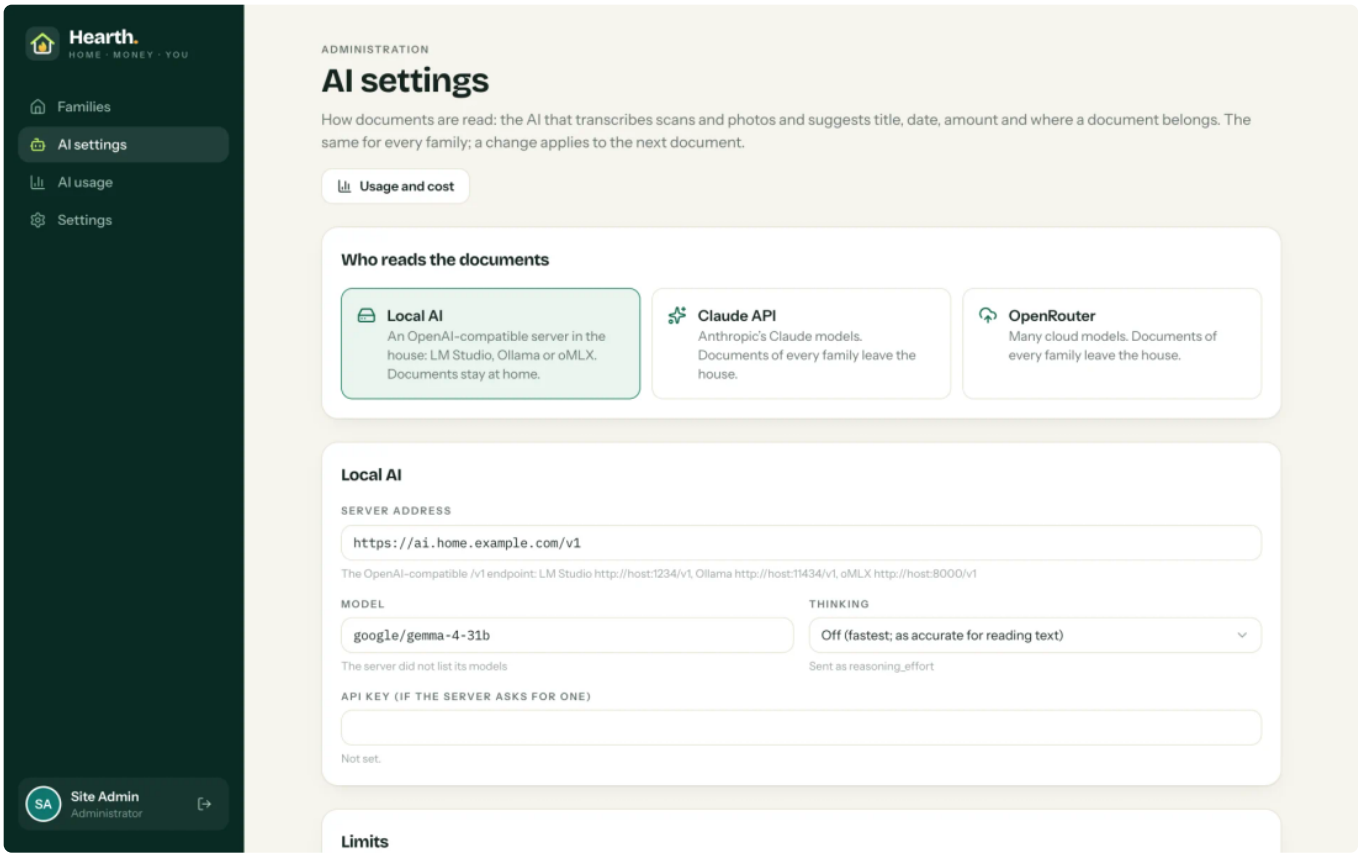


K busca cada palabra dentro de cada documento, esté archivado donde esté.

Las llamadas telefónicas reciben el mismo trato. Tocas el micrófono después de hablar con el techador, cuentas de qué se trató y Hearth lo transcribe en una conversación con ese proveedor. La transcripción corre en un servidor Whisper en un Mac de la casa, así que la grabación nunca sale del hogar.

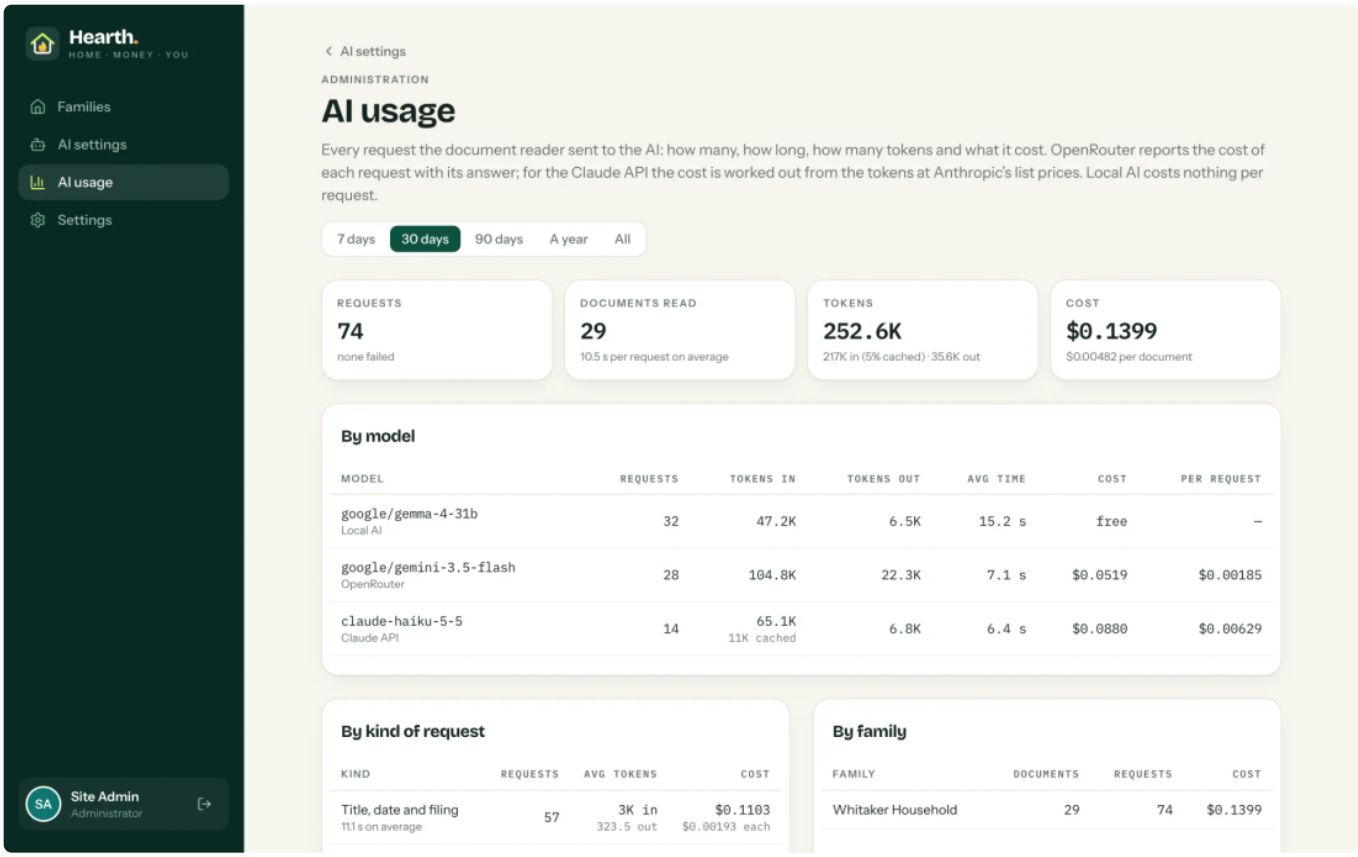
Cualquier proveedor de IA, incluido uno que vive en casa

En el hogar se elige cómo se leen los documentos, en la configuración de administración y sin necesidad de una nueva versión. Hearth habla con tres tipos de proveedores: la Claude API de Anthropic, cualquier modelo de OpenRouter (que reúne los modelos de OpenAI, Google, Anthropic, Mistral y muchos otros detrás de una sola API) o un modelo local en cualquier servidor compatible con OpenAI, como LM Studio u Ollama. Con la opción local, ni una sola página sale de casa -- algo que expliqué en detalle para Delphi en Give Your Delphi App a Brain.



La configuración de IA: Claude, cualquier modelo de OpenRouter o un modelo local que mantiene cada página en casa.

Como la «IA» sin factura es un cuento de hadas, cada solicitud se registra con sus tokens, su tiempo, sus errores y su costo. Los modelos en la nube informan o estiman su precio; el modelo local no cuesta más que la electricidad. El administrador lo ve por modelo, por tipo de solicitud, por familia y por día.



Uso y coste de la IA: solicitudes, tokens y coste por modelo y proveedor -- el modelo local no cuesta nada (datos de demostración).

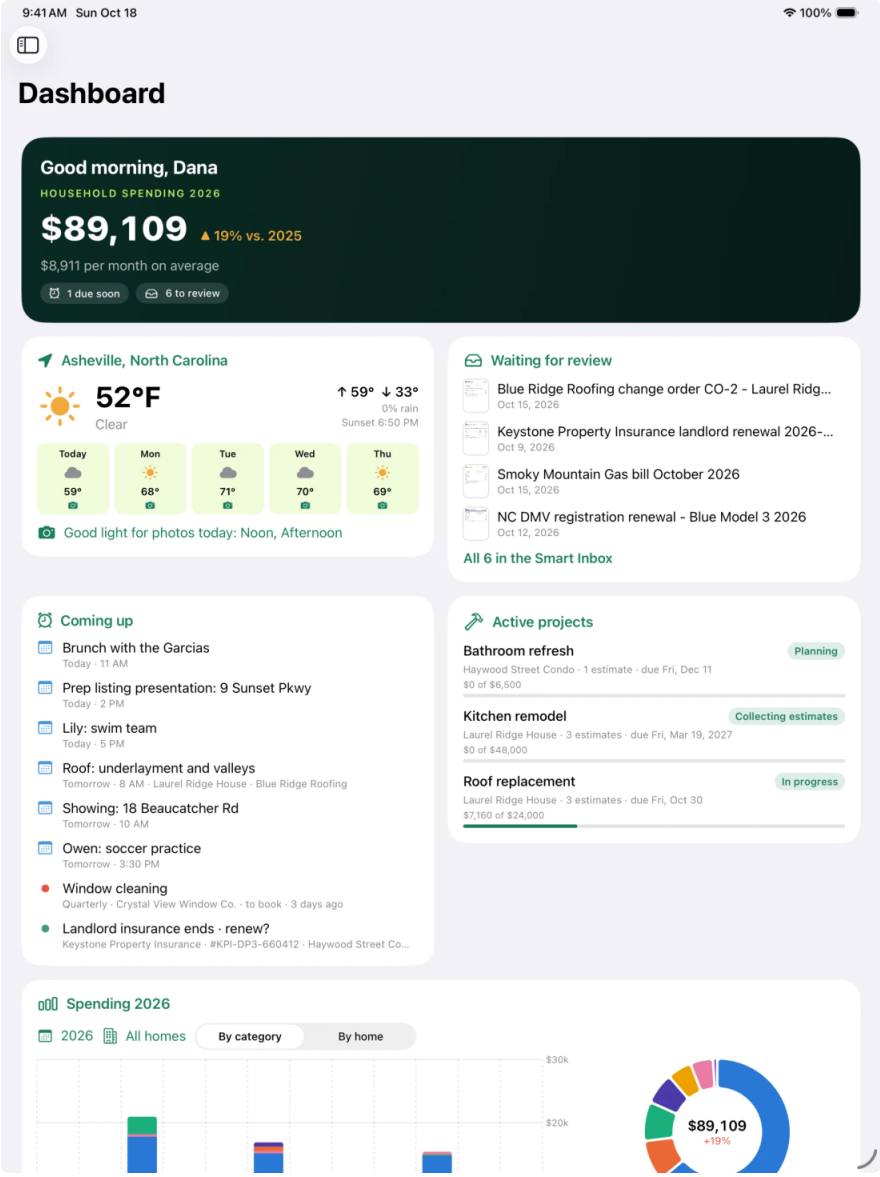
> **Por qué la libertad de proveedor importa para tu producto**
> Si hoy integras IA en un producto, es muy probable que el mejor modelo del año que viene venga de otro proveedor que el de este año. Una aplicación que trata al proveedor como un ajuste, y no como una decisión de arquitectura, puede cambiar de proveedor en una tarde. Además, permite que un cliente con normas estrictas de privacidad lo ejecute todo en local, lo cual ya es un argumento de venta por sí mismo.

Un backend, siete puertas de entrada

Hearth no es una app; es un servidor con varias formas de entrar. La aplicación web lo hace todo. Una app de solo lectura para iPhone y iPad muestra cada una de sus secciones. Una app para Apple TV convierte la pantalla de la sala en un panel familiar. Notes for Mac y Passwords for Mac son sustitutos nativos completos de las apps a las que reemplazaron, y una extensión de Chrome rellena los inicios de sesión en los propios sitios web.

Un servidor, una base de datos, un conjunto de reglas -- y siete formas de entrar

El diagrama parece sencillo porque la parte difícil no se ve: cada puerta de entrada pasa por las mismas reglas en el servidor. Cada app inicia sesión por separado, con un código que se confirma en la web, y cada tipo de app solo puede hacer aquello para lo que fue creada. La extensión de Chrome puede leer y guardar contraseñas, y nada más; si alguien robara su sesión del navegador, no podría abrir ni un solo documento.

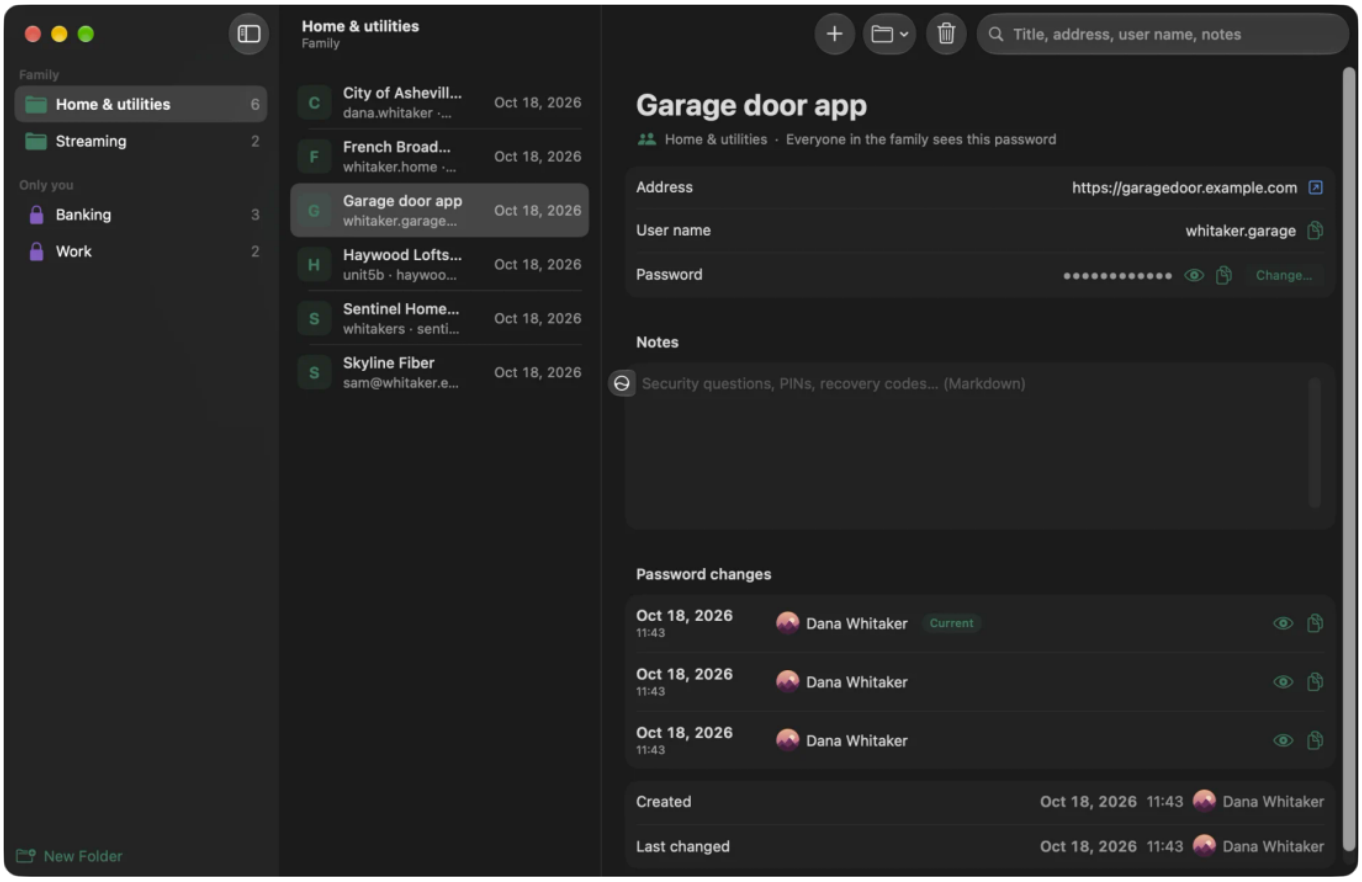


La app para iPad muestra cada sección de Hearth, en solo lectura, con los documentos en Vista Rápida.

La app para Apple TV merece un comentario aparte, porque es la que saca sonrisas. Muestra las citas de hoy, el mes, el tiempo con las horas de buena luz para fotos y los gastos del año, y funciona como un pase de diapositivas que mantiene alejado el salvapantallas. En un hogar, eso es un panel familiar. En una oficina, una sala de exposición o una sala de espera, exactamente la misma idea es **cartelería digital**, alimentada en vivo con tus propios datos en lugar de con una memoria USB que alguien actualiza una vez por trimestre.

La app para Apple TV: el día de hoy, el calendario, el tiempo y los gastos -- un panel familiar, o cartelería digital en una oficina.

En el Mac, las dos apps nativas reemplazaron a las apps de notas y de contraseñas que usábamos antes. Las contraseñas quedan ocultas tras Touch ID y se vuelven a bloquear a los cinco minutos, y la extensión de Chrome ofrece guardar o actualizar un inicio de sesión justo después de que entras en algún sitio.

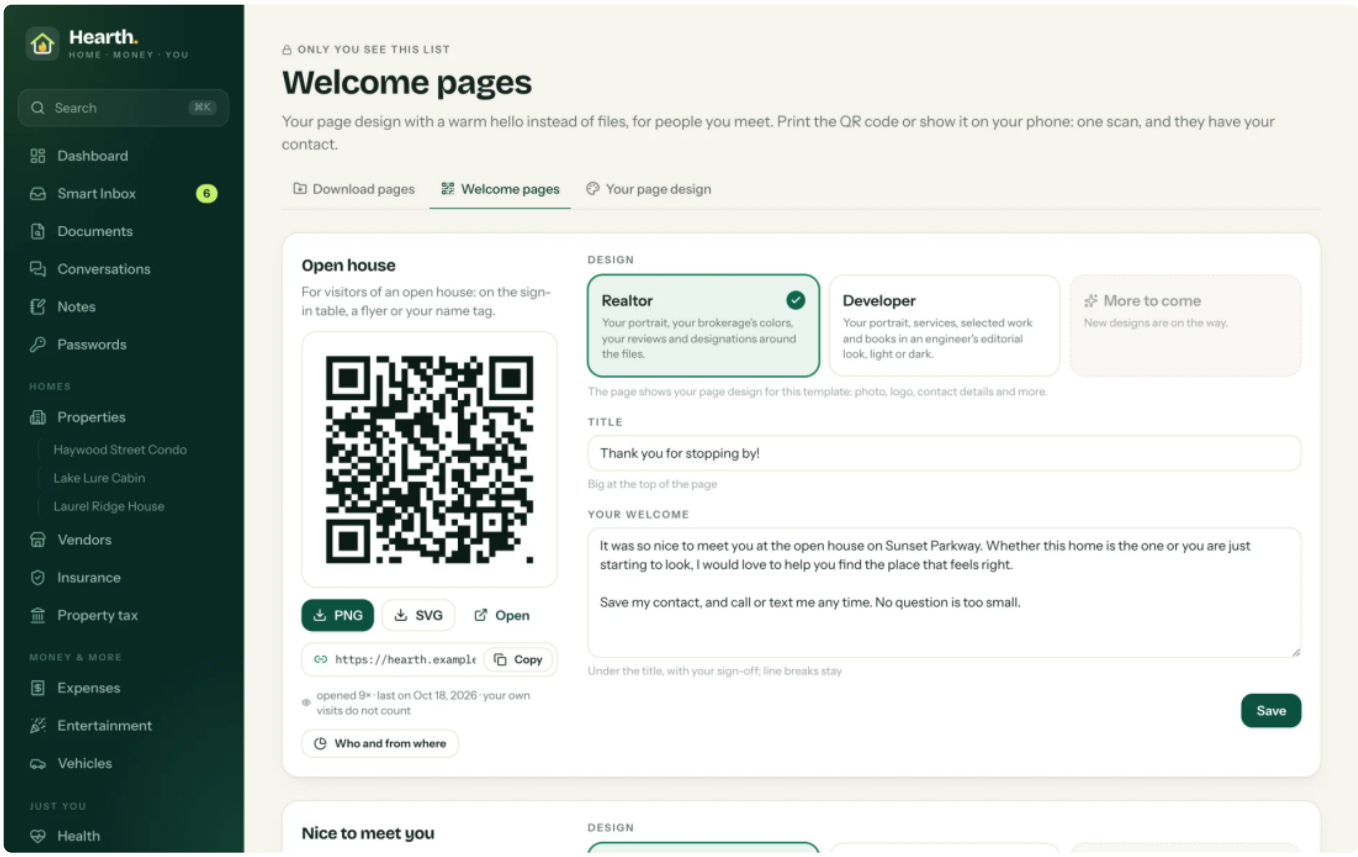


Passwords for Mac: bloqueado tras Touch ID, con el historial de contraseñas y un generador.

Hecho para una agente inmobiliaria, no para un millón de usuarios

Aquí es donde el software a medida se gana el sueldo. Un producto genérico tiene que servir a todo el mundo, así que no le sirve exactamente a nadie. Hearth sabe que su dueña vende casas, así que tiene cosas que ninguna app doméstica incluiría jamás.

Las páginas de bienvenida son un buen ejemplo. Una agente inmobiliaria puede crear una página para un cliente o para una jornada de puertas abiertas -- a partir de una plantilla, con un código QR para imprimir -- y Hearth registra cuándo se abrió, aproximadamente desde dónde y con qué navegador. La tarjeta del tiempo marca los días con buena luz para las fotos de los anuncios. Nada de esto llevó mucho tiempo construirlo. Cada una de estas cosas sería una solicitud de función en el backlog de un gran producto, votada por miles de usuarios que no la necesitan.



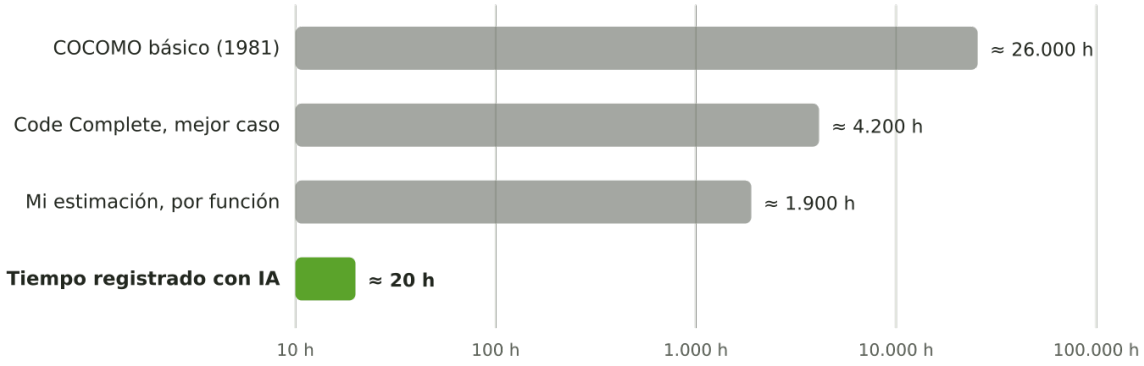
Una página de bienvenida con código QR, hecha a partir de una plantilla en un minuto (datos de demostración).

> **El mismo patrón, en una empresa**
> Cambia «agente inmobiliaria» por «corredor de seguros», «administrador de fincas», «laboratorio dental» o «fabricante de maquinaria», y la historia no cambia. Una empresa que compra un gran CRM paga por cientos de funciones que nunca usa y luego vuelve a pagar -- en consultores o en soluciones improvisadas -- por las cinco que de verdad necesita. Un sistema a medida empieza por esas cinco.

¿Cuánto habría llevado sin IA?

Doce días es el tiempo de calendario entre el primer prompt, el 7 de octubre, y la última versión, el 18 de octubre. Los registros de mis sesiones de Claude Code muestran unas **20 horas** de trabajo activo con la IA en ese periodo. Esa cifra solo cuenta el tiempo en que una sesión estaba ocupada, sin las pausas de más de veinte minutos, así que no incluye las pruebas en el iPad ni el rato pensando en el porche.

El resultado son **58.588 líneas no vacías** de TypeScript, Swift, SQL y JavaScript, 52 pull requests fusionados y 48 versiones etiquetadas. Las líneas de código son una mala medida del valor -- Edsger Dijkstra sostuvo, en una frase célebre, que deberían contabilizarse como «líneas gastadas» y no como «líneas producidas» --, pero son la única medida de tamaño que aceptan los modelos de estimación clásicos, así que preguntémosles a ellos.



Esfuerzo estimado para construir Hearth a mano, frente al tiempo registrado con IA (horas, escala logarítmica)

Que el gráfico necesite una escala logarítmica ya dice algo por sí solo. Este es el origen de cada barra.

1. **COCOMO básico.** El Constructive Cost Model de Barry Boehm, de 1981, estima el esfuerzo en 2,4 × KLOC^1,05 personas-mes para un proyecto «orgánico». Para 58,6 mil líneas, eso da unas 172 personas-mes, o aproximadamente 26.000 horas con las 152 horas mensuales del modelo. El modelo es antiguo y presupone un mundo sin frameworks modernos, así que lo tomo como techo.
2. **Code Complete, mejor caso.** *Code Complete* (2.ª edición), de Steve McConnell, da entre 2.000 y 25.000 líneas por año-persona en proyectos de 10.000 líneas, y entre 1.000 y 20.000 en proyectos de 100.000 líneas (tabla reproducida aquí). Tomando el extremo más optimista, 25.000 líneas al año, Hearth equivale a unos 2,3 años-persona, aproximadamente 4.200 horas a razón de 1.800 horas laborables al año.
3. **Mi propia estimación.** La hice como hago las de los proyectos de clientes: función por función, para un desarrollador experimentado, con frameworks modernos y sin IA. El modelo de datos y el inicio de sesión con passkeys y aislamiento entre familias: 25 días. El flujo de documentos con los proveedores de IA: 25. Los módulos del hogar, de las propiedades al seguro médico: 55. Calendario, panel principal, búsqueda, notas, bóveda de contraseñas y páginas de bienvenida: 44. Pruebas, revisión de seguridad y despliegue: 17. La API para las apps, la app para iPhone y iPad, la app para Apple TV, las dos apps para Mac y la extensión de Chrome: 69. En total, 235 días laborables, o algo menos de 1.900 horas -- y, a propósito, es la más optimista de las tres.

Incluso frente a mi propia estimación optimista, la diferencia es de unos dos órdenes de magnitud. Sé cómo suena eso, así que la siguiente sección es la parte en la que discuto conmigo mismo.

> **Dónde no se sostiene esta comparación**
> Las 20 horas son tiempo de sesión, no la contabilidad completa de mi día: pensar, probar en los dispositivos y decidir qué construir ocurrió alrededor de ellas. La IA escribió mucho más código del que yo habría tecleado, y más líneas no significan más valor. Un proyecto desde cero, con una sola usuaria exigente y sin datos heredados, es el mejor caso para la IA -- una migración con veinte años de reglas de negocio escondidas en una base de código de Delphi 7 es otra historia, como describí en Adapt or Disappear.
>
> Y la investigación no es unánime: en un estudio aleatorizado de METR, desarrolladores experimentados de código abierto fueron un 19 % *más lentos* con las herramientas de IA de principios de 2025 en sus propias bases de código maduras (la propia METR señala que esos resultados están desactualizados), y el informe DORA de 2024 encontró que la adopción de IA iba de la mano de una menor estabilidad en las entregas. Mi conclusión no es «la IA hace a todo el mundo 100 veces más rápido». Es más acotada: un desarrollador experimentado que especifica con precisión, lo revisa todo e insiste en las pruebas puede construir hoy en días lo que antes llevaba un año.

Cómo evito que esto se convierta en «vibe coding» -- las pruebas, la revisión de seguridad, la barrera antes de cada versión, las comprobaciones de aislamiento -- es el tema de mi próximo artículo, No es vibe coding.

¿Comprar o construir? La respuesta honesta es «depende»

Hay un cálculo sencillo que muchas empresas nunca hacen. La edición empresarial de un CRM de marca conocida cuesta, a precio de lista, aproximadamente **entre $165 y $175 por usuario al mes**, según a qué sitio de seguimiento de precios le preguntes (Capterra, Redress Compliance). Para diez usuarios, eso son unos $20.000 al año solo en licencias, antes de complementos, integraciones y el consultor que lo configura.

Licencias acumuladas a precio de lista para diez usuarios de CRM a $165 por usuario al mes (sin complementos ni implantación)

Aproximadamente $100.000 en cinco años, por un software que seguirá sin saber a qué se dedica realmente tu empresa. En mi opinión, con ese presupuesto hoy se paga un sistema a medida, y sobra -- uno que es tuyo, que habla tu idioma y que crece con el negocio y no con la hoja de ruta del proveedor.

Dicho esto, una suscripción compra cosas reales que es fácil olvidar: alojamiento, actualizaciones, trabajo de cumplimiento normativo, soporte a las 2 de la madrugada e integraciones con otros mil productos. El software a medida necesita mantenimiento, copias de seguridad y alguien que responda cuando se rompe. Si tu proceso es estándar y hay un producto que encaja, compra el producto -- lo digo en serio. Y si te atrae la idea de que el sistema sea tuyo pero no quieres empezar de cero, hay CRM de código abierto excelentes, como Odoo Community, SuiteCRM, EspoCRM y Twenty, que se pueden personalizar y alojar donde quieras.

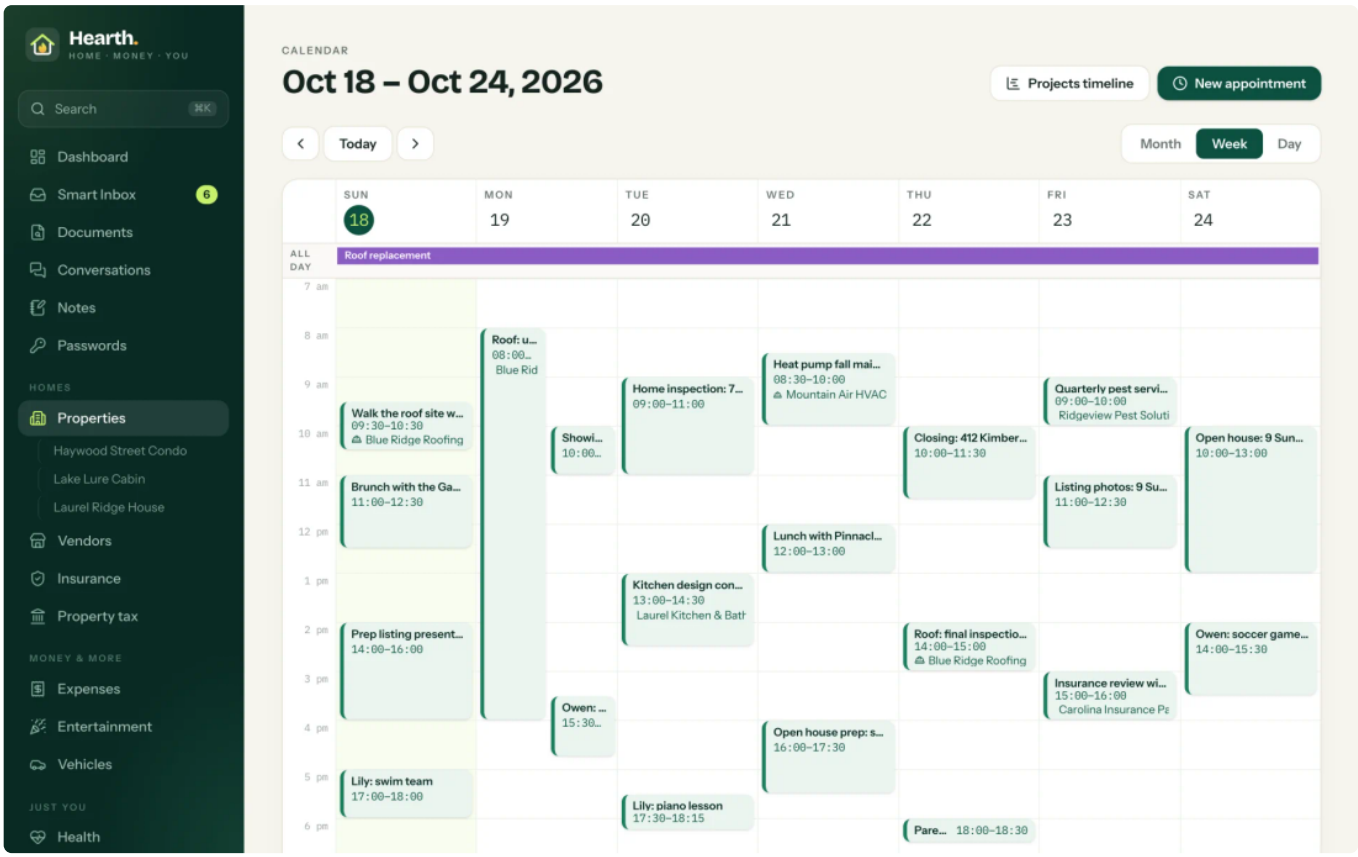
Cuándo construir

Construye cuando el proceso *es* tu ventaja: cuando tu gente se pasa el día sorteando una herramienta, cuando los datos viven en cinco sitios distintos o cuando la única función que necesitas es justo la que ningún proveedor va a priorizar jamás. Compra cuando el proceso es un commodity -- nóminas, correo electrónico, contabilidad -- y el producto ya lo resuelve bien.

Qué significa esto para tu aplicación Delphi

La mayoría de mis clientes no llegan con una hoja en blanco. Llegan con una aplicación Delphi que lleva quince o veinticinco años sosteniendo su negocio. Funciona, los usuarios la adoran y, poco a poco, se está convirtiendo en la razón por la que la empresa no puede hacer algo nuevo: sin acceso web, sin teléfono, sin tableta, sin IA, y con una cantera de desarrolladores que no deja de encogerse.

Hearth es una imagen de lo que puede llegar a ser una aplicación así, y hay varios caminos honestos para llegar ahí. Puedes conservar el código Delphi y ponerle delante una API REST -- con TMS XData o con nada más que WebBroker -- y luego construir sobre ella interfaces web, móviles y para TV. Puedes generar las páginas web desde el propio Delphi con WebStencils. O puedes llevarlo todo a un stack web moderno, el camino que describí en mi serie Delphi a Next.js. La respuesta correcta depende de tu código, tu equipo y tus usuarios, y prefiero decirte qué camino encaja antes que venderte el que más me gusta.



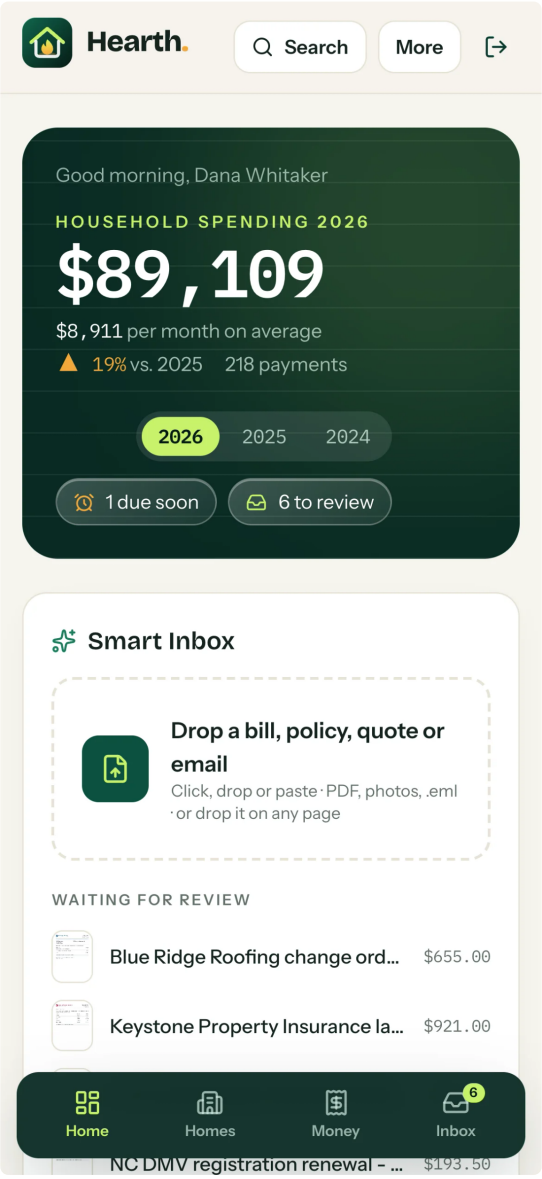
El calendario, con citas que puedes arrastrar, redimensionar y añadir con un clic.

Modernizar un sitio web sigue la misma regla. Tu marca no es una plantilla, así que mantengo tus colores, tu tipografía y tu logotipo, y modernizo lo que hay debajo: velocidad, móviles, accesibilidad, buscadores y un flujo de contenidos que no necesite a un desarrollador para cada coma. Si ya tienes un sistema de diseño, trabajamos dentro de él.

Equipos y desarrolladores en solitario

Trabajo de las dos maneras, y las dos son habituales. Algunos clientes son un único desarrollador que lleva años cargando con un producto y necesita un socio que ya haya hecho esto antes -- alguien que revise el plan, que se encargue de las partes que no encajan con sus habilidades o que programe con él el primer módulo. Otros son equipos que quieren el stack moderno y un proceso asistido por IA que funcione, introducido en su propio repositorio, con su propia gente haciendo el trabajo y yo revisando a su lado. En ambos casos, el trabajo ocurre en un repositorio que puedes ver desde el primer día, en versiones cortas que puedes ejecutar.

La forma de empezar es siempre la misma y, como dice mi página de servicios, no cuesta nada: una conversación de treinta a sesenta minutos sobre el sistema y lo que duele. Si ya hay código, el siguiente paso es una evaluación de alcance cerrado -- lo leo, lo compilo, lo ejecuto, y recibes un plan por escrito y una estimación realista que son tuyos, sigamos o no. Puedes ver más sistemas que he entregado en Work.



La app web en un teléfono: el mismo sistema, adaptado a la pantalla pequeña.

En resumen

Hearth empezó como un regalo de cumpleaños y terminó siendo el mejor argumento que puedo dar a favor del software a medida. En doce días, un desarrollador experimentado con IA construyó lo que los manuales sitúan entre uno y varios años-persona: una IA que lee y archiva documentos con cualquier proveedor, passkeys y una bóveda cifrada, y apps nativas para iPhone, iPad, Apple TV y Mac, además de Chrome. La IA no tomó las decisiones; hizo que llevarlas a cabo saliera barato.

> La pregunta ya no es si puedes permitirte un software a medida. Es si puedes permitirte seguir adaptando tu negocio al software de otros.

Si tienes una aplicación Delphi que merece un futuro, o un proceso al que no le encaja ningún producto, me encantaría conocerlo. Hablemos.

INDEPENDENT · AD-FREE · READER-SUPPORTED

Free to read. Not free to make.

Every tutorial, article, and line of example code on Holger's Code is published without a paywall and without ads. This page is about what keeps it that way.

Professional work, published free

The content here is held to the same standard as professional client work: every tutorial is built as a real project, tested against current versions, documented step by step, and revisited when the frameworks move on. A single in-depth tutorial routinely takes days — and that's before the bills behind it: servers, storage, build infrastructure, licenses, and the hardware everything is verified on.

If an article or tutorial here has saved you an afternoon of debugging, taught you a technique, or helped you ship, it created real value for you. Supporting it isn't charity — it's paying for professional work you found useful, at whatever price you choose. And if you can't, keep reading anyway. That's what the free part means.

Ways to support

KO-FI

Buy me a coffee

A one-time tip — no account, no subscription, any amount.

BOOKS & COURSES

Buy the paid work

The paid tier of everything published free here — and you get something lasting in return.

SPREAD THE WORD

Share an article

Send a tutorial to a colleague, cite an article, subscribe to the RSS feed.

Nothing here goes behind a paywall, whether you chip in or not. But if something on this site earned its keep today, this is where to say so.

Support on Ko-fi —