SAE 301 - Initiation à l’IAg

Rapport

1. Tâche initiale :

Nous avons utilisé l’IA générative dans le cadre des filtres de notre catalogue. Nous souhaitions que les clients de notre site puissent trier les éléments du catalogue selon plusieurs critères, tels que la taille, la marque ou le prix des produits proposés.

Initialement, nous avions une fonction permettant de récupérer les produits depuis la base de donnée selon les filtres et, visuellement, nous pouvions cliquer sur des boutons filtres. Notre problème était de pouvoir lier ces deux éléments sans avoir à valider et rafraîchir la page à chaque nouveau filtre ajouté.

2. Utilisation de l’IA générative :

Pour pallier ce problème, nous nous sommes tournés vers Claude sonnet 4.5. Développé par Anthropic, cet agent possède de forts atouts en programmation et sera capable de nous expliquer les différents choix de conception. Nous lui avons demandé de nous expliquer comment coder ce lien manquant entre nos deux éléments en lui donnant le code du contrôleur en paramètre. Nous avons aussi utilisé Claude pour déboguer les problèmes qui ont pu survenir après avoir intégré son code au nôtre. L’IA a été également très utile pour générer des pages de test afin de mieux comprendre, de préciser et de comprendre la raison des erreurs. Pour finir elle a plus généralement été utilisée pour aider à coder quelques parties de code assez complexe et nouvelle dans l’apprentissage

Exemple d’une requête :

> J'aimerais que tu m'aides à comprendre comment peut on permettre d'appliquer un filtre sur une page de front end avec des boutons spécifiques qui affiche différent filtre possible (ex : taille (affiche les tailles dispo dans la base de donnée), pour t'aider de ça, j'aimerais que tu reprennes le controller 'ProduitController" qui va chercher les infos dans la base de donnée.

Extrait de la réponse de Claude :

Comprendre le système de filtrage dynamique

Je vais t'expliquer comment fonctionne le filtrage dans ton `ProduitController` et comment il interagit avec le frontend.

🔄 Architecture du système

1. Récupération des filtres disponibles (`getAvailableFilters()`)

Cette méthode interroge la base de données pour obtenir **les valeurs uniques** de chaque filtre :

```php
private function getAvailableFilters(): array
{
    // Récupère toutes les MARQUES distinctes des produits disponibles
    $marques = $this->produitModel
        ->distinct()
        ->select('marque')
        ->where('est_dispo', 1)  // Seulement les produits disponibles
        ->orderBy('marque', 'ASC')
        ->findAll();

    // Pareil pour couleurs, tailles, catégories...

    return [
        'marques' => ['NIKE', 'ADIDAS', 'PUMA', ...],
        'couleurs' => ['NOIR', 'BLANC', 'ROUGE', ...],
        'tailles' => ['38', '39', '40', ...],
        'categories' => ['SNEAKERS', 'BOOTS', ...],
        'prix_min' => 50,
        'prix_max' => 300
    ];
}
```

Conclusion de Claude dans la même réponse :

💡 Points clés

1. **Les filtres dynamiques :** Tu interroges la BD pour savoir quelles valeurs existent réellement
2. **Groupement SQL :** `groupStart()/groupEnd()` créent des parenthèses pour les OR
3. **AJAX :** Évite de recharger toute la page, plus fluide
4. **Toggle des filtres :** L'utilisateur peut activer/désactiver plusieurs filtres
5. **Stock :** Les tailles vérifient automatiquement le stock disponible

Tu veux que je t'aide à implémenter une partie spécifique (par exemple, le CSS des boutons ou une fonctionnalité particulière) ?

L’IA a su répondre correctement en proposant plusieurs étapes de code, comme vu dans sa conclusion. Elle a d’abord proposé comment récupérer les valeurs dans la base de données, même si un contrôleur existait déjà pour faire cela. Ensuite, elle a montré comment les appliquer sur la page via du JavaScript. Ce code était, certes, une amélioration du nôtre, mais n’était pas demandé. Pour finir, elle a renvoyé le code entier et un fichier en format Markdown récapitulant l’ensemble de ses propositions. Cela est important pour comprendre sa logique.

Même si elle a ajouté des éléments que nous n’avions pas demandés, comme du code HTML pour afficher les filtres, ils se sont avérés utiles pour déboguer d’éventuels problèmes et voir si notre code déjà existant concordait avec le sien.

3. Retour sur notre utilisation :

Description et analyse des résultats :

Les réponses générées par l’IA étaient très précises, mais parfois, elles allaient un peu trop loin dans la demande par rapport à ce que nous attendions. Par exemple, elles changeaient le design du site ou ajoutaient de nouvelles fonctionnalités non demandées. L’IA se trompait parfois lorsqu’il s’agissait de noms de fichiers à importer ou de variable, mais les noms étaient assez intuitifs pour que nous puissions facilement les corriger. Dans l’ensemble elle a su répondre correctement à nos attentes.

Critique de notre approche :

Notre approche n’était pas complètement adaptée dans le cadre éducatif. En effet, il aurait été plus adapté de coder la fonctionnalité par nous même afin de prouver nos compétences techniques. Si la tâche était trop compliquée, nous aurions peut-être dû revoir nos exigences afin de coder des choses que nous pouvons coder par nous même.

Au regard de la démarche CRISTAL :

Critique : Nous savons que les IA génératives souffrent souvent d’un problème d’hallucination. Par conséquent, quand nous ne comprenions pas une fonction générée dans notre code, nous avons privilégié la recherche dans la documentation du langage de programmation utilisé ou sur un moteur de recherche plutôt que via l’IA générative.

Responsable : Ce projet étant fait au sein d'une entreprise, nous connaissons les risques d'utiliser une IA générative quant aux informations sensibles de l'entreprise ou des utilisateurs. Nous avons donc fait attention à injecter un minimum de code et à ne pas injecter d'informations confidentielles. Par exemple, quand il s'agissait de filtrer par marque, nous avons donné comme exemple « marque1 », « marque2 » au lieu du vrai nom de la marque.

Intègre : L'usage de l'IA générative était demandée dans le cadre de la ressource R310, nous avons donc considéré qu'il était juste pour nous de l'utiliser. Cependant, nous comprenons que l'objectif de la SAE est de juger nos capacités à programmer par nous même. Nous avons donc limité notre usage à cette ressource et nous avons mentionné quelle partie du code a été faite par assistance avec une IA générative. Enfin, nous possédons les compétences nécessaires pour juger de la qualité du code généré.

Sobre : Nos requêtes ont été révisées avant envoi afin d'être sûr qu'elles correspondent à nos besoins. L'IA générative utilisée a été choisie car elle possède de bonnes performances en programmation et explique de façon très claire chaque étape de sa logique, permettant d'éviter de devoir faire plusieurs requêtes sur un même sujet pour mieux le comprendre.

Transparent : Nous avons écrit en commentaire de notre code quelles parties avaient été générées par IA générative et nous l'avons précisé dans ce rapport.

Autonome : Nos requêtes utilisaient parfois un langage un peu trop personnel, mais nous restons conscients que l'IA générative n'est pas une personne qui peut penser. Aussi, les propositions de l'IA ont été revues et nous nous sommes laissés le choix d'accepter ou non le code généré selon nos besoins.

Libre : La tâche de créer des filtres nécessite plusieurs choses, comme un contrôleur permettant de chercher des produits via des filtres ou un affichage graphique des filtres en question. Pour cela, nous les avons codés sans IA générative. Cela nous a laissé un moment important pour réfléchir par nous même.

Des recommandations pour une meilleure utilisation :

Si cet exercice était à refaire, nous aurions essayer de nous concentrer davantage sur coder par nous même, puis nous faire aider par l'IA générative si nous étions vraiment bloqués. Cela nous aurait permis d'être dans une démarche plus éducative et plus intègre compte tenu des attentes de la SAE. Il serait également bon de revoir comment écrire une requête efficace afin de minimiser au maximum la quantité effectuée.