14

PROJET 2 · ATELIER · SÉANCE 3

Votre application, puis votre skill

Thibault Six · IFOSUP Wavre · 2026-2027



La séance enchaîne votre application seul, le kit, le skill du groupe, puis votre projet

Seul, vous construisez une application de votre choix en quatre tours. En groupe, vous posez le kit de départ dans le dépôt du projet 2, puis vous écrivez le skill de votre stack.



Choisissez une application qui vous servirait vraiment, ou un petit jeu : rien n'est imposé

Pour	Idées
Votre vie d'étudiant	un minuteur de révision qui récompense la concentration · des cartes de révision tirées de vos notes
Votre travail	un suivi des heures par client · un générateur de comptes rendus de réunion
Votre vie de tous les jours	le partage des dépenses d'une colocation · un suivi d'entraînement
Jouer	un quiz entre amis · un jeu de mémoire



Vous pouvez prendre plusieurs écrans, une base de données, la stack de votre choix. L'application ne va pas dans le dépôt du projet 2. Créez un dossier vide, tapez `git init` dedans, puis ouvrez ce dossier dans votre agent.

La fiche tient en cinq lignes, et « ce qu'elle ne fait pas » est celle où vous coupez

- Pour qui
- Ce qu'elle fait
- **Ce qu'elle ne fait pas**
- Stack
- Fini veut dire

Ligne	L'exemple « Pousse », un minuteur de révision
Pour qui	moi, étudiant en cours du soir, qui révise sur mon portable ou sur mon téléphone
Ce qu'elle fait	une session de travail chronométrée [...], une plante dessinée qui grandit en cinq étapes [...]
Ce qu'elle ne fait pas	pas de compte, pas de serveur, rien n'est envoyé sur le réseau. Pas de son, pas de notification
Stack	Vite et TypeScript sans framework, Vitest pour les tests [...]
Fini veut dire	l'application tourne en local, la logique [...] est testée, la page se lit sur un téléphone, et il y a un commit

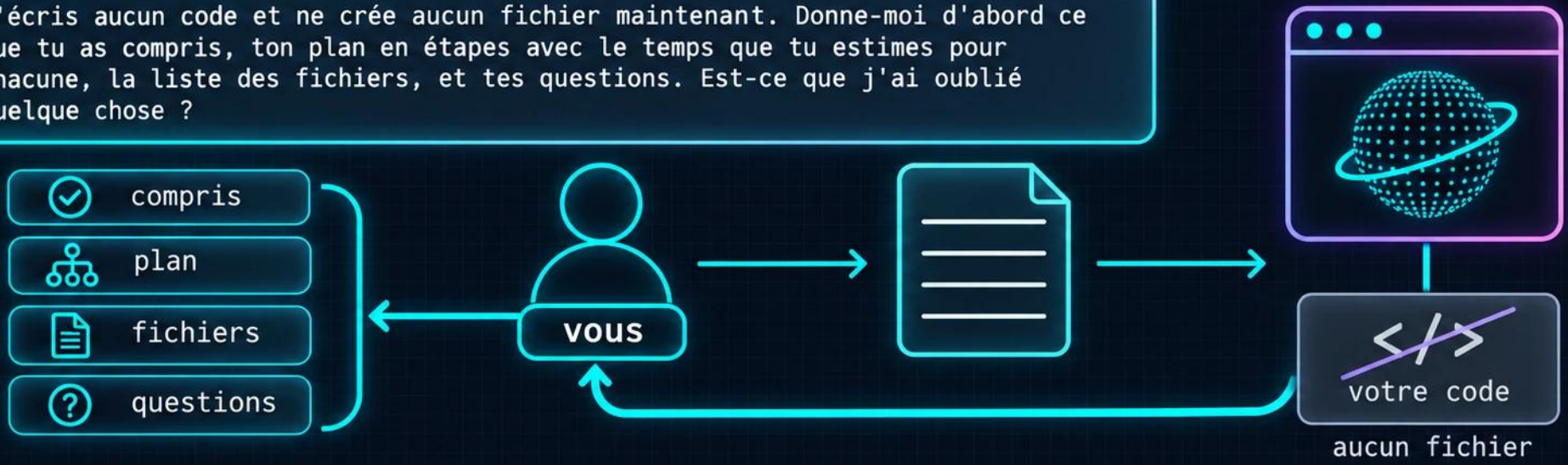
Racontez d'abord votre idée en quelques phrases, comme à un ami. Ajoutez ces cinq lignes. C'est fini quand les cinq sont écrites.

Au tour 1, l'agent lit votre fiche et pose ses questions, sans écrire une ligne de code

Votre idée et votre fiche entrent dans la demande, puis cette fin, mot pour mot.

votre demande

```
N'écris aucun code et ne crée aucun fichier maintenant. Donne-moi d'abord ce
que tu as compris, ton plan en étapes avec le temps que tu estimes pour
chacune, la liste des fichiers, et tes questions. Est-ce que j'ai oublié
quelque chose ?
```



C'est fini quand l'agent a répondu sans créer aucun fichier. Son plan annonce des heures ? Ne coupez pas votre idée : un agent estime en heures d'humain, et il va dix à vingt fois plus vite.

Au tour 2, répondez ligne par ligne, coupez ce qui déborde, puis l'agent construit tout

Copiez-collez son message et répondez sous chaque ligne, en commençant par >. Un exemple à adapter, la fin de la réponse pour « Pousse » ;

```
la réponse pour Pousse (fin)

6. Simple page web, ou installable (PWA) ?
> simple page web. Pas de PWA, hors périmètre.

Ton plan me va, mais 6 heures c'est une estimation d'humain : tu vas beaucoup
plus vite. Construis tout maintenant, en une fois. Pour que je puisse tester
sans attendre 25 minutes, ajoute un paramètre d'adresse ?demo=1 où une session
dure 60 secondes.

Quand c'est fait, lance les tests et le build et montre-moi leurs sorties
réelles. Termine par deux listes : ce que tu as vérifié, ce que tu n'as pas
vérifié. Ne fais pas de commit, je veux d'abord voir l'écran.
```

1. elle coupe
2. tester vite
3. des preuves, pas de commit

C'est fini quand l'agent montre ses sorties réelles et ses deux listes, sans commit.

Au tour 3, regardez l'écran vous-mêmes, puis écrivez ce que vous voyez et ce que vous voulez

Question : les 60 tests et le build sont verts. L'écran est-il bon pour autant ?

Lancez l'application (`npm run dev` ici). F12 puis `Ctrl+Maj+M` montrent la page à la largeur d'un téléphone. Faites un vrai parcours, du premier clic au résultat.

votre critique

```
J'ai regardé l'écran. (Nombre) choses me gênent :
1. Sur (téléphone / grand écran), (ce que vous voyez). (Ce que vous voulez à la place.)
2. (…)

Corrige ces points, rien d'autre. Relance les tests et le build, et dis-moi
ce que tu as vérifié et ce que tu n'as pas vérifié. Toujours pas de commit.
```

C'est fini quand vos corrections se voient à l'écran.



après le tour 2 : tests verts, plante perdue



après le tour 3

Au tour 4, regardez l'écran une dernière fois, puis demandez le commit et contrôlez-le

C'est fini quand `git log --oneline` montre votre commit et que `git status` ne signale plus rien. Votre poste est prêt si l'agent a lancé l'application, montré une sortie réelle et un résultat que vous avez vu de vos yeux, puis fait le commit à votre demande.

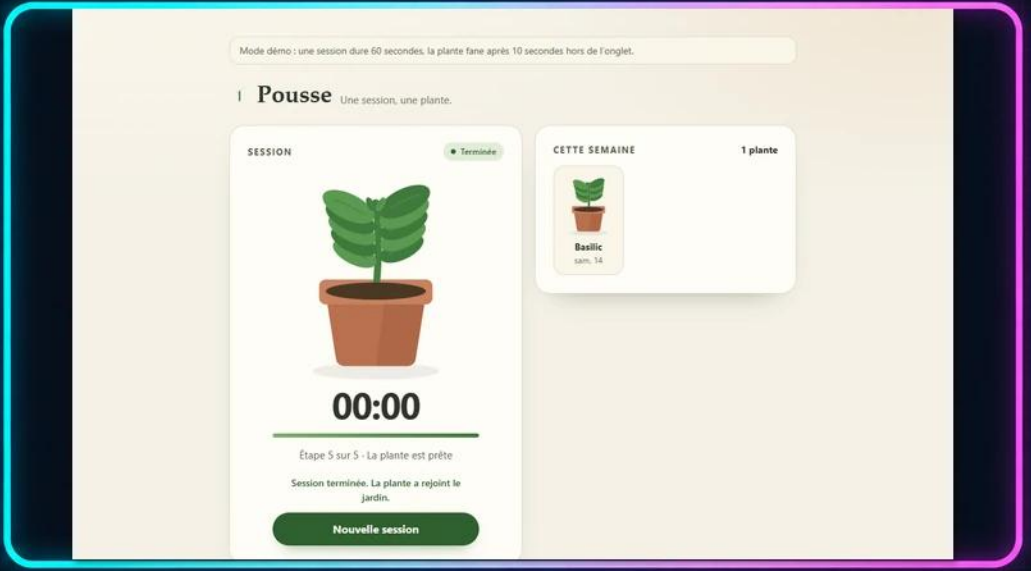
votre demande

```
C'est bon pour moi. Fais le commit, message clair
en français, ne pousse rien.
```

terminal

```
git log --oneline
git status
```





ce qui entre dans le commit

Montrez la page à quelqu'un qui n'a rien lu : elle doit dire seule à quoi elle sert et pourquoi

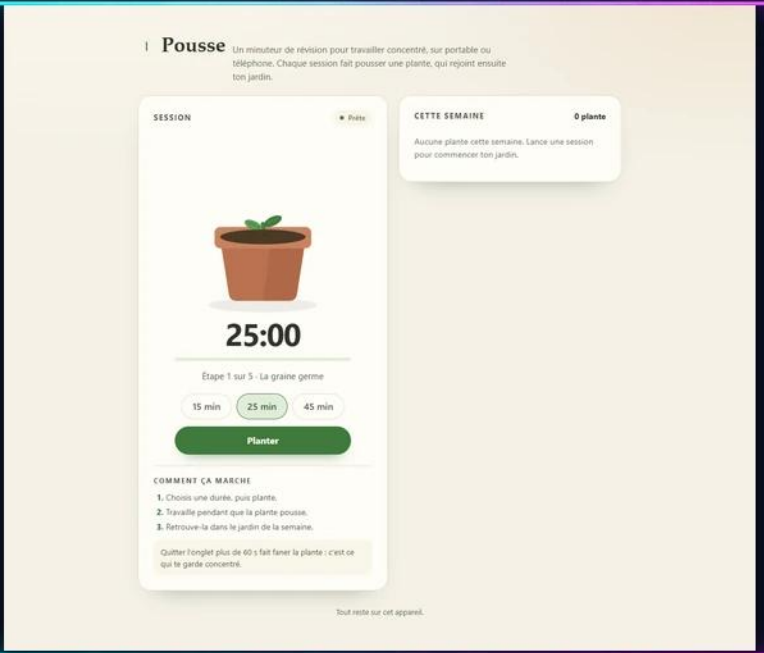
Question : que comprend de votre page quelqu'un qui n'a rien lu de votre idée ?

La page répond à quatre questions : à quoi sert l'application, comment elle fonctionne, par où commencer, pourquoi il se passe ce qui se passe. Reprenez la remarque telle qu'elle a été dite, comme dans cet exemple pour « Pousse ». C'est fini quand la page répond seule, puis un commit.

la demande pour Pousse (début)

J'ai montré l'application à quelqu'un qui ne connaissait pas l'idée. Sa réaction : « on comprend rien à quoi sert l'application ». Il s'est demandé quel est le but, comment ça fonctionne, à quoi ça sert, et pourquoi la plante fane.

Ajoute sur la page de quoi répondre à ces questions, sans rien lire ailleurs : […]



la page d'accueil après ce tour

Un membre pose le kit sur une branche, remplit AGENTS.md, puis le groupe relit sa PR

Il télécharge le kit en tête du chapitre et le copie à la racine, dossier caché `.agents/` compris. Dans `AGENTS.md`, il remplit le nom du projet, la phrase qui le présente et le tableau du groupe. Chacun donne sa valeur de `git config user.name` : l'agent sait ainsi pour qui il travaille. C'est fini quand la PR est fusionnée.



AGENTS.md

Prénom	Nom d'auteur git (`git config user.name`)	Dossier	Travaille sur
Léa	lea-dupont	`docs/lea/`	la génération des menus

Après la fusion, chacun demande « Mets en place ma bureaucratie. » et obtient son dossier

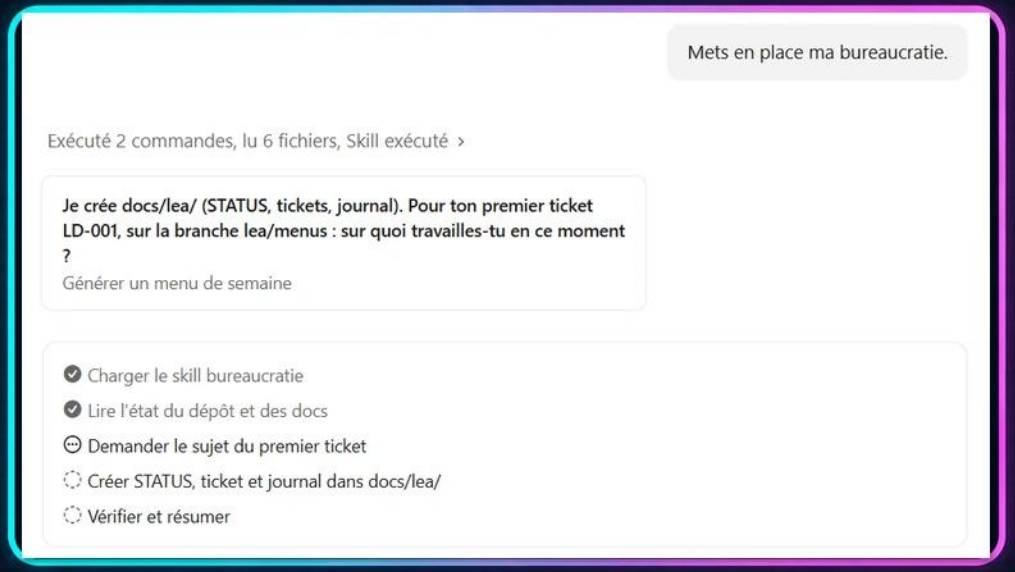
Récupérez `main` et créez votre branche. Votre sujet devient votre premier ticket, une fiche par tâche. Votre agent n'écrit que dans votre dossier. C'est fini quand ces trois fichiers existent : commitez-les, ils rejoignent `main` par votre PR.

votre demande

```
Mets en place ma bureaucratie. En ce moment je travaille
sur (votre partie du projet, en une ou deux phrases).
```

ce qui doit exister

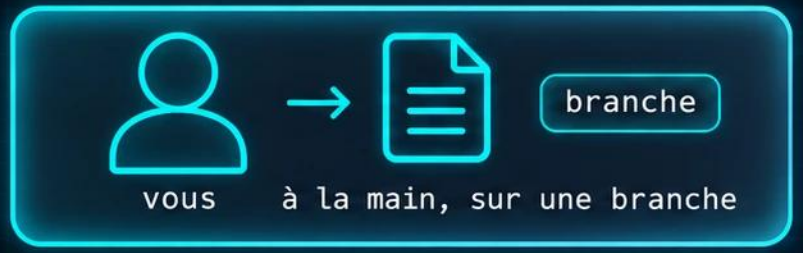
```
docs/lea/
├── STATUS.md
|    18 lignes, trois actions numérotées
├── tickets/LD-001-generation-menus.md
|    cinq critères vérifiables par oui ou par non
└── journal/AAAA-MM-JJ.md
```



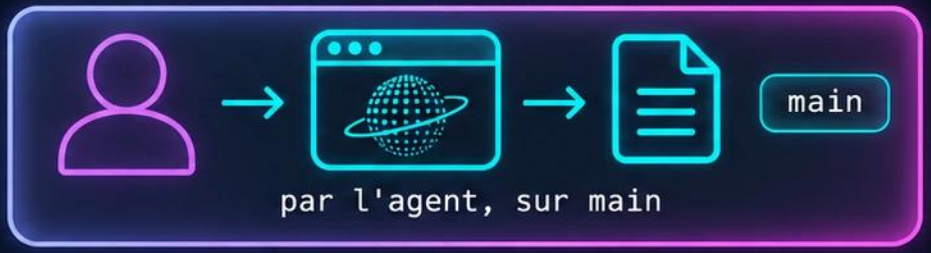
l'agent charge le skill bureaucratie

Écrivez le skill de votre stack à la main, et un autre membre le fait écrire par l'agent

Un membre, sur une branche, renomme `.agents/skills/ma-stack` au nom de la stack (`stack-django-htmx`). Il commence par « Nos règles », avec des limites qu'on peut compter, puis fait le commit « v1 : écrite à la main ». Pendant ce temps, un autre membre reste sur `main` et demande à l'agent sa version. C'est fini quand la v1 est commitée et que `skill-agent/SKILL.md` existe.



l'agent ne voit pas votre version



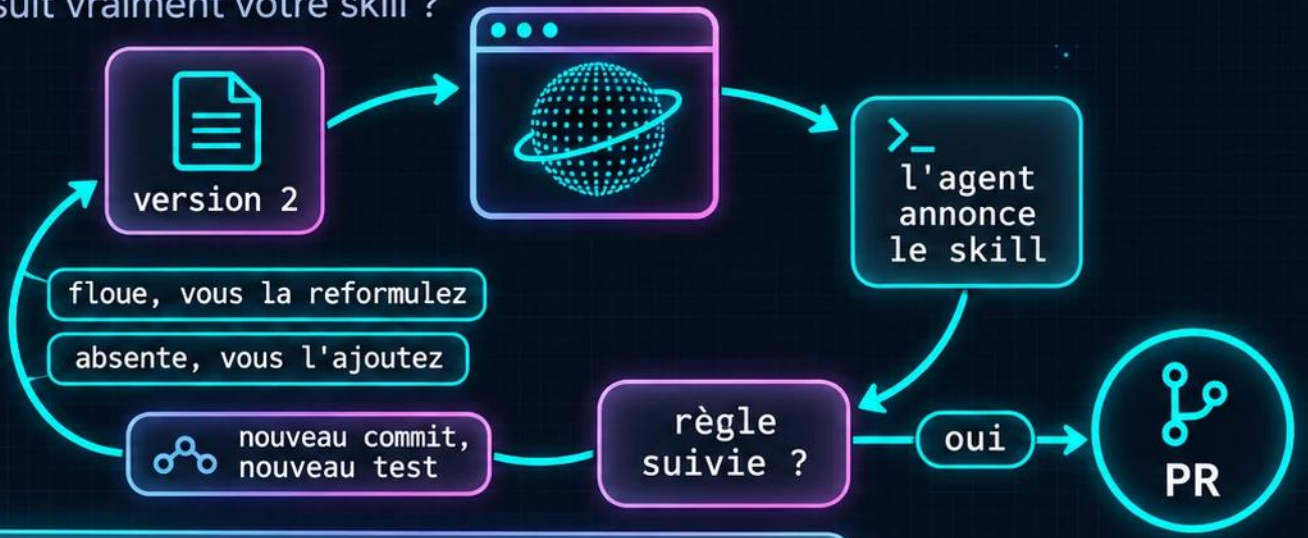
la demande de l'autre membre

```
Écris un skill pour la stack de notre projet : (frameworks, base, bibliothèque
du modèle de langage, hébergeur). Si tu as un skill qui écrit des skills,
utilise-le.
Lis d'abord les fichiers de dépendances du dépôt pour trouver les versions
réelles. N'ouvre pas le dossier .agents/skills/ma-stack : je veux ta version.
Le skill dit comment le code doit être écrit, les vérifications dans l'ordre
et les pièges connus. Enregistre-le dans skill-agent/SKILL.md et ne fais
aucun commit.
```

Gardez le meilleur des deux versions, puis testez le skill dans une session neuve

Question : comment savoir qu'un agent suit vraiment votre skill ?

Cherchez ce que l'agent a prévu et pas vous, ce que vous avez prévu et pas lui, ce qui reste flou. La version 2 garde le meilleur des deux : relisez-la ligne par ligne, puis commitez. C'est fini quand l'agent annonce le skill, lit les versions de vos dépendances et suit vos règles.




```
nouvelle session
(Une petite tâche de votre projet 2, par exemple : ajoute la page qui liste les réservations.) Propose un plan d'abord.
```

Livrable : `.agents/skills/<stack>/SKILL.md`, dans une PR relue par le groupe puis fusionnée.

À la fin de la séance, vous avez une application commitée, un skill en PR et un dossier à jour



- **Tester :** donnez à l'agent une vraie tâche de votre ticket
- **Valider :** une règle qui a tenu sur deux ou trois tâches reste telle quelle
- **Améliorer :** quand vous corrigez l'agent deux fois pour la même chose, la correction entre dans le skill

Avant de fermer, faites mettre à jour `docs/<prénom>/`, relisez les cases cochées de vos tickets, puis faites le commit.

fermer la session

```
J'ai fini pour aujourd'hui. Fait : (ce qui marche, et sa preuve). Pas commencé : (…). Problème rencontré : (le problème, et ce qui l'a réglé). Mets tout à jour.
```