NEWS · OCTOBER 19, 2026

Desenvolva, não compre: software sob medida em 12 dias no lugar de mais uma assinatura SaaS

A casa de uma corretora de imóveis funcionava à base de planilhas, papel e um punhado de assinaturas -- então construí para ela um sistema próprio em doze dias, com arquivamento de documentos por IA, passkeys, um cofre criptografado e apps para iPhone, iPad, Apple TV, Mac e Chrome, e aqui está o que isso significa para a sua aplicação Delphi e para a sua próxima decisão entre comprar ou desenvolver.

By Dr. Holger Flick

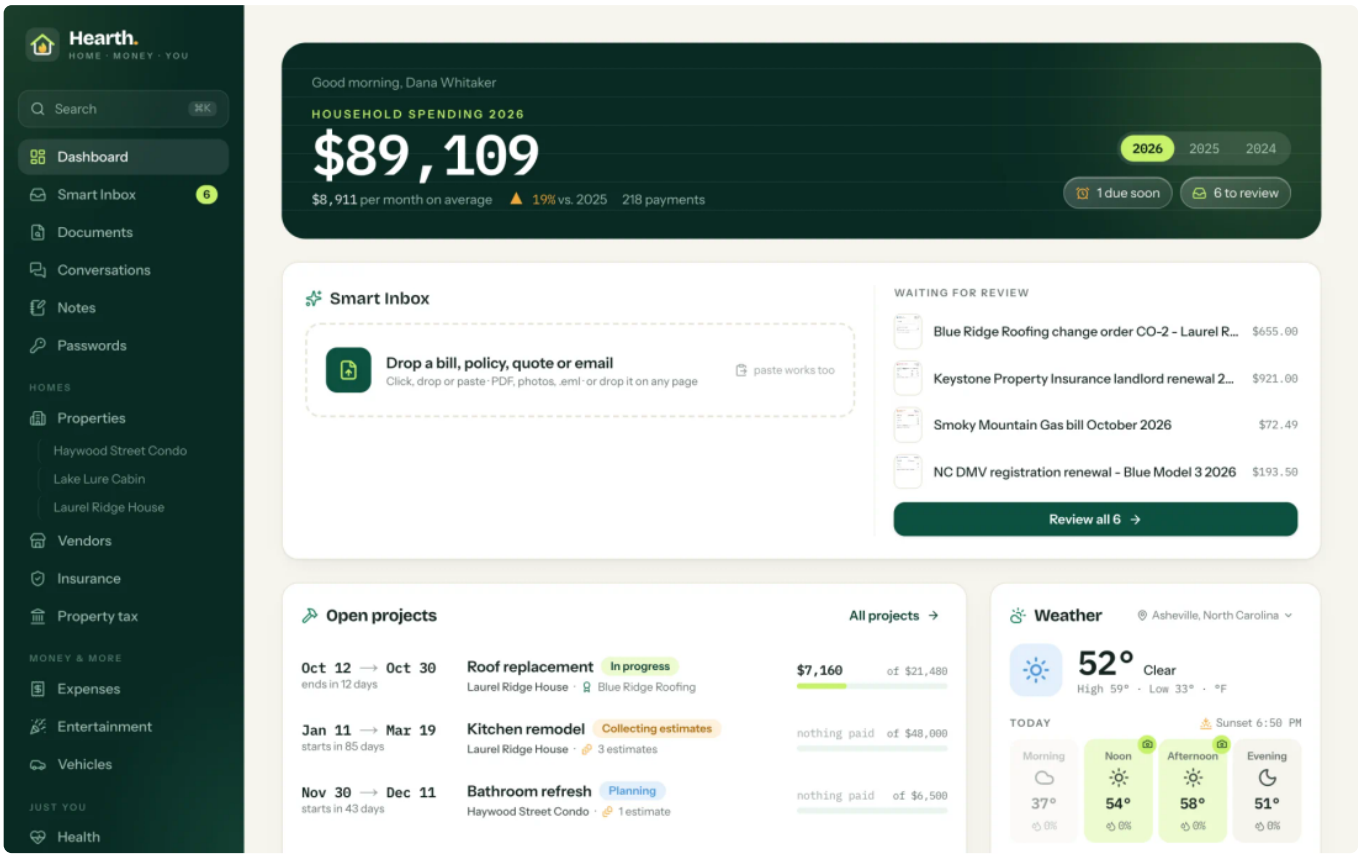
Minha esposa é corretora de imóveis. Durante anos, a nossa casa funcionou do jeito que a maioria das casas -- e, sinceramente, a maioria das pequenas empresas -- funciona: uma planilha para as apólices de seguro, uma pasta de PDFs que ninguém encontrava, um app de calendário, um app de notas, um gerenciador de senhas e uma pilha crescente de papel no balcão da cozinha, que todo mundo contornava. Cada uma dessas ferramentas era boa no que fazia. Nenhuma sabia das outras, e nenhuma sabia nada sobre o trabalho *dela*.

Por isso, no aniversário dela, construí um único sistema no lugar de tudo isso. Ele se chama **Hearth**. Ele lê cada conta, apólice e recibo com IA, arquiva tudo na casa, no carro ou no projeto certo, cuida do calendário, das notas e das senhas, e mostra tudo isso na web, no iPhone e no iPad dela, na Apple TV da sala, no Mac e dentro do Chrome. Do primeiro prompt à última versão, levou doze dias.

Este post é o tour, e também é um argumento. Na minha opinião, a pergunta "qual produto devemos comprar?" hoje, com muita frequência, é a pergunta errada para começar. Para um número crescente de empresas, a pergunta melhor é "como seria um software feito exatamente para nós?" -- porque o preço dessa resposta despencou. Vou mostrar o que foi construído, quanto tempo teria levado sem IA, onde o meu argumento se sustenta e onde não se sustenta, e o que tudo isso significa se você tem uma aplicação desktop em Delphi que merece uma segunda vida.

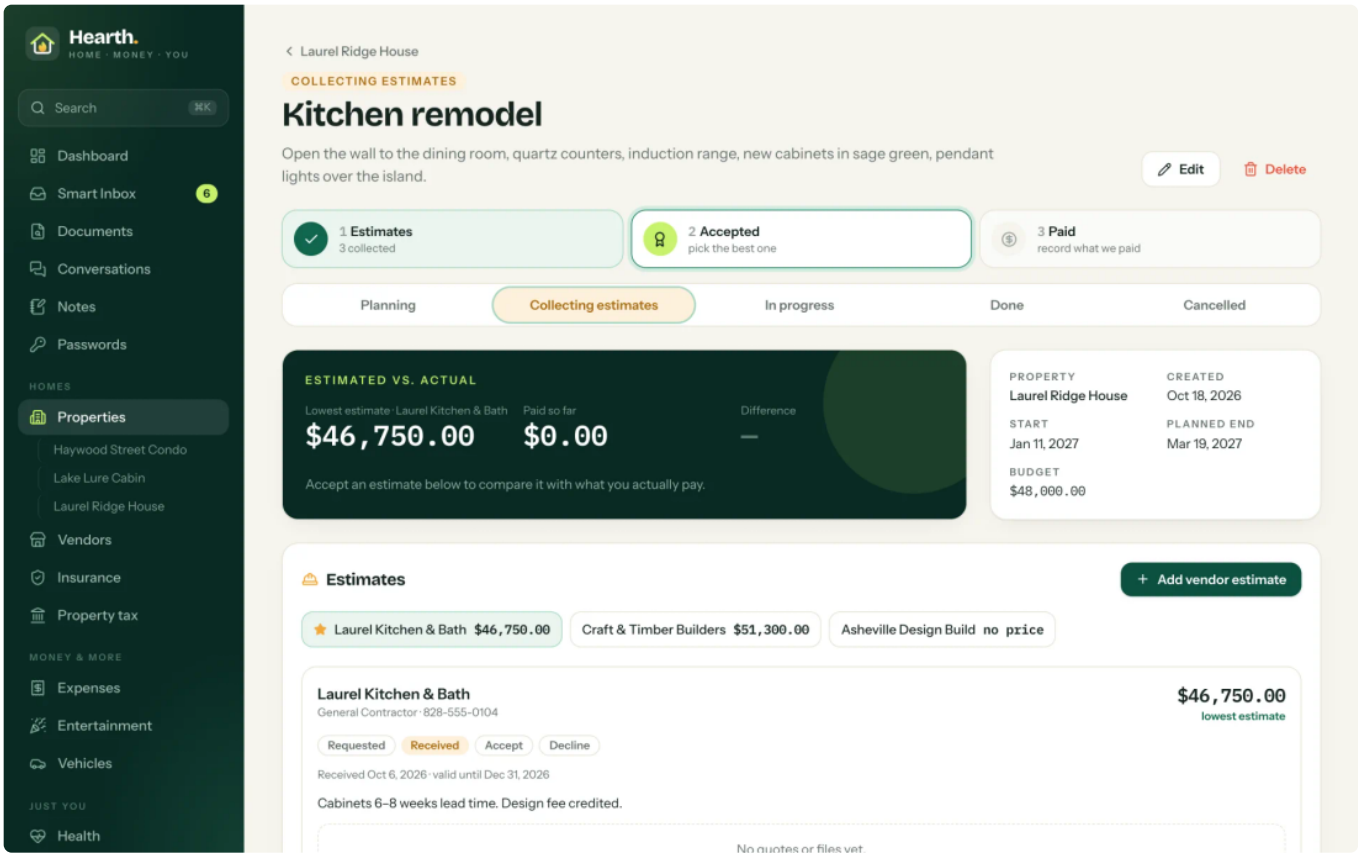
O que é o Hearth, em uma tela

O Hearth administra uma casa com vários imóveis e carros do jeito que uma pequena empresa administra as suas operações. O painel é onde todo dia começa: o clima (com um aviso nos dias de boa luz para fotos de anúncio), os projetos em aberto, os gastos do ano e tudo o que está esperando uma decisão.



O painel de uma casa fictícia de demonstração: o clima com os dias de boa luz para fotos, os projetos em aberto e os gastos do ano.

Por trás do painel ficam as partes que antes eram ferramentas separadas ou pastas separadas. Os imóveis vêm com a associação de proprietários (HOA), as contas de imposto predial e as suas parcelas, e os projetos, cada um com vários orçamentos de prestadores de serviço. Os veículos vêm com os contratos de leasing, os seguros vêm com vigências, planos de pagamento e uma linha do tempo de cada apólice, e o plano de saúde vem com as suas contas. Os fornecedores têm os seus contatos e telefones, as conversas reúnem e-mails e ligações gravadas, e as despesas recorrentes se lançam sozinhas. Notas, senhas e um calendário completo fecham o quadro.

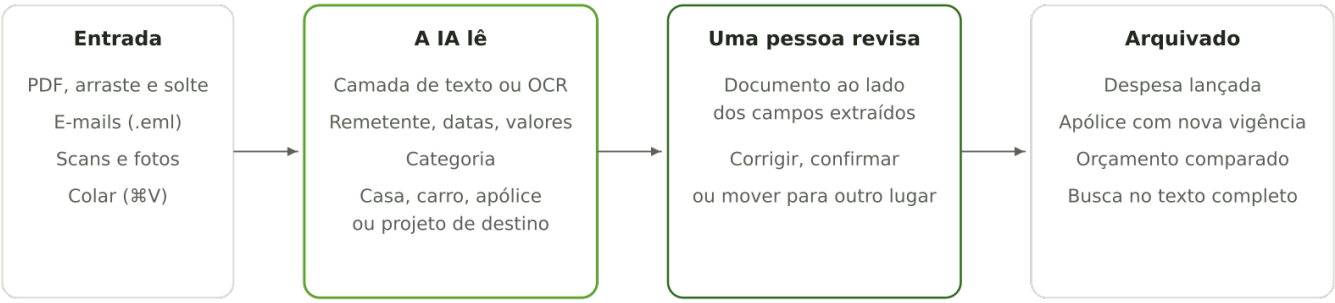


Um projeto reunindo orçamentos: três prestadores lado a lado, com o mais barato destacado (dados de demonstração).

Nada disso é exótico. E é exatamente esse o ponto. Cada uma dessas funções existe em algum produto comercial. O que não existe é o único produto que tem exatamente essas funções, conectadas exatamente desse jeito, para exatamente essa pessoa.

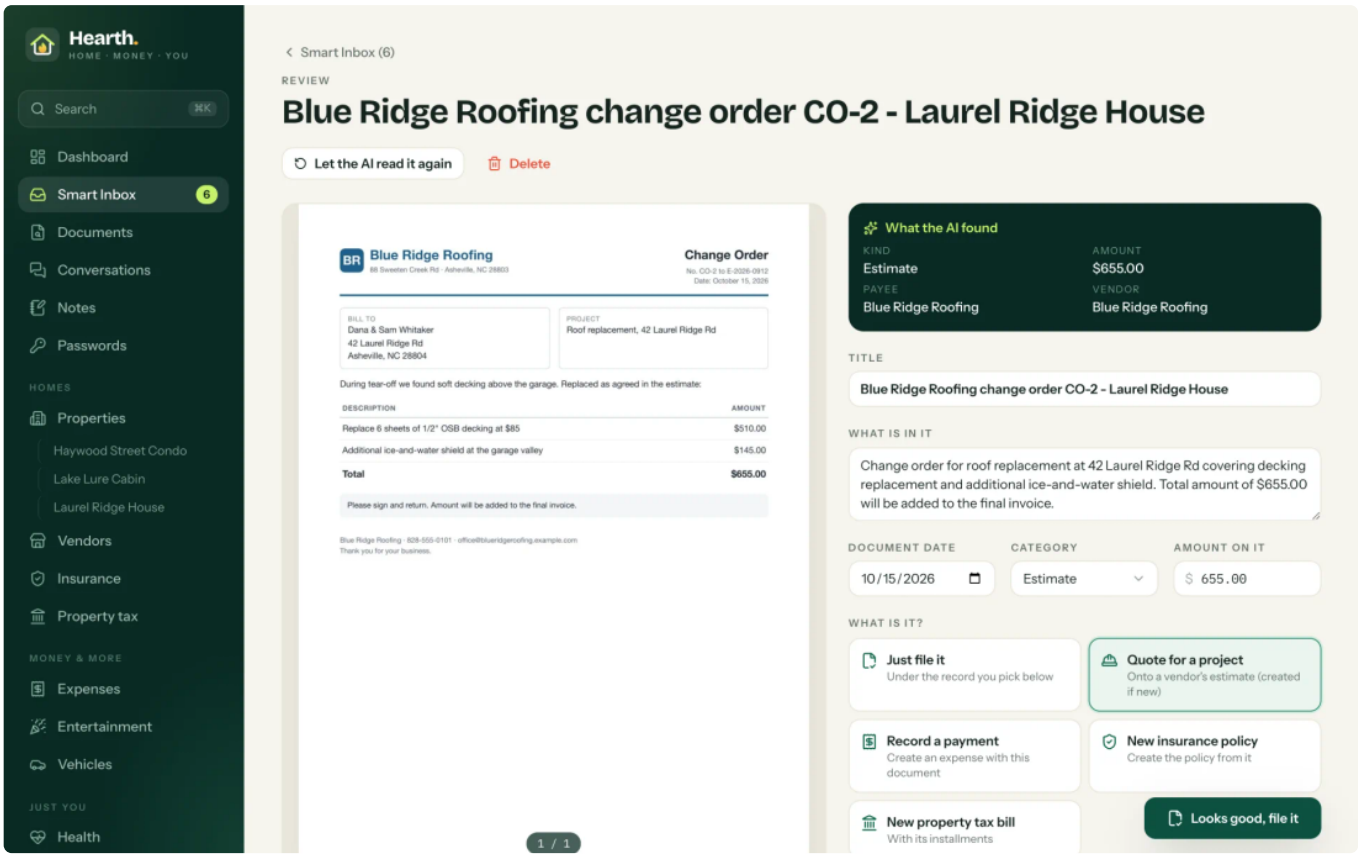
A pilha de papel, resolvida: OCR, categorias e arquivamento

A função que mudou o balcão da cozinha é o pipeline de documentos. Um documento chega como PDF, foto, digitalização ou e-mail salvo com os seus anexos, arrastado para a página ou colado com V. Quando um PDF já traz o próprio texto, o Hearth lê esse texto; quando é uma digitalização ou uma foto, cada página passa por um modelo de visão como OCR. Em seguida, a IA extrai o que importa: quem enviou, a data, o valor, o vencimento, o número da apólice ou da conta. Depois, ela sugere onde o documento deve ficar: qual categoria, qual casa ou carro, qual apólice ou projeto.



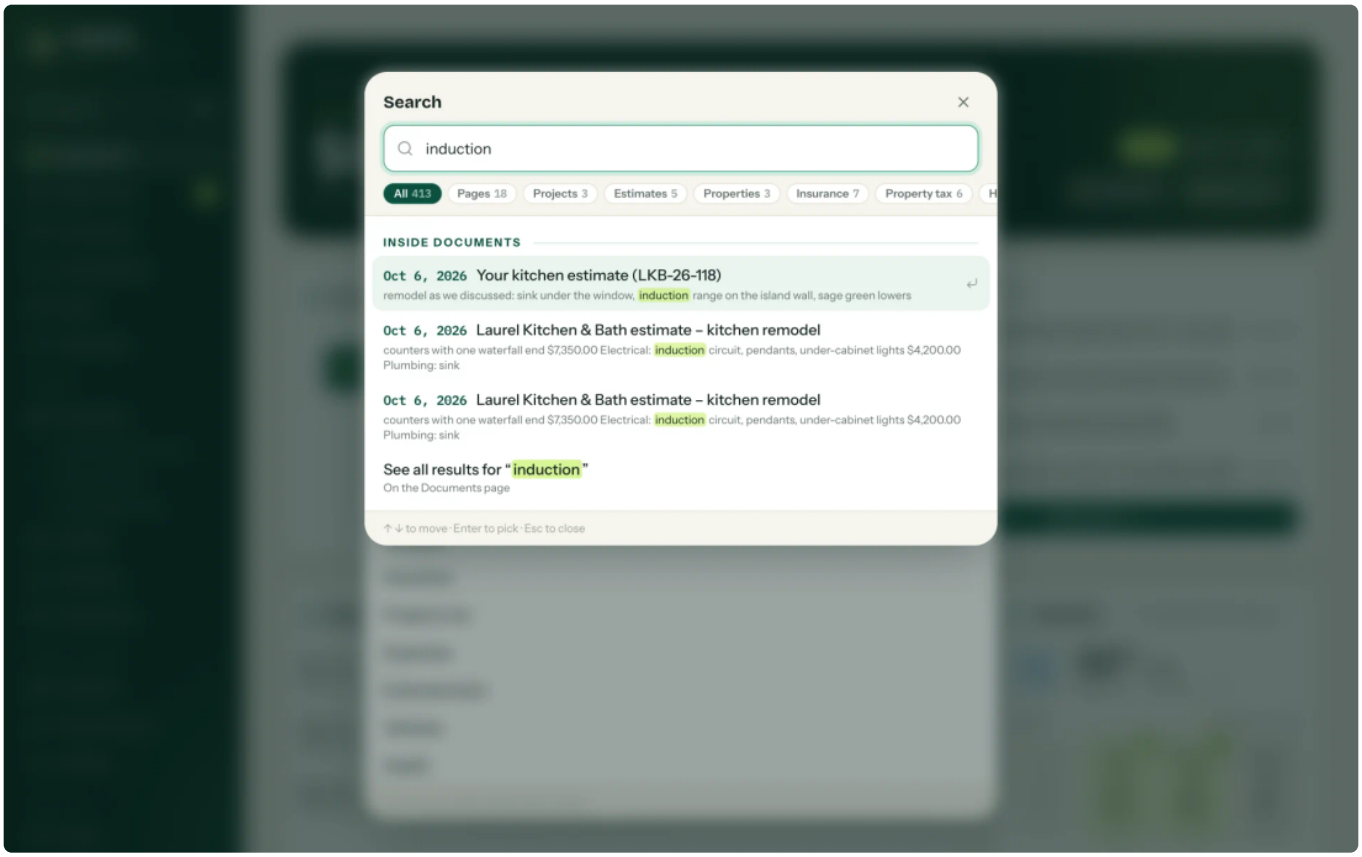
De uma pilha de papel a um registro arquivado e pesquisável -- com uma pessoa no circuito

A terceira caixa é a que mais me importa. A IA nunca decide sozinha para onde algo vai: tudo o que chega à Smart Inbox espera até que uma pessoa confirme ou corrija a sugestão. (Se você soltar um documento direto em um projeto ou em uma apólice, já disse onde ele deve ficar, então ele pula a fila.) A tela de revisão mostra o documento bem ao lado do que a IA leu, então conferir leva segundos em vez de minutos.



A Smart Inbox: de um lado o documento, do outro o que a IA leu e onde ela sugere arquivá-lo.

Depois de arquivado, o documento trabalha de verdade. Uma conta vira uma despesa no imóvel certo, uma renovação de seguro estende a apólice certa, e o orçamento de um prestador cai ao lado dos outros orçamentos do mesmo projeto, de modo que a oferta atual mais barata salta aos olhos. Cada palavra dentro de cada documento pode ser pesquisada de qualquer lugar com K, e é nesse momento que as pessoas costumam parar de me perguntar por que iriam querer isso.

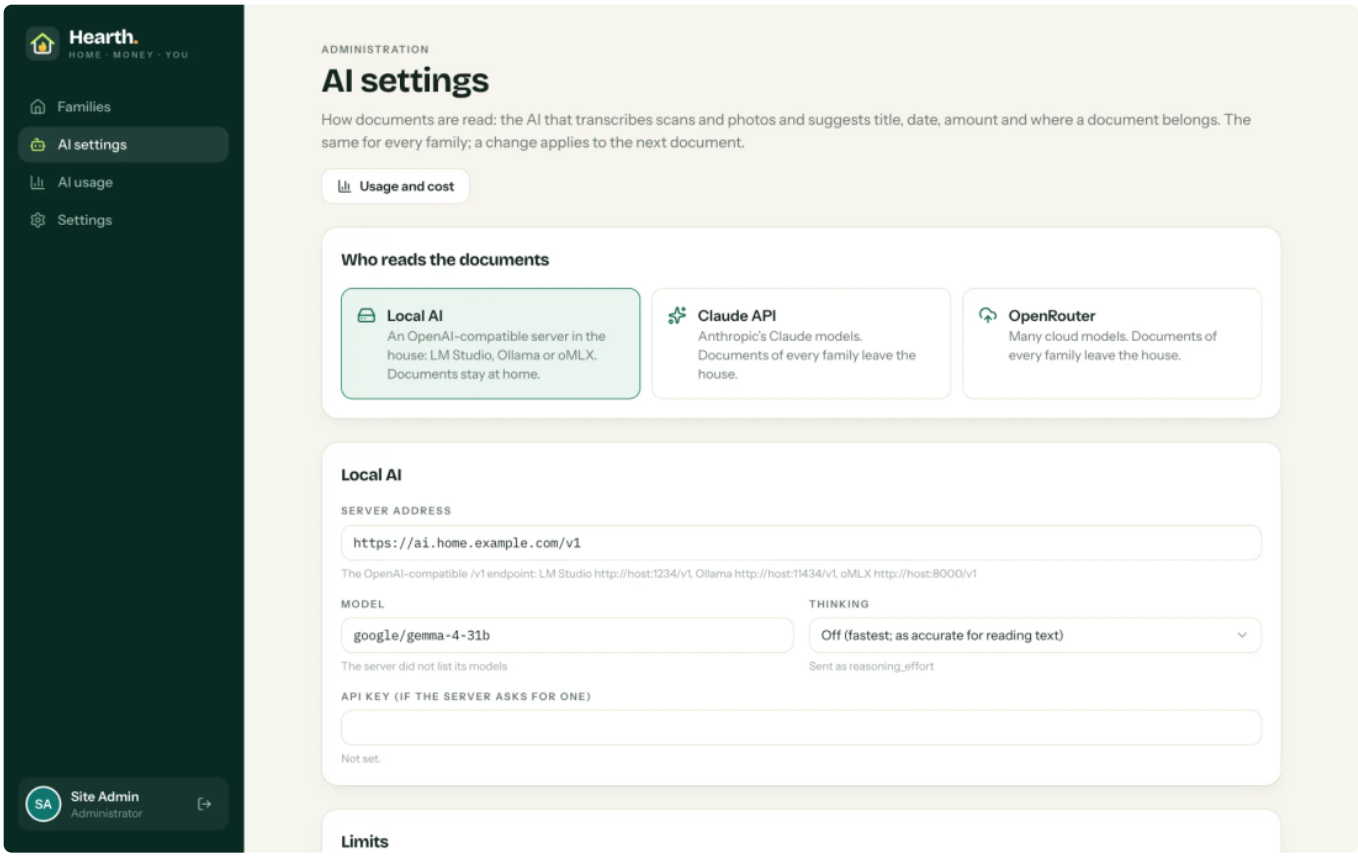


O K pesquisa cada palavra dentro de cada documento, onde quer que ele esteja arquivado.

Ligações telefônicas recebem o mesmo tratamento. Depois de uma ligação com o telhadista, toque no microfone, diga do que se tratava, e o Hearth transcreve tudo em uma conversa com aquele fornecedor. A transcrição roda em um servidor Whisper em um Mac da casa, então a gravação nunca sai de casa.

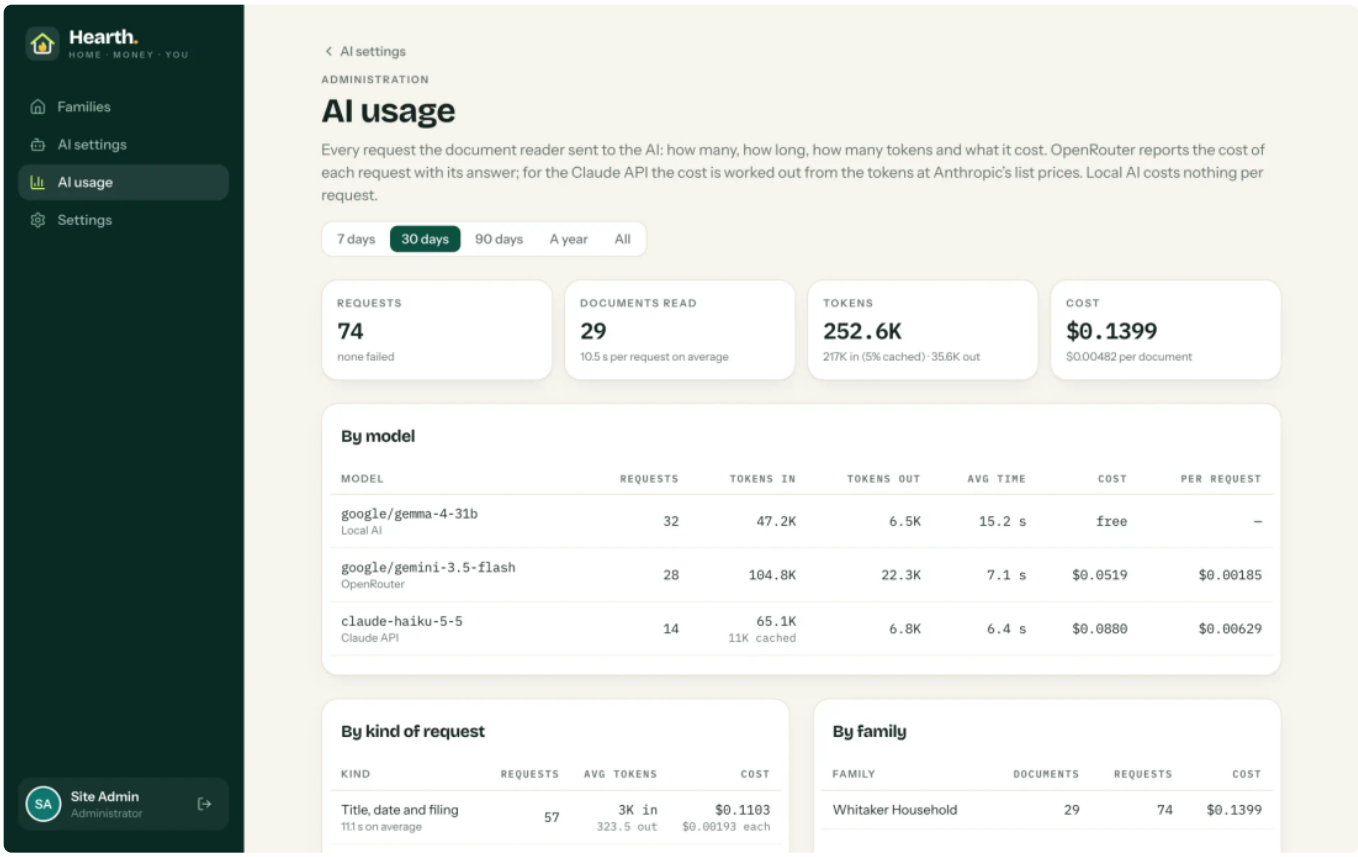
Qualquer provedor de IA, inclusive um que mora em casa

A família escolhe como os documentos são lidos, nas configurações de administração, sem precisar de uma nova versão. O Hearth conversa com três tipos de provedores: a Claude API da Anthropic, qualquer modelo no OpenRouter (que coloca os modelos da OpenAI, do Google, da Anthropic, da Mistral e de muitos outros atrás de uma única API) ou um modelo local em qualquer servidor compatível com OpenAI, como o LM Studio ou o Ollama. Com a opção local, nem uma única página sai de casa -- algo sobre o qual escrevi em detalhes, para Delphi, em Give Your Delphi App a Brain.



As configurações de IA: Claude, qualquer modelo no OpenRouter ou um modelo local que mantém cada página dentro de casa.

Como "IA" sem a conta anexada é conto de fadas, cada requisição é registrada com os seus tokens, o seu tempo, os seus erros e o seu custo. Os modelos na nuvem informam ou estimam o preço; o modelo local não custa nada além de eletricidade. O administrador vê tudo por modelo, por tipo de requisição, por família e por dia.



Uso e custo da IA: requisições, tokens e custo por modelo e provedor -- o modelo local não custa nada (dados de demonstração).

Por que a liberdade de provedor importa para o seu produto

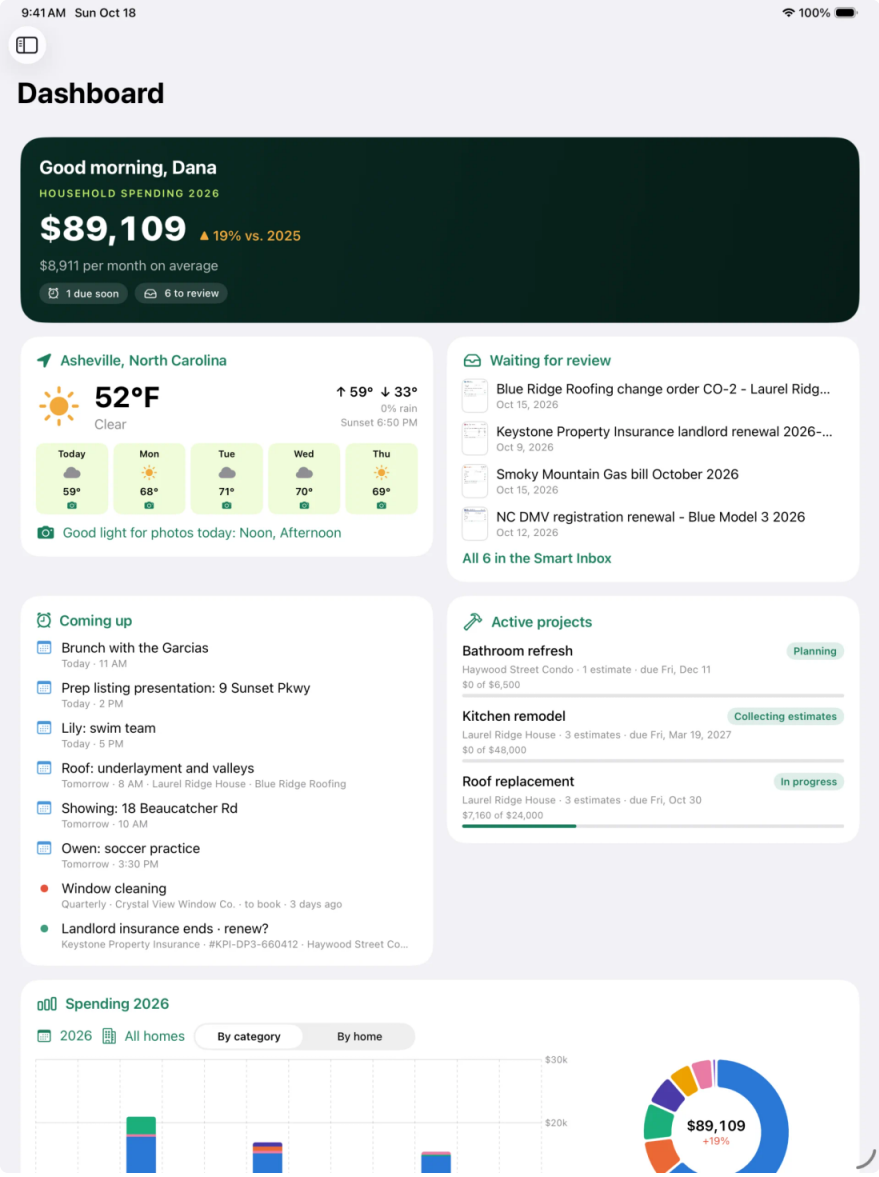
Se você coloca IA em um produto hoje, o melhor modelo do ano que vem muito provavelmente virá de um fornecedor diferente do deste ano. Uma aplicação que trata o provedor como uma configuração, e não como uma decisão de arquitetura, consegue trocar de provedor em uma tarde. Isso também permite que um cliente com regras rígidas de privacidade rode tudo localmente, o que já é um argumento de venda por si só.

Um backend, sete portas de entrada

O Hearth não é um app; é um servidor com várias portas de entrada. A aplicação web faz tudo. Um app somente leitura para iPhone e iPad mostra cada seção dela. Um app para Apple TV transforma a tela da sala em um painel da família. Notes for Mac e Passwords for Mac substituem por completo, de forma nativa, os apps que usávamos antes, e uma extensão do Chrome preenche os logins nos próprios sites.

Um servidor, um banco de dados, um conjunto de regras -- e sete portas de entrada

O diagrama parece simples porque a parte difícil é invisível: toda porta de entrada passa pelas mesmas regras no servidor. Cada app faz login por conta própria, com um código confirmado na web, e cada tipo de app só pode fazer aquilo para que foi feito. A extensão do Chrome pode ler e salvar senhas, e nada mais; se alguém roubasse o login dela do navegador, não conseguiria abrir um único documento.



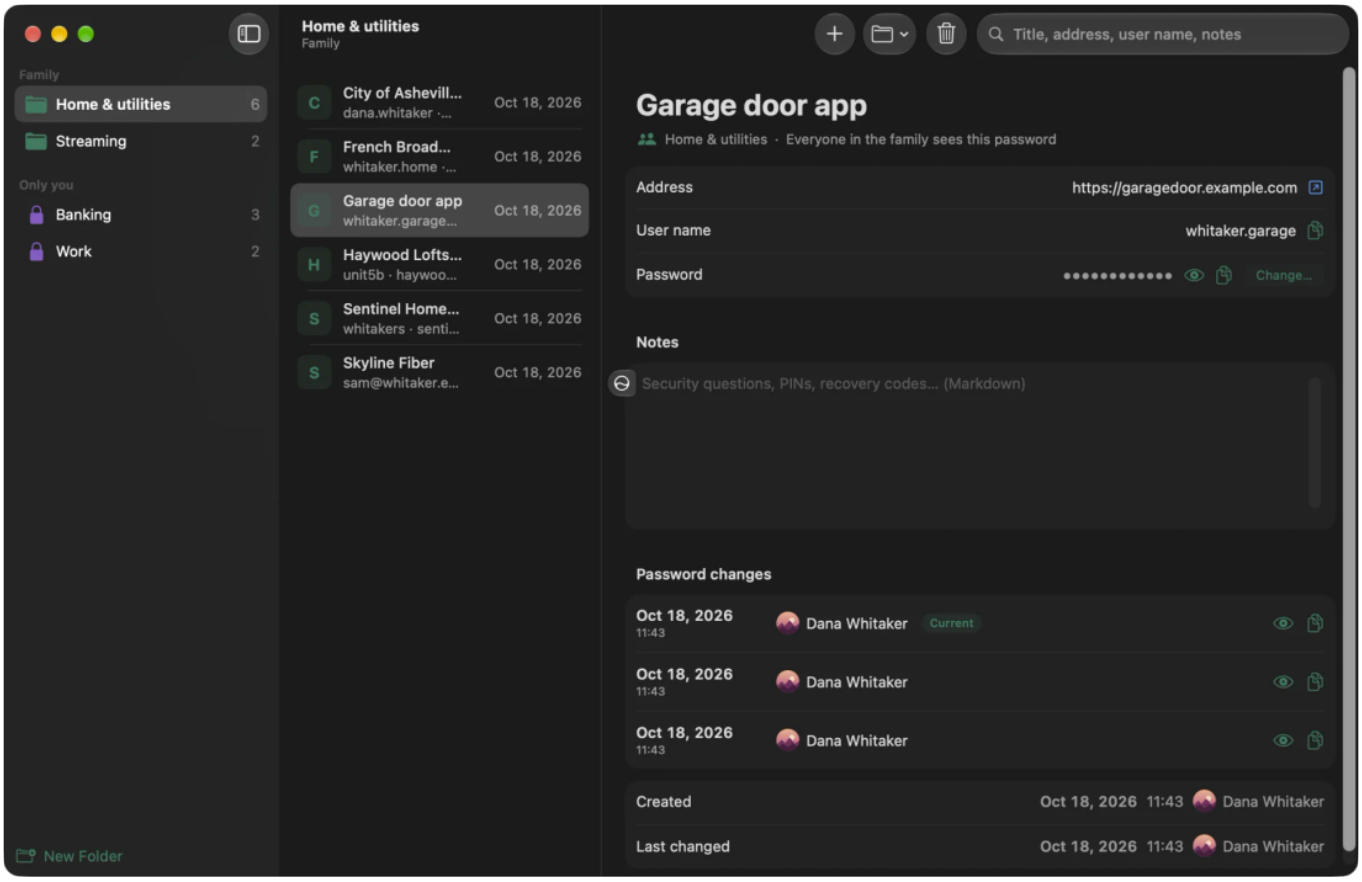
O app para iPad mostra cada seção do Hearth, somente leitura, com os documentos no Quick Look.

O app para Apple TV merece um comentário, porque é o que arranca sorrisos. Ele mostra os compromissos de hoje, o mês, o clima com as horas de boa luz para fotos e os gastos do ano, e roda como uma apresentação de slides que mantém o protetor de tela longe. Em uma casa, isso é um painel da família. Em um escritório, um showroom ou uma sala de espera, a mesmíssima ideia se chama **digital signage**, ou sinalização digital,

alimentada ao vivo pelos seus próprios dados em vez de um pendrive que alguém atualiza uma vez por trimestre.

O app para Apple TV: o dia de hoje, o calendário, o clima e os gastos -- um painel da família, ou sinalização digital em um escritório.

No Mac, os dois apps nativos substituíram os apps de notas e de senhas que usávamos antes. As senhas ficam escondidas atrás do Touch ID e voltam a se bloquear depois de cinco minutos, e a extensão do Chrome oferece salvar ou atualizar um login logo depois que você entra em algum site.

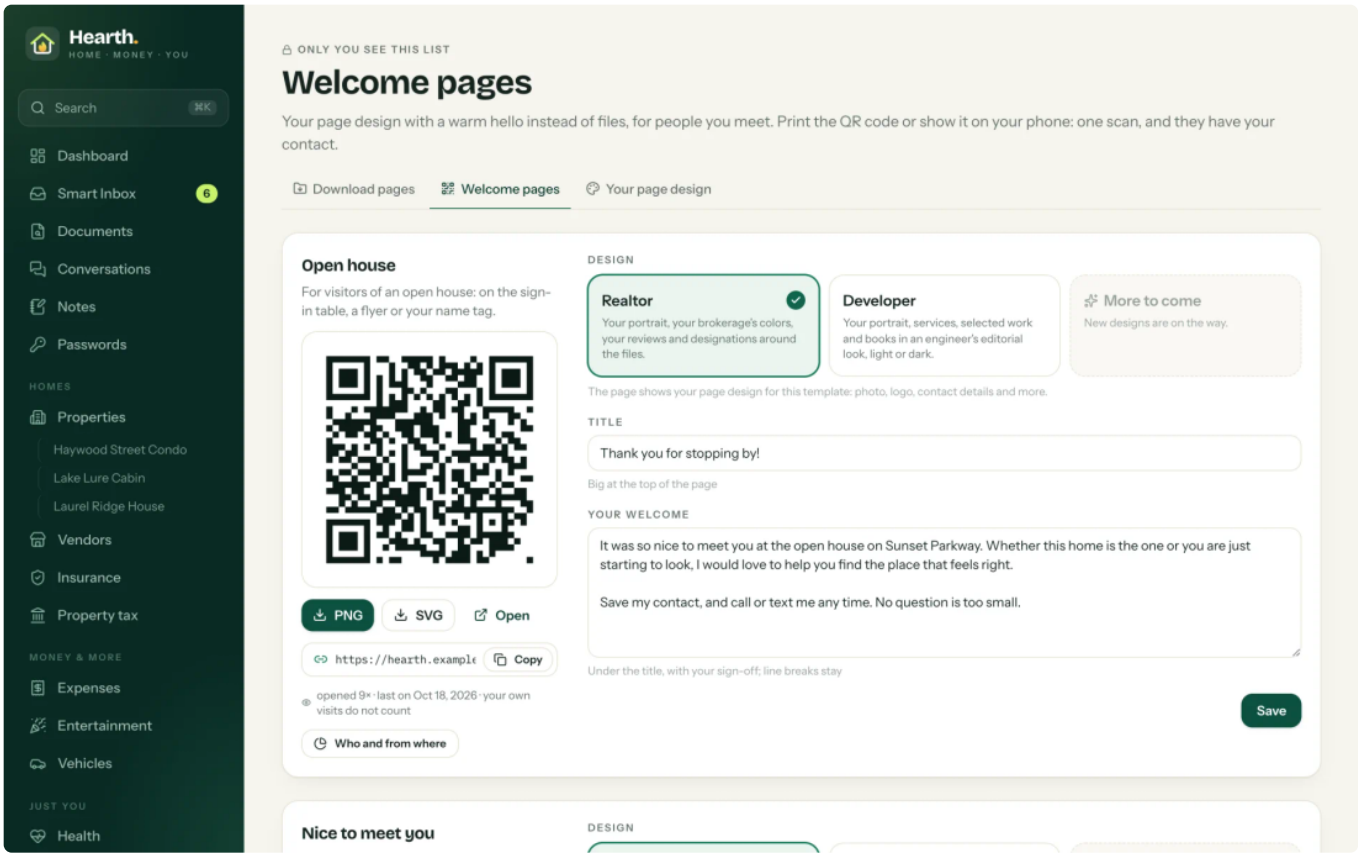


Passwords for Mac: protegido pelo Touch ID, com o histórico de senhas e um gerador.

Feito para uma corretora, não para um milhão de usuários

É aqui que o software sob medida se paga. Um produto genérico precisa atender todo mundo, então não atende ninguém com exatidão. O Hearth sabe que a dona dele vende casas, por isso tem coisas que nenhum app doméstico jamais traria.

As páginas de boas-vindas são um bom exemplo. Uma corretora pode criar uma página para um cliente ou para um open house -- a partir de um modelo, com um QR code para imprimir -- e o Hearth registra quando ela foi aberta, mais ou menos de onde e com qual navegador. O card do clima marca os dias com boa luz para fotos de anúncio. Nada disso levou muito tempo para construir. Cada um desses itens seria um pedido de funcionalidade no backlog de um grande produto, votado por milhares de usuários que não precisam dele.



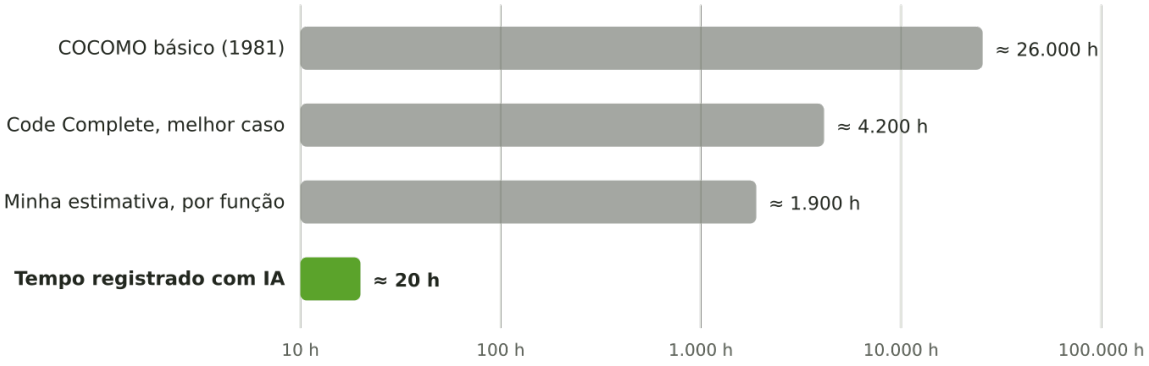
Uma página de boas-vindas com QR code, criada a partir de um modelo em um minuto (dados de demonstração).

> **O mesmo padrão, em uma empresa**
> Troque "corretora de imóveis" por "corretor de seguros", "administradora de imóveis", "laboratório de prótese dentária" ou "fabricante de máquinas", e a história não muda. Uma empresa que compra um grande CRM paga por centenas de funções que nunca usa e depois paga de novo -- em consultores ou em gambiarras -- pelas cinco de que realmente precisa. Um sistema sob medida começa por essas cinco.

Quanto tempo isso teria levado sem IA?

Doze dias é o tempo de calendário entre o primeiro prompt, em 7 de outubro, e a última versão, em 18 de outubro. Os meus logs de sessão do Claude Code mostram cerca de **20 horas** de trabalho ativo com a IA nesse período. Esse número conta apenas o tempo em que uma sessão estava ocupada, sem as pausas de mais de vinte minutos, então os testes no iPad e as reflexões na varanda não entram nele.

O resultado são **58.588 linhas não vazias** de TypeScript, Swift, SQL e JavaScript, 52 pull requests mesclados e 48 versões com tag. Linhas de código são uma medida ruim de valor -- Edsger Dijkstra argumentou, numa frase famosa, que elas deveriam ser contabilizadas como "linhas gastas", e não como "linhas produzidas" -- mas são a única medida de tamanho que os modelos clássicos de estimativa aceitam, então vamos perguntar a eles.



Esforço estimado para construir o Hearth à mão, comparado ao tempo registrado com IA (horas, escala logarítmica)

O gráfico precisa de uma escala logarítmica, o que por si só já diz alguma coisa. Veja de onde vem cada barra.

1. **COCOMO básico.** O Constructive Cost Model de Barry Boehm, de 1981, estima o esforço em 2,4 × KLOC^1,05 pessoas-mês para um projeto "orgânico". Para 58,6 mil linhas, isso dá cerca de 172 pessoas-mês, ou aproximadamente 26.000 horas, com as 152 horas por mês do modelo. O modelo é antigo e pressupõe um mundo sem frameworks modernos, então eu o trato como o teto.
2. **Code Complete, melhor caso.** O livro *Code Complete* (2ª edição), de Steve McConnell, indica de 2.000 a 25.000 linhas por pessoa-ano para projetos de 10.000 linhas, e de 1.000 a 20.000 para projetos de 100.000 linhas (tabela reproduzida aqui). Tomando o extremo mais otimista, 25.000 linhas por ano, o Hearth equivale a cerca de 2,3 pessoas-ano, aproximadamente 4.200 horas com 1.800 horas de trabalho por ano.
3. **Minha própria estimativa.** Estimei do jeito que estimo projetos de clientes: função por função, para um desenvolvedor experiente, com frameworks modernos e sem IA. O modelo de dados e o login com passkeys e isolamento entre famílias: 25 dias. O pipeline de documentos com os provedores de IA: 25. Os módulos da casa, dos imóveis ao plano de saúde: 55. Calendário, painel, busca, notas, cofre de senhas e páginas de boas-vindas: 44. Testes, a revisão de segurança e a implantação: 17. A API dos apps, o app para iPhone e iPad, o app para Apple TV, os dois apps para Mac e a extensão do Chrome: 69. Isso dá 235 dias úteis, ou um pouco menos de 1.900 horas -- e é, de propósito, a mais otimista das três.

Mesmo contra a minha própria estimativa otimista, a diferença é de cerca de duas ordens de grandeza. Sei bem como isso soa, por isso a próxima seção é a parte em que argumento contra mim mesmo.

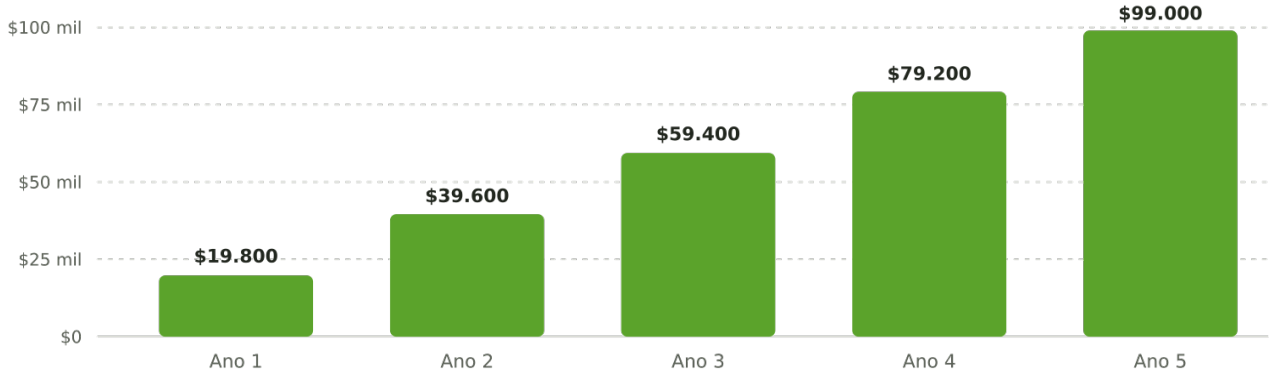
Onde essa comparação não se sustenta

As 20 horas são tempo de sessão, não uma contabilidade completa do meu dia: pensar, testar nos dispositivos e decidir o que construir aconteceu em volta delas. A IA escreveu muito mais código do que eu teria digitado, e mais linhas não significam mais valor. Um projeto do zero, com um único usuário exigente e sem dados legados, é o melhor cenário possível para a IA -- uma migração com vinte anos de regras de negócio escondidas em uma base de código Delphi 7 é outro bicho, como descrevi em Adapt or Disappear. E as pesquisas não apontam todas para o mesmo lado: em um estudo randomizado do METR, desenvolvedores experientes de código aberto foram 19% *mais lentos* com as ferramentas de IA do início de 2025 em suas próprias bases de código maduras (o próprio METR observa que esses resultados estão desatualizados), e o relatório DORA de 2024 constatou que a adoção de IA andou junto com uma estabilidade de entrega menor. A minha conclusão não é "a IA deixa todo mundo 100 vezes mais rápido". É mais restrita: um desenvolvedor experiente que especifica com precisão, revisa tudo e insiste em testes hoje consegue construir em dias o que antes levava um ano.

Como eu evito que isso vire "vibe coding" -- os testes, a revisão de segurança, a barreira antes de cada versão, as verificações de isolamento -- é o assunto do meu próximo post, Não é vibe coding.

Comprar ou desenvolver? A resposta honesta é "depende"

Existe uma conta simples que muitas empresas nunca fazem. Um CRM de nome famoso tem a sua edição enterprise listada por cerca de **$165 a $175 por usuário por mês**, dependendo de qual rastreador de preços você consulta (Capterra, Redress Compliance). Para dez usuários, isso dá cerca de $20.000 por ano só em licenças, antes de complementos, integrações e do consultor que configura tudo.



Licenças acumuladas a preço de tabela para dez usuários de CRM a $165 por usuário por mês (antes de complementos e implantação)

Cerca de $100.000 em cinco anos, por um software que continuará sem saber o que a sua empresa realmente faz. Na minha visão, esse orçamento hoje compra um sistema sob medida com folga -- um sistema que é seu, que fala o seu vocabulário e que cresce com o negócio, e não com o roadmap do fornecedor.

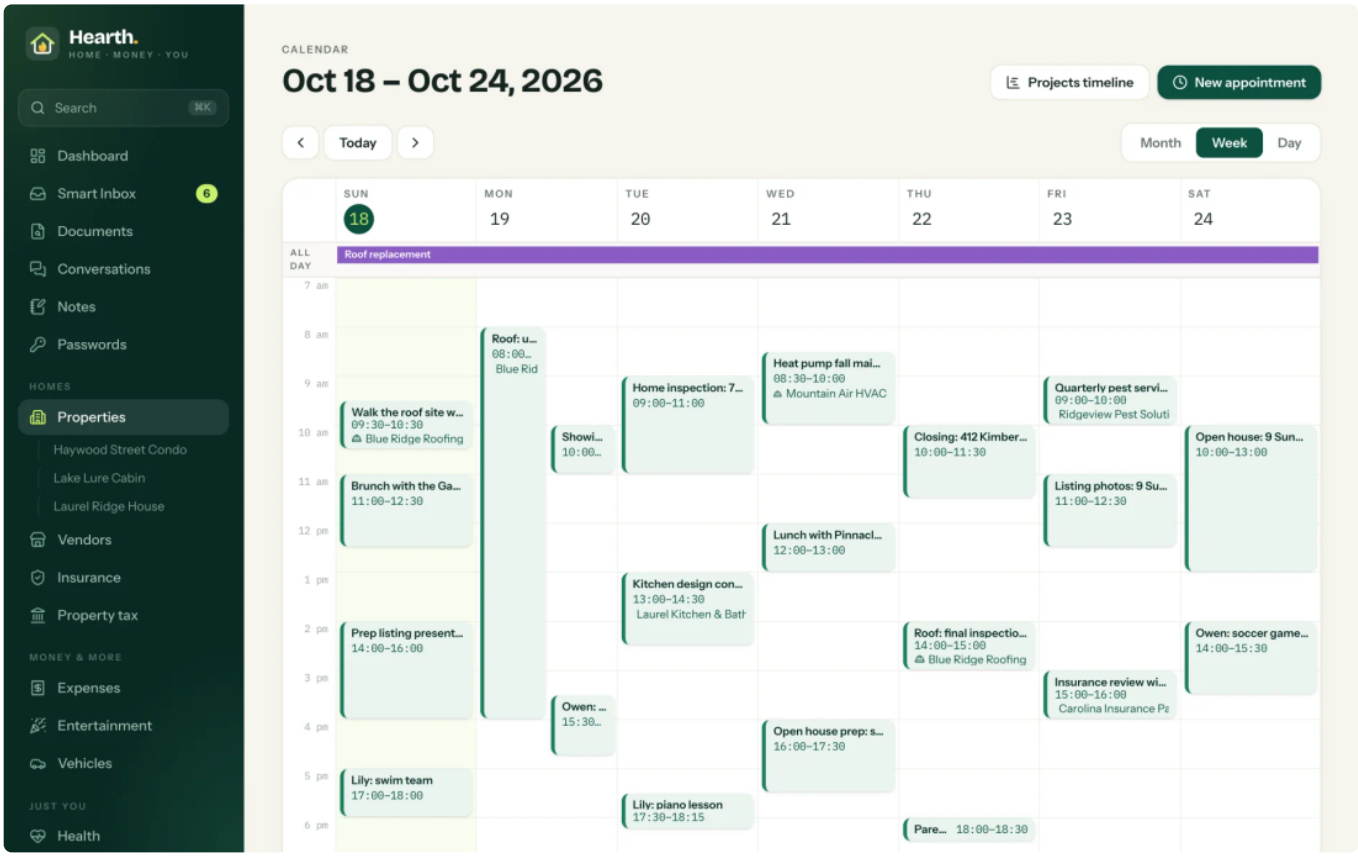
Dito isso, uma assinatura compra coisas reais que é fácil esquecer: hospedagem, atualizações, trabalho de compliance, suporte às 2 da manhã e integrações com mil outros produtos. Um software sob medida precisa de manutenção, de backups e de alguém que atenda quando ele quebra. Se o seu processo é padrão e um produto se encaixa nele, compre o produto -- estou falando sério. E se você gosta da ideia de ser dono do seu sistema, mas não quer começar do zero, existem excelentes CRMs de código aberto, como o Odoo Community, o SuiteCRM, o EspoCRM e o Twenty, que podem ser personalizados e hospedados onde você quiser.

> **Quando desenvolver**
> Desenvolva quando o processo *é* a sua vantagem: quando o seu pessoal passa o dia contornando uma ferramenta, quando os dados vivem em cinco lugares, ou quando a única função de que você precisa é justamente aquela que nenhum fornecedor jamais vai priorizar. Compre quando o processo é commodity -- folha de pagamento, e-mail, contabilidade -- e o produto já faz isso bem.

O que isso significa para a sua aplicação Delphi

A maioria dos meus clientes não chega até mim com uma página em branco. Eles chegam com uma aplicação Delphi que roda o negócio há quinze ou vinte e cinco anos. Ela funciona, os usuários adoram, e aos poucos ela está se tornando o motivo pelo qual a empresa não consegue fazer nada de novo: sem acesso pela web, sem celular, sem tablet, sem IA, e com um grupo de desenvolvedores que não está ficando maior.

O Hearth é um retrato do que uma aplicação dessas pode se tornar, e há vários caminhos honestos para chegar lá. Você pode manter o código Delphi e colocar uma API REST na frente dele -- com o TMS XData ou com nada além do WebBroker -- e depois construir front-ends web, mobile e para TV sobre ela. Você pode renderizar as páginas web a partir do próprio Delphi com o WebStencils. Ou pode levar tudo para uma stack web moderna, o caminho que descrevi na minha série de Delphi para Next.js. A resposta certa depende do seu código, da sua equipe e dos seus usuários, e eu prefiro dizer qual caminho se encaixa a vender aquele de que eu mais gosto.



O calendário, com compromissos que você pode arrastar, redimensionar e adicionar com um clique.

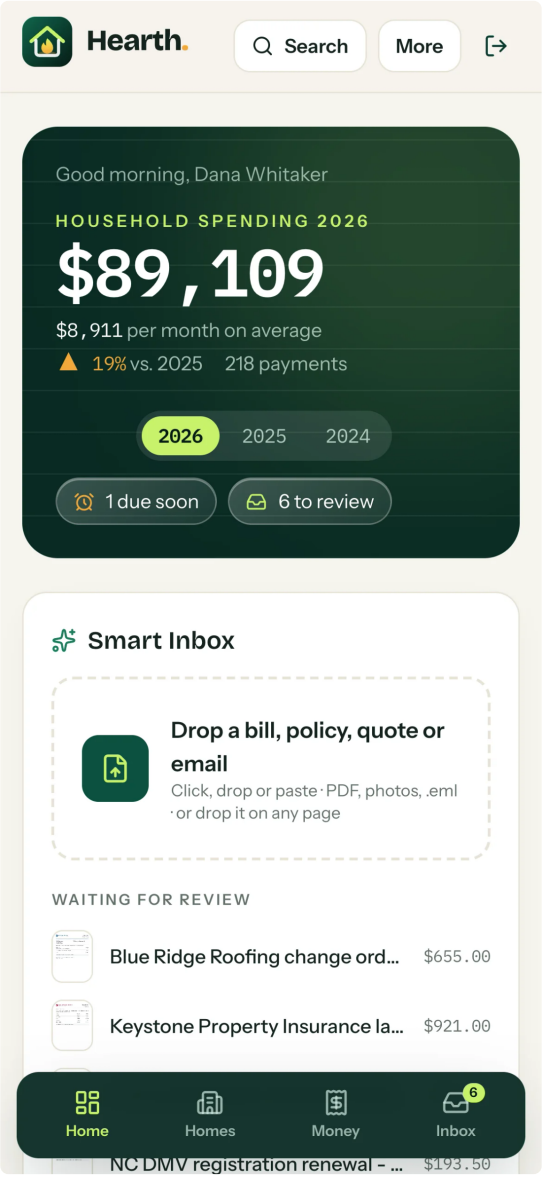
Modernizar um site segue a mesma regra. A sua marca não é um template, então eu mantenho as suas cores, a sua tipografia e o seu logotipo no lugar e modernizo o que está por baixo: velocidade, celulares, acessibilidade, busca e um fluxo de conteúdo que não precisa de um desenvolvedor para cada vírgula. Se você já tem um design system, trabalhamos dentro dele.

Equipes e desenvolvedores individuais

Trabalho dos dois jeitos, e os dois são comuns. Alguns clientes são um único desenvolvedor que carrega um produto há anos e precisa de um parceiro que já fez isso antes -- alguém para revisar o plano, assumir as partes que não combinam com as habilidades dele ou programar em par no primeiro módulo. Outros são equipes que querem a stack moderna e um processo assistido por IA que funcione de verdade, implantado no próprio repositório, com o próprio pessoal fazendo o trabalho e eu revisando junto. Nos dois casos, o trabalho acontece em um repositório que você enxerga desde o primeiro dia, em versões curtas que você pode rodar.

O começo é sempre o mesmo e, como diz a minha página de serviços, não custa nada: uma conversa de trinta a sessenta minutos sobre o sistema e sobre o que dói. Para código existente, o próximo passo é uma avaliação de escopo fechado -- eu leio, compilo e rodo o código, e você recebe um plano por escrito e uma estimativa

realista que são seus, quer a gente continue ou não. Você encontra mais sistemas que entreguei na página Work.



O app web no celular: o mesmo sistema, organizado para a tela pequena.

Conclusão

O Hearth começou como presente de aniversário e acabou se tornando o melhor argumento que eu consigo apresentar a favor do software sob medida. Em doze dias, um desenvolvedor experiente com IA construiu o que os livros situam entre uma e várias pessoas-ano: IA que lê e arquiva documentos com qualquer provedor, passkeys e um cofre criptografado, e apps nativos para iPhone, iPad, Apple TV e Mac, além do Chrome. A IA não tomou as decisões; ela tornou barato executá-las.

> A pergunta já não é se você pode bancar um software sob medida. É se você pode se dar ao luxo de continuar moldando o seu negócio ao software dos outros.

Se você tem uma aplicação Delphi que merece um futuro, ou um processo em que nenhum produto se encaixa, eu adoraria ouvir sobre isso. Vamos conversar.

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 —