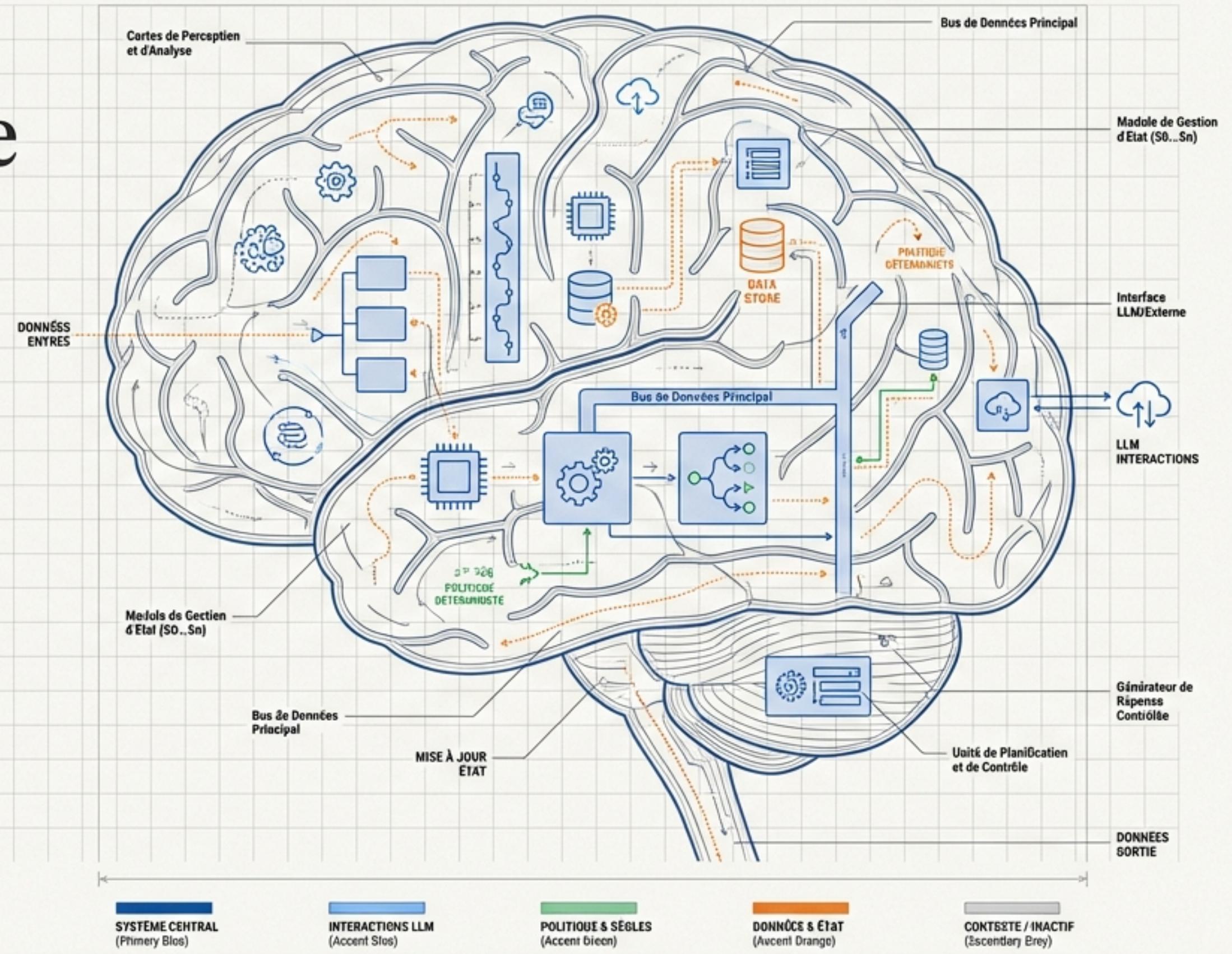
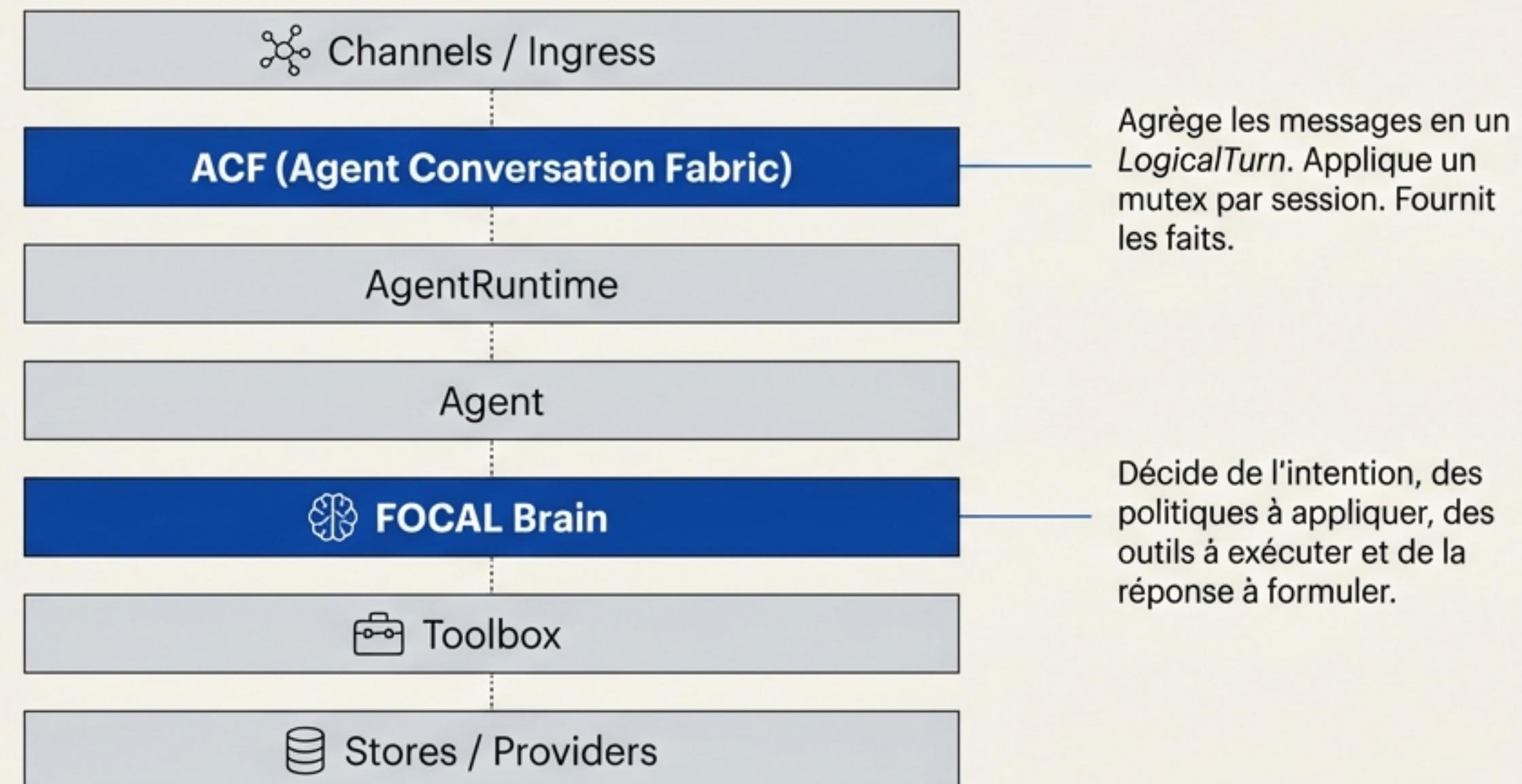


FOCAL Brain: De la Philosophie à l'Exécution

Une visite architecturale du
moteur d'alignement
conversationnel



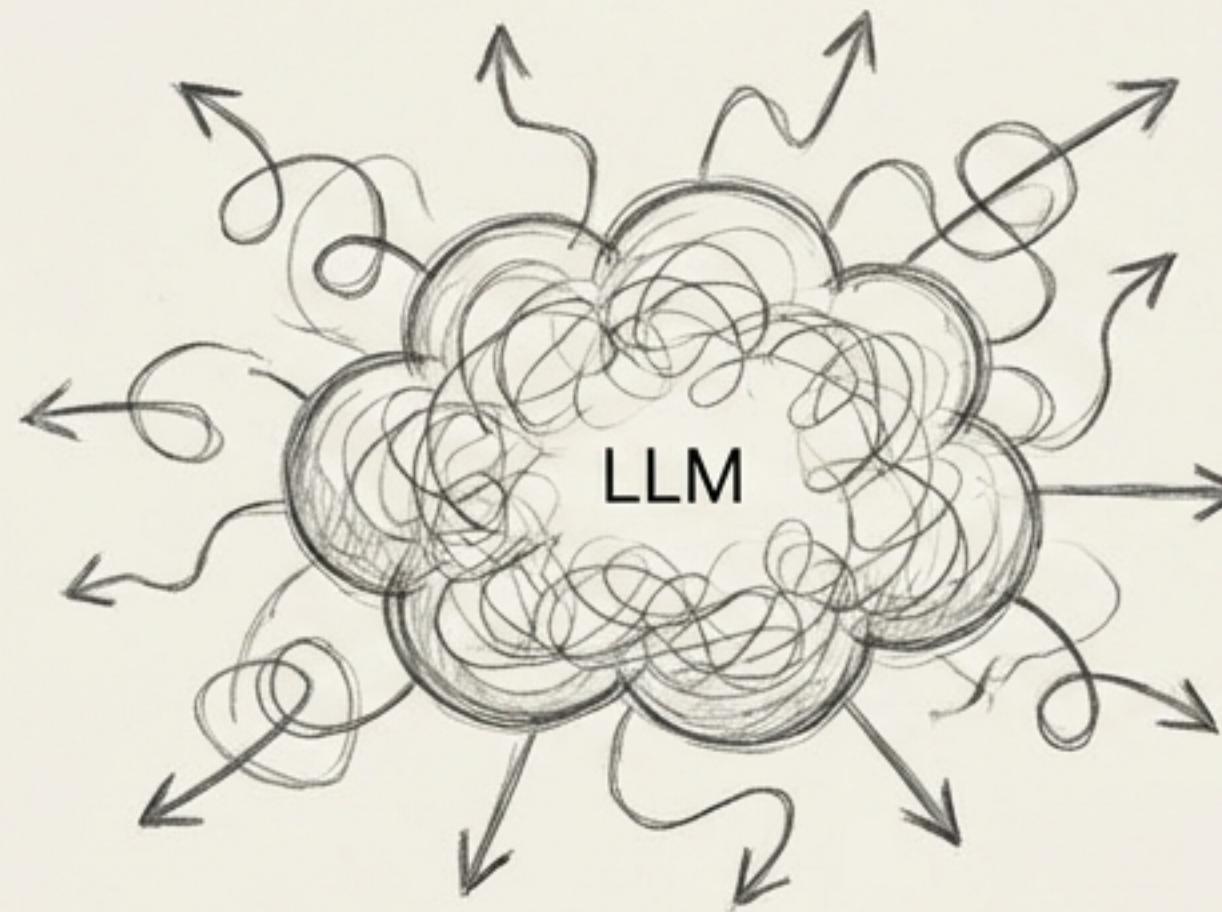
Un cerveau spécialisé dans un écosystème plus large



FOCAL n'est pas la plateforme ; c'est l'unité de décision au sein d'une enveloppe d'exécution robuste.

Notre philosophie fondatrice : Maîtriser le chaos

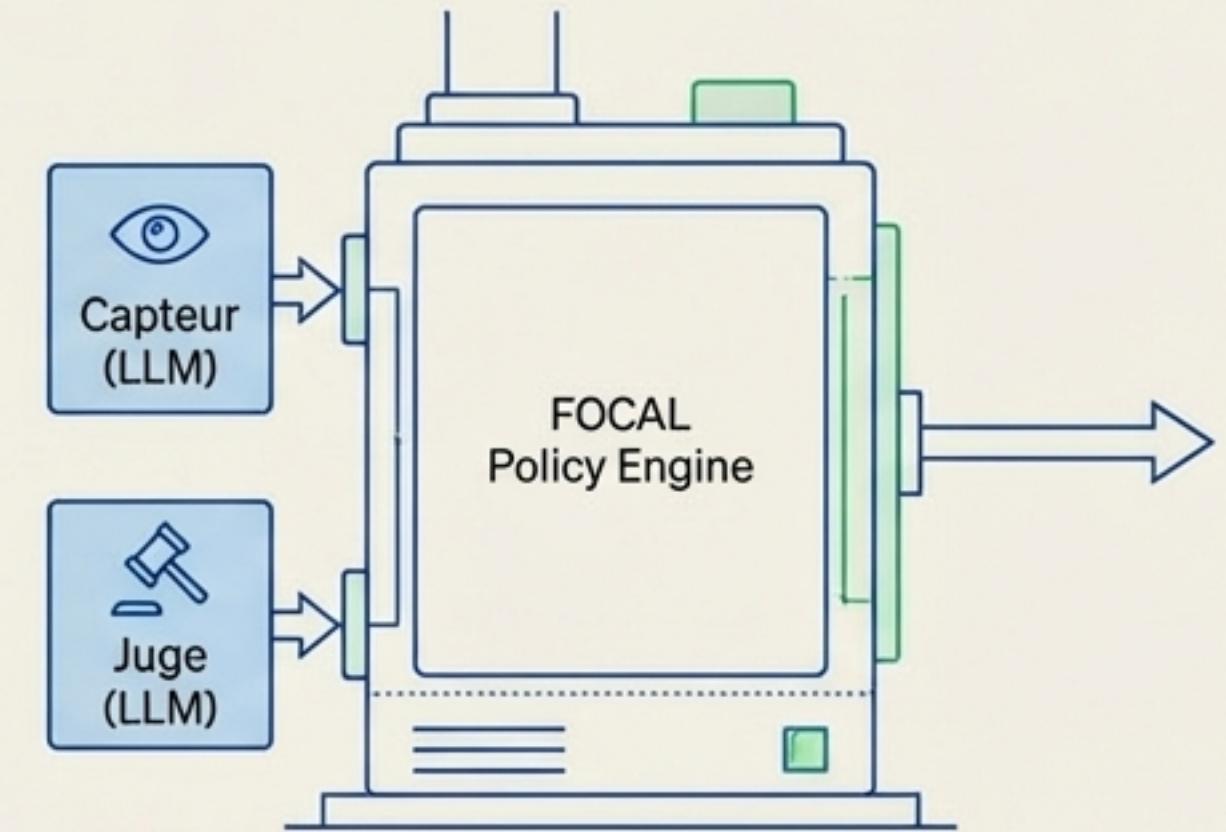
Ce que nous évitons



Le LLM comme moteur de politique

Les LLMs sont des capteurs et des juges, pas le moteur de politique.

Notre approche



- **Capteur (Sensor):** Extrait des signaux structurés (intention, ton) de l'input utilisateur.
- **Juge (Judge):** Évalue la pertinence sémantique (quelles règles s'appliquent) ou la conformité subjective.
- **Politique (Policy Engine):** La logique de décision réside dans la configuration déterministe et est appliquée au moment de l'exécution.

Les fondations de la fiabilité : Quatre 'stores' alignés par domaine



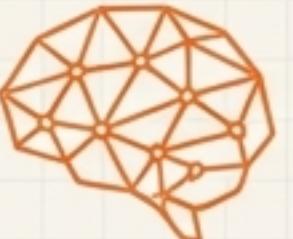
ConfigStore

'Comment doit-il se comporter ?'
(Règles, Scénarios, Templates.
Lecture intensive, versionné).



SessionStore

'Que se passe-t-il maintenant ?'
(Etat de la session active. Faible
latence en lecture/écriture).



MemoryStore

"De quoi se souvient-il ?" (Mémoire
à long terme, recherche
sémantique. Ajout intensif).



AuditStore

"Que s'est-il passé ?" (Traces
immuables, conformité. Ajout
seul, requetable).

Principe clé : 'Zero in-memory state'

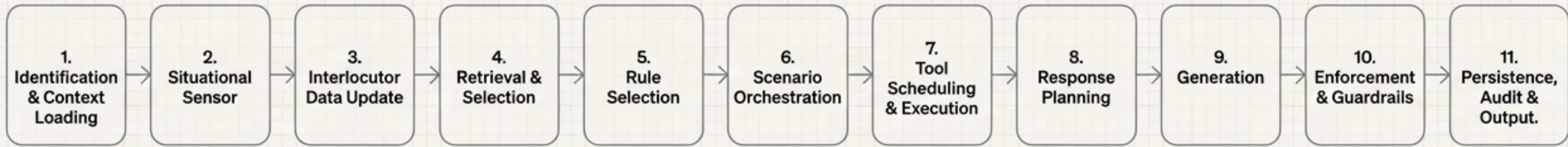
Aucun état local au pod n'est
le système de référence.
N'importe quel pod peut traiter
n'importe quelle requête,
garantissant la scalabilité
horizontale et la résilience.

Chaque domaine a des exigences différentes (latence, durabilité, requêtes).

Cette séparation permet l'optimisation indépendante et prévient les
fuites accidentnelles d'état.

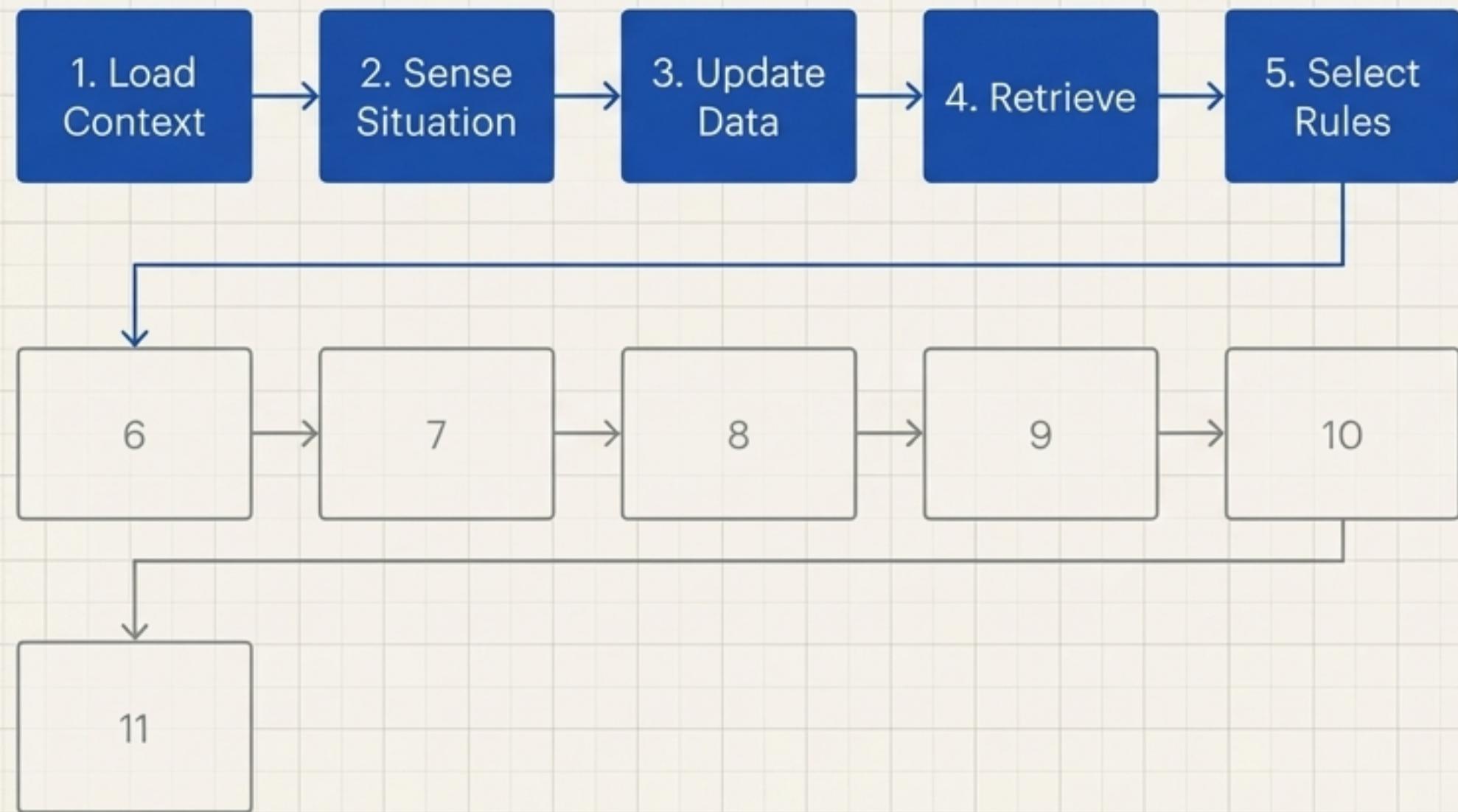
Le voyage d'un `LogicalTurn` à travers le pipeline

Le cerveau FOCAL s'exécute une fois par `LogicalTurn` – une unité conversationnelle créée par ACF. Nous allons maintenant suivre ce tour à travers notre pipeline en 11 phases.



Acte I : Comprendre la situation (Phases 1-5)

Transformer un message utilisateur brut en une compréhension structurée de la situation et un ensemble de règles applicables.



1. Load Context

Charger l'état de la session, les données de l'interlocuteur et la configuration.



2. Sense Situation

Extraire l'intention, le ton et les variables candidates.



3. Update Data

Mettre à jour les 'faits' connus sur l'utilisateur en mémoire.



4. Retrieve

Aller chercher les règles, scénarios et souvenirs pertinents.

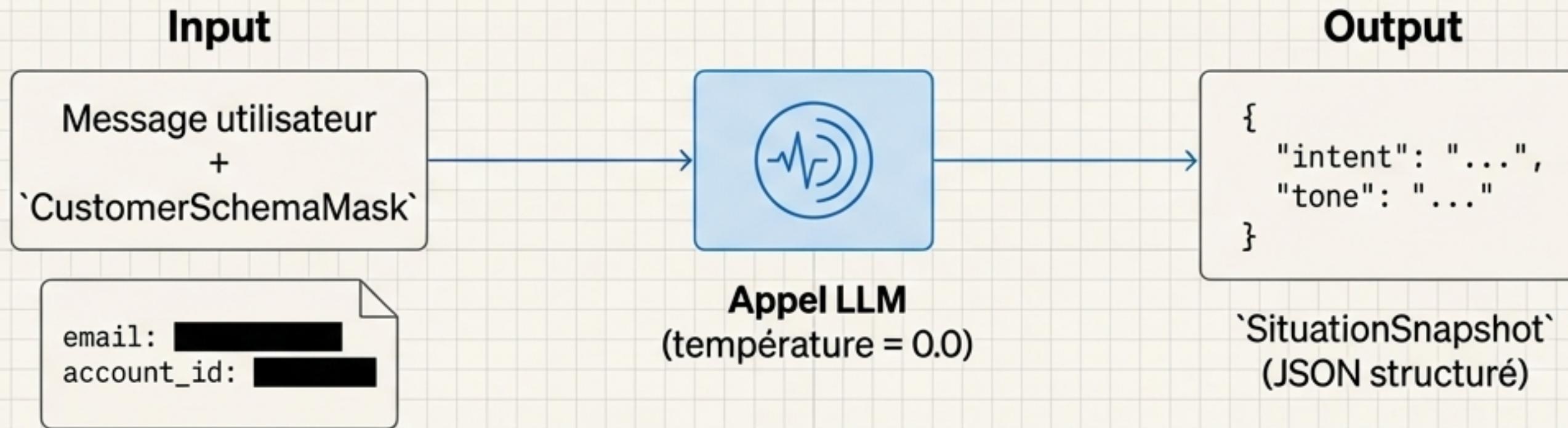


5. Select Rules

Filtrer et sélectionner l'ensemble final de règles à appliquer.

Zoom : Le 'Situational Sensor' (Phase 2)

L'extraction structurée avant la décision



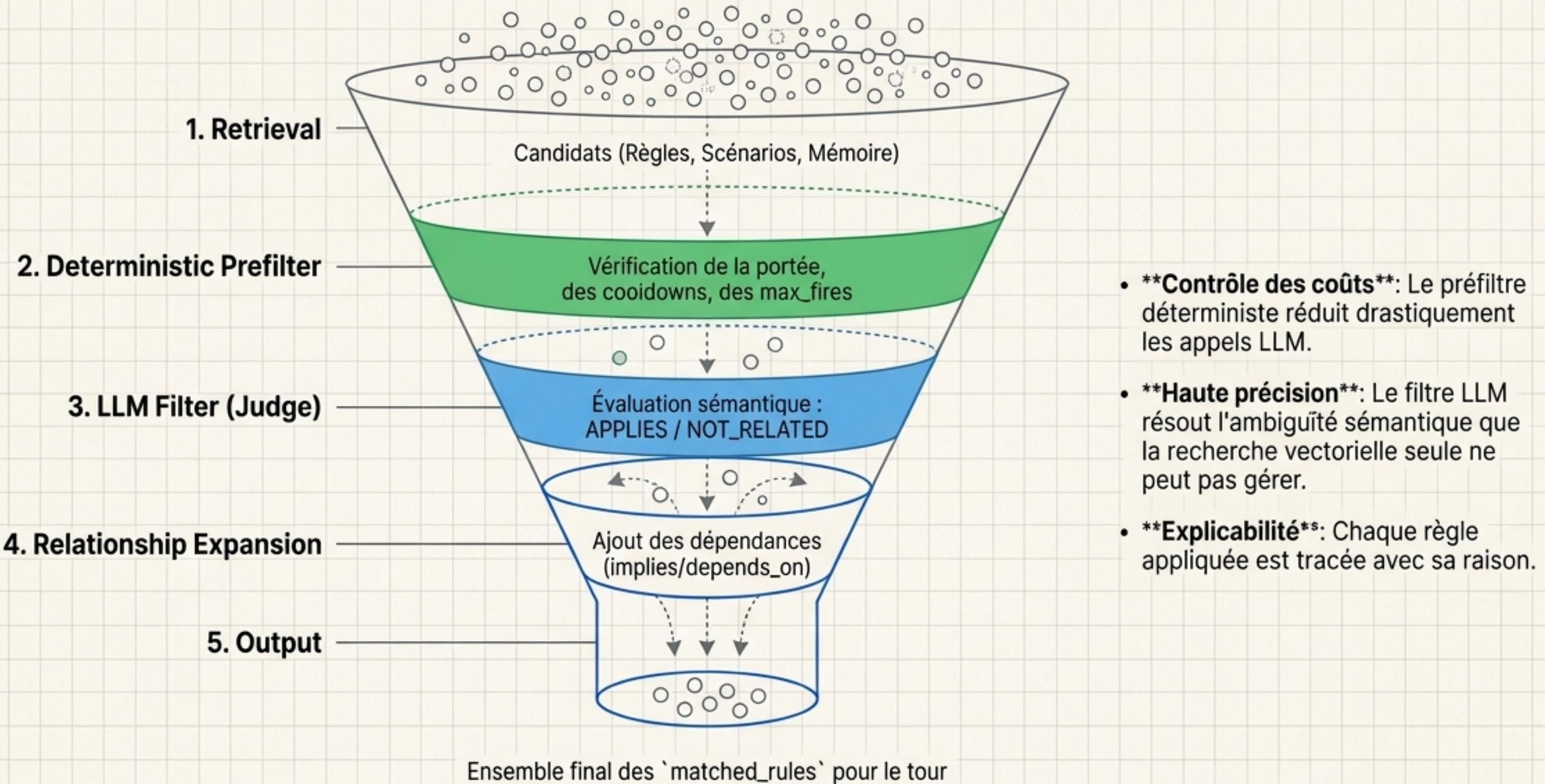
Le LLM voit seulement *quelques* champs existent et s'ils ont une valeur, mais jamais les *valeurs* elles-mêmes. C'est une protection essentielle contre la fuite de PII.

Bénéfices

- **Séparation des préoccupations:** L'extraction est isolée de la génération.
- **Traitement typé:** Les phases suivantes opèrent sur des données structurées.
- **Auditabilité:** Le `SituationSnapshot` est un artefact clé dans le `TurnRecord`.

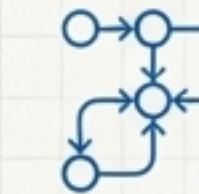
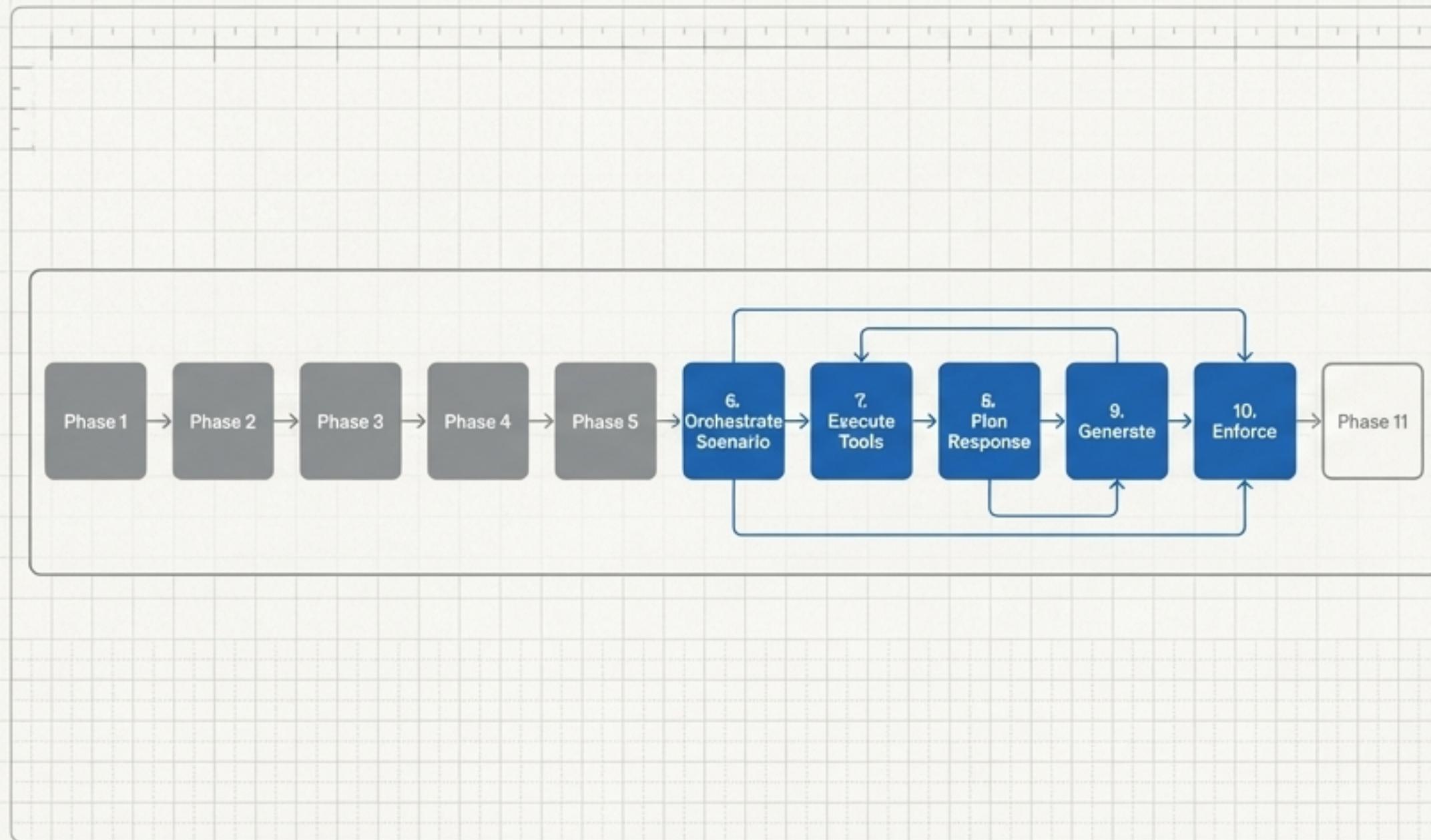
Zoom : Sélection de Règles (Phases 4-5)

De la recherche vectorielle à l'application de politiques



Acte II : Décider et Agir (Phases 6-10)

Passer de la compréhension à un plan d'action concret, à l'exécution d'outils et à la génération d'une réponse rigoureusement vérifiée.



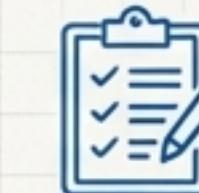
6. Orchestrator Scenario

Déterminer l'état du workflow (machine d'état explicite).



7. Execute Tools

Exécuter les effets de bord de manière déterministe.



8. Plan Response

Créer un `ResponsePlan` structuré pour guider la génération.



9. Generate

Produire la réponse en langage naturel.



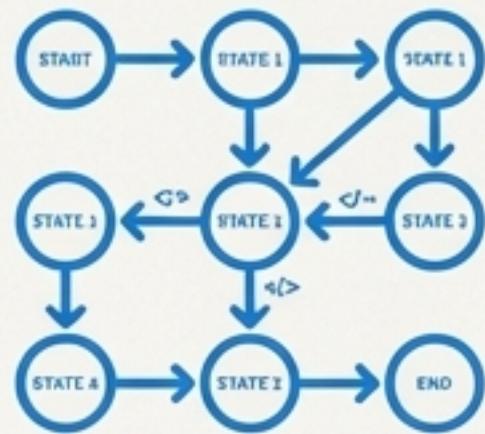
10. Enforce

Valider la conformité de la réponse aux contraintes.

Zoom : Orchestration et Exécution (Phases 6-7)

Des workflows fiables et des effets de bord déterministes

Scenarios (Phase 6)



Les scénarios sont des machines d'état explicites, pas un comportement implicite du prompt. Ils permettent des sessions de longue durée et des mises à jour en toute sécurité via des plans de migration.

La logique de navigation est en cours d'implémentation ; supporte déjà le démarrage, la continuation et le "step skipping".

Tool Execution (Phase 7)



Les outils s'exécutent parce qu'une règle ou une étape l'exige, et non parce que le modèle l'a deviné. La `Toolbox` applique les politiques de sécurité et l'idempotence.

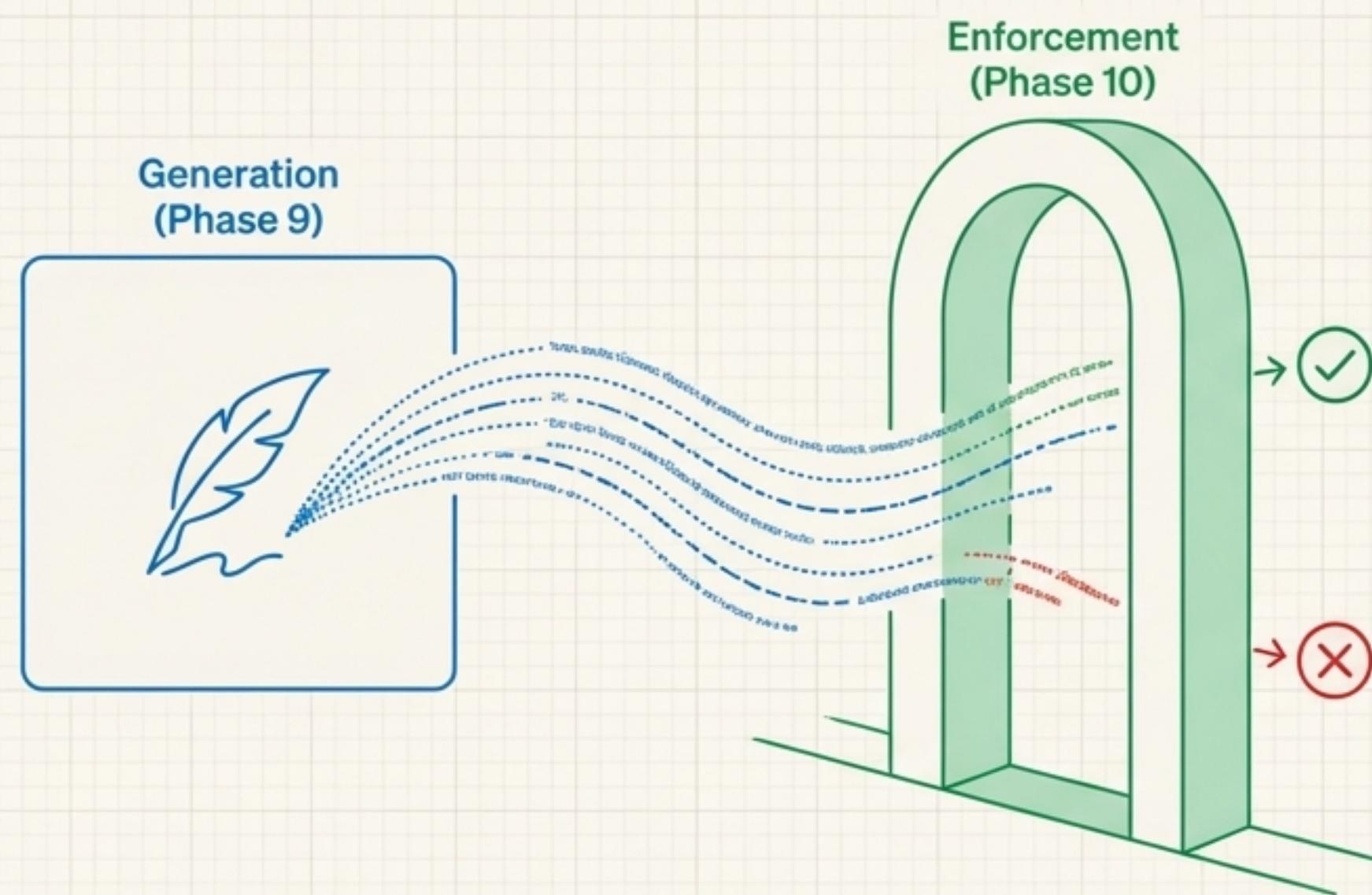
L'ordonnancement est conçu, l'exécution s'appuie encore sur un mécanisme hérité.

Zoom : Génération et Garde-fous (Phases 9-10)

La créativité sous contrainte, validée avant la livraison

Points clés sur la Génération (Phase 9)

- Peut être contournée par des templates `EXCLUSIVE` pour des réponses déterministes et à faible latence.
- Le prompt est structuré par le `ResponsePlan` ; ce n'est pas un simple 'fourre-tout'.



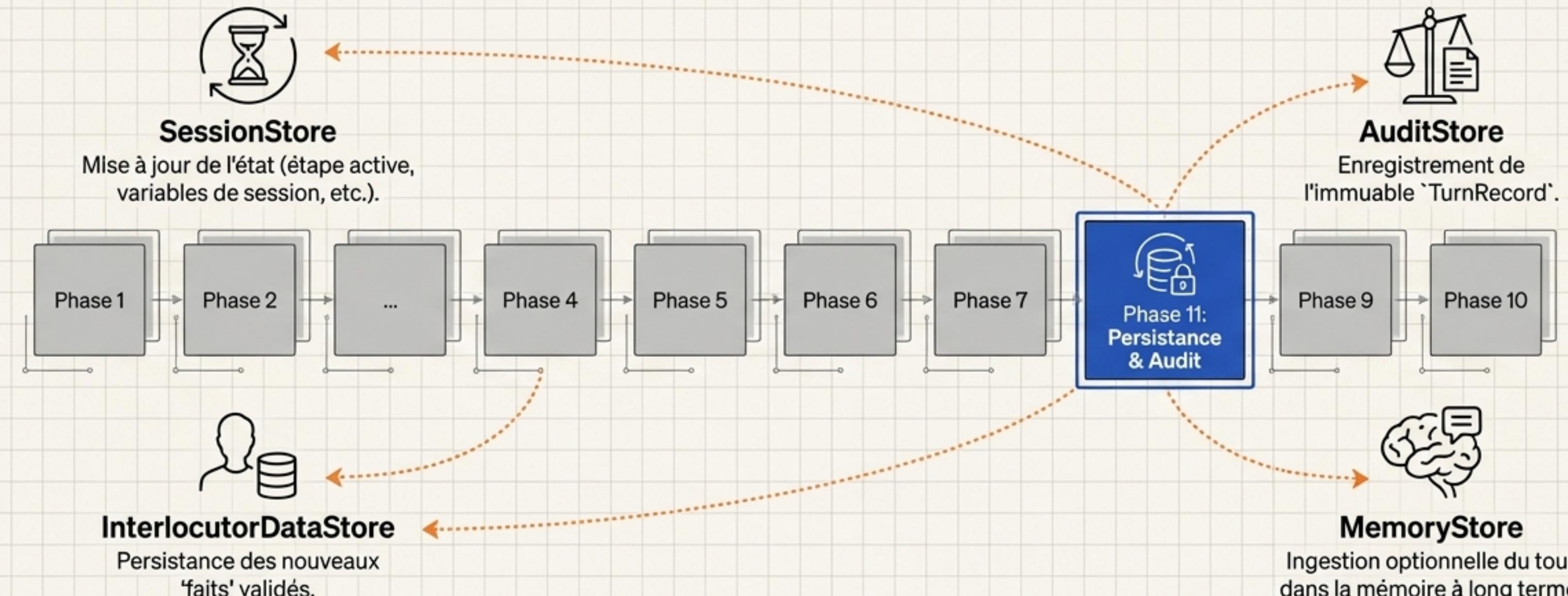
Deux voies de validation

- ****Voie 1 (Déterministe)*:** Évalue une `enforcement_expression` (ex: `amount <= 50`). Rapide et sans ambiguïté.
- ****Voie 2 (Subjective)*:** Un LLM-Juge vérifie la conformité à des contraintes non-expressibles (ex: 'Ne pas faire de promesses').

Les contraintes `GLOBAL` sont toujours appliquées, même si elles n'ont pas été récupérées, garantissant une sécurité par défaut.

Acte III : Persistance et Audit (Phase 11)

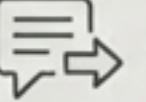
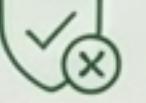
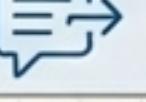
Rendre chaque tour durable et observable



La persistance est **parallélisée** grâce à la **séparation des domaines**, optimisant la performance tout en assurant la cohérence pour le tour suivant.

Le résultat : Une traçabilité complète pour chaque décision

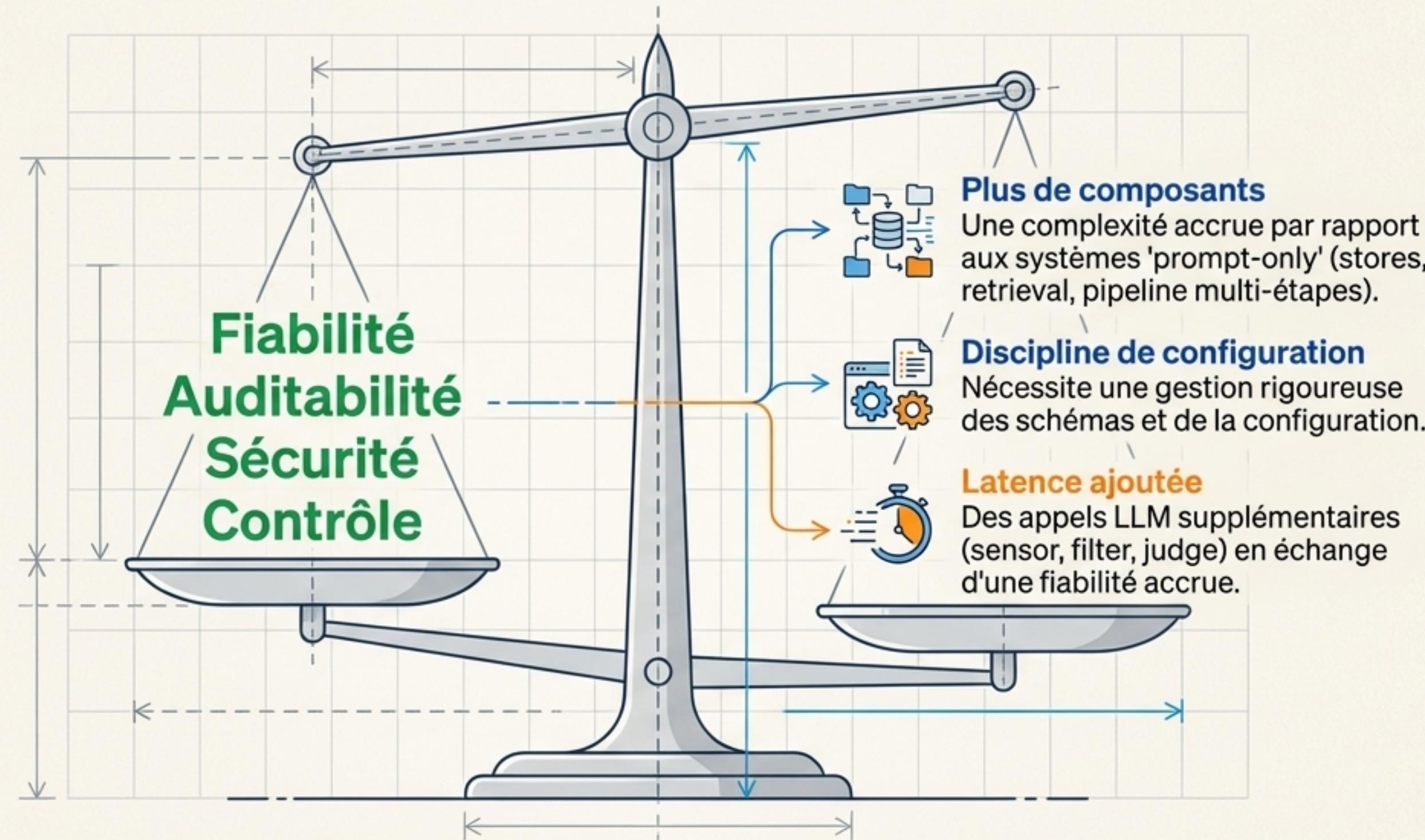
Chaque tour produit une trace forensique immuable. Le débogage en production passe d'une conjecture à une analyse de faits.

`TurnRecord` (Immutable Forensic Trace)	
 Input	LogicalTurn: { content: '...', timestamp: '...' }
 Understanding	SituationSnapshot: { intent: '...', tone: '...', entities: [...] }
 Policy	MatchedRules: [{ rule_id: 'R123', reason: '...' }, { rule_id: 'R124', reason: '...' }]
 State	ScenarioState: { active_scenario: '...', current_step: '...' }
 Execution	ToolCalls: [{ tool: '...', params: {..}, result: '...' }]
 Validation	EnforcementViolations: [{ violation: '...', constraint: '...' }]
 Output	FinalResponse: { content: '...', timestamp: '...' }

Ce que cette conception architecturale nous apporte



Une évaluation honnête : Les compromis d'une approche délibérée



Ce sont les compromis que nous faisons consciemment pour construire des agents IA de production qui sont dignes de confiance, auditables et évolutifs.