DOCUMENT D’ANTÉRIORITÉ

SOPHIA

Sovereign Orchestrator Platform for Holistic Intelligence Architecture

Auteur : Damien Guesdon
Date : 2026

Le présent document constitue une description technique de l’architecture logicielle SOPHIA conçue par Damien Guesdon en Février 2026.

La présente description vise à documenter les principes architecturaux, les mécanismes techniques et l’organisation conceptuelle du système SOPHIA tels qu’imaginés et conçus par l’auteur à cette date.

---

1. Objet du document

Le présent document décrit l’architecture conceptuelle d’un système nommé **SOPHIA**, destiné à permettre l’orchestration, l’exécution et la gouvernance de systèmes d’intelligence artificielle dans un environnement souverain.

SOPHIA constitue une architecture d’infrastructure destinée à orchestrer, exécuter et gouverner des systèmes d’intelligence artificielle hétérogènes au sein d’un environnement cohérent, sécurisé et souverain.

---

2. Définition de SOPHIA

SOPHIA signifie :

Sovereign Orchestrator Platform for Holistic Intelligence Architecture

Cet acronyme désigne une plateforme logicielle permettant :

- l'orchestration de systèmes d'intelligence artificielle
- la gestion d'agents spécialisés
- la gestion du contexte et de la mémoire
- la coordination d'environnements d'exécution
- la sécurisation des interactions avec l'extérieur
- la gouvernance des flux de données.

L'architecture vise à transformer des technologies d'intelligence artificielle isolées en un **système opérationnel intégré**.

---

3. Domaine technique

L'architecture SOPHIA concerne les domaines suivants :

- orchestration de systèmes d'intelligence artificielle
- systèmes multi-agents
- gestion de contexte pour IA
- gestion de mémoire et bases vectorielles
- infrastructures d'exécution d'agents
- architectures sécurisées pour IA.

Elle peut intégrer différents types de modèles :

- modèles de langage
- modèles de machine learning
- modèles de deep learning
- modèles de vision par ordinateur
- modèles d'embedding.

---

4. Problématique

L’utilisation opérationnelle de l’intelligence artificielle rencontre plusieurs difficultés.

Parmi celles-ci :

- fragmentation des outils et frameworks
- absence d’orchestration cohérente entre agents
- gestion limitée du contexte
- dépendance aux services cloud externes
- difficulté à contrôler la circulation des données
- absence de mémoire durable structurée
- difficulté à reproduire les environnements d’exécution.

Ces limitations empêchent la construction de systèmes d’IA :

- souverains
- auditables
- reproductibles
- gouvernables.

---

5. Objectif de l’architecture

L’objectif de SOPHIA est de fournir une **infrastructure d’orchestration** permettant de coordonner agents, modèles, environnements d’exécution et sources de connaissance afin de construire **des systèmes d’intelligence artificielle** opérationnels et gouvernables.

Cette infrastructure permet notamment :

- la planification et l’exécution de tâches complexes
- la coordination d’agents spécialisés
- la gestion du contexte et de la mémoire
- l’exécution sécurisée de code et d’analyses
- la protection des données
- l’intégration de différentes technologies d’IA.

---

6. Architecture générale

SOPHIA constitue une infrastructure d'orchestration destinée à coordonner les différents composants d'un système d'intelligence artificielle.

Cette infrastructure agit comme une couche de gouvernance et d'exécution indépendante des agents, des modèles et des outils.

L'architecture de SOPHIA repose sur une infrastructure d'orchestration coordonnant les différents composants d'un système d'intelligence artificielle.

Cette infrastructure agit comme une couche de gouvernance et d'exécution indépendante des agents, des modèles et des outils utilisés par le système.

Elle organise notamment :

- la résolution des projets
- la gestion du contexte
- la gestion des environnements d'exécution
- le routage des moteurs d'intelligence artificielle
- la gestion de la mémoire et des connaissances
- la sécurisation des interactions externes.

---

7. Orchestrateur cognitif

SOPHIA inclut un orchestrateur chargé de :

- analyser les requêtes
- identifier les intentions
- planifier les actions
- sélectionner les agents appropriés
- coordonner les tâches.

L'orchestrateur agit comme un moteur de décision permettant d'organiser les interactions entre les différents composants du système.

Cette fonction architecturale est indépendante d'un framework particulier.

Les composants de l'architecture peuvent être implémentés avec différentes technologies logicielles.

La présente description concerne l'architecture conceptuelle et non une implémentation spécifique.

---

8. Gestion des projets

SOPHIA introduit un modèle de **projet structuré**.

Chaque projet est défini par un **manifeste de projet**.

Ce manifeste contient notamment :

- un identifiant de projet
- la source de travail
- la configuration du projet
- les agents par défaut
- les paramètres d'exécution.

Le manifeste constitue la **source de vérité** permettant au système de résoudre un projet.

Architecture des espaces de projet

SOPHIA introduit une séparation structurelle entre trois espaces distincts :

1. Brain

Le brain constitue l'espace de mémoire et de pilotage du projet.

Il contient notamment :

- les spécifications
- la mémoire projet
- les décisions architecturales
- la configuration du projet.

2. Repository

Le repository constitue l'espace des livrables et artefacts produits.

Il contient notamment :

- le code source
- les notebooks
- les données de travail
- les ressources opérationnelles du projet.

3. Workspace runtime

Le workspace runtime constitue l'environnement d'exécution temporaire du projet.

Il peut être créé et détruit dynamiquement et permet :

- l'exécution de code
- l'analyse de données
- la production d'artefacts.

Cette séparation permet l'indépendance entre mémoire, production et exécution, et garantit la reproductibilité du système.

---

9. Résolution de projet

La résolution d'un projet dans SOPHIA s'effectue à partir de son manifeste.

Le manifeste permet de déterminer :

- l'identité du projet
- son espace de mémoire (brain)
- sa source de travail (repository)
- ses paramètres d'exécution
- ses capacités disponibles.

Le manifeste constitue la source de vérité permettant à l'orchestrateur de reconstruire l'environnement de travail du projet.

10. Workspace Management

Le système inclut un mécanisme de gestion d'environnements d'exécution appelés **workspaces**.

Un workspace constitue un environnement isolé permettant :

- l'exécution de code
- la manipulation de données
- l'analyse de fichiers
- la génération d'artefacts.

Les workspaces peuvent être :

- temporaires
- isolés
- reproductibles.

Ils peuvent être implémentés sous forme de conteneurs ou de pods dans une infrastructure conteneurisée.

---

11. Context Management

SOPHIA inclut un mécanisme de gestion du contexte permettant de :

- limiter la taille des informations envoyées aux modèles
- sélectionner les informations pertinentes
- maintenir un état cohérent entre les interactions.

Le contexte peut être construit à partir de différentes sources :

- mémoire de session
- mémoire projet
- base documentaire
- connaissances indexées.

12. Prompt Management

Les interactions avec les systèmes d’intelligence artificielle sont structurées via un système de gestion des prompts.

Ce système permet notamment :

- la gestion des instructions globales
- la version des prompts
- la gestion des rôles
- la standardisation des interactions avec les modèles.

13. Système de mémoire

La mémoire du système comprend plusieurs couches.

Par exemple :

- mémoire conversationnelle
- mémoire projet
- mémoire documentaire
- base vectorielle.

La base vectorielle permet la recherche sémantique dans les documents et les connaissances.

---

14. Model routing

SOPHIA inclut un mécanisme de routage permettant de sélectionner dynamiquement le moteur d'intelligence artificielle approprié.

Ce mécanisme permet d'utiliser plusieurs familles de modèles :

- modèles de langage
- modèles de machine learning
- modèles de deep learning
- modèles de vision par ordinateur
- modèles d'embedding.

Le routage peut être déterminé en fonction :

- du type de tâche
- du contexte
- des ressources disponibles
- des politiques de sécurité.

---

15. Architecture Zero Trust

SOPHIA adopte un modèle de sécurité basé sur le principe **Zero Trust**.

Ce modèle comprend notamment :

- isolation réseau stricte
- segmentation par namespaces
- politique de refus par défaut
- contrôle des flux de données.

---

16. SAS paranoïaque

Le système inclut un composant appelé **SAS paranoïaque**.

Ce composant agit comme une zone tampon sécurisée entre le système interne et les ressources externes.

Il permet notamment :

- l'exécution de tâches externes dans des environnements isolés
- l'anonymisation des requêtes
- la transformation des contenus récupérés.

---

17. Brouillard de recherche

Le système peut générer des requêtes simulées afin de masquer les intentions réelles d'une recherche.

Ce mécanisme est appelé **brouillard de recherche**.

Il vise à empêcher la corrélation entre les requêtes et les analyses réalisées par le système.

---

18. Tribunal sémantique

Les informations récupérées depuis l'extérieur peuvent être soumises à un mécanisme de validation appelé **tribunal sémantique**.

Ce mécanisme peut :

- croiser plusieurs sources
- détecter les contenus artificiels
- valider ou rejeter les informations.

Les données jugées incertaines peuvent être placées en quarantaine.

---

19. Protection des données

Le système inclut un mécanisme de filtrage destiné à éviter la fuite de données sensibles.

Ce mécanisme peut combiner :

- règles déterministes
- analyse sémantique.

---

20. Infrastructure technique

SOPHIA peut être déployé sur une infrastructure comprenant :

- serveurs dédiés
- cluster conteneurisé
- stockage sécurisé.

L'architecture peut fonctionner sans dépendance obligatoire à des services cloud externes.

---

21. Industrialisation

L'architecture est conçue pour permettre des déploiements reproductibles.

Cela peut inclure :

- déploiements conteneurisés
- automatisation de l'infrastructure
- packaging et déploiement automatisé des composants

---

22. Conclusion

SOPHIA constitue une architecture d'infrastructure permettant d'orchestrer, d'exécuter et de gouverner des systèmes d'intelligence artificielle au sein d'un environnement souverain.

L'architecture introduit notamment :

- un orchestrateur central
- une gestion structurée des projets
- une séparation entre mémoire, production et exécution
- une gestion du contexte et des interactions avec les modèles
- des mécanismes de sécurité et de validation de la connaissance.

Ces principes permettent de transformer des technologies d'intelligence artificielle isolées en un système cohérent, reproductible et gouvernable.

Cette architecture permet la construction de systèmes d'intelligence artificielle :

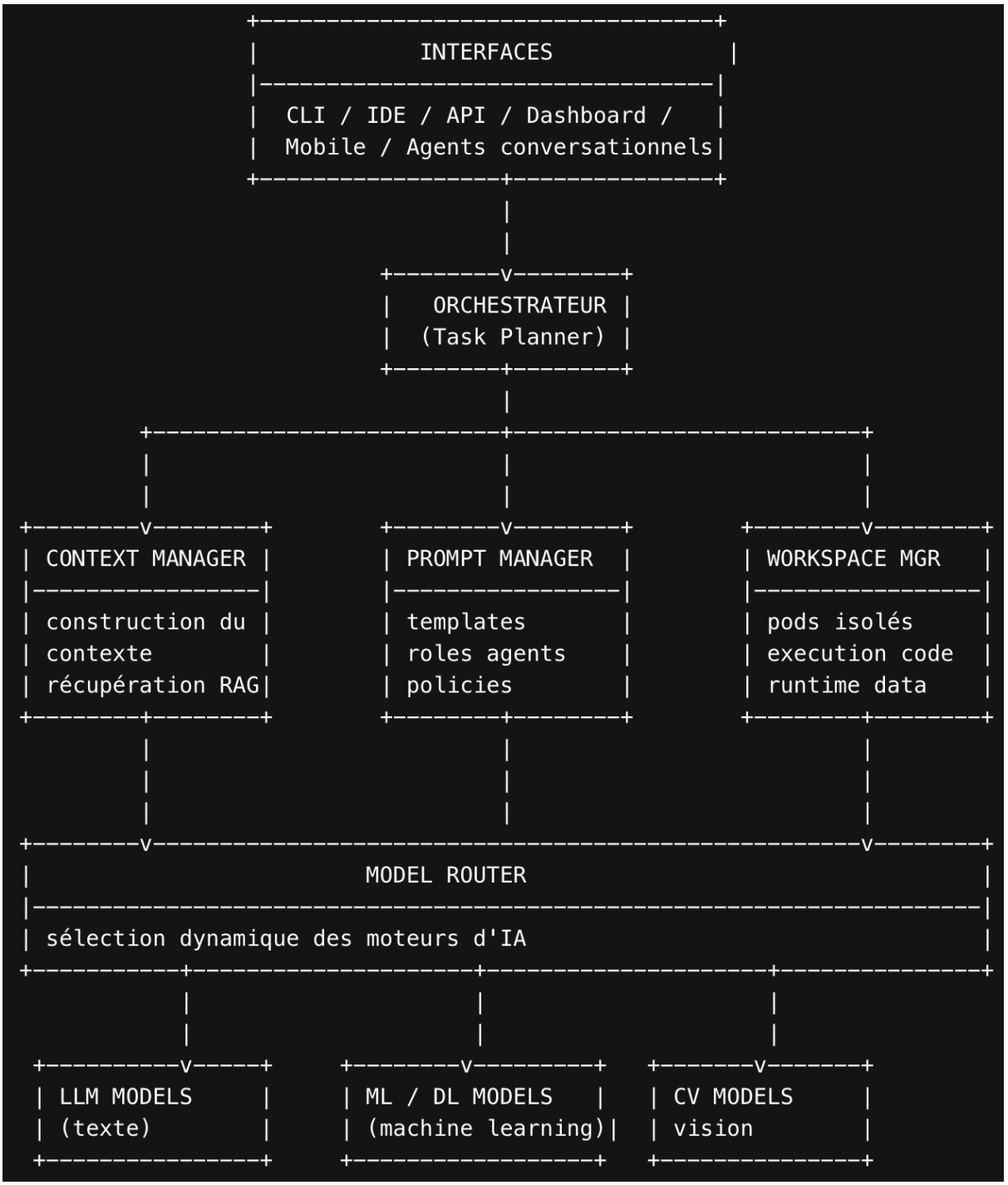
- souverains
- sécurisés
- auditables
- reproductibles.

Le présent document décrit l'architecture conceptuelle du système SOPHIA telle qu'elle a été conçue par l'auteur à la date indiquée.

Cette description vise à établir l'antériorité de la conception, incluant notamment les principes d'orchestration, la gestion du contexte, la gestion des environnements d'exécution, les mécanismes de validation de connaissance et les dispositifs de sécurité décrits dans le document.

Annexe A - Architecture globale SOPHIA

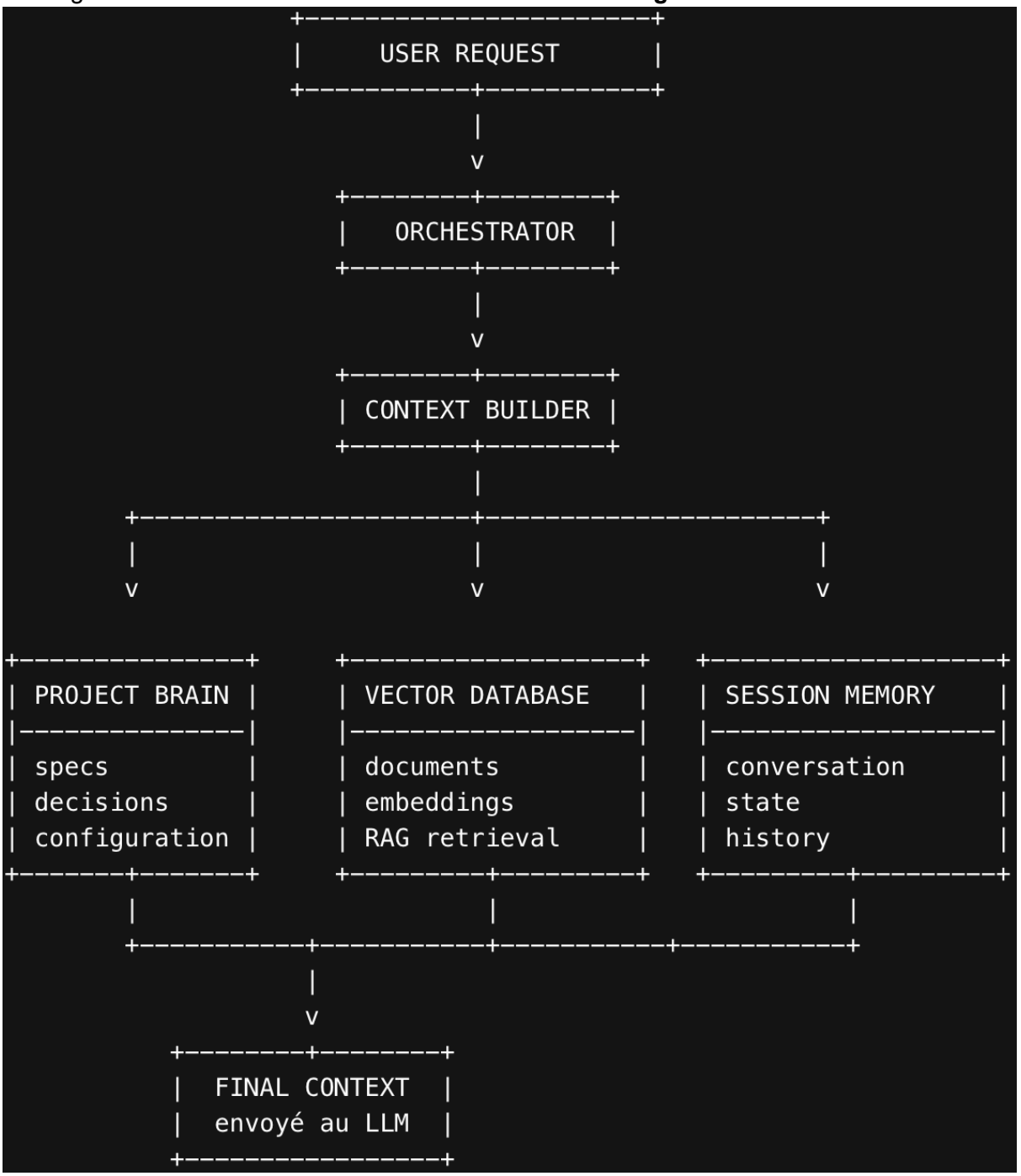
Ce diagramme montre **toutes les briques du système** et leurs relations.



SOPHIA n'est pas un simple système LLM mais une **plateforme d'orchestration IA multi-modèles**.

Annexe B - Gestion du contexte

Ce diagramme décrit **le fonctionnement du Context Management**.

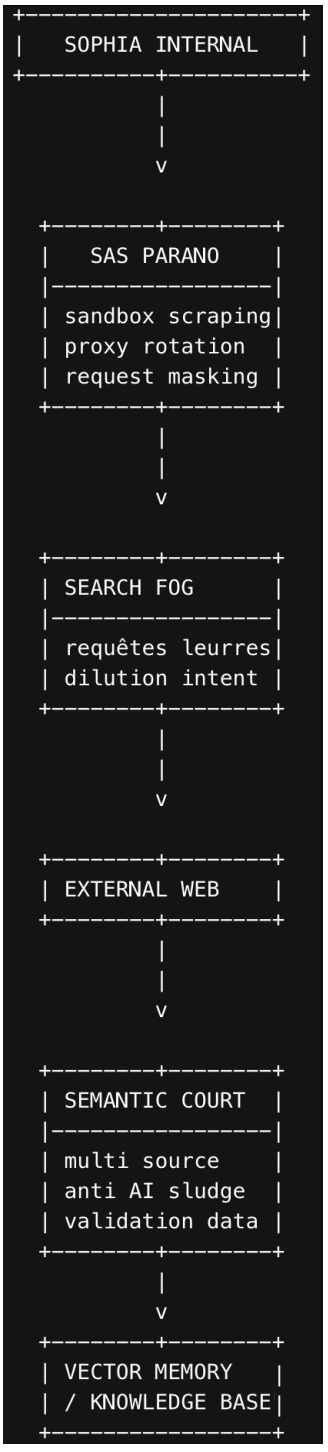


Ce mécanisme sert à :

- éviter les contextes trop volumineux
- sélectionner uniquement les informations utiles
- intégrer mémoire et documents.

Annexe C - Sécurité et flux externes

Ce diagramme montre les **mécanismes spécifiques de sécurité**.

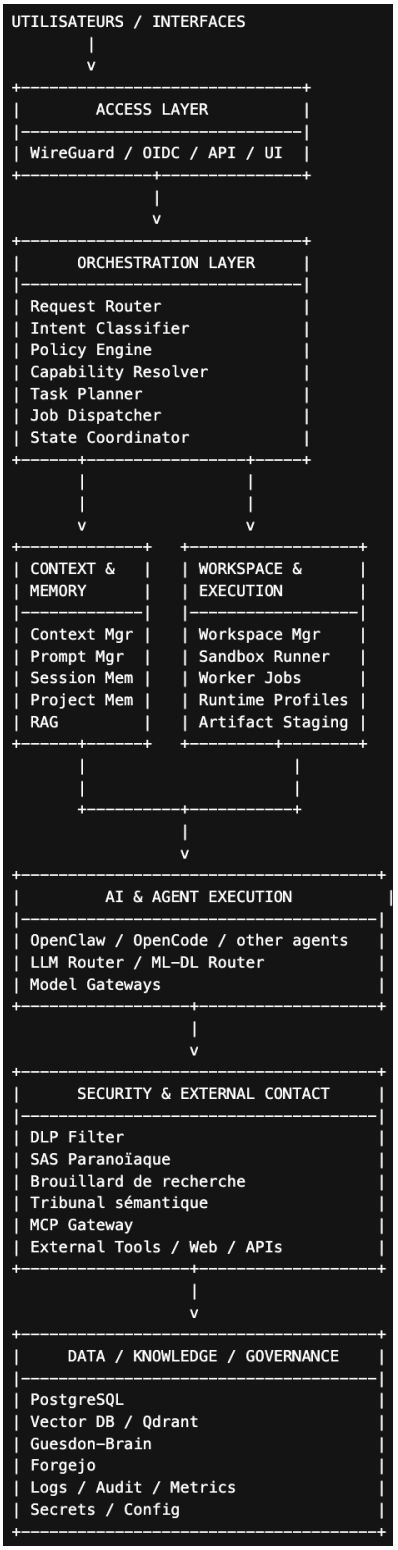


Ce diagramme met en évidence **les concepts différenciants de SOPHIA** :

- SAS paranoïaque
- brouillard de recherche
- tribunal sémantique

Annexe D - modules internes de SOPHIA

1. Vue macro



---

2. Décomposition détaillée par domaines

A. Couche d’accès et d’interface

A1. WireGuard access gateway
A2. UI web / dashboard
A3. API gateway
A4. IDE integration
A5. Mobile / PWA interface
A6. OIDC / identity entrypoint

Rôle :

- point d’entrée humain ou applicatif
- authentification
- contrôle d’accès initial
- exposition minimale du système

B. Couche d’orchestration centrale

B1. Request Router
B2. Intent Classifier
B3. Scope Resolver
B4. Policy Engine
B5. Capability Resolver
B6. Task Planner
B7. Job Dispatcher
B8. State Coordinator
B9. Agent Routing Logic
B10. Result Aggregator

Rôle :

- recevoir une demande
- comprendre ce qu’elle implique
- décider quel composant doit agir
- transformer une requête en tâches, puis en jobs si nécessaire
- centraliser la logique de contrôle

Important :

- l’orchestrateur n’est pas lié à un framework unique
- il peut être implémenté avec différents moteurs ou services
- il reste la **fonction architecturale centrale**

C. Couche de contexte, prompts et mémoire

C1. Context Manager
C2. Prompt Manager
C3. Session Memory
C4. Project State Store
C5. User Profile Store
C6. Knowledge Retriever
C7. RAG Index Resolver
C8. Memory Update Engine
C9. Spec Loader
C10. Brain Resolver

Rôle :

- construire le contexte minimal utile
- gérer les prompts et instructions
- séparer mémoire conversationnelle, mémoire projet et connaissance
- éviter l'explosion de contexte
- relier le système au Sanctuaire (`Guesdon-Brain`)

D. Couche projet et résolution structurelle

D1. Project Manifest Resolver
D2. Project Registry
D3. Space Resolver
D4. Organization Resolver
D5. Repo Resolver
D6. Runtime Resolver
D7. Branch Strategy Resolver

Rôle :

- résoudre un projet à partir de son manifeste
- retrouver :
    - son identité
    - son brain
    - son dépôt
    - sa stratégie de runtime
    - ses agents et capacités par défaut

E. Couche workspace et exécution

E1. Workspace Manager
E2. Workspace Provisioner
E3. Runtime Profile Manager
E4. Sandbox Runner
E5. Worker Job Runner
E6. Artifact Workspace
E7. Ephemeral Pod Manager
E8. Runtime Cleanup Engine
E9. Execution Monitor

Rôle :

- créer et détruire les environnements de travail
- monter repo + brain + artefacts
- exécuter des tâches réelles dans des environnements jetables
- fournir un terrain d'action aux agents

F. Couche agents

F1. Speaking Assistant
F2. Coding Agent
F3. Research Agent
F4. Image Agent
F5. Data Agent
F6. Future domain agents

Rôle :

- exécuter une spécialité
- ne jamais être la source d'autorité globale
- ne jamais s'appeler directement entre eux sans passer par l'orchestrateur

---

G. Couche modèles et routage IA

G1. LLM Router
G2. Local Model Router
G3. ML/DL Model Router
G4. Embedding Model Router
G5. Inference Namespace
G6. Local/air-gapped inference services
G7. Failover model selection

Rôle :

- sélectionner le moteur approprié
- gérer plusieurs familles de modèles :
    - LLM
    - deep learning
    - machine learning
    - computer vision
    - embeddings

H. Couche outils et intégration

H1. MCP Gateway
H2. MCP Server Registry
H3. Internal Tools Registry
H4. API Tool Connectors
H5. Repo Operations Tooling
H6. Search Tooling
H7. Execution Tooling

Rôle :

- connecter les agents aux capacités du système
- standardiser l’accès aux outils
- éviter la duplication des outils dans chaque projet

I. Couche sécurité, protection et filtrage

I1. DLP Filter
I2. Secrets Scrubber
I3. Semantic Output Filter
I4. Access Policy Enforcement
I5. Namespace Isolation Controls
I6. Network Egress Controls
I7. Zero Trust Access Controls

Rôle :

- empêcher les fuites de secrets
- contrôler les flux sortants
- imposer des politiques de sécurité avant toute interaction externe

---

J. Couche d'exposition externe contrôlée

J1. SAS Paranoïaque
J2. Search Fog / Shadow Query Engine
J3. Ephemeral Scraping Jobs
J4. Content Sanitizer
J5. DOM Destroyer / Markdown Extractor
J6. Proxy / anonymity layer
J7. External Source Intake

Rôle :

- interagir avec l'extérieur sans exposer le cœur du système
- anonymiser et dégrader la corrélabilité des recherches
- extraire du contenu propre et stérilisé

K. Couche de validation de la connaissance externe

K1. Tribunal Sémantique
K2. Multi-source Consensus Engine
K3. Anti-AI-Sludge Filter
K4. Quarantine Queue
K5. Validation / arbitration layer

Rôle :

- auditer les données externes
- exiger plusieurs sources
- empêcher la pollution de la mémoire par du contenu synthétique ou douteux
- limiter le model collapse

L. Couche mémoire et connaissance

L1. PostgreSQL
L2. Qdrant / Vector DB
L3. Conversation Summaries
L4. Project State Tables
L5. User Profiles
L6. Knowledge Chunks
L7. Research Archive
L8. Specs Index
L9. Project Memory Index

Rôle :

- distinguer la mémoire structurée et la mémoire vectorielle
- conserver :
    - l'état
    - les résumés
    - la connaissance indexée
    - les recherches utiles
    - les documents de projet

M. Couche forges, code et sources de vérité

M1. Forgejo
M2. Repository Registry
M3. Branch Policies
M4. CI/CD Hooks
M5. Documentation Sources
M6. Guesdon-Brain
M7. Sophia-Brain
M8. Projects root

Rôle :

- stocker le code, les spécifications et les connaissances
- séparer :
    - moteur
    - sanctuaire
    - livrables

N. Couche observabilité et gouvernance

N1. Auto Log Session
N2. Audit Trail
N3. Token Consumption Logs
N4. Security Alerting
N5. License Compliance Logs
N6. Metrics / Traces
N7. Dashboard / SOC View

Rôle :

- fournir une traçabilité absolue
- observer l'usage
- auditer les actions
- suivre les coûts et événements de sécurité

O. Couche déploiement et industrialisation

O1. OKD / Kubernetes runtime
O2. Namespace-as-a-Border model
O3. LVM service isolation
O4. Encrypted volumes (LUKS)
O5. Kustomize / Helm packaging
O6. Tenant namespaces
O7. Deployment profiles

Rôle :

- permettre le déploiement reproductible
- industrialiser l'architecture
- gérer plusieurs profils :
    - personnel
    - PME
    - cluster souverain

---

3. Schéma de relation simplifié entre modules