🏆 Projet : Inventaire de parc informatique

Construire en mini un vrai inventaire de parc informatique : collecte PowerShell, API PHP, dashboard et fiches par PC.

5TQ Synthèse

Imagine une entreprise avec 200 ordinateurs. Comment savoir lesquels manquent de mémoire ? Lequel n'a pas reçu les dernières mises à jour ? Combien d'espace disque reste-t-il sur le poste de la compta ? À la main, c'est impossible.

C'est là qu'intervient un **inventaire de parc informatique** : un système qui collecte automatiquement les informations de chaque PC et les centralise pour les techniciens. Tu vas en construire un, en miniature, mais qui fonctionne pour de vrai.

🎯 Objectifs

À la fin de ce projet, tu seras capable de :

1. **Collecter** des informations système d'un PC Windows avec PowerShell.
2. **Construire** une API PHP minimale qui reçoit du JSON et le stocke proprement.
3. **Lire** ces données et les afficher dans un dashboard Bootstrap 5 + Chart.js.
4. **Ajouter** un module spécifique à ton domaine d'expertise.
5. **Sécuriser** une API avec un token et comprendre les limites d'un tel système.
6. **Travailler avec une IA** comme un collègue débutant doué — pas comme un distributeur de solutions.

Ce que tu vas construire

Un site web PHP qui :

- reçoit en JSON les rapports envoyés par chaque PC du parc ;
- les stocke dans des fichiers (pas de base de données : on reste simple) ;
- affiche un **dashboard global** de la classe ;
- propose une **fiche détaillée par PC** avec graphiques et historique.

En parallèle, tu écriras un **script PowerShell** qui interroge ton propre PC Windows, met les résultats en forme et les envoie à l'API.

À la fin, le parc de toute la classe sera visible dans le même tableau de bord.

Ce que tu vas pratiquer

- **PHP** : routage simple, lecture/écriture JSON, affichage de données
- **PowerShell** : commandes `Get-*`, conversion JSON, `Invoke-RestMethod`
- **HTTP** : méthodes `GET` / `POST`, en-têtes, codes de statut
- **JSON** : structure, parsing, validation
- **UI** : Bootstrap 5, Chart.js
- **Sécurité** : pourquoi et comment protéger une API
- **Méthode** : travailler proprement **avec une IA**, sans qu'elle fasse tout

Organisation

Séance	Durée	Objectif principal
Séance 1	2 h	PowerShell → JSON → API PHP de réception
Domicile	~1 h	Affichage de base d'une fiche PC
Séance 2	2 h	Dashboard, graphes, historique, sécurisation token
Domicile	~1 h	Module spécifique (ton domaine d'expertise)

Chaque étape a :

- un **objectif** clair (ce que tu dois obtenir),
- des **consignes** pas-à-pas,
- des **extraits de code** que tu dois comprendre, pas copier sans réfléchir,
- des **pièges** identifiés à éviter,
- des **choix** où tu décides toi-même,
- un encart « **Travailler avec l'IA** » pour t'aider à prompter intelligemment.

Tronc commun et module spécifique

Tout le monde construit le **même tronc commun** : infos système, CPU, RAM, disques, processus, applications installées, fichiers récents.

En plus, chaque élève reçoit **un domaine spécifique** qu'il devient seul à maîtriser pour la classe. Les 8 domaines sont :

1. Réseau et configuration IP
2. Sécurité et patchs Windows
3. Utilisateurs et groupes locaux
4. Services et programmes au démarrage
5. Périphériques USB et matériel branché
6. Tâches planifiées
7. Connexions réseau actives
8. État de santé du disque physique

Le numéro qui te correspond est affiché sur le tableau. Ton module enrichit l'inventaire commun avec une section qui te distingue.

Travailler avec l'IA — la règle d'or

Tu peux utiliser **Codex** (ou Claude, ou tout autre assistant) pendant ce projet. **Mais** : l'IA n'est pas un distributeur de solutions. C'est un **collègue débutant doué** que tu diriges. Si tu ne sais pas ce que tu veux, elle te livrera du code qui ne sert à rien.

> 🧠 Trois règles, à chaque étape :
>
> 1. **Tu décris d'abord le résultat attendu** avant de demander du code.
> 2. **Tu lis et tu comprends** chaque ligne avant de la coller.
> 3. **Tu testes par petit morceau**, jamais 200 lignes d'un coup.

Tu verras dans chaque chapitre des **encarts** intitulés « Travailler avec l'IA ». Ils te montrent un prompt-type, ce qu'il faut donner comme contexte, et comment vérifier la réponse.

Premier exemple de prompt mal et bien formulé

❌ **Mauvais prompt :**

> Fais-moi un site PHP d'inventaire.

Trop vague. L'IA va inventer une architecture, choisir une base de données, proposer un framework… et tu vas être perdu.

✅ **Bon prompt :**

> Je suis élève de 5e secondaire en option info. Je commence un mini-projet PHP. Niveau attendu : débutant (pas de framework, pas de POO complexe). Le HTML peut utiliser Bootstrap 5. Première étape : j'ai besoin d'un endpoint PHP qui reçoit un POST avec un corps JSON, vérifie qu'il contient bien un champ

> `hostname`, et l'enregistre dans `data/pcs/{hostname}/latest.json`. Donne-moi le code minimal, commenté, sans router framework.

Tu fixes : ton niveau, les contraintes, l'objectif précis, le périmètre. Résultat : du code utile, lisible, juste ce qu'il faut.

Écoresponsabilité — pourquoi un inventaire ça compte aussi pour la planète

Un PC qui rame n'est pas forcément un PC à jeter. Un inventaire bien tenu permet de :

- repérer les machines **sous-utilisées** (à mutualiser plutôt que racheter) ;
- identifier les disques **en fin de vie** avant qu'ils ne lâchent (et qu'on doive tout remplacer en urgence) ;
- détecter les **logiciels inutiles** qui consomment du CPU pour rien ;
- prolonger la durée de vie d'un parc, ce qui est **le levier numéro un** de l'impact environnemental d'un parc informatique (la fabrication représente ~75 % de l'empreinte carbone d'un PC, pas son usage).

Tu intégreras dans ton dashboard au moins **un indicateur** lié à ces enjeux.

Avant de commencer

Vérifie que tu as bien :

- [ ] **Laragon** installé et fonctionnel (Apache + PHP)
- [ ] **PowerShell 7** ou Windows PowerShell 5.1 minimum
- [ ] **VS Code** avec extensions PHP et PowerShell
- [ ] **Postman** ou **Thunder Client** (pour tester l'API)
- [ ] **Git** initialisé sur ton dossier projet

Si quelque chose manque, tu sais quoi faire : lever la main *ou* demander à l'IA « comment installer X sur Windows », en lisant la réponse avant de lancer un installeur.

Suite

Allez, on attaque l'étape 1 : collecter avec PowerShell.