

Rapport d'expérience : Exploitation de l'IAg dans la SAÉ 301

Ressource R3.10 - Initiation à l'IAg

Groupe : 4-02

Membres :

AIGNELOT Youenn
LACHAISE Mattys
BERNARD Adam
FILMONT Félix
PLU Niels

1. Description de la tâche sélectionnée

Dans le cadre du développement de notre projet "TapisMarket" (SAÉ 301), nous avons choisi de travailler sur la **gestion du panier d'achat** (Module R3.01 - Développement Web).

La tâche spécifique consistait à **sécuriser et robustifier l'affichage des produits dans le panier**, en particulier la gestion des images. Le système doit gérer deux cas de figure complexes :

1. Les utilisateurs invités (panier stocké en session).
2. Les utilisateurs connectés (panier stocké en base de données).

De plus, une problématique technique critique a émergé : les images des produits ne s'affichent pas correctement (erreurs 404) car les chemins d'accès diffèrent entre l'environnement de développement local (Podman) et les données injectées par les "Seeders" (liens externes ou chemins relatifs manquants).

2. Niveau d'avancement avant l'exercice

Avant l'utilisation de l'IA, le contrôleur `Cart.php` était fonctionnel dans sa logique métier de base (ajout/suppression). Cependant :

- L'interface utilisateur (Vue) était cassée : les images ne chargeaient pas.
- La gestion des erreurs d'affichage était inexistante (icônes d'image brisée visibles).
- Nous rencontrions des difficultés à déboguer les chemins de fichiers (`file_exists` PHP échouant à cause de la structure Podman).

Notre niveau d'avancement n'était donc pas livrable en raison de défauts majeurs d'interface et de robustesse.

3. Présentation de l'outil IA utilisé

Nous avons utilisé **Gemini (modèle Google)** via l'interface conversationnelle de Copilot et celle de Google.

Pourquoi ce choix ?

- **Capacité contextuelle** : Gemini permet l'upload de fichiers multiples (code source PHP, logs d'erreurs, captures d'écran), ce qui est essentiel pour qu'il comprenne l'architecture MVC de CodeIgniter 4.
- **Analyse multimodale** : La capacité de l'IA à analyser une capture d'écran du navigateur (montrant l'erreur 404) a permis un diagnostic plus rapide que la simple lecture du code.

4. Explication détaillée de l'utilisation de l'IA

Nous avons utilisé l'IA selon une approche itérative pour résoudre le problème d'affichage et optimiser le code.

Étape 1 : Contextualisation et Diagnostic

Nous avons fourni à l'IA :

1. Le fichier `app/Controllers/Cart.php` pour qu'elle comprenne la logique backend (gestion Session / DB).
2. Le fichier de vue `cart.php`
3. Une capture d'écran montrant les images manquantes.

Prompt utilisé :

"Voici mon contrôleur Cart.php et ma vue. J'ai des erreurs 404 sur les images dans le navigateur. Les images sont stockées dans public/uploads/products/ mais le code actuel ne les trouve pas. Aide-moi à corriger la logique d'affichage en m'expliquant les modifications que tu aurais apporté à mon code actuel"

Étape 2 : Correction et Robustesse (Refactoring)

L'IA a d'abord suggéré une vérification PHP (`file_exists`), qui a échoué. Nous avons alors relancé l'IA en lui expliquant que le chemin absolu serveur posait problème.

Prompt de correction :

"La vérification PHP ne marche pas bien avec Podman. Propose une solution plus robuste qui gère aussi les URLs externes (http) et les images manquantes."

L'IA a générée une solution avec du PHP et du JS en intégrant :

- Une détection des URLs externes (via `strpos`).
- Une construction dynamique du chemin (`base_url`).
- L'ajout d'un attribut `onerror` dans la balise HTML `` pour charger une image de remplacement (placeholder) automatiquement si l'image principale échoue.

Étape 3 : Documentation et Déploiement

Nous avons également demandé à l'IA de générer la documentation (`README.md`) pour faciliter le lancement du projet et la génération de la documentation technique Sphinx, assurant ainsi la compréhension du code produit pour nos professeurs et les futur développeurs du projet .

5. Retour d'expérience argumenté

Description et analyse des résultats

L'intervention de l'IA a permis de passer d'une vue non fonctionnelle à une interface robuste en moins d'une heure. Le code final du panier gère désormais parfaitement :

- Les images locales (avec ID produit dans le chemin).
- Les images distantes (Unsplash/CDN).
- Les images absentes (Placeholder automatique).

Le contrôleur `Cart.php` n'a pas eu besoin de modifications lourdes, l'intelligence a été déportée vers la Vue, respectant le principe de séparation des rôles.

Critique de l'approche

- **Positif** : Gain de temps considérable sur le débogage des chemins de fichiers, souvent fastidieux. L'IA a proposé une solution (le `onerror` JS) à laquelle nous n'aurions jamais pensé, privilégiant une solution PHP pure.
- **Négatif** : La première réponse de l'IA était techniquement juste mais inadaptée à notre environnement Podman spécifique. Il a fallu faire preuve d'esprit critique pour ne pas copier-coller aveuglément et demander une alternative.

Retour en utilisant la démarche CRISTAL

- **Critique (C)** : L'IAg a produit une première solution technique (basée sur `file_exists` en PHP) qui était syntaxiquement correcte mais pas adaptée à notre contexte Podman (problème de chemin absolu vs chemin URL). L'IA a généralisé une solution certainement souvent utilisée sans tenir compte de la configuration de notre serveur. Nous avons dû critiquer cette première approche pour obtenir la solution robuste en JavaScript (`onerror`).
- **Responsable (R)** : Nous avons veillé à utiliser l'outil de manière sécuritaire en ne donnant aucune donnée personnelle. Seule la structure du code (`Cart.php`) et des noms de fichiers génériques ont été partagés. La requête était neutre et orientée sur la technique, visant la résolution d'un bug sans biais possible sur le résultat.
- **Intègre (I)** : L'utilisation de l'IA a respecté les règles académiques : elle a servi d'assistant de débogage et non de producteur final. Nous n'avons pas demandé à l'IA de "faire le projet", mais de "corriger un dysfonctionnement précis". Nous avons appris une nouvelle technique (l'utilisation de l'attribut `onerror` pour le fallback d'image) que nous ne connaissions pas, validant ainsi la dimension d'apprentissage de l'échange.
- **Sobre (S)** : L'IA était réellement utile pour ce cas précis de débogage qui nous bloquait depuis plusieurs heures. Nous avons limité les interactions en fournissant un contexte complet en une seule fois (code + erreur + capture d'écran) plutôt que de multiplier les petites requêtes ce qui consomme plus d'énergie. L'outil choisi (Gemini) était adapté grâce à sa capacité multimodale (analyse d'image).
- **Transparent (T)** : L'usage de l'IA est clairement indiqué dans ce rapport. Dans le code, bien que la logique vienne d'une suggestion de l'IA, nous l'avons intégrée et commentée nous-mêmes.
- **Autonome (A)** : Nous avons gardé le contrôle total sur le contenu final. Nous n'avons pas copié-collé aveuglément la première réponse (qui ne fonctionnait pas). Nous avons considéré l'IA comme un outil de documentation avancé et non comme une personne, en gardant un ton directif et technique dans nos prompts pour orienter la résolution vers le résultat souhaité.
- **Libre et créatif (L)** : Nous avons pris le temps de réfléchir à la proposition de l'IA. Nous avons rejeté la partie de sa réponse qui suggérait de modifier la configuration serveur (trop risqué) pour privilégier et adapter la solution côté client (Frontend). Nous avons modifié les variables du code généré pour qu'elles correspondent parfaitement à nos entités (`CartItem / Product`), montrant que nous nous sommes correctement approprié le code.

Recommandations pour une meilleure exploitation

1. **Fournir le contexte complet dès le début :** L'IA est beaucoup plus performante si on lui donne l'arborescence des fichiers (`tree`) et les fichiers de configuration (`.env`, `Paths.php`) en plus du code problématique que l'on veut modifier.
2. **Challenger la solution :** Ne pas accepter la première proposition de code. Demander "Y a-t-il une méthode plus sécurisée ?" ou "Est-ce compatible avec Podman ?" devrait nous permettre d'obtenir un code d'une meilleure qualité.
3. **Utiliser l'IA pour la documentation :** L'IA excelle à expliquer le code qu'elle a aidé à corriger. Lui demander de générer les commentaires PHPDoc ou le README permet de gagner du temps car ce projet était à faire sur une courte période de temps. Cela nous a permis d'être plus efficaces en développement.