Diana Margarian

Synthèse Veille Technologique

Comment l'IA intégrée dans VS Code révolutionne-t-elle le développement logiciel tout en posant des défis pour préserver l'autonomie et l'expertise des développeurs ?

Introduction

Premièrement, l'intelligence artificielle (IA) change notre façon de travailler en tant que développeurs. Visual Studio Code propose maintenant des outils d'IA comme GitHub Copilot qui aident à coder plus vite en suggérant du code et en automatisant les tâches répétitives. Mais cela soulève une question importante : devenons-nous trop dépendants de ces outils ? Apprenons-nous vraiment à programmer ou nous contentons-nous de valider ce que l'IA nous propose ?

Cette synthèse explore l'impact des outils d'IA dans VS Code. Elle analyse comment ils transforment le développement d'applications mais aussi les problèmes qu'ils peuvent poser pour l'apprentissage et l'autonomie des développeurs. Elle examine les fonctionnalités disponibles, leurs avantages et inconvénients, et ce que cela implique pour l'avenir du métier de développeur.

I. Outils d'IA disponibles dans VS Code

Les outils d'intelligence artificielle intégrés à Visual Studio Code représentent une nouvelle génération d'assistants de développement qui vont bien au-delà de la simple complétion de code. Ces outils analysent le contexte du projet, apprennent des patterns de code existants et proposent des solutions complètes à des problèmes de programmation. À la différence des anciens systèmes d'autocomplétion qui se basaient sur des règles prédéfinies, ces nouveaux assistants utilisent des modèles d'IA entraînés sur des millions de dépôts de code open-source pour comprendre les intentions du développeur et proposer des solutions pertinentes.

A. IntelliCode

IntelliCode est l'assistant de Microsoft intégré à Visual Studio Code. Contrairement à d'autres solutions d'IA plus récentes, IntelliCode a été conçu avec une approche différente. Il utilise l'apprentissage automatique pour analyser des millions de lignes de code provenant de projets populaires sur GitHub afin d'identifier les patterns de codage les plus courants. Cette technologie permet à IntelliCode de comprendre le contexte de votre code et de proposer des suggestions adaptées à votre situation spécifique, plutôt que de simplement suggérer des compléments alphabétiques comme le faisaient les anciens systèmes d'autocomplétion.

→ Source :

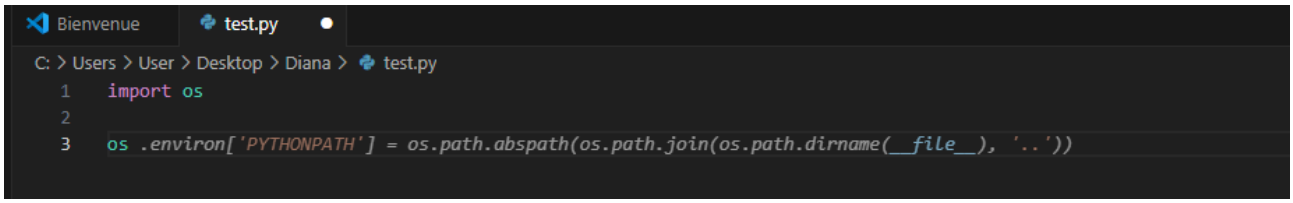
(https://marketplace.visualstudio.com/items?itemName=VisualStudioExptTeam.vscodeintellicode)

(https://learn.microsoft.com/fr-fr/visualstudio/ide/ai-assisted-development-visual-studio?view=vs-2022)

Un avantage majeur d'IntelliCode est qu'il est inclus gratuitement dans VS Code et ne nécessite pas d'abonnement supplémentaire, contrairement à d'autres solutions plus avancées. Il s'intègre directement à l'interface familière de l'éditeur sans nécessiter de configuration complexe.

Une fois installé, IntelliCode fonctionne avec l'autocomplétion standard. Quand vous tapez du code, des suggestions marquées d'une étoile apparaissent, indiquant que ces suggestions sont basées sur l'analyse contextuelle et l'apprentissage du modèle.

Voici un exemple :



Ses avantages :

- Gratuit et parfaitement intégré à VS Code
- Suggestions basées sur le contexte du projet
- Consomme peu de ressources

Inconvénients :

- Fonctionnalités plus limitées que Copilot
- Génère seulement des compléments de fonctions, pas des fonctions entières
- Moins efficace pour les langages moins courants
- Ne remplace pas la compréhension des frameworks

B. GitHub Copilot

En parallèle, GitHub Copilot est actuellement l'outil d'IA le plus puissant pour VS Code. Contrairement à IntelliCode qui se concentre principalement sur l'autocomplétion contextuelle, Copilot peut générer des blocs entiers de code, des fonctions complètes, et même des fichiers entiers à partir de commentaires en langage naturel ou de contexte existant. Il est développé par GitHub et OpenAI, il est basé sur une version spécialisée du modèle GPT pour le code.

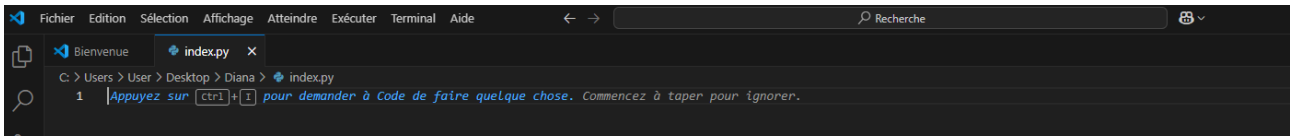
Le fonctionnement de Copilot va bien au-delà de la simple suggestion : il analyse en permanence votre contexte de travail comprend les variables que vous utilisez, la structure de votre code, et même le style de programmation que vous préférez. Au fil du temps il s'adapte à vos habitudes de codage, rendant ses suggestions de plus en plus pertinentes. Cette capacité d'adaptation contextuelle en fait un outil particulièrement puissant pour les développeurs travaillant sur des projets complexes.

Copilot offre une version gratuite limitée pour les étudiants et certains projets open source mais un abonnement payant est généralement nécessaire pour avoir accès sans limite.
Pour utiliser Copilot, vous devez vous connecter à un compte GitHub ce qui permet au service de

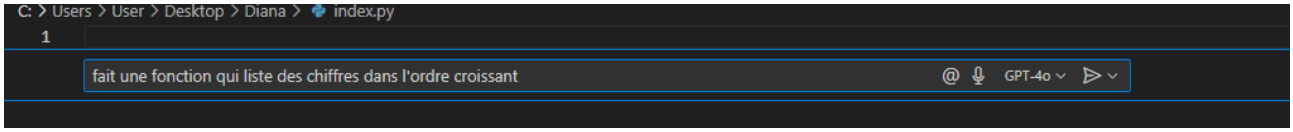
synchroniser vos préférences et d'améliorer continuellement ses suggestions basées sur votre utilisation.

Voici des exemples d’étapes d’utilisation de l’IA en question :

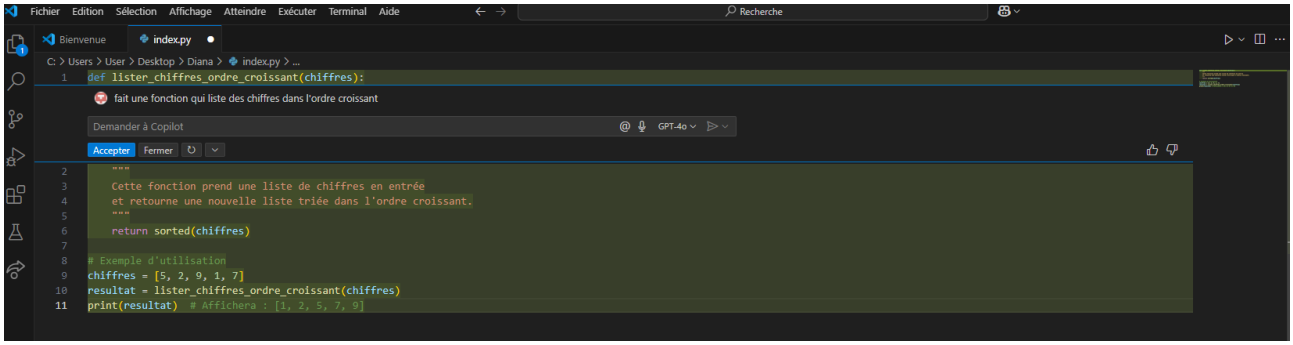
- Avant de commencer à coder l’IA me propose de demander à Coder si je le souhaite :



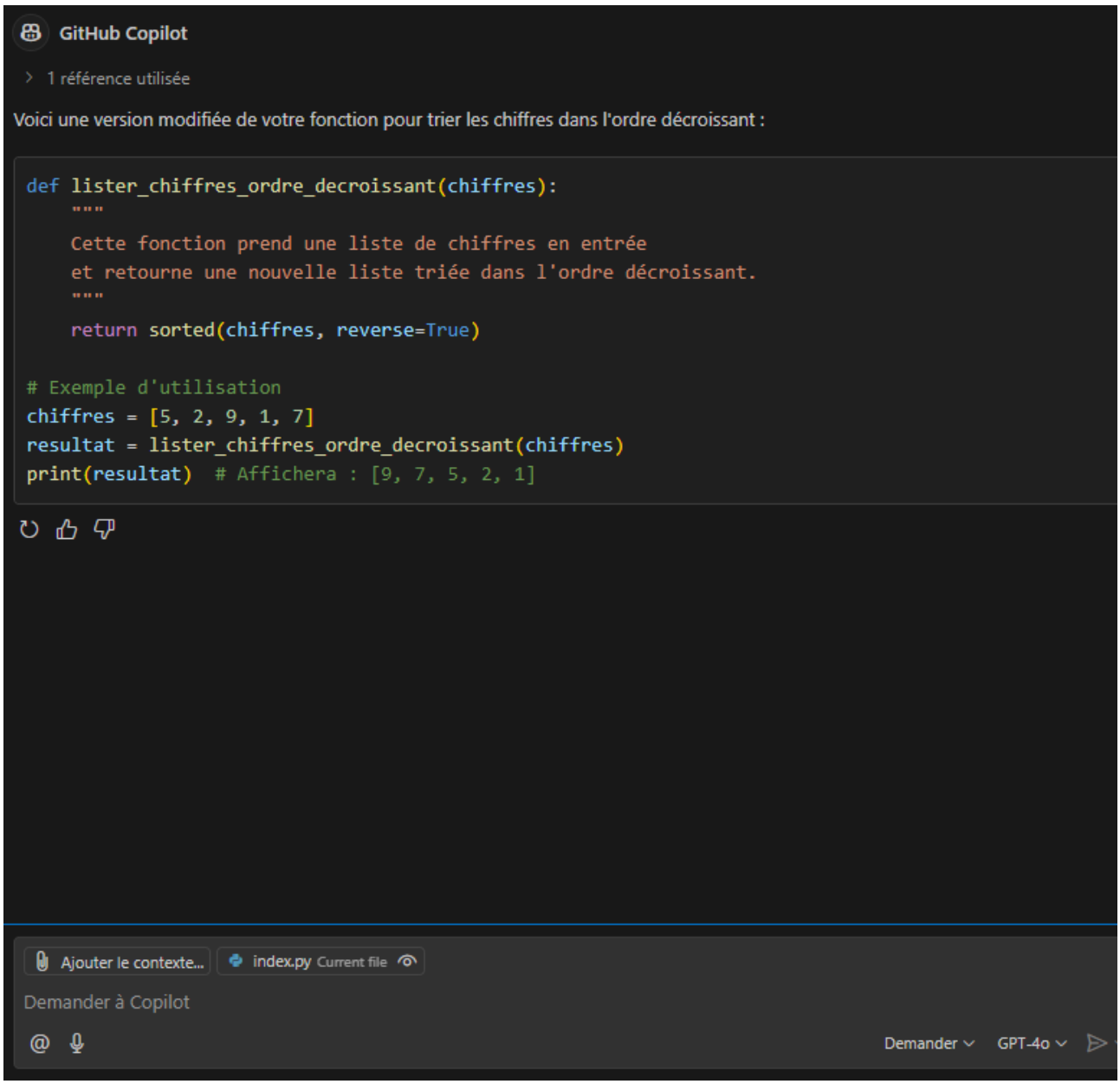
- Je demande à l’IA de faire une fonction qui liste des chiffres dans l’ordre croissant :



- Voici ce que l’IA propose automatiquement:



- Je demande maintenant de modifier la fonction en tri par ordre décroissant



Cependant, pour avoir un accès complet, il faut un abonnement payant.

Voici la page d'abonnement de GitHub Copilot montrant les tarifs :

Comparaison des plans Copilot

Les tableaux ci-dessous présentent les fonctionnalités disponibles dans chaque plan Copilot.

	Copilote gratuit	Copilot Pro	Copilot Pro+	Copilote Affaires	Copilote Entreprise
Tarification	Non applicable	10 USD par mois ou 100 USD par an (gratuit pour certains utilisateurs)	39 USD par mois ou 390 USD par an	19 USD par siège accordé par mois	39 USD par siège accordé par mois
Demandes premium	50 par mois	300 par mois	1500 par mois	300 par utilisateur et par mois	1000 par utilisateur et par mois
Achetez des demandes premium supplémentaires à 0,04 $/demande	×	✓	✓	✓	✓

Ses avantages :

- Génère des fonctions complètes à partir de simples commentaires
- S'adapte au style de code du projet
- Anticipe les intentions du développeur
- Supporte de nombreux langages de programmation

Ses inconvénients :

- Version complète payante par abonnement
- Risque de créer une dépendance aux suggestions automatisées
- Questions de propriété intellectuelle sur le code généré

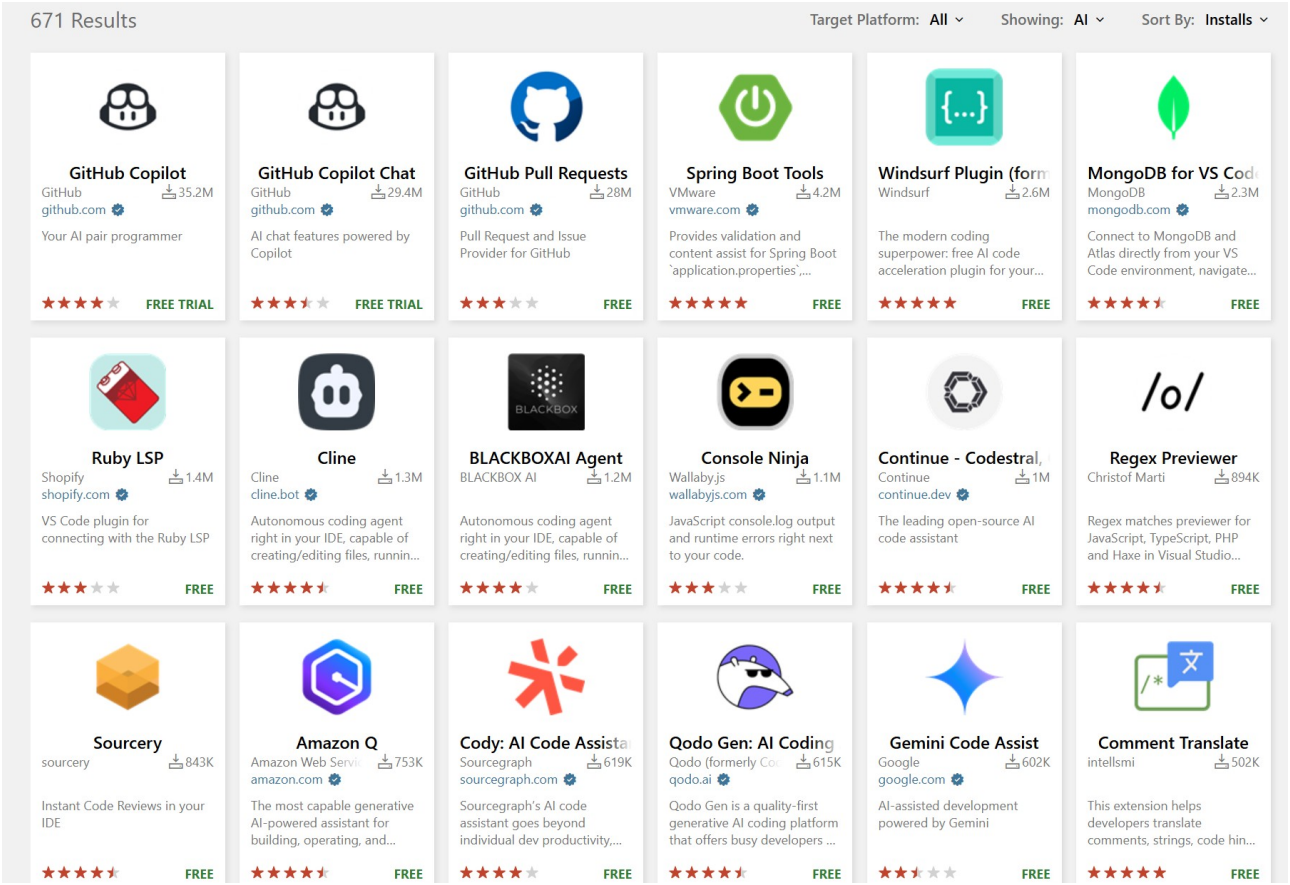
Sources :
(https://code.visualstudio.com/docs/copilot/overview)
(https://github.com/features/copilot)
(https://apprendre-la-programmation.net/github-copilot/#mon-utilisation-et-avis-sur-github-copilot)
(https://learn.microsoft.com/fr-fr/visualstudio/ide/ai-assisted-development-visual-studio?view=vs-2022)
(https://www.netguru.com/blog/github-copilot#:~:text=Dependency%20Risk%3A%20Relying%20heavily%20on,of%20the%20suggestions%20can%20vary.)

C. Autres extensions d'IA dans VS Code

Au-delà d'IntelliCode et de GitHub Copilot, VS Code propose de nombreuses autres extensions d'IA avec différents modèles de tarification et spécialisations. On trouve des solutions comme Amazon Q, un assistant de codage développé par AWS qui excelle particulièrement dans l'intégration avec les services Amazon. Cody AI se spécialise dans la compréhension approfondie du contexte de codebase et peut répondre à des questions sur le fonctionnement global d'un projet. Gemini Code Assist, développé par Google, propose une approche multimodale qui peut analyser à la fois du code et des images pour résoudre des problèmes complexes. Qobo Gem, une option plus récente, se concentre sur l'optimisation de code existant et la détection de bugs potentiels.

Ces extensions se différencient non seulement par leurs capacités techniques, mais aussi par leurs modèles de tarification (gratuit, freemium, ou entièrement payant) et par leur spécialisation dans certains langages ou frameworks spécifiques. Certaines sont plus adaptées aux débutants avec des explications détaillées, tandis que d'autres ciblent les développeurs expérimentés en offrant des optimisations avancées.

Exemple des différentes extensions d'IA disponibles dans la marketplace VS Code :



Ces extensions offrent des fonctionnalités variées comme la traduction de commentaires, l'assistance pour des frameworks spécifiques, ou l'optimisation du code existant. Le choix entre ces

différentes solutions dépend généralement des besoins spécifiques du développeur, de son budget, et des technologies avec lesquelles il travaille principalement.

Sources :

(Recherche Internet pour quelques IA présent dans le tableau)
(https://visualstudiomagazine.com/articles/2023/03/08/vs-code-ai-tools.aspx)

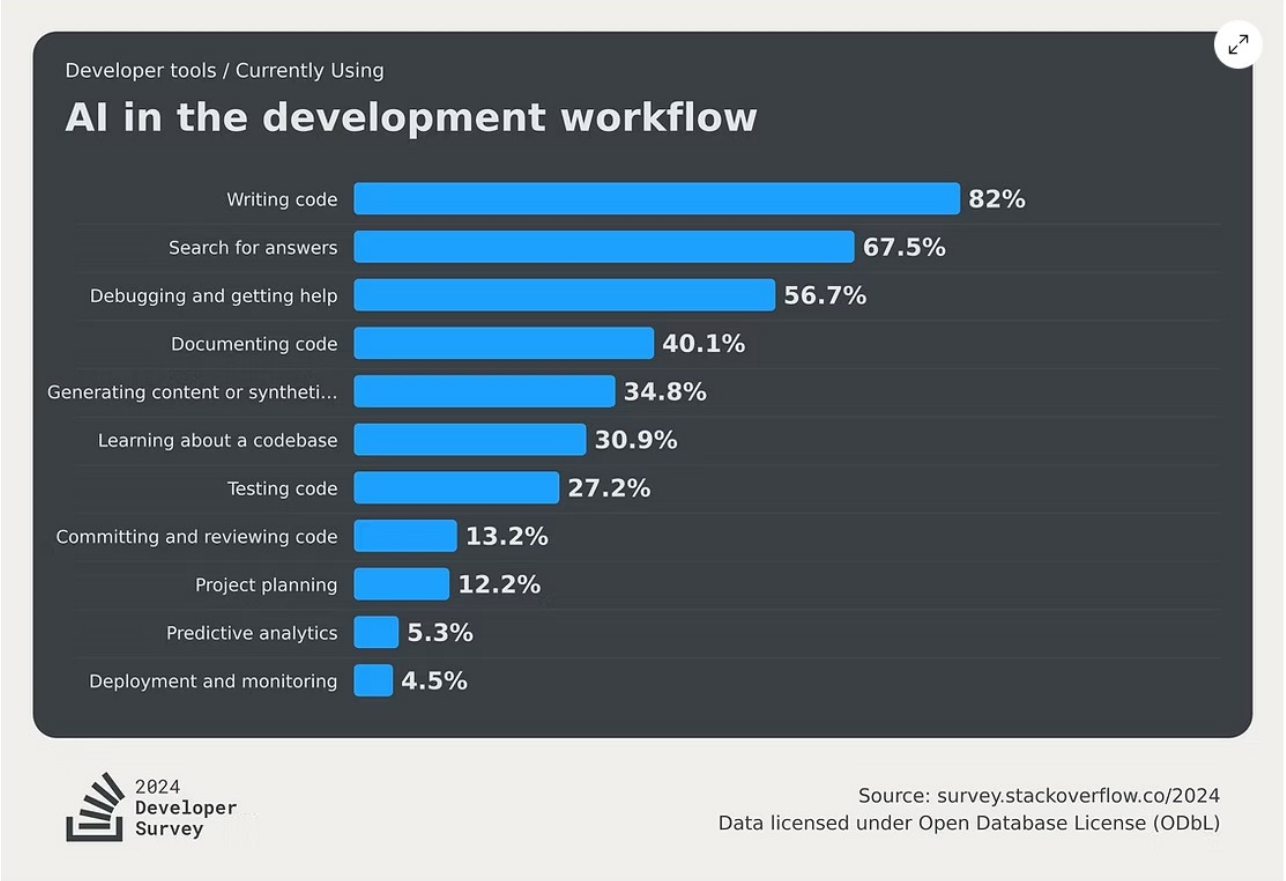
II. Impacts positifs de l'IA

L'introduction des outils d'intelligence artificielle dans les environnements de développement comme VS Code a profondément transformé la façon dont les programmeurs travaillent au quotidien. Ces technologies ne sont pas simplement des ajouts mineurs à nos outils existants, mais représentent un changement fondamental dans notre approche du développement logiciel. Leurs impacts positifs se manifestent à travers plusieurs dimensions clés, notamment la productivité, la qualité du code, et l'accessibilité des pratiques de développement avancées.

A. Accélération de la productivité

L'IA dans VS Code aide beaucoup les développeurs à travailler plus vite. Ce n'est pas juste quelques secondes gagnées. C'est un vrai changement dans la façon de coder. L'IA s'occupe des tâches répétitives et ennuyeuses. Ainsi, les développeurs peuvent se concentrer sur des choses plus importantes comme créer l'architecture du projet ou résoudre des problèmes difficiles.

Les débutants et les experts ne profitent pas de l'IA de la même façon. Pour les débutants, l'IA comble leurs manques de connaissances. Elle leur montre comment faire les choses correctement et leur évite de faire des erreurs classiques. Pour les développeurs expérimentés, l'IA fait les tâches basiques qu'ils connaissent déjà mais qui prennent du temps. Ils peuvent alors se concentrer sur la partie créative et stratégique de leur travail.



Plusieurs choses rendent les développeurs plus productifs avec l'IA :

L'IA écrit le code répétitif à notre place. Elle crée facilement des classes de données. Faire ces choses à la main prend du temps et on fait souvent des erreurs.

On passe moins de temps à chercher dans la documentation. Avant, les développeurs passaient beaucoup de temps à chercher comment utiliser des API ou des bibliothèques. L'IA connaît déjà beaucoup de code et peut nous montrer des exemples utiles tout de suite, sans qu'on ait besoin de quitter VS Code.

L'IA peut créer des tests pour notre code. Écrire des tests est important mais ennuyeux, donc beaucoup de développeurs ne le font pas assez. L'IA peut regarder notre code et faire des tests qui vérifient différentes situations, même celles auxquelles on n'aurait pas pensé.

L'IA nous aide à utiliser des bibliothèques qu'on ne connaît pas bien. Quand on travaille avec une nouvelle technologie, l'IA nous montre comment l'utiliser avec des exemples qui correspondent à notre projet.

Des études de GitHub montrent que les développeurs qui utilisent Copilot font certaines tâches jusqu'à 55% plus vite que ceux qui n'utilisent pas d'IA. Grâce à ça, on peut finir les projets à temps et passer plus de temps sur des choses importantes comme améliorer les performances, rendre l'application plus facile à utiliser, et bien planifier l'architecture.

Sources :

(https://www.legitsecurity.com/aspm-knowledge-base/ai-code-generation-benefits-and-risks)

(https://numericoach.fr/lia-dans-le-quotidien-dun-developpeur/)

B. Amélioration de la qualité du code

L'IA ne fait pas seulement coder plus vite, elle aide aussi à créer du meilleur code. Au début, certains pensaient que l'IA ferait du code rapide mais de mauvaise qualité. En réalité, c'est le contraire : les assistants IA suggèrent souvent les bonnes façons de faire les choses.

Pourquoi le code est meilleur avec l'IA ? D'abord parce que ces outils ont appris en étudiant des millions de projets de code, y compris beaucoup de projets de grande qualité faits par des équipes d'experts. L'IA a ainsi appris à reconnaître et à copier les bonnes façons de structurer le code, les modèles qui fonctionnent bien, et les meilleures solutions pour différents problèmes.

Voici un exemple où l'IA suggère une optimisation ou une amélioration de code :

ajouter_depense.php

```php
<?php
    require_once 'connexion.php';
    $requete = "INSERT INTO depense (NUMDEP, NUMNOTE, NUMTYPE, DATEDEP, MONTANTDEP, COMMENTAIRE)
    VALUES (:numdep, :numnote, :numtype, :datedep, :montantdep, :commentaire)";
    $statement = $connexion->prepare($requete);

    // Liaison des param�tres de la requ�te avec les valeurs du formulaire
    $statement->bindParam(':numdep', $_POST['numD']);
    $statement->bindParam(':numnote', $_POST['numN']);
    $statement->bindParam(':numtype', $_POST['numT']);
    $statement->bindParam(':datedep', $_POST['dateD']);
    $statement->bindParam(':montantdep', $_POST['montantD']);
    $statement->bindParam(':commentaire', $_POST['Commentaire']);

    // Exécution de la requ�te
    $statement->execute();

    echo "La dépense a été ajouté avec succés !";
    // Redirection vers la page d'accueil après 2 secondes
?>
```

Exemples concrets d'amélioration pour la qualité du code :

- **Ajout de validations et gestion des erreurs :** Quand on code vite, on oublie souvent de gérer les erreurs possibles. L'IA n'oublie pas. Elle ajoute automatiquement des vérifications pour les paramètres, gère les cas limites, et traite les exceptions. Cela rend le code plus solide.

- **Amélioration des performances :** L'IA peut repérer où le code pourrait fonctionner plus vite. Elle suggère des algorithmes plus efficaces, des meilleures structures de données, ou enlève les calculs inutiles. Même sans être expert en optimisation, on peut avoir un code qui fonctionne mieux.

- **Utilisation des modèles de conception** : L'IA reconnaît quand utiliser des modèles comme Singleton, Factory ou Observer. Elle peut les implémenter correctement dans notre code. Ces modèles sont testés et approuvés, ce qui rend le code plus facile à maintenir.

- **Création de commentaires et documentation :** On néglige souvent d'expliquer notre code, même si c'est important pour le futur. L'IA peut écrire des commentaires utiles, décrire ce que font les méthodes, et donner des exemples. Cela rend le code plus facile à comprendre et à modifier plus tard.

L'IA est aussi très bonne pour trouver les problèmes de sécurité. Par exemple, elle peut repérer les risques d'injection SQL dans notre code et nous suggérer d'utiliser des requêtes préparées plus sûres. C'est très important aujourd'hui où les failles de sécurité peuvent causer de gros problèmes.

Sources :

(https://spaininter.com/fr/news/81-intelligence-artificielle-integree-dans-vs-code)

III. Défis et risques pour l'autonomie des développeurs

Malgré les avantages indéniables apportés par l'IA dans les environnements de développement comme VS Code plusieurs défis et risques émergent concernant l'autonomie des développeurs et la préservation de leurs compétences fondamentales. Ces préoccupations ne sont pas simplement théoriques mais commencent à se manifester concrètement dans la pratique professionnelle et les environnements d'apprentissage.

A. Dépendance aux outils d'IA

La dépendance aux outils de l'IA est un des plus gros risques quand on l'utilise tout le temps. Cette dépendance ne vient pas d'un coup, mais petit à petit. Les développeurs s'habituent à la facilité et à la vitesse que l'IA leur offre. Au fil du temps, ce qui était juste une aide devient un besoin. Le développeur peut alors perdre sa capacité à résoudre les problèmes tout seul.

On peut voir cette dépendance à l'IA dans certains comportements, tant chez les professionnels que chez les étudiants. Ces comportements ne sont pas forcément mauvais s'ils arrivent de temps en temps. Mais s'ils deviennent une habitude c'est le signe que le développeur perd son autonomie technique.

Voici les signes qui montrent qu'un développeur devient trop dépendant de l'IA :

- **Accepter sans réfléchir :** Le développeur fait trop confiance à l'IA et accepte ses suggestions sans les vérifier. Il ne se demande pas si le code proposé est vraiment bon ou adapté. En faisant ça, il perd l'habitude de réfléchir par lui-même.

- **Ne plus savoir coder sans aide :** Certains développeurs, surtout ceux qui ont appris à coder avec l'IA, se sentent perdus quand ils doivent travailler sans elle. Ils sont moins productifs et moins à l'aise. C'est un signe qu'ils sont devenus dépendants.

- **Ne plus réfléchir aux problèmes difficiles :** L'IA peut résoudre des problèmes complexes toute seule. Si le développeur s'habitue à laisser l'IA trouver toutes les solutions, il perd peu à peu sa capacité à analyser et à résoudre des problèmes par lui-même.

- **Adapter les problèmes à l'IA :** Un signe inquiétant est quand le développeur change la façon de poser un problème pour que l'IA puisse mieux y répondre. Au lieu de chercher la meilleure solution, il cherche une solution que l'IA peut facilement générer.

Cette dépendance peut avoir plusieurs conséquences graves :

- **Perte d'autonomie :** Un bon développeur doit savoir créer des solutions par lui-même. Sans cette capacité, il sera moins polyvalent et aura du mal à s'adapter à de nouvelles situations.

- **Difficulté avec les problèmes inhabituels :** Quand un problème sort de l'ordinaire et que l'IA ne peut pas aider, le développeur dépendant se retrouve bloqué. Ces problèmes demandent une pensée originale que seul un humain peut avoir.

- **Utilisation de solutions inadaptées :** Les suggestions de l'IA ne sont pas toujours parfaites pour chaque situation. Si on les accepte sans réfléchir, on risque d'utiliser des solutions qui ne correspondent pas bien aux besoins spécifiques du projet.

- **Frustration sans l'IA :** Un développeur dépendant peut se sentir frustré et moins efficace quand il doit travailler sans IA, par exemple dans une entreprise qui ne l'autorise pas pour des raisons de sécurité.

B. Impact sur l'apprentissage et les compétences

Au-delà de la dépendance, une autre inquiétude concerne comment l'IA affecte l'apprentissage des compétences de base en programmation. C'est particulièrement important pour les étudiants comme ceux du BTS SIO qui sont encore en train d'apprendre les bases.

Normalement, on apprend à programmer étape par étape : on comprend d'abord les concepts de base, puis on les applique dans des exercices de plus en plus difficiles, on fait des erreurs, on les analyse et on les corrige. Ce chemin, même s'il est parfois frustrant, construit des compétences solides et durables. L'IA peut court-circuiter ces étapes importantes, ce qui risque de donner des connaissances superficielles.

Voici les compétences qui peuvent être affectées :

- **Compréhension des algorithmes et structures de données** : Ces bases sont essentielles pour bien programmer. Normalement, on les apprend en écrivant soi-même différentes solutions et en comparant leurs performances. L'IA peut générer ces solutions tout de suite, ce qui prive l'étudiant de l'expérience d'apprendre vraiment comment elles fonctionnent.

- **Capacité à créer des solutions adaptées à des problèmes précis :** Créer des solutions sur mesure demande de bien comprendre les contraintes et les compromis. Cette capacité se développe en essayant différentes approches et en les améliorant. L'IA peut raccourcir ce processus d'apprentissage.

- **Maîtrise du langage de programmation :** On apprend vraiment un langage en l'utilisant encore et encore. L'autocomplétion de l'IA peut réduire cette pratique répétée qui est nécessaire pour bien maîtriser la syntaxe et les particularités du langage.

- **Capacité à déboguer et résoudre des problèmes :** Le débogage s'apprend surtout par l'expérience. La capacité à trouver et corriger des bugs complexes se développe en affrontant des problèmes difficiles et en trouvant des méthodes pour les résoudre.

Pour les professeurs, cette évolution pose de vrais défis. Comment savoir si un étudiant a vraiment compris un concept quand l'IA peut générer des solutions complètes à partir de simples descriptions. Comment faire la différence entre quelqu'un qui maîtrise vraiment un concept et quelqu'un qui sait juste bien formuler des demandes à l'IA.

Sources :

(https://www.ibm.com/fr-fr/think/topics/ai-in-software-development#:~:text=La%20d%C3%A9pendance%20excessive%20%C3%A0%20l,IA%20ou%20de%20r%C3%A9sultats%20erron%C3%A9s.)

(https://www.itta.net/blog/lia-fait-elle-lunanimite-chez-les-developpeurs/)

IV. Former les développeurs pour une meilleure expertise

Face aux défis que pose l'IA dans le développement, il faut repenser comment on forme les développeurs. Le but n'est pas de rejeter ces outils puissants mais d'apprendre à les utiliser de façon équilibrée, en gardant notre autonomie technique. Cette partie propose des stratégies concrètes pour trouver cet équilibre.

A. Pratiques d'utilisation recommandées

Adopter une approche réfléchie quand on utilise l'IA est la première défense contre la dépendance. Ces pratiques ne visent pas à limiter l'utilité des outils, mais à maximiser leurs avantages tout en minimisant les risques pour l'autonomie et les compétences du développeur.

Une utilisation équilibrée demande de bien connaître les forces et limites de l'IA, et d'avoir une stratégie pour maintenir ses compétences de base. Tous les développeurs, débutants comme experts, peuvent adopter ces pratiques :

- **Utiliser l'IA surtout pour le code répétitif :** L'IA est très utile pour générer du code standard et des structures répétitives. En l'utilisant principalement pour ces cas, on gagne du temps sans perdre sa capacité à résoudre des problèmes plus créatifs.

- **Configurer les outils pour limiter certaines fonctions :** On peut personnaliser les outils d'IA pour qu'ils suggèrent des approches plutôt que des solutions complètes, ou pour qu'ils se concentrent sur certains types de code. Cela aide à garder un bon équilibre.

- **Toujours comprendre le code généré :** Avant d'accepter une suggestion de l'IA, il faut prendre le temps de la comprendre. Cela évite d'utiliser du code inapproprié et transforme l'outil en opportunité d'apprentissage.

- **Alterner entre périodes avec et sans IA :** Prévoir des moments où on désactive l'IA peut aider à maintenir ses compétences. C'est particulièrement utile quand on apprend une nouvelle technologie.

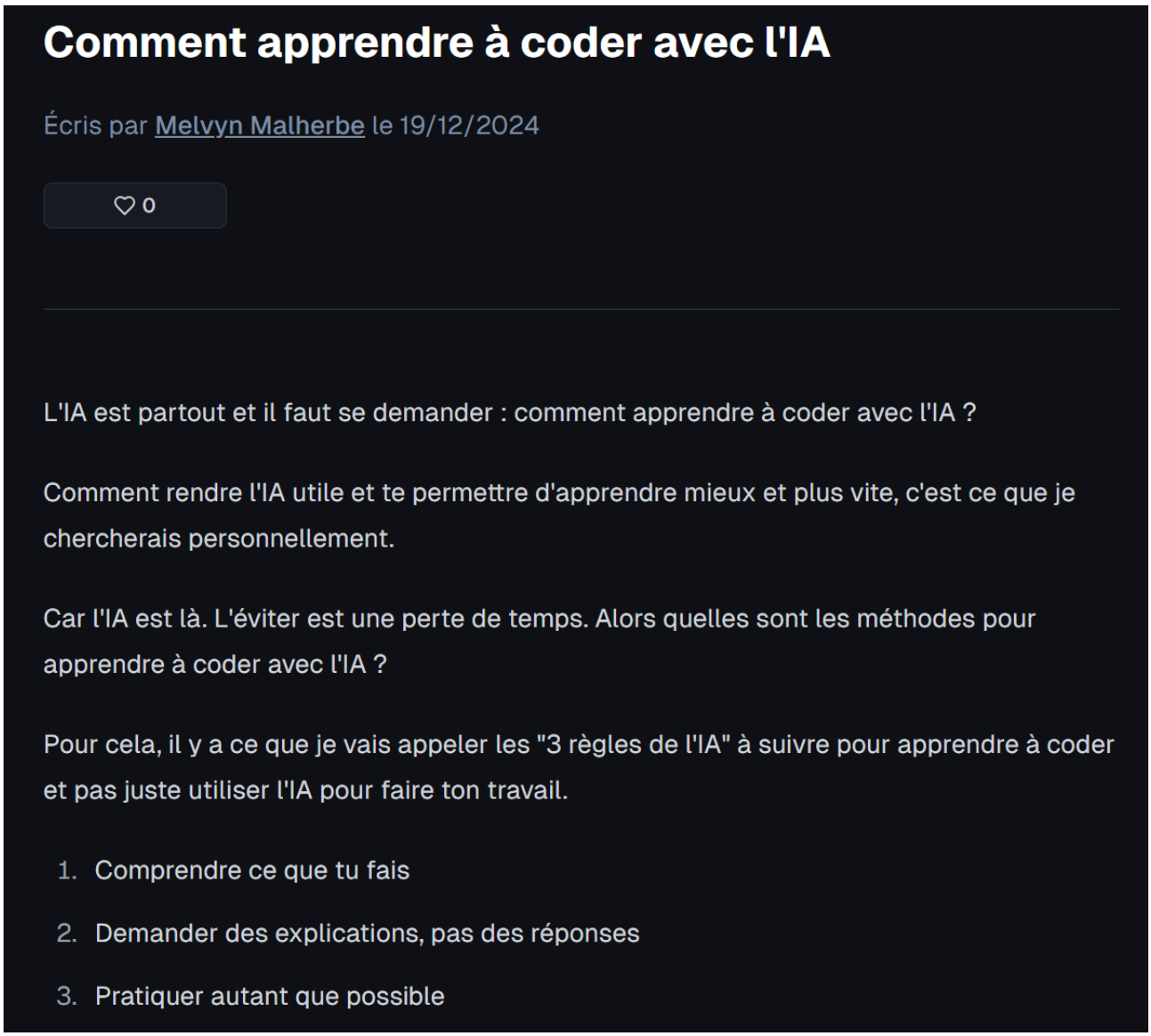
Il est aussi important d'avoir un regard critique sur les suggestions de l'IA :

- **Évaluer si les suggestions conviennent au contexte :** L'IA ne comprend pas toujours parfaitement les particularités d'un projet. Il faut vérifier si la solution proposée correspond bien aux besoins spécifiques du projet.

- **Comparer plusieurs suggestions :** Quand c'est possible, demander plusieurs variantes à l'IA permet d'exercer son jugement technique et de choisir la meilleure solution.

- **Modifier le code généré :** Le code de l'IA devrait être considéré comme un point de départ, pas une solution finale. L'adapter et l'optimiser pour le contexte permet de garder sa créativité.

- **Documenter sa compréhension :** Ajouter des commentaires qui expliquent comment fonctionne le code généré, ou même le réécrire dans ses propres mots, aide à solidifier sa compréhension.



Les "3 règles de l'IA" qui ressortent de ma veille résument bien cette approche :

1. Comprendre ce que vous faites (toujours analyser le code généré)
2. Demander des explications, pas seulement des réponses
3. Pratiquer régulièrement sans assistance IA

Sources :

https://www.numendo.com/blog/intelligence-artificielle/comment-utiliser-lia-quand-on-est-developpeur/

https://marketing.ces.ncsu.edu/ai-guidance/#:~:text=Fact%2DChecking%3A%20Always%20verify%20AI,provide%20false%20%E2%80%9Cfacts%E2%80%9D%20at%20times

https://codelynx.dev/posts/comment-apprendre-a-coder-avec-lai

B. Formation et adaptation des développeurs

Les écoles et les programmes de formation ont un rôle crucial pour préparer les développeurs à cette nouvelle ère. Les méthodes traditionnelles d'enseignement conçues pour un monde où le code était entièrement écrit à la main doivent évoluer pour intégrer ces nouveaux outils tout en préservant les objectifs fondamentaux de l'éducation technique.

Cette évolution demande un équilibre délicat : adopter l'innovation technologique sans compromettre l'acquisition des compétences fondamentales. Les programmes comme le BTS SIO sont particulièrement concernés, car ils forment des professionnels qui entreront directement sur un marché du travail où ces outils sont de plus en plus présents.

Sources :

https://www.ekole.fr/blog/integration-lia-dans-formation-enseignants-defi

https://www.sorbonne-universite.fr/dossiers/intelligence-artificielle/lia-dans-leducation-entre-opportunites-et-defis

Exemple de formation sur OpenClassRoom :

Les programmes de formation devraient inclure :

- **Intégration de l'utilisation critique de l'IA dans les cours :** Au lieu d'ignorer ces outils ou de les considérer comme de la triche, les programmes devraient les intégrer explicitement, en enseignant comment les utiliser efficacement et éthiquement.

- **Renforcement de l'enseignement des principes fondamentaux :** Paradoxalement, l'arrivée de l'IA rend encore plus important l'enseignement approfondi des concepts de base comme les algorithmes et les structures de données.

Conclusion

Pour conclure, l'IA dans VS Code change profondément la façon dont les développeurs travaillent au quotidien. Ces outils font gagner beaucoup de temps en automatisant les tâches simples et répétitives comme l'écriture de fonctions standard ou la recherche dans la documentation. Ils aident

aussi à produire du code de meilleure qualité en suggérant des approches optimisées et en rappelant les bonnes pratiques de programmation.

Cependant, cette évolution pose plusieurs problèmes importants pour les développeurs. Le principal risque est de devenir trop dépendant de ces assistants IA et de perdre progressivement certaines compétences fondamentales en programmation. Quand on s'habitue à ce que l'IA propose des solutions, on peut perdre l'habitude de résoudre des problèmes par soi-même. Il existe aussi des questions sans réponse claire concernant les droits d'auteur du code généré par l'IA et la sécurité des applications créées avec cette aide.

Pour faire face à ces défis, la formation des développeurs doit évoluer. Il ne suffit pas d'apprendre à utiliser l'IA pour coder plus vite. Les développeurs doivent aussi apprendre à garder leur esprit critique face aux suggestions de l'IA et à maintenir leurs compétences techniques. Les écoles comme le BTS SIO ont un rôle important à jouer : elles doivent intégrer ces nouveaux outils dans leurs cours tout en continuant à enseigner les bases solides de la programmation.

Dans les années à venir, le métier de développeur va certainement changer. L'IA prendra en charge de plus en plus les tâches basiques de programmation comme la création de fonctions simples ou le débogage de base. Les développeurs, eux, se concentreront davantage sur la conception d'ensemble, l'architecture des applications et les aspects plus créatifs du développement. Ce n'est pas une diminution de l'importance du métier, mais plutôt une évolution vers des compétences de plus haut niveau où la créativité et l'expertise humaines restent indispensables.

En fin de compte, l'IA dans VS Code est un outil puissant qui peut améliorer notre travail mais c'est à nous de décider comment l'utiliser de façon équilibrée pour en tirer les bénéfices sans en subir les inconvénients.