



Do Tank « Impacts de l'IA sur les activités d'ingénierie logicielle »

GT 2/6

« Vibe Coding / Serveurs MCP »

Les sociétés ayant participé au GT



Pilote du GT :
Jean-Pierre Brajal

amadeus

bpi**france**

The ENGIE logo features a blue curved line above the word 'ENGIE' in a blue, lowercase, sans-serif font.

hagergroup

in**vivo**

The mgen logo consists of a green square containing the word 'mgen' in white lowercase letters, with a small white star above the 'n'. Below the square, the text 'GROUPE vyv' is written in a smaller, black, lowercase font.

The TotalEnergies logo features a stylized 'T' and 'E' in a rainbow gradient, with the word 'TotalEnergies' in red below it.

- 1. Introduction : les tendances émergentes du développeur augmenté**
- 2. Focus sur le Vibe Coding**
- 3. Focus sur les serveurs MCP**
- 4. Conclusion et perspectives**
- 5. Annexe : bibliothèque partagée de serveurs MPC**

01

Introduction : tendances émergentes du développeur augmenté

Introduction

L'année 2025 a été structurante sur l'impact de l'IA générative et agentique dans le milieu de l'ingénierie logicielle.

Deux tendances ont été particulièrement marquantes :



L'apparition du « Vibe Coding »



Décrire en langage naturel (écrit ou oral) ce que l'on veut coder, et laisser l'IA générer le code



L'émergence de MCP (*Model Context Protocol*)



Protocole standard d'exposition d'outils (bases de données, APIs, fichiers) aux agents IA



Toutes deux permettent de **présager le futur du développement logiciel**, et l'impact de l'IA sur l'ensemble des activités associées.

02

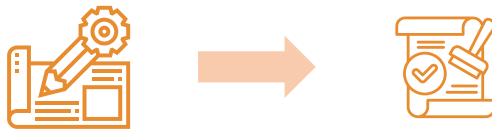
Focus sur le Vibe Coding

Vibe Coding - Introduction

Le Vibe Coding consiste à **décrire en langage naturel** (écrit ou oral) ce que l'on veut coder, et **laisser l'IA générer le code**. Au-delà des **interactions conversationnelles** (*Chat*) et de **l'édition assistée** (*Edit*) rendues possibles par les LLM, le Vibe Coding introduit une approche de type agent, permettant de **concevoir automatiquement tout ou partie d'une application**.

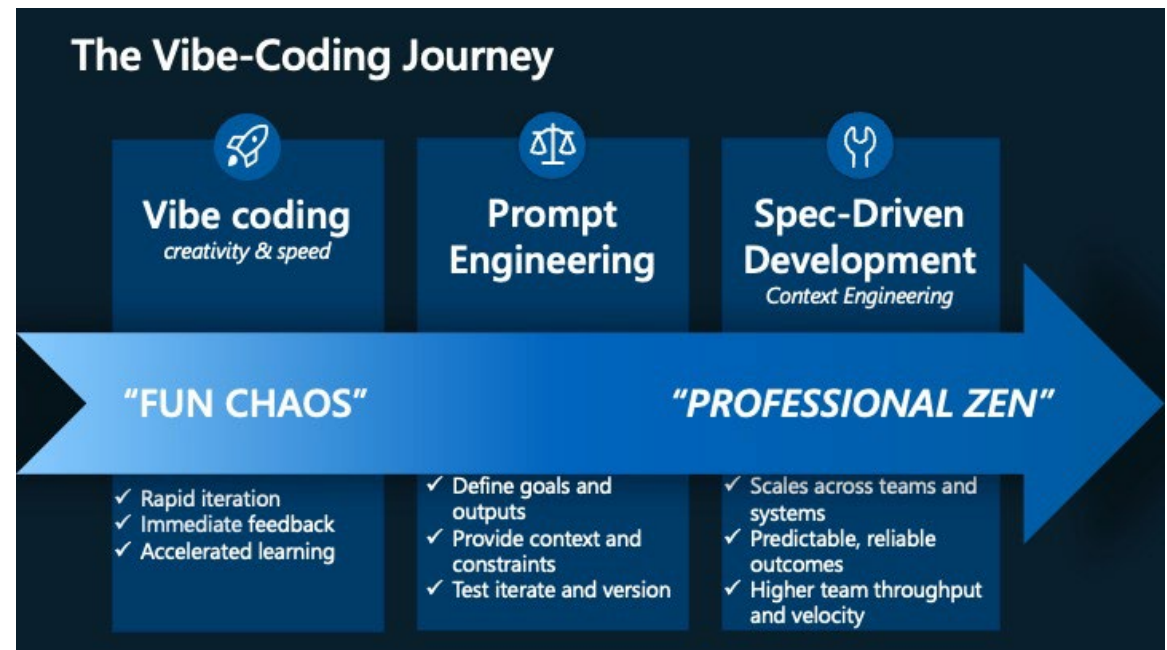


Depuis 2025, le Vibe Coding s'impose comme une **pratique majeure**, portée par l'essor des LLM et des outils agentiques.



Il amène à un **changement de posture** dans le rôle du développeur. Celui-ci, jusque là centré sur **l'écriture manuelle de code**, s'oriente désormais vers la **conception, l'orchestration** et la **validation** de ce qui est proposé par l'IA.

Si le Vibe Coding peut être associé à une perte de maîtrise, il peut cependant s'inscrire dans des pratiques d'entreprise avec de forts standards.



Source : AWS ([lien](#))

Des cas d'usage qui ont gagné en complexité et valeur ajoutée dans le courant de l'année 2025

Si le Vibe Coding a été associé sur le premier semestre à des **cas d'usage simples et rapides**

...



Le second semestre 2025 a marqué un tournant, avec l'apparition de **cas d'usage plus complexes et à plus forte valeur ajoutée.**



Prototypage rapide et MVP (*Minimum Viable Product*)

Tests d'idées, préparation de démonstrations en quelques heures (au lieu de semaines)



Applications simples et sites vitrines

Création de landing pages, petits scripts, automatisations internes



Apprentissage et exploration

Découverte de nouveaux frameworks ou technologies sans effort initial.



Développement assisté pour projets complexes

Avec des outils comme GitHub Copilot en mode Agent, on peut générer des tests unitaires, migrer des frameworks (ex. Spring Boot), ou améliorer la qualité du code



Créativité et design

Génération d'interfaces, wireframes (maquettes fonctionnelles), et ajustements visuels en temps réel.

Des prérequis légers mais essentiels, afin d'éviter que la vitesse ne se traduise par une perte de qualité

Il est nécessaire de donner un cadre d'utilisation du Vibe Coding, afin d'activer tout son potentiel et de prévenir les risques de dérive



Au niveau du Vibe-Coder

Compétences de base en informatique

Savoir installer des outils, gérer des fichiers.

Notions de sécurité et bonnes pratiques

Vérifier le code généré, éviter les secrets en clair, tester avant déploiement.

Culture du prompt engineering :

Savoir rédiger des instructions claires et structurées (contexte, contraintes, format), et guider les développements pour éviter du code trop verbeux.

Outillage

Parmi les outils recommandés en décembre 2025 : VS Code, Cursor, GitHub Copilot, Windsurf, Antigravity, Kiro.

Au niveau de l'entreprise

Architecture et design system

Définition d'un cadre de développement, et sa mise à disposition aux outils IA (formats intégrables et possibilités d'accès).

Mise en place de garde-fous

Audits de sécurité, gouvernance des prompts.

Tests automatisés

Intégration et/ou génération de tests unitaires pour vérifier les productions. Mise à disposition d'outils permettant à l'IA de valider ses résultats.

Documentation technique et fonctionnelle toujours à jour

Y compris en utilisant l'IA.

Les risques

Le Vibe Coding apporte vitesse et créativité, mais comporte des dangers :



Sécurité (*Création de failles dans le code généré*)



Niveaux de qualité et de maintenabilité qui peuvent être inférieurs si mal encadrés



Manque de tests et de validation par défaut



Risque de dépendance des collaborateurs aux outils et de perte de maîtrise



Augmentation des coûts, perte de maîtrise du modèle économique et évolution des coûts à la hausse



Fausse impression de productivité, risque de passer plus de temps à corriger l'IA qu'à coder soi-même



Risque de se passer de l'expertise liée au développement qui reste nécessaire



Risque géopolitique qui pourrait interrompre notre capacité à développer en cas de conflit

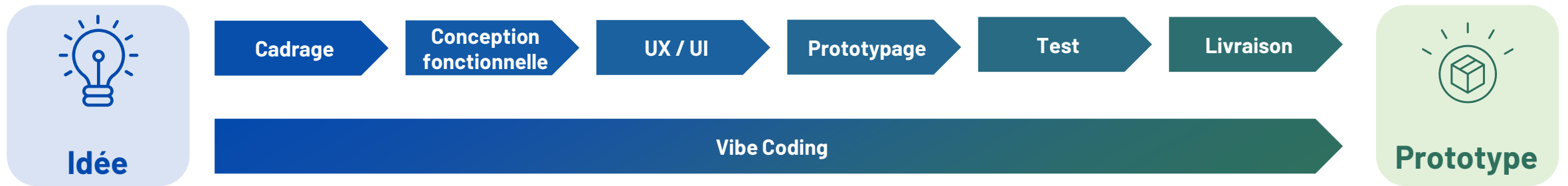


La trajectoire du Vibe Coding s'oriente vers une intégration profonde dans les processus de développement d'entreprise, avec des outils agentiques et des standards de gouvernance.

L'enjeu : concilier vitesse et robustesse.

Les impacts organisationnels associés

Le Vibe Coding permet d'accélérer drastiquement le passage de l'idée au prototype fonctionnel, en s'adaptant au contexte métier et technique de l'entreprise :



Le goulot d'étranglement devient la pratique

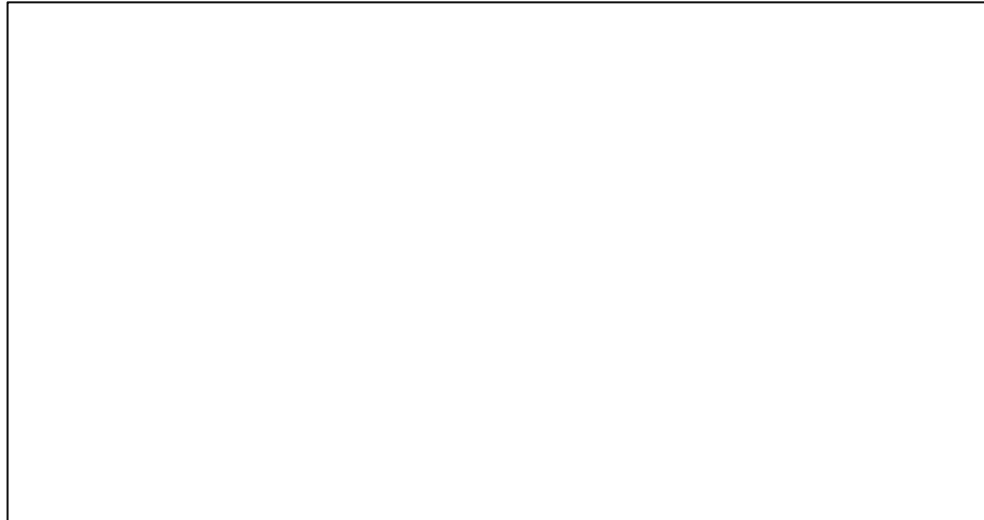
Si on passe trop de temps à revoir du code ou une architecture définie par l'IA en quelques heures, on n'accélère pas le cycle de vie logiciel, et les gains espérés vont rester marginaux.



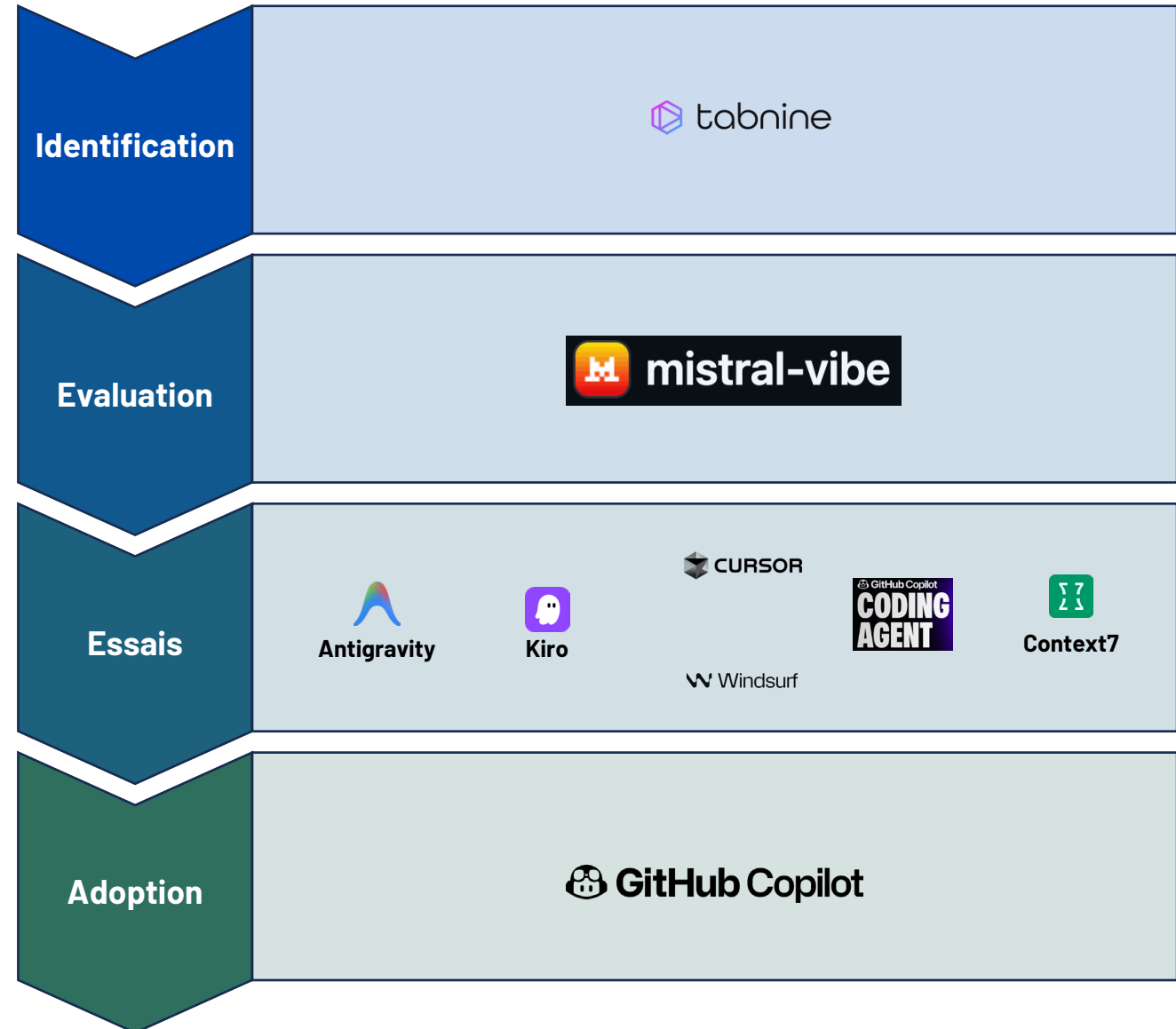
Ce nouveau paradigme amène à redéfinir la façon de faire du développement logiciel

Il faut revoir la gouvernance (comitologie, rituels, processus), les rôles et les pratiques. Ces changements profonds doivent s'envisager à long terme, mais doivent être préparés dès maintenant.

Vibe Coding – Positionnement par stade d'utilisation, des outils identifiés par les entreprises participantes



Radar des outils des GT
(cf. annexe)



03

Focus sur les serveurs MCP

MCP (Model Context Protocol) – Introduction

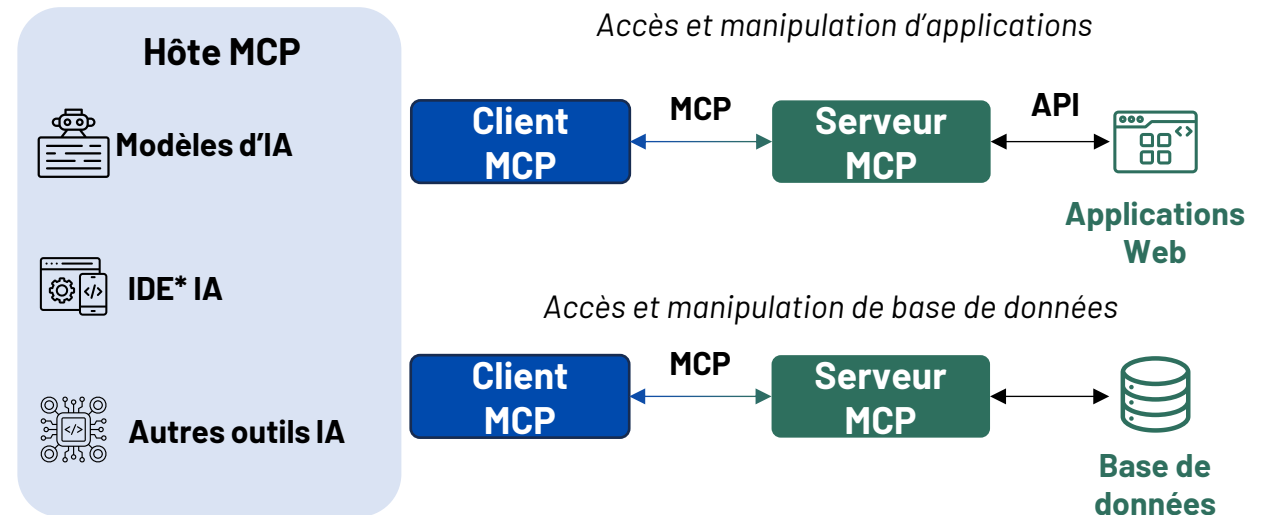


L'utilisation du MCP qui permet d'exposer des outils aux agents IA, s'est imposée comme la **condition sine qua non pour faire de l'agentique**, en connectant les « cerveaux » des IA à ces outils



Ce protocole **étend considérablement les possibilités permises par les modèles d'IA**, sous réserve de respecter certaines **précautions** (cf. planches suivantes).

Les serveurs MCP sont des composants d'un protocole ouvert standardisé qui **connecte les modèles d'IA**, comme les grands modèles de langage, à des **outils dynamiques** (bases de données, APIs ou fichiers) via une architecture client-serveur basée sur JSON-RPC.



* L'IDE (*Integrated Development Environment*) est une application logicielle qui permet aux développeurs de créer, d'éditer et de compiler du code au sein d'un même programme.



Quelques chiffres sur l'adoption en 2025

- Plus de **5 500 serveurs MCP** ont été développés en moins d'un an, avec une croissance rapide des serveurs distants (x4 depuis mai 2025), soutenue par des intégrations chez OpenAI, Anthropic, AWS et des acteurs comme Figma.
- Gartner prévoit que **75% des fournisseurs de passerelles API** intégreront MCP d'ici 2026.

MCP – Cas d'usage



Accès unifié aux outils de *Delivery* (livraison)

MCP est utilisé comme un « USB-C pour l'IA ». C'est un standard pour brancher des assistants (Claude, Copilot, etc.) sur Jira/Confluence, GitHub/Git, bases SQL, Sentry, fichiers, IDE... Ceci favorise la réutilisabilité des workflows IA, quels que soient les modèles/outils.



Rétro-documentation / qualité logicielle

MCP alimente des assistants capables de croiser code source, documentation, tickets pour produire des livrables (rétro-documentation, tests, READMEs...).



Développeur augmenté dans l'IDE (Environnement de Développement Intégré)

Dans un IDE, connecter des serveurs MCP permet de faire de la recherche de code/documentation, d'interroger des bases de données, de manipuler des fichiers, de lancer des outils d'automatisation, etc.



Intégration connaissance d'entreprise

L'exposition des APIs et des bases de données via des serveurs MCP « maison » permet de mettre en œuvre des assistants génériques, dont la connaissance et les droits sont adaptés à l'entreprise.



Automatisation transverse (agents)

Avec des serveurs MCP comme « Sentry », « Slack », « Gmail », « Jira », un agent peut, sous contrôle, lire un incident, ouvrir un ticket, communiquer, interroger une base et proposer un correctif.

Outils et écosystèmes accessibles avec le MCP

Liste à janvier 2026



Exemples de Clients MCP

 **Claude**

(Desktop / Claude Code)



+ Autres IDE

 Microsoft Copilot Studio

 Google Workspace



Serveurs MCP

(services exposés par le serveur MCP)

 Playwright

 git

 GitHub

 perplexity

 aws

 Jenkins

 Open Street Map

 CLI

 Context7

 Firebase

 ATLASSIAN

 Robot Framework

 TIME

 Figma

 Jira

 Filesystem

Une bibliothèque partagée de serveurs MCP a été réalisée par le GT et est mise à disposition en annexe

Prérequis de mise en œuvre des MCP (organisationnels & techniques)



GOUVERNANCE & PÉRIMÈTRE

Définir des cas d'usage prioritaires
(ex. Retro-doc, Testing, Design-to-Code, Vibe Coding).

Nommer des référents/architectes côté IT et métiers pour valider sources et droits.



SÉCURITÉ & CONFORMITÉ

Définir l'identité et les habilitations :
OAuth2/OIDC, scopes minimaux, secrets hors du modèle, consentement explicite utilisateur.

Effectuer l'isolation des serveurs
(conteneurs, réseaux segmentés), logs/audit.



PRÉPARATION TECHNIQUE

Établir un catalogue interne des serveurs approuvés (versions, dépendances, périmètres). Github MCP Registry : liste de sources fiables pour les serveurs MCP.

Intégrer le MCP à la chaîne d'intégration et de développement continu (CI/CD) : lint/sécurité, signatures, SBOM, tests d'intégration (prompt injection/poisoning).

MCP – Risques & contrôles : ce qu’il faut absolument cadrer

Risques

Prompt injection & tool poisoning (empoisonnement des outils)

Les contenus externes (web, wiki, tickets) peuvent injecter des instructions cachées dans les outils/ressources, menant à une exfiltration ou à des actions non désirées. Un serveur MCP mal cloisonné peut ouvrir un nouvel angle d’attaque (accès fichiers, DB, cloud).

Chaîne d’approvisionnement

Des paquets MCP non audités (GitHub) peuvent contenir des failles RCE (*Remote Code Execution*) ou portes dérobées.

Autorisation/consentement

Le flux d’autorisation était un point faible, des progrès récents sont cependant constatés : URL Elicitation / OAuth flows qui renforcent l’authentification côté MCP.

Maturité du standard (*évolutions rapides*)

Le protocole évolue (sampling, workflows longue durée, autorisations simplifiées). À long terme, MCP a le risque d’être trop « couteau suisse », et donc d’être coûteux ou non rentable (car il faut trouver le bon outil).

Multiplication des MCP

Le nombre croissant de serveurs entraîne un *over engineering* non nécessaire dans certains cas. Ceci pose un risque sur les performances et la qualité lors de l’utilisation en cascade et crée par ailleurs un risque de complexité/boite noire.

Bonnes pratiques

▶ Valider strictement schémas, filtrage, nonces/signatures, *human-in-the-loop* pour opérations sensibles.
Imposer least-privilege, segmentation réseau, comptes techniques à durée de vie courte, journaux et alerting.

▶ Exiger des revues de code, signatures, SBOM, et idéalement fork interne durci + registry privé.

▶ URL Elicitation / OAuth flows sont à adopter là où ils sont disponibles et à intégrer aux politiques de fournisseurs d’identité (IdP).

▶ Tenir une veille et verrouiller les versions dans les déploiements.

04

Conclusion et perspectives

Conclusion : plusieurs interrogations à considérer pour 2026



Si les changements sont durables, ceux-ci **restent émergents**, et laissent un certain nombre de **questions ouvertes**

Rôles et compétences

Quels **rôles seront associés au développement logiciel** de demain ?

Le Vibe Coder deviendra-t-il le **développeur augmenté de demain** ?

Méthodologie / Processus

Comment modifier nos **processus** pour appréhender ces technologies ?
Condition essentielle pour en tirer des bénéfices !

Impacts applicatifs et technologiques

Quel avenir pour les **solutions de No Code / Low Code** par rapport au Vibe Coding ?

Le Vibe Coding permet-il de développer en interne une alternative à des solutions de **gros fournisseurs de SaaS** ?

Modèles économiques

Quel sera le **coût du Vibe Coding** demain sachant que les nouveaux modèles coûtent de plus en plus cher ?

Les **sociétés de services** (notamment **offshore** ou **centrées sur la seule production de code**) sont-elles viables désormais ?

05

Annexe Bibliothèque partagée de serveurs MCP

Avant-propos

Cette bibliothèque a été constituée à partir des serveurs MCP utilisés par les membres du Groupe de Travail « **Vibe Coding / Serveurs MCP** » du Do Tank sur les impacts de l'IA Générative sur les activités d'ingénierie logicielle.

Il ne s'agit pas d'un inventaire exhaustif, ni d'une évaluation objective de chaque serveur.

La dernière mise à jour de ce radar a été effectuée **début janvier 2026.**

Bibliothèque partagée



- Permet d'interagir avec les projets Angular via la CLI : génération de composants, services et modules, exécution de builds, gestion de la configuration et automatisation des tâches de développement front-end.

[Lien vers la documentation](#)



- Fournit des outils pour Confluence et Jira (gestion des pages, commentaires, informations sur les utilisateurs, etc.)

[Lien vers la documentation](#)



- Donne un accès contrôlé aux services AWS (stockage, calcul, bases de données, IAM, déploiement), facilitant l'automatisation des opérations cloud et l'orchestration d'infrastructures.

[Lien vers la documentation](#)

Bibliothèque partagée



Azure

- Permet d'interagir avec les ressources Azure (machines virtuelles, fonctions serverless, stockage, bases de données, réseaux), afin d'automatiser la gestion d'infrastructures et de services cloud.

[Lien vers la documentation](#)



Context7

- Fournit un accès structuré à de la documentation technique, des extraits de code et des références issues de multiples sources, afin d'améliorer la précision des réponses techniques et le contexte de développement.

[Lien vers la documentation](#)



- Permet d'interagir avec des fichiers de design Figma : lecture de composants, extraction de styles, génération ou modification d'éléments d'interface, facilitant la collaboration entre design et développement.

[Lien vers la documentation](#)

Bibliothèque partagée



- Fournit des outils pour les opérations sur le système de fichiers : création, lecture, modification et suppression de dossiers et de fichiers locaux, permettant d'interagir directement avec l'environnement de stockage.

[Lien vers la documentation](#)



firebase

- Donne la capacité d'interagir avec les services Firebase (authentification, bases de données temps réel, Firestore, hébergement, fonctions cloud), afin d'automatiser le développement et l'exploitation d'applications.

[Lien vers la documentation](#)



- Fournit des outils permettant de mieux comprendre le code dans son contexte

[Lien vers la documentation](#)

Bibliothèque partagée



- Permet de lire les *repositories* et les fichiers de code, de gérer les problèmes et les *Pull Requests*, d'analyser le code et d'automatiser les workflows, grâce au langage naturel

[Lien vers la documentation](#)



- Permet de rechercher, créer, mettre à jour et gérer des tickets Jira sans navigation manuelle

[Lien vers la documentation](#)



- Fournit des outils pour Jenkins (gestion des tâches, informations de compilation, intégration SCM, etc.)

[Lien vers la documentation](#)

Bibliothèque partagée



- Donne accès à des services basés sur la localisation et des données géospatiales.

[Lien vers la documentation](#)



- Connecte aux fonctionnalités de Perplexity (recherche Web en temps réel, IA conversationnelle et raisonnement avancé)

[Lien vers la documentation](#)



- Permet d'interagir avec les pages via des snapshots d'accessibilité générées automatiquement (représentations structurées de la page)

[Lien vers la documentation](#)

Bibliothèque partagée



- Fournit des informations sur l'heure actuelle et les conversions de fuseau horaire

[Lien vers la documentation](#)



**Robot
Framework**

- Permet aux assistants IA d'interagir avec des suites de tests Robot Framework : lecture, génération et exécution de scénarios de test automatisés, ainsi que l'analyse des résultats de validation logicielle.

[Lien vers la documentation](#)