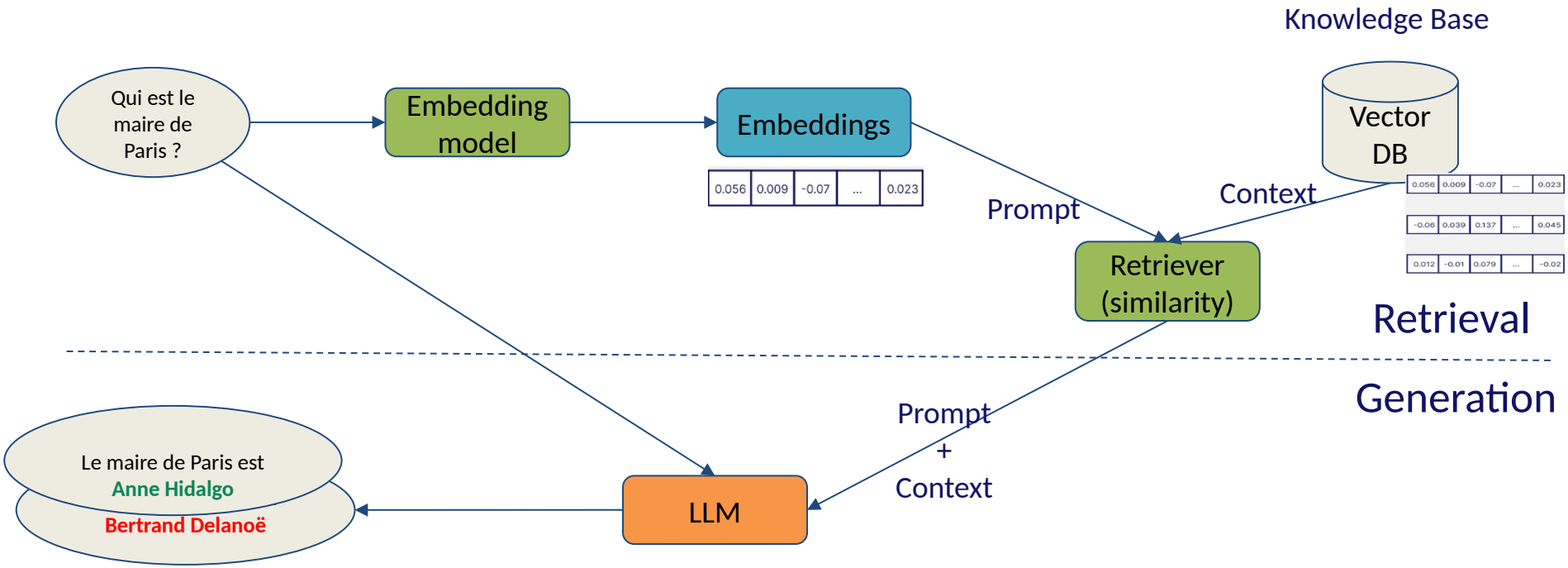
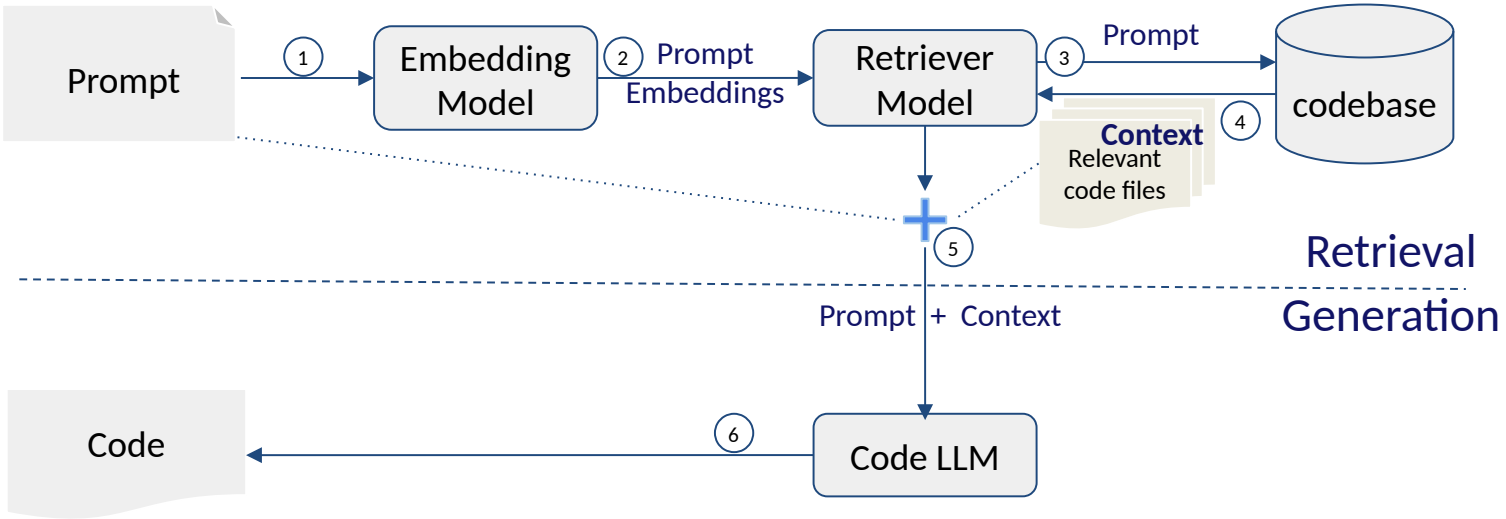
Indexation de code sous Github Copilot

Limites des LLMs



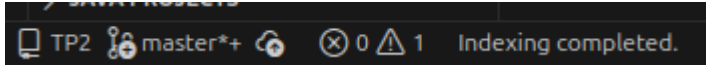
Retrieval-Augmented Generation (RAG)

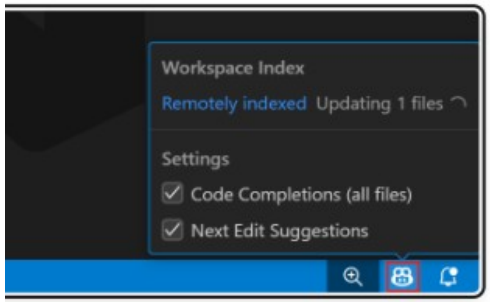
- La génération augmentée par la recherche (RAG) intègre la recherche d'informations externes au processus de génération de réponses par les grands modèles de langage (LLM).
- Elle interroge une base de données pour trouver des informations complémentaires à sa base de connaissances pré-entraînée, améliorant ainsi considérablement la précision et la pertinence des réponses générées.



RAG sous Copilot

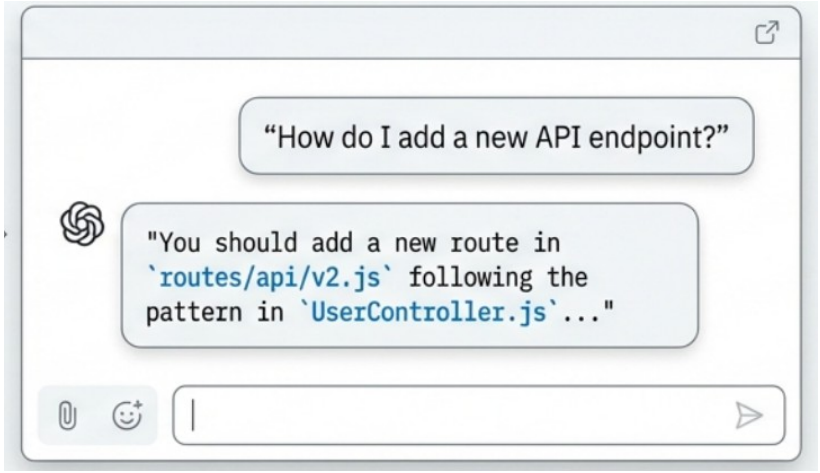
- VS Code utilise un index de l'espace de travail pour permettre à Copilot Chat de rechercher efficacement dans votre code.
- Cet index peut être
  - distant (géré par GitHub/Azure DevOps)
  - local (stocké sur votre machine).
- Le tableau de bord Copilot dans la barre d'état indique l'index utilisé.





Contexte de l’espace de travail

- Mécanisme qui permet à Copilot-Chat de comprendre l’ensemble de votre code.
- Permet de poser des questions globales au niveau du projet avec des réponses basées sur votre code réel.



Fonctionnement

- Stratégies de recherche intelligentes à travers les fichiers du projet.
  - Utilise la structure des dossiers, les noms de fichiers, les symboles et le texte sélectionné.
  - Garde uniquement les éléments les plus pertinents si le contexte est trop large.

Types d’index de l’espace de travail

- Index distant
  - pour les dépôts GitHub/Azure DevOps.
- Index local
  - construit sur votre machine pour les petits/moyens projets.
- Index basique
  - solution de repli pour les très grands dépôts.

Index distant

- Construit à partir de la version committée du dépôt sur GitHub ou Azure DevOps (les modifications locales non committées ne sont pas incluses).
- En présence de modifications locales, VS Code combine l'index distant avec le suivi des fichiers locaux et le contenu actuel de l'éditeur.
- Activation:
  - automatiquement lors de l'utilisation de @workspace ou #codebase,
  - ou peut être déclenchée manuellement :
    - Exécuter la commande **Build Remote Workspace Index** dans la palette de commandes.
- Fonctionne automatiquement pour les dépôts GitHub et Azure DevOps après connexion.
- Rapide pour les petits et moyens projets.
- Peut prendre plus de temps pour les très grands dépôts.
- L'indexation GitHub fonctionne pour GitHub.com et GitHub Enterprise Cloud, mais pas pour GitHub Enterprise Server.

Index local

- Utilisé lorsqu'un index distant n'est pas disponible.
- Supporte jusqu'à **2500 fichiers** indexables.
- Moins de **750** fichiers :
  - index local créé automatiquement.
- Entre **750 et 2500** fichiers :
  - construction manuelle via Build local workspace index.
- **Plus de 2500** fichiers :
  - index local avancé non disponible → passage à l'index basique.
- Peut prendre du temps à construire ou mettre à jour, notamment après de nombreux changements (ex. : changement de branche).

Index basique

- Solution de repli pour les grands projets (**>2500 fichiers**) sans index distant.
- Utilise des algorithmes de recherche plus simples.
- Fonctionne pour la majorité des requêtes, mais peut être moins pertinent.
- Si les résultats sont insuffisants, il est recommandé d’utiliser un dépôt GitHub ou Azure DevOps pour activer un index distant.

Limites de taille

- La capacité de contexte, context window, représente la quantité de code, de texte ou d'instructions qu'un modèle peut charger d'un coup. (inputs+outputs)
- Une grande fenêtre permet d'interroger une grande base de code, d'ingérer plusieurs fichiers simultanément sans perte d'information.
- Claude
  - Par défaut (modèles payants), Claude offre une fenêtre d'environ **200 000** tokens.
  - Pour les utilisateurs "Enterprise" (modèle Sonnet 4.5), Claude peut offrir une fenêtre étendue à **500 000** tokens. (100 tokens représentent environ 75 mots)
- GPT-5
  - Via l'API (usage développeur), GPT-5 peut gérer jusqu'à **400 000** tokens.
  - Dans l'interface grand public (chat), la fenêtre dépend du plan :
    - les versions "Pro/Enterprise" peuvent atteindre **128 000** tokens.

Stratégie de recherche

- Pour les petits projets, l'espace de travail entier peut être directement inclus dans le contexte de la conversation.

- Pour les projets plus importants, VS Code utilise différentes stratégies afin de trouver les informations les plus pertinentes à inclure dans le contexte de la conversation pour répondre à votre question.

Stratégie de recherche

1. Détermination des informations de l'espace de travail nécessaires pour répondre à votre question, y compris l'historique des conversations, la structure de l'espace de travail et l'éditeur actuellement sélectionné.

2. Collecte d'extraits de code pertinents à partir de l'index de l'espace de travail à l'aide de différentes méthodes :
    a. Recherche de code GitHub
    b. Recherche sémantique locale pour trouver du code correspondant au sens de votre question, et pas seulement des mots-clés exacts
    c. Recherche textuelle par nom de fichier et par contenu
    d. IntelliSense de VS Code pour ajouter des détails tels que les signatures de fonctions, les paramètres, etc.
3. Si le contexte résultant est trop volumineux pour tenir dans la fenêtre de contexte, seules les parties les plus pertinentes sont conservées.

Utilisation du contexte dans le chat

- Utilisez @workspace pour traiter le projet comme un expert.
- Utilisez l'outil #codebase pour une recherche flexible.
- Utilisez l'outil #nom_de_fichier pour être plus explicite.
- Le mode Ask/Edit ajoute automatiquement du contexte pertinent.
- Le mode Agent/Plan effectue des recherches supplémentaires.

Conseils pour une utilisation efficace

- Posez des questions spécifiques avec des mots-clés liés au code.
- Faites référence à des fichiers, dossiers ou sélectionnez du code pour une meilleure précision.
- Tout le contexte ne contient pas les métadonnées (auteurs, revues, etc.).

Confidentialité et limites

- Respecte .gitignore et les exclusions de fichiers de VS Code.
- Les fichiers binaires ne sont pas indexés.
- L’index distant peut avoir un léger retard par rapport aux modifications locales.

@workspace vs #codebase

- @workspace
  - un participant expert
  - aucun autre outil ne peut être utilisé.
  - Utiliser quand le prompt concerne votre code seulement.
- #codebase
  - un outil de recherche flexible.
  - d'autres outils sont utilisés.
  - Utiliser quand vous avez besoin de chercher du code et entreprendre d'autres actions.
- Recommandation : privilégier #codebase pour les tâches plus larges.

Limitations

- Les grands projets peuvent réduire le contexte.
- Les fichiers exclus sont ignorés (.gitignore)sauf s’ils sont ouverts.
- Pas idéal pour des refactorings massifs nécessitant une vue complète du projet.

Résumé

- Le contexte de l’espace de travail apporte une compréhension globale du projet.
- Utilisez @workspace ou #codebase.
- Combinez avec du contexte explicite pour obtenir les meilleurs résultats.