Création d'un web SIG simple par IA générative

AXES Conseil

Pôle Systèmes d'information de Ségic

axes.fr

axesconseil@axes.fr

En ce quart de siècle où la vie se transforme au rythme accéléré d'une IA générative triomphante, les structures de la société se brisent chaque jour davantage derrière la façade de l'Internet généralisé.

La liste des craintes augmente dans des proportions d'autant plus inquiétantes que cette offre nouvelle s'appuie sur des évolutions techniques innovantes et fait échec à des modes de fonctionnement SI anciens, souvent assis sur des certitudes et des méthodes qui n'ont guère évolué depuis plus de 20 ans. Un chiffre est éloquent : selon une projection du cabinet Roland Berger, jusqu'à 800 000 emplois en France pourraient être détruits d'ici la fin de la décennie à cause de l'IA générative.

Dans ce contexte, quel est le sort des développeurs, géomaticiens et développements SIG ?

Pour tenter de répondre à cette question, une expérience a été menée : générer un web SIG simple, déployable sur Git ou équivalent, en prenant le parti pris d'une simulation d'absence de compétences techniques, afin d'analyser les options prises par l'IA générative et les méthodes déployées.

Méthodologie

Des solutions simples ont été préalablement retenues : choix de ChatGPT version courante et Claude Sonnet v4.6. La seule option technique préemptée a été le choix de l'environnement d'exécution serveur Node.js.

Article rédigé sans IA générative

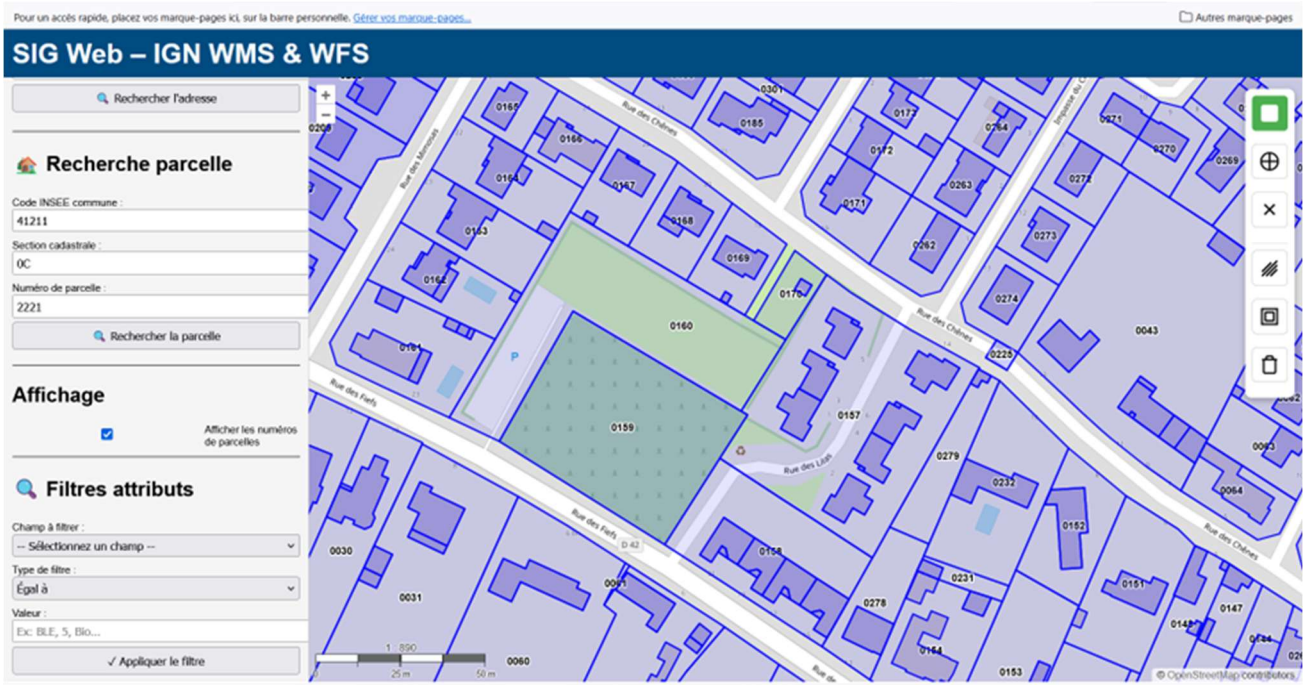
Un prompt précis a été rédigé, structuré en trois blocs :

1. La description globale du web SIG : afficher, interroger et analyser des services géographiques **WMS** et **WFS**, principalement issus de l'IGN, mais compatibles avec n'importe quel service OGC.
2. La description de l'interface souhaitée : panneau latéral listant les sources WMS et WFS, boutons pour charger, afficher et filtrer les couches, barre d'outils flottante (zoom rectangle, recentrage France, effacer sélection, mesures distance/surface, effacer mesures), une carte affichant les couches et les résultats de recherche, export GeoJSON et Shapefile.
3. La description des fonctionnalités principales : chargement dynamique des capacités WMS / WFS et affichage, sélection d'entités WFS au clic, affichage détaillé des attributs dans une fenêtre popup, Recherche d'adresse, recherche parcellaire (INSEE + section + numéro).

Résultats

Le résultat obtenu, coté utilisateur, répond à la demande exprimée et les fonctions sont globalement opérationnelles :

- IHM conforme aux attentes décrites
- Affichage des services de données
- Recherche et sélection d'objets via critère
- Export de données



Web SIG généré

Commentaires

Une analyse qualitative du code généré par Claude a été demandée à ChatGPT : Les points positifs ont été identifiés, des failles techniques ont été détectées (faute de demande précise au niveau du prompt) : beaucoup de variables globales, très peu de mutualisation du code, beaucoup de code répétitif (sélection, création de couches, fetch, etc.), aucune gestion d'erreurs détaillée WMS/WFS, pas de limitation de taille pour les données WFS… Une faille majeure de sécurité a été détectée : absence de filtrage de sécurité sur le proxy Node.js qui accepte toute URL.

Autrement dit, le code généré contient les erreurs qu'un développeur peu expérimenté aurait pu faire, faute d'encadrement par un responsable aguerri…

Il est donc observé, sur ce projet, que les deux IA générative exploitées :

- Génèrent des bugs : il appartient au développeur de prompter précisément les modifications à opérer, soit au niveau technique, soit au niveau IHM et fonctionnel
- Laissent traîner du code mort suite à des modifications apportées sur l'application (eg. pas de clear() sur les sources, pas de suppression des objets null…)
- Créent un code parfois peu orthodoxe
- Sont très fortes pour expliquer pourquoi ça ne fonctionne pas, la correction d'un bug n'étant pas systématique
- Ont une proactivité variable (eg. pas de proposition d'installer un serveur web tant que la remarque sur le non fonctionnement en son absence n'est pas indiquée par le développeur)
- Testent le code développé, ou pas, sauf prompting dédié

De développeur à pilote de développement ?

A court terme, les développeurs SIG restent « compétitifs ».

A moyen terme, (18 mois ? 2 ans ?), les développeurs devront probablement basculer partiellement vers du «pilotage» de développement :

- Définition et application d'une méthodologie de prompting (eg. Role / Structured prompting, CoT, etc.)
- Vérification (fonctionnelle et sécurité) avancée de l'application générée
- Vérification technique et qualitative du code
- Demande de génération partielle (eg. fonction particulière à intégrer dans une application existante)

- Versionning avancé (forte régression fonctionnelle quasi systématique constatée du code généré de version en version)
- Construction de modèles de prompt, à des fins de productivité.

Les solutions commerciales SIG basées sur un générateur d'application permettront, un temps encore, de limiter les problèmes constatés lors de la présente expérience. Les principaux éditeurs cherchent d'ores et déjà à maximiser leur expérience, en exploitant les technologies IA. Des évolutions significatives de l'offre, basées sur l'IA sont, à n'en pas douter, à attendre dans les mois et années à venir (approche agentique, VLM,…)

Et, dernier point, une réelle interrogation se pose sur le maintien et le devenir des compétences techniques des développeurs et la dépendance croissante vis-à-vis d'outil d'IA. Le passage d'un rôle de créateur de code à celui de contrôleur de code pourrait également sembler être moins stimulant ?

Article rédigé sans IA générative