VASIC DIGITAL · EVIDENCE-BASED PORTFOLIO

Miloš Vasić

Portfolio · Helix family · vasic-digital · Server Factory

A unified, evidence-based portfolio — the Helix product family, the vasic-digital utility fleet, and the Server Factory toolchain, on a shared engineering Constitution.

milosvasic.ru · vasic.digital · github.com/HelixDevelopment · github.com/vasic-digital

140+	43	25	439
REPOSITORY FLEET	LLM PROVIDERS	LINUX DISTROS PROVISIONED	PASSING TESTS · MAIL SERVER FACTORY

1. Überblick – das große Ganze zuerst

Dies ist ein einziges, einheitliches Portfolio, das sowohl von vasic.digital als auch von milosvasic.ru genutzt wird. Es beschreibt keine verstreuten Nebenprojekte, sondern eine gezielt konstruierte **Flotte**: **große Produktanwendungen**, die auf **Dutzenden kleiner, entkoppelter und unabhängig getesteter Module** aufbauen – das Ganze gesteuert von einem gemeinsamen **Constitution**-Engineering und überprüft durch eine **Anti-Bluff-QS**-Disziplin. Diese Struktur ist der entscheidende Unterschied. Die meisten Portfolios sind eine Auflistung von Dingen, die gebaut wurden; dies hier ist ein System, in dem jedes Produkt aus bewährten, wiederverwendbaren Komponenten zusammengesetzt wird, jede Komponente denselben unabdingbaren Regeln unterliegt und jede beworbene Funktion durch nachweisbare Belege gestützt wird. Die Hauptsprache ist **Go**, ergänzt durch Kotlin/KMP, TypeScript/React, Python, Swift und Shell – jeweils dort eingesetzt, wo sie am besten passen: Go für Hochdurchsatz-Dienste und Bibliotheken, Kotlin für Provisionierung und plattformübergreifende Mobile-Entwicklung, TypeScript für typisierte Frontends, Python als AI/ML-Bindeglied.

Was diesen Werkkorpus zusammenhält, ist die Tatsache, dass die Disziplin mechanisch und nicht bloß ambitioniert ist. Ein gemeinsames **Constitution** wird als Git-Submodul ausgeliefert und über eine Flotte von über 140 Repositories vererbt, sodass eine einzige Regeländerung überall greift; eine Anti-Bluff-QS-Schicht verweigert die Freigabe ohne Laufzeitnachweis. Die **Helix**-Familie, die Utility-Flotte und die **Server Factory**-Toolchain sind drei Ausprägungen ein und derselben Idee: einmal bauen, überall wiederverwenden und nachweisen, dass es funktioniert, bevor es als fertig gilt.

Alles Folgende ist nach Priorität geordnet:

1. **Governance- & QS-Grundpfeiler** – HelixConstitution, HelixQA (die Disziplin, die alles Übrige vertrauenswürdig macht).
2. **Die Helix-Produktfamilie** – der AI-Entwicklungszyklus (HelixTrack zuerst).
3. **LLM-Infrastruktur** – Provider-Abstraktion, Orchestrierung, Verifizierung.
4. **vasic-digital Utilities** – produktionsreife Standalone-Tools.
5. **Server Factory** – Infrastrukturautomatisierungs-Herkunft (an letzter Stelle).

Die verbindende These: **Eine Funktion gilt erst dann als fertig, wenn ein echter Nutzer sie verwenden kann und es nachweisbare Belege dafür gibt.**

2. Governance- & QS-Grundpfeiler

Diese beiden Punkte stehen an erster Stelle, denn alles andere in diesem Portfolio bezieht seine Glaubwürdigkeit von ihnen. Gemeinsam verwandeln sie *„Vertrau mir, es funktioniert"* in eine überprüfbare Tatsache – das **Constitution** kodifiziert die Regeln, **HelixQA** beweist, dass sie eingehalten wurden.

- **HelixConstitution** – ein universelles, projektunabhängiges Engineering-Regelwerk, das als Git-Submodul ausgeliefert und über eine Flotte von über 140 Repositories vererbt wird. Anti-Bluff-Nachweisschranken, Immunität gegen falsch-positive Ergebnisse, Datensicherheit, Testabdeckung; Vererbung mit Erweiterungs-, aber ohne Abschwächungsmöglichkeit; Ausbreitungskontrollen, die flächendeckend nach erforderlichen Klauseln suchen; jede Schranke wird durch einen Mutationstest ergänzt, der beweist, dass sie kein Schein ist.
- **HelixQA** – Anti-Bluff-QS-Orchestrierung (Go). Geschriebene **YAML**-Testbatterien plus vollautonome **LLM**- und Computer-Vision-QS-Sitzungen auf Android, Android TV, Web und Desktop; keine Freigabe ohne erfasste Nachweise (Screenshots, Logcat, Video, Stack Traces). Die vom **Constitution** vorgeschriebene QS-Testart (§11.4.169).

3. Die Helix-Produktfamilie

Die Helix-Reihe ist das Flaggschiff: eine ineinandergreifende Produktfamilie, die den gesamten AI-Entwicklungszyklus abdeckt – von Planung und Spezifikation über Bau, Speicherung, Übersetzung bis hin zur Auslieferung. Jedes Produkt ist für sich genommen ein vollwertiges Angebot, doch die Konzeption zielt darauf ab, dass sie sich nahtlos zusammenfügen: dieselbe Governance, dieselben wiederverwendbaren Module, dieselbe disziplinierte Nachweisführung – allesamt unter einem Dach.

- **HelixTrack** – eine JIRA-Alternative für die freie Welt; Flaggschiff der Helix-Track-Linie.
- **HelixAgent** – Ensemble-LLM-Dienst: Mehrere Modelle diskutieren und liefern die Antwort, auf die sie sich einigen, mit verifizierungsbasierter Anbieterauswahl.
- **HelixCode** – unternehmensgerechte, verteilte AI-Entwicklungsplattform; verteilt Aufgaben auf SSH-gesteuerte Worker mit automatischer Checkpoint-/Rollback-Funktion; REST-, CLI-, TUI- und MCP-Schnittstellen.
- **HelixCluster** – ein verteiltes Betriebssystem für AI-Computing, von Rechenzentrums-GPUs bis zu Edge-Handhelds, gesteuert über eine einzige Kontrollebene.
- **HelixBuilder** – eine von AI angetriebene Pipeline zum Aufbau von Anwendungen, Kategorie für Kategorie.
- **HelixSkills** – ein reguliertes, verfassungsgestütztes Kompetenzsystem für CLI-AI-Agenten (Skills, MCP-Toolserver, Claude-Code-Plugins).
- **HelixSpecifier** – spezifikationsgetriebene Entwicklung, deren Formalitäten sich am Arbeitsumfang orientieren.
- **HelixMemory** – ein gemeinsames Gedächtnis für AI-Agenten, das vier erstklassige Engines vereint.
- **HelixTranslate** – verifizierte Modellübersetzung von Büchern; von Grund auf transparent, ohne stille Ausweichlösungen.
- **HelixTerminator** – eine Zero-Trust-Terminalplattform: Jede SSH-Sitzung ist gesichert, teilbar und AI-gestützt.

- **HelixGitpx** – föderiertes Git über ein Dutzend Hosts hinweg; eine einzige Quelle der Wahrheit, überall gespiegelt.
- **HelixOTA** – universelle, entkoppelte Over-the-Air-Updates; von Grund auf ausfallsicher.
- **HelixPlay** – verwandelt jeden GPU-Rechner in Ihr persönliches Cloud-Gaming-Gerät.
- **Helix-Flow** – Helix-Plattform-Inferenzprodukt. *UNVERIFIZIERT / blockiert durch Quelle: Das öffentliche Repository enthält derzeit nur eine einzeilige README; wird erst bei Vorliegen echter Dokumentation in voller Produkttiefe dargestellt.*

4. LLM-Infrastruktur

Unter den Produkten liegt die Grundlage, die sie anbieterunabhängig und zuverlässig macht: eine einzige Schnittstelle für Dutzende LLM-Anbieter, eine Kontrollebene für kopflose Coding-Agenten und eine einzige Verifizierungsquelle der Wahrheit. Diese Schicht ermöglicht es allem darüber, Modelle auszutauschen, Anbieterausfälle zu überstehen und keinem LLM-Anbieter zu vertrauen, der nicht nachweisen kann, dass er die Aufgabe versteht.

- **HelixLLM** – ein Binary, sechs Modi: OpenAI- und Anthropic-kompatible Inferenz über HTTP/3, lokale llama.cpp, bewertete Fallback-Kette, RAG-Pipeline, ReAct-Agenten.
- **LLMProvider** – eine Schnittstelle, 43 Anbieter, mit integrierten Schutzschaltern, Wiederholungsversuchen und Gesundheitsprüfungen.
- **LLMOrchestrator** – eine Kontrollebene für jeden kopflosen CLI-Coding-Agenten (OpenCode, Claude Code, Gemini, Junie, Qwen Code).
- **LLMsVerifier** – verifizieren, überwachen, optimieren: die einzige Quelle der Wahrheit für LLM-/Anbieter-/Verifizierungsmetadaten, mit obligatorischer Modellverständnisprüfung.

5. vasic-digital Utilities

Eigenständige, produktionsreife Werkzeuge, die jeweils ein komplexes Problem für sich lösen – und nebenbei die wiederverwendbare Modulbibliothek unter realen Bedingungen auf die Probe stellen. Sie reichen von einem robusten Multi-Protokoll-Mediasystem über eine Markdown-zu-Video-Kurs-Pipeline bis hin zu einer inhaltsbasierten Dokumenten-/Datenbank-Synchronisationsengine; einige machen keinen Hehl aus ihrem Entwicklungsstand, und diese Hinweise werden offen kommuniziert, nicht verschleiert.

- **Catalogizer** – Multi-Protokoll (SMB/FTP/NFS/WebDAV/lokal), verschlüsselt, selbsthostbares Medienverwaltungssystem; Go/Gin API + React UI; ausfallsichere Offline-Überwachung; basierend auf 21 `digital.vasic.*`-Submodulen.
- **Courses-Creator** – Markdown-zu-Video-Kurs-Pipeline; Multi-LLM-Anreicherung, TTS (Bark/SpeechT5), Player für Desktop/Mobil/Web; funktionierender Modus ohne API-Schlüssel.
- **VisionEngine** – Entkoppeltes Go-Toolkit, das klassische Computer Vision mit Multi-Anbieter-LLM-Bilderkennung verbindet; Navigationsgraphen mit BFS + DOT/JSON/Mermaid-Export; OpenCV mit Build-Tag-Steuerung.
- **DocProcessor** – Dokumentation-zu-Feature-Mapping mit Verifizierungsabdeckungs-Tracking; LLM- oder heuristische/Offline-Extraktion; Apache-2.0-Lizenz.

- **Docs Chain** – Inhaltsbasierte, bidirektionale, atomare Dokumenten-/Datenbank-Synchronisation (Salsa-ähnliche inkrementelle Neuberechnung über einen DAG). *Phasen 1–5 GRÜN; Phasen 6–7 IN PLANUNG.*
- **Herald** – Zuverlässige Multi-Kanal-Benachrichtigungen mit natürlichsprachlicher Drei-Stufen-Intent-Auflösung (Befehl → LLM → Klärung); erster Docs Chain-Nutzer.
- **task_bridge** – Entkoppelte, bidirektionale Aufgaben-/Board-Synchronisation (SQLite Single Source of Truth ? Dokumente ? ClickUp). *P1-Grundgerüst – Synchronisationslogik noch nicht implementiert.*
- **Vasic Digital – Wiederverwendbare Modulbibliothek** – Die `digital.vasic.*`-„Standardbibliothek": Infrastruktur-Grundbausteine, AI-Komponenten und defensive LLM-Sicherheitsvorkehrungen, plus ein Kotlin Multiplatform-Spiegel. *Mehrere Organisations-Repos sind IM AUFBAU/IN ARBEIT – gekennzeichnet, nicht veröffentlicht.*

6. Server Factory (Infrastrukturautomatisierung – historisch gewachsen, bewusst an letzter Stelle)

An letzter Stelle – nicht wegen mangelnder Qualität, sondern aus Prinzip: Die Server Factory-Toolchain entstand vor der AI-Linie und zeigt, wo die Philosophie „Einmal bauen, überall wiederverwenden" erstmals Gestalt annahm. Ihr Flaggschiff – ein Mailserver, den man in JSON beschreibt und überall bereitstellt – ist ein ausgereiftes, gründlich getestetes Produkt; die begleitenden Komponenten werden in ihrem tatsächlichen, unterschiedlichen Reifegrad präsentiert, statt beschönigt.

- **Mail Server Factory** – Deklarative JSON → vollständig provisionierter, dockerisierter Mailserver für 12 Verbindungstypen und 25 Linux-Distributionen; Unternehmenssicherheit; meldet 439 bestandene Tests und eine saubere SonarQube-Prüfung. Flaggschiff der Server-Factory-Organisation.
- **Server Factory Core Framework** – Die gemeinsame Kotlin-Engine, auf der alle Factories aufbauen.
- **Qemu-Utils** – QEMU-VM-Images wie Artefakte verwalten: herunterladen/zwischenspeichern/ausführen, komprimieren/veröffentlichen, Bridge/TAP-Netzwerke, ISO-Installationen; Linux + macOS.
- **Parallels-Utils** – Parallels (macOS)-VM-Image-Komprimierung, -Veröffentlichung und -Abruf über einfache Konfigurationsdateien.
- **Server Factory – Zusätzliche Komponenten** – Service-Factories (Web/SonarQube/Caching-Proxy), Definitions-Pakete und Hilfsprogramme. *Service-Factories sind vorläufig dokumentiert – UNGEPRÜFT / frühe Entwicklungsphase.*

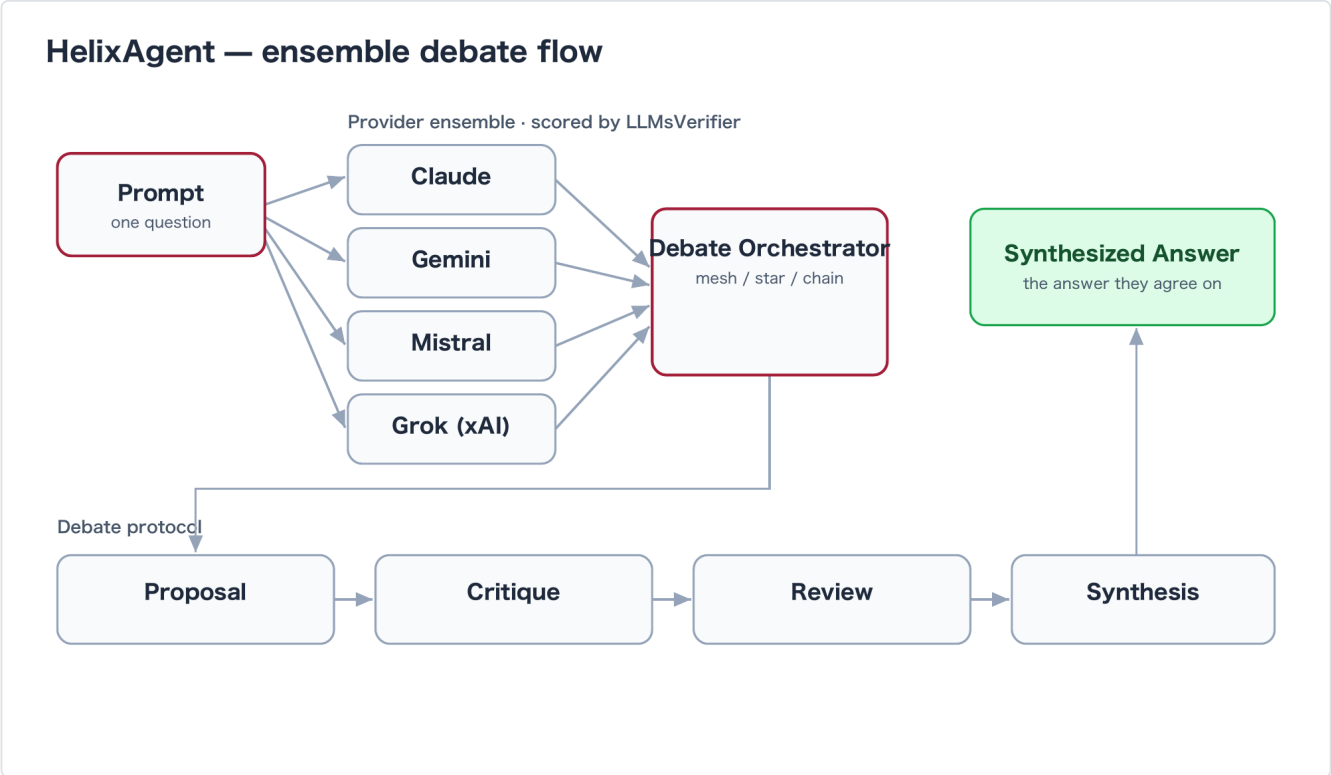
---

7. Technologieindex (evidenzbasiert)

- **Sprachen:** Go (dominant), Kotlin & Kotlin Multiplatform, TypeScript, Python, Swift, Shell; PL/pgSQL; TLA+ (formale Spezifikationen in helix_cluster).
- **AI / LLM:** 40+ Anbieterzugänge, MCP, RAG, vector-Datenbanken & embeddings, Planung (HiPlan/MCTS/Tree-of-Thoughts), LLMOps, Benchmarking (SWE-bench/HumanEval/MMLU), TTS (Bark/SpeechT5), Computer-Vision + LLM-Vision, Schutzmechanismen/Red-Team.
- **Backend:** Gin, gRPC + Protobuf, HTTP/3 (QUIC), WebSockets, Angular, React, Kafka/RabbitMQ.
- **Daten:** PostgreSQL, SQLite, SQLCipher, Redis, Neo4j, ClickHouse, MinIO/S3/GCS/Azure.

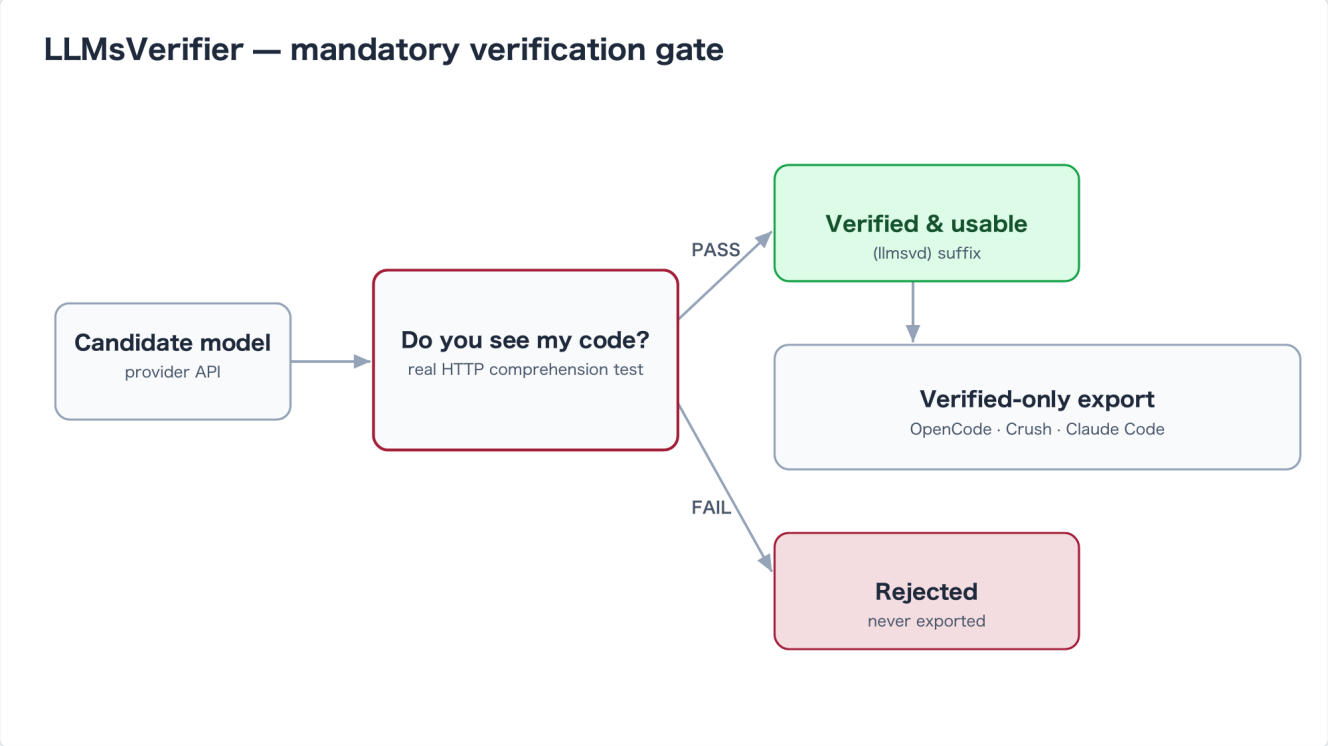
- **Infrastruktur / DevOps:** Docker & Compose, Kubernetes + Helm, Prometheus + Grafana, OpenTelemetry, QEMU/Libvirt/Parallels; GitHub Actions, Gradle, Make.
- **Testing / QS:** HelixQA, Challenge-Harnesses mit Mutations-Gates, `go test -race`, visuelle Regressions-Toolings, ADB-Gerätetests, SonarQube, Sicherheitsscans (semgrep/gosec/trivy/snyk/gitleaks/nancy), TLA+-Modellprüfung.

Selected architecture



HelixAgent — an ensemble LLM service where providers debate under a four-stage protocol and ship the answer they agree on, scored by LLMsVerifier.

HelixCluster — a distributed operating system for AI compute, from datacenter GPUs to edge handhelds, under one control plane.



LLMsVerifier — verify, monitor, optimize: the single source of truth for LLM/provider/verification metadata, gated by a mandatory model-comprehension check.