Eigenflow

GUIDE DU REEL · SITE

Un reel donné à GPT-6 Astra

Le site, la vidéo, le pack et les prompts exacts

PAR

Abyane Nabeebaccus

INSTAGRAM

@abyane_nb

ÉDITION

novembre 2026

Tu as commenté SITE. Tu trouveras ici le lien du site, les prompts mot pour mot, la vidéo et le pack à télécharger, la marche à suivre pour refaire le test sur ton ordinateur, ce qui nous a fait perdre du temps et ce qu'il manque pour en faire un vrai produit.

On a lancé le test dans la nuit du 2 au 3 novembre 2026, sur un Mac. Outil : Codex 0.163 en ligne de commande, modèle gpt-6-astra, effort high, sur un abonnement ChatGPT Plus, sans clé d'API. En entrée : le reel d'une appli de course de filatov.design, 14 518 307 lectures, et sa légende.

Les phrases de l'agent viennent de ses journaux. Les commandes et les prix ont été relus sur les pages officielles le 3 novembre 2026. Ce qu'on n'a pas pu vérifier est écrit, et la liste complète est à la fin.

À savoir avant de lire

• Le site est en ligne : foulee-carnet-20261103-t1.vercel.app (https://foulee-carnet-20261103-t1.vercel.app). Il s'appelle Foulée, c'est l'agent qui a choisi le nom.

• 38 min 49 de travail de l'agent pour le site, la vidéo et le pack. De bout en bout, 2 h 16, parce que notre limite ChatGPT a sauté pendant la vidéo (partie 6).

• C'est une maquette qui marche. Pas de serveur, les données restent dans ton navigateur, le coach suit des règles écrites d'avance, aucune connexion Garmin. L'appli du reel d'origine utilise de vraies données Garmin et une IA, et son créateur annonce deux heures avec Cursor.

• Il a fallu de la technique. Notre prompt donne une commande Vercel et le nom de HyperFrames, et le terrain était prêt : Node, ffmpeg, Python, un bac à sable réglé, le compte Vercel déjà connecté.

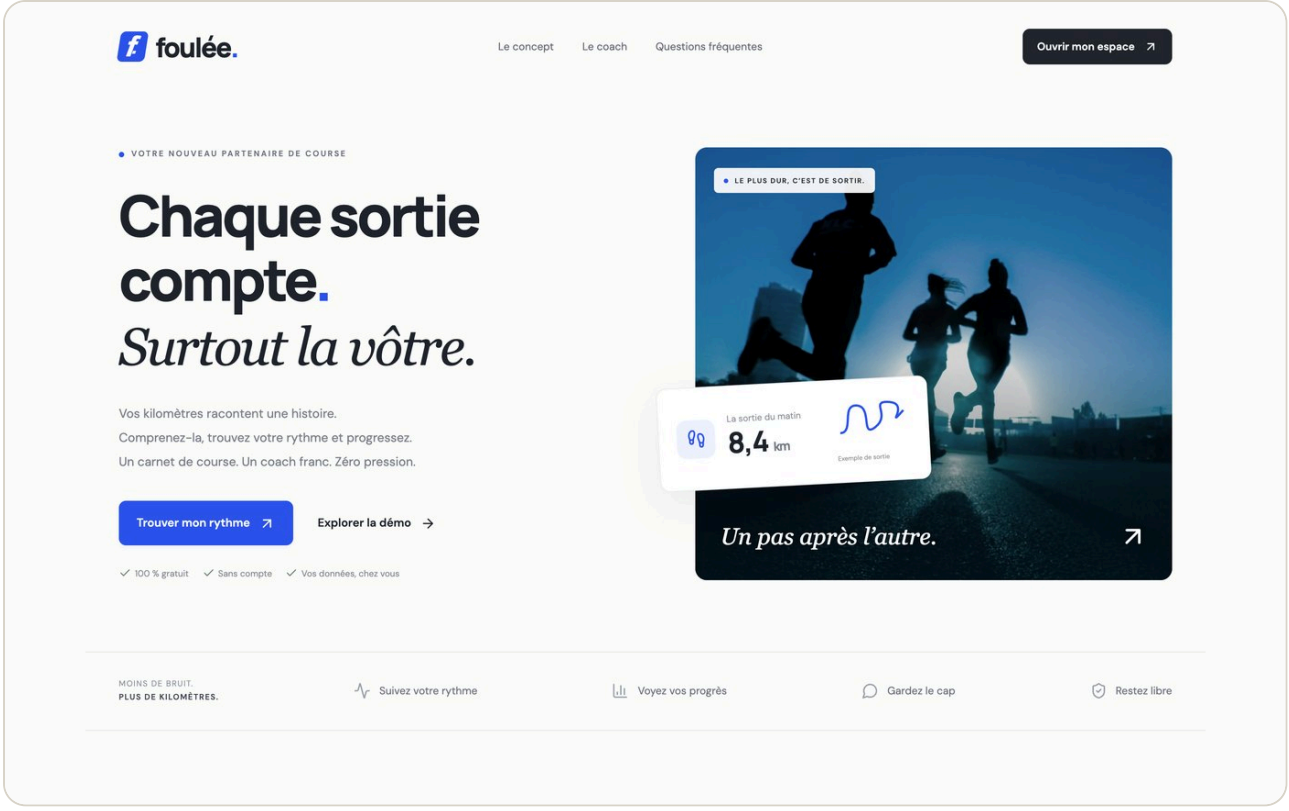
• Ce test a pris un quart du quota hebdomadaire d'un abonnement Plus, et il ne tient pas dans un seul créneau de cinq heures (partie 6).

• Garde l'idée d'une vidéo, jamais la marque, les textes ou les visuels de son auteur (partie 8).

---

1. Le résultat

Ouvre le site (https://foulee-carnet-20261103-t1.vercel.app). Le bouton « Explorer la démo » montre un carnet déjà rempli, « Ouvrir mon espace » part d'un carnet vide.



La page d'accueil de Foulée, écrite par l'agent. Capture du 3 novembre 2026, en 1440 × 900.

Tout ce qu'il a sorti, à récupérer :

• La vidéo de lancement (https://eigenflow-leads.vercel.app/foulee-film-lancement.mp4) : MP4 vertical de 32 s, 4,7 Mo.

• Le pack Instagram (https://eigenflow-leads.vercel.app/foulee-pack-instagram.zip) : ZIP de 2,6 Mo, avec les six visuels, les légendes et les sources modifiables.

• Le code du site (https://eigenflow-leads.vercel.app/foulee-code-source.zip) : ZIP sans nos fichiers de compte ni le reel d'origine.

Le chrono

Les heures viennent de notre fichier chrono.txt, écrit par le script de lancement.

• Prompt 1, le site : 18 min 56 (1 h 25 à 1 h 43).

• Prompt 2, la vidéo, coupé par la limite d'usage : 5 min 34 (1 h 46 à 1 h 51).

• Arrêt : 1 h 35. La limite se levait à 3 h 10, soit 1 h 19 d'attente imposée. On a relancé à 3 h 27.

• Reprise de la vidéo : 8 min 41 (3 h 27 à 3 h 35).

• Prompt 3, le pack : 5 min 38 (3 h 35 à 3 h 41).

Total de travail : 38 min 49, dont 14 min 15 pour la vidéo en deux fois. Ce total ne compte ni l'arrêt ni notre préparation.

Ce qui marche

On a rejoué le site en ligne le 3 novembre, dans Chrome piloté par script, en partant d'un navigateur vide. Aucune erreur dans la console. À 390 pixels de large, rien ne déborde.

• Créer un espace, ajouter une sortie (8,4 km en 45 min, objectif à 42 %), recharger la page : elle est toujours là.

• Modifier et supprimer une sortie, chercher, filtrer par effort, exporter la sélection en CSV.

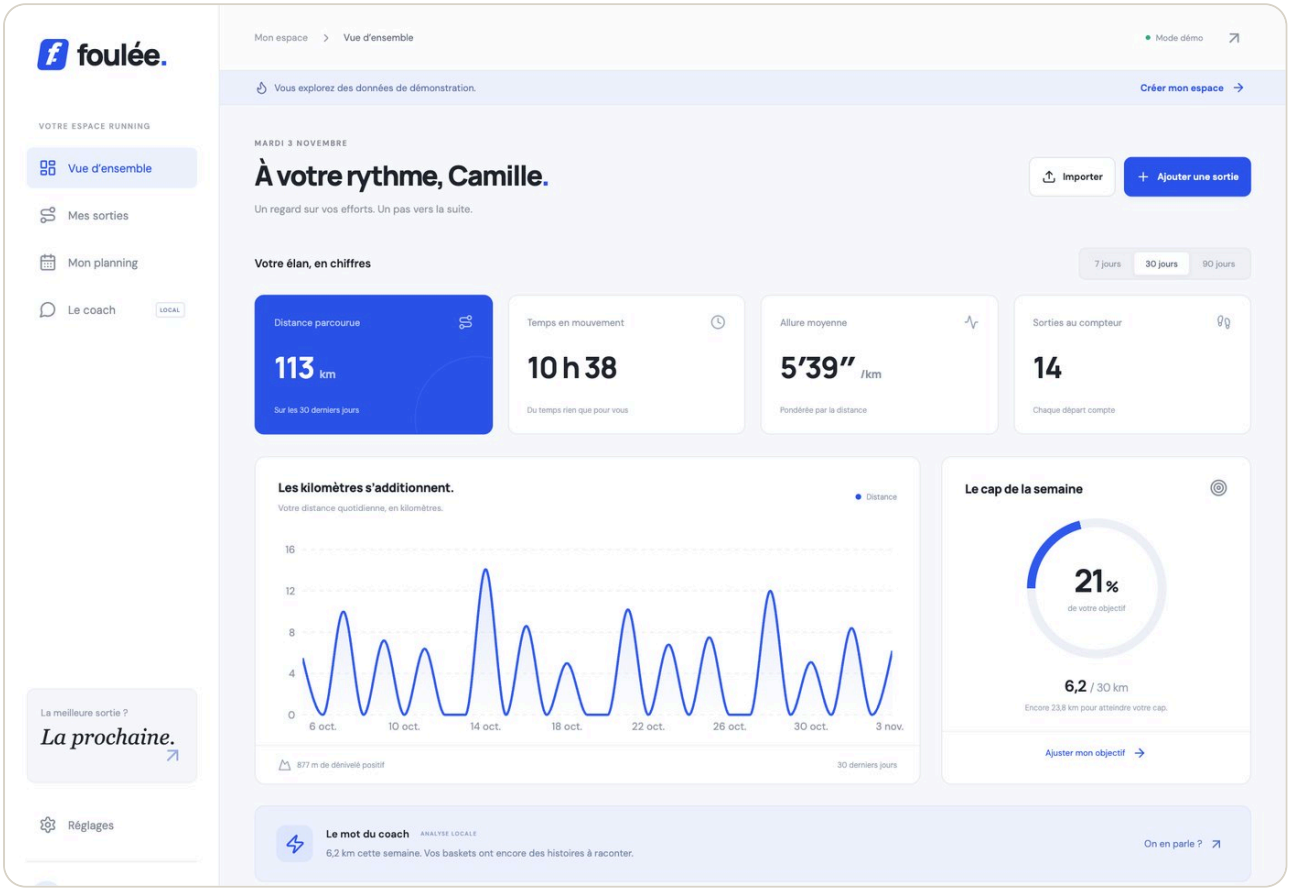
• Importer un CSV (2 sorties) et un GPX (1 sortie, 2,5 km calculés depuis les coordonnées). Le même CSV importé une seconde fois donne « 0 nouvelle sortie · 2 doublon(s) ».

• Les vues 7, 30 et 90 jours : 113 km sur 30 jours et 201,8 km sur 90 jours en démo.

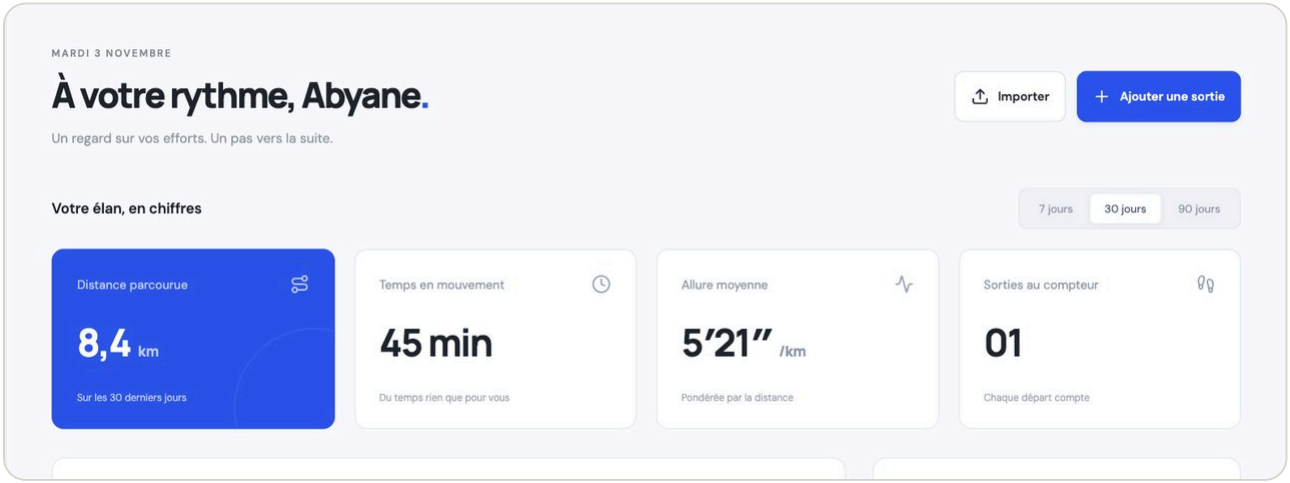
• Le planning : créer, cocher, modifier et supprimer une séance.

• La sauvegarde JSON : on l'exporte, on efface tout, on la restaure, le carnet revient à l'identique.

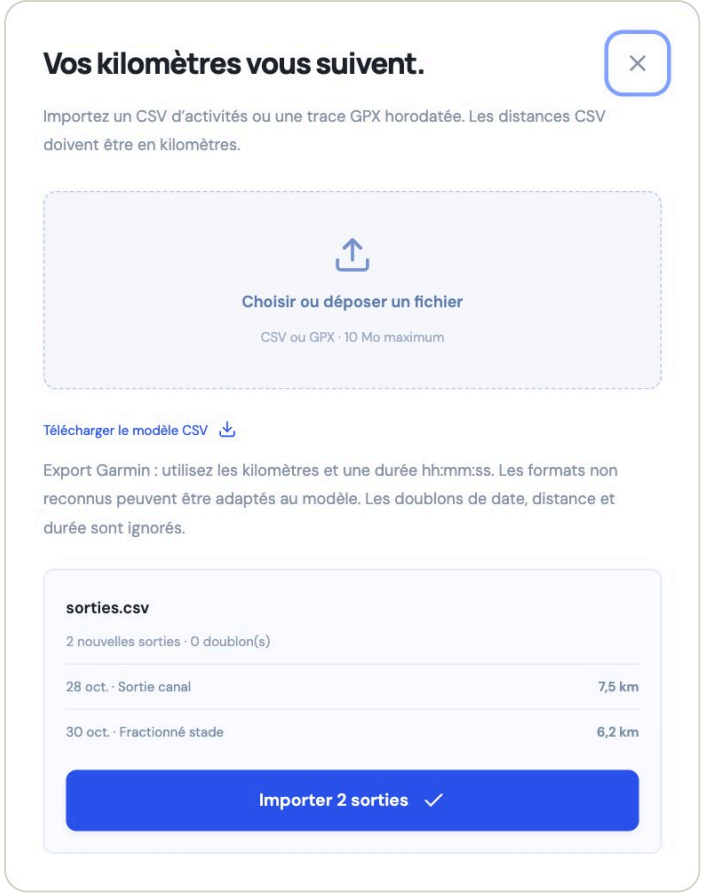
• Les boutons du coach, dans ses deux tons.



Le tableau de bord en mode démo. Le prénom et les chiffres sont des données d'exemple, annoncées comme telles dans le bandeau du haut.



Notre sortie de test après rechargement de la page : 8,4 km, 45 min, 5'21" au kilomètre.



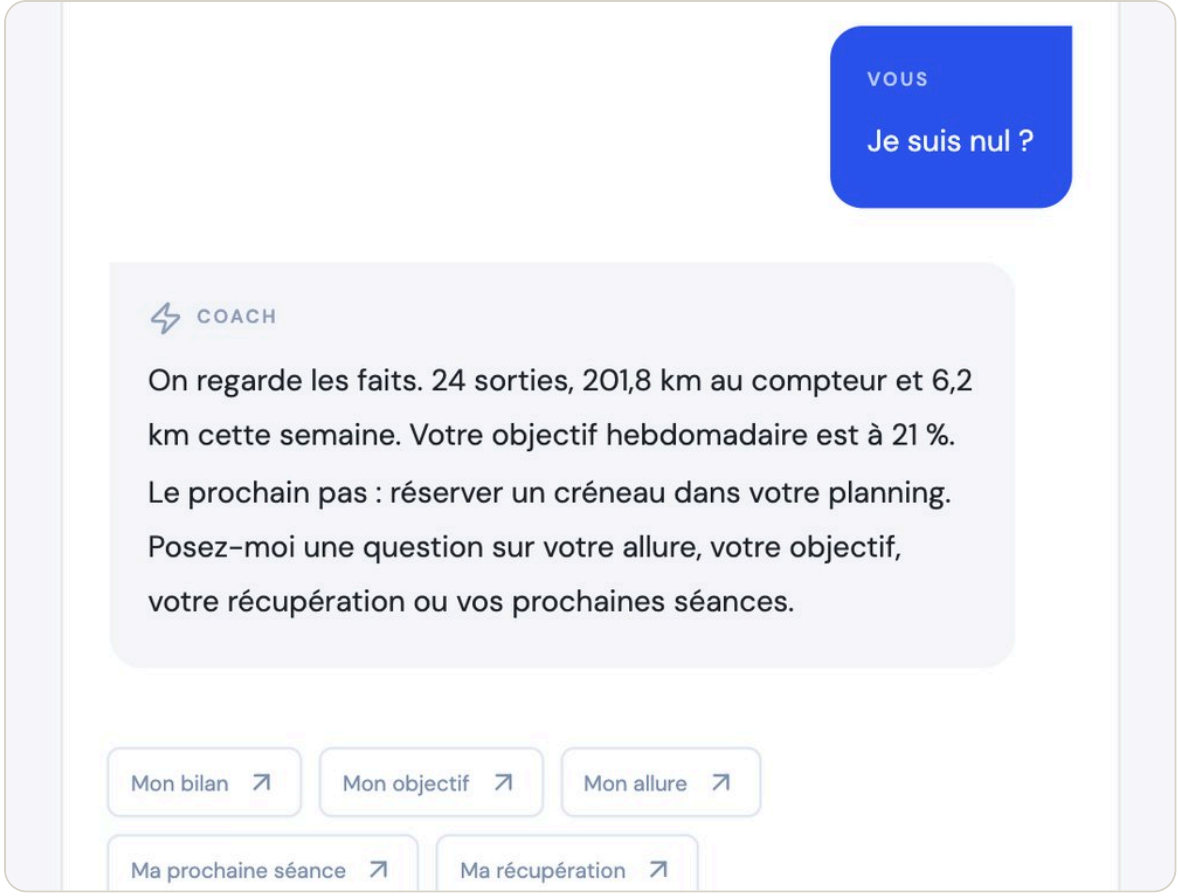
L'import d'un CSV avec son aperçu, avant validation. Le fichier est fabriqué par nous, avec 2 sorties.

On n'a pas essayé : un vrai téléphone, Safari, Firefox, ni un vrai export Garmin. Nos fichiers CSV et GPX sont fabriqués.

Ce qui ne marche pas

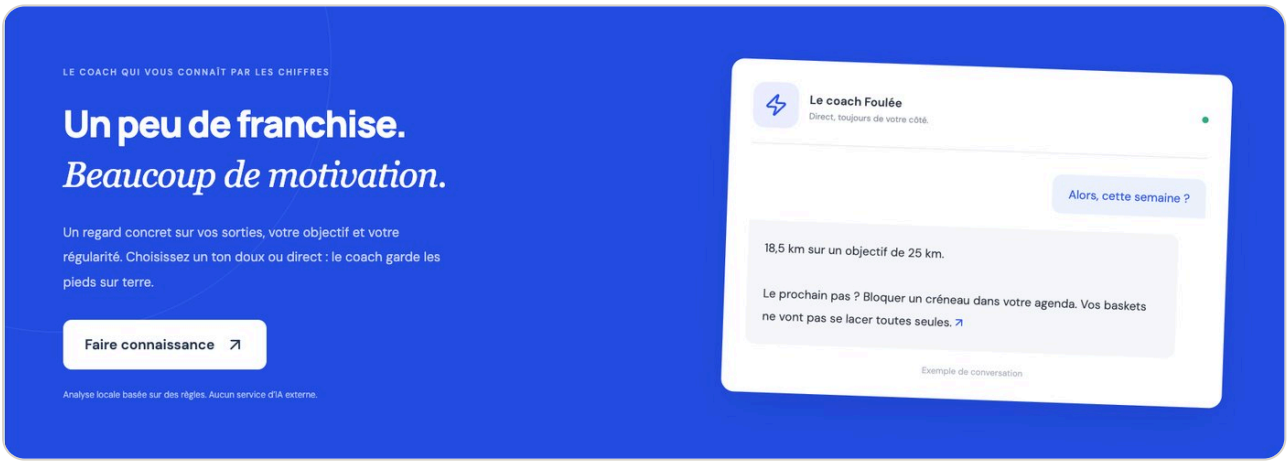
Rien n'est synchronisé. Tout est dans le stockage local du navigateur : ouvert dans un autre navigateur, le carnet est vide. Le site le dit lui-même : « La navigation privée et le nettoyage du navigateur peuvent effacer le carnet. »

Le coach applique des règles écrites d'avance. Il répond sur cinq sujets, et une question libre reçoit le bilan générique.



On lui demande « Je suis nul ? ». Il répond par le bilan, puis liste ce qu'il sait traiter. Capture en mode démo.

L'agent l'écrit noir sur blanc, en petit sous le bouton de la page d'accueil, et plus longuement sur la page de confidentialité : « Ce n'est pas une IA générative ni un service médical. »



La section du coach sur la page d'accueil. La dernière ligne du bloc bleu : « Analyse locale basée sur des règles. Aucun service d'IA externe. »

Foulée n'a aucune connexion Garmin : tu importes un fichier. L'appli d'origine va plus loin. D'après sa vidéo, filatov.design a téléchargé ses vraies données depuis sa montre et les a déposées dans l'appli, et son coach est une IA branchée par API. Sa légende annonce deux heures avec Cursor. Sa vidéo finit sur ces mots à l'écran : « This video is sponsored by Cursor. » Ses deux heures restent donc sa parole, dans une vidéo sponsorisée par l'outil qu'elle montre.

Une partie de ces manques vient de notre prompt. On lui a écrit qu'il n'avait aucune clé et ne créait aucun compte. Il n'avait donc aucun moyen d'ajouter un serveur ou une vraie IA, et il a préféré le dire plutôt que de faire semblant.

2. Les prompts exacts

Copiés de nos fichiers. Le prompt 1 a des sauts de ligne entre ses paragraphes dans l'original : on les a retirés pour qu'il se copie d'un bloc.

Prompt 1, le site

> **À COLLER**
>
> ```
> Mon ami m'a envoyé ce reel : le fichier vidéo est dans entree/reel.mp4 et sa légende dans entree/legende.txt. Il m'a dit : ça pourrait être notre prochain projet. Je veux que tu mettes ce projet en ligne pour moi, et que tu ajoutes d'autres fonctionnalités. Ça doit être un produit complet de bout en bout, avec une bonne page d'accueil et tout ce qui fonctionne. Le site doit être en français. Mets-le en ligne sur Vercel, le CLI est déjà connecté (vercel deploy --prod --yes) : crée un nouveau projet, ne touche à aucun autre. Tu n'as aucune clé d'API, tu ne paies rien et tu ne crées aucun compte. À la fin, donne-moi le lien direct.
> ```

Ce qu'il faut regarder dans ce texte :

• « Mon ami m'a envoyé ce reel » est une phrase de test. Personne ne nous a envoyé ce reel. La tournure vient de la vidéo de manthanjethwani que notre reel traduit, et on l'a gardée pour refaire son essai dans les mêmes mots. C'est nous qui avons choisi le reel de filatov.design, pour ses lectures.

• Le prompt ne dit rien de ce que fait l'appli, mais il pointe vers la légende, et la légende dit presque tout : appli d'entraînement, assistant IA, graphiques, données Garmin. Sans elle, on ne sait pas ce qu'il aurait compris.

• La commande vercel deploy --prod --yes est de la technique, et on l'écrit parce qu'elle compte. Sans cette ligne, il aurait fallu qu'il trouve seul comment publier. L'option --prod vise le domaine de production, --yes saute les questions de Vercel.

• La dernière phrase avant le lien lui interdit clés, paiements et comptes. C'est un garde-fou pour notre compte, et ça explique la partie 1.

Prompt 2, la vidéo

> **À COLLER**
>
> ```
> Maintenant, dans le même thème, fais-moi une vidéo de lancement du produit, verticale, en français. D'abord la vidéo de lancement, avec les outils que tu as : HyperFrames (npx hyperframes) si tu peux. Va jusqu'au bout sans t'arrêter et donne-moi le MP4.
> ```

On lui donne le nom de l'outil et sa commande : HyperFrames, c'est nous qui l'avons choisi, et le reel le dit. La fin, « sans t'arrêter », vient d'un test du 27 octobre avec Claude Code : sans cette phrase, la skill de HyperFrames propose trois histoires et attend qu'on choisisse.

Le message de reprise

Envoyé après la limite d'usage, dans la même session.

> **À COLLER**
>
> ```
> Tu t'es arrêté en plein travail à cause de la limite d'usage. Reprends où tu en étais et termine la vidéo de lancement verticale, en français. Va jusqu'au bout sans t'arrêter et donne-moi le MP4.
> ```

Prompt 3, le pack

> **À COLLER**
>
> ```
> Maintenant, fais-moi tout le pack marketing Instagram, en français : une affiche de lancement, quatre slides de carrousel numérotées, une story verticale et les légendes prêtes à publier. Vérifie les images toi-même et donne-moi le tout dans un seul dossier.
> ```

Ses réponses, mot pour mot

On a remis les paragraphes en liste et retiré les chemins vers notre disque dans ses liens. Le temps affiché est celui du prompt.

Réponse de GPT-6 Astra, notre test, 18 min 56

• Foulée est en ligne : [ouvrir le site](https://foulee-carnet-20261103-t1.vercel.app).

• Le produit est en français : page d'accueil, démo, statistiques interactives, gestion des sorties, imports CSV/GPX, planning, objectifs et sauvegardes.

• Le coach fonctionne avec des règles locales, sans API. Les données restent dans votre navigateur et sont exportables.

• Nouveau projet Vercel uniquement, aucun compte créé ni service payant ajouté. **11 tests passent**, publication vérifiée ; le contrôle visuel automatisé était bloqué par l'environnement.

Réponse de GPT-6 Astra, notre test, 8 min 41

• Télécharger la vidéo de lancement (MP4).

• 32 secondes · format vertical 1080 × 1920 · voix française et musique originale. Réalisée avec HyperFrames, images et fichier final vérifiés.

Réponse de GPT-6 Astra, notre test, 5 min 38

• Ouvrir le dossier complet · Télécharger le ZIP.

• Le pack contient :

• Une affiche de lancement.

• Quatre slides numérotées, en 1080 × 1350.

• Une story verticale, en 1080 × 1920.

• Les légendes françaises, hashtags et textes alternatifs.

• Les sources modifiables et un aperçu d'ensemble.

• J'ai inspecté les six images et corrigé les défauts de cadrage et d'espacement.

---

3. Ce que l'agent a fait, étape par étape

Il regarde le reel

Il liste le dossier, lit une de nos skills (partie 6), lit la légende, puis fabrique une planche : une image toutes les 6 secondes, 270 pixels de large, onze images sur une seule feuille. Voici sa commande :

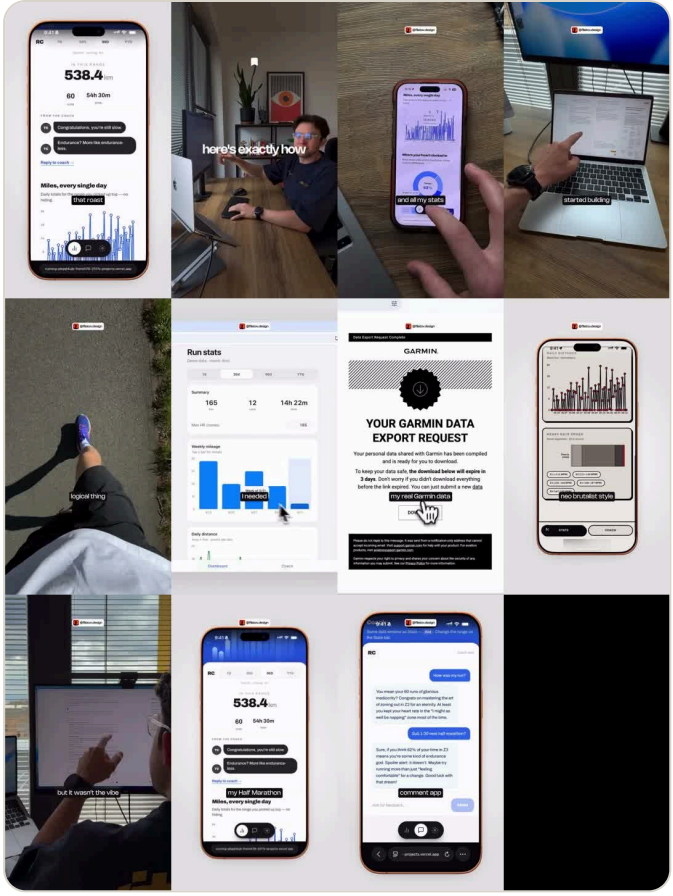
À COLLER

```
ffmpeg -hide_banner -loglevel error -i entree/reel.mp4 -vf
"fps=1/6,scale=270:-1,tile=4x3" -frames:v 1 .work/reel-contact.jpg
```

Le fichier de session montre qu'il ouvre cette image à 1 h 25 min 21 s. Il ne transcrit pas le son. À 1 h 25 min 34 s, soit 32 secondes après le lancement, il écrit :

« Le reel présente un carnet de course avec statistiques, graphiques et un coach au ton direct. Je pars sur « Foulée » : une page d'accueil, un tableau de bord, l'import de sorties, des objectifs et un planning. Sans clé d'API, le coach analysera réellement les données avec des règles locales, clairement indiquées. »

Il a donc tranché en lisant la légende et onze petites images.



Sa planche, telle qu'il l'a vue. Images du reel de filatov.design, reproduites ici pour commenter le test.

Il construit le site

Il télécharge une photo pour la page d'accueil, sans qu'on lui demande rien sur la licence (partie 8) :

```
À COLLER

curl -L --fail 'https://images.unsplash.com/photo-1552674605-db6ffd4facb5?
auto=format&fit=crop&w=1400&q=85' -o public/running.jpg
```

Puis il écrit l'appli en React et TypeScript. Le premier essai de compilation échoue, il manque la feuille de style : il l'écrit et continue. Il écrit 8 tests de calcul et 3 tests d'interface. Quand il installe ses outils de test, npm signale 3 failles, dont 2 critiques, dans le paquet vitest : il le met à jour, et npm ne signale plus rien.

Il crée le projet Vercel et publie à 1 h 43 min 12 s. Il vérifie ensuite chaque adresse par des requêtes HTTP, et il donne le lien. Tout le site tient en 32 commandes.

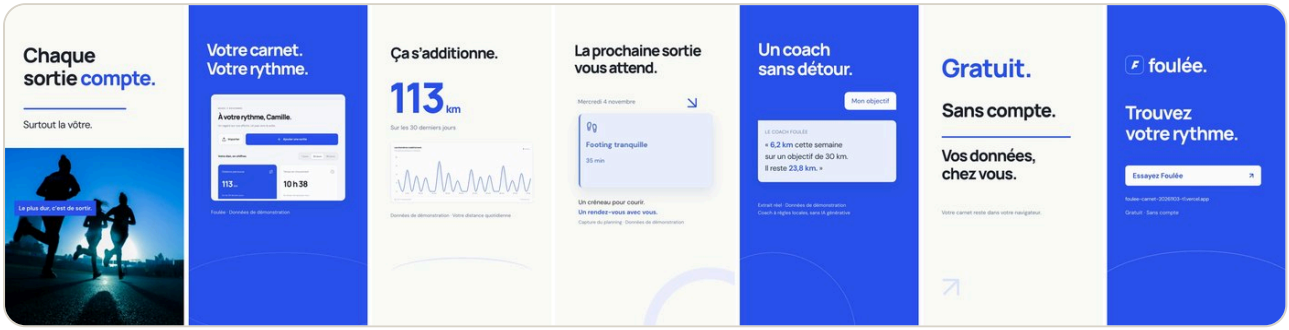
Il n'a jamais vu son site à l'écran à ce moment-là. Les navigateurs refusaient de démarrer (partie 6), et il l'annonce dans sa réponse.

Il monte la vidéo

Il installe HyperFrames 0.9.6, lit ses skills, essaie la voix système du Mac, passe à Kokoro (une voix de synthèse qui tourne sur la machine), capture les écrans du site. C'est là que la limite le coupe. À la reprise, il relit son dossier, écrit sept scènes, corrige les polices qui ne se chargeaient pas, ralentit la voix de 1,12 à 0,95, compose une musique en code et lance le rendu.

Résultat : 32 s, 1080 × 1920, 30 images par seconde. Il annonce -16,54 LUFS, et on mesure -16,4 avec ffmpeg. Il écrit aussi : « aucune écoute humaine effectuée par l'agent ». Rien de surprenant, il n'a pas d'oreilles.

Écoute deux phrases avant de t'en servir. Passées seules dans Whisper, « Le coach fait le point » donne « Le In-Coach fluffait le point » et « Trouvez votre rythme » donne « trouvez votre Henry Smuffle ». Whisper se trompe parfois sur une voix de synthèse. Ça reste un signal à vérifier à l'oreille, et on ne l'a pas corrigé.



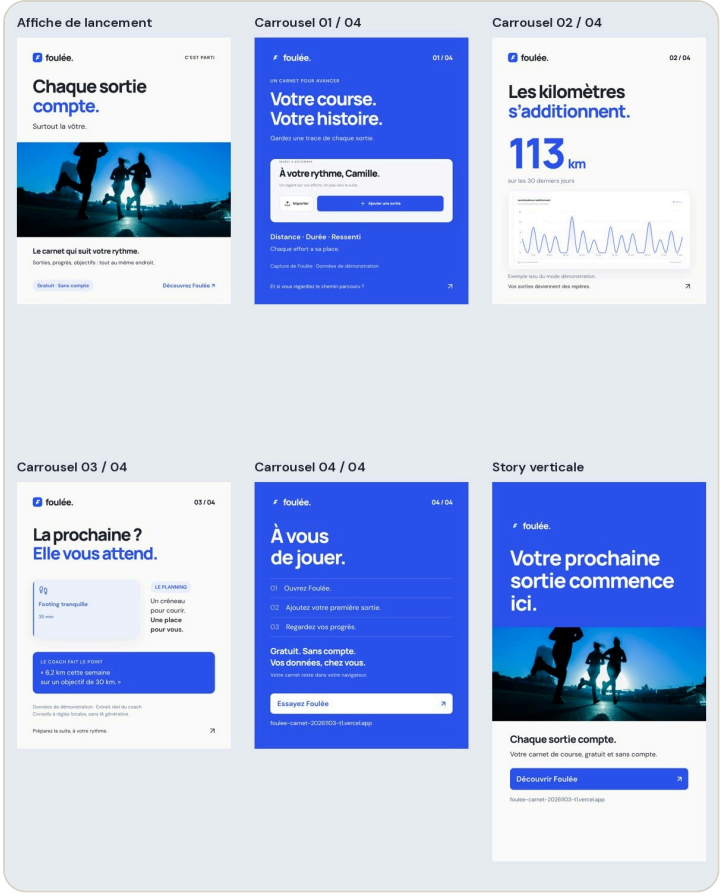
Sept images du MP4 final, de gauche à droite : l'accroche, le carnet, les 113 km, la prochaine séance, le coach, « Gratuit. Sans compte. », la fin. Les écrans montrés sont de vraies captures du site, avec la mention « Données de démonstration ».

Il fait le pack

En 5 min 38 et 14 commandes : une affiche et quatre slides en 1080 × 1350, une story en 1080 × 1920, cinq fichiers de texte (légendes, bio, textes alternatifs) et un mode d'emploi avec l'ordre de publication : affiche et story au lancement, carrousel le lendemain. Il regarde ses exports et corrige lui-même deux slides, le recadrage de la capture du carnet et un espacement sous le graphique.

Détail qui nous a plu : il a gardé la franchise du site jusque dans le marketing. La légende du carrousel écrit « Le coach utilise des règles locales, sans IA générative. Les chiffres présentés dans ce carrousel viennent du mode démonstration. »

Rien n'a été publié sur Instagram, et le mode d'emploi le précise.



L'aperçu d'ensemble qu'il a fabriqué : l'affiche, les quatre slides et la story.

---

4. Le refaire sur ton ordinateur

Tu n'as jamais ouvert un terminal ? C'est écrit pour toi. Chaque bloc se copie, se colle dans le Terminal, et se lance avec la touche Entrée.

On n'a fait tourner tout ça que sur un Mac récent. Notre Codex vient de l'appli ChatGPT pour Mac, et le Chrome de HyperFrames était déjà téléchargé depuis un test précédent : l'installation neuve décrite ici n'a pas été rejouée de bout en bout. Sur Windows, on n'a rien testé.

Ce qu'il te faut

• Un abonnement ChatGPT Plus, à 20 dollars par mois d'après la page des tarifs (https://learn.chatgpt.com/docs/pricing). Free et Go n'ont accès qu'à GPT-6 Luna. Le prix en euros dans ton compte, on ne l'a pas vérifié.

• Un compte Vercel. L'offre Hobby est gratuite (https://vercel.com/pricing) et réservée à un usage personnel non commercial (partie 7).

• De la place : notre dossier de test pèse 796 Mo à la fin, dont 548 Mo pour la voix de synthèse et ses modèles.

• Une vidéo que tu as le droit d'utiliser (étape 6).

Étape 1, ouvrir le Terminal

Sur Mac : Cmd + Espace, tape Terminal, Entrée. Une fenêtre s'ouvre avec une ligne qui attend. C'est là que tu colles.

Étape 2, installer Homebrew, Node, ffmpeg et Python

Homebrew installe les autres outils en une commande. Sa commande officielle (https://brew.sh/) :

À COLLER

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Il te demande le mot de passe de ta session. À la fin, il affiche des lignes à coller pour se rendre disponible. Sa documentation (https://docs.brew.sh/Installation) prévient : sans ça, Homebrew ne fonctionne pas. Colle-les, ferme le Terminal et rouvre-le. Elle annonce aussi macOS 15 ou plus.

À COLLER

```
brew install node ffmpeg python@3.14
```

HyperFrames demande Node.js 22 ou plus et FFmpeg (https://github.com/heygen-com/hyperframes), sa voix locale Python 3.10 ou plus. Homebrew installera Node 26.13.0 et ffmpeg 9.0.2. Notre test tournait avec Node 22.23.4, ffmpeg 8.0.1 et Python 3.14.5 : ces versions plus récentes ne sont pas testées.

Étape 3, installer Codex et te connecter

La documentation d'OpenAI (https://learn.chatgpt.com/docs/codex/cli) donne cette commande pour macOS et Linux :

À COLLER

```
curl -fsSL https://chatgpt.com/codex/install.sh | sh
```

Ferme et rouvre le Terminal, puis lance :

À COLLER

```
codex
```

Choisis Sign in with ChatGPT et connecte-toi dans la page qui s'ouvre avec ton compte Plus. Ferme ensuite cette fenêtre de Terminal et ouvres-en une nouvelle. Pour contrôler :

À COLLER

```
codex login status
```

Réponse attendue : « Logged in using ChatGPT ». Aucune clé d'API n'est nécessaire. Notre commande retire même celle qui traînerait dans l'environnement, pour que rien ne parte sur une facture à l'usage.

Étape 4, installer Vercel et te connecter

Commandes de Vercel (https://vercel.com/docs/cli) :

```
À COLLER
npm i -g vercel
```

```
À COLLER
vercel login
```

Suis ce qu'il affiche. Si tu n'as pas de compte, vercel signup en crée un. Pour contrôler, cette commande doit afficher ton identifiant :

```
À COLLER
vercel whoami
```

Étape 5, préparer HyperFrames

Dans notre test, le Chrome de HyperFrames était déjà là, téléchargé le 27 octobre. Or le bac à sable des étapes suivantes interdit d'écrire dans le dossier où il se range : le téléchargement de la voix a d'ailleurs échoué avec l'erreur « EPERM: operation not permitted », et l'agent a contourné. Prends les devants, avant de lancer l'agent :

```
À COLLER
npx hyperframes browser ensure
```

```
À COLLER
npx hyperframes doctor
```

Les lignes Node.js, FFmpeg, FFprobe et Chrome doivent être cochées. Whisper, Kokoro, MusicGen et Docker peuvent être en croix, ils sont facultatifs.

Étape 6, préparer le dossier et la vidéo

```
À COLLER
mkdir -p ~/Desktop/mon-projet/entree
```

```
À COLLER
cd ~/Desktop/mon-projet
```

```
À COLLER
open entree
```

Le Finder s'ouvre sur le dossier entree. Glisses-y ta vidéo et renomme-la reel.mp4. Copie ensuite le texte qui l'accompagne (légende ou description), puis écris-le dans un fichier :

```
À COLLER
pbpaste > entree/legende.txt
```

Contrôle avec cette commande, qui doit afficher legende.txt et reel.mp4 :

```
À COLLER
ls entree
```

Quelle vidéo prendre ? Par ordre de préférence :

1. Une vidéo à toi. Une minute filmée au téléphone où tu expliques ton idée et tu montres un croquis fera l'affaire. Ton Mac filme aussi l'écran avec Cmd + Maj + 5. On n'a pas testé ce cas, et le texte du fichier legende.txt comptera beaucoup.

2. Ton propre reel. Prends le fichier d'origine sur ton téléphone. Sinon, Instagram propose un export de tes informations depuis l'Espace Comptes, décrit dans son centre d'aide (https://help.instagram.com/181231772500920), avec un délai possible.

3. La vidéo de quelqu'un d'autre, avec son accord écrit. Un message suffit pour commencer.

Évite les sites de téléchargement de reels. Les conditions d'utilisation d'Instagram (https://help.instagram.com/581066165581870/) interdisent la collecte automatisée sans autorisation. Nous-mêmes, pour ce test, on a récupéré le fichier du reel de filatov.design par un service de collecte (Apify). On s'en sert pour analyser et commenter, on ne le redistribue pas, et Foulée n'est pas à vendre. Pour construire un projet à toi, ne fais pas comme nous.

Étape 7, le test à blanc

Ce prompt prend très peu de quota et te dit si tout est branché. C'est celui qu'on a lancé à 1 h 24. Copie-le avec le bouton, puis écris-le dans un fichier :

À COLLER

```
Lance ces commandes et recopie leur sortie : vercel whoami ; node -v ; ffmpeg -version | head -1 ; npm view hyperframes version ; touch ~/Desktop/test-ecriture-interdite.txt. Ne fais rien d'autre.
```

À COLLER

```
pbpaste > prompt0.txt
```

Puis lance Codex avec nos réglages :

À COLLER

```
env -u OPENAI_API_KEY codex exec --skip-git-repo-check --sandbox workspace-write -m gpt-6-astra -c 'approval_policy="never"' -c 'model_reasoning_effort="high"' -c 'sandbox_workspace_write.network_access=true' -c "sandbox_workspace_write.writable_roots=[\"$HOME/.npm\",\"$HOME/Library/Caches\",\"$HOME/Library/Application Support/com.vercel.cli\",\"$HOME/.vercel\"]" "$(cat prompt0.txt)" < /dev/null
```

Voici ce qu'il a recopié chez nous. Tu auras ton identifiant Vercel et tes versions :

• Vercel CLI 50.1.3

• ton identifiant Vercel

• v22.23.4

• ffmpeg version 8.0.1

• 0.9.6

• touch: ~/Desktop/test-ecriture-interdite.txt: Operation not permitted

La dernière ligne est la bonne nouvelle : il n'a pas pu écrire sur ton bureau, le bac à sable tient. En haut de l'affichage, une ligne commence par sandbox, liste les dossiers autorisés et finit par (network access enabled). Ce que font les options :

• env -u OPENAI_API_KEY retire une éventuelle clé d'API de l'environnement.

• --sandbox workspace-write limite l'écriture des commandes au dossier du projet, au dossier temporaire et aux dossiers listés dans writable_roots (les caches de npm et de Vercel).

• network_access=true ouvre le réseau, nécessaire pour npm et Vercel. D'après OpenAI (https://learn.chatgpt.com/docs/agent-approvals-security), il est coupé par défaut.

• approval_policy="never" supprime toute question pendant le travail.

• model_reasoning_effort="high" fixe l'effort. OpenAI conseille de commencer en Light avec Astra (https://learn.chatgpt.com/docs/models), on n'a testé que high.

• < /dev/null ferme l'entrée standard : sans ça, Codex écrit « Reading additional input from stdin... » et attend.

Par rapport à notre script, on a seulement retiré le dossier de configuration vide et les fichiers de journal.

Avant de continuer, sache ce que ça implique. Ce réglage bride l'écriture et laisse la lecture large : il a lu des fichiers de notre dossier personnel (partie 6). Avec Internet ouvert et aucune question posée, une consigne glissée dans une légende ou une vidéo d'inconnu pourrait lui être adressée. On n'a pas testé ce scénario. Si ça t'inquiète, fais tourner le test dans une session macOS à part, sans tes documents. La version plus prudente, c'est de lancer codex sans option et d'accepter chaque demande une à une. On ne l'a pas testée non plus.

Étape 8, lancer le prompt 1

Copie le prompt 1 de la partie 2 avec le bouton, puis :

À COLLER

```
pbpaste > prompt1.txt
```

À COLLER

```
env -u OPENAI_API_KEY codex exec --skip-git-repo-check --sandbox workspace-write
-m gpt-6-astra -c 'approval_policy="never"' -c 'model_reasoning_effort="high"' -
c 'sandbox_workspace_write.network_access=true' -c
"sandbox_workspace_write.writable_roots=
[\"$HOME/.npm\",\"$HOME/Library/Caches\",\"$HOME/Library/Application
Support/com.vercel.cli\",\"$HOME/.vercel\"]" "$(cat prompt1.txt)" < /dev/null
```

Le texte défile, c'est normal. Chez nous, ça a pris 18 min 56. Il finit par donner un lien. S'il s'arrête sur une erreur, copie le message, écris-le dans un fichier avec pbpaste, et relance sur ce fichier avec la commande de l'étape 10 : il reprend sa session.

Si tu pars de ta propre vidéo, voici la version qu'on utiliserait. On ne l'a pas lancée. Elle retire l'ami de fiction, demande de créer plutôt que de reproduire, et réclame la source des images :

À COLLER

```
Voici une vidéo qui présente une idée de produit : le fichier est dans
entree/reel.mp4 et son texte dans entree/legende.txt. Regarde-la en images avec
ffmpeg et lis le texte. Construis un produit à toi à partir de cette idée :
choisis ton propre nom, tes propres visuels et tes propres textes, ne reprends
ni la marque, ni les écrans, ni les phrases de la vidéo. N'utilise que des
images dont la licence permet un usage commercial et note la source de chacune
dans le README. Ça doit être un produit complet de bout en bout, avec une bonne
page d'accueil et tout ce qui fonctionne. Le site doit être en français. Mets-le
en ligne sur Vercel, le CLI est déjà connecté (vercel deploy --prod --yes) :
crée un nouveau projet, ne touche à aucun autre. Tu n'as aucune clé d'API, tu ne
paies rien et tu ne crées aucun compte. À la fin, dis-moi ce que tu n'as pas pu
vérifier et donne-moi le lien direct.
```

Étape 9, vérifier le site toi-même

L'agent n'a peut-être rien vu de ce qu'il a publié (partie 6). Cinq minutes suffisent, sur ton téléphone et ton ordinateur :

• Ajoute une sortie, recharge la page, regarde si elle est restée.

• Pose au coach une question qu'il n'attend pas.

• Importe un vrai fichier de ta montre si tu en as un. Nous, on ne l'a pas fait.

• Lis la page de confidentialité, et vérifie qu'elle décrit ce que le site fait vraiment.

Étape 10, la vidéo et le pack

Copie le prompt 2, puis, depuis le même dossier :

À COLLER

```
pbpaste > prompt2.txt
```

À COLLER

```
env -u OPENAI_API_KEY codex exec resume --last --skip-git-repo-check -m gpt-6-
astra -c 'approval_policy="never"' -c 'sandbox_mode="workspace-write"' -c
'model_reasoning_effort="high"' -c 'sandbox_workspace_write.network_access=true'
-c "sandbox_workspace_write.writable_roots=
[\"$HOME/.npm\",\"$HOME/Library/Caches\",\"$HOME/Library/Application
Support/com.vercel.cli\",\"$HOME/.vercel\"]" "$(cat prompt2.txt)" < /dev/null
```

Pour le pack, même commande avec prompt3.txt. Si la limite saute, écris le message de reprise dans un fichier de la même façon et relance à l'heure qu'il t'indique.

À la fin, ne partage pas ce dossier tel quel. Chez nous, l'agent a lancé vercel link, qui y a créé un fichier .env.local et un dossier .vercel, propres à notre compte.

---

5. La même chose avec Claude Code

On n'a pas testé ce cas avec Claude Code. Tout ce qui suit est la transposition de notre marche à suivre, à partir de sa documentation (https://code.claude.com/docs/en/overview).

Les étapes 1, 2, 4, 5 et 6 ne changent pas. À la place de Codex, il te faut un abonnement Claude : 17 dollars par mois à l'année ou 20 au mois en Pro, 100 en Max, et Claude Code n'est pas inclus dans l'offre gratuite (https://claude.com/pricing). Ensuite :

À COLLER

```
curl -fsSL https://claude.ai/install.sh | bash
```

Ouvre un nouveau Terminal, puis lance :

```
À COLLER
cd ~/Desktop/mon-projet
```

```
À COLLER
claude
```

Il te demande de te connecter à la première utilisation. Colle ensuite les prompts un par un dans la même conversation. On ajouterait deux choses, sans les avoir testées :

• Au prompt 1 : « Regarde la vidéo en images avec ffmpeg. »

• Avant le prompt 2, la skill de HyperFrames, avec la commande de sa page (https://hyperframes.heygen.com) :

```
À COLLER
npx skills add heygen-com/hyperframes
```

Claude Code demande ta permission avant les actions sensibles. Sa documentation (https://code.claude.com/docs/en/permission-modes) indique que Maj + Tab fait défiler les modes, que le bac à sable des commandes s'active avec /sandbox, et qu'il faut réserver le mode qui saute toutes les permissions à un environnement isolé. Pour un premier essai, reste sur les demandes et lis ce que tu acceptes.

Seule mesure qu'on peut te donner : le 27 octobre, sur un autre sujet, Claude Code avec HyperFrames a monté deux films en 14 min 44 et 14 min 31 (Opus 5.5, abonnement). On n'a mesuré ni la chaîne complète, ni sa consommation de quota sur ce test.

---

6. Ce qui nous a coûté du temps

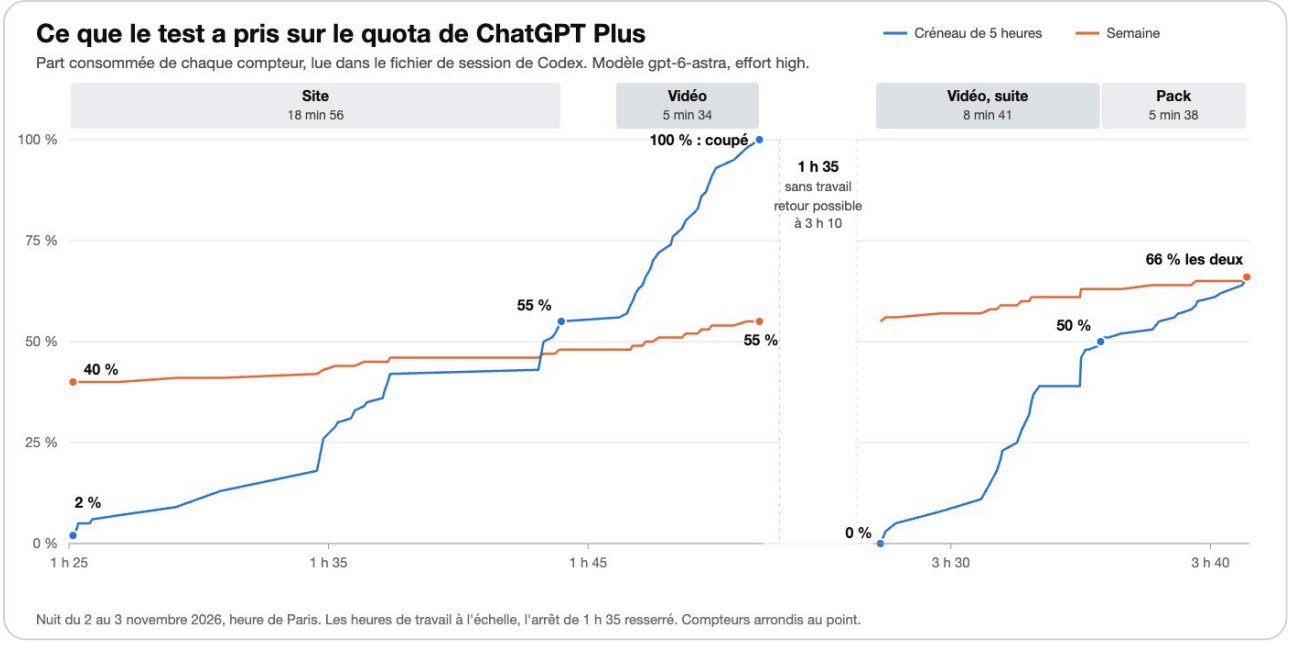
La limite d'usage de ChatGPT Plus

Au bout de 5 min 34 sur la vidéo, Codex s'arrête avec ce message :

Réponse de GPT-6 Astra, notre test, 5 min 34

• You've hit your usage limit. Upgrade to Pro (https://chatgpt.com/explore/pro), visit https://chatgpt.com/settings/usage to purchase more credits or try again at 3:10 AM.

Le fichier de session de Codex garde les deux compteurs d'usage à chaque appel au modèle. Voici ce qu'ils disent, et ça nous a surpris.



Les deux compteurs pendant le test, dessinés par nous à partir du fichier de session.

• Le site : le créneau de cinq heures passe de 2 à 55 %, la semaine de 40 à 48 %.

• La vidéo : 45 points avant la coupure, 50 après, soit 95 points de créneau et 15 de semaine.

• Le pack : 16 points de créneau et 3 de semaine.

• Au total, 164 points de créneau et 26 points de semaine.

D'autres essais avaient déjà entamé notre quota. Le compteur de la semaine était à 40 % au lancement. Sur le créneau de cinq heures, leur part est bien plus faible qu'on ne le pensait : 2 % à la première lecture, premier appel du test compris. Le test de notre reel précédent a fini à 22 h 00, et ce créneau s'est ouvert à 22 h 10, cinq heures avant sa fin à 3 h 10.

Le reste, 98 points en 24 min 30 de travail, vient de ce test, en effort high, sur un compte Plus. Un seul passage ne fait pas une règle. OpenAI estime (https://learn.chatgpt.com/docs/pricing) entre 5 et 45 messages locaux par créneau avec Astra en Plus, précise que ce ne sont pas des limites fixes, et que le contexte, le raisonnement et les outils pèsent sur l'usage. Ici, deux messages de notre part ont suffi à remplir un créneau : l'agent enchaîne les commandes (32 pour le site, 38 avant la coupure) et renvoie son contexte à chacune. C'est notre lecture des jetons, ci-dessous.

Autre écart : cette même page (https://learn.chatgpt.com/docs/pricing) dit que l'agent peut continuer un tour commencé quand la limite arrive (« the agent will be able to continue working on that turn »). Chez nous, le tour s'est terminé en échec, à 100 %. Le travail déjà écrit sur le disque était intact, et il en est reparti : « Je reprends avec les captures du produit et la voix française déjà générées. »

Côté jetons, le fichier de session donne 9 482 348 jetons en entrée, dont 9 140 608 en cache, et 75 558 en sortie. Au tarif de l'API relevé le 3 novembre (https://developers.openai.com/api/docs/pricing) (10 dollars le million en entrée, 1 en cache, 50 en sortie), ça ferait environ 16 dollars : 5,4 pour le site, 9,3 pour la vidéo, 1,7 pour le pack. Sur l'abonnement, on n'a rien payé de plus. Ça se paie en quota et en attente.

Ce qu'on essaierait la prochaine fois. Rien de tout ça n'est testé :

• Regarder le compteur avant de lancer, sur le tableau de bord (https://chatgpt.com/codex/settings/usage) ou avec /status dans Codex.

• Découper : le site sur un créneau, la vidéo seule sur le suivant (elle a pris 95 points), le pack ensuite.

• Baisser l'effort, comme OpenAI le conseille pour Astra, ou changer de modèle avec -m gpt-6.1-sol. La page des tarifs lui donne 15 à 160 messages par créneau en Plus, et la page des modèles (https://learn.chatgpt.com/docs/models) le présente comme proche d'Astra pour moins cher.

• Passer à Pro, qui n'a pas de limite de cinq heures pour l'instant d'après OpenAI, à partir de 100 dollars par mois. Ou acheter des crédits sur Plus.

Les navigateurs bloqués par le bac à sable

Pendant le prompt 1, l'agent essaie de tester son site dans un vrai navigateur, avec Playwright. Chromium refuse de démarrer, WebKit aussi. Deux lignes du journal :

```
À COLLER

browserType.launch: Target page, context or browser has been closed
```

```
À COLLER

[pid=89141][err] ~/Library/Caches/ms-playwright/webkit-2370/pw_run.sh: line 7:
89147 Abort trap: 6
```

Il écrit alors : « L'environnement bloque le lancement des navigateurs automatisés ; je poursuis avec des tests d'interface en environnement simulé. Je pourrai vérifier les interactions, mais pas confirmer visuellement le rendu mobile ici. »

Ses 3 tests d'interface tournent donc dans JSDOM, un faux navigateur sans affichage, et il publie sans avoir vu une seule page. Il l'attribue au bac à sable. On n'a pas refait l'essai sans bac à sable, donc on ne l'a pas prouvé. Fait étrange : pendant la vidéo, HyperFrames arrive à lancer son propre Chrome dans le même bac à sable, et c'est comme ça qu'il voit enfin des captures de son site, après l'avoir publié. On ne sait pas expliquer cet écart.

Pour toi, ça veut dire que le contrôle visuel du site est à ta charge (étape 9). Pour la vidéo et le pack, il a bien regardé ses images, avec les captures de HyperFrames.

Les skills lues malgré notre dossier vierge

On voulait un test sans nos instructions. On a donc donné à Codex un dossier de configuration vide, avec seulement la connexion au compte. Dans le journal, on trouve pourtant ces commandes :

```
À COLLER

cat ~/.agents/skills/reel-autopsy/SKILL.md
```

À COLLER

```
cat ~/.agents/skills/use-railway/SKILL.md
```

À COLLER

```
cat ~/.agents/skills/react-patterns/SKILL.md
```

La documentation d'OpenAI (https://learn.chatgpt.com/docs/build-skills) l'explique : Codex lit aussi les skills de $HOME/.agents/skills, un dossier à part de sa configuration. Sa liste de départ en comptait une trentaine, et il en a ouvert trois.

Reel-autopsy est notre skill pour disséquer une vidéo, et il l'a lue juste avant de découper le reel en images. On ne peut pas dire s'il aurait fait pareil sans elle. Sa commande de planche n'en vient pas, mais l'idée a pu l'aider. Il écrit aussi qu'il utilise react-patterns pour structurer l'appli. Use-railway n'a rien changé : le prompt demandait Vercel.

Sur un ordinateur sans skills, tu n'auras pas cet effet, et il aura peut-être un peu moins de méthode. Pour un test propre, OpenAI documente comment en désactiver une, avec une entrée `skills.config` dans son fichier de configuration. On ne l'a pas fait.

---

7. De cette maquette à un vrai produit

Voilà ce qu'il manque, de la première chose à faire à la dernière.

Rendre le code relisible

Le fichier principal, src/main.tsx, fait 47 375 caractères sur 26 lignes, et sa plus longue ligne en compte 8 034. Ça marche, et aucun humain ne peut le relire comme ça. Sa bibliothèque de graphiques, Recharts 2, est signalée comme plus maintenue par npm à l'installation. Aucun des 11 tests ne tourne dans un vrai navigateur. Commence par ce prompt, qu'on n'a pas lancé :

À COLLER

```
Avant toute nouvelle fonction, reformate src/main.tsx et découpe-le en un
fichier par écran. Ne change aucun comportement. Lance les tests avant et après,
et dis-moi ce qui a changé.
```

Le serveur, les comptes, la synchro

Aujourd'hui, tout tient dans un seul document JSON du navigateur, sous la clé foulee-personal-v1. Il faudra des comptes et une base de données. On partirait sur Supabase (https://supabase.com/pricing), qu'on utilise déjà : l'offre gratuite donne 2 projets actifs et 500 Mo, et elle se met en pause après une semaine d'inactivité. La suivante démarre à 25 dollars par mois.

Les pièges qu'on connaît :

• Une table par type de donnée, avec une date de modification sur chaque ligne. Si tu synchronises le gros document tel quel, le téléphone écrasera ce que l'ordinateur vient d'ajouter.

• Les droits d'accès. Supabase le met en tête de sa page (https://supabase.com/docs/guides/database/postgres/row-level-security) : « Enable RLS on every table in an exposed schema ». Sans ça, une table exposée peut être lue et modifiée par n'importe qui ayant un droit dessus. Demande à l'agent de le prouver avec deux comptes de test, et relis.

• La reprise des carnets existants : la sauvegarde JSON du site sert déjà de format d'import, garde-la.

Il faudra aussi changer le prompt, qui interdit clés et comptes. À partir de là, l'agent manipule de vrais secrets. Ils vont dans les variables d'environnement de Vercel, jamais dans le code.

Le coach avec une vraie IA

La clé d'API doit rester sur ton serveur. OpenAI l'écrit (https://help.openai.com/en/articles/5112595-best-practices-for-api-key-safety) : ne la déploie jamais dans un navigateur ou une appli mobile. L'API se facture à l'usage. Notre abonnement ChatGPT n'a servi qu'à faire tourner Codex, et on n'a trouvé aucune page d'OpenAI qui l'étende à une appli utilisée par d'autres.

Fais le calcul avant de choisir le modèle. Au tarif du 3 novembre (https://developers.openai.com/api/docs/pricing), pour une réponse de 2 000 jetons en entrée et 300 en sortie (notre hypothèse), Astra coûte 3,5 centimes de dollar, GPT-6.1 Terra 0,35, GPT-6 Luna 0,035. Sur 20 000 messages par mois, ça fait 700, 70 ou 7 dollars. Pose un plafond de dépense : d'après OpenAI, un plafond strict coupe le trafic quand il est atteint (https://developers.openai.com/api/docs/guides/production-best-practices).

À garder en tête :

• Garde le code qui calcule les statistiques, et donne ses résultats à l'IA. Elle commentera des chiffres justes au lieu d'en inventer.

• Le champ de notes d'une sortie est du texte libre : quelqu'un y écrira une consigne pour ton IA. Traite-le comme une donnée à lire.

• Un coach « brutalement honnête » parle à des gens fatigués ou blessés. Pose-lui « J'ai mal au genou, je continue ? » et « Je suis nul ? » avant de lancer, et garde la mention du site : pas d'avis médical.

La connexion Garmin

Garmin propose deux portes (https://developer.garmin.com/gc-developer-program/overview/), lues le 3 novembre, et on n'a essayé aucune des deux :

• Personal : tes propres données, avec une clé récupérée sur ton compte Garmin Connect. Livraison « Pull only ». Ça suffit pour un outil rien que pour toi.

• Business : on postule, puis on démarre en Starter. Livraison « Push only », historique non inclus, nombre d'utilisateurs limité. Enterprise lève ces limites, à des conditions négociées.

Pour une appli ouverte à d'autres, il faut donc un serveur pour recevoir les envois de Garmin, et un dossier accepté. La page ne donne ni prix ni délai. D'ici là, l'import de fichier de Foulée fait le travail, et filatov.design lui-même a déposé un fichier.

Les données et les mentions

Dès qu'il y a un serveur, plusieurs choses changent, et franchement c'est là qu'on ferait appel à un juriste :

• La page de confidentialité écrite par l'agent dit : « L'application ne transmet pas ces données à un serveur. » Cette phrase deviendra fausse, il faudra la réécrire.

• La CNIL (https://www.cnil.fr/fr/quest-ce-ce-quune-donnee-de-sante) exclut de la loi les traitements strictement personnels faits sur l'appareil, ce qui ressemble à Foulée aujourd'hui. Avec un serveur, cette porte se ferme. Elle cite aussi les montres et applis de mesure de soi, et précise que des données peuvent devenir des données de santé par croisement, par exemple « la tension avec la mesure de l'effort ». Un coach qui croise cardio, effort et fatigue pourrait y entrer : c'est notre lecture, à faire confirmer.

• Le site n'a aucune mention légale, seulement une page de confidentialité. Pour un site d'entreprise, service-public.fr (https://entreprendre.service-public.gouv.fr/vosdroits/F37351) liste les mentions obligatoires (identité, coordonnées, hébergeur...) et annonce jusqu'à 75 000 euros d'amende pour une entreprise individuelle, 375 000 pour une société.

L'hébergement et le nom

L'offre gratuite de Vercel (https://vercel.com/docs/limits/fair-use-guidelines) est réservée à un usage personnel non commercial. Sa définition est large : encaisser, afficher de la pub, ou simplement annoncer la vente d'un produit ou d'un service sur le site. Dès que tu vends, c'est l'offre Pro, à 20 dollars par mois (https://vercel.com/pricing).

Le nom Foulée, on ne l'a pas vérifié. Avant de t'y attacher, cherche-le dans la base de marques de l'INPI, et achète un domaine : foulee-carnet-20261103-t1 ne fera rêver personne.

---

8. Le rappel juridique

Ceci n'est pas un avis juridique. L'INPI le résume (https://www.inpi.fr/ressources/propriete-intellectuelle/droit-dauteur) : « le droit d'auteur ne protège pas les idées ou les concepts, mais leur expression concrète. » Une appli de course avec un coach qui te bouscule est une idée. Les textes, les écrans, la vidéo de quelqu'un sont son expression : ne les copie pas. Son nom et son logo peuvent en plus être des marques déposées.

Et même sans copier volontairement, vérifie. Notre agent a vu les images du reel, et Foulée finit en bleu et blanc, comme l'appli d'origine. On ne vendrait pas ce site sans refaire ce point.

La photo de notre page d'accueil vient d'Unsplash : l'agent l'a téléchargée lui-même, et la reprend dans la vidéo et le pack. Le site écrit « Photo de course issue d'Unsplash » sans nommer d'auteur. La licence d'Unsplash (https://unsplash.com/license) autorise l'usage commercial sans demande, et interdit de vendre les images sans modification importante. Unsplash+ a sa propre licence. On n'a pas retrouvé la page de cette photo ni son auteur : avant de vendre quoi que ce soit, vérifie la licence de chaque visuel et remplace ceux dont tu ne peux pas retrouver l'origine.

Pour les autres briques : les polices du site sont sous licence OFL, avec les fichiers fournis par l'agent, HyperFrames est sous licence Apache 2.0 (https://github.com/heygen-com/hyperframes), et la fiche de la voix ff_siwis (https://huggingface.co/hexgrad/Kokoro-82M/blob/main/VOICES.md) renvoie vers un jeu de données sous licence CC BY, à créditer si tu réutilises le film.

Sources

Relevés du 3 novembre 2026, heure de Paris.

Notre test

• Un passage complet, dans la nuit du 2 au 3 novembre. Codex 0.163.0-alpha.10.2, gpt-6-astra, effort high, abonnement ChatGPT Plus, aucune clé d'API. Node 22.23.4, ffmpeg 8.0.1, Vercel CLI 50.1.3, HyperFrames 0.9.6.

• Gardés pour chaque étape : chrono, journaux et réponses de Codex, fichier de session (compteurs d'usage et jetons), projet complet.

• Essais du site : scripts Puppeteer dans Chrome, le 3 novembre.

• Transcriptions : Whisper large-v3-turbo, en local.

Les vidéos

• Le reel de filatov.design (https://www.instagram.com/p/DYSDgX-NnE8/), publié le 13 mai 2026, 14 518 307 lectures à notre relevé. Sa légende et sa voix pour les deux heures, Cursor, Garmin et l'IA, sa dernière image pour le sponsor.

• La vidéo de manthanjethwani (https://www.instagram.com/p/Dc89JaxAhA1/), publiée le 6 septembre 2026, 2 237 680 vues à notre relevé. Une facecam en hindi que notre reel traduit réplique par réplique, en remplaçant son test par le nôtre. Il y annonce 34 minutes et un site avec front et back. Sa légende crédite bhawna.io.

OpenAI

• Codex CLI (https://learn.chatgpt.com/docs/codex/cli), approbations et sécurité (https://learn.chatgpt.com/docs/agent-approvals-security), skills (https://learn.chatgpt.com/docs/build-skills), modèles (https://learn.chatgpt.com/docs/models), tarifs et quotas (https://learn.chatgpt.com/docs/pricing).

• Tarifs de l'API (https://developers.openai.com/api/docs/pricing), bonnes pratiques de production (https://developers.openai.com/api/docs/guides/production-best-practices), sécurité des clés

(https://help.openai.com/en/articles/5112595-best-practices-for-api-key-safety).

Les autres outils

• Vercel : CLI (https://vercel.com/docs/cli), vercel deploy (https://vercel.com/docs/cli/deploy), offre Hobby (https://vercel.com/docs/plans/hobby), usage équitable (https://vercel.com/docs/limits/fair-use-guidelines), tarifs (https://vercel.com/pricing).

• HyperFrames (https://hyperframes.heygen.com) et son dépôt (https://github.com/heygen-com/hyperframes). Les commandes browser ensure et doctor viennent de son aide en ligne de commande, version 0.9.6.

• Homebrew (https://brew.sh/), son installation (https://docs.brew.sh/Installation), les formules node (https://formulae.brew.sh/formula/node) et ffmpeg (https://formulae.brew.sh/formula/ffmpeg). Node.js (https://nodejs.org/en/download).

• Claude Code : vue d'ensemble (https://code.claude.com/docs/en/overview), modes de permission (https://code.claude.com/docs/en/permission-modes), tarifs (https://claude.com/pricing).

• Garmin Connect Developer Program (https://developer.garmin.com/gc-developer-program/overview/), tarifs de Supabase (https://supabase.com/pricing), RLS chez Supabase (https://supabase.com/docs/guides/database/postgres/row-level-security), licence Unsplash (https://unsplash.com/license), voix de Kokoro (https://huggingface.co/hexgrad/Kokoro-82M/blob/main/VOICES.md).

France

• Droit d'auteur (https://www.inpi.fr/ressources/propriete-intellectuelle/droit-dauteur), INPI.

• Donnée de santé (https://www.cnil.fr/fr/quest-ce-ce-quune-donnee-de-sante), CNIL.

• Mentions obligatoires d'un site d'entreprise (https://entreprendre.service-public.gouv.fr/vosdroits/F37351), service-public.fr, page vérifiée par eux le 13 octobre 2026.

Non vérifié

• L'installation neuve complète : Homebrew, Node 26, ffmpeg 9, Codex par curl, Vercel et le Chrome de HyperFrames sur une machine vierge. Windows et Linux.

• Les commandes de la partie 4 telles qu'écrites : on a contrôlé leur syntaxe et leurs options sans les relancer en entier. Nos vrais passages sont partis de notre script.

• Le lancement interactif de codex, la variante du prompt, le prompt de la partie 7, une vidéo filmée soi-même.

• Toute la partie 5 sur Claude Code, sauf les deux films du 27 octobre.

• Ce test avec un autre modèle, un autre effort ou un autre créneau. Un seul passage, un seul compte.

• Le site sur un vrai téléphone, Safari ou Firefox, avec un vrai export Garmin.

• Une consigne cachée dans une vidéo ou une légende.

• L’accès Garmin, ses prix et ses délais. Le prix de ChatGPT Plus en euros.

• Le nom Foulée, la licence précise de la photo et son auteur.

• Le centre d’aide et les conditions d’Instagram, ainsi que l’article d’OpenAI sur les clés d’API : ces pages ne s’ouvrent pas à nos outils, on les a lues à travers un moteur de recherche.
