Vasic Digital

Ingénierie logicielle native AI · Portfolio fondé sur des preuves

La famille de produits Helix, la flotte utilitaire vasic-digital et la chaîne d'outils Server Factory — des systèmes de développement AI conçus pour inspirer confiance, bâtis sur une base d'ingénierie Constitution commune.

140+	43	25	439
FLOTTE DE DÉPÔTS	FOURNISSEURS LLM	DISTRIBUTIONS LINUX PROVISIONNÉES	TESTS RÉUSSIS · MAIL SERVER FACTORY

1. Vue d’ensemble — d’abord le tableau complet

Il s’agit d’un portfolio unifié et unique utilisé à la fois par vasic.digital et milosvasic.ru. Il ne décrit pas une simple dispersion de projets annexes, mais une **flotte** délibérément conçue : des **applications produits de grande envergure** reposant sur **des dizaines de modules petits, découplés et testés de manière indépendante**, le tout régi par un **Constitution** d’ingénierie partagé et validé par une discipline de **QA anti-bidon**. Cette structure fait la différence. La plupart des portfolios se contentent d’énumérer ce qui a été construit ; ici, il s’agit d’un système où chaque produit est assemblé à partir de composants éprouvés et réutilisables, où chaque composant respecte les mêmes règles non négociables, et où chaque fonctionnalité annoncée s’appuie sur des preuves tangibles. Le langage dominant est le **Go**, avec Kotlin/KMP, TypeScript/React, Python, Swift et Shell utilisés là où chacun s’avère le plus pertinent — Go pour les services et bibliothèques à haut débit, Kotlin pour le provisionnement et les applications mobiles multiplateformes, TypeScript pour les interfaces typées, Python pour l’intégration AI/ML.

Ce qui donne sa cohérence à cet ensemble, c’est que la discipline est mécanique, et non simplement aspirationnelle. Un **Constitution** partagé est déployé sous forme de sous-module Git et hérité par une flotte de plus de 140 dépôts, si bien qu’une seule modification de règle se propage partout ; une couche de QA anti-bidon refuse d’enregistrer un succès sans preuve d’exécution. La famille de produits **Helix**, la flotte d’utilitaires et la chaîne d’outils **Server Factory** ne sont que trois expressions d’une même idée : construire une fois, réutiliser partout, et prouver que ça fonctionne avant de déclarer le travail terminé.

Tout ce qui suit est présenté par ordre de priorité :

1. **Piliers de gouvernance et de QA** — HelixConstitution, HelixQA (la discipline qui rend le reste digne de confiance).
2. **La famille de produits Helix** — le cycle de développement AI (HelixTrack en premier).
3. **Infrastructure LLM** — abstraction des fournisseurs, orchestration, vérification.
4. **Utilitaires vasic-digital** — outils autonomes de niveau professionnel.
5. **Server Factory** — héritage d’automatisation de l’infrastructure (classé en dernier).

La thèse unificatrice : **une fonctionnalité n’est terminée que lorsqu’un utilisateur réel peut l’utiliser et qu’il existe des preuves tangibles pour le démontrer.**

---

2. Piliers de gouvernance et de QA

Ces deux éléments viennent en premier, car tout le reste de ce portfolio emprunte sa crédibilité à eux. Ensemble, ils transforment le *« faites-moi confiance, ça marche »* en un fait vérifiable — le **Constitution** encode les règles, et le **HelixQA** prouve qu'elles ont été respectées.

- **HelixConstitution** — un recueil de règles d'ingénierie universel et indépendant des projets, déployé sous forme de sous-module Git et hérité par une flotte de plus de 140 dépôts. Portes de validation anti-bidon, immunité aux faux positifs, sécurité des données et des hôtes, discipline de couverture ; héritage extensible mais jamais affaibli ; portes de propagation vérifiant la présence des clauses requises dans toute la flotte ; chaque porte couplée à un test de mutation prouvant qu'elle n'est pas une supercherie.
- **HelixQA** — orchestration de QA anti-bidon (Go). Banques de tests **YAML** écrites, ainsi que des sessions de QA entièrement autonomes combinant **LLM** et vision par ordinateur sur Android, Android TV, Web et Desktop ; aucun succès enregistré sans preuve tangible (captures d'écran, logcat, vidéo, traces de pile). Type de test QA imposé par le **Constitution** (§11.4.169).

3. La famille de produits Helix

La gamme Helix en constitue le fer de lance : une famille de produits interconnectés couvrant l'intégralité du cycle de développement AI, de la planification et la spécification à la construction, la mémoire, la traduction et la livraison. Chacun est un produit à part entière, mais leur conception vise à ce qu'ils s'assemblent – mêmes règles de gouvernance, mêmes modules réutilisables, même rigueur probatoire sous-jacente.

- **HelixTrack** — une alternative JIRA pour le monde libre ; produit phare de la ligne Helix-Track.
- **HelixAgent** — service d'ensemble LLM : plusieurs modèles débattent et livrent la réponse sur laquelle ils s'accordent, avec une sélection des fournisseurs basée sur la vérification.
- **HelixCode** — plateforme de développement AI distribuée de niveau entreprise ; répartit le travail entre des agents gérés par SSH avec points de contrôle et retour arrière automatiques ; interfaces REST, CLI, TUI et MCP.
- **HelixCluster** — système d'exploitation distribué pour le calcul AI, des GPU de datacenter aux appareils mobiles en périphérie, sous une seule et même couche de contrôle.
- **HelixBuilder** — pipeline alimenté par AI pour la construction d'applications, catégorie par catégorie.
- **HelixSkills** — système de compétences gouverné et adossé à une constitution pour les agents CLI AI (compétences, serveurs d'outils MCP, plugins Code Claude).
- **HelixSpecifier** — développement piloté par les spécifications, dont le formalisme s'adapte à l'ampleur du travail.
- **HelixMemory** — cerveau mémoire unique pour les agents AI, fusionnant quatre moteurs de pointe.
- **HelixTranslate** — traduction de livres par modèles vérifiés ; conçue pour être transparente, sans repli silencieux.
- **HelixTerminator** — plateforme de terminal à confiance zéro : chaque session SSH sécurisée, partagée et assistée par AI.
- **HelixGitpx** — Git fédéré sur une douzaine d'hébergeurs ; une seule source de vérité, répliquée partout.

- **HelixOTA** — mises à jour universelles et découplées par voie hertzienne ; conçues pour éviter tout blocage.
- **HelixPlay** — transforme n'importe quelle machine GPU en appareil de cloud gaming personnel.
- **Helix-Flow** — produit d'inférence pour la plateforme Helix. *NON VÉRIFIÉ / bloqué en attente de la source : son dépôt public ne contient actuellement qu'un fichier README d'une ligne ; présenté en détail uniquement lorsque la documentation réelle existera.*

4. Infrastructure LLM

Sous les produits se trouve le socle qui les rend indépendants des fournisseurs et fiables : une interface unique pour des dizaines de fournisseurs LLM, un plan de contrôle pour les agents de codage sans interface, et une source de vérité unique pour la vérification. C'est cette couche qui permet à tout ce qui se trouve au-dessus de changer de modèles, de survivre aux pannes des fournisseurs et de refuser de faire confiance à un LLM incapable de prouver qu'il comprend la tâche.

- **HelixLLM** — un seul binaire, six modes : inférence compatible OpenAI et Anthropic sur HTTP/3, llama.cpp local, chaîne de repli notée, pipeline RAG, agents ReAct.
- **LLMProvider** — une interface unique, 43 fournisseurs, avec disjoncteurs, nouvelles tentatives et surveillance intégrée.
- **LLMOrchestrator** — un plan de contrôle pour chaque agent de codage CLI sans interface (OpenCode, Code Claude, Gemini, Junie, Code Qwen).
- **LLMsVerifier** — vérifier, surveiller, optimiser : la source unique de vérité pour les métadonnées LLM/fournisseur/vérification, avec une porte obligatoire de compréhension des modèles.

5. utilitaires vasic-digital

Des outils autonomes et prêts pour la production, chacun résolvant un problème complexe à part entière — et, ce n'est pas un hasard, mettant à l'épreuve la flotte de modules réutilisables avec exigence. Ils vont d'un système multimédia résilient et multi-protocoles à une chaîne de production de cours vidéo à partir de Markdown, en passant par un moteur de synchronisation de documents et de bases de données par hachage de contenu. Plusieurs affichent sans détour leur niveau de maturité, et ces indicateurs sont maintenus, non dissimulés.

- **Catalogizer** — gestion de collections multimédias multi-protocoles (SMB/FTP/NFS/WebDAV/local), chiffrée et auto-hébergeable ; interfaces Go/Gin API + React ; surveillance résiliente et tolérante aux coupures ; construit sur 21 sous-modules `digital.vasic.*`.
- **Courses-Creator** — chaîne de production de cours vidéo à partir de Markdown ; enrichissement multi-LLM, TTS (Bark/SpeechT5), lecteurs pour bureau/mobile/web ; mode dégradé sans clé API.
- **VisionEngine** — boîte à outils Go découplée fusionnant la vision par ordinateur classique avec des modèles LLM multi-fournisseurs ; graphes de navigation avec export BFS + DOT/JSON/Mermaid ; compilation conditionnelle avec balises OpenCV.
- **DocProcessor** — cartographie des fonctionnalités à partir de la documentation avec suivi de la couverture de vérification ; extraction LLM ou heuristique/hors ligne ; licence Apache 2.0.

- **Docs Chain** — synchronisation bidirectionnelle et atomique de documents et de bases de données par hachage de contenu (recalcul incrémental de type Salsa sur un DAG). *Phases 1 à 5 VALIDÉES ; Phases 6 et 7 PRÉVUES.*
- **Herald** — notifications multi-canaux fiables avec résolution d'intention en trois niveaux (commande → LLM → clarification) en langage naturel ; premier consommateur de Docs Chain.
- **task_bridge** — synchronisation bidirectionnelle et découplée des tâches et tableaux (SQLite SSoT ? docs ? ClickUp). *Ébauche P1 — logique de synchronisation non encore implémentée.*
- **Vasic Digital Suite de modules réutilisables** — la « bibliothèque standard » `digital.vasic.*` : primitives d'infrastructure, blocs de construction AI et garde-fous défensifs LLM, ainsi qu'un miroir Kotlin Multiplatform. *Plusieurs dépôts de l'organisation sont en ÉBAUCHE/EN COURS — signalés, non livrés.*

6. Server Factory (héritage de l'automatisation d'infrastructure — classé en dernier)

Classé en dernier par choix, et non par qualité : la chaîne d'outils Server Factory précède la lignée AI et montre où la philosophie « construire une fois, réutiliser partout » a d'abord pris forme. Son produit phare — un serveur de messagerie que l'on décrit en JSON et que l'on provisionne n'importe où — est un outil mature et éprouvé ; les outils complémentaires sont présentés dans leur état réel de maturité, sans fard.

- **Mail Server Factory** — serveur de messagerie entièrement provisionné et conteneurisé à partir d'une description JSON, compatible avec 12 types de connexions et 25 distributions Linux ; sécurité niveau entreprise ; 439 tests réussis et validation SonarQube sans faille. Produit phare de l'organisation Server-Factory.
- **Server Factory Cadre de base** — le moteur Kotlin partagé sur lequel s'appuient toutes les usines.
- **Qemu-Utils** — gestion d'images de machines virtuelles QEMU comme des artefacts : téléchargement/mise en cache/exécution, compression/publication, réseau pont/TAP, installations ISO ; Linux + macOS.
- **Parallels-Utils** — compression, publication et récupération d'images de machines virtuelles Parallels (macOS) via des fichiers de configuration simples.
- **Server Factory — Composants supplémentaires** — usines de services (Web/SonarQube/Proxy de cache), packs de définitions et utilitaires. *Les usines de services sont documentées à titre indicatif — NON VÉRIFIÉES / stade précoce.*

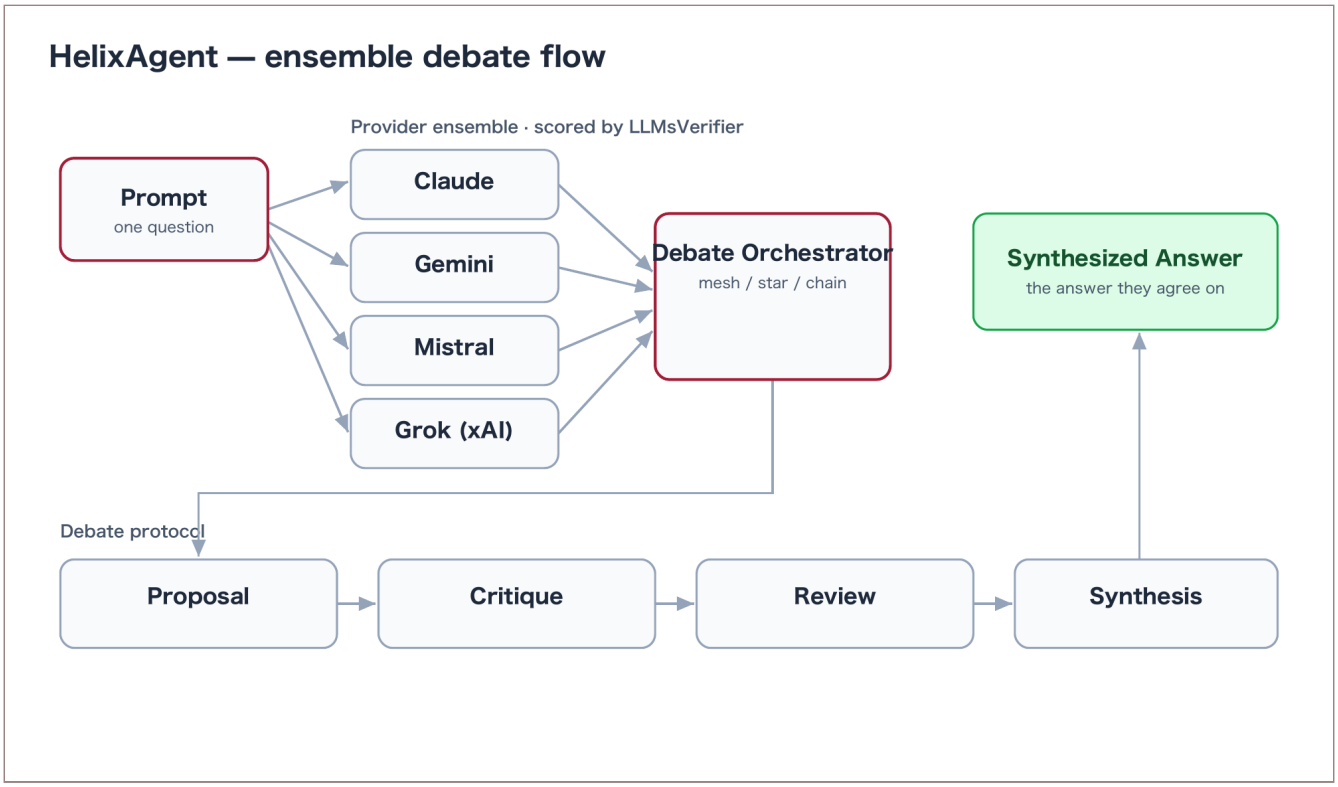
---

7. Index technologique (basé sur des preuves)

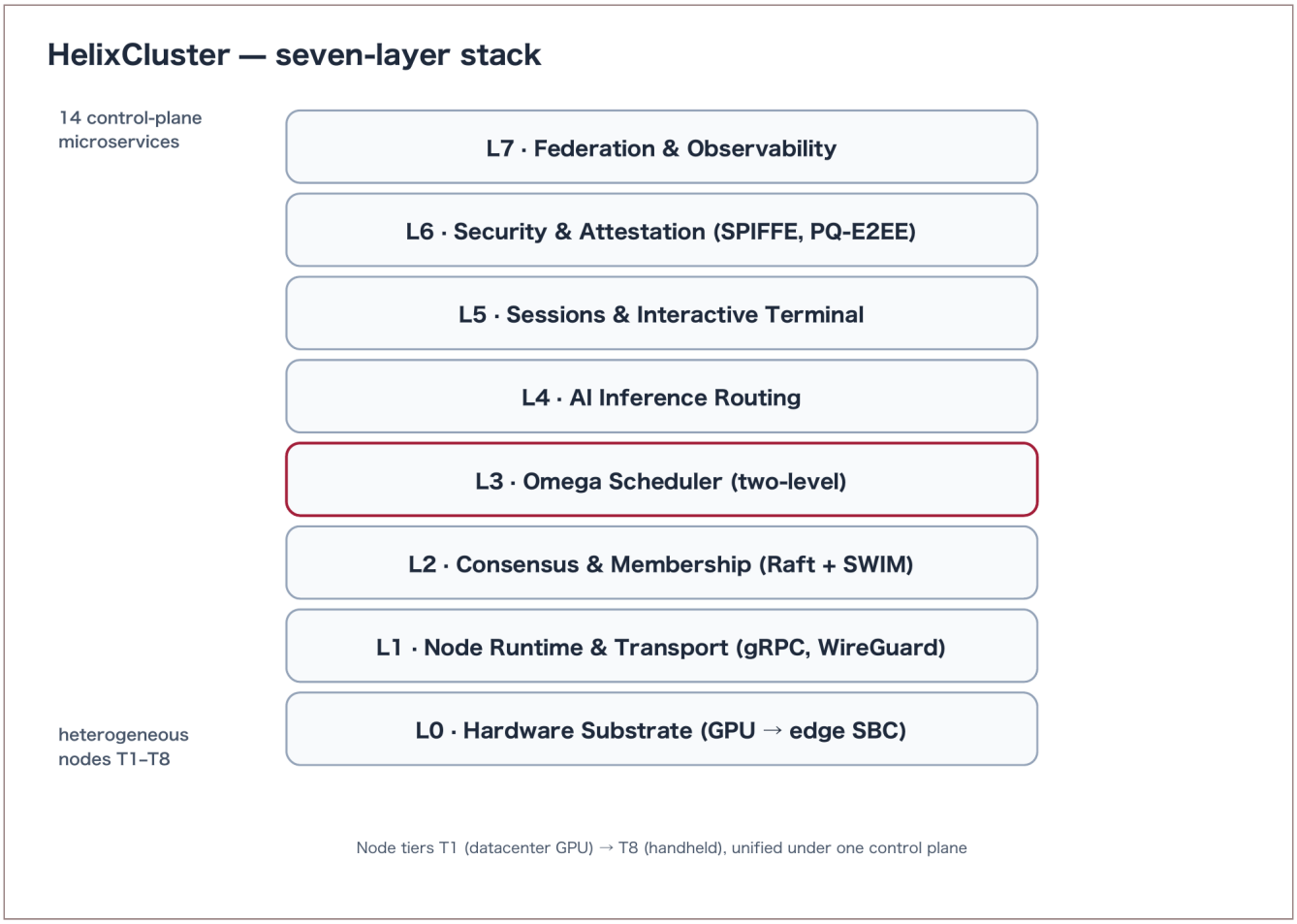
- **Langues :** Go (dominante), Kotlin & Kotlin Multiplatform, TypeScript, Python, Swift, Shell ; PL/pgSQL ; TLA+ (spécifications formelles dans helix_cluster).
- **AI / LLM :** accès à plus de 40 fournisseurs, MCP, RAG, bases de données vector & embeddings, planification (HiPlan/MCTS/Tree-of-Thoughts), LLMOps, évaluation comparative (SWE-bench/HumanEval/MMLU), TTS (Bark/SpeechT5), vision par ordinateur + LLM-vision, garde-fous/red-team.
- **Backend :** Gin, gRPC + Protobuf, HTTP/3 (QUIC), WebSockets, Angular, React, Kafka/RabbitMQ.

- **Données :** PostgreSQL, SQLite, SQLCipher, Redis, Neo4j, ClickHouse, MinIO/S3/GCS/Azure.
- **Infrastructure / DevOps :** Docker & Compose, Kubernetes + Helm, Prometheus + Grafana, OpenTelemetry, QEMU/Libvirt/Parallels ; GitHub Actions, Gradle, Make.
- **Tests / AQ :** HelixQA, harnais de défis avec portes de mutation, `go test -race`, outils de régression visuelle, tests sur appareils ADB, SonarQube, analyse de sécurité (semgrep/gosec/trivy/snyk/gitleaks/nancy), vérification de modèles TLA+.

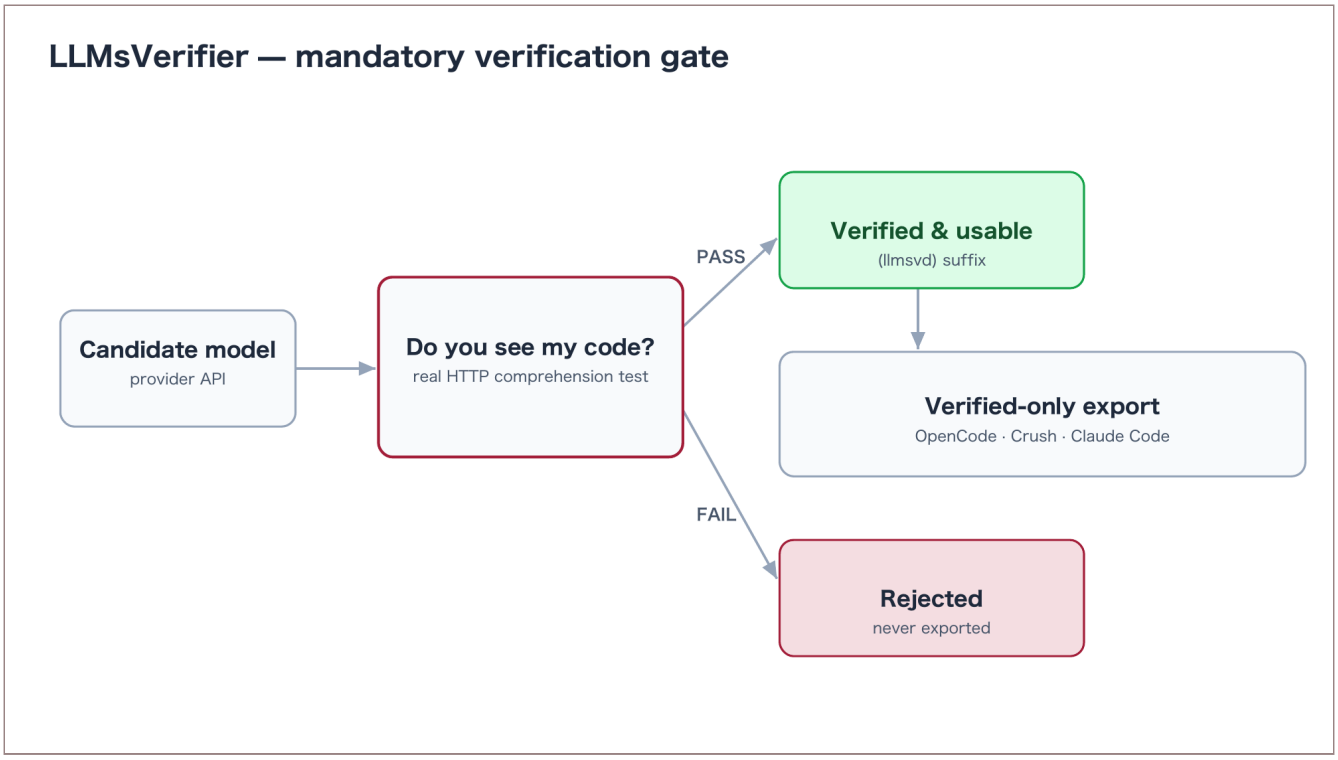
Architecture sélectionnée



HelixAgent — un service LLM en ensemble où les fournisseurs débattent selon un protocole en quatre étapes et livrent la réponse sur laquelle ils s'accordent, notée par LLMsVerifier.



HelixCluster — un système d'exploitation distribué pour le calcul AI, des GPU de datacenter aux appareils mobiles en périphérie, sous une seule interface de contrôle.



LLMsVerifier — vérifier, surveiller, optimiser : la source unique de vérité pour les métadonnées LLM/fournisseur/vérification, soumise à un contrôle obligatoire de compréhension des modèles.