

AI Foundation

Arhitectură, contracte și portofoliu de
capabilități.

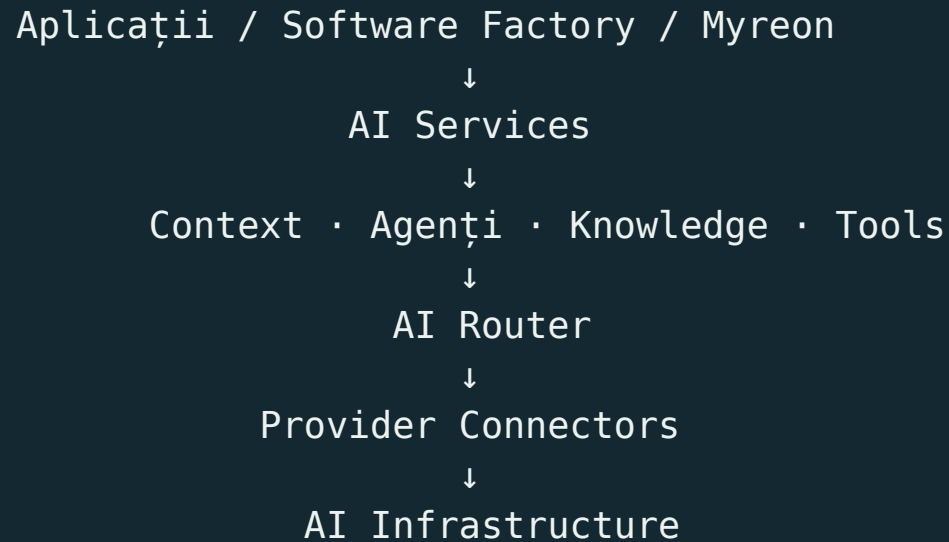
Prezentare tehnică · SoftKit · 10 septembrie 2026

Un serviciu independent care coordonează inteligența prin subsisteme cu
responsabilități explicite.



01 · Harta responsabilităților

Capabilitățile sunt contractul stabil dintre aplicații și platformă.



Security, Governance, Memory, Prompt și Observability susțin transversal execuția. Admin Console accesează separat Administration API.

02 · Organizare și extensibilitate

Monolit modular, serviciu independent, contracte explicite.

Kernel deține bootstrap, configurația de pornire, registrele interne și lifecycle-ul.

Modulele publică servicii prin chei tipizate și declară dependențele.

Domain deține invariantele; Application orchestrează; Repository persistă.

HTTP/gRPC traduc cereri și rezultate; composition root assemblează implementările.

Conectorii, instrumentele și backend-urile se extind în spatele contractelor existente.

03 · Trei niveluri de vocabular

Catalogul produsului, tipurile de rutare și serviciile publice au roluri diferite.

NIVEL	EXAMPLE
Capabilități de platformă	general-chat, software-development, embeddings, reranking, retrieval
Modalități și procesare	OCR, document-analysis, image-understanding, speech-to-text, text-to-speech
Creație și raționament	image-generation, video-understanding, reasoning, planning, simulation
Tipuri Model Management / Router	chat, completion, tool-use, vision, embedding
Servicii pentru aplicații	Chat, Completion, Summarization, Extraction, Agent, Knowledge, Batch, Feedback

Tabelul separă nivelurile conceptuale; nu declară o mapare nouă între enum-uri sau rute API.

04 · Suprafața AI Services

Aplicațiile se integrează prin API-uri versionate și SDK-uri.

SERVICIU	CONTRACT / FAMILIE DE RUTE
Chat	sesiuni, SendMessage, StreamMessage · /v1/chat
Completion	Complete, Stream · /v1/completion
Summarization	Summarize · /v1/summarize
Extraction / Classification	Extract, Classify · /v1/extract, /v1/classify
Agent / Knowledge	execuții de agenți și căutare · contractele de serviciu
Batch / Feedback	/v1/batch, /v1/feedback

Tenant-ul se rezolvă din Principal. TraceID corelează răspunsul; apelurile de model includ Usage. Erorile au o taxonomie comună.

05 · Flux de conversație cu knowledge

Fiecare subsistem contribuie prin propriul contract.

```
Principal → autentificare → autorizare AI Services
  → sesiune + solicitare
  → Context Engine
    ← Prompt Engine / Memory Engine / Knowledge Services
  → AI Router → Model Management → Provider Connector
  → filtrare rezultat → memorie → răspuns + Usage + TraceID
```

Knowledge furnizează fragmente cu proveniență. Context decide selecția și ordinea. Router decide dispatch-ul; providerul execută inferența.

06 · Kernel: bootstrap și lifecycle

Compoziția este explicită și ordinea execuției rezultă din dependențe.

Configurație încărcată și validată înainte de pornirea modulelor.

Înregistrare, rezolvarea dependențelor, Init și Start.

Health publicat de module și agregat de Kernel.

Oprire în ordine inversă, cu timeout pentru cleanup.

Barieră de oprire autorizată de composition root pentru admiterea durabilă a auditului.

Evenimente pentru bootstrap, stări și schimbări de health.

07 · Registre și rezoluție de modele

Ownership-ul registrelor păstrează semantica fiecărei informații.

Capability Registry asociază capacități cu implementări la nivel de platformă.

Service Registry urmărește servicii; registrul intern Kernel publică dependențe tipizate.

Model Management deține descriptori, aliasuri și rezoluția pentru Router.

Descriptorii exprimă provider, modalitate, context window, cost și capacități.

Aliasurile permit repoint controlat; deprecarea urmează announce → warn → block-new-usage → retire.

Registrele interne de Module, Tool și Prompt păstrează responsabilitățile subsistemelor respective.

08 · AI Router

Punctul unic de acces intern pentru invocarea modelelor.

Autorizare, rate limiting și rezervarea cotei de utilizare.

Rezoluție de alias și listă ordonată de fallback-uri eligibile.

Dispatch pentru completion, streaming și embeddings.

Retry cu backoff limitat și tratarea RetryAfter.

Normalizarea chunk-urilor, semnal terminal și erori de stream.

Propagarea TraceID, Usage și a rezultatului modelului folosit.

Router nu construiește context, nu păstrează conversația și nu execută instrumente.

09 · Provider Connectors

Integrarea de provider este izolată într-un adaptor înlocuibil.

Contract comun: Complete, Stream, Embed, Capabilities și HealthCheck.

Traducere de request/response și normalizare de erori.

CredentialSource și SecretClient furnizează credențialele.

Conector real, conector de test și instanțe self-hosted.

Streaming incremental și Usage raportat de sursă.

Instanțe configurate separat, cu Provider ID distinct și endpoint propriu.

Modelele, serverele de inferență și resursele GPU rămân responsabilitatea infrastructurii.

10 · Prompt Engine

Promptul publicat este un artefact versionat și evaluat.

Template cu segmente, variabile, condiții și indicii de model.

Versiuni imuabile și alias latest-stable.

Registry pentru authoring și lifecycle; Renderer pentru materializare.

Evaluator cu aserțiuni contains, regex și json_schema.

Publicarea este condiționată de testele de referință.

Providerul primește instrucțiuni rezolvate, fără template-uri de interpretat.

11 · Knowledge Services

RAG construiește și interoghează cunoaștere cu proveniență.

Source Connector → Document → Chunking
→ AI Router / embeddings → Vector Store
→ Search → ranking → Snippets → Context / AI Services

KnowledgeBase și surse au lifecycle și apartenență la tenant.

Ingestia, reindexarea și actualitatea surselor se urmăresc explicit.

Backend-ul vectorial este înlocuibil;
reprezentarea rămâne independentă de provider.

Retenția și ștergerea acoperă conținutul ingerat și reprezentările asociate.

12 · Memory Engine

Memoria păstrează experiența; knowledge păstrează informația curată și administrată.

Contract: Write, Read, Search, Compact și Forget.

Record cu tenant, scope, type, content, embedding și expiry.

Scope-uri de sesiune, agent, utilizator și organizație.

Backend de stocare înlocuibil, cu izolare și retenție.

Compactare și rezumare pentru continuitate în limite de context.

Accesul se realizează prin Engine; consumatorii nu accesează direct stocarea.

13 · Context Engine

Selecția contextului este separată de selecția modelului.

```
system → developer → retrieved knowledge  
      → memory → conversation → latest turn
```

AssembleContext întoarce AssembledContext.

Buget de tokeni pe secțiuni, ranking și selecție deterministă.

Trunchiere și compactare la depășirea bugetului.

Rezultatele instrumentelor participă la asamblare conform contractului.

Redactare și inspecție de securitate înainte de expunerea către model.

14 · Agent Runtime

O definiție de agent devine o execuție cu stare și limite.

```
Agent Definition → Run
  → Plan [Context + Router]
  → Act [Tool Runtime]
  → Observe [rezultat + memorie]
  → pas următor / aprobare / terminare / anulare
```

AllowedTools, AllowedModels, MaxSteps și condițiile de terminare delimitează execuția. Checkpoint-urile permit intervenție umană și reluare la limitele definite de contract.

15 · Tool Runtime

Un instrument este o capabilitate declarată, autorizată și verificabilă.

1. Rezolvarea Tool Definition și verificarea permisiunilor.
2. Validarea intrării după JSON Schema.
3. Dispatch către handler în limitele de timp și resurse declarate.
4. Validarea și redactarea rezultatului.
5. Înregistrarea rezultatului execuției și a auditului.

Registry, Dispatcher și limita de izolare susțin moduri sincrone și asincrone; aplicațiile pot înregistra propriile instrumente.

16 · Security și contextul de identitate

Identitatea verificată este baza izolării și a autorizării.

TokenVerifier consumă validarea identității SoftKit Core pentru audiența aif.

TenantContext identifică organizația și workspace-ul pentru workload.

ResourceScope păstrează ținta administrativă exactă pe cinci niveluri.

PolicyEngine este evaluatorul unic de autorizare; refuză implicit.

SecretClient, Redactor și OutputFilter controlează secretele și conținutul.

Operațiile privilegiate folosesc autoritate explicită, fără tenant sintetic.

17 · Governance și administrarea politicilor

Governance publică politica; Security o evaluează în execuție.

Draft → Review → Approved → Published → Active
→ Deprecated → Retired

Versiuni, aprobări, precedence și activare explicită.

Publicarea transmite politica efectivă către Security.

Istoricul deciziilor și corelarea cu versiunea politicii susțin auditul.

Reconcilierea stării durabile și a stării servite tratează recuperarea după erori.

Politicile de rutare și rezidență păstrează aceeași separare a responsabilităților.

18 · Configuration Management și Administration API

Starea administrată rămâne deținută de AF.

Configuration Sets, versiuni, scheme și rezultate de validare.

Activare, rollback și istoric autoritativ al tranzițiilor.

Administration API: Governance, Observability și Configuration.

Transport separat la /api/v1/administration.

Admin Console execută query-uri și comenzi administrative prin adaptor.

Comenzile Admin Console folosesc Jobs durabile; AF autorizează din nou ținta.

Configurația versionată este distinctă de configurația bootstrap a Kernel-ului.

19 · Observability și contabilizarea consumului

Telemetria descrie execuția și permite atribuirea ei.

Tracer, Meter, Logger și CostAccountant au contracte distincte.

TraceID corelează Router, Connector, agent steps și tool calls.

Dimensiuni de bază: tenant, model, provider, capability.

Latență, tokeni, cost, erori, cache hit rate și alerte de buget.

Conținutul prompturilor și răspunsurilor este exclus implicit din loguri.

Captura verbose este explicită și auditată; observabilitatea rămâne pasivă.

20 · Rezultate structurate și feedback

Controlul structurii și feedback-ul au contracte independente.

ExtractionService validează după subsetul JSON Schema declarat.

Schemele neacceptate sunt respinse înainte de apelarea modelului.

Buget de reparare: implicit 1, plafon 3; erori structurale fără valori sensibile.

Feedback: rating, correction și flag, corelate prin TraceID.

Textul de feedback este redactat înainte de persistare.

Metadata declarată de client rămâne caller-asserted; feedback-ul nu devine automat politică de rutare.

21 · Batch Jobs și Response Cache

Două mecanisme distincte pentru volum și reutilizare.

BatchService: Submit, Get, ListItems și Cancel, cu stare durabilă pe job și item.

Router folosește batch-ul providerului sau dispatch individual cu concurență limitată.

O trimitere cu rezultat incert după restart este semnalată, fără retrimiteri automate.

Fallback-ul individual are semantică at-least-once pentru itemii încă pending.

Cache-ul este separat pe tenant și cheie canonică; Router deține logica de hit.

Un hit reexecută filtrarea și contorizează apelul, cu zero consum de inferență nouă.

22 · Bugete, showback și limite distribuite

Optimizarea costului este distinctă de autorizare și de telemetrie.

Budget downgrade: Governance publică, Security potrivește directiva, Router aplică substituția.

Selecția respectă praguri, aliasul explicit și lista de aliasuri permise.

Router deține registrul de consum folosit de directivă; Observability rămâne pasivă.

Showback agregă Usage.CostUSD raportat de sursă, cu workspace și aplicație.

Limiter-ul distribuit coordonează cote, rezervări și sloturi cu lease.

Admiterea la provider este FIFO după ticket; nu presupune echitate ponderată între tenants.

23 · Rezidență regională și rotația secretelor

Politica de date și lifecycle-ul credențialelor traversează modulele prin contracte.

Data Residency filtrează eligibilitatea modelelor și a stocărilor după regiune.

Modelele sau stocările fără eticheta necesară sunt neeligibile pentru tenants fixați regional.

Memory, Knowledge și auditul folosesc resursele regionale corespunzătoare.

Secret Rotation urmărește schimbarea versiunii în backend-ul de secrete.

CredentialRefreshService aparține Provider Connectors; observatorul nu apelează Connector direct.

Un refresh eșuat păstrează clientul anterior și publică starea degradată.

24 · Injection Guard

Conținutul recuperat este tratat ca intrare care nu este de încredere.

Security inspectează secțiuni din knowledge, memorie și rezultate de instrumente.

Detector determinist bazat pe pattern-uri și normalizare versionată.

Verdict: allow, flag, quarantine sau block.

Flag păstrează secțiunea; quarantine o exclude; block oprește asamblarea.

Eroarea detectorului blochează asamblarea când inspecția este activă.

Logurile și motivele sunt structurale; scorul este derivat din ponderi, nu o garanție de detecție universală.

25 · Policy Simulation și Audit Export

Politicile pot fi analizate separat de activare, iar evidența poate fi verificată.

Simularea proiectează o politică eligibilă peste evenimente de refuz din audit.

Datele insuficiente produc Indeterminate; execuția simulării este autorizată.

Replay-ul utilizează evaluatorul Security fără efectele unei activări reale.

Audit Export citește fluxul primar printr-un cursor durabil.

Lanțuri per tenant, hash-uri și verificare față de evenimentele primare.

Export HTTP JSON semnat, retry, idempotency și raportarea rupturilor de integritate.

26 · Execuție adaptivă și modele personalizate

Portofoliul include mecanisme de selecție și experimentare pe mai multe niveluri.

Rutare după cost, latență și calitate; optimizare a politicilor din observații.

Batching între tenants și execuție speculativă la mai mulți provideri.

Descoperire automată de modele și capabilități publicate de provideri.

Aliasuri dedicate, auto-tuning și distribuție A/B sau canary.

Cache nativ la provider și failover între endpoint-uri sau regiuni.

Endpoint-uri fine-tuned, modele personalizate și pipeline-uri autonome de fine-tuning.

Aceste mecanisme sunt prezentate funcțional; formele API rămân cele ale contractelor normative aplicabile.

27 · Agenți și instrumente compozabile

Autonomia este delimitată prin definiții, permisiuni și controlul execuției.

Orchestrare multi-agent pe plan comun și replanificare a obiectivelor.

Execuții persistente, cu reluare inclusiv în interiorul pașilor.

Fan-out paralel de instrumente într-o fază de acțiune.

Definiții dinamice de instrumente și pipeline-uri declarative.

Execuție de cod într-un sandbox și selecție de instrumente după cost.

Marketplace-uri pentru definiții de agenți și instrumente.

28 · Cunoaștere și memorie federată

Partajarea este o funcție explicită, cu delimitarea participanților și a accesului.

Knowledge federat între tenants și produse;
catalog de baze reutilizabile.

Ingestie de imagini, audio și video.

Ranking hibrid vectorial + lexical și adaptare a
relevanței.

Chunking semantic sau bazat pe structură;
captură din rezultate de agenți.

Memorie multimodală și federată, cu ranking
învățat.

Tipuri și scope-uri de memorie personalizabile;
replay și export pentru utilizatori.

29 · Context și prompturi adaptabile

Relevanța și instrucțiunile pot fi administrate ca mecanisme reutilizabile.

Ranking învățat sau bazat pe recență și context partajat între sesiuni.

Secțiuni multimodale și ordine configurabilă de către consumator.

Preasamblare speculativă și cache de context între cereri.

Authoring de prompturi fără cod și variante multilingve.

Evaluare LLM-as-judge și optimizare automată din feedback.

Marketplace de template-uri partajate între tenants.

30 · Protecție și observabilitate aprofundată

Analiza conținutului, a comportamentului și a calității completează controlul de acces.

Clasificare la nivel de câmp și scanare DLP a bazelor de cunoștințe.

Chei gestionate de client, moderare și politici de securitate federate.

Detectarea anomaliilor în comportamentul agenților.

Vizualizare opt-in pentru prompturi, răspunsuri și execuții.

Scoruri de calitate și regresie, benchmark-uri agregate și anomalii de cost sau latență.

Retenție multianuală și stocare rece pentru telemetrie.

31 · Distribuție, extensii și SDK-uri

Forma serviciului permite integrarea și operarea în topologii diferite.

Module încărcate dinamic, module remote și hot reload.

Distribuție prin marketplace și extensii cu lifecycle controlat.

SDK Go, SDK-uri suplimentare, API extern și catalog self-service de capabilități.

Deployment multi-region active-active și instanțe dedicate per tenant.

Migrare de schemă online, blue/green și topologie configurabilă self-service.

Autoscaling după încărcare și politici bazate pe cost sau calitate.

Granturile speciale de acces la subsisteme au limite individuale de autoritate; catalogul nu le transformă în acces implicit.

32 · Operare și verificare

Contractele operaționale și testele susțin schimbările controlate.

Serviciu independent, configurație externalizată și persistență separată.

Health și readiness pentru AF; interfață operațională standard pentru runtimes de inferență.

Kubernetes: resurse declarative, scalare, rollout și izolare de rețea.

Teste unitare, de contract, de repository, de integrare și end-to-end.

Scenarii de eroare: timeout, retry, refuz, pierdere de lease, restart și anulare.

Compatibilitate versionată și schimbări de alias cu observare și rollback controlat.

33 · Harta celor 13 extensii

Responsabilitățile rămân în modulele care dețin decizia sau datele.

FAMILIE	CAPABILITĂȚI
Calitatea rezultatului	Structured Output · Feedback Capture
Volum și consum	Batch Jobs · Response Cache · Budget Downgrade · Cost Showback
Execuție și infrastructură	Self-hosted Connector · Distributed Rate Limiter
Protecție	Data Residency · Secret Rotation · Injection Guard
Guvernanță și evidență	Policy Simulation · Audit Export

Validatorul de schemă este bibliotecă; self-hosted extinde Providers; Injection Guard extinde Security; directivele de buget traversează Governance → Security → Router.

34 · Principiul de integrare

Aplicațiile cer capabilități; platforma păstrează ownership-ul execuției.

AI Services oferă integrarea pentru aplicații.

AI Router concentrează invocarea modelelor.

Context, Memory, Knowledge, Prompt, Agent și Tool au contracte proprii.

Governance definește; Security evaluează; Observability înregistrează.

Administration API administrează; AI Infrastructure execută inferența.

Prezentare de portofoliu derivată din repository. Contractele și ADR-urile canonice definesc semantica normativă.

35 · Integrare avansată pentru batch și extragere

Catalogul funcțional acoperă forme suplimentare de procesare și integrare.

Pipeline-uri batch de rezumare și extragere;
conector batch compatibil OpenAI.

Cross-tenant packing, webhooks și clase de
prioritate pentru loturi.

Reconcilierea automată a trimiterilor cu
rezultat incert.

Provider-native tool/JSON mode pentru
structurarea rezultatului.

Streaming de rezultate structurate parțiale și
inferență de schemă din tipuri Go.

Knowledge SDK pentru traversarea relațiilor,
ontologiilor și metadata de încredere.

Acest inventar funcțional nu introduce semnături sau garanții suplimentare peste contractele sursă.