Pourquoi Google Code Assist est peut-être enfin l'outil de programmation dont vous avez besoin

Google Code Assist inclut désormais Gemini 2.5 dans sa version gratuite. Reste à savoir qui peut utiliser les nouveaux agents d'intelligence artificielle.

PAR DAVID GEWIRTZ
Publié le 14/04/2025 à 18:01

6 min

© Samuel Boivin/NurPhoto via Getty Images

Google Next 2025 (Las Vegas) - Il y a du nouveau pour les développeurs avec les **annonces concernant Gemini Code Assist**. Déjà l'an passé, l'outil d'aide à la programmation **Google Code Assist** proposait de nombreuses possibilités d'intégration. Mais l'outil était cependant entravé par une limitation : Gemini codait très mal. Lorsque j'ai effectué mes tests de codage l'année dernière, Gemini Advanced, basé sur **Gemini 1.5**, a échoué à trois tests sur quatre.

Mais Google a maintenant **Gemini 2.5**. Et *cette* version n'a pas échoué à mes tests de codage. **En fait, elle a réussi.** Donc cette fois, le code produit (ou qu'il vous aide à produire) fonctionnera probablement.

Google Code Assist passe ainsi du statut de petite curiosité agréable à celui d'un outil qui pourrait bien devenir le cœur de vos efforts de programmation. J'ai réussi à capter votre attention, n'est-ce pas ? Alors, entrons dans le vif du sujet.

Quoi de neuf avec Gemini Code Assist ?

Gemini Code Assist se décline **en trois offres** :

- Individuel
- Standard
- Entreprise

Des différences en fonction des abonnements

- La variante individuelle est gratuite.
- En passant à la version standard pour 22,80 $/utilisateur/mois ou 228 $/utilisateur/an, vous bénéficiez de Gemini dans Firebase (la plateforme de Google pour les développeurs) et de Gemini dans les bases de données.
- Le passage à la version entreprise pour 54 $/utilisateur/mois ou 540 $/utilisateur/an permet d'ajouter des suggestions de code personnalisées à partir de vos bases de code GitHub, GitLab et Bitbucket, de Gemini dans BigQuery et Apigee, ainsi que des fonctionnalités de flux d'automatisation pour l'intégration d'applications.

La grande nouveauté pour la version gratuite de Gemini Code Assist est qu'elle sera désormais livrée avec Gemini 2.5. Ce qui est à mon avis une bonne nouvelle. Mais il y a un petit mic-mac. Google précise que Gemini 2.5 et Gemini 2.5 Pro sont deux produits distincts. La version Pro est un modèle de réflexion. Voici un **billet de blog** qui explique la différence.

Mais Google assure que les développeurs peuvent bénéficier des résultats de performance de Gemini 2.5 Pro. Selon le responsable des relations publiques, "Gemini 2.5 Pro est dé[illegible]nais disponible pour les développeurs dans le cadre de notre offre Gemini Code As[illegible]or individuals [la version gratuite]."

Des agents d'IA pour Gemini Code Assist

Google annonce des capacités d'agent pour Gemini Code Assist.

Nulle part dans le **tableau de comparaison des versions de produits** ou dans le communiqué de presse, il n'est précisé si ces fonctionnalités d'agent sont disponibles pour les versions individuelle, standard ou entreprise, ou seulement pour une ou deux de ces variantes. Mais Google m'a précisé : "Les nouveaux agents sont disponibles en preview privée et tout le monde peut s'y inscrire. La disponibilité des niveaux sera communiquée ensuite."

Il y a un grand pas entre l'écriture d'un code sur votre ordinateur et la livraison d'un produit à un utilisateur. De bout en bout, c'est ce que nous appelons le cycle de vie du logiciel. Les agents Gemini Code Assist de Google doivent faciliter les tâches tout au long du cycle de vie. L'entreprise a annoncé des agents dans les sept domaines suivants.

1. Générer de nouveaux logiciels

L'entreprise affirme que Google Code Assist créera des applications à partir de spécifications de produits rédigées dans Google Docs. Et cela pourrait poser beaucoup de problèmes.

À moins que Google ne documente également un langage d'écriture des spécifications, l'IA risque de manquer de nombreuses fonctionnalités. Et ce type de fonctionnalité donnera aux non-programmeurs l'idée qu'ils peuvent générer un code fonctionnel. Mais décrire ce que l'on veut n'est qu'une petite partie du projet global.

Et le processus de va-et-vient n'est pas clair. Le nouveau code est-il généré chaque fois qu'un document est mis à jour ? Faut-il écrire une application complète ou peut-on spécifier des fonctions et des sous-programmes qui seront ensuite incorporés ? Autant de questions à répondre.

2. Migrer le code

L'idée est que les agents peuvent transformer le code d'un langage à l'autre et traduire le code entre les langages et les frameworks.

J'ai testé et j'ai constaté que ça marche. En supposant que les langages de départ et d'arrivée disposent de facilités pour l'algorithme à convertir, il s'agit d'un outil assez efficace.

Mais le passage d'un framework à l'autre peut s'avérer compliqué. S'il s'agit d'une seule fonction, je pense que la migration est possible. Mais si vous envisagez de transférer l'intégralité de votre dossier et d'essayer de faire fonctionner l'intelligence artificielle sur une plate-forme différente, je pense que vous aurez du pain sur la planche.

3. Implémenter de nouvelles fonctionnalités depuis GitHub

L'idée est que l'IA lise les "issues" de GitHub et les implémente dans le code. Et ces problèmes sont des rapports de bogues ou des demandes de fonctionnalités. Ainsi, si un

problème répertorié est "Add 2FA to login", l'idée serait que l'IA lise ce problème et écrive un code d'authentification à deux facteurs dans la base de code.

Et oui, pour certains problèmes, ce serait un gain de temps appréciable. Mais l'exemple de l'authentification à deux facteurs présente de sérieuses limites. Par exemple, quelle méthode d'authentification à deux facteurs serait utilisée ? Quels sont les authentificateurs pris en charge ? Quelles sont les bibliothèques à utiliser ? Faut-il utiliser des bibliothèques gratuites ou acquérir une licence avec des fonctionnalités supplémentaires ?

Il est assez facile de mettre en œuvre une fonctionnalité. Ce qui me préoccupe, c'est que les codeurs le fassent et cessent de réfléchir à la manière de mettre en œuvre ces fonctionnalités.

4. Effectuer des revues de code

Je pense qu'il s'agit là d'une excellente utilisation des agents d'intelligence artificielle. Même sans agents, j'ai donné à ChatGPT (parce que jusqu'à récemment Gemini ne pouvait pas le gérer) un prompt disant "qu'est-ce qui ne va pas avec ce code", suivi d'un bloc de code.

J'ai fait cela pour voir si l'IA pouvait trouver une faille dans mon implémentation. Quelques fois, j'ai eu de la chance et les suggestions faites étaient mineures. Mais la plupart du temps, ce prompt a permis de découvrir une erreur ou une omission de codage assez grave.

La formalisation de ce processus dans une procédure d'examen du code pourrait s'avérer extrêmement utile. J'ai hâte de tester cet agent par moi-même.

5. Générer des tests

J'ai vu une version de cette démonstration lors de mon test de programmation avec Gemini 2.5 Pro. J'ai demandé à Gemini de corriger un calcul d'expression régulière. Et non seulement l'IA a fait ce que je demandais, mais elle m'a également fourni un ensemble de cas de test (à la fois positifs, où cela devait fonctionner, et négatifs, où cela était censé échouer).

Cela m'a permis de confirmer rapidement et facilement la fonctionnalité du code.

L'utilisation d'une IA pour construire et exécuter des tests et rendre compte des résultats est une excellente utilisation de cette technologie. Bien sûr, vous devrez examiner les tests, étudier la manière dont ils sont mis en œuvre. Il s'agit néanmoins d'un excellent moyen d'accroître la fiabilité du code tout en gagnant du temps.

6. Effectuer des tests de modèles d'IA

Si vous souhaitez voir comment un modèle d'IA se comporte avec certaines entrées et sorties, vous pouvez utiliser l'IA pour créer une série de tests de validation.

Cela peut s'avérer particulièrement utile si vous vous préoccupez de la sécurité du contenu et des performances des sous-systèmes d'IA mis en œuvre dans les architectures de codage.

7. Créer de la documentation

Google a intégré un générateur de wiki avec l'un de ses agents. De nombreux programmeurs ont tendance à ne pas se préoccuper de la documentation.

Même si l'IA ne produit pas le niveau de documentation qu'un bon rédacteur technique ayant une formation en codage peut produire, elle pourrait créer un très bon point de départ qui peut être modifié, amélioré, corrigé et peaufiné.
Source : "ZDNet.com"

/ Plus d'actualités

La MaJ Gemini 2.5 Pro améliore les capacités de codage de l'IA de Google

La mise à jour devait être présentée lors de la conférence Google I/O. Mais Google l'a publiée plus tôt en réponse aux réactions positives de...

PAR SABRINA ORTIZ | 07 MAI 2025

Voici les nouvelles fonctionnalités d'IA de Microsoft qui arrivent sur les PC Copilot+ - dont certaines pour tous les utilisateurs de Windows 11

Voici toutes les nouvelles compétences d'IA et les améliorations qui viennent d'être annoncées, ainsi que les personnes qui peuvent...

PAR LANCE WHITNEY | 07 MAI 2025

La main de fer de Google sur la recherche est rouillée - voilà ce que j'utilise à la place

La part de marché de Google Search diminue et ce n'est pas seulement à cause de l'IA. Explications.

PAR STEVEN VAUGHAN-NICHOLS | 07 MAI 2025

ZDNET Morning 07/05/2025 : Bercy et le cloud souverain, Pare-feu spécial IA, Onglet Opera Android,...

Le ZDNET Morning le brief de l'actu tech pour les pros tous les matins à 9h00. Transformation numérique, IA, matériel, logiciels,... ne passez pa...

PAR GUILLAUME SERRIES | 07 MAI 2025

/ Plus de guides d'achat

Google Pixel 8a vs Google Pixel 9a : quel smartphone choisir ?

Le précédent modèle de mobile disposait de fonctionnalités phares à un prix de milieu de gamme lors de sa sortie. Mais avec l'arrivée du...

PAR JASON HOWELL | 18 AVRIL 2025