Phase 2 – Synthèse Équipe : Regard Critique sur l'utilisation de l'IA

Risques de dépendance et perte de savoir-faire

Utiliser l'intelligence artificielle dans le développement peut présenter des risques, on peut très vite en devenir dépendant et perdre les notions fondamentales du développement. Alors que la première phase du projet exigeait une compréhension profonde de l'algorithme de Voronoï et du calcul des distances euclidiennes, le passage à l'IA a permis d'obtenir du code fonctionnel et organisé en quelques secondes sans avoir à maîtriser tout la logique derrière le diagramme de voronoï, ou même avoir besoin de savoir coder. Cette vitesse cache un vrai danger, si le developpeur ne comprend pas le code genere, il devienet incapable de faire quoi que ce soit lorsqu'un bug arrivera aussi minime soit-il. Il ne parviendra pas non plus à maintenir et faire évoluer son code.

Problèmes de fiabilité et erreurs techniques

La fiabilité reste également un point de vigilance car chaque IA qu'on a testée a produit des bugs. ChatGPT présente des faiblesses sur le parsing de fichiers, tandis que Gemini et Mistral ont généré des conflits dans la phase de test avec Vitest et Playwright, ainsi que des erreurs d'import de modules. De plus, Mistral a fourni un diagramme mal centré et une encapsulation du code qui rendait les fonctions impossibles à tester directement. Claude, bien que le plus performant sur la structure logique, a parfois proposé des solutions dépendant de bibliothèques externes non installées, créant des erreurs d'exécution immédiates. Ces erreurs nous montrent que l'IA produit vite du code qui semble correct visuellement mais qui échoue souvent dans des conditions réelles d'utilisation ou dans des environnements techniques spécifiques. Le code peut être bon mais il ne va pas plus loin, il ne se pose aucune question au sujet de sa longévité.

Sur-ingénierie et complexité inutile du code

On a aussi remarqué que l’IA a tendance à compliquer les choses pour rien. Elle proposent souvent des codes très lourds et difficiles à comprendre, avec une séparation stricte des fichiers et utiliser des concepts complexes (comme la POO) même quand ce n’est pas nécessaire pour un petit projet. Par rapport à notre code de la première phase qui était beaucoup plus simple, celui de l’ia est devenu plus difficile à suivre. De plus, les

tests que l'IA génère automatiquement ne sont pas toujours fiables, il ne vérifient pas souvent les vrais problèmes , au final on perd du temps car quelqu'un doit repasser derrière pour tout vérifier à la main.

Limites de la génération automatique et importance du pilotage

Globalement, aucune des IA testées, qu'il s'agisse de ChatGPT, Claude, Gemini ou Mistral, n'a produit un résultat parfait du premier coup. Chaque génération a besoin entre quatre et sept itérations de prompts pour avoir le résultat qu'on cherchait. Le gain de temps pour le prototypage rapide et la mise en place d'environnements de test était clairement visible si on devait comparer à la phase 1, mais ce temps gagné est en partie réinvesti dans le débogage et la validation. La qualité finale dépend entièrement de la précision du prompt et de la capacité du développeur à piloter l'outil. L'IA reproduit des schémas connus sans toujours les adapter intelligemment au contexte, comme le montre l'exportation de fichiers inutilement lourds.

Conclusion

Pour conclure, l'IA générative s'impose comme un coéquipiers puissant capable de coder plus vite et d'appliquer de bonnes pratiques de structuration quand on le lui demande, mais elle ne remplace pas notre propre logique. Pour un usage responsable, il est indispensable de comprendre chaque ligne de code avant de l'intégrer, de rédiger des instructions précises et de valider les sorties par des tests indépendants. On ne doit pas laisser l'IA tout gérer mais nous devons le piloter pour atteindre notre but. C'est un outil fabuleux pour essayer et itérer très vite, et il est encore plus efficace quand on sait comment s'en servir. L'IA est un excellent serviteur, il n'est pas fait pour jouer le rôle du maître.