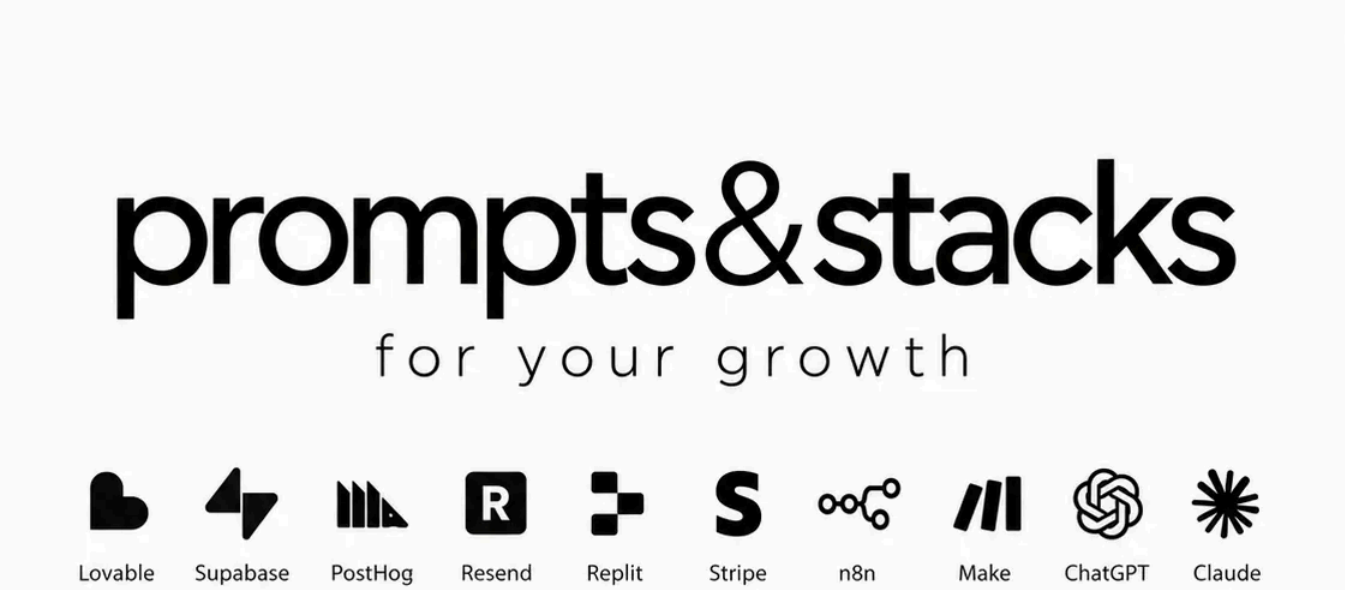
Dicas e ferramentas de IA #1 — outubro de 2026

28 de set. de 2026 · 12 min de leitura · Por Viktor Zaika

Três combinações práticas de ferramentas para testar uma ideia, criar um MVP e automatizar o trabalho, com prompts, diagramas e guias para baixar.

Ferramentas de IA | Gerenciamento de produtos | MVP | Automação | Sem código



Ser capaz de criar o seu próprio projeto não é mais a capacidade dos engenheiros sozinhos. Os bons gestores de produtos devem ser capazes de construir os PoCs, até mesmo os MVPs e o teste manual e de ter algo que os ajude a comunicar as ideias. Não só isso, mas virtualmente qualquer pessoa expert em tecnologia hoje em dia tem que usar alguns condutores de produtividade pessoal que aumentam seu desempenho, dar- lhes mais tempo livre em um dia, ajudá- los a idear, aumentar sua observabilidade online, etc. Então, aqui vou dar-lhe as dicas mais essenciais, fáceis de ler e fáceis de usar, que vão ajudá- lo imensamente nestes cenários:

Lovável Página + UI	**Supabase** Dados + Auth	**PostHog** Eventos + reprodução	**Reenviar** E-mail
Replicar Aplicativos Backend	**n8n** Fluxos de trabalho	**Marcar** Integrações	**Limpar** Pagamentos

As ferramentas discutidas abaixo. Toque em qualquer carta para a documentação ou página de integração do produto. O ícone Lovable é de Lovable; os outros são de Icones Simples.

1. Você quer testar a ideia, não construir a empresa

Digamos que você tem uma ideia para algum SaaS, mercado, ferramenta de IA, que seja. O erro aqui seria começar imediatamente a construir tudo. Você não sabe se alguém ainda o quer.

Minha pilha para isso seria:

Lovável + Supabase + PostHog + Reenvia

Lovável para a página e talvez alguma interação muito básica. Supabase para armazenar inscrições e qualquer pequeno backend que você precisar. PostHog para realmente entender o que as pessoas estão fazendo em vez de apenas olhar para o contador de visualização de páginas. Reenviar para e-mails de confirmação, listas de espera, convites de acesso precoces, etc.

PostHog é particularmente útil aqui porque você pode ir dos números básicos de tráfego para metas de conversão e funil e mesmo Reproduções de sessão, significando que você pode literalmente ver onde as pessoas clicam, onde ficam confusas e onde desaparecem.

Um truque realmente útil aqui é falsificar a parte cara antes de construí- la.

Digamos que a sua ideia é um serviço de IA que analisa um PDF e retorna um relatório complicado. A sua página de destino já pode permitir ao usuário carregar o PDF e pressionar o botão Analyze. Atrás das cenas você pode receber o arquivo, fazer as primeiras 10 solicitações manualmente e enviar o resultado de volta por e- mail.

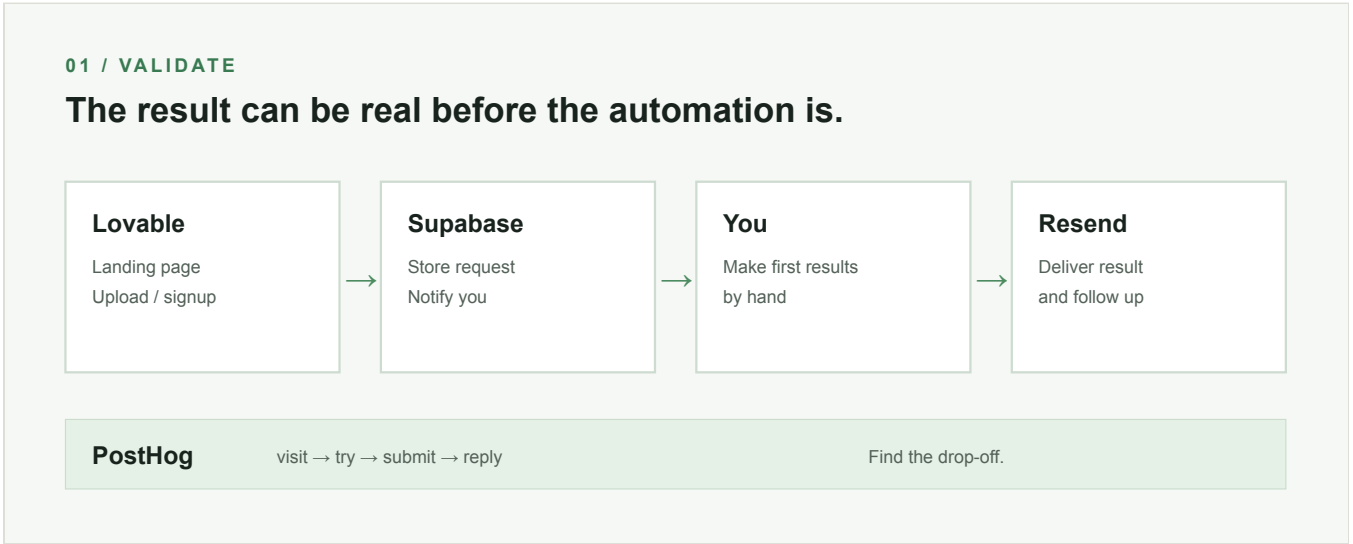
Sim, é automação falsa.

Mas você está testando se as pessoas realmente querem o resultado antes de passar duas semanas construindo o sistema que o cria automaticamente.

Para este cenário em particular:

Página de aterragem → inscrição/upload → Supabase → notificação para você → criar manualmente o resultado → Reenviar e-mail

Uma vez que as pessoas realmente começarem a usá- lo, automatize o meio.



Um possível fluxo de trabalho de validação. O resultado feito à mão testa a demanda antes da automação cara existir. Integração de Supabase Lovável, Funais PostHog, e Reenviar API de e-mail suportar as peças mostradas aqui. Diagrama do Viktor Zaika. Toque na imagem para vê- la em tamanho completo.

Outra coisa útil: rastrear eventos, não apenas visitas.

Não meça apenas:

1.400 pessoas visitadas

Medir:

382 clicou em tentar
117 iniciou o formulário
83 terminou
19 respondeu ao e-mail

Isso diz-lhe muito mais sobre se a ideia é interessante. Se você tem o evento real conta, Calculadora de funil de conversão pode mostrar onde a maior queda ocorre antes de colocar esses passos em um funil PostHog.

Validar a sua ideia Copiar prompt ↗

Quero validar esta ideia do produto: [DESCRIBE IDEA].
Eu NÃO quero construir o produto completo ainda.
Desenhe a menor arquitetura de validação possível usando Lovable, Supabase, PostHog e Reenvia.
Explicar:
1. o que realmente deve ser construído
2. o que pode ser falsificado manualmente nos bastidores
3. que dados devem ser armazenados em Supabase
4. quais eventos PostHog eu devo rastrear
5. que funil de conversão eu deveria criar
6. que e-mails devem ser enviados através do Reenvio
7. o que contaria como evidência suficiente para prosseguir para um MVP
Mantenha a arquitetura intencionalmente simples. Não introduza serviços adicionais a menos que haja uma boa razão.

2. Você realmente precisa de um aplicativo de trabalho agora

Uma vez que você precisa fazer login, salvar algo, voltar amanhã, talvez pagar, talvez enviar arquivos, então você já está construindo um aplicativo.

Aqui eu dividiria em dois caminhos.

Caso A: SaaS ou aplicativo web normal

Lovável + Supabase + Banda + Reenviar + PostHog

Isto é honestamente suficiente para um número surpreendentemente grande de MVPs. Se você escolher esta pilha, conectar um projeto Supabase que você possui ao LovableO lovable atualmente

também oferece seu próprio backend integrado por padrão.

O Supabase dá-lhe o real Postgres abaixo, mais Segurança de nível de auth e linha, para que a sua frente gerada não tenha que se tornar o seu modelo de segurança.

E esta é uma daquelas coisas técnicas chatas que os aplicativos gerados pela AI podem estragar horrivelmente: a autorização não é a mesma coisa que esconder um botão.

Se a Alice fizer login, ela não deve ser capaz de alterar algum pedido e de repente ler os registros do Bob.

Isso é onde o Supabase RLS realmente importa. O Auth identifica o usuário, enquanto o RLS decide quais linhas que o usuário está autorizado a acessar. Guia de RLS do Supabase explica como as subvenções e as políticas do banco de dados funcionam em conjunto.

Para operações lado servidor, chamadas API, Stripe webhooks ou algum passo de geração de IA, Funções de borda de Supabase são uma camada próxima bastante conveniente. Eles executam o TypeScript lateral do servidor e podem integrar- se com Auth, Postgres e APIs externas.

Um MVP muito normal poderia, portanto, parecer:

Frontend amovível → Supabase Auth → Postgres com RLS → Função de borda → API externa

Em seguida:

Stripe webhook → Função Edge → atualizar assinatura no Postgres

E:

ação do usuário → Função do Edge → Reenviar

Isso já é uma arquitetura verdadeira.

Caso B: o backend em si é o produto

Se você estiver construindo algo com trabalhadores, APIs incomuns, raspamento, processamento de arquivos, filas, pacotes Python ou alguma lógica de backend estranha, eu provavelmente me moveria para Replicar em vez de forçar tudo através de um construtor de frentes.

Replit Agent pode gerar o aplicativo a partir da linguagem natural, enquanto Replit também fornece implantação e integração de banco de dados no mesmo ambiente.

Por exemplo:

Você deseja construir uma ferramenta de monitoramento de concorrentes.

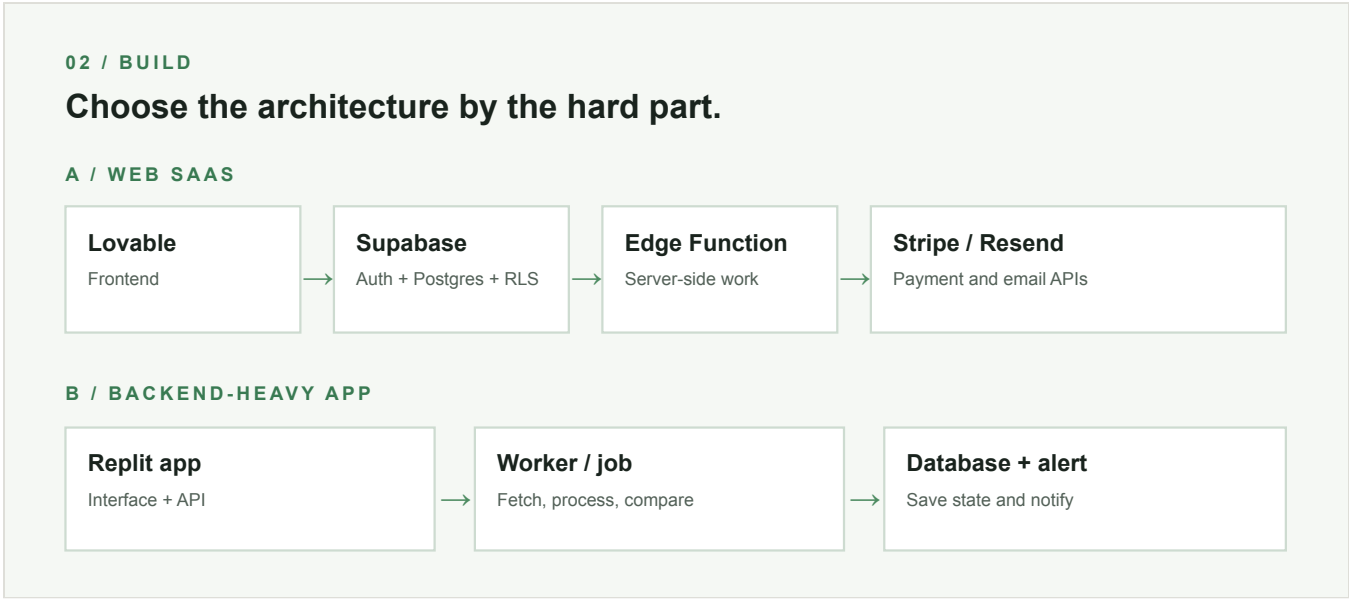
Entrada:

URLs dos concorrentes

Sistema:

obter páginas → comparar com a versão anterior → perguntar ao LLM o que materialmente mudou → salvar o resultado → enviar a notificação

Isso já é muito mais natural um aplicativo backend do que uma interface de usuário bastante gerada.



Dois caminhos MVP ilustrativos, escolhidos pelo que o produto realmente necessita. O caminho web-app usa Conexão de Supabase amovível e Funções de borda; o caminho de backend- heavy usa Agente de reposição. As setas são exemplos de arquitetura, não uma alegação que estes produtos integram automaticamente. Diagrama por Viktor Zaika. Toque na imagem para vê- la em tamanho completo.

Uma regra muito útil aqui:

Não coloque as chaves secretas da API na frente.

Tecla OpenAI, Stripe Secret, Reenviar, qualquer que seja. Se o navegador puder vê- lo, suponha que alguém possa vê- lo também.

Ponha estas operações atrás de um endpoint ou função do servidor. Guia de segurança de dados do Supabase cobre este limite do lado do servidor.

Planeje o seu MVP ï Copiar prompt ↗

Eu quero construir este MVP: [PRODUTO DE DESCRIÇÃO].
Ajude- me a escolher entre:
A. Amovível + Supabase
B. Replit
C. uma combinação de ambos
Não escolha com base em que ferramenta você gosta mais. Escolha com base nos requisitos técnicos reais.
Primeiro identificar:
1. Requisitos de frontend
2. autenticação
3. entidades e relacionamentos do banco de dados
4. regras de autorização
5. APIs externas
6. background ou trabalhos de longa duração
7. armazenamento de arquivos
8. pagamentos
9. e- mail transacional
10. análise
Em seguida, propor a menor arquitetura que ainda é tecnicamente sã.
Diga- me explicitamente quais operações devem acontecer lado servidor e quais segredos NUNCA devem ser expostos na frente.
Se usar Supabase, propor as tabelas e as regras RLS também.

3. Você quer parar de fazer algo manualmente todos os dias

Esta pode ser a categoria mais útil para a maioria das pessoas.

Você não precisa necessariamente de um produto.

Você precisa de um tubo.

O meu padrão aqui provavelmente seria:

n8n

Então conecte o que você já usar ao redor dele.

Google Drive, Gmail, Slack, Telegrama, Noção, Mesa Aérea, Supabase, OpenAI, Claude, APIs etc.

Se você quiser algo mais fácil e muito visual, Marcar é uma alternativa muito razoável e suporta milhares de integrações, incluindo conexões HTTP genéricas quando o serviço que você necessitar não é diretamente suportado.

E a parte interessante começa quando você para de pensar:

ação ativar →

e começar a pensar:

entrada → entender → decidir → ato → lembrar

Por exemplo, aqui está um conduto pessoal genuinamente útil:

Gmail → n8n → LLM → classificar → extrair prazo → criar tarefa → Notificação de telegrama

Mas não peça ao LLM que simplesmente leia o meu e- mail e decida tudo.

Faça com que retorne dados estruturados:

```
{
  "important": true,
  "category": "invoice",
  "deadline": "2026-10-14",
  "action_required": "pay invoice",
  "confidence": 0.94
}
```

Copy

Agora o fluxo de trabalho pode realmente usar a saída de forma confiável. n8nÕs Parser estruturado de saída é uma forma de impor a forma de um resultado.

Isso é um pequeno detalhe técnico que torna as automações da AI muito menos frágiles.

Outro:

Não deixe que a IA realize ações irreveríveis diretamente sempre que puder evitá- la.

Por exemplo:

e-mail → AI pensa que isto é spam → DELETE

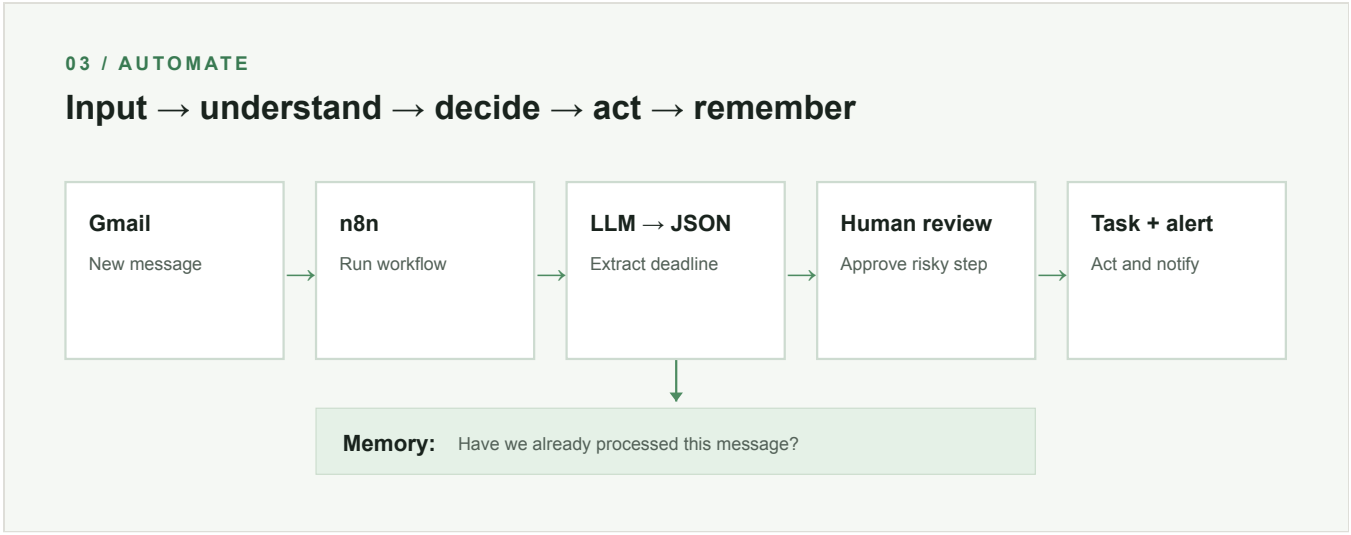
Mau.

Melhor:

E-mail → AI pensa que isso é spam → rótulo ïprovavelmente spam ♫ → você confirma

Igual ao envio de e- mails, publicação de conteúdo, exclusão de arquivos, atualização de dados de produção, etc.

Colocar um passo de aprovação humana onde o custo de uma decisão errada é elevado.



Um fluxo de trabalho diário ilustrativo. Saída estruturada Permite que os passos posteriores usem campos; revisão humana mantém as ações de alto custo sob o seu controle. Diagrama por Viktor Zaika. Toque na imagem para vê- la em tamanho completo.

Alguns pipelines realmente úteis

Oleoduto de pesquisa pessoal

RSS / sites / newsletters → n8n → extrair texto → deduplicar → LLM classifica relevância → salvar interessantes → um digestivo de Telegrama

Em vez de ler 80 coisas, você leu 7.

Reunião do pipeline

A reunião do calendário termina → aparece a transcrição → O LLM extrai decisões e ações → crie tarefas → envie um resumo curto

Conteúdo do pipeline

Ideia no Telegrama → n8n → salvar em Noção → LLM categoriza → gera perguntas de pesquisa → espera

Note que eu não estou dizendo "gerar o post".

A parte útil pode ser simplesmente remover o trabalho organizacional estúpido em torno de escrevê- lo.

Observabilidade online

Google Alertas / Reddit pesquisa / X ou APIs externas / análises de produtos → fluxo de trabalho → remover duplicados → classificar sentimento/tópico → notificar apenas quando algo interessante acontece

Então, em vez de procurar no Google, empresa ou concorrente a cada poucos dias, o sistema faz essa parte.

4. A parte verdadeiramente interessante: deixe a IA produzir dados, não prosa

Esta é provavelmente uma das maiores diferenças entre uma automação divertida e algo em que você pode realmente depender.

Não pergunte:

Leia este ticket de suporte e me diga o que fazer.

Pergunte:

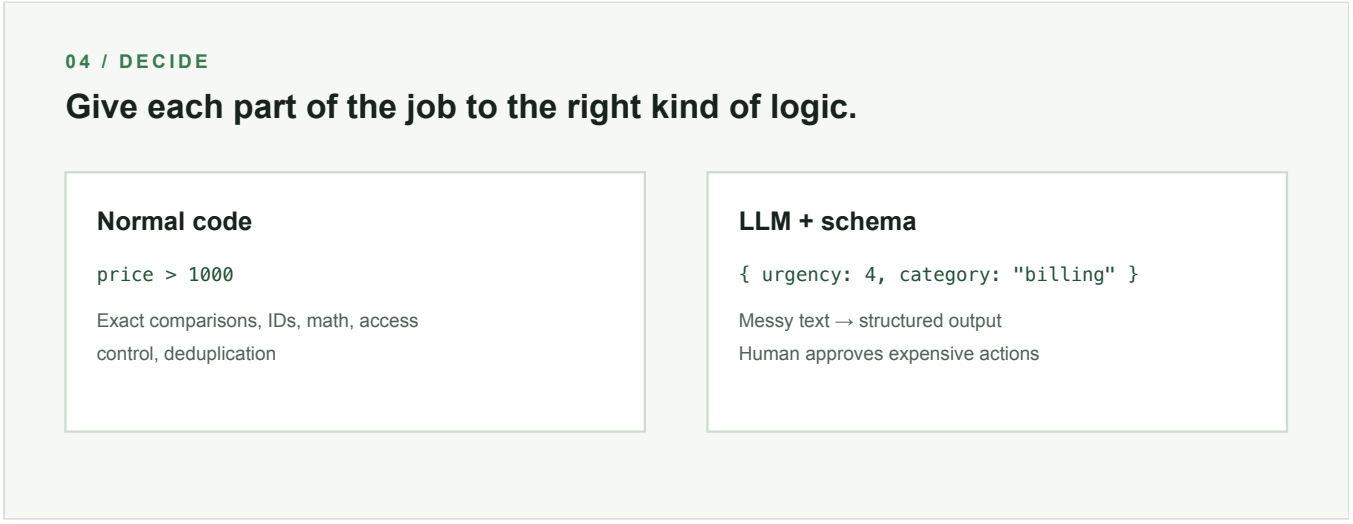
```
Analyze this support ticket and return ONLY valid JSON using this schema:
category: billing | bug | feature_request | account | other
urgency: 1 to 5
requires_human: boolean
customer_sentiment: positive | neutral | negative
summary: maximum 200 characters
```

Copy

Agora, outra parte do seu gasoduto pode realmente encaminhar.

urgência >= 4 → Slack
faturamento → fila de faturamento
requires_human = false → gerar rascunho

Os LLMs são muito mais úteis dentro do software quando são tratados como uma função ligeiramente confiável dentro de um sistema determinístico maior, ao invés de algum cara mágico que controla tudo.



Usar código normal para regras exatas e um LLM para entradas fuzzy. O lado JSON ilustra o tipo de campos um n8n etapa de saída estruturada pode passar ao longo. Diagrama por Viktor Zaika. Toque na imagem para vê- la em tamanho completo.

5. E dê a sua memória de fluxo de trabalho

Deixe que diga que você cria um tubo de pesquisa de IA.

Sem memória:

Segunda- feira encontra o artigo X.
Terça- feira encontra o artigo X novamente.
Quarta-feira encontra alguém discutindo o artigo X e com orgulho diz- lhe novamente.

Armazene o que você já processou.

Isto pode literalmente ser uma tabela Supabase:

```
source_url
hash
first_seen
summary
topic
```

Copy

Então antes de processar algo caro:

Já vi isso?

Se sim, pule- o.

Muito simples. De repente, o seu oleoduto sente- se cerca de 5 vezes mais inteligente.

6. Não faça tudo automaticamente IA

Isto soa estúpido num post chamado Dicas & Ferramentas de IA, mas importa.

Se isso funcionar:

```
price > 1000
```

Copy

Não o substitua por:

```
Ask GPT whether this price appears relatively expensive in the context of the transact
```

Copy

Por favor.

O código normal é mais rápido, mais barato e determinístico.

Use o LLM onde a entrada é fuzzy.

Entender a linguagem.
Classificação onde as regras se tornam horríveis.
Sumarização.
Extração de documentos confusos.
Correspondendo as coisas pelo significado.
Gerando algo.

Deixe matemática, IDs, comparações exatas, controle de acesso e regras de negócios mais chatas para chatear o código normal.

O aborrecimento é fantástico quando funciona sempre.

As três pilhas que o IPd realmente lembra

Teste se alguém se importa

Lovável + Supabase + PostHog + Reenvia

Construir algo que as pessoas possam realmente usar

Lovável + Supabase para web SaaS normal

ou

Replit quando a lógica do backend se torna a parte interessante

Em seguida, adicione Stripe, Reenviar e PostHog quando você realmente precisar deles.

Automatizar o seu próprio trabalho

n8n + as ferramentas que você já usa + LLM + um pequeno banco de dados para memória

E provavelmente o prompt mais útil deste post inteiro:

Automatizar o seu trabalho Copiar prompt ↗

Atualmente faço isso manualmente:
[DESCRIBE THE PROCESS IN NORMAL HUMAN LANGUAGE]
Desenho de uma automação para ele.
Separe o fluxo de trabalho em:
passos determinísticos que devem usar a lógica normal,
passos confusos onde um LLM realmente faz sentido,
ações que devem requerer aprovação humana,
o estado que precisa ser armazenado para que o fluxo de trabalho se lembre do que já fez.
Prefere n8n e serviços que já usei. Mantenha o número de peças móveis o mais pequeno possível.
Para cada passo do LLM, definir a saída exata estruturada do JSON que deve retornar.
Também me diga como este fluxo de trabalho pode falhar e o que devo registrar para que possa debugá- lo mais tarde.

Porque este é o ponto inteiro. A IA não significa que você tenha que se tornar um engenheiro de repente. Mas também não significa que a engenharia tenha parado de importar.

Agora você pode construir uma quantidade absurda de coisas sem saber tudo por baixo dela. Você só precisa saber o suficiente para entender onde as partes perigosas estão, onde a IA é útil e onde você realmente não deve deixá- lo estilo livre.