

Conseils et outils IA n° 1 — octobre 2026

28 sept. 2026 · 12 min lire · Par [Viktor Zaika](#)

Trois combinaisons d'outils pour tester une idée, créer un MVP et automatiser votre travail, avec des prompts, des schémas de workflow et des guides à télécharger.

- Outils d'IA
- Gestion de produit
- MVP
- Automatisation
- Sans code



Être capable de créer son propre projet n'est plus l'apanage des seuls ingénieurs. Les bons chefs de produit doivent être capables de créer des PoC, même des MVP, de tester manuellement et de disposer de quelque chose qui les aide à communiquer les idées. Non seulement cela, mais pratiquement toutes les personnes férues de technologie doivent aujourd'hui utiliser des pipelines de productivité personnels qui augmentent leurs performances, leur donnent plus de temps libre dans une journée, les aident à imaginer, augmentent leur observabilité en ligne, etc. Je vais donc vous donner ici les conseils les plus essentiels, faciles à lire et à utiliser, qui vous aideront énormément dans ces scénarios :

 Lovable Page + interface utilisateur	 Supabase Données + Auth	 PostHog Événements + replay	 Resend Courriel
 Replit Applications de base	 n8n Workflows	 Marque Intégrations	 Stripe Paiements

Les outils discutés ci-dessous. Appuyez sur n'importe quelle carte pour accéder à la documentation ou à la page d'intégration de ce produit. L'icône de Lovable vient de Lovable ; les autres proviennent d'icônes simples.

1. Vous voulez tester l'idée, pas construire l'entreprise

Disons que vous avez une idée pour un SaaS, une place de marché, un outil d'IA, etc. L'erreur ici serait de commencer immédiatement à tout construire. Vous ne savez pas encore si quelqu'un le veut.

Ma pile pour ça serait :

Lovable + Supabase + PostHog + Resend

Lovable pour la page et peut-être une interaction très fondamentale. Supabase pour stocker les inscriptions et tout petit backend dont vous avez besoin. PostHog pour réellement comprendre ce que font les gens au lieu de simplement regarder le compteur de pages vues. Resend pour les e-mails de confirmation, les listes d'attente, les invitations à accès anticipé, etc.

PostHog est particulièrement utile ici car vous pouvez passer des chiffres de trafic de base aux objectifs de conversion et entonnoir et même lectures de session, ce qui signifie que vous pouvez littéralement voir où les gens cliquent, où ils sont confus et où ils disparaissent.

Une variante vraiment utile consiste à simuler la pièce coûteuse avant de la construire.

Disons que votre idée est un service d'IA qui analyse un PDF et renvoie un rapport compliqué. Votre page de destination peut déjà permettre à l'utilisateur de télécharger le PDF et d'appuyer sur « Analyser ». En coulisses, vous pouvez recevoir le fichier, effectuer les 10 premières demandes manuellement et renvoyer le résultat par courrier électronique.

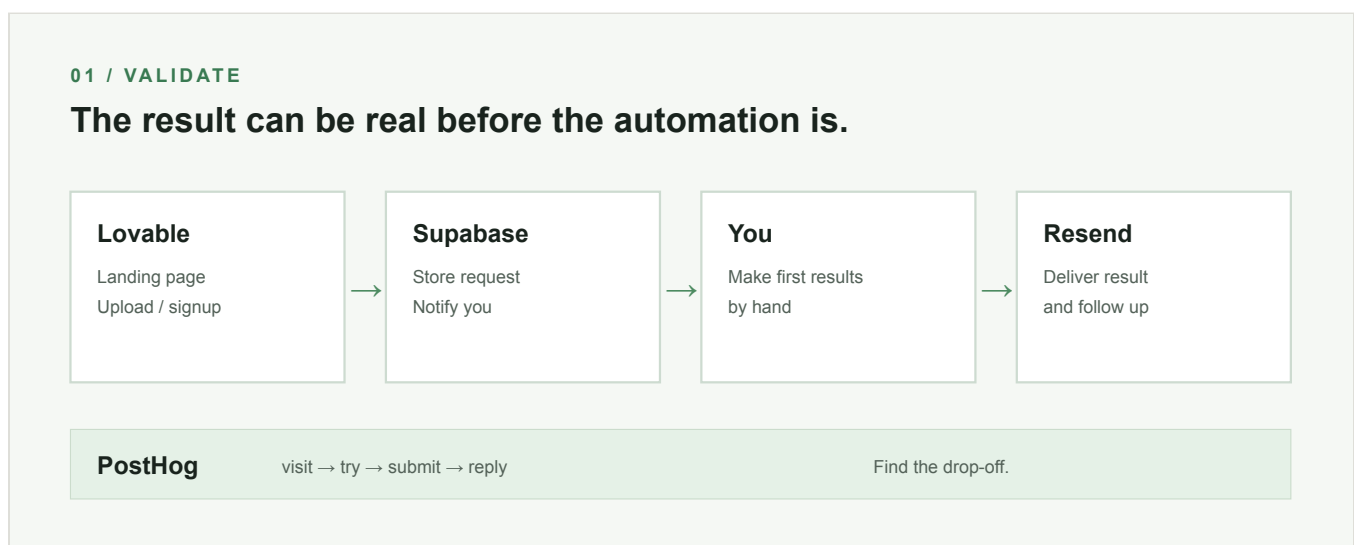
Oui, c'est une fausse automatisation.

Mais vous testez si les gens veulent réellement le résultat avant de passer deux semaines à construire le système qui le crée automatiquement.

Pour ce scénario particulier:

Page d'accueil → inscription ou téléversement → Supabase → notification → résultat créé à la main → e-mail via Resend

Une fois que les gens commencent à l'utiliser, automatisez le milieu.



Un workflow de validation possible. Les résultats des tests faits à la main sont exigés avant que l'automatisation coûteuse n'existe. Intégration Lovable de Supabase, Entonnoirs PostHog, L'API e-mail de Resend soutenir les pièces présentées ici. Schéma de Viktor Zaika. Appuyez sur l'image pour la voir en taille réelle.

Autre chose utile : suivre les événements, pas seulement les visites.

Ne mesurez pas simplement:

1 400 personnes visitées

Mesure:

382 cliqué dessus essayez

117 a commencé le formulaire

83 l'ont terminé.

19 réponses à l'email

Cela vous en dit beaucoup plus sur l'intérêt de l'idée. Si vous avez un nombre réel d'événements, mon [Calculateur d'entonnoir de conversion](#) peut montrer où se produit la plus grande baisse avant de placer ces étapes dans un entonnoir PostHog.

Valider votre idée ĩ

Copier le prompt ↗

Je souhaite valider cette idée de produit : [DESCRIBE IDEA].

Je ne veux PAS encore créer le produit complet.

Concevez la plus petite architecture de validation possible en utilisant Lovable, Supabase, PostHog et Resend.

Expliquer:

1. que faut-il réellement construire
 2. Qu'est-ce qui peut être simulé manuellement en coulisses
 3. quelles données doivent être stockées dans Supabase
 4. quels événements PostHog je dois suivre
 5. quel entonnoir de conversion dois-je créer
 6. quels e-mails doivent être envoyés via Resend
 7. Qu'est-ce qui compterait comme preuve suffisante pour passer à un MVP
- Gardez l'architecture intentionnellement simple. N'introduisez pas de services supplémentaires à moins d'avoir une très bonne raison.

2. Vous avez réellement besoin d'une application de travail maintenant

Une fois que vous avez besoin que les utilisateurs se connectent, enregistrent quelque chose, reviennent demain, peut-être payer, peut-être télécharger des fichiers, alors vous créez déjà une application.

Ici, je le diviserai en deux chemins.

Cas A: SaaS ou application web normale

Lovable + Supabase + Stripe + Resend + PostHog

Honnêtement, c'est assez pour un nombre étonnamment élevé de MVP. Si vous choisissez cette pile, connecter un projet Supabase que vous possédez à Lovable; Lovable propose actuellement également son propre backend intégré par défaut.

Supabase vous donne réelle Postgres en dessous, plus Sécurité auth et au niveau des rangées, votre interface générée ne doit donc pas nécessairement devenir votre modèle de sécurité.

Et c'est l'une de ces choses techniques ennuyeuses que les applications générées par l'IA peuvent horriblement gâcher : l'autorisation n'est pas la même chose que cacher un bouton.

Si Alice se connecte, elle ne devrait pas pouvoir modifier certaines demandes et lire soudainement les enregistrements de Bob.

C'est là que Supabase RLS compte réellement. Auth identifie l'utilisateur, tandis que RLS décide à quelles lignes cet utilisateur est autorisé à accéder. Guide RLS de Supabase explique comment les subventions de base de données et les politiques fonctionnent ensemble.

Pour les opérations côté serveur, les appels API, Webhooks Stripe ou une étape de génération d'IA, Fonctions de bord Supabase sont une couche suivante assez pratique. Ils exécutent TypeScript côté serveur et peuvent s'intégrer à Auth, Postgres et aux API externes.

Un MVP très normal pourrait donc ressembler à :

Interface Lovable → Supabase Auth → Postgres avec RLS → Edge Function → API externe

Alors :

Webhook Stripe → Fonction Edge → mettre à jour l'abonnement dans Postgres

Et :

Action de l'utilisateur → Edge Function → Resend

C'est déjà une véritable architecture.

Cas B: le moteur lui-même est le produit

Si vous construisez quelque chose avec des travailleurs, des API inhabituelles, du scraping, du traitement de fichiers, des files d'attente, des packages Python ou une logique backend étrange, je passerais probablement à Replit plutôt que de tout forcer à travers un constructeur frontalier.

Replit Agent peut générer l'application à partir du langage naturel, tandis que Replit fournit également déploiement et intégration des bases de données dans le même environnement.

Par exemple :

Vous voulez construire un outil de surveillance des concurrents.

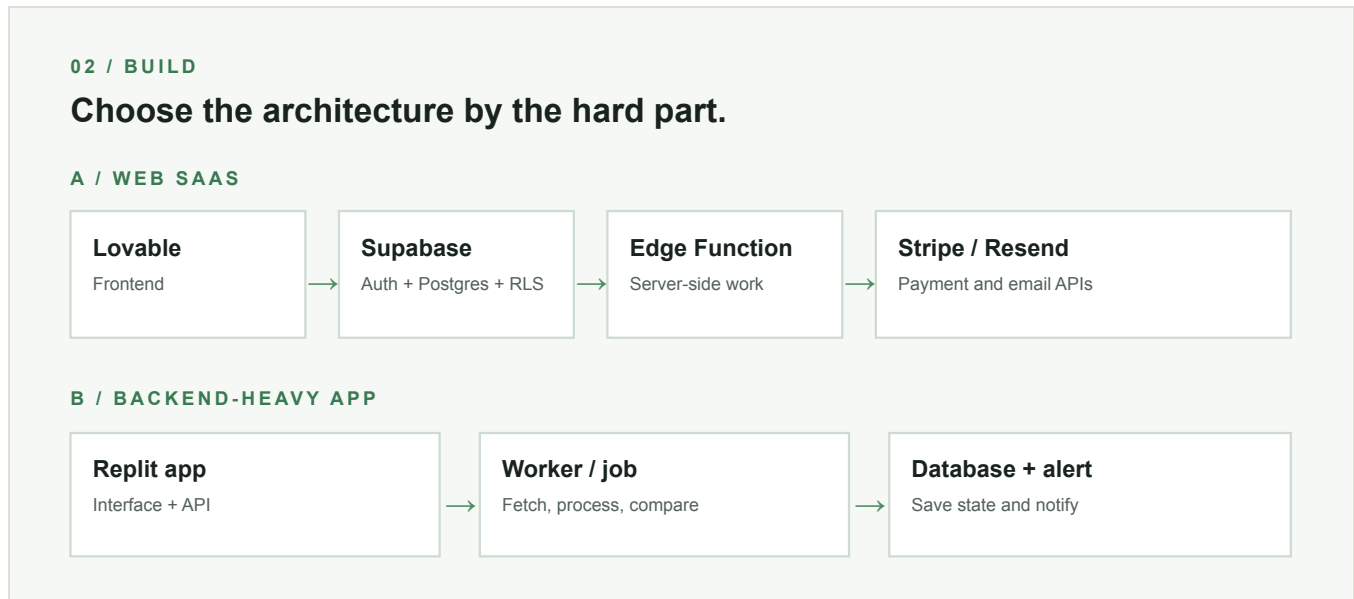
Entrée:

URL des concurrents

Système:

recupérer les pages → comparer avec la version précédente → demander à LLM ce qui a sensiblement changé → enregistrer le résultat → envoyer une notification

C'est déjà beaucoup plus naturellement une application backend qu'une jolie interface utilisateur générée.



Deux chemins MVP illustratifs, choisis en fonction des besoins réels du produit. Le chemin de l'application Web utilise Connexion Supabase Lovable et Fonctions de bord; le chemin lourd du backend utilise : Agent Replit. Les flèches sont des exemples d'architecture et ne prétendent pas que ces produits s'intègrent automatiquement. Schéma de Viktor Zaika. Appuyez sur l'image pour la voir en taille réelle.

Une règle très utile ici :

Ne mettez pas vos clés API secrètes dans le frontend.

Clé OpenAI, secret Stripe, clé Resend : peu importe. Si le navigateur peut les voir, partez du principe que quelqu'un d'autre le peut aussi.

Mettez ces opérations derrière un endpoint ou une fonction du serveur. Guide de sécurité des données de Supabase couvre cette limite côté serveur.

Je souhaite créer ce MVP : [DESCRIBE PRODUCT].

Aide-moi à choisir entre :

- A. Lovable + Supabase
- B. Replit
- C. une combinaison des deux

Ne choisissez pas en fonction de l'outil que vous préférez. Choisissez en fonction des exigences techniques réelles.

Identifiez d'abord :

1. Exigences du front-end
2. authentification
3. entités et relations de base de données
4. règles d'autorisation
5. API externes
6. tâches en arrière-plan ou de longue durée
7. stockage de fichiers
8. paiements
9. e-mail transactionnel
10. analyses

Proposez ensuite la plus petite architecture encore techniquement saine.

Dites-moi explicitement quelles opérations doivent avoir lieu côté serveur et quels secrets ne doivent JAMAIS être exposés dans le frontend.

Si vous utilisez Supabase, proposez également les tables et les règles RLS.

3. Vous voulez arrêter de faire quelque chose manuellement tous les jours

Il s'agit peut-être en fait de la catégorie la plus utile pour la plupart des gens.

Vous n'avez pas nécessairement besoin d'un produit.

Vous avez besoin d'une pipeline.

Mon défaut ici serait probablement :

n8n

Alors connectez ce que vous utilisez déjà autour de lui.

Google Drive, Gmail, Slack, Telegram, Notion, Airtable, Supabase, OpenAI, Claude, API etc.

Si vous voulez quelque chose de plus facile et très visuel, Marque est une alternative très raisonnable et prend en charge des milliers d'intégrations, y compris connexions HTTP génériques lorsque le service dont vous avez besoin est directement pris en charge.

Et la partie intéressante commence quand vous arrêtez de penser:

déclencheur → action

et commencez à penser:

entrée → comprendre → décider → agir → se souvenir

Par exemple, voici un pipeline personnel vraiment utile :

Gmail → n8n → LLM → classer → extraire la date limite → créer une tâche → notification par télégramme

Mais ne demandez pas au LLM de simplement « lire mon courrier électronique et tout décider ».

Faites-le retourner des données structurées:

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

Copy

Maintenant, le flux de travail peut effectivement utiliser la sortie de manière fiable. n8n's Structured Output Parser est une façon de faire respecter la forme d'un résultat.

C'est un tout petit détail technique qui rend les automatisations de l'IA beaucoup moins fragiles.

Un autre :

Ne laissez pas l'IA effectuer directement des actions irréversibles chaque fois que vous pouvez l'éviter.

Par exemple :

email → AI pense que c'est du spam → DELETE

Mauvais.

Mieux :

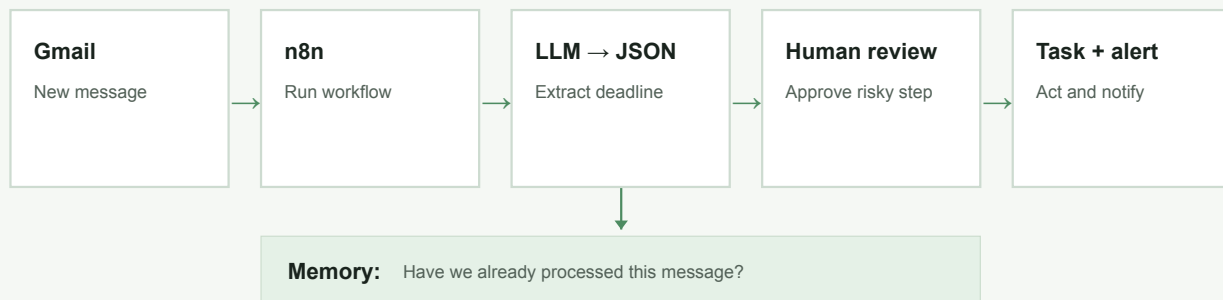
email → L'IA pense qu'il s'agit d'un spam → étiquetez « probablement du spam » → vous confirmez

Idem pour l'envoi d'e-mails, la publication de contenu, la suppression de fichiers, la mise à jour des données de production, etc.

Mettez un étape d'approbation humaine où le coût d'une mauvaise décision est élevé.

03 / AUTOMATE

Input → understand → decide → act → remember



Un workflow quotidien illustratif. Produit structuré permet aux étapes ultérieures d'utiliser les champs; examen humain garde les actions coûteuses sous votre contrôle. Schéma de Viktor Zaika. Appuyez sur l'image pour la voir en taille réelle.

Quelques pipelines réellement utiles

Travaux de recherche

RSS / sites Web / newsletters → n8n → extraire le texte → dédupliquer → pertinence des classements
LLM → enregistrer les plus intéressants → un résumé Telegram

Au lieu de lire 80 choses, vous lisez 7.

Gaine de réunion

La réunion du calendrier se termine → la transcription apparaît → LLM extrait les décisions et les actions
→ crée des tâches → envoie un bref résumé

Contenu du pipeline

Idée dans Telegram → n8n → enregistrer dans Notion → LLM la catégorise → génère des questions de recherche → attend

Notez que je ne dis pas "générer le post".

La partie utile pourrait simplement être de supprimer le stupide travail d'organisation entourant sa rédaction.

Observabilité en ligne

Alertes Google / Recherche Reddit / API X ou externes / avis sur les produits → workflow → supprimer les doublons → classer les sentiments/sujets → notifier uniquement lorsque quelque chose d'intéressant se produit

Ainsi, au lieu de rechercher sur Google vous-même, votre entreprise ou votre concurrent tous les quelques jours, le système se charge de cette partie.

4. La partie vraiment intéressante : laisser l'IA produire des données, pas de la prose

C'est probablement l'une des plus grandes différences entre une automatisation amusante et quelque chose sur lequel vous pouvez réellement compter.

Ne demandez pas :

Lisez ce billet de soutien et dites-moi ce qu'il faut faire.

Demandez :

```
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

Maintenant, une autre partie de votre pipeline peut effectivement l'axer.

Urgence $\geq 4 \rightarrow$ Slack

facturation $\rightarrow$ queue de facturation

requires_human = false $\rightarrow$ générer une ébauche

Les LLM sont beaucoup plus utiles dans les logiciels lorsqu'ils sont traités comme une fonction légèrement peu fiable au sein d'un système déterministe plus vaste, plutôt que comme un magicien contrôlant le tout.

04 / DECIDE

Give each part of the job to the right kind of logic.

Normal code

```
price > 1000
```

Exact comparisons, IDs, math, access
control, deduplication

LLM + schema

```
{ urgency: 4, category: "billing" }
```

Messy text $\rightarrow$ structured output
Human approves expensive actions

Utilisez du code normal pour les règles exactes et un LLM pour les entrées floues. Le côté JSON illustre le type de champs et n8n étape de sortie structurée peut transmettre. Schéma de Viktor Zaika. Appuyez sur l'image pour la voir en taille réelle.

5. Et donner votre mémoire de flux de travail

Laissez-nous dire que vous créez un pipeline de recherche sur l'IA.

Sans mémoire :

Lundi, il trouve l'article X.

Mardi, il retrouve l'article X.

Mercredi, il trouve quelqu'un en train de discuter de l'article X et vous le répète fièrement.

Enregistrez ce que vous avez déjà traité.

Ceci peut être littéralement une table Supabase :

```
source_url
hash
first_seen
summary
topic
```

Copy

Ensuite, avant de traiter quelque chose de cher:

J'ai déjà vu ça ?

Si oui, sautez-le.

Très simple. Soudain, votre pipeline semble environ 5 fois plus intelligent.

6. Ne pas faire automatiquement tout IA

Cela semble stupide dans un article intitulé AI Tips&Tools, mais c'est important.

Si cela fonctionne :

```
price > 1000
```

Copy

ne pas le remplacer par:

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

Copy

S'il te plaît.

Le code normal est plus rapide, moins cher et déterministe.

Utilisez le LLM où l'entrée est floue.

Comprendre le langage.

Classification où les règles deviennent horribles.

Résumé.

Extraction de documents désastreux.

- Il correspond aux choses par sens.

Générer quelque chose.

Laissez les mathématiques, les identifiants, les comparaisons exactes, le contrôle d'accès et les règles commerciales les plus ennuyeuses au code normal ennuyeux.

Ennuyeux est fantastique quand il fonctionne à chaque fois.

Les trois combinaisons d'outils à retenir

Testez si quelqu'un s'en soucie

Lovable + Supabase + PostHog + Resend

Construire quelque chose que les gens peuvent réellement utiliser

Lovable + Supabase pour le web SaaS normal

ou

Replit lorsque la logique backend devient la partie centrale

Ajoutez ensuite Stripe, Resend et PostHog lorsque vous en avez réellement besoin.

Automatisez votre propre travail

n8n + les outils que vous utilisez déjà + LLM + une petite base de données pour la mémoire

Et probablement le prompt le plus utile à partir de tout ce post:

Automatisez votre travail ĩ

[Copier le prompt ↗](#)

Actuellement, je le fais manuellement :

[DESCRIBE THE PROCESS IN NORMAL HUMAN LANGUAGE]

Concevez une automatisation pour cela.

Séparez le flux de travail en :

des étapes déterministes qui devraient utiliser la logique normale,

les étapes ambiguës où un LLM est réellement utile,

des actions qui devraient nécessiter l'approbation humaine,

état qui doit être stocké pour que le flux de travail se souvienne de ce qu'il a déjà fait.

Préférez n8n et les services que j'utilise déjà. Gardez le nombre de pièces mobiles aussi petit que possible.

Pour chaque étape LLM, définissez la sortie JSON structurée exacte qu'elle doit renvoyer.

Dites-moi également comment ce flux de travail peut échouer et ce que je dois enregistrer afin de pouvoir le déboguer plus tard.

Parce que c'est un peu tout le problème. L'IA ne signifie pas que vous devez soudainement devenir ingénieur. Mais cela ne signifie pas non plus que l'ingénierie n'a plus d'importance.

Vous pouvez désormais construire une quantité absurde de choses sans tout connaître. Vous avez juste besoin d'en savoir suffisamment pour comprendre où se trouvent les parties dangereuses, où l'IA est utile et où vous ne devriez vraiment pas la laisser faire du freestyle.