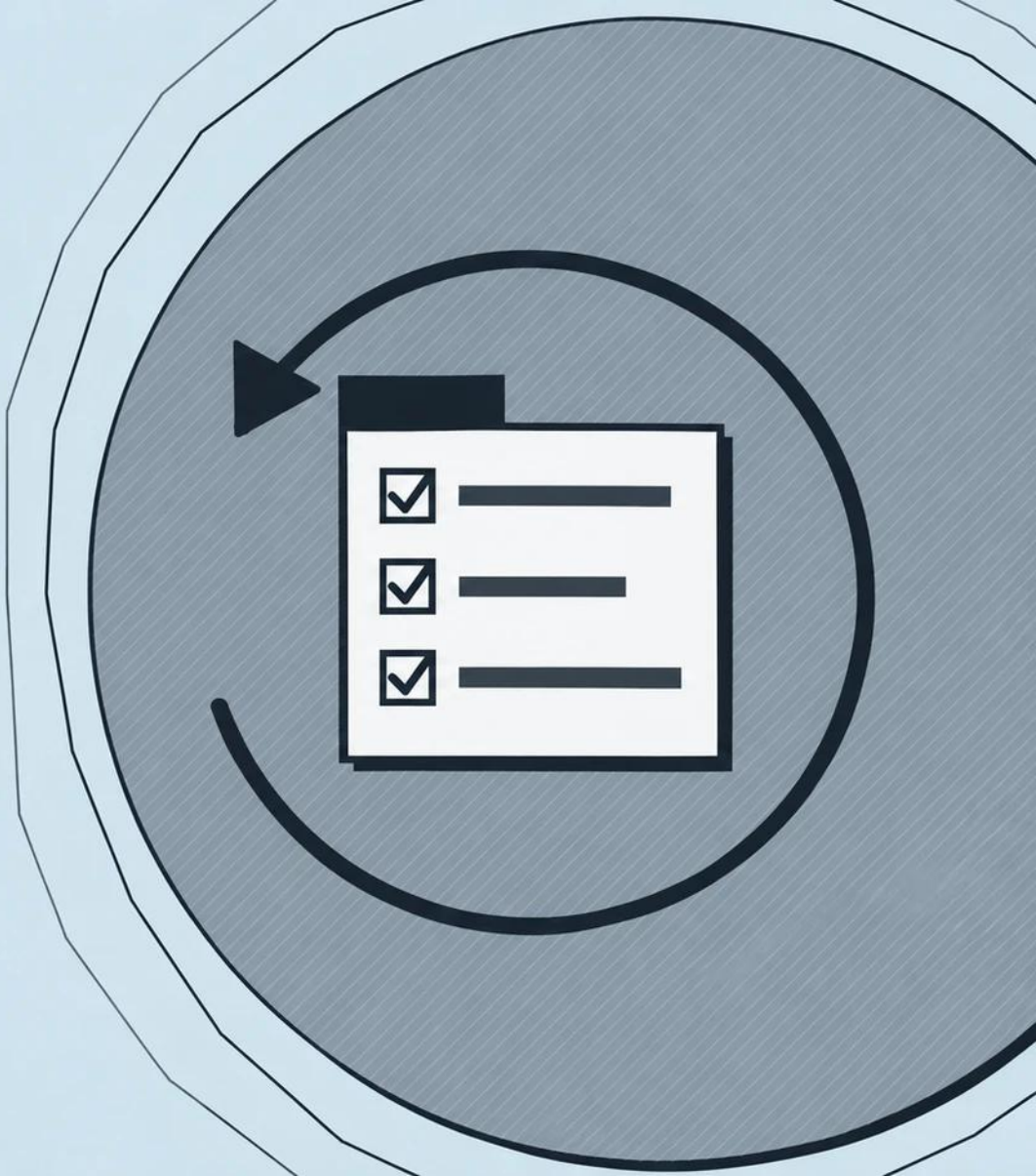
S2

SÉANCE 2 · POUR ALLER PLUS LOIN

Faire son app de suivi

IFOSUP Wavre · 2026-2027

5XVTE · VEILLE TECHNOLOGIQUE · S2



? Le site de synthèse montre vos résultats. Que lui manque-t-il pour servir chaque semaine ?

Une application dit quoi lire d'abord, retient vos décisions et montre celles du groupe.

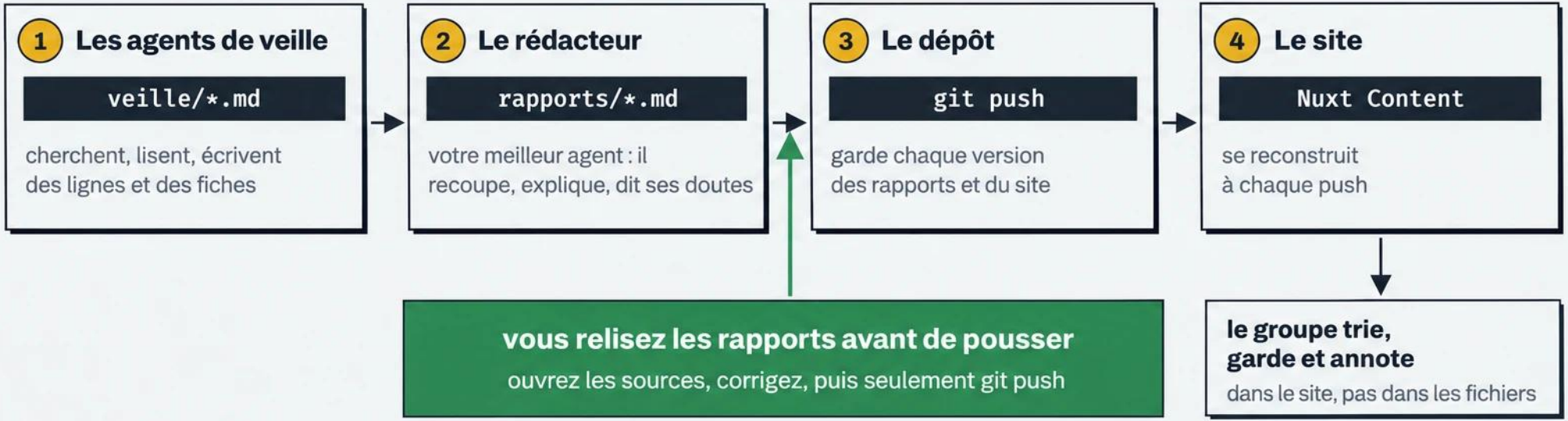
LE SITE DE SYNTHÈSE	UNE APPLICATION
✕ **Il ne retient rien de vous.** la semaine suivante, tout est à relire	✓ **Elle retient vos décisions.**
✕ **Il ne dit pas quoi lire d'abord.** vingt nouveautés ont toutes le même poids	✓ **Elle range les nouveautés par importance.**
✕ **Il ne sert qu'à vous.** vos coéquipiers ne voient pas ce que vous avez gardé	✓ **Elle montre les décisions du groupe.**

l'application de ce chapitre quatre tours de corrections et un testeur

Ce chapitre est facultatif : il prolonge « Écrire son skill », chez vous. L'application que vous allez voir n'est pas sortie d'une seule demande, et c'est l'idée principale du chapitre.

? Qui écrit ce que le site du groupe affiche ?

Les agents écrivent des fichiers, vous les relisez, et le site ne fait que les montrer.

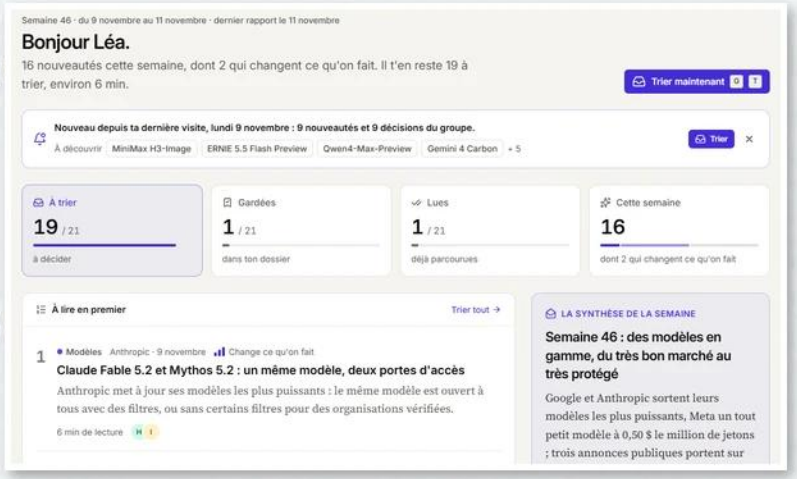


Personne n'écrit de rapport dans l'application : pour en ajouter un, on ajoute un fichier. Un agent peut faire le travail, et git en garde l'historique. Pousser, c'est envoyer vos dernières versions sur GitHub.

? Dix-neuf nouveautés restent à trier. Laquelle lisez-vous d'abord ?

L'accueil dit quoi lire d'abord, et le tri montre une nouveauté à la fois.

l'accueil



À trier



1. une phrase résume la semaine
2. « À lire en premier » : trois nouveautés, les plus importantes
3. l'essentiel en trois phrases, puis une touche : 1 garder, 2 lu, 3 plus tard, 4 écarter

Vous lisez l'essentiel, vous décidez d'une touche, et la nouveauté suivante arrive : dix-neuf nouveautés se trient en quelques minutes. La touche U annule la dernière décision.

? Une ligne de veille dit qu'une nouveauté existe. Qui vous l'explique ?

Un rapport sépare ce que dit la source de ce que l'agent explique, et il dit ses doutes.

un rapport de l'application


Modèles Google mardi 10 novembre Change ce qu'on fait

Gemini 4 Carbon : le modèle le plus puissant de Google, à prix réduit jusqu'au 30 novembre

Google ouvre son modèle le plus puissant, pensé pour la sécurité informatique et les longues tâches d'agent, à un prix qui double après le 30 novembre.

Gardé | 5 min de lecture | Confiance haute | Vu par Claude Code, Hermes, opencode

L'essentiel en trente secondes

1. Google rend disponible Gemini 4 Carbon, qu'il présente comme son modèle le plus puissant, surtout en cybersécurité et pour les longues tâches d'agent.
2. Par l'API, il coûte 2 $ par million de jetons en entrée et 10 $ en sortie jusqu'au 30 novembre 2026, puis 4 $ et 20 $.
3. Google conseille lui-même Gemini 4 Argon pour la plupart des tâches : Carbon vise les cas difficiles et consomme plus de jetons.

LES PARTIES D'UN RAPPORT

L'essentiel en trente secondes	trois phrases
Pourquoi on en parle	ce qui la rend importante
Ce que c'est	l'explication, signée « Explication du rédacteur »
Ce que dit la source	les faits, tels que l'agent les a lus
Ce que ça change pour moi	pour un développeur, un designer
Limites et doutes	ce qui n'est pas prouvé
Niveau de confiance	dit par la source, expliqué par le rédacteur, ou non vérifié

vous savez toujours qui parle

Le rédacteur écrit un rapport par nouveauté importante, toujours avec les mêmes parties. Dans l'application, un mot souligné ouvre sa définition sans quitter la page.

? Un deuxième agent qui relit le premier, est-ce bien utile ?

Le rédacteur rouvre les sources et trouve des erreurs, mais il se trompe aussi.

CE QU'UN AGENT DE VEILLE A ÉCRIT	CE QUE DIT LA PAGE D'ORIGINE
✕ **« 50 à 70 % inférieur »**	✓ `50–70% of comparable mainstream offerings` 50 à 70 % du prix : 30 à 50 % de moins
✕ **Gemini 4 Carbon, « à tous les développeurs et au grand public »**	✓ une API payante et, pour le grand public, les seuls abonnés à l'offre la plus chère

il se trompe aussi : 11 rapports sur 21 « En bref »
la veille reprise, sans rouvrir la source

vous, avant de pousser
ouvrez deux rapports, et pour chacun la source

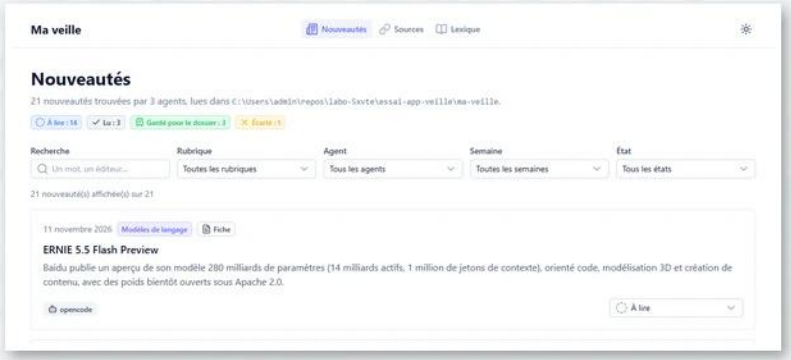
Le rédacteur écrit ces erreurs de la veille dans le rapport, à la partie « Limites et doutes ». C'est la raison d'être de cette étape : un deuxième agent qui relit le premier en rouvrant la source.

S2 VOUS

? La première version de l'application marchait. Pourquoi ne pas s'arrêter là ?

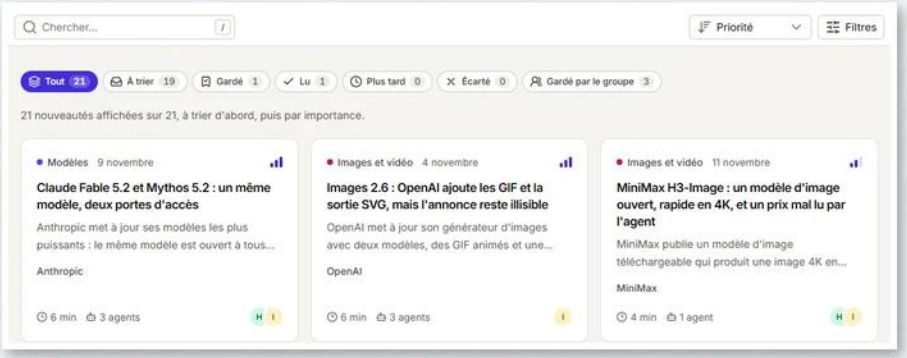
La première version avait tout, mais rien ne disait quoi lire d'abord.

la première version : la liste des nouveautés



quatre tours →

la même liste, quatre tours plus tard



- ✗ rien ne dit quoi lire d'abord
- ✗ les vingt et une lignes se ressemblent
- ✗ le suivi de vos décisions : un formulaire au milieu de la page

- ✓ un titre en gras tant que la nouveauté n'est pas triée
- ✓ des phrases entières
- ✓ un filtre : ce que le groupe a gardé

Un agent qui a fait tout ce que vous lui demandiez dit « c'est fini ». C'est à vous de dire que ça ne l'est pas.

? Comment trouver ce qui ne va pas dans une application qui marche ?

À chaque tour, l'agent lance, capture, critique et corrige, et vous regardez avec lui.



CE QUE LES CAPTURES ONT MONTRÉ

TOUR 1	TOUR 2	TOUR 3	TOUR 4
sur un téléphone, les boutons de tri sous trois écrans de texte	« À trier » sur toutes les cartes : du bruit	une grande tuile « Médias : 0 », pour dire « rien »	le rapport du testeur

Aucun de ces défauts n'était dans la demande de départ : ils se voient sur l'écran, pas dans la consigne. Ajoutez à la liste ce qui vous gêne, vous : c'est votre outil.

? Après trois tours, l'agent qui construit ne voit plus ses défauts. Qui les trouve ?

Un testeur qui ne connaît pas le projet trouve ce que trois tours n'avaient pas vu.

l'agent qui construit
après trois tours, il regarde
les endroits où ça marche

le testeur, dans une nouvelle conversation
il utilise l'application comme un étudiant
et dit ce qui ne va pas, sans rien corriger

DEUX DÉFAUTS TROUVÉS

1 **un mot du lexique affiché en code**

`l':termAPI payante`

au lieu de « l'API payante », dans dix rapports

2 **un suivi en retard de trente secondes**

deux navigateurs côte à côte,
et rien ne le disait

✓ **corrigé : quelques secondes**

Le testeur a aussi dit ce qui manquait pour s'en servir chaque semaine : c'est ajouté. Des défauts connus restent : une application n'est jamais finie, et vous décidez où vous arrêter.

? Faut-il chercher la demande parfaite avant de commencer ?

Faites une première demande claire, puis des tours, et finissez par un testeur.

LA PREMIÈRE DEMANDE

- dans votre agent le plus solide
- d'abord un `SPEC.md` : les écrans, les fichiers, le suivi
- avant « fini » : le code vérifié, chaque écran ouvert

→

UN TOUR à répéter

- des captures en grand, en sombre, sur téléphone
- la liste de ce qui gêne, avant de corriger
- `ITERATIONS.md`

→

LE TESTEUR

- une nouvelle conversation, ou un autre agent
- il ne modifie rien
- `TEST-CRITIQUE.md`

`nuxt-ui` **un skill qui rassemble une vingtaine d'erreurs et la marche à suivre**
le zip du chapitre, à décompresser dans `.agents\skills\`

Les demandes complètes sont dans le chapitre. Elles résument ce que le professeur a demandé à ses agents, sans avoir été rejouées mot pour mot : adaptez-les, et comptez du temps.

? Dans un groupe, comment vos coéquipiers voient-ils les rapports et vos décisions ?

Le contenu se partage par git, et le suivi par une petite API fermée par un code.



Une décision doit arriver tout de suite chez les autres : la pousser dans git serait trop lourd. L'API, ce sont quelques adresses du site qui lisent et écrivent le suivi.

? Une application en ligne a une adresse. Qui peut appeler son API ?

Une API ouverte laisse n'importe qui remplir votre suivi, et un code de groupe la ferme.

1. **Celui qui met le site en ligne choisit un code**
   rangé dans les réglages de l'hébergeur du site, jamais dans le dépôt
2. **Chaque membre tape ce code une fois, avec un pseudo**
   le navigateur les retient
3. **Chaque écriture envoie le code**
   sans le bon code, l'API refuse

⚠ **pas un vrai compte**
un membre peut écrire sous le pseudo d'un autre ;
si le code fuit, il faut le changer

le bloc « Le groupe » d'un rapport



une décision y apparaît en 1,5 à 8 secondes

Pour un groupe de trois qui se fait confiance, cela suffit ; une application ouverte au public demanderait de vrais comptes. Le suivi ne contient qu'un pseudo, un état et une note : ni nom complet, ni adresse e-mail.

Rien n'a été mis en ligne pour le cours : vérifiez chaque point avant de vous y fier.

LA METTRE EN LIGNE

La base de données en ligne n'a pas été essayée : vérifiez chaque point avec votre agent.

- Vercel héberge une application Nuxt gratuitement, pour un usage personnel.
- Le dépôt doit être sous un compte GitHub personnel, pas d'une organisation.
- Le suivi va dans une petite base de données, pas dans un fichier.
- Le code de groupe et les accès à la base vont dans les réglages de Vercel, jamais dans le dépôt.
- Les rapports sont publics : le code protège le suivi, pas la lecture.

VÉRIFIER, EN UN QUART D'HEURE

- [ ] Comptez vos lignes de veille, puis comparez au nombre de nouveautés affiché.
- [ ] Prenez une ligne au hasard dans chaque fichier de veille, et retrouvez-la.
- [ ] Dans deux rapports, vérifiez un chiffre et une date contre la source.
- [ ] Avec deux navigateurs, mesurez le temps qu'une décision met à arriver.

De ce chapitre, gardez ces six idées.

À RETENIR

1. Un site de synthèse montre vos résultats. Une application dit quoi lire d'abord, retient vos décisions et montre celles du groupe.
2. La chaîne : les agents de veille écrivent des lignes, votre meilleur agent en fait des rapports, vous relisez, vous poussez, le site se reconstruit.
3. Un rapport sépare ce que dit la source de ce que l'agent explique, et dit ses doutes.
4. La première version marche, elle n'est pas pratique. Un tour : lancer, capturer, critiquer, corriger. Puis un testeur qui ne connaît pas le projet.
5. Le contenu se partage par git. Le suivi se partage par une API, fermée par un code de groupe : jamais une API ouverte.
6. Rien n'est sûr tant que vous ne l'avez pas vérifié : le compte, deux rapports contre leur source, le suivi à deux navigateurs.