NEWS · OCTOBER 19, 2026

Bauen statt kaufen: Individualsoftware in 12 Tagen statt noch einer SaaS-Lizenz

Der Haushalt einer Maklerin lief über Tabellen, Papier und eine Handvoll Abonnements -- also habe ich ihr in zwölf Tagen ein eigenes System gebaut, mit KI-gestützter Dokumentenablage, Passkeys, einem verschlüsselten Tresor und Apps für iPhone, iPad, Apple TV, Mac und Chrome, und hier lesen Sie, was das für Ihre Delphi-Anwendung und Ihre nächste Entscheidung zwischen Kaufen und Bauen bedeutet.

By Dr. Holger Flick



Meine Frau ist Immobilienmaklerin. Jahrelang lief unser Haushalt so, wie die meisten Haushalte -- und offen gesagt auch die meisten kleinen Unternehmen -- laufen: eine Tabelle für die Versicherungspolicen, ein Ordner voller PDFs, die niemand wiederfand, eine Kalender-App, eine Notiz-App, ein Passwortmanager und ein wachsender Papierstapel auf der Küchenarbeitsplatte, um den alle herumliefen. Jedes dieser Werkzeuge war gut in seiner Aufgabe. Keines wusste etwas von den anderen, und keines wusste irgendetwas über *ihre* Arbeit.

Also habe ich ihr zum Geburtstag stattdessen ein einziges System gebaut. Es heißt **Hearth**. Es liest jede

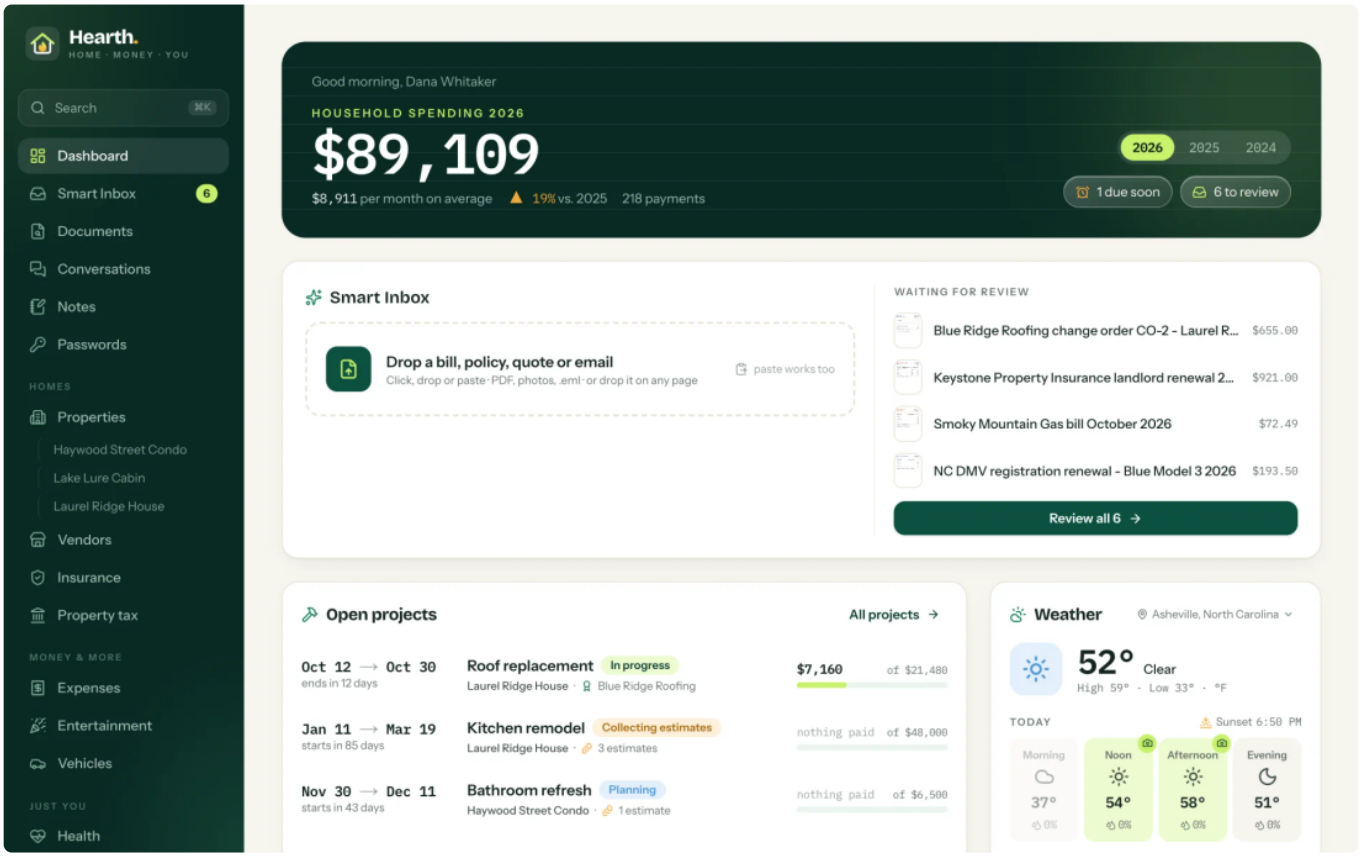
Rechnung, jede Police und jeden Beleg mit KI, legt alles beim richtigen Haus, Auto oder Projekt ab, führt den Kalender, die Notizen und die Passwörter und zeigt das Ganze im Web, auf ihrem iPhone und iPad, auf dem Apple TV im Wohnzimmer, auf dem Mac und direkt in Chrome. Vom ersten Prompt bis zum letzten Release vergingen zwölf Tage.

Dieser Beitrag ist ein Rundgang, und er ist zugleich ein Plädoyer. Meiner Meinung nach ist die Frage „Welches Produkt sollen wir kaufen?“ heute sehr oft die falsche erste Frage. Für immer mehr Unternehmen lautet die bessere Frage: „Wie sähe Software aus, die genau zu uns passt?“ -- denn der Preis für die Antwort ist

eingebrochen. Ich zeige Ihnen, was entstanden ist, wie lange es ohne KI gedauert hätte, wo mein Argument trägt und wo nicht, und was das alles bedeutet, wenn Sie eine Delphi-Desktopanwendung haben, die ein zweites Leben verdient.

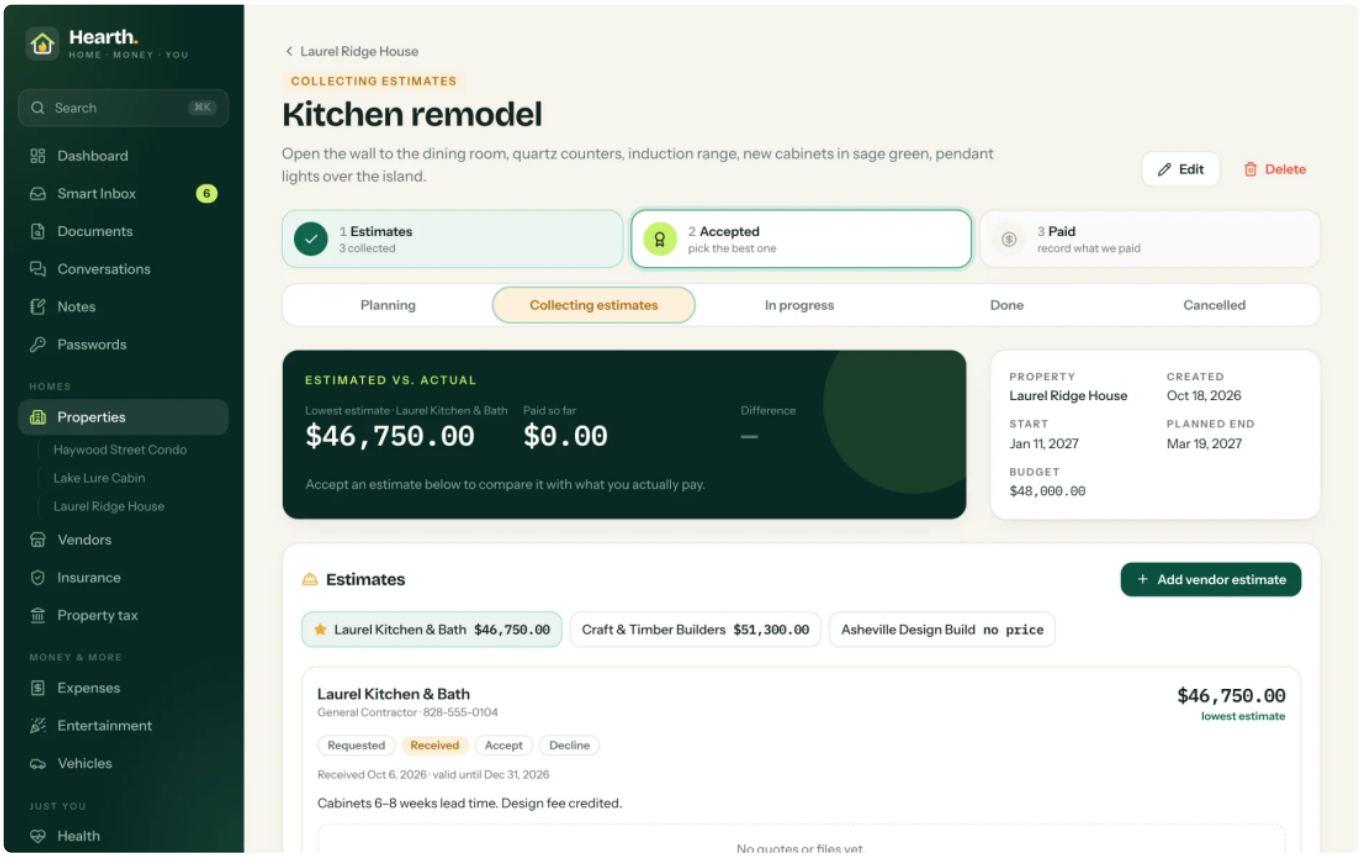
Was Hearth ist, auf einem Bildschirm

Hearth verwaltet einen Haushalt mit mehreren Häusern und Autos so, wie ein kleines Unternehmen seinen Betrieb verwaltet. Auf dem Dashboard beginnt jeder Tag: das Wetter (mit einem Hinweis auf die Tage mit gutem Licht für Exposé-Fotos), die offenen Projekte, die Ausgaben des Jahres und alles, was auf eine Entscheidung wartet.



Das Dashboard eines fiktiven Demo-Haushalts: das Wetter mit den Tagen mit gutem Licht für Fotos, offene Projekte und die Ausgaben des Jahres.

Hinter dem Dashboard liegen die Bereiche, die früher eigene Werkzeuge oder eigene Ordner waren. Zu den Immobilien gehören ihre Eigentümergemeinschaft (HOA), die Grundsteuerbescheide mit ihren Raten und die Projekte mit jeweils mehreren Handwerkerangeboten. Zu den Fahrzeugen gehören die Leasingverträge, zu den Versicherungen die Laufzeiten, Ratenpläne und eine Zeitleiste jeder Police, und zur Krankenversicherung ihre Rechnungen. Bei den Dienstleistern stehen ihre Ansprechpartner und Telefonnummern, Unterhaltungen sammeln E-Mails und aufgezeichnete Anrufe, und wiederkehrende Ausgaben buchen sich von selbst. Notizen, Passwörter und ein vollständiger Kalender runden das Bild ab.

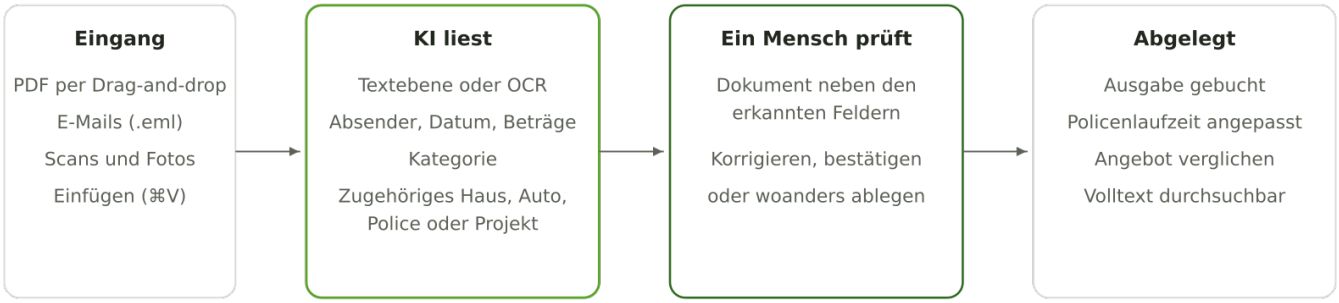


Ein Projekt sammelt Angebote: drei Handwerksbetriebe nebeneinander, das günstigste markiert (Demodaten).

Nichts davon ist exotisch. Genau darum geht es. Jede einzelne Funktion gibt es irgendwo in einem kommerziellen Produkt. Was es nicht gibt, ist das eine Produkt, das genau diese Funktionen hat, genau so miteinander verbunden, für genau diesen Menschen.

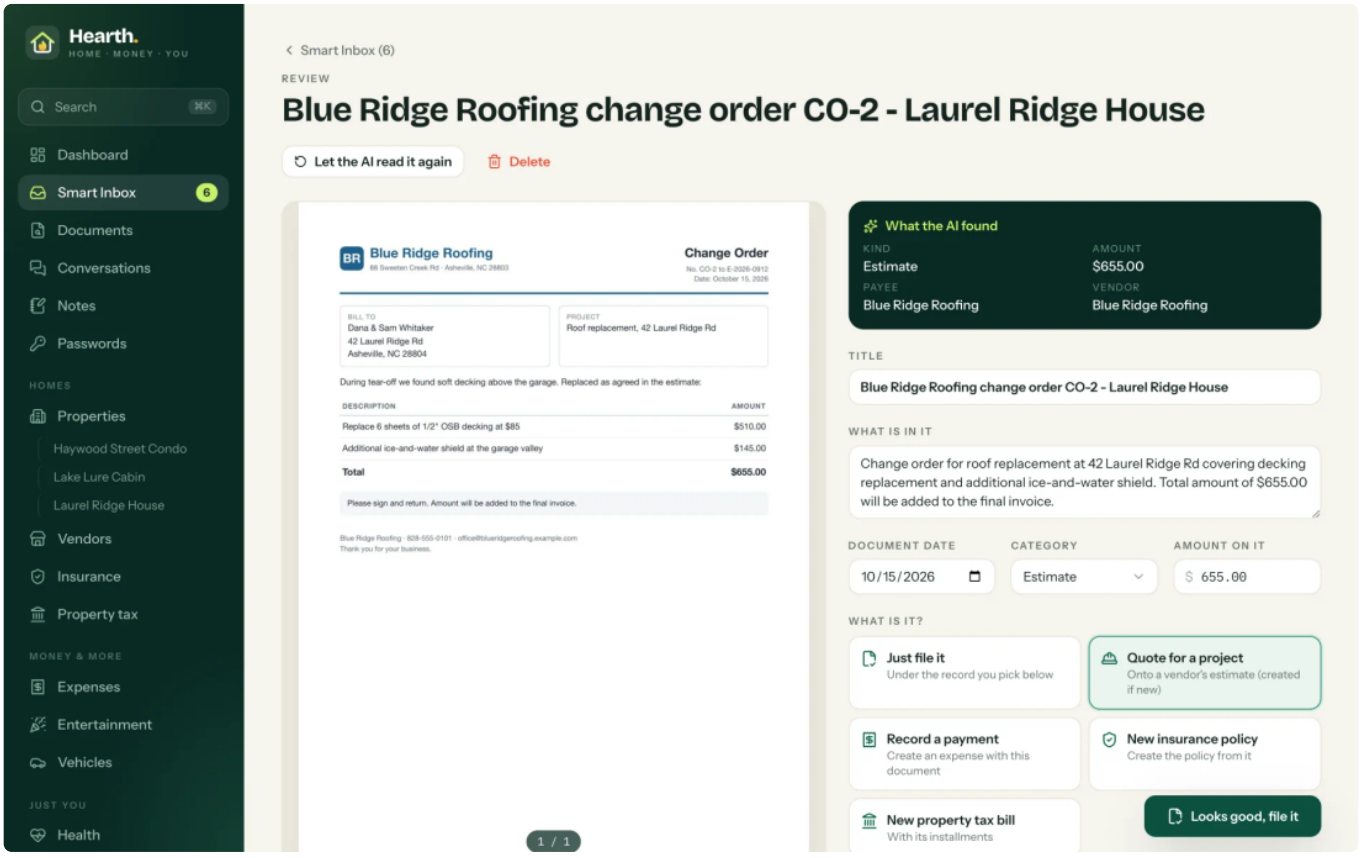
Der Papierstapel, erledigt: OCR, Kategorien und Ablage

Die Funktion, die die Küchenarbeitsplatte verändert hat, ist die Dokumenten-Pipeline. Ein Dokument kommt als PDF, als Foto, als Scan oder als gespeicherte E-Mail mit Anhängen an, wird auf die Seite gezogen oder mit V eingefügt. Bringt ein PDF seinen eigenen Text mit, liest Hearth diesen; ist es ein Scan oder ein Foto, geht jede Seite als OCR durch ein Vision-Modell. Dann zieht die KI heraus, worauf es ankommt: wer es geschickt hat, das Datum, den Betrag, die Fälligkeit, die Policen- oder Kontonummer. Und dann schlägt sie vor, wohin das Dokument gehört: welche Kategorie, welches Haus oder Auto, welche Police oder welches Projekt.



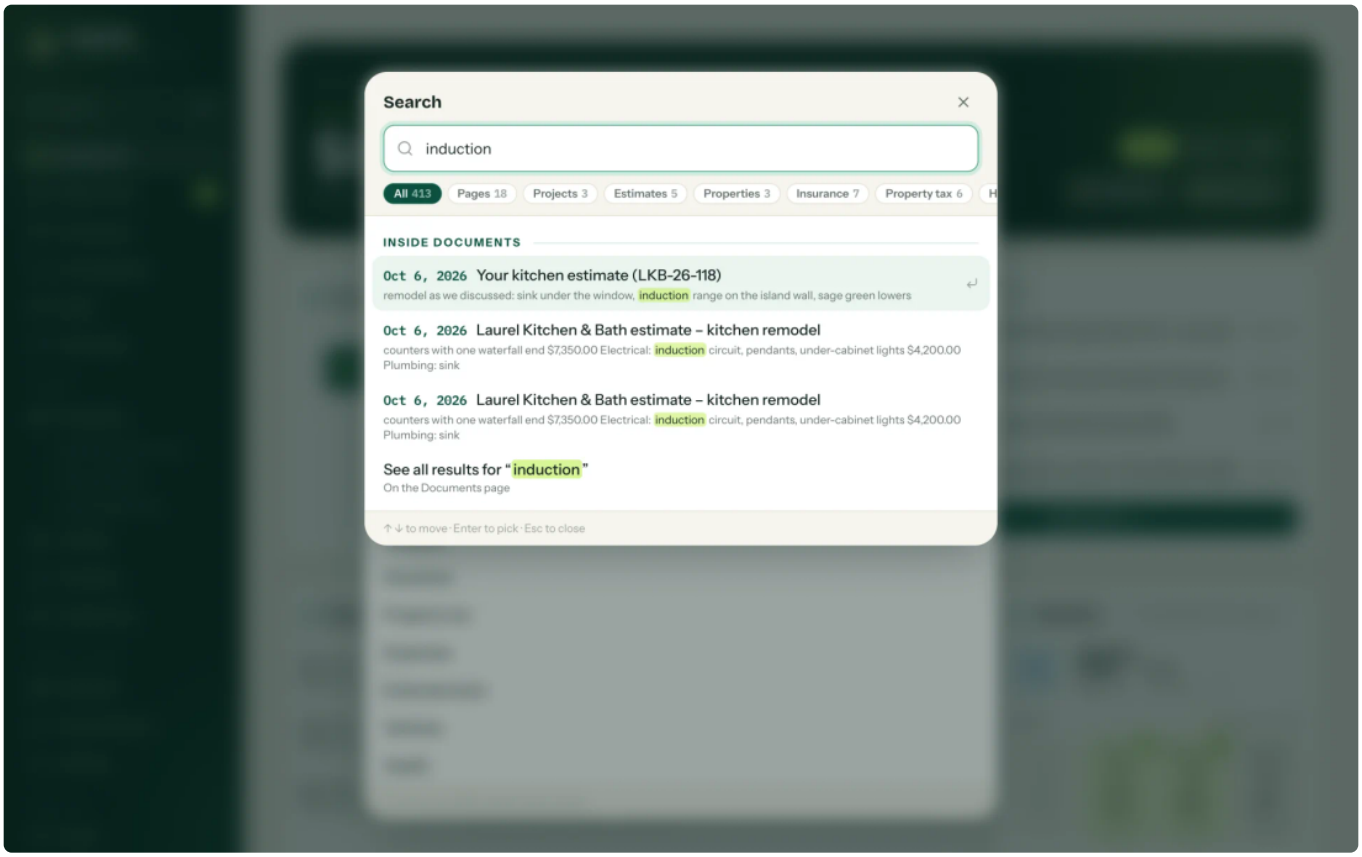
Vom Papierstapel zum abgelegten, durchsuchbaren Datensatz -- mit einem Menschen in der Schleife

Der dritte Kasten ist mir am wichtigsten. Die KI entscheidet nie allein, wohin etwas gehört: Was in der Smart Inbox landet, wartet, bis ein Mensch den Vorschlag bestätigt oder korrigiert. (Wer ein Dokument direkt auf ein Projekt oder eine Police zieht, hat bereits gesagt, wohin es gehört, also überspringt es die Warteschlange.) Die Prüfansicht zeigt das Dokument direkt neben dem, was die KI gelesen hat, sodass die Kontrolle Sekunden statt Minuten dauert.



Die Smart Inbox: auf der einen Seite das Dokument, auf der anderen, was die KI gelesen hat und wo sie es ablegen würde.

Ist ein Dokument erst abgelegt, leistet es echte Arbeit. Eine Rechnung wird zur Ausgabe beim richtigen Haus, eine Verlängerung der Versicherung verlängert die richtige Police, und das Angebot eines Handwerkers landet neben den anderen Angeboten für dasselbe Projekt, sodass das günstigste aktuelle Angebot sofort ins Auge fällt. Jedes Wort in jedem Dokument lässt sich von überall mit K durchsuchen, und das ist meist der Moment, in dem die Leute aufhören, mich zu fragen, wozu sie das brauchen sollten.



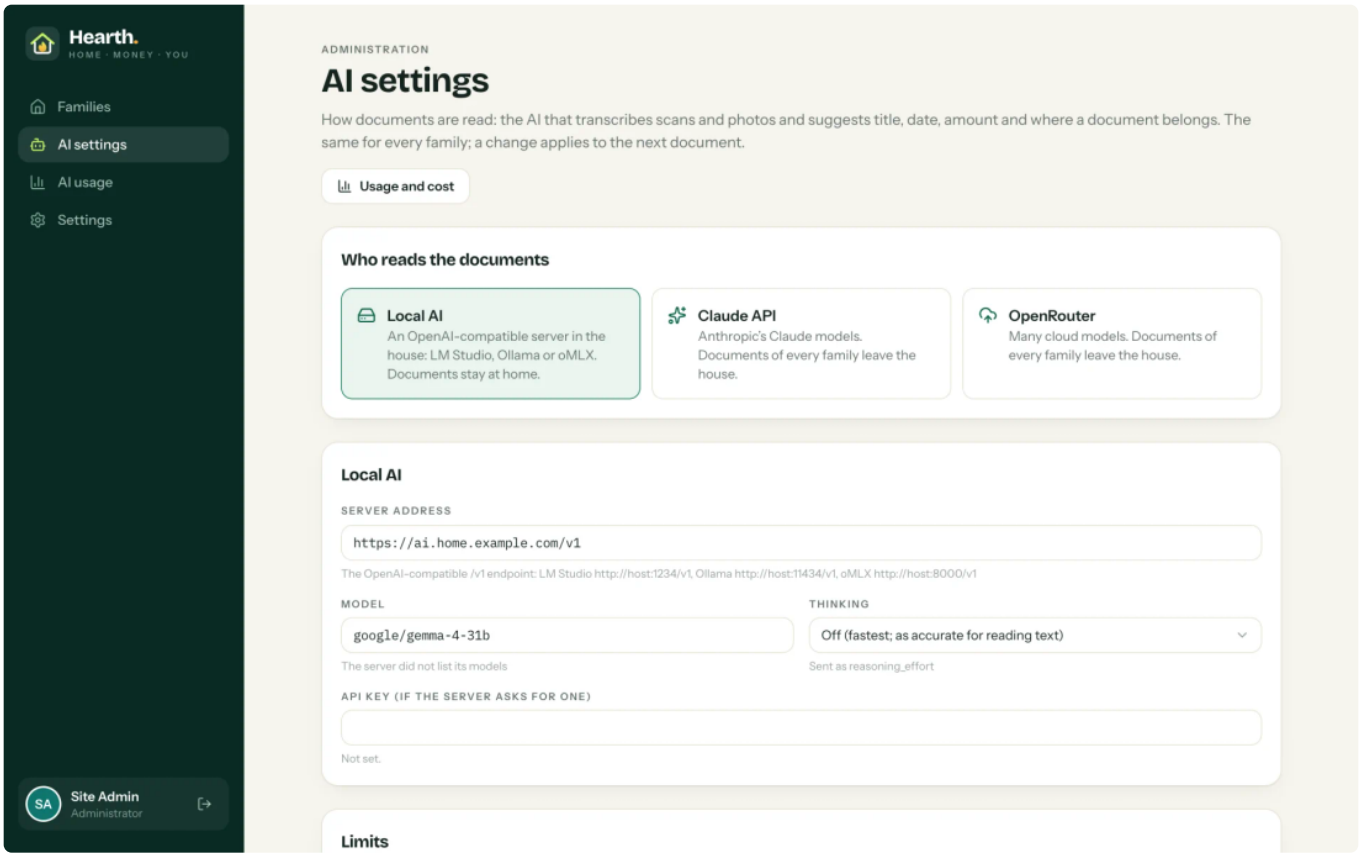
K durchsucht jedes Wort in jedem Dokument, ganz gleich, wo es abgelegt ist.

Telefonate werden genauso behandelt. Nach dem Gespräch mit dem Dachdecker tippt man auf das Mikrofon, sagt, worum es ging, und Hearth transkribiert das Ganze in eine Unterhaltung mit diesem Dienstleister. Die Transkription läuft auf einem Whisper-Server auf einem Mac im Haus, sodass die Aufnahme das Haus nie verlässt.

Jeder KI-Anbieter, auch einer, der im Haus wohnt

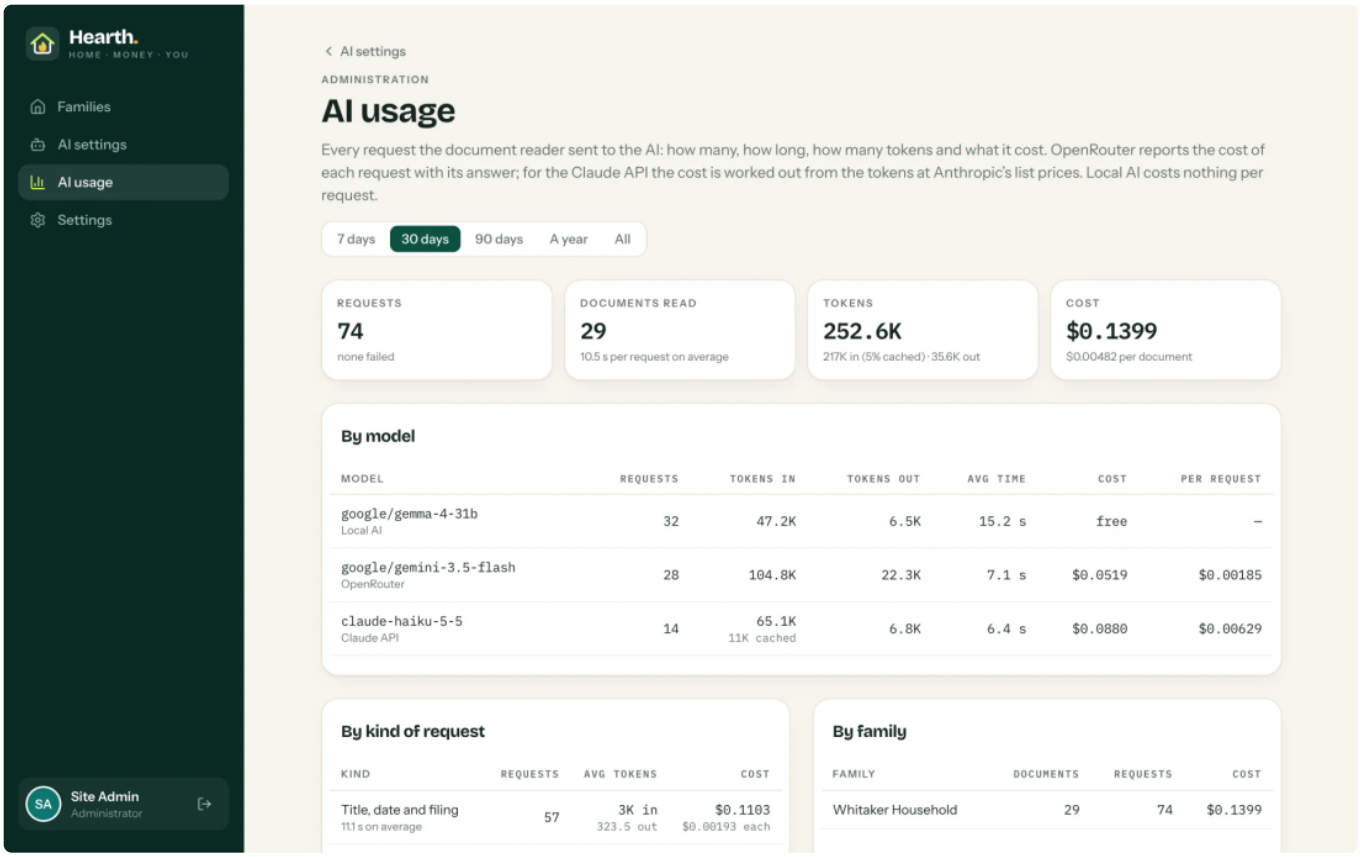
Der Haushalt wählt in den Admin-Einstellungen, wie Dokumente gelesen werden, ganz ohne neues Release. Hearth spricht mit drei Arten von Anbietern: mit der Claude API von Anthropic, mit jedem Modell auf OpenRouter (das die Modelle von OpenAI, Google, Anthropic, Mistral und vielen anderen hinter einer einzigen API bündelt) oder mit einem lokalen Modell auf einem beliebigen OpenAI-kompatiblen Server wie LM Studio oder Ollama.

Mit der lokalen Variante verlässt keine einzige Seite das Haus -- darüber habe ich für Delphi ausführlich in Verpassen Sie Ihrer Delphi-App ein Gehirn geschrieben.



Die KI-Einstellungen: Claude, jedes Modell auf OpenRouter oder ein lokales Modell, das jede Seite im Haus behält.

Weil „KI“ ohne Rechnung ein Märchen ist, wird jede Anfrage mit ihren Tokens, ihrer Dauer, ihren Fehlern und ihren Kosten protokolliert. Cloud-Modelle melden oder schätzen ihren Preis; das lokale Modell kostet nichts außer Strom. Der Administrator sieht das Ganze nach Modell, nach Art der Anfrage, nach Familie und nach Tag.



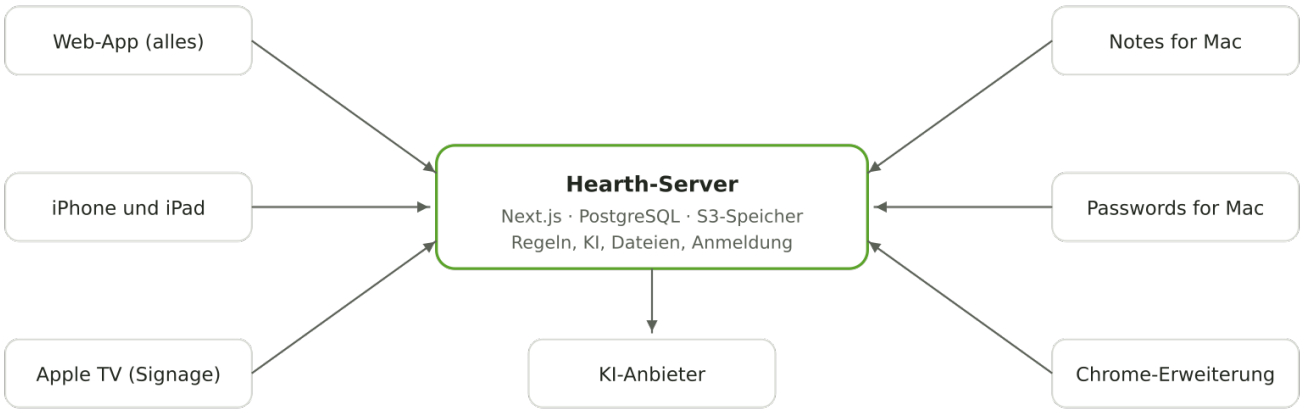
KI-Nutzung und Kosten: Anfragen, Tokens und Kosten je Modell und Anbieter -- das lokale Modell kostet nichts (Demodaten).

Warum die freie Wahl des Anbieters für Ihr Produkt zählt

Wenn Sie heute KI in ein Produkt einbauen, kommt das beste Modell im nächsten Jahr sehr wahrscheinlich von einem anderen Anbieter als in diesem. Eine Anwendung, die den Anbieter als Einstellung behandelt und nicht als Architekturentscheidung, kann an einem Nachmittag wechseln. Außerdem kann ein Kunde mit strengen Datenschutzvorgaben dann alles lokal betreiben, und das ist ein Verkaufsargument für sich.

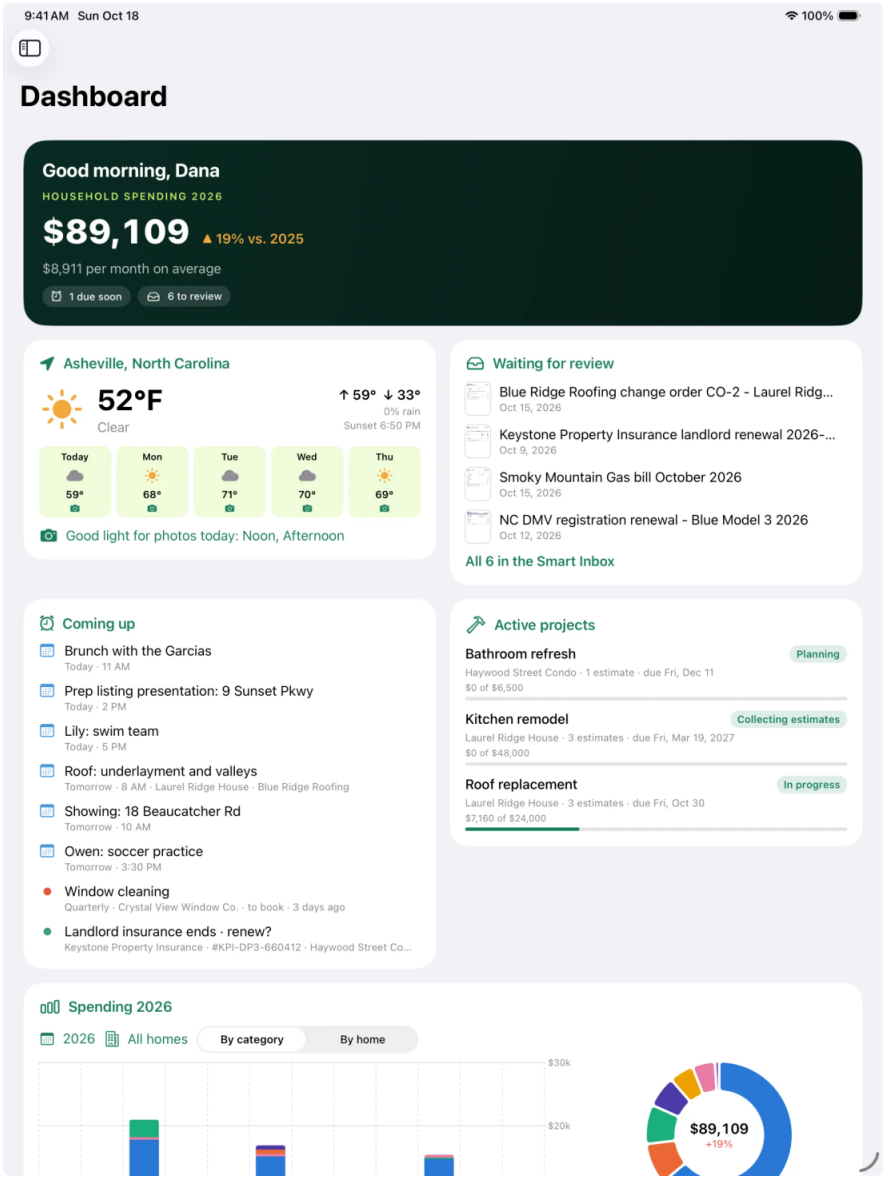
Ein Backend, sieben Eingangstüren

Hearth ist nicht eine App, sondern ein Server mit mehreren Zugängen. Die Webanwendung kann alles. Eine App für iPhone und iPad zeigt jeden Bereich davon, nur lesend. Eine App für Apple TV macht den Fernseher im Wohnzimmer zum Familien-Dashboard. Notes for Mac und Passwords for Mac sind vollwertige native Nachfolger der Apps, die sie ersetzt haben, und eine Chrome-Erweiterung füllt Anmeldedaten direkt auf den Websites aus.



Ein Server, eine Datenbank, ein Satz Regeln -- und sieben Zugänge

Das Diagramm wirkt schlicht, weil der schwierige Teil unsichtbar ist: Jede Eingangstür läuft über dieselben Regeln auf dem Server. Jede App meldet sich einzeln an, mit einem Code, der im Web bestätigt wird, und jede Art von App darf nur das tun, wofür sie gemacht ist. Die Chrome-Erweiterung kann Passwörter lesen und speichern, sonst nichts; wer ihre Anmeldung aus dem Browser stiehlt, kann damit kein einziges Dokument öffnen.



Die iPad-App zeigt jeden Bereich von Hearth, nur lesend, mit Dokumenten in Quick Look.

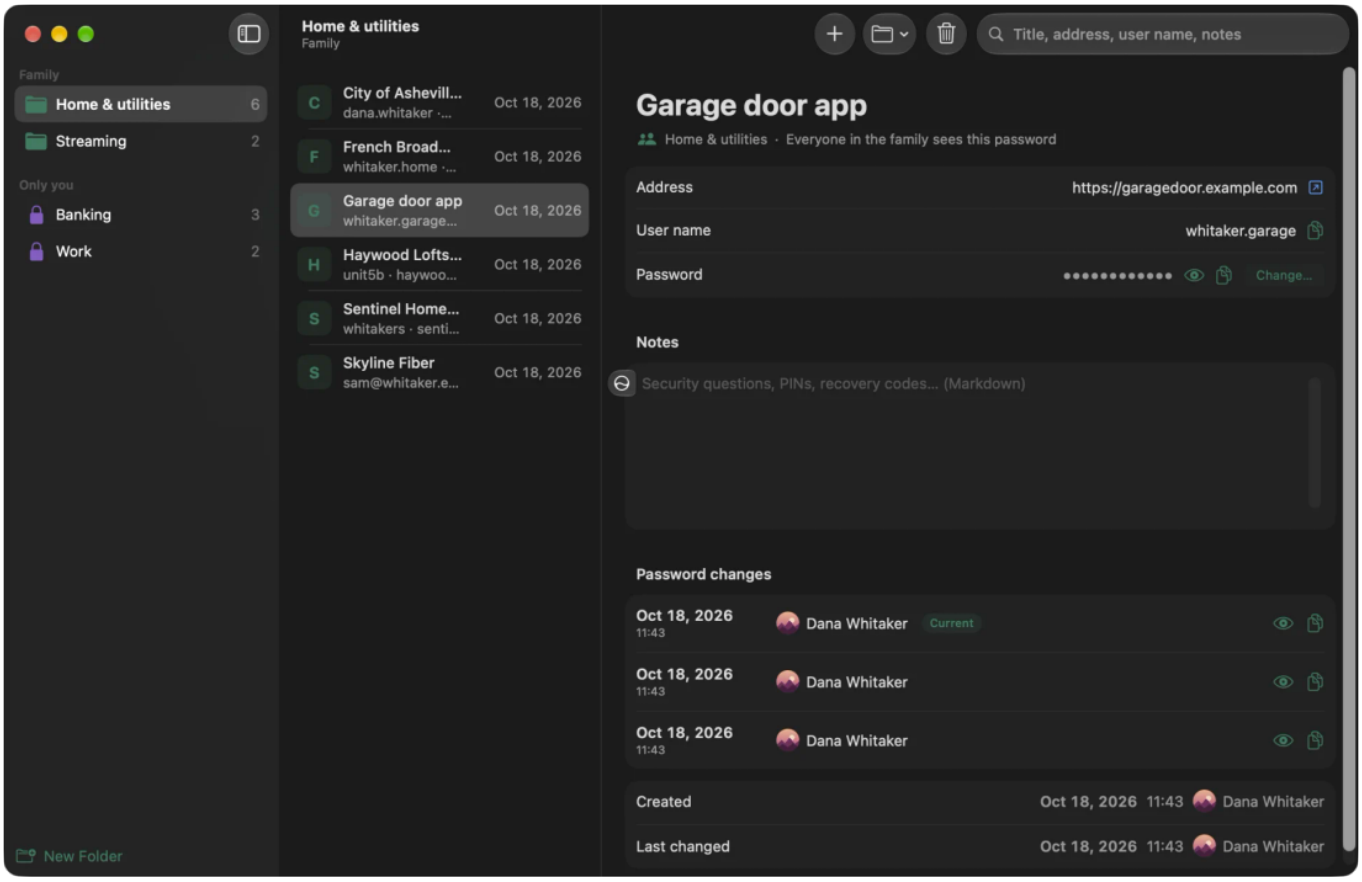
Die Apple-TV-App verdient ein eigenes Wort, denn bei ihr lächeln die Leute. Sie zeigt die Termine des Tages, den Monat, das Wetter mit den Stunden mit gutem Licht für Fotos und die Ausgaben des Jahres, und sie läuft als Diashow, die den Bildschirmschoner fernhält. In einem Zuhause ist das ein Familien-Dashboard. In einem Büro, einem Showroom oder einem Wartezimmer ist genau dieselbe Idee **Digital Signage**, live gespeist aus

Ihren eigenen Daten statt von einem USB-Stick, den jemand einmal im Quartal aktualisiert.



Die Apple-TV-App: der heutige Tag, der Kalender, das Wetter und die Ausgaben -- ein Familien-Dashboard oder Digital Signage im Büro.

Auf dem Mac haben die beiden nativen Apps die Notiz- und Passwort-Apps abgelöst, die wir vorher benutzt haben. Die Passwörter bleiben hinter Touch ID verborgen und sperren sich nach fünf Minuten wieder, und die Chrome-Erweiterung bietet an, eine Anmeldung zu speichern oder zu aktualisieren, gleich nachdem Sie sich irgendwo angemeldet haben.

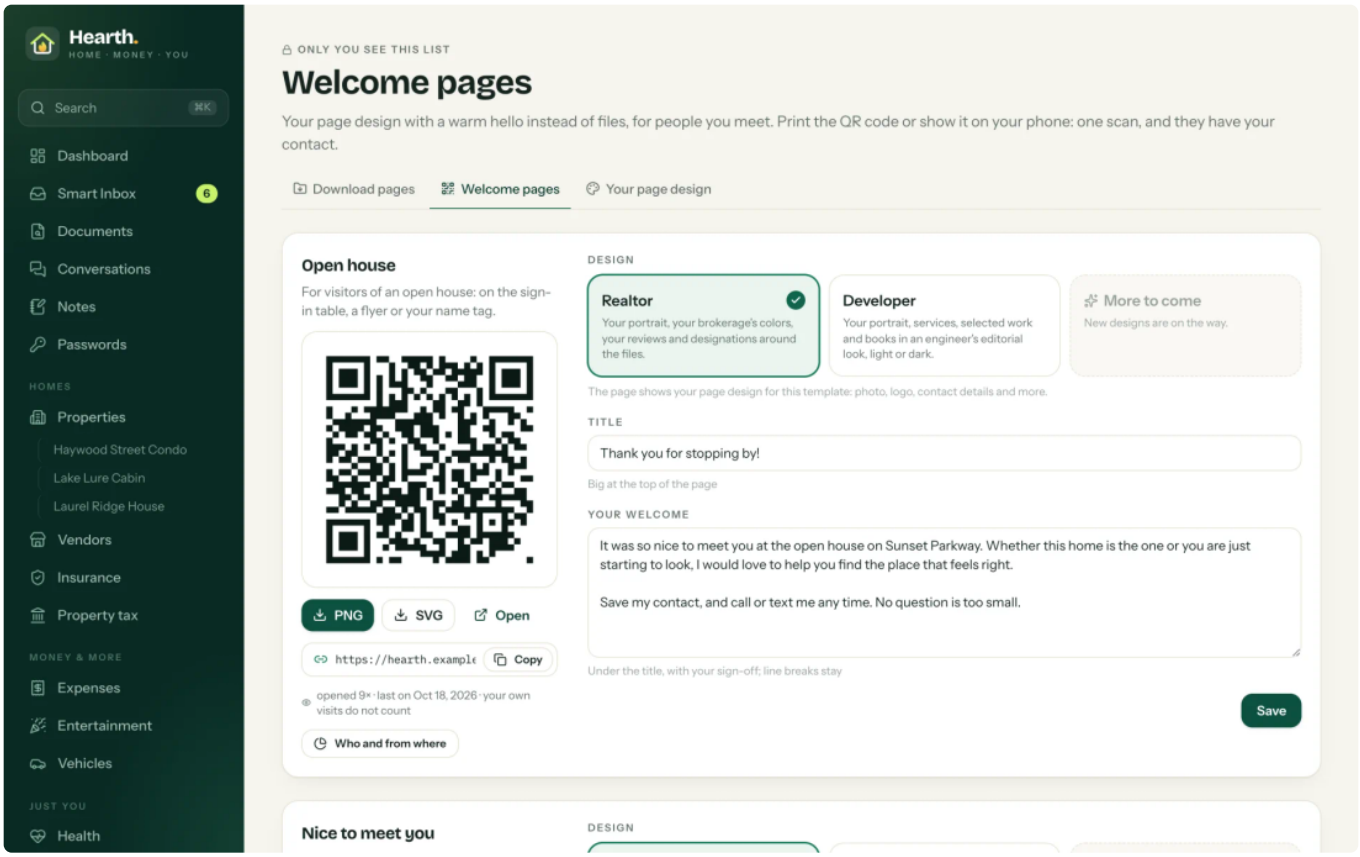


Passwords for Mac: gesperrt hinter Touch ID, mit Passwortverlauf und Generator.

Gebaut für eine Maklerin, nicht für eine Million Nutzer

Hier zeigt Individualsoftware, was sie wert ist. Ein Standardprodukt muss allen dienen, also dient es niemandem genau. Hearth weiß, dass seine Besitzerin Häuser verkauft, und hat deshalb Dinge, die keine Haushalts-App je ausliefern würde.

Willkommensseiten sind ein gutes Beispiel. Eine Maklerin kann eine Seite für einen Kunden oder eine Besichtigung anlegen -- aus einer Vorlage, mit einem QR-Code zum Ausdrucken --, und Hearth protokolliert, wann sie geöffnet wurde, ungefähr von wo und mit welchem Browser. Die Wetterkarte markiert die Tage mit gutem Licht für Exposé-Fotos. Nichts davon hat lange gedauert. Jedes einzelne davon wäre in einem großen Produkt ein Feature-Wunsch im Backlog, über den Tausende Nutzer abstimmen, die ihn nicht brauchen.



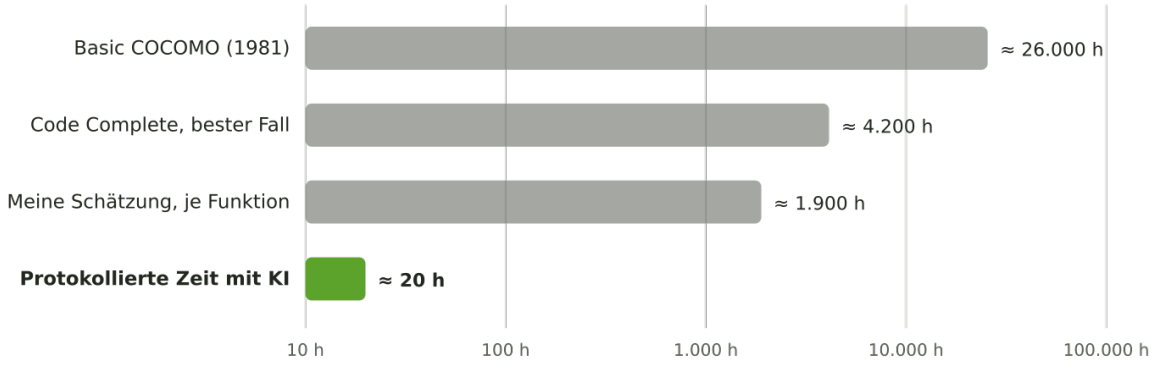
Eine Willkommensseite mit QR-Code, in einer Minute aus einer Vorlage erstellt (Demodaten).

> **Dasselbe Muster, in einem Unternehmen**
> Ersetzen Sie „Maklerin“ durch „Versicherungsmakler“, „Hausverwaltung“, „Dentallabor“ oder „Maschinenbauer“, und die Geschichte bleibt dieselbe. Ein Unternehmen, das ein großes CRM kauft, bezahlt für Hunderte Funktionen, die es nie nutzt, und bezahlt dann noch einmal -- für Berater oder für Umwege -- für die fünf, die es wirklich braucht. Ein maßgeschneidertes System beginnt mit diesen fünf.

Wie lange hätte das ohne KI gedauert?

Zwölf Tage sind die Kalenderzeit vom ersten Prompt am 7. Oktober bis zum letzten Release am 18. Oktober. Meine Claude-Code-Sitzungsprotokolle zeigen in diesem Zeitraum etwa **20 Stunden** aktive Arbeitszeit mit der KI. Diese Zahl zählt nur die Zeit, in der eine Sitzung beschäftigt war, Pausen über zwanzig Minuten nicht mitgerechnet; das Testen auf dem iPad oder das Nachdenken auf der Veranda steckt also nicht darin.

Das Ergebnis sind **58.588 nicht leere Zeilen** TypeScript, Swift, SQL und JavaScript, 52 gemergte Pull-Requests und 48 getaggte Releases. Codezeilen sind ein schlechtes Maß für Wert -- Edsger Dijkstra hat bekanntlich gefordert, sie als „verbrauchte Zeilen“ zu verbuchen, nicht als „produzierte Zeilen“ --, aber sie sind das einzige Größenmaß, das die klassischen Schätzmodelle akzeptieren. Fragen wir sie also.



Geschätzter Aufwand, Hearth von Hand zu bauen, verglichen mit der protokollierten Zeit mit KI (Stunden, logarithmische Skala)

Dass das Diagramm eine logarithmische Skala braucht, sagt schon für sich genommen einiges. Und so kommt jeder Balken zustande.

1. **Basic COCOMO.** Barry Boehms Constructive Cost Model von 1981 schätzt den Aufwand für ein „organisches“ Projekt auf 2,4 × KLOC^1,05 Personenmonate. Für 58,6 Tausend Zeilen sind das etwa 172 Personenmonate, also rund 26.000 Stunden bei den 152 Stunden pro Monat, mit denen das Modell rechnet. Das Modell ist alt und geht von einer Welt ohne moderne Frameworks aus, daher betrachte ich es als Obergrenze.
2. **Code Complete, bester Fall.** Steve McConnells *Code Complete* (2. Auflage) nennt 2.000 bis 25.000 Zeilen pro Personenjahr für Projekte mit 10.000 Zeilen und 1.000 bis 20.000 für Projekte mit 100.000 Zeilen (hier ist die Tabelle wiedergegeben). Nimmt man das optimistischste Ende, 25.000 Zeilen pro Jahr, kommt Hearth auf etwa 2,3 Personenjahre, also rund 4.200 Stunden bei 1.800 Arbeitsstunden im Jahr.
3. **Meine eigene Schätzung.** Ich habe so geschätzt, wie ich Kundenprojekte schätze: Funktion für Funktion, für einen erfahrenen Entwickler, mit modernen Frameworks und ohne KI. Das Datenmodell und die Anmeldung mit Passkeys und Familien-Isolation: 25 Tage. Die Dokumenten-Pipeline mit den KI-Anbietern: 25. Die Haushaltsmodule von den Immobilien bis zur Krankenversicherung: 55. Kalender, Dashboard, Suche, Notizen, Passwort-Tresor und Willkommensseiten: 44. Tests, das Sicherheits-Review und das Deployment: 17. Die App-API, die App für iPhone und iPad, die Apple-TV-App, beide Mac-Apps und die Chrome-Erweiterung: 69. Das sind 235 Arbeitstage oder knapp 1.900 Stunden -- und von den dreien ist das mit Absicht die optimistischste.

Selbst gegenüber meiner eigenen optimistischen Schätzung liegt der Unterschied bei etwa zwei Größenordnungen. Mir ist bewusst, wie das klingt, deshalb argumentiere ich im nächsten Abschnitt gegen mich selbst.

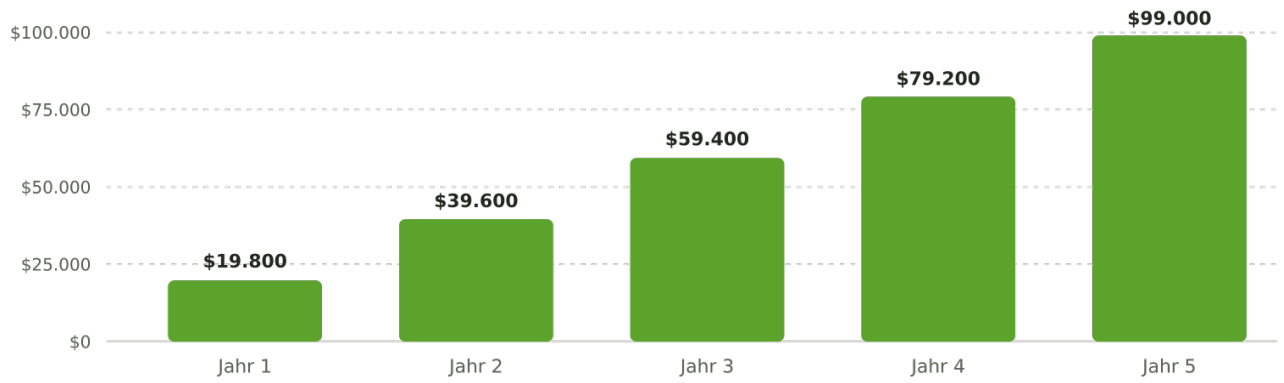
> **Wo dieser Vergleich nicht trägt**
> Die 20 Stunden sind Sitzungszeit, keine vollständige Abrechnung meines Tages: Nachdenken, Testen auf den Geräten und die Entscheidung, was gebaut wird, fanden drumherum statt. Die KI hat weit mehr Code geschrieben, als ich getippt hätte, und mehr Zeilen bedeuten nicht mehr Wert. Ein Neuprojekt mit einer anspruchsvollen Nutzerin und ohne Altdaten ist der günstigste Fall für KI -- eine Migration mit zwanzig Jahren Geschäftsregeln, versteckt in einer Delphi-7-Codebasis, ist ein ganz anderes Kaliber, wie ich in Adapt or Disappear beschrieben habe. Und die Forschung ist uneinheitlich: In einer randomisierten Studie von METR waren erfahrene Open-Source-Entwickler mit KI-Werkzeugen vom Anfang des Jahres 2025 in ihren eigenen ausgereiften Codebasen um 19 % *langsamer* (METR weist selbst darauf hin, dass diese Ergebnisse überholt sind), und der DORA-Report 2024 stellte fest, dass der Einsatz von KI mit einer

> geringeren Stabilität der Auslieferung einherging. Mein Schluss lautet nicht „KI macht alle hundertmal schneller“. Er ist enger gefasst: Ein erfahrener Entwickler, der präzise spezifiziert, alles prüft und auf Tests besteht, kann heute in Tagen bauen, wofür man früher ein Jahr gebraucht hat.

Wie ich verhindere, dass daraus „Vibe Coding“ wird -- die Tests, das Sicherheits-Review, die Release-Schranke, die Isolationsprüfungen --, ist das Thema meines nächsten Beitrags, Kein Vibe Coding.

Kaufen oder bauen? Die ehrliche Antwort lautet „kommt drauf an“

Es gibt eine einfache Rechnung, die viele Unternehmen nie aufstellen. Ein CRM eines großen Namens listet seine Enterprise-Edition für rund **$165 bis $175 pro Nutzer und Monat**, je nachdem, welchen Preisvergleich man fragt (Capterra, Redress Compliance). Für zehn Nutzer sind das etwa $20.000 im Jahr allein an Lizenzen, vor Zusatzmodulen, Integrationen und dem Berater, der das Ganze konfiguriert.



Kumulierte Lizenzkosten zum Listenpreis für zehn CRM-Plätze zu $165 pro Nutzer und Monat (vor Zusatzmodulen und Einführung)

Rund $100.000 in fünf Jahren, für Software, die danach immer noch nicht weiß, was Ihr Unternehmen eigentlich tut. Meiner Ansicht nach bekommt man für dieses Budget heute ein maßgeschneidertes System, und es bleibt noch Luft -- eines, das Ihnen gehört, das Ihre Sprache spricht und das mit dem Unternehmen wächst statt mit der Roadmap des Anbieters.

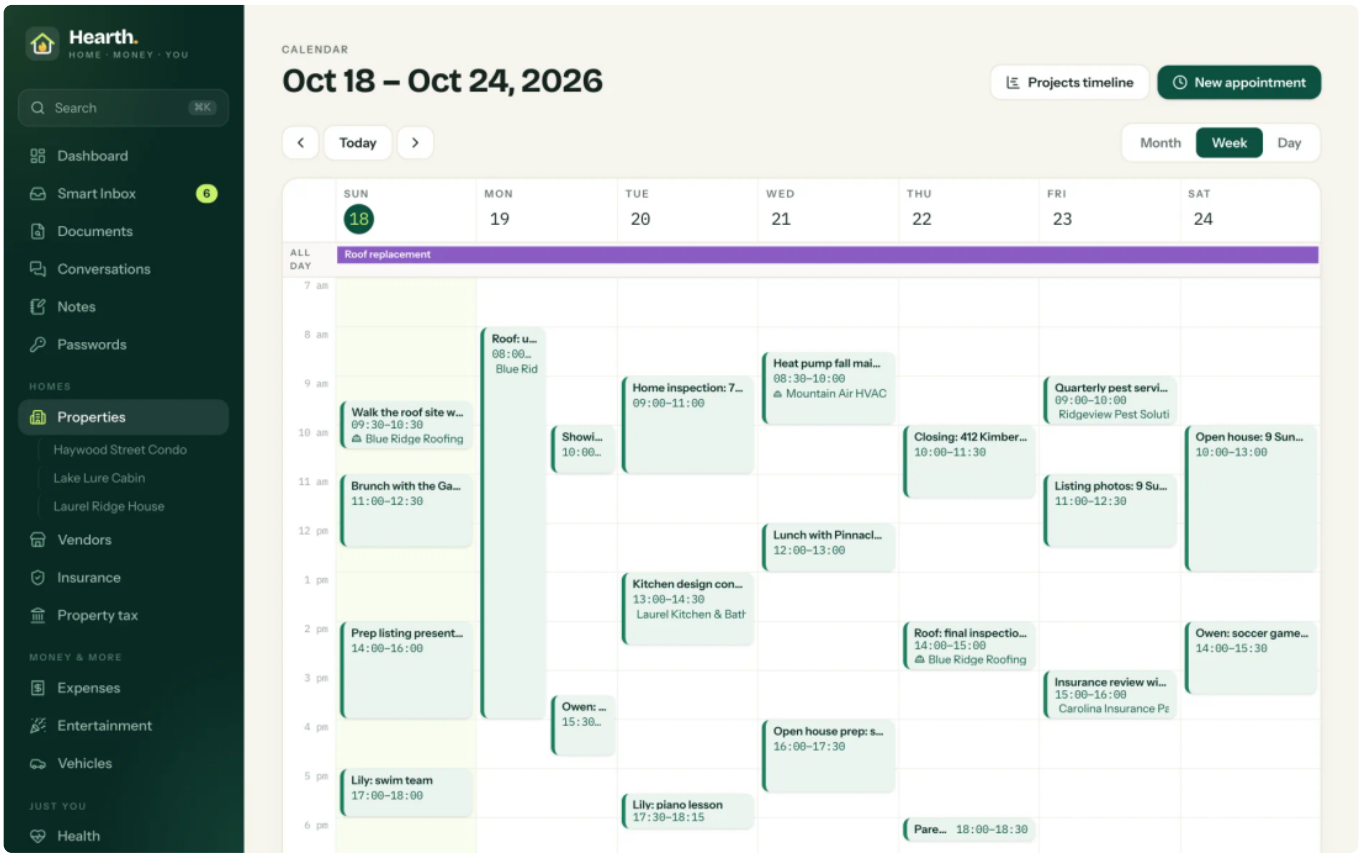
Allerdings kauft man mit einem Abonnement auch echte Leistungen, die man leicht vergisst: Hosting, Updates, Compliance-Arbeit, Support um zwei Uhr nachts und Integrationen mit tausend anderen Produkten. Individualsoftware braucht Wartung, Backups und jemanden, der ans Telefon geht, wenn etwas kaputtgeht. Wenn Ihr Prozess Standard ist und ein Produkt dazu passt, kaufen Sie das Produkt -- das meine ich ernst. Und wenn Ihnen die Idee gefällt, Ihr System zu besitzen, Sie aber nicht bei null anfangen möchten, gibt es hervorragende Open-Source-CRMs wie Odoo Community, SuiteCRM, EspoCRM und Twenty, die sich anpassen und betreiben lassen, wo immer Sie möchten.

> **Wann bauen?**
> Bauen Sie, wenn der Prozess *selbst* Ihr Vorteil ist: wenn Ihre Leute den ganzen Tag um ein Werkzeug herumarbeiten, wenn die Daten an fünf Orten liegen oder wenn die eine Funktion, die Sie brauchen, genau die ist, der kein Anbieter je Priorität geben wird. Kaufen Sie, wenn der Prozess Massenware ist -- Gehaltsabrechnung, E-Mail, Buchhaltung -- und das Produkt ihn bereits gut erledigt.

Was das für Ihre Delphi-Anwendung bedeutet

Die meisten meiner Kunden kommen nicht mit einem leeren Blatt zu mir. Sie kommen mit einer Delphi-Anwendung, die ihr Geschäft seit fünfzehn oder fünfundzwanzig Jahren trägt. Sie funktioniert, die Anwender lieben sie, und langsam wird sie zum Grund, warum das Unternehmen nichts Neues machen kann: kein Webzugriff, kein Smartphone, kein Tablet, keine KI und ein Pool an Entwicklern, der nicht gerade größer wird.

Hearth ist ein Bild davon, was aus einer solchen Anwendung werden kann, und es gibt mehrere ehrliche Wege dorthin. Sie können den Delphi-Code behalten und ihm eine REST-API vorschalten -- mit TMS XData oder mit nichts weiter als WebBroker -- und dann Frontends für Web, Mobilgeräte und Fernseher dagegen bauen. Sie können die Webseiten mit WebStencils direkt aus Delphi rendern. Oder Sie ziehen das Ganze auf einen modernen Web-Stack um, den Weg, den ich in meiner Serie Von Delphi zu Next.js beschrieben habe. Die richtige Antwort hängt von Ihrem Code, Ihrem Team und Ihren Anwendern ab, und ich sage Ihnen lieber, welcher Weg passt, als Ihnen den zu verkaufen, der mir am besten gefällt.



Der Kalender: Termine lassen sich verschieben, in der Länge ändern und mit einem Klick anlegen.

Bei der Modernisierung einer Website gilt dieselbe Regel. Ihre Marke ist keine Vorlage, deshalb behalte ich Ihre Farben, Ihre Typografie und Ihr Logo bei und modernisiere, was darunter liegt: Geschwindigkeit, Smartphones, Barrierefreiheit, Suche und einen Redaktionsablauf, der nicht für jedes Komma einen Entwickler braucht. Wenn Sie bereits ein Designsystem haben, arbeiten wir innerhalb davon.

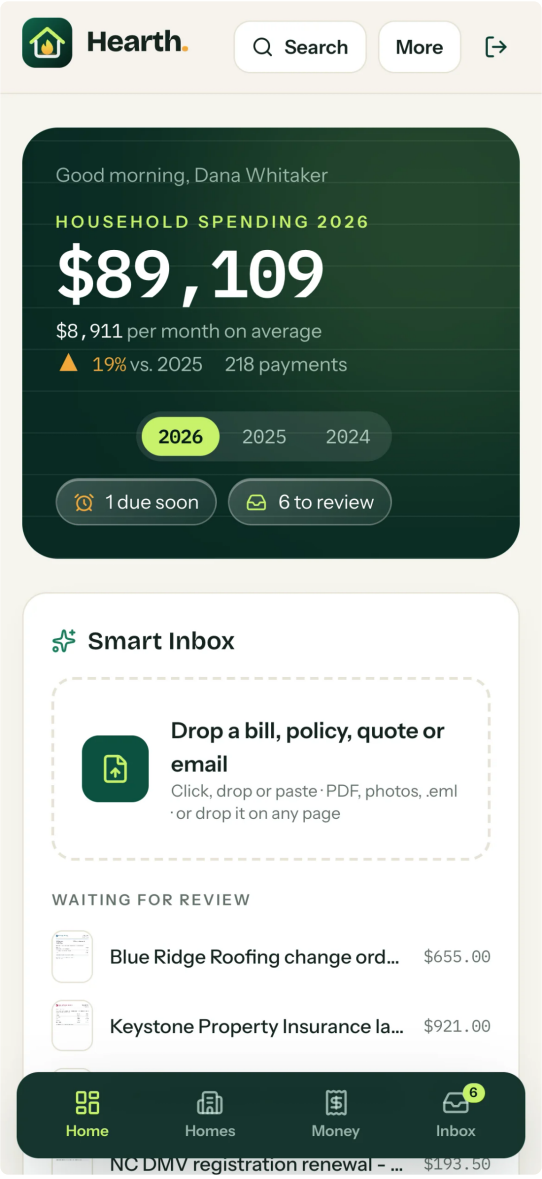
Teams und Einzelentwickler

Ich arbeite auf beide Arten, und beide sind häufig. Manche Kunden sind ein einzelner Entwickler, der ein Produkt seit Jahren trägt und einen Partner braucht, der so etwas schon gemacht hat -- jemanden, der den Plan prüft, die Teile übernimmt, die nicht zu den eigenen Stärken passen, oder beim ersten Modul mit ihm zusammen programmiert. Andere sind Teams, die den modernen Stack und einen funktionierenden KI-gestützten Prozess in ihr eigenes Repository bringen möchten, wobei ihre eigenen Leute die Arbeit machen und ich begleitend

reviewe. In beiden Fällen findet die Arbeit in einem Repository statt, das Sie vom ersten Tag an einsehen können, in kurzen Releases, die Sie selbst ausführen können.

Der Einstieg ist immer derselbe und kostet, wie es auf meiner Seite zu meinen Leistungen steht, nichts: ein

Gespräch von dreißig bis sechzig Minuten über das System und darüber, wo es wehtut. Bei bestehendem Code ist der nächste Schritt eine Analyse mit festem Umfang -- ich lese den Code, kompiliere ihn, führe ihn aus, und Sie erhalten einen schriftlichen Plan und eine realistische Schätzung, die Ihnen gehören, ob wir weitermachen oder nicht. Weitere Systeme, die ich ausgeliefert habe, finden Sie unter Work.



Die Web-App auf dem Smartphone: dasselbe System, aufbereitet für den kleinen Bildschirm.

Unterm Strich

Hearth begann als Geburtstagsgeschenk und wurde zum besten Argument für Individualsoftware, das ich vorbringen kann. In zwölf Tagen hat ein erfahrener Entwickler mit KI gebaut, wofür die Lehrbücher irgendwo zwischen einem und mehreren Personenjahren ansetzen: eine KI, die Dokumente liest und ablegt, mit jedem beliebigen Anbieter, Passkeys und einen verschlüsselten Tresor sowie native Apps für iPhone, iPad, Apple

TV und den Mac, dazu Chrome. Die KI hat die Entscheidungen nicht getroffen; sie hat ihre Umsetzung billig gemacht.

> Die Frage ist nicht mehr, ob Sie sich Individualsoftware leisten können. Sondern ob Sie es sich leisten können, Ihr Geschäft weiter um die Software eines anderen herumzubiegen.

Wenn Sie eine Delphi-Anwendung haben, die eine Zukunft verdient, oder einen Prozess, zu dem kein Produkt passt, würde ich sehr gern davon hören. Sprechen wir darüber.

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 —