Les prompts pour *cadrer et tester*

Les prompts et la règle à coller, dans l'ordre du tuto, pour construire un outil interne avec Claude Code sans usine à gaz ni faille oubliée. Remplacez chaque **{{variable}}**.

Il vous faut · Un abonnement Claude Pro, Claude Code installé et ouvert dans un dossier vide, une idée d'outil et 30 minutes avec ses futurs utilisateurs.

LES 1 OUTILS À BRANCHER

OUTIL 1

La règle anti-dérive (CLAUDE.md)

Empêche Claude Code d'ajouter des fonctions absentes du PRD.

Demandez à Claude Code de créer le fichier CLAUDE.md avec ce texte :

```
## Règle de périmètre
Le fichier PRD.md fait foi.
Tu ne développes aucune fonctionnalité
absente de sa section "Fonctionnalités
de la version 1". Sinon : tu l'ajoutes
à BACKLOG.md avec la date, tu me le dis,
et tu attends que j'écrive
"j'autorise : {{nom_de_la_fonction}}".
```

Contrôle : demandez une fonction hors PRD, elle part dans BACKLOG.md.

LES 7 PROMPTS

Les prompts *dans l'ordre*

Les deux premiers se collent dans une conversation Claude, les suivants dans Claude Code, ouvert dans le dossier du projet.

Prompt 1 · La fiche besoin

Je veux créer un outil interne pour **{{description_courte_de_l_outil}}**.
Voici les utilisateurs que j'envisage et leur problème actuel :
{{liste_utilisateurs_et_problemes}}

Pose-moi une question à la fois pour vérifier, pour chaque utilisateur : son vrai problème, ce que l'outil lui ferait gagner, et s'il en a réellement besoin. À la fin, rédige une fiche besoin d'une page : utilisateurs retenus, problème de chacun, 3 fonctionnalités maximum qui y répondent.

Une question à la fois : répondez honnêtement, rayez les utilisateurs à qui l'outil n'apporte rien.

Prompt 2 · Le PRD avec sa section hors cadre

À partir de la fiche besoin ci-dessus, rédige un PRD (product requirement document) en Markdown avec ces sections :
1. Objectif de l'outil en une phrase
2. Utilisateurs et leur problème
3. Fonctionnalités de la version 1 (3 maximum), chacune avec un critère de réussite vérifiable
4. Hors cadre : tout ce qui ne sera pas fait en version 1, même si c'est tentant
5. Données manipulées et qui a le droit de voir quoi
6. Niveau d'ambition (à compléter à l'étape suivante)
Écris simplement, pour une personne qui ne code pas.

Enregistrez le résultat sous PRD.md dans le dossier du projet.

Prompt 3 · Le niveau d'ambition

Niveau d'ambition
- Utilisateurs : **{{nombre_utilisateurs}}**
- Usage : **{{local_ou_en_ligne}}**
- Connexion par compte : **{{oui_ou_non}}**
- Perte des données : **{{acceptable_ou_grave}}**
Consigne : choisis l'architecture la plus simple qui répond à ce niveau. Avant d'ajouter un serveur, une base de données, un service payant ou un déploiement automatique, explique-moi pourquoi et attends mon accord.

À coller dans la section 6 du PRD, rempli.

Prompt 4 · Construire une étape à la fois

> Lis PRD.md et CLAUDE.md. Découpe la version 1 en étapes courtes, chacune avec ce que je pourrai vérifier moi-même à l'écran. Réalise uniquement l'étape 1, puis arrête-toi et dis-moi comment la tester. Pour toute partie invisible (base de données, gestion des erreurs, accès), explique en deux phrases ce que tu as fait et ce qui pourrait mal tourner.

Testez ce qui s'affiche avant d'écrire "étape suivante".

Prompt 5 · L'audit des quatre angles morts

> Fais un audit de sécurité simple de ce projet, en langage non technique, sur quatre points :
> 1. Droits d'accès : un utilisateur peut-il lire des données qui ne le concernent pas en contournant l'interface (adresse directe, appel à l'API) ? Si une base Supabase est utilisée, la sécurité par ligne (Row Level Security) est-elle activée sur chaque table ?
> 2. Clés secrètes : y a-t-il une clé, un mot de passe ou un jeton écrit dans le code ou dans un fichier envoyé sur GitHub ?
> 3. Pannes : que se passe-t-il si le réseau coupe, si un formulaire est mal rempli, si un service externe ne répond pas ? Des données peuvent-elles être perdues ?
> 4. Dépendances : liste les bibliothèques ajoutées et dis lesquelles pourraient casser le projet lors d'une mise à jour.
> Pour chaque point : OK ou à corriger, et pourquoi.

Relancez l'audit après chaque correction. Données sensibles en ligne : faites relire par un développeur.

Prompt 6 · Le plan de test hors chemin parfait

> Rédige un plan de test pour la version 1, sous forme de tableau : action, résultat attendu, résultat obtenu (vide). Inclus : champs vides, fautes de frappe, coupure réseau au milieu d'une action, **{{volume_realiste}}** enregistrements chargés, et un test par rôle d'utilisateur pour vérifier qu'il ne voit que ses données. Ajoute des données de test fictives pour ce volume.

Créez au moins deux comptes de rôles différents.

Prompt 7 · Test, production et sauvegarde

> Avant de publier, mets en place deux environnements séparés : test et production, avec deux bases de données distinctes et un nom visible "TEST" sur la version de test. Explique-moi comment faire une sauvegarde des données de production et comment la restaurer. Propose l'hébergement le plus simple adapté au niveau d'ambition du PRD, avec son coût mensuel.

Restaurez une sauvegarde au moins une fois avant d'en avoir besoin.

AVANT D'OUVRIR L'OUTIL

Sauvegardez les données, vérifiez que la version de test porte bien la mention TEST, ouvrez l'accès à deux ou trois vrais utilisateurs, corrigez, puis ouvrez au reste de l'équipe. Les nouvelles idées vont dans BACKLOG.md pour la version 2.

TOUS LES LIENS

Claude et Claude Code	claude.com
Documentation Claude Code	docs.claude.com
Supabase (sécurité par ligne)	supabase.com/docs
Le tuto complet	formation-chatgpt.ai/tuto-claude-code-cadrer-tester-app/

Le réflexe qui débloque tout

Ne collez jamais une clé secrète ni de vraies données clients dans Claude ou Claude Code : demandez où créer le fichier de configuration local et saisissez la clé vous-même.

Le tuto complet : https://formation-chatgpt.ai/tuto-claude-code-cadrer-tester-app/