

LIVRE BLANC · ADTECH & IA AGENTIQUE

IA agentique & achat média

Le guide pour investir

Où investir, quoi construire, quoi standardiser - et dans quoi ne surtout pas investir.

Sommaire

Introduction	2
Le vocabulaire de départ	2
Ce que couvre ce guide	3
Partie 1 - Maîtriser la connectivité avec le standard MCP	4
1.1 Se connecter à un système qui possède un connecteur MCP	4
1.2 Envelopper son propre écosystème avec un serveur MCP	6
1.3 Précision clé : usage interne et usage externe	7
Partie 2 - MCP et protocoles du marché : la question de l'arbitrage	8
2.1 Brancher des agents « maison » sur AAMP ou AdCP	8
Partie 3 - L'arbitrage stratégique : agents maison ou agents protocolaires	10
3.1 Cas n° 1 - Agents de transaction externe (buyer / seller agents)	10
3.2 Cas n° 2 - Agents à usage purement interne	11
Partie 4 - Comment on construit les agents aujourd'hui	13
4.1 La fin des agents uniques et isolés	13
4.2 Le développement des systèmes agentiques	13
4.3 La baisse des coûts de développement	14
4.4 Le nouveau rôle du développement	15
4.5 Rappel : arbitrage « agent maison » vs agent protocolaire	16
Partie 5 - Les autres problématiques d'investissement	17
5.1 Investir face au risque d'obsolescence	17
5.2 La règle d'or : ne pas sur-ingénierer	18
Partie 6 - Piloter : coûts, preuve, contrôle	20
6.1 Le coût réel : ce qui baisse et ce qui monte	20
6.2 L'évaluation de la performance : objective et récurrente	20
6.3 L'auditabilité	21
6.4 Les garde-fous	21
Conclusion - Ce qu'il faut retenir	24
À propos d'Epicflare	25

POURQUOI CE DOCUMENT

Introduction

L'IA agentique est en train de rebattre les cartes de l'achat média. Mais entre agents maison, MCP, AdCP, AAMP, frameworks agentiques et solutions propriétaires, les choix technologiques se multiplient - et les mauvaises décisions peuvent rapidement coûter cher.

Faut-il développer ses propres agents ou s'appuyer sur les nouveaux standards du marché ? Quand le MCP est-il réellement utile ? Dans quels cas adopter un protocole devient-il incontournable ? Et surtout, où investir aujourd'hui sans construire une architecture qui sera obsolète dans douze mois ?

Ce guide propose une grille de lecture concrète pour faire les bons arbitrages. Il décrypte les architectures qui s'imposent, les coûts qui baissent - et ceux qui augmentent -, les pièges du sur-engineering et les nouveaux impératifs de performance, d'auditabilité et de contrôle.

Le vocabulaire de départ

Quatre notions reviennent régulièrement dans les pages qui suivent. Les préciser dès à présent évite toute confusion.

Agent IA	Un logiciel autonome capable d'effectuer des actions variées (analyse, tâches opérationnelles, etc.) et de façon adaptables. Il bénéficie des capacités d'un LLM combinées à une couche software propriétaire. Un agent IA possède une autonomie décisionnelle : il planifie, exécute, interagit avec des plateformes externes et itère jusqu'à la résolution de l'objectif fixé.
Système agentique	Une cohorte d'agents sous-spécialisés qui collaborent, orchestrés par un agent maître. Par opposition à un Agent IA traditionnel qui endosse l'intégralité d'une tâche, un système agentique est composé d'une multitude d'agents collaborant et travaillant en parallèle pour l'atteinte d'un même objectif.
MCP	Un connecteur, comparable à des API nouvelles générations s'appuyant sur les capacités des IA génératives. Un expose les fonctionnalités d'un système (e.g. SaaS, CRM, Database, etc.) sous forme d'outils qu'un modèle ou un agent IA peut appeler directement, sans intégration sur mesure pour chaque plateforme.
Protocole métier	Principalement représentés AAMP et AdCP. Protocoles agentiques open source dont l'objectif est de permettre l'automatisation des opérations de la publicité digitale. Chaque protocole dispose d'une ossature technique préconçue ainsi que d'un vocabulaire normé, des schémas d'échange, une spécialisation aux pratiques de l'industrie intégrée à la racine du protocole. AAMP est développé par l'IAB Tech. Lab tandis qu'AdCP est développé par l'Agent Adversiting Organization.

Ce que couvre ce guide

- **Partie 1 - Maîtriser la connectivité avec le standard MCP.** Quand et comment se brancher sur un système tiers via MCP, et comment ouvrir le sien.
- **Partie 2 - MCP et protocoles du marché.** Pourquoi « MCP ou protocole ? » n'est pas un arbitrage, et dans quels cas brancher ses propres agents sur AAMP ou AdCP.
- **Partie 3 - L'arbitrage stratégique.** La question qui tranche entre développer ses propres agents et s'appuyer sur ceux des protocoles.
- **Partie 4 - Comment on construit les agents aujourd'hui.** La fin de l'agent isolé, la montée des systèmes agentiques et l'effondrement des coûts de développement.
- **Partie 5 - Les autres problématiques d'investissement.** Investir face au risque d'obsolescence, et la règle d'or pour contrôler le sur-engineering.
- **Partie 6 - Piloter : coûts, preuve, contrôle.** Ce qui baisse et ce qui monte, l'évaluation de la performance, l'auditabilité et les garde-fous.

Le document se termine sur une synthèse en huit points à retenir.

PARTIE 1

Maîtriser la connectivité avec le standard MCP

Le MCP (Model Context Protocol) est un système de connecteur standardisé qui permet de brancher des systèmes IA indépendants entre eux. Son usage se divise en deux approches symétriques : se connecter au système d'un tiers, ou ouvrir le sien.

Techniquement, le MCP fonctionne comme une couche d'API pensée nativement pour l'IA : il expose les fonctionnalités d'un système sous forme d'outils qu'un modèle ou un agent IA peut appeler directement, sans intégration sur mesure pour chaque plateforme - une sorte d'API 2.0, consommée par de l'IA plutôt que par du code écrit à la main.

AVANT DE COMMENCER

Deux points de vigilance

La plupart des serveurs MCP déployés aujourd'hui n'exposent que des fonctionnalités de **lecture** (extraction de données, reporting). L'**écriture** - lancer une campagne, modifier un budget, mettre en pause un ad set - reste minoritaire, pour des raisons de sécurité et de contrôle des risques. Elle se généralisera à mesure que la confiance dans l'autonomie des agents augmentera (voir section 6.4).

Ce même jeu de contrôle rend risqué le branchement direct d'assistants IA ou d'agents sur les API de plateformes tierces sans passer par les canaux officiels : depuis le début 2026, plusieurs comptes publicitaires connectés à des outils IA non autorisés sur Meta Ads ont été suspendus pour usage jugé non conforme aux conditions d'utilisation de la plateforme.

1.1 Se connecter à un système qui possède un connecteur MCP

Lorsque nous voulons interagir avec une plateforme tierce, deux méthodes existent. Elles ne se distinguent que par le degré d'automatisation de l'interaction.

Méthode	Fonctionnement et caractéristiques
Connexion par Chat IA (ChatGPT, Claude, etc.)	L'utilisateur ajoute l'URL MCP de la plateforme tierce comme connecteur dans son chat. Le chat permet ensuite d'interroger la plateforme tierce en langage naturel. L'échange se passe donc comme un chat classique « prompt → réponse » avec la plateforme tierce.
Connexion par Agents IA (système à système)	Des agents sont branchés directement sur le système tiers via le MCP. Ce sont donc les agents (auxquels on a préalablement attribué un objectif spécifique) qui vont interagir avec le système tiers au lieu de l'utilisateur par chat.

Méthode	Fonctionnement et caractéristiques
	Au même titre qu'un utilisateur dans un chat qui aurait effectué une série de questions x réponses avec le MCP tiers, l'agent enchaîne une séquence complexe de dialogues, d'interrogations et d'actions jusqu'au bout mais sans intervention humaine. C'est une exécution complète et dynamique, dont l'output final sera résumé par les agents.

L'avantage commun aux deux méthodes

- Pour un utilisateur, plus besoin de naviguer dans une UI et de réaliser différentes tâches pour obtenir des réponses (e.g. ouvrir une campagne, extraire un rapport, l'analyse)
- Pour un développeur ou tout système automatisé, plus besoin de faire d'appel API classique à la plateforme pour obtenir des informations, ceci nécessitant non seulement de créer un script dédié mais aussi de retraiter les informations brutes reçues.
- Dans tous les cas, l'interrogation se fait « naturellement », les informations clés sont trouvées automatiquement, et des actions peuvent également être effectuées.

En pratique, connecter un chat IA à un MCP tiers revient à récupérer une URL de type <https://mcp.plateforme.com/mcp> et à la renseigner dans la section « connecteurs » ou « plugins » du chat (e.g. ChatGPT, Claude, etc.).

Ajouter un connecteur personnalisé

Connectez Claude à vos données et outils. [En savoir plus sur les connecteurs](#) ou commencez avec [ceux prêts à l'emploi](#).

GoogleAds_MCP

Affiché dans la liste des connecteurs.

<https://google-ads.mcp.pipeboard.co>

L'adresse HTTPS où le serveur accepte les requêtes MCP, par exemple <https://mcp.example.com/mcp>.

^ Paramètres avancés

abcd1234

N'utilisez que des connecteurs provenant de développeurs en qui vous avez confiance. Anthropic ne contrôle pas les outils que les développeurs mettent à disposition et ne peut pas garantir qu'ils fonctionneront comme prévu ou qu'ils ne changeront pas.

Vous développez un serveur MCP ? [Signalez les problèmes et abonnez-vous aux mises à jour ici](#)

Annuler
Ajouter

Figure 1 - Un exemple de connexion par chat IA : déclaration du serveur MCP tiers comme connecteur.

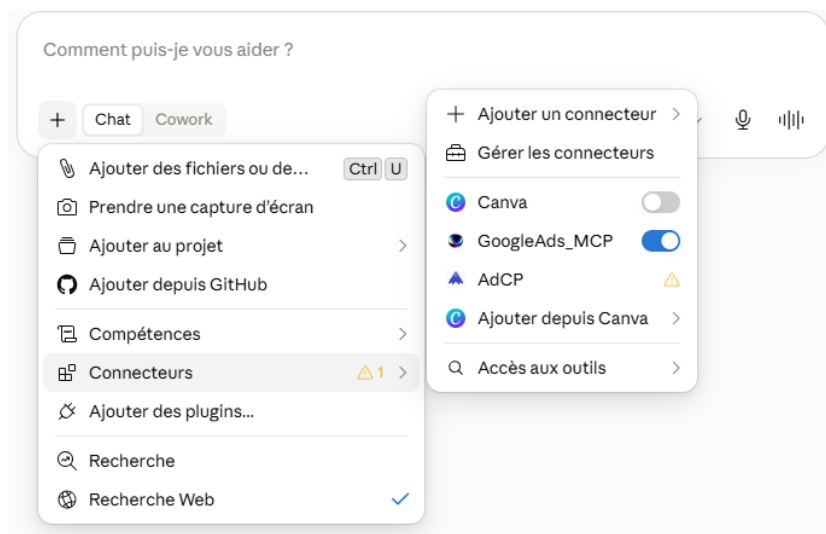


Figure 2 - Le connecteur devient ensuite accessible directement depuis la conversation.

1.2 Envelopper son propre écosystème avec un serveur MCP

La logique symétrique consiste à déployer un serveur MCP autour de sa propre technologie pour autoriser des tiers à s'y connecter de façon « IA friendly ». Deux cas d'usage justifient cet investissement :

- **Cas A - Upgrader une plateforme traditionnelle.** Une entreprise possède une technologie classique (accès par API / UI) sans capacités IA ou agentiques natives. L'envelopper dans un MCP permet à des utilisateurs externes ou internes de l'interroger et de l'utiliser via leurs propres chats IA ou leurs agents. C'est donc un moyen de rendre « AI friendly » un système qui ne possède historiquement pas de fonctionnalités IA.
- **Cas B - Donner accès à une solution agentique.** Une entreprise possède un système d'agents qu'elle souhaite ouvrir à des interactions tierces. Ce n'était pas le cas d'usage principal pensé pour le MCP (des protocoles agent-to-agent avaient historiquement été pensés pour permettre d'accéder à des agents tiers) mais c'est devenu la méthode la plus courante et la plus efficace pour ouvrir ses agents au monde extérieur car les MCP fonctionnent comme un connecteur tout terrain et s'appliquent donc très bien aux agents IA comme aux plateformes traditionnelles.

ÉCLAIRAGE

Pourquoi le MCP fonctionne dans tous les cas

L'A2A (Agent2Agent, protocole initié par Google puis transféré à la Linux Foundation) a été conçu pour la relation entre agents ; le MCP, pour la relation aux systèmes.

L'usage a tranché autrement : l'adoption massive du MCP a produit un effet de réseau, et l'industrie s'est mise à l'utiliser pour tout, de par son efficacité et sa facilité de déploiement.

1.3 Précision clé : usage interne et usage externe

La notion de « tiers » ou de « systèmes indépendants » ne se limite pas aux interactions entre deux entreprises distinctes. La connexion par MCP est tout aussi pertinente en interne : elle permet de relier des plateformes propriétaires et des agents développés dans des univers techniques différents au sein d'une même organisation.

ILLUSTRATION

Exemple : un DSP interne développé par une agence et un outil de reporting propriétaire, construits par deux équipes différentes à des moments différents, peuvent être connectés via MCP sans réécrire d'intégration dédiée entre eux.

PARTIE 2

MCP et protocoles du marché : la question de l'arbitrage

Une question revient systématiquement : faut-il choisir entre équiper son système d'un connecteur MCP et / ou adopter un protocole du marché comme AAMP ou AdCP ?

La réponse est non. Ce sont deux outils de nature totalement différente, ce n'est pas un arbitrage direct. S'il existe un arbitrage, alors celui-ci se manifesterait dans des cas beaucoup plus subtils et avancés, mais dans la majorité des cas cette problématique ne se posera pas.

Le MCP	C'est un connecteur. Il assure le branchement entre les systèmes.
AAMP et AdCP	Ce sont des protocoles agentiques open source dont l'objectif est de permettre l'automatisation des tâches opérationnelles du marketing digital. Ils disposent d'agents préalablement codés et enrichis d'un vocabulaire normé, des schémas d'échange et de divers spécialisations aux pratiques de l'industrie intégrées à la racine du protocole. Ils exécutent des tâches spécifiques comme rechercher et analyser des sources d'inventaire pour le média planning, négocier des deal IDs ou encore implémenter automatiquement des campagnes. AdCP est porté par un consortium mené par Scope3, avec la participation d'éditeurs et de DSP ; AAMP est porté par l'IAB Tech Lab, l'organisme de standardisation historique du secteur.

Créer un serveur MCP n'exclut pas l'adoption d'AAMP ou d'AdCP. Au contraire AAMP et AdCP disposent eux-mêmes de connecteurs MCP :

- On peut se connecter et interagir avec AdCP et AAMP en se connectant à leur MCP
- On peut plugger AdCP et AAMP à des systèmes tiers en utilisant un connecteur MCP
- On s'y connecte exactement de la même manière - par chat ou par agents.

2.1 Brancher des agents « maison » sur AAMP ou AdCP : faisabilité et limites

Il est techniquement possible de se connecter par chat ou de brancher ses propres « agents maison » - qui ne suivent pas nativement les protocoles - sur ces univers protocolaires (AAMP & AdCP) via un connecteur MCP.

Pourquoi brancher un chat ou des agents aux protocoles ?

Cas 1 - Le pilotage en langage naturel

Je veux piloter le travail effectué par le protocole (e.g. découverte d'inventaire, négociation, implémentation de campagne) en langage naturel et sans avoir à recréer une interface

dédiée. Je vais alors connecter mon ChatIA au protocole pour le gérer et lui fournir des instructions.

Cas 2 - L'automatisation et l'enrichissement du pilotage

Une autre possibilité est de placer un agent en amont, qui s'interface directement avec le protocole plutôt que de réaliser les actions manuellement par chat. Se pose alors la question de la valeur ajoutée réelle de cet agent.

Dans la plupart des cas, elle demeure faible. Si l'agent se contente de relayer les requêtes vers le protocole, on ajoute simplement une étape qui ajoute de la latence, du coût en tokens et un point de maintenance supplémentaire. Un chat connecté fait exactement la même chose gratuitement.

La valeur ajoutée existe dans un cas précis : quand l'agent qui intervient avant le protocole y injecte une expertise ou des données qui n'existent pas dans le protocole lui-même ; alors l'agent pluggé en amont devient complémentaire et apporte une forme d'enrichissement.

L'exemple le plus évident est le croisement des informations remontées par le protocole avec une expertise métier et / ou des données propriétaires, puis l'application de règles de décision propres à l'agence ou à l'annonceur. Le protocole peut par exemple effectuer un cycle de tâche (e.g. découverte d'audience / inventaire avec le planning associé) qui nécessite d'être réévalué ou complété à partir des « best practices » de l'agence.

CRITÈRE DE DÉCISION

Chat ou connexion par agent ?

Mon agent apporte-t-il quelque chose que le protocole ne fait pas ou pour lequel il est moins compétent ?

- Si oui, l'agent se justifie.
- Si non, ce n'est qu'un intermédiaire supplémentaire avec une valeur ajoutée limitée.

Est-ce viable de brancher des agents aux protocoles ?

L'efficacité d'une telle connexion doit être testée avec vigilance, en évaluant l'existence d'obstacles au cas par cas.

L'univers protocolaire est fortement normé (champs obligatoires, taxonomies imposées, formats attendus) tandis qu'un agent maison ne l'est pas ; ce qui entraîne des frictions pour la bonne collaboration entre un système maison et un système protocolaire :

- Un appel non conforme de l'agent maison vers le protocole sera rejeté
- Tandis qu'un appel conforme mais imprécis (e.g. tous les champs sont remplis mais avec des erreurs) entraîne une baisse de performance et de qualité significative et favorise les hallucinations.

Par ailleurs, en admettant que les schémas du protocole sont bien respectés, il faut systématiquement vérifier que le protocole sera en mesure de recevoir et adapter son travail en fonction de ce que l'agent maison va lui soumettre, le risque étant que ces informations ne soient jamais véritablement exploitées car le protocole ne disposerait pas de la flexibilité nécessaire pour les utiliser.

PARTIE 3

L'arbitrage stratégique : agents maison ou agents protocolaires

Il y a un véritable choix stratégique entre développer ses propres agents et s'appuyer sur ceux des protocoles. Ce choix dépend avant tout du cas d'usage considéré puis du temps et du budget alloué au développement d'un nouveau système. Une seule question suffit à trancher un premier grand groupe de scénarios.

CRITÈRE DE DÉCISION

Le critère de décision

Les agents ont-ils vocation à transacter avec des agents extérieurs à l'entreprise ?

- **SI OUI** → Protocole obligatoire.
- **SI NON** → Un arbitrage entre une solution interne et une solution protocolaire est pertinent.

3.1 Cas n° 1 - Agents de transaction externe (Buyer / Seller agents)

Contexte. On veut créer des agents destinés à échanger avec le monde extérieur pour découvrir de l'inventaire / audience, négocier et réaliser des transactions. Il faut donc faire dialoguer les agences et annonceurs avec les médias et propriétaires d'inventaire.

Recommandation. Utiliser directement les protocoles (AAMP ou AdCP).

Pourquoi. Ces protocoles ont été normés précisément pour que les buyer agents et les seller agents discutent selon des règles communes. Sortir de la norme, c'est se priver de la capacité à discuter efficacement. Cela reviendrait à mettre des systèmes qui parlent des langages sensiblement différents, entraînant une augmentation du nombre d'erreurs, d'hallucination et une baisse de la performance (voir section 2.1).

Une règle simple. Quand on veut réaliser un échange d'information entre différentes entités, pour exécuter des transactions, l'ensemble de la chaîne de valeur doit suivre les mêmes règles rigoureuses.

ÉCLAIRAGE

L'historique protocolaire de l'industrie et l'analogie OpenRTB

L'exemple du programmatique : Personne ne crée son propre protocole d'achat média en temps réel. On suit OpenRTB parce que la norme a été conçue pour optimiser les transactions entre DSP et SSP. Avoir un système de transaction sur mesure qui ne suit pas l'OpenRTB mais qui tenterait de s'y greffer pour opérer des transactions programmatiques apporterait des challenges techniques et opérationnels supplémentaires.

Le fondement des protocoles : la valeur d'un protocole est proportionnelle à sa symétrie d'utilisation. Si toutes les parties prenantes l'utilisent, alors chacun peut en tirer les bénéfices. Si certains l'utilisent tandis que d'autres viennent avec une « variante », alors on ajoute en complexité et on augmente le risque de défaillance sans en exploiter tous les bénéfices.

3.2 Cas n° 2 - Agents à usage purement interne

Contexte. Les agents n'ont pas vocation à transacter avec l'extérieur (i.e. pas de Buyer / Seller Agent). Ils ont plutôt un rôle en interne, comme faire gagner en productivité et efficacité opérationnelle : agent de data analyse, agent d'audit de campagne, agent d'implémentation de campagne, etc.

NOTE

Un agent interne peut parfaitement se connecter à un système externe comme un SSP ou un DSP. Le terme « interne » est ici utilisé pour souligner que cet agent n'a pas pour but de « parler » aux buyer et seller agents des protocoles.

Dans ce cas, l'arbitrage porte sur l'ossature technique de départ : construire hors protocole, ou récupérer l'ossature agentique open source fournie par AAMP ou AdCP.

Option A - Agent hors protocole (custom)	Option B - Agent basé sur l'ossature protocolaire
Principe. Construire sa propre solution agentique sans reprendre l'ossature normée d'AAMP ou d'AdCP. Avantages. Customisation totale. L'agent reflète exactement les pratiques internes de l'agence ou de l'annonceur, intègre les compétences clés spécifiques à l'entité, son expertise et sa valeur ajoutée. Potentiellement bien plus performant sur son cas d'usage précis.	Principe. Réutiliser l'ossature agentique pré-codée fournie en open source par les protocoles. Avantages. Réduction considérable du coût de développement initial. Spécialisation adtech déjà en place. Inconvénients. Le système est normé et n'est pas conçu pour être customisé à souhait. Vigilance technique. Il faut disposer d'une maîtrise suffisante pour « couper » les éléments

Option A - Agent hors protocole (custom)	Option B - Agent basé sur l'ossature protocolaire
Inconvénients. Temps de développement plus long et coût plus élevé.	du protocole qui n'ont pas vocation à être utilisés (e.g. désactiver les buyer agents), activer les agents uniquement utiles au use case en question (e.g. agent d'implémentation de campagne) et réalimenter ce système isolé de son architecture originelle.

CRITÈRE DE DÉCISION

Les trois questions qui tranchent entre A et B

Question 1 - Ma valeur ajoutée réside-t-elle dans une manière de travailler qui m'est propre ?

Autrement dit : mes performances viennent-elles de pratiques, de méthodes ou d'une expertise que mes concurrents n'ont pas ?

- **Si oui** → L'option A est la seule qui préserve cette valeur. Une ossature normée vous fera converger vers les pratiques standards du secteur, c'est-à-dire vers celles de tout le monde.

Question 2 - Le coût et le délai de développement sont-ils un point de blocage réel ?

- **Si non** → La problématique doit être décidée à partir des questions 1 et 3.
- **Si oui** → L'arbitrage entre A et B doit être évalué spécifiquement sans présupposer que B sera systématiquement moins cher. L'ossature protocolaire fait gagner du temps sur le socle, mais l'écart réel du coût / temps de développement dépend de facteurs plus complexes (voir ci-après).

Question 3 - Suis-je prêt à ce que l'évolution de mes agents dépende de celle du protocole ?

En choisissant l'option B, on hérite d'une feuille de route qu'on ne maîtrise plus : les évolutions, les dépréciations et les changements de spécification s'imposeront, au rythme et selon les priorités de l'organisation qui produit le protocole.

- **Si non** → L'option B n'est pas viable, quel que soit le gain de développement immédiat.

La question 1 tranche sur la stratégie, la 2 sur les moyens, la 3 est un veto - elle peut invalider l'option B même quand les deux premières y menaient.

PARTIE 4

Comment on construit les agents aujourd'hui

Le paradigme a connu un bouleversement complet en douze mois, sur le plan technique comme financier. Ce qui était vrai l'an dernier ne l'est plus, et beaucoup de décisions d'investissement reposent encore sur des références périmées.

4.1 La fin des agents uniques et isolés

La première vague a vu émerger de nombreux agents considérés comme techniquement « monolithiques ». Ces agents monolithiques étaient extrêmement simplistes, se contentant d'employer un modèle LLM dotés de deux ou trois capacités d'interaction avec des plateformes tierces. Il était d'ailleurs courant de voir bon nombre de ces agents créés directement à partir de solutions « no-code » compte tenu du faible niveau de sophistication. Le constat est sans appel - peu coûteux, certes, mais leur capacité de raisonnement, leur fiabilité et leur niveau de customisation métier restaient très limités.

Ces agents uniques sont aujourd'hui moins populaires et ne sont employés que dans des cas opérationnels bien spécifiques, généralement assez simples. Leur performance demeure assez rudimentaire ; cette fiabilité limitée réduit considérablement les cas d'usage et la volonté de déploiement de ces solutions.

4.2 Le développement des systèmes agentiques

On parle de plus en plus de systèmes agentiques, par opposition aux agents uniques. On abandonne l'agent monolithique et on le remplace par une cohorte d'agents sous-spécialisés qui collaborent et qui sont orchestrés par un agent maître.

Ces systèmes adressent progressivement les problématiques et les craintes rencontrées jusqu'ici - erreurs fréquentes, manque d'expertise, fiabilité douteuse - grâce à quatre mécanismes :

- **L'injection d'expertise pointue.** Ils permettent d'intégrer des connaissances métier très spécifiques et répliquent fidèlement l'expertise opérationnelle.
- **La fragmentation des tâches.** Un travail complexe est découpé et attribué à des agents spécialisés qui collaborent, augmentant la précision.
- **Les boucles d'itération et l'auto-évaluation.** Le système réévalue son travail en permanence et itère jusqu'à une solution stable, afin de rendre les résultats produits plus fiables.
- **Le « challenge » des outputs.** Le système produit plusieurs résultats valides, les compare et sélectionne celui qui a le plus de valeur.

Ces nouvelles architectures, plus profondes et plus « denses » augmentent significativement la qualité des outputs produits, permettant à la fois de répliquer des expertises métier pointues et d'offrir une qualité d'output / minimiser le risque d'erreur d'une façon considérable.

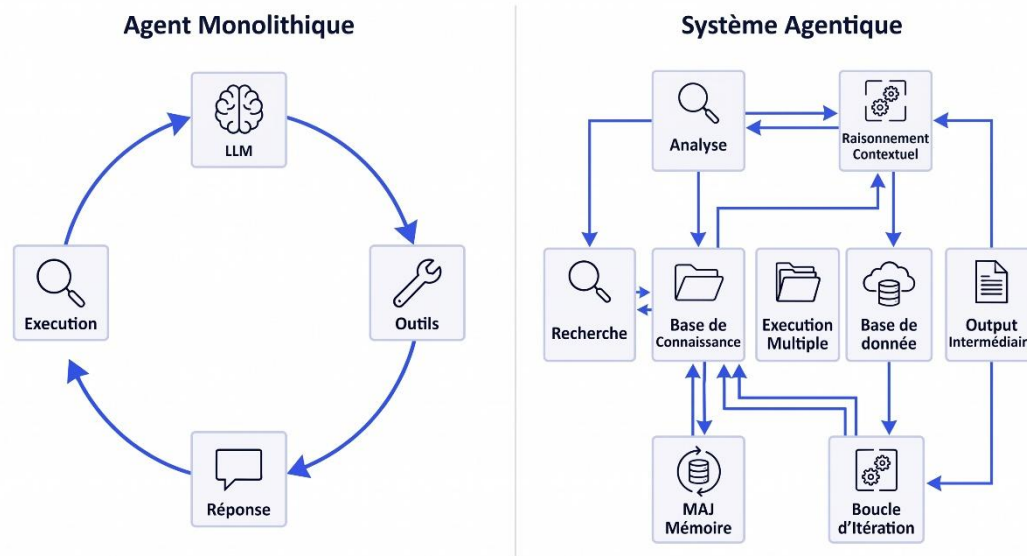


Figure 1 - Un exemple comparatif des capacités et du contenu d'exécution entre un agent monolithique et un système agentique actuel

4.3 La baisse des coûts de développement

Avant : le coût significatif, et pourquoi

Il y a douze mois, un agent d'une performance élevée exigeait des temps de développement longs (un trimestre minimum, pouvant aller jusqu'à 6-9 mois pour les cas les plus complexes) et des budgets se comptant en plusieurs centaines de milliers d'euros.

Ce coût s'expliquait principalement par la nécessité de créer une architecture agentique complète afin de maximiser la performance de l'agent en question. Ces architectures étaient d'ailleurs les prémices des systèmes agentiques actuels car elles avaient pour but de permettre l'injection d'expertise, l'itération et l'évaluation des outputs, etc.

NOTE

À ne pas confondre avec un agent no code d'entrée de gamme : l'explication proposée concernant le coût de développement porte sur des agents considérés comme « entreprise grade », dont le fonctionnement s'inspirait des systèmes agentiques actuels et d'une qualité incomparable aux agents no code considérés comme « low cost ».

Aujourd'hui : un abaissement des coûts et du temps de développement

L'évolution principale permettant de réduire les coûts de développement des agents et plus particulièrement d'un système agentique complet vient de l'émergence de frameworks agentiques matures. Ces frameworks offrent une base technique pour la mise en place de

l'architecture de collaboration entre agents ainsi que la logique de vérification et de boucles d'itération - c'est-à-dire précisément la vraie difficulté technique.

Ces frameworks sont aujourd'hui matures et largement adoptés : LangGraph, CrewAI, AutoGen en sont des exemples courants.

Pour construire des agents « entreprise grade », il n'est donc plus nécessaire de bâtir des solutions de zéro : les frameworks accélèrent largement le temps de développement, en offrant un socle technique pour le système agentique ce qui explique la baisse des coûts associés. A la place le travail se focalise beaucoup plus sur l'intégration des pratiques métiers et de la spécialisation du système agentique à des problématiques opérationnelles.

L'ORDRE DE GRANDEUR

Il est aujourd'hui possible d'obtenir un système agentique performant en moins d'un trimestre, et de diviser le temps de développement et le budget par trois ou quatre (selon notre expérience, comparé à un développement partant de zéro ; varie selon le périmètre).

4.4 Le nouveau rôle du développement

Si le framework offre une base technique forte, alors le travail de conception se place sur d'autres problématiques :

- L'hyperspécialisation des agents sur le métier de l'entreprise.
- La customisation avec des outils utilisés par les agents et spécifiques aux métiers.
- La connexion aux plateformes tierces (e.g. DSP, ad server, etc.).
- La création d'un « cerveau » de connaissances et le bon usage de ces connaissances par les agents.

Ce que je dois faire et ne plus faire pour construire mes solutions IA

L'expertise et le profil des développeurs évoluent. Il est moins nécessaire de faire appel à un développeur généraliste ou à des prestataires IA classiques. À la place, il est impératif de favoriser le développement par des experts techniques dotés d'une connaissance profonde de votre métier.

L'expertise ne s'ajoute plus après coup, elle doit être présente dès la conception. Le point de l'expertise métier dans le développement de la solution est encore plus fondamental qu'auparavant, parce qu'un système agentique spécialisé sur un métier ne se construit pas comme un logiciel traditionnel.

Le schéma de développement traditionnel repose sur une chaîne de passation

L'expert métier décrit son challenge / use case. Le product manager le traduit en fonctionnalités et en interfaces. Le développeur code l'outil. Chaque maillon reformule ce que lui a transmis le précédent, avec le risque bien connu de l'effet « passe-plat » : l'expertise métier se déforme ou se perd en route. L'outil est un contenant ; l'expertise est ce que l'utilisateur y apporte au moment de s'en servir.

Le schéma de développement de systèmes IA est « solidaire »

Dans un système agentique, cette séparation n'existe plus. L'expertise n'est pas apportée par l'utilisateur au moment de l'usage : elle est inscrite dans le système lui-même - dans l'architecture qui nourrit les agents, dans le découpage des rôles, dans les critères par lesquels ils évaluent leur propre travail. L'agent ne fournit pas un outil au media trader : il exécute le raisonnement du media trader.

La conception technique détermine la qualité de l'exécution métier, et le métier détermine la conception technique, dès la première ligne. Si cette imbrication n'est pas assumée dès le lancement du projet, le système sera sous-performant par construction - et le problème ne se corrigera pas plus tard car le problème se situe directement autour de la façon dont le système a été pensé.

4.5 Rappel : arbitrage « agent maison » vs agent protocolaire

Nous avons présenté plus haut (partie 3) la possibilité de réemploi de la base technique d'un agent protocolaire comme une alternative permettant de réduire les coûts par rapport à un agent maison développé de bout en bout. Nous avons également indiqué que cet arbitrage (étant particulièrement centré sur le coût, la maîtrise technique et la dépendance à un protocole tiers) devait s'évaluer au cas par cas.

Les sections 4.3 et 4.4 apportent des clarifications quant aux éléments nourrissant l'arbitrage. Le changement de paradigme souligné, où le développement d'un système agentique ne part plus de zéro mais de frameworks techniques désormais matures, offre également des économies substantielles de temps et d'ingénierie.

L'économie promise par l'ossature protocolaire se mesurait autrefois face à un développement intégral ; elle se mesure aujourd'hui face à un agent maison déjà construit sur un framework. L'écart se resserre et n'est plus systématiquement en faveur du protocolaire.

C'est précisément pour cette raison qu'aucune règle de décision standard ne peut s'appliquer. L'arbitrage doit être instruit au cas par cas.

PARTIE 5

Les autres problématiques d'investissement

5.1 Investir face au risque d'obsolescence

La technologie évolue si vite que le système construit aujourd'hui sera potentiellement dépassé dans six à dix-huit mois. Faut-il attendre la stabilité au risque de prendre du retard, ou se lancer au risque de perdre son investissement ?

RECOMMANDATION

Y aller tout de suite, mais de manière pragmatique.

Privilégier l'apprentissage

L'IA générative et l'agentique évoluent à une cadence élevée. Attendre, c'est accumuler un retard de connaissance qui devient rapidement difficile à combler - et ce retard porte moins sur la technologie elle-même que sur la capacité à l'implanter dans ses propres processus.

Ce que vous apprenez aujourd'hui en construisant des systèmes servira directement à construire ceux de demain. Cette valeur est difficilement quantifiable, mais elle compense largement le coût monétaire engagé : c'est elle qui vous rendra capables de saisir les opportunités suivantes.

Investir pour rester adaptable

Le risque d'obsolescence est indiscutable. Il ne justifie pas pour autant d'attendre, il nécessite en revanche de calibrer les projets, les coûts correspondants ainsi que les délais engagés. Dans ce but, quatre principes peuvent être appliqués :

- **Petit plutôt que large.** Pas de méga-projets ambitionnant la perfection ou un périmètre très étendu. Il faut favoriser des systèmes légers, ciblés sur une problématique précise.
- **Accepter l'imperfection.** Visez un ratio effort / bénéfice élevé, quitte à ce que le système ne couvre pas tous les cas. Le gain marginal pour traiter les cas imparfaits fait généralement s'effondrer ce ratio.
- **Déployer vite, itérer vite.** La vitesse de mise en production compte plus que l'exhaustivité de la version initiale.
- **Garder un œil sur le marché.** Suivre les évolutions techniques pour pouvoir pivoter au bon moment plutôt que de continuer à investir dans un système qui serait déjà sur la pente descendante.

Si la technologie est dépassée dans un an, le coût subi reste faible et maîtrisé - et vous abordez la génération suivante avec un temps d'avance plutôt qu'un projet à démanteler.

Prendre un avantage concurrentiel même temporaire

Le débat « investir ou attendre l'obsolescence » est souvent mal posé, parce qu'il se focalise sur le risque du « sunk cost » (i.e. l'argent engagé dans une solution qui pourrait être dépassée).

Le coût d'opportunité est trop souvent ignoré. Pendant que certains attendent, d'autres concurrents se lancent. Ils gagnent en efficacité opérationnelle, ou font évoluer leur offre commerciale. Dans les deux cas, le concurrent en ressort plus fort.

La véritable prise de décision se fait donc entre un coût financier limité et maîtrisé d'un côté, et un coût d'opportunité de l'autre : celui de voir un concurrent prendre des parts de marché ou gagner en influence pendant que d'autres observent.

À RETENIR

Un avantage temporaire reste un avantage. Et l'apprentissage qu'il laisse derrière lui, lui, est durable.

5.2 La règle d'or : ne pas sur-ingénierer

Toute conception agentique se heurte aux faiblesses techniques des modèles et des frameworks du moment que l'on peut définir comme des « faiblesses structurelles ». Le terme mérite d'être précisé : malgré une performance en progression rapide et déjà très élevée, il existe - et il existera toujours - des angles morts qui empêchent de construire le système parfait. Ces manques sont parfois attribuables à certains modèles LLM, d'autres aux frameworks agentiques, etc. Ces derniers varient d'ailleurs fortement d'une industrie à l'autre.

Il s'agit d'un problème structurel qu'il faut accepter.

ILLUSTRATION

Par exemple, au lancement de l'IA générative il y a trois ans, on reprochait aux modèles une fenêtre de contexte minuscule, incapable d'absorber plus de quelques paragraphes ou un petit document. D'autres critiques portaient sur le fait que les premiers modèles répondaient directement, sans raisonnement ni capacité à relier plusieurs concepts.

Ces reproches ont disparu ; d'autres les ont remplacés.

Aujourd'hui, les critiques portent davantage sur la fiabilité des agents dans des tâches longues et multi-étapes (dérive progressive, erreurs qui s'accumulent), ou sur leur difficulté à distinguer une instruction légitime d'une tentative de manipulation (*prompt injection*) lorsqu'ils traitent des contenus tiers non fiables.

Chaque génération de modèle et de système IA / agentique apporte ses propres limites, inhérentes à l'innovation qu'elle propose.

Pourquoi il ne faut pas chercher à les combler

La règle d'or est de ne jamais tenter de compenser ces manques structurels par une ingénierie complexe et des surcouches lourdes. Cette tâche incombe à ceux qui produisent les modèles fondamentaux et les grands frameworks.

Ces limites connaîtront l'un des deux destins suivants : elles seront comblées par les acteurs technologiques majeurs, ou elles deviendront simplement non pertinentes du fait de l'innovation du marché. Dans les deux cas, avoir créé une technologie pour combler certaines faiblesses n'aura plus de raison d'être.

ILLUSTRATION

- Les architectures d'extraction et de compaction de données, conçues pour faire tenir l'information dans des contextes étroits, ont perdu tout intérêt le jour où les fenêtres de contexte ont explosé.
- Les workflows multi-appels complexes, construits pour simuler un raisonnement étape par étape, sont devenus inutiles dès lors que les modèles ont intégré nativement cette capacité.

Ces reproches ont disparu ; d'autres les ont remplacés.

Dans ces deux exemples, du temps et de l'argent investis pour rien. C'est un sunk cost au sens strict (contrairement au risque d'obsolescence décrit en section 5.1).

Le piège est d'autant plus efficace qu'au moment où l'on développe la surcouche, elle semble parfaitement justifiée : elle résout un problème réel.

Notre rôle : les meilleurs utilisateurs, pas les concepteurs

C'est la distinction qui protège de cet écueil. Nous ne sommes pas concepteurs de modèles ou de framework ; nous en sommes utilisateurs - et notre ambition doit être d'en être les meilleurs et d'en délivrer la forme la plus optimisée.

Concrètement : utiliser le système tel qu'il est, l'appliquer à notre métier, y ajouter les composants spécifiques à notre industrie, y injecter notre connaissance. L'investissement doit porter sur la manière d'emballer les technologies existantes pour en faire un actif utile à notre secteur - jamais sur la tentative de résoudre ce que les grands acteurs technologiques n'ont pas encore résolu.

PARTIE 6

Piloter : coûts, preuve, contrôle

L'efficacité et la fiabilité étant désormais atteignables, les challenges se sont déplacés. Le développement, l'intégration et l'usage de systèmes agentiques nécessitent une nouvelle approche autour de ces challenges.

6.1 Le coût réel : ce qui baisse et ce qui monte

Le coût de construction s'effondre pendant que le coût d'exploitation augmente. Les systèmes agentiques sont plus puissants et plus efficaces grâce à leur capacité à consommer une plus grande quantité d'information et à faire fonctionner plusieurs agents à la fois.

La conséquence est qu'une cohorte d'agents qui itère, se challenge et vérifie dépense une quantité bien plus grande de tokens.

Le coût unitaire du token baisse régulièrement d'une génération de modèle à l'autre, mais le volume consommé augmente plus vite encore - plus de contexte, plus d'itérations, plus d'agents en parallèle - ce qui explique que la facture totale grimpe malgré la baisse du prix unitaire.

RECOMMANDATION

Bien que le coût soit une véritable préoccupation, il ne doit pas venir handicaper la performance du système qui doit être construit.

Il est fortement recommandé de favoriser d'abord la performance maximale du système, sans le brider, pour évaluer ses capacités réelles. Optimiser les coûts ensuite revient à un arbitrage coût / performance incrémental : chaque euro économisé se compare à la perte de performance qu'il entraîne, plutôt qu'à une coupe automatique.

Optimiser le budget dès le premier jour est dangereux car il pourrait significativement brider le potentiel réel du système, amenant à une conclusion - erronée - que le système n'est pas viable.

6.2 L'évaluation de la performance : objective et récurrente

Un système non déterministe ne se teste pas comme un logiciel classique. Le même input peut produire deux résultats différents, tous deux acceptables mais potentiellement de qualité différente.

Un dispositif d'évaluation de la performance est donc central pour - encore une fois - éviter toute conclusion erronée.

Ce n'est pas un sujet à traiter en profondeur ici, mais un point de vigilance à inscrire dans tout plan de projet, avec trois exigences minimales :

- **Un jeu de cas de référence.** Constitué avec l'expert métier, couvrant les situations normales et les cas limites, permettant d'évaluer de façon systématique la performance du système (e.g. après une mise à jour technique, pour vérifier que sa performance ne baisse pas dans le temps, etc.)
- **Un seuil d'acceptation explicite et objectif.** Décidé avant le développement, basé sur des éléments objectifs et mesurables pour éviter tout biais de perception.
- **Une réévaluation périodique.** Définie comme une évaluation régulière, pour s'assurer qu'aucune dérive n'apparaît.

6.3 L'auditabilité

Un système agentique, avec ses cohortes d'agents, produit une quantité considérable d'informations : raisonnements, échanges entre agents, décisions intermédiaires. Le volume et la vitesse dépassent très largement ce qu'un humain peut lire.

Tel quel, dans le format brut produit par les agents, tout en gardant en tête les chemins de transmission d'une information entre plusieurs agents, un système agentique est de fait impossible à auditer dans un délai utile.

La solution est de ne pas tout lire. Un dispositif d'audit efficace ne cherche pas à évaluer chaque action du système. Il fonctionne en deux temps :

- **Localiser.** Détecter automatiquement les zones de friction, de sous-performance ou de défaillance - c'est-à-dire réduire un flux illisible à une poignée de points qui méritent l'attention.
- **Analyser.** Déployer sur ces points précis la capacité d'analyse adaptée au type de problème recherché.

Selon les cas, ces dispositifs seront « human friendly » (restitution lisible du cheminement), automatisés, ou simplifiés (traces réduites à l'essentiel). Ce qui compte est le principe : on n'audite pas le système, on audite ce que le système signale comme anormal.

À RETENIR

C'est un chantier à part entière, à anticiper dès la conception plutôt qu'à ajouter après le premier incident.

6.4 Les garde-fous

Le passage à des systèmes agentiques accroît fortement l'autonomie accordée à la machine. Des dizaines d'agents interagissent simultanément entre eux et avec des environnements tiers (e.g. bases de données, DSP, ad servers, outils internes, etc.).

Cette autonomie est la source du gain de performance sans commune mesure avec les premiers agents d'il y a douze mois.

Elle est aussi la source du risque : plus un système décide seul, plus il peut déborder du cadre prévu. Le danger n'est pas seulement qu'un agent « s'échappe » et compromette

l'exécution ; c'est qu'il compromette les environnements tiers auxquels il est connecté (e.g. décider de mettre en pause ou supprimer toutes les campagnes du DSP).

Il est donc impératif de poser des barrières strictes et des points de vérification humaine.

LE PRINCIPLE

Trois règles structurent cette démarche, et toutes reposent sur le même principe :

L'autonomie ne se donne pas, elle se gagne.

A. Valider manuellement avant d'accorder l'autonomie

En phase de développement / déploiement initiale, toutes les actions clés passent par une évaluation humaine, assortie d'une mesure du taux d'erreur et du risque associé.

Ce n'est qu'une fois ces erreurs devenues rares et mineures que l'on retire la validation humaine et que l'on laisse le système opérer seul. La validation n'est pas une contrainte de démarrage : c'est l'instrument de mesure qui autorise le passage à l'étape suivante.

B. Étendre le périmètre progressivement

Un système agentique doit gagner le droit de s'étendre. Lorsqu'il est créé puis testé, il doit l'être avec un nombre réduit de tâches sur un périmètre restreint. À chaque fois que les erreurs et les risques observés sont jugés acceptables, il est possible d'élargir le périmètre et la liste d'actions autorisées.

Point critique : chaque élargissement remet temporairement à risque ce qui avait été validé avant. Un agent fiable sur cinq tâches peut se dégrader sur dix - parce que le contexte s'alourdit, que les interactions se multiplient et que les cas limites se croisent. Toute extension impose donc une réévaluation des cas précédents, pas seulement du nouveau.

C. Poser des règles infranchissables

Quel que soit le degré de maturité atteint, certaines limites ne doivent jamais tomber. Trois dispositifs les matérialisent :

- **Les interdits absolus et les validations permanentes.** Une liste d'actions que le système n'exécutera jamais seul, quel que soit son niveau de confiance - typiquement les actions irréversibles, engageantes ou de grande ampleur.
- **La détection automatique de franchissement.** Un principe de précaution absolue suppose d'adopter la philosophie suivante : « toute barrière, aussi solide soit-elle, peut être contournée tôt ou tard ». Il faut donc mettre en place des alertes permettant de détecter instantanément le franchissement de ces barrières théoriquement hermétiques, pour se prémunir de tout événement catastrophique.
- **La sauvegarde systématique.** Sur le même principe que les datacenters qui répliquent leurs données pour survivre à la perte d'un site, conserver des points de restauration, pour qu'une perte de contrôle reste réparable, est également une bonne pratique à adopter.

À RETENIR

Construire un agent n'est pas qu'un travail technique, aussi bien en termes d'effort que de temps.

Une fois la conception initiale livrée, une grande partie du temps du projet se joue sur : **valider, itérer, mesurer la performance et le risque, ajuster.**

Cette phase est tout aussi critique que le développement, pour valider la robustesse du système. Elle doit être budgétée comme telle dès le départ.

Un planning qui s'arrête à la livraison technique produira un système certes fonctionnel mais fragile, incomplet, voire même dangereux.

SYNTHÈSE

Ce qu'il faut retenir

Huit points résument les arbitrages développés dans ce guide. Chaque renvoi permet de revenir à la partie correspondante.

01 MCP n'est pas un concurrent des protocoles métier

Le MCP est un connecteur ; AAMP et AdCP sont des protocoles métier. La question « MCP ou protocole ? » ne se pose pas : les protocoles s'exposent eux-mêmes via MCP. → [Partie 2](#)

02 Une seule question tranche l'arbitrage stratégique

Vos agents ont-ils vocation à transacter avec des agents extérieurs à l'entreprise ? Si oui, le protocole est obligatoire. Si non, l'arbitrage entre solution interne et ossature protocolaire redevient pertinent. → [Partie 3](#)

03 Ajouter des agents en plus d'un protocole doit se justifier

S'il se contente de relayer les requêtes, il n'ajoute que de la latence, du coût en tokens et un point de maintenance. Un agent complémentaire doit apporter une valeur ajoutée spécifique qui peut être employée par le protocole. → [Partie 2](#)

04 L'agent isolé a vécu ; place aux systèmes agentiques

Le standard agentique le plus performant est représenté par les systèmes agentiques : Une cohorte d'agents sous-spécialisés, orchestrés par un agent maître, qui fragmentent les tâches, itèrent et challengent leurs propres résultats. → [Partie 4](#)

05 Le coût de construction s'effondre, le coût d'exploitation augmente

Selon notre expérience, les frameworks matures divisent le temps et le budget de développement par trois ou quatre. Mais la facture d'exploitation grimpe : plus de contexte, plus d'itérations, plus d'agents en parallèle. → [Parties 4 et 6](#)

06 N'attendez pas la stabilité technologique

Investissez petit, ciblé et vite plutôt que large et parfait. Le coût d'opportunité de l'attente dépasse le risque d'obsolescence de l'investissement. → [Partie 5](#)

07 Ne sur-ingénieriez jamais pour combler les manques des modèles

Les angles morts des modèles et des frameworks seront comblés par ceux qui les produisent, ou rendus non pertinents par le marché. Toute surcouche construite pour les compenser est un coût perdu. → [Partie 5](#)

08 L'autonomie ne se donne pas, elle se gagne

Validation humaine d'abord, élargissement progressif du périmètre ensuite, règles infranchissables toujours. Cette phase se budgete comme le développement. → [Partie 6](#)

EPICFLARE

À propos d'Epicflare

Conseil AdTech et développement IA

Epicflare accompagne les médias et les media buyers pour améliorer leur performance publicitaire et leurs opérations. Nous associons expertise métier et réalisation technique, en tenant compte des outils et des pratiques de chaque organisation.

Conseil AdTech

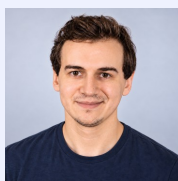
Monétisation et stacks publicitaires des médias ; méthodes d'achat, paramétrage des campagnes et mesure de la performance des media buyers.

Conseil et développement IA

Identification des usages, étude de faisabilité et conception des workflows. Développement d'agents, d'automatisations et de connexions sur mesure, du prototype à la solution intégrée aux outils du client.

Formation IA

Accompagnement des dirigeants dans leurs arbitrages et des équipes dans l'adoption de pratiques et de solutions adaptées à leur métier.



Raphaël Ambit

Fondateur d'Epicflare

Ancien VP Advertising chez Dailymotion et ancien Senior Director Technology chez Jellyfish. Également CTO d'Alliance Digitale. Son expérience côté éditeur et agence relie les enjeux métier aux choix opérationnels et techniques.

Quelques références



Dailymotion



Homa



Everyday



InfoPro Digital



Waterwiser



Le Parisien



Technip Energies

PARLONS DE VOTRE PROJET

contact@epicflare.io · epicflare.io

