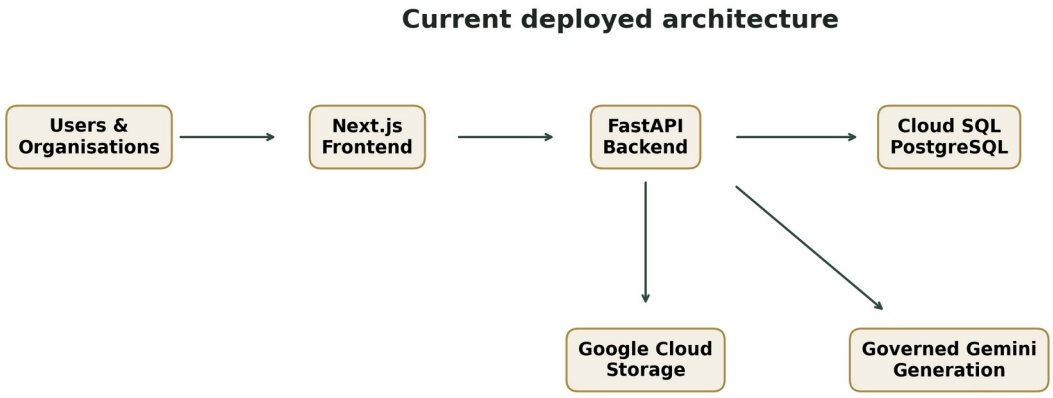
COLLECTIVE INTELLIGENCE OS

CIOS Product, Architecture and Governance

How the live system turns project formation into governed coordination infrastructure



Cloud Run hosts the frontend and backend services; migrations evolve the shared PostgreSQL model.

Current implementation state: 18 July 2026
Prepared by Donatas Starkus / DS Blockchain Lab
External briefing edition • Version 1.0

1. Executive summary

CIOS is a live production system built as a modular coordination platform rather than a generic AI wrapper. Its user-facing workflows help people and organisations form projects, identify stakeholders, analyse funding paths, prepare funding applications, preserve artifacts and reusable knowledge, and accept work as durable commitments.

Beneath those workflows is a governed project memory. Identity, authority, provenance, revisions, evidence references, artifact origins, funding requirements and commitment origins are preserved so that later users can understand not only what exists, but where it came from and who accepted responsibility for it.

> **Architectural thesis**
>
> The product should remain simple at the surface while preserving richer structure underneath. Users solve practical problems; coordination infrastructure grows as a consequence of that work.

The current platform includes:

- a Next.js App Router frontend and FastAPI backend;
- PostgreSQL on Cloud SQL with SQLAlchemy and Alembic migrations;
- Google Cloud Storage for project artifacts and media;
- Google Cloud Run production services;
- users, organisations, memberships and project-scoped owner/editor/viewer authority;
- durable artifacts, reusable knowledge and review semantics;
- funding application packages, requirements, answers, evidence mapping and export;
- a governed Gemini generation path with estimates, task allowlisting, immutable runs and human acceptance;
- persisted commitments with immutable revisions and Action Plan origin.

2. Architecture and governance principles

Modular monolith first

CIOS uses a single coordinated application architecture because product semantics are still evolving rapidly. This reduces operational fragmentation while keeping internal modules and contracts explicit.

PostgreSQL before a dedicated graph database

The current graph-memory foundation is represented through durable relational models and explicit relationships. A dedicated graph database should be justified by real query demand, not by architecture fashion.

Project-scoped authority

Authorisation is resolved in the context of each project. Membership and role determine what a user may read, change or manage.

Human acceptance before authority

AI-generated material may be previewed, edited and regenerated, but it does not silently become authoritative project state.

Immutable origin; editable working state

Generated outputs and revisions preserve lineage while users continue to edit the current project record.

Artifacts and knowledge are distinct

A project artifact may remain part of one project. Reusable knowledge requires explicit promotion, review and governance.

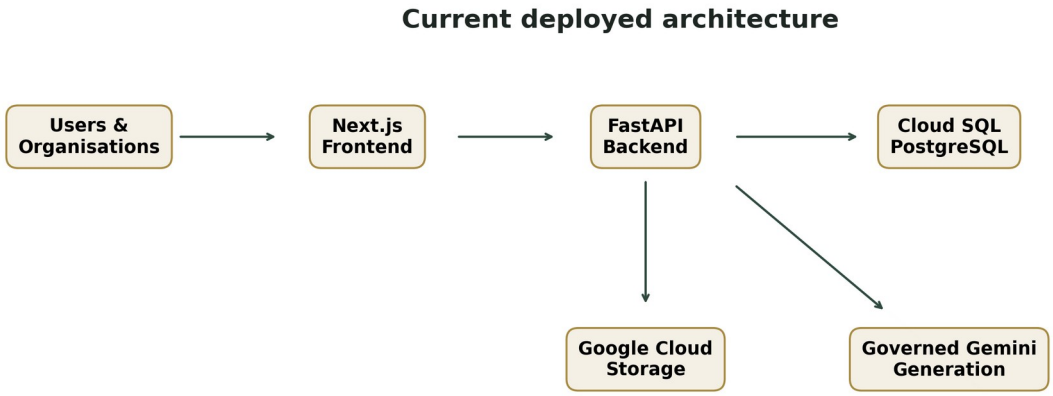
Commitments are not generic tasks

A commitment preserves ownership, origin, due date, status and revision history. It is an accountable acceptance of work.

Selective memory

CIOS should preserve meaningful context and decision lineage without turning every interaction into prominent permanent history.

3. Current deployed architecture



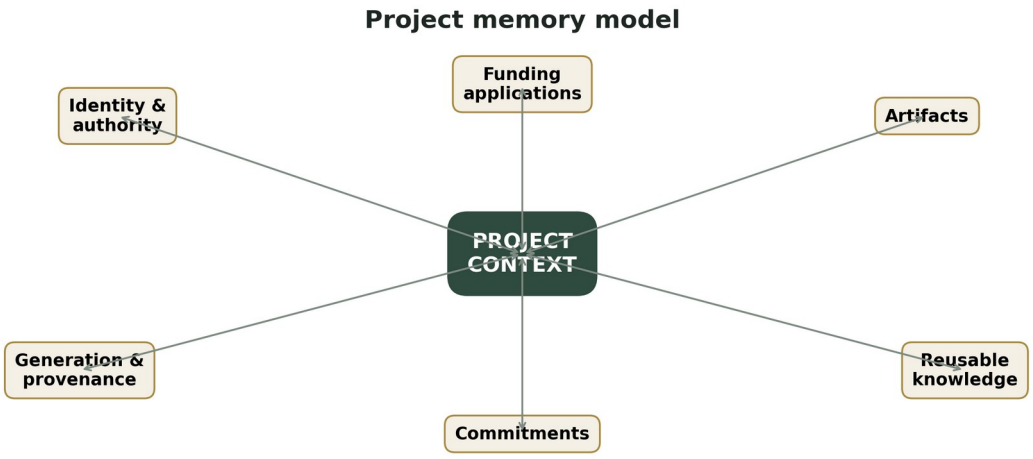
Cloud Run hosts the frontend and backend services; migrations evolve the shared PostgreSQL model.

Layer	Current implementation	Primary responsibility	Important boundary
Frontend	Next.js App Router with TypeScript	Guided project workflows, workspaces, review surfaces and administrative interfaces.	The frontend presents state and requests changes; backend authorization remains authoritative.
Backend	FastAPI modular application	Business rules, validation, project authorization, lifecycle, generation orchestration and persistence.	No frontend-only permission should be treated as security enforcement.
Database	PostgreSQL on Cloud SQL; SQLAlchemy + Alembic	Transactional project records, identity, funding, artifacts, knowledge, generation lineage and commitments.	Schema changes are versioned through migrations; production data changes require explicit operations.
Object storage	Google Cloud Storage	Generated artifacts, media derivatives and persisted project files.	Database records preserve project semantics; storage objects are not the sole source of truth.
AI provider	Gemini for approved real-provider tasks	Bounded drafting and structured generation through shared guardrails.	Model access is task-allowlisted and human acceptance remains required.
Hosting	Google Cloud Run	Containerised frontend and backend services with revision-based deployment.	Traffic can be moved between revisions; deployment is separated from database migration.

4. Product workflow architecture

CIOS is organised around a formation-to-coordination flow. The stages are not independent document generators; they contribute to a shared project context and increasingly durable records.

Workflow	Primary output	Durable project meaning	Current maturity
Intake and clarifying questions	Structured understanding of an initial problem or intention	Preserves raw context and improves the basis for later generation.	Live
Project Brief	Editable project definition	Becomes the central shared description of the project.	Live
Stakeholders	Generated and edited stakeholder roles	Provides the basis for later real-user, organisation and representation resolution.	Live generation; operational resolution still developing
Funding discovery and analysis	Funding paths and fit analysis	Connects project context to external opportunities and readiness considerations.	Live
Funding applications	Package, requirements, answers, evidence and lifecycle	Creates a durable application workspace linked to one project and opportunity.	Strong live workflow
Knowledge and artifacts	Project outputs and promoted reusable items	Separates project history from governed reuse.	Live foundation
Action Plan	Generated next-step structure	Remains the origin surface for later commitments.	Live
Commitments	Accepted accountable work records	Moves the product from planning into durable coordination and execution.	First live slice



5. Identity, organisations and project authority

Phase A established a production-proven v0 identity and authority foundation. The current system supports explicit users, organisations, organisation memberships, project memberships and optional represented organisations.

Concept	Current rule	Why it matters	Open maturity work
User	A durable application identity record exists for each recognised user.	Actions, memberships and stewardship can be attributed.	Replace compatibility identity with real login identities.
Organisation	A durable organisation record may have memberships and lifecycle state.	Projects and representation are not reduced to individual accounts.	Stronger organisation-wide authority and invitations remain future work.
Project membership	A user may be owner, editor or viewer on a project.	Project access is explicit rather than inferred from global identity.	External collaborators and invitation flows are still open.
Representation	A project member may optionally represent an organisation.	Separates personal participation from institutional representation.	Needs clearer operational workflows and authority semantics.
Admin allowlisting	Administrative access is explicitly allowlisted.	Prevents ordinary project roles from silently becoming system administration.	Broader admin policy, auditing and delegation can mature later.
Project lifecycle	Projects can be active, archived, scheduled for deletion, restored or permanently deleted under guardrails.	Lifecycle is explicit and visible rather than destructive by default.	Longer-term retention policy and automated cleanup remain open.

Authority rule

Project roles govern project capability. Organisation membership and representation add context, but they do not silently override project-scoped authorization.

6. Artifacts, knowledge and reuse

CIOS treats project outputs as durable records with origin and stewardship. Reusable knowledge is a separate governed state rather than a synonym for every saved file.

Dimension	Project artifact	Reusable knowledge
Purpose	Preserve an output or asset belonging to a project.	Support future projects with reviewed, reusable understanding.
Origin	Linked to the project and optionally to an originating artifact.	May preserve project and artifact origin and promotion history.

Dimension	Project artifact	Reusable knowledge
Governance	Visibility, steward, licence, version and evidence metadata.	Adds reuse status, review status, promotion notes and reusable semantics.
Lifecycle	May remain private or project-specific.	Should be promoted intentionally and may later be deprecated or restricted.
User value	Keeps project outputs durable and traceable.	Allows later users to begin with relevant methods, examples or lessons.

Current Phase B capabilities include:

- origin project and artifact references;
- steward user and organisation metadata;
- visibility, licence and version fields;
- origin evidence and promotion notes;
- review workflow integration;
- promotion from artifact into knowledge;
- media derivatives and fallback behaviour;
- Graph Memory foundations.

7. Funding application architecture

Funding applications are modelled as durable project workspaces, not as one-off generated documents. One application package connects a project with a funding opportunity and contains structured requirements, answers, evidence and lifecycle state.

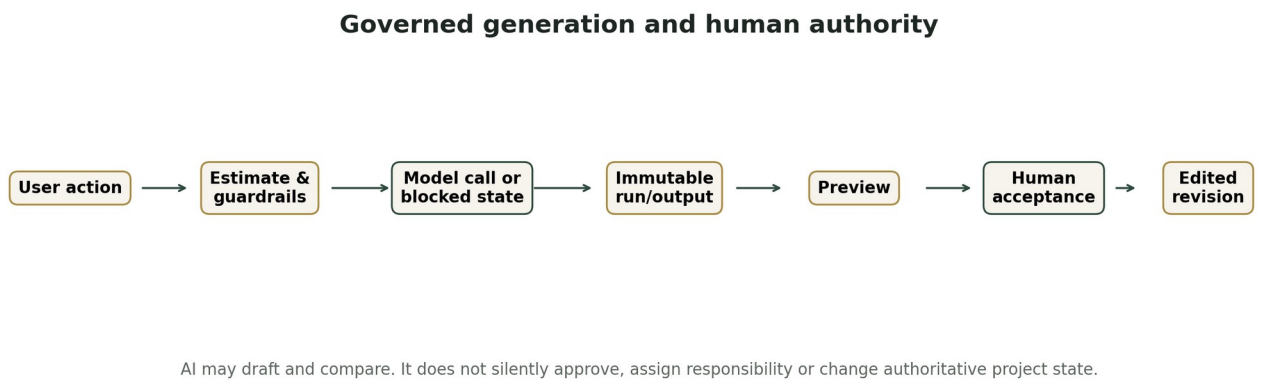
Object	Purpose	Key semantics
Funding application package	Represents the project-opportunity application workspace.	Project, opportunity, title, lifecycle status, package state, deadline and source snapshots.
Requirement	Represents one application question, section or rule.	Requirement key, section, original wording, guidance, required flag, type, review state and evidence references.
Answer	Stores the user's current working response.	Answer text, progress state and reviewer feedback.
Evidence mapping	Links application requirements to existing project material.	May reference brief content, stakeholders, funding analysis, knowledge, action-plan items, artifacts and project identity.
AI draft run	Produces a requirement-level preview.	Uses explicit project, application and requirement context; does not overwrite the editor automatically.

Object	Purpose	Key semantics
Export	Creates a portable application package representation.	Current export is markdown-oriented and preserves structured content.

Current boundary

The application workspace is live. Official form and annex ingestion, exact source extraction, compliance rules, contradiction detection and source-version rebasing remain the major unfinished document-intelligence track.

8. Shared generation and AI governance



The shared generation layer separates model availability from product authority. A generation action is estimated, checked against guardrails, recorded as an immutable run and shown as a preview. The user decides whether to use the draft and whether to save the edited result.

Governance mechanism	Current behaviour	Purpose
Provider/model registry	Approved providers and models are represented centrally rather than hard-coded per screen.	Creates consistent estimates, availability and metadata.
Task allowlisting	Only explicitly approved generation tasks may call a real provider.	Prevents accidental expansion of paid or high-risk generation.
Estimates	Supported workflows expose model, token and cost estimates before generation.	Makes model usage visible and governable.
Blocking/unavailable states	The system can refuse a real call when configuration, quota or policy does not allow it.	Avoids pretending a provider call occurred.
Immutable run/output	Reference workflows preserve the generation event and output.	Creates a trustworthy historical origin.
Preview before use	Generated text is shown separately from the editable authoritative answer.	Prevents silent replacement of human work.

Governance mechanism	Current behaviour	Purpose
Explicit acceptance	The user chooses whether to use a draft.	Keeps human judgement at the authority boundary.
Generated-then-edited lineage	Later revisions can remain connected to generated origin.	Distinguishes machine assistance from final human work.

Current real-provider reference:

- Gemini is the configured real provider for approved tasks;
- the production model family is governed through the shared registry;
- funding_application_draft is an allowed task;
- mock and fallback modes remain available for controlled development and unavailable-provider states;
- daily usage events and per-server-run limits constrain real calls.

9. Provenance and revision lineage

Provenance in CIOS is not a claim that every click is permanently recorded. It is a structured record of meaningful origin and change for selected authoritative workflows.

Record type	What is preserved	Why it matters
Generation run	Task, provider/model metadata, prompt version, context, result and status.	Explains how an AI-assisted output originated.
Generation output	The produced content before later human changes.	Allows generated content to remain distinct from the final edited state.
Revision	A later version or edit associated with an authoritative object.	Preserves change without requiring the current UI to display full history at all times.
Origin snapshot	The Action Plan item, project context or source object from which a durable record was accepted.	Keeps commitments and promoted knowledge connected to their source.
Evidence reference	A typed link to supporting project material or source content.	Lets users inspect why a claim or requirement answer is supported.
Review status	The current governance state of a reusable or reviewable record.	Separates existence from approval or reuse readiness.

Selective provenance

Full lineage should be accessible when needed, but ordinary workflows should remain focused on the current decision or task.

10. Commitment architecture

Commitments are the first durable execution object in CIOS. They are created when a user explicitly accepts work from the Action Plan rather than when the system merely generates an action item.

Capability	Current implementation	Next development
Persistence	Commitments are stored as durable project records.	Consolidated execution surface and richer filtering.
Origin	A commitment preserves the Action Plan item and origin snapshot.	Evidence and dependency links.
Ownership	A commitment may be linked to a project member or an unresolved role.	Stakeholder resolution, invitations and reassignment.
Due date	A due date can be accepted with the commitment.	Editing and revision visibility.
Lifecycle	Initial lifecycle behaviour exists.	Accepted, in progress, blocked, completed and cancelled management.
Revisions	Commitment changes create immutable revisions.	Compact detail/history view.
Duplicate-origin protection	The same originating Action Plan item is protected from accidental duplicate acceptance.	More explicit accepted-state UX.

Why commitments are different

A task says what might need doing. A commitment records that someone has accepted responsibility for work with an origin, due date, state and history.

11. Review, administration and lifecycle

CIOS includes an administrative layer for identity, organisations, project access, review and lifecycle. This is not intended to replace project ownership; it provides controlled system-level governance.

Administrative area	Current capabilities	Governance value
Identity management	Create and inspect users and organisations; manage organisation memberships.	Makes authority and organisational context explicit.
Project access	Inspect and manage project memberships and represented organisations.	Supports recovery, curation and exceptional administration.
Review queue	Review pending funding opportunities, knowledge items and project artifacts.	Separates draft existence from approved or reusable state.
Project lifecycle	Archive, restore, schedule deletion, cancel deletion and permanently delete under guardrails.	Avoids unsafe hard-deletion as the default.

Administrative area	Current capabilities	Governance value
User lifecycle	Deactivate users.	Allows access state to change without erasing historical attribution.
Organisation lifecycle	Archive organisations.	Preserves records while removing them from active use.

12. Security and data boundaries

This document does not claim completed enterprise security certification or full authentication maturity. The current boundaries are application-level controls that have been production-tested across key read and mutation paths.

Boundary	Current state	Important limitation
Authentication	A compatibility identity supports the current production environment.	Full real login identities, external invitations and recovery flows are not yet complete.
Authorization	Project-scoped membership and role checks protect project reads and mutations.	Organisation-wide policy is still limited.
Administrative access	Explicit allowlisting controls system administration.	Delegated and auditable admin policy can mature further.
Artifact visibility	Artifacts and knowledge carry visibility and stewardship semantics.	End-user sharing and reuse permissions remain incomplete.
Provider access	Real AI calls require enabled configuration, allowlisted tasks and available limits.	This is not a substitute for broader data-processing and privacy policy work.
Deletion	Project lifecycle supports archive, scheduled deletion and guarded permanent deletion.	Long-term retention and automated policy enforcement remain future work.

13. Testing, migrations and release discipline

CIOS development uses migration-backed schema evolution, automated backend tests, frontend type and build validation, and Cloud Run revision deployment.

Practice	Current approach	Purpose
Database migrations	Alembic migration history evolves identity, lifecycle, artifacts, funding applications and commitments.	Keeps schema change explicit and repeatable.

Practice	Current approach	Purpose
Backend tests	Focused domain tests and full-suite runs validate authorization, funding, generation, knowledge and lifecycle behaviour.	Reduces regression risk across interconnected semantics.
Frontend validation	Type checking, production builds and targeted assertions are used before deployment.	Catches routing, API and state integration failures.
Revision deployment	Frontend and backend are deployed as named Cloud Run revisions.	Supports controlled traffic movement and rollback.
Production verification	Health checks and selected API/product paths are tested after traffic changes.	Confirms that deployment, migration and routing agree.
Checkpointing	Commits and tags preserve stable product milestones.	Provides a clear handoff and rollback reference.

14. Key architecture decisions

Decision	Current choice	Rationale	Trade-off / revisit trigger
Application architecture	Modular monolith	Product semantics are evolving and benefit from shared transactions and simpler operations.	Revisit when one subsystem has distinct scaling, security or deployment needs.
Graph storage	PostgreSQL relational models first	Current workload needs strong transactional consistency more than a separate graph stack.	Revisit when real multi-hop queries or graph analytics become central and relational approaches become limiting.
AI authority	Explicit human acceptance	Keeps responsibility and authoritative state with people.	More interaction than full autonomy; this is an intentional product principle.
Generation history	Immutable run/output plus editable revisions	Preserves origin without preventing human refinement.	Requires careful UI so history remains available but not overwhelming.
Artifacts vs knowledge	Explicit promotion	Avoids turning every output into reusable truth.	Requires review and curation effort.
Execution object	Commitment separate from task	Preserves origin, ownership and accountability.	Must remain lightweight enough for real adoption.
Action Plan role	Origin surface, not execution database	Keeps generated planning distinct from accepted responsibility.	Requires a later consolidated commitment surface.

Decision	Current choice	Rationale	Trade-off / revisit trigger
Document intelligence	Separate bounded strategic track	Reduces risk of mixing execution, graph and document reasoning into one oversized phase.	Requires disciplined integration contracts with Funding Applications.
Repository documentation	Product truth close to code	Reduces drift between implementation and documentation.	Commercial and confidential material still needs a separate internal workspace.

15. Current limitations and next technical priorities

Area	Current limitation	Next bounded priority
Real authentication	Compatibility identity remains.	Introduce real login identity and invitation flows while preserving project-scoped authorization.
Commitment execution	The durable record exists, but management is still minimal.	Add detail view, lifecycle management, revisions, evidence and dependencies.
Document intelligence	Official forms and annexes are not yet ingested as structured sources.	Implement one bounded ingestion pipeline with source identity, extraction, exact citations and human verification.
Reuse governance	Promotion exists, but reuse permissions and accounting remain immature.	Add methodology, application flow, relevance signals and governance.
Provenance breadth	Strong reference workflows exist, but older workflows are uneven.	Adopt shared generation and lineage patterns more broadly.
Accessibility	The product still needs systematic keyboard, narrow-screen and accessibility QA.	Run targeted audits and remediate shared components.
Policy maturity	Retention, audit export and privacy policy versioning are incomplete.	Define policies before enterprise or sensitive-data claims.
External API and agents	No public agent-facing surface exists.	Delay until permissions, graph quality and bounded capabilities are mature.

16. Selected technical glossary

Term	Meaning in this architecture
Authoritative state	The current human-accepted project record used by the application.

Term	Meaning in this architecture
Compatibility identity	The current transitional user identity mechanism used before full real-login maturity.
Generation run	An immutable record of one model-generation event, including task and model metadata.
Generated-then-edited lineage	The relationship between original model output and later human-edited authoritative content.
Origin snapshot	A preserved copy of the source context from which a durable record was accepted.
Project-scoped authorization	Capability derived from a user's membership and role on a specific project.
Promotion	The explicit transition of an artifact into governed reusable knowledge.
Requirement-level drafting	AI assistance focused on one funding-application requirement at a time.
Selective memory	Preserving meaningful origin and change while avoiding indiscriminate permanent noise.
Unresolved role	A commitment owner placeholder representing a needed role before a real member is assigned.

Where to continue

This brief explains the live product and technical foundation. The complementary documents are CIOS Overview, CIOS Current State, Roadmap and Strategic Direction, and the CIOS Handbook - Edition One.

Technical and Operational Appendix

Verified request chains, performance evidence, architecture risks and near-term optimization plan

This appendix adds implementation-level depth from read-only technical audits and latency investigations performed against the active CIOS checkout and production environment during July 2026. Audit snapshots are dated evidence, not permanent promises: branch heads, revisions and measurements can change as the product evolves.

Evidence discipline

The sections below distinguish observed production behavior, source-confirmed request chains, proposed optimizations, and unresolved risks. No benchmark or architecture observation should be read as a timeless guarantee.

17. Verified implementation baseline

The deeper audits confirmed a clean active checkout, a single migration head, broad automated test coverage, and a largely modular domain structure with several concentrated orchestration files.

Evidence area	Observed state	Interpretation
Repository	Active checkout: /Users/donatasstarkus/Developer/cios-mvp; branch generation-metadata-copy-polish-v0; clean worktree in the reviewed snapshots.	A single active checkout reduced the risk of documenting an obsolete source tree.
Migration chain	Alembic head 0026_generation_provenance; one-head contract confirmed in source/tests.	Schema evolution is disciplined, but newer domain work must continue to preserve single-head migration integrity.
Runtime	Python 3.12.x backend; Node 22.x and Next 15.5.x frontend in the audited snapshot.	The live stack is current enough for the product stage; runtime versions should be recorded per release rather than described generically.
Validation	Full backend suite observed at 575 passed in one audit; focused suite 242 passed; frontend typecheck and production build passed.	Automated coverage is substantial, but browser-level and responsive behavior remain weaker than backend/API coverage.
Code concentration	Large route/component surfaces include projects.py, funding.py, FundingApplicationWorkspace.tsx and ArtifactLibrary.tsx.	The architecture is modular enough to continue, but orchestration seams should be extracted before document processing and execution semantics expand.

Evidence area	Observed state	Interpretation
Provenance adoption	Funding Application and Project Identity are the strongest GenerationRun/GeneratedOutput reference workflows.	New workflows should use the shared model; older records must remain truthfully labelled as legacy or partially reconstructable.

18. Domain and coupling map

Domain	Primary durable objects	Main orchestration boundary	Current risk
Projects and lifecycle	Project, ProjectMembership, lifecycle timestamps and deletion state	Project routes and workspace aggregation	Workspace is a broad aggregate; several reads historically bypassed the central access helper.
Identity and organisations	User, Organisation, organisation and project memberships, representation	Admin identity routes and project access service	Compatibility identity and admin boundary are not yet a final multi-tenant authentication model.
Artifacts and knowledge	ProjectArtifact, KnowledgeItem, project-knowledge links and review state	Artifact storage/service plus project routes	Storage and metadata are durable; reuse workflow, permissions and retrieval quality remain immature.
Funding discovery	Funding sources, opportunities, matches, analysis and review	Funding routes and matcher/import services	Large route module and repeated project-context assembly.
Funding applications	Package, requirements, answers, revisions, source pins and citations	FundingApplicationWorkspace plus funding API	Strong domain, but requirement structure remains partly JSON-shaped and full workspace context is over-fetched.
Generation and provenance	GenerationRun, GeneratedOutput, domain revisions and usage events	Shared generation services and domain integrations	Uneven adoption across legacy workflows; model-registry calls are duplicated client-side.
Action plans and commitments	ActionPlanItem plus persisted commitments and commitment revisions	Workspace Action Plan and coordination routes	Commitment foundation is live; lifecycle, evidence, dependencies and detail views are still partial.
Graph Memory	Graph entities, edges, linked project outputs and knowledge semantics	Workspace graph assembly and sync services	Full graph attributes and unbounded aggregate payloads can become expensive; mature relevance remains future work.

Domain	Primary durable objects	Main orchestration boundary	Current risk
Media and storage	GCS/local storage abstraction, originals and derivatives	Artifact storage service and file/thumbnail routes	Missing-object fallback works, but diagnostics and historical backfill visibility should be generalized.

Current architectural posture

CIOS does not need microservices or a dedicated graph database merely because the domain is broad. The near-term need is clearer domain services, compact contracts and selective loading around the existing modular monolith.

19. Authorization consistency and isolation risks

The audits found that project mutation and generation paths had moved toward a central capability helper, while some older read, export, artifact and funding-analysis routes used direct project lookup or narrower role helpers.

Area	Observed enforcement pattern	Risk	Required direction
Project mutations and generation	Central require_project_access capability checks	Good v0 reference pattern	Use as the default for every new project-scoped route.
Workspace, brief and export reads	Historically used direct project lookup in reviewed snapshots	A caller with an identifier may reach data without consistent membership/lifecycle evaluation	Normalize read and export routes through the same project-access policy.
Artifact list, file and thumbnails	Project existence checks and storage lookup	File paths and identifiers are sensitive object-level boundaries	Apply membership-aware access before serving metadata or bytes.
Funding applications	Dedicated application/project role helper	Can diverge from central archived/scheduled-deletion semantics	Consolidate on shared capability resolution.
Funding analysis and generation provenance	Mixed direct lookup or role helper patterns	Inconsistent access behavior across related project outputs	Centralize and regression-test route coverage.
Administration	Explicit allowlisting plus compatibility-mode assumptions	Suitable for a trusted MVP environment, not a final tenant boundary	Define real session identity and durable admin authorization before broad external collaboration.

Phase E implication

Durable commitments introduce assignees, roles, evidence and activity history. Object-level authorization must become boringly consistent before those records are exposed to real external collaborators.

20. Current client request chains

Warm-navigation latency is shaped less by the same-origin proxy than by the number, size and duplication of client requests triggered by a route.

Route or surface	Observed initial chain	Classification	Optimization opportunity
/app	GET /api/projects + GET /api/dashboard/summary	Both useful, but backend project-list batching is similar	Derive visible home summary from ProjectListItem[] and reserve dashboard summary for other/admin use.
Workspace - default Brief	GET workspace + GET knowledge-proposals, then GET access + GET members	Workspace required; proposals/access/members can be deferred	Target one blocking workspace request and at most one deferred request.
/projects/{id}/brief	GET full workspace	One request, but payload is oversized for brief-only rendering	Add a compact brief/bootstrap endpoint only after client dedupe is measured.
Workspace - Funding tab	Parent workspace + funding applications + saved funding analysis + model registry	Applications are tab-specific; saved analysis duplicates workspace content	Seed FundingPathList from parent data and cache the model registry.
Workspace - Artifacts tab	Parent workspace + artifacts + identity + model registries	Artifacts and identity overlap with the parent workspace	Pass initial workspace data; refetch only after mutation or manual refresh.
Funding application route	Application detail + full workspace + model registry	Application is required; full workspace is used mainly for evidence options	Return compact funding-application context or combine application + context in one endpoint.
/funding?projectId=...	Opportunities + sources + ingestion runs, then project applications	First three support browse; applications are conditional	Keep conditional application load and avoid unrelated workspace fetches.
Focus / visibility refresh	Full workspace + proposals on focus and visibilitychange	Refresh can be redundant and race-prone	Use stale age, visibility, in-flight dedupe, coalescing and sequence guards.

21. Workspace payload and state ownership

The workspace endpoint is useful as a broad bootstrap, but the observed production payload for a sampled project was approximately 98 KB and contained many sections not visible on the first render.

Workspace section	Why it is useful	Why it can be expensive	Recommended ownership
Project and intake	Core identity and brief context	Always needed, but often mixed with unrelated sections	Keep in the route-level workspace owner.

Workspace section	Why it is useful	Why it can be expensive	Recommended ownership
Clarifying questions and project brief	Supports formation history and Brief tab	Detailed history may exceed first-render need	Keep current summary; load deep provenance on demand.
Stakeholders and funding analysis	Supports tabs and evidence selection	Duplicated by tab-specific requests in some flows	Pass parent data into child components as initial state.
Knowledge links and proposals	Supports Knowledge/Artifacts workflows	Not required for default Brief rendering	Defer until relevant tab or after meaningful first paint.
Artifacts and applied identity	Supports Artifacts and application evidence	Repeated through /artifacts and /identity child calls	Use workspace-derived initial data and authoritative mutation responses.
Graph entities and edges	Supports Graph Memory view	Full attributes can be large and unbounded	Load graph detail only when the graph surface is opened.
Admin reviews and full provenance	Supports curation and technical inspection	Usually not visible during ordinary project work	Move to on-demand detail/disclosure endpoints.

Near-term workspace data-owner rules:

- one route-scoped owner for the current project workspace response;
- one in-flight request at a time for the same project and refresh intent;
- sequence IDs or cancellation so older responses cannot overwrite newer route state;
- local state patching after successful mutations instead of immediate full refetch;
- stale-time refresh rather than unconditional focus/visibility refresh;
- tab children receive parent-derived initial data and fetch only missing or intentionally refreshed detail.

22. Funding Application data contract

Funding Application is the strongest durable domain and the clearest place to replace broad aggregate loading with a purpose-built context contract.

Current behavior	Cost or risk	Preferred next contract
Load application detail and full workspace in parallel	The workspace is oversized when only evidence-option summaries are needed	GET application plus a compact funding-application-context response.
Load model registry separately for each route or tab visit	Repeated identical task metadata calls	Short-lived module-level promise/value cache keyed by generation task.
PATCH an answer, then GET the full application	Extra round trip despite mutation returning authoritative requirement state	Patch the returned requirement into local application state.

Current behavior	Cost or risk	Preferred next contract
Repeat full reads after feedback/evidence updates	Unnecessary serialization and UI latency	Reuse the mutation response for local state.
Load provenance/revisions with primary content	Technical detail competes with first render	Keep provenance and revision detail on demand.
Use package JSON requirements as source truth	Flexible for v0, weak for source rebasing and indexing	Introduce relational requirement/evidence records when document intelligence begins, with compatibility reads during migration.

Target

A funding application should reach its first meaningful render with one application/context request. Model metadata can be cached or deferred, and mutations should update local state without a full refetch.

23. Production latency evidence

The July latency investigation separated natural Cloud Run cold starts from warm API behavior and proxy overhead.

Measurement	Observed result	Interpretation
Natural first request	Frontend and backend both started new instances; first /api/projects path took roughly 4.4-4.5 seconds in the observed production event.	Cold start was the primary cause of the worst delay.
Warm HTML routes	Most sampled routes returned in roughly 0.27-0.79 seconds total.	Server-rendered route response was acceptable once warm.
Warm API endpoints	Lightweight endpoints were roughly 0.25-0.31 seconds; project list/workspace commonly around 0.45-0.52 seconds total.	Large payload serialization and multiple client requests create the remaining warm-navigation feel.
Same-origin proxy	Matched proxy overhead was small and noisy, often only a few to tens of milliseconds.	Removing the proxy would not solve the dominant problem.
Workspace size	Sampled workspace response around 98 KB.	Compact route contracts and deferred detail are more valuable than proxy changes.
Artifact media	One 1.3 MB artifact response took about 1.57 seconds.	Image/file-heavy surfaces need caching, derivatives and missing-object diagnostics.
Concurrency sample	Three-request warm checks showed no queueing, 429, 5xx or new instance evidence.	Current low concurrency was not the observed bottleneck.

Measurement	Observed result	Interpretation
Performance classification		
Cold-start dominant for the worst first interaction; request-waterfall and payload-shape dominant for warm navigation.		

24. Cloud Run configuration and deployment implications

Area	Observed state in the audit snapshot	Implication
Compute	Frontend and backend reported 1 CPU / 512 MiB, concurrency 80 and 300-second timeout.	No evidence that simply raising CPU/memory is the highest-value first step.
Scaling	Production metadata showed min unset/0 and max 20; startup CPU boost enabled.	Scale-to-zero explains natural cold starts; checked-in scripts had drift from live max settings.
Staging experiment	Setting min instances to 1 on isolated staging removed ordinary request-triggered instance starts during the observation window.	A reversible min-instance experiment is the cleanest infrastructure lever for cold starts.
Production proxy	Frontend runtime CIOS_BACKEND_URL routes same-origin API traffic to the backend.	The runtime proxy is the intended production path; legacy NEXT_PUBLIC_API_BASE_URL documentation should be treated as compatibility history.
Probes	Default TCP startup probe; no richer liveness/readiness evidence in the reviewed snapshot.	Operational hardening can later add clearer health semantics, but it was not the source of warm-navigation waterfalls.
Image/build	The frontend used a larger runtime image path in one audit; standalone output was enabled in a later source snapshot.	Build/deploy documentation must be versioned because implementation changed during the same period.

Cloud Run settings should be documented as release-specific operational facts, not permanent architecture. The repository, deployed service metadata and runbook must be reconciled whenever live settings drift from checked-in scripts.

25. Refresh, cache and stale-response policy

Concern	Policy
Focus and visibility events	Refresh only when the document is visible and the last successful load is older than a defined stale threshold, such as 60-120 seconds.

Concern	Policy
Duplicate triggers	Coalesce focus and visibility events occurring within a short window.
In-flight work	Do not start a duplicate workspace request while an equivalent request is running.
Stale responses	Use request sequence IDs or AbortController so older responses cannot replace newer route state.
Known mutations	Patch local state from authoritative mutation responses; do not use a full workspace refetch as the default consistency strategy.
Manual recovery	Preserve an explicit manual refresh path for unusual failures or external changes.
Model registry	Cache promises and resolved values by task for a short TTL; expose refresh for intentional bypass.
Evidence/provenance detail	Load only when the user opens the technical detail or evidence selector that needs it.

26. Testing and observability depth

Area	Existing strength	Blind spot	Smallest useful improvement
Backend domain tests	Broad API and domain coverage across memberships, funding applications, provenance, artifacts and lifecycle.	Coverage can still miss cross-route authorization consistency.	Add route-inventory tests asserting central project access on every project-scoped endpoint.
Frontend assertions	Fast structural checks protect important copy, routing and shared-generation patterns.	Static assertions do not prove browser state, responsiveness or interaction timing.	Add a small Playwright suite for workspace, funding application and access-sensitive routes.
Build validation	Typecheck and production build are regular gates.	Build success does not validate runtime environment drift.	Record build artifact, runtime config and deployed revision in a release manifest.
Generation observability	Usage events and generation identifiers exist.	Not every workflow returns generation_run_id/generated_output_id consistently.	Return stable IDs in every new generation response and log them with request correlation.
Storage diagnostics	Fallback behavior prevents some broken media.	Missing-object behavior is not uniformly visible.	Emit structured storage-miss logs with project, artifact, object key and fallback result.

Area	Existing strength	Blind spot	Smallest useful improvement
Latency diagnostics	Read-only curl and Cloud Run log correlation identified cold starts.	No ongoing user-journey timing or browser waterfall telemetry.	Add lightweight route/request timing and a bounded client diagnostic payload.
Release checkpoints	Git tags and revision names preserve milestones.	Operational facts can remain in handoff memory rather than canonical docs.	Create a release-checkpoint record with migration, images, revisions, traffic and smoke result.

27. Prioritized technical risk register

Risk	Severity	Confidence	Blocks	Recommended direction
Inconsistent project read/file/export authorization	High	Confirmed in reviewed snapshots	Production-grade Phase E	Normalize every project-scoped route through the central access service and test route coverage.
Compatibility identity and admin boundary	High	Strongly indicated	Broad external collaboration	Implement real session identity, invitations and explicit admin authorization before claiming multi-tenant maturity.
JSON-shaped funding requirements	High	Confirmed	Document intelligence	Introduce relational requirements, evidence and source-unit references when ingestion starts.
Uneven GenerationRun/GeneratedOutput adoption	High	Confirmed	Document intelligence; some future coordination generation	Require the shared lineage contract for every new generation-enabled workflow.
No parser/extraction/review substrate	High	Confirmed	Document intelligence	Add immutable document versions, parser jobs, extracted units, requirement candidates and review states.
Workspace and funding orchestration concentration	Medium	Confirmed	Neither immediately	Extract compact services/contracts as product slices require them; avoid broad rewrite.
Cold starts with min instances 0	Medium	Confirmed for observed event	User experience	Run a reversible cost/latency experiment and document the chosen service policy.
Client request duplication and unconditional refresh	Medium	Confirmed from source	Warm navigation quality	Introduce route-scoped ownership, stale guards, dedupe and local mutation patching.

Risk	Severity	Confidence	Blocks	Recommended direction
Storage-miss visibility	Medium	Confirmed pattern	Media/document growth	Generalize missing-object telemetry and backfill reporting.
Lack of browser/responsive regression tests	Medium	Confirmed gap	Neither, but increases release risk	Add a small behavioral browser suite around critical flows.

28. Recommended optimization sequence

1	**Client request ownership** - Add a route-scoped workspace data owner with in-flight dedupe, stale-response protection and mutation-local updates.
2	**Home request reduction** - Derive the visible home summary from the projects response and stop duplicating project-list work on /app.
3	**Deferred knowledge load** - Move knowledge proposals out of the default blocking workspace path.
4	**Seed tab state** - Pass artifacts, identity and funding analysis from the parent workspace into tab components.
5	**Model registry cache** - Use a shared short-TTL cache keyed by task.
6	**Funding mutation patching** - Update local application state from PATCH responses rather than refetching the full application.
7	**Compact context contracts** - Only after measurement, add brief/bootstrap and funding-application-context endpoints.
8	**Authorization normalization** - Ensure every new compact endpoint and existing project read/file route uses the same project-access policy.
9	**Infrastructure experiment** - Evaluate min instances 1 against actual budget and observed first-request value.
10	**Behavioral verification** - Add browser-level checks and route timing before declaring the optimization complete.

29. Documentation status model

Deep architecture documentation should label both implementation status and evidence status. This allows the document to remain broad without confusing live capability, design direction and strategic possibility.

Dimension	Recommended labels
Product status	Live; partially live; foundation implemented; designed; planned; strategic hypothesis; long-term possibility; not currently planned.

Dimension	Recommended labels
Evidence status	Repository-verified; production-verified; demonstrated locally; documented design; founder hypothesis; requires market validation; requires legal or security review.
Time status	Current as of a named date/revision; historical audit snapshot; superseded; pending verification.
Risk status	Blocks Phase E; blocks document intelligence; should fix in next two slices; later improvement; research only.

Why this depth belongs here

CIOS now has enough implementation history that architecture is not just a stack diagram. Request shape, authorization consistency, data ownership, provenance adoption, deployment drift, testing and operational evidence are part of the real system design.