ChatGPT

Audit-Roadmap: Repo-Hygiene und Governance für Claire de Binare

Einleitung

Das Projekt **Claire de Binare (CDB)** besteht aus einem **Working Repo** für den ausführbaren Code und einem **Docs Hub** für Governance-, Knowledge- und Agentendokumente. Laut Verfassung und Policy-Stack ist diese Trennung fest definiert – der Arbeits-Code bleibt bereinigbar und deterministisch, während kanonische Dokumente im Docs Hub versioniert werden. Ziel dieser Audit-Roadmap ist es, den aktuellen Zustand des Working Repos zu analysieren, Abweichungen von den Governance-Richtlinien zu identifizieren und konkrete Maßnahmen für eine robuste Repo-Hygiene und Governance-Überarbeitung zu planen.

Grundlagen & Policies

Verfassung und Policy-Stack

- **CDB Constitution (v1.1)** – höchste Instanz: sie definiert das deterministische, event-getriebene Trading-System und legt die Rangordnung fest (Verfassung → Governance → Policies → Implementierung) [1]. Sie garantiert Souveränität des Users, Dezentralisierung, Transparenz und Auditierbarkeit [2]. Governance-Entscheidungen folgen dem Proposal-Review-Approval-Merge-Prozess [3].
- **CDB Governance (v1.1)** – operationalisiert die Verfassung. Das Dokument regelt Rollen, Rechte, Zonen und Change Control; es ist als Systemvertrag zu sehen und trennt Ausführung, Autonomie und Sicherheit (Zeilen 13–31 des Dokuments) und ist bindend für alle Beteiligten.
- **CDB Policy Stack Mini** – definiert den kanonischen Dokumentensatz (Constitution, Governance, Agent Policy, Infra Policy, RL-Safety Policy, Tresor Policy, PSM Policy und das Delivery Gate) [4]. Änderungen an diesen Dokumenten sind nur via Proposal → Review → explizite User-Freigabe → versioniertem Merge zulässig [5].
- **Agent Policy** – legt die Handlungszonen der KI-Agenten fest. In Zone A dürfen Agenten im Rahmen des Mandats autonom agieren, in Zone C dürfen sie nur Vorschläge unterbreiten und Zone D ist strikt verboten (z.B. Tresor-Zugriff, Änderung von Hard Limits) [6] [7]. Persistente Schreibrechte sind auf den Knowledge-Bereich beschränkt; Schreibzugriffe in den Working-Repo-Code oder in `knowledge/governance` sind untersagt [8].
- **Infra Policy** – fordert Infrastructure as Code und GitOps als Single Source of Truth [9] [10]. Services müssen kubernetes-ready sein (stateless, Konfiguration über ENV/Secrets, Health/Liveness-Probes) [11]. Secrets dürfen nicht im Repo liegen und werden ausschließlich über Secret-Manager bereitgestellt [12].
- **RL-Safety Policy** – definiert deterministische Guardrails für reinforcement learning. RL-Aktionen werden durch einen Risk-Layer maskiert; ein Kill-Switch mit abgestuften Stufen (Reduce Only, Hard Stop, Emergency) ist vorgeschrieben [13]. RL darf niemals Hard Limits ändern, Keys anfassen oder Governance verändern [14].
- **Tresor Policy** – sichert kryptografische Assets, Hard Limits und Governance-Dateien in einer Tresor-Zone außerhalb des Repos. Kein autonomer Zugang ist erlaubt; alle Änderungen sind

human-only und nachvollziehbar [15] [16]. Secrets und Keys dürfen niemals im Repository liegen [17].

- **Repo-Guidelines & Repo-Structure** – beschreiben die erlaubten Verzeichnisse und Cleanroom-Regeln für das Working Repo. Erlaubt sind `/core`, `/services`, `/infrastructure`, `/tests`, `/tools`, `/scripts`, Compose-Dateien und Makefile [18] [19]. Verboten sind `/knowledge`, `/governance`, `/agents` und Logs im Working Repo [20]. Das Working Repo soll jederzeit löschbar und neu aufbaubar sein [21]. Commit- und PR-Nachrichten müssen konventionell strukturiert sein und die Änderungen, Tests und Risiken beschreiben [22].

Diese Grundlagen bilden den Maßstab für das Audit.

Aktueller Zustand des Working Repos

Struktur & Verzeichnisse

- Die Verzeichnisstruktur entspricht weitgehend den Richtlinien: Es gibt `core/`, `services/`, `infrastructure/`, `tests/`, `tools/` und `scripts/`. Ein `knowledge`- oder `governance`-Verzeichnis existiert nicht, was der Trennung zwischen Working Repo und Docs Hub entspricht [19] [20].
- Für Dokumentation wurde eine Datei `DOCS_MOVED_TO_DOCS_HUB.md` hinterlegt, die erklärt, dass alle Dokumente in das Docs Hub migriert wurden und das Working Repo nur Code enthält [23] [24]. Dies entspricht der Clean-Repo-Philosophie.
- Es existiert keine `LICENSE`-Datei und kein `CODE_OF_CONDUCT.md`. Die repo-Guidelines verlangen zwar keine konkreten Lizenzen, aber für Open-Source-Transparenz und klare Nutzungsrechte sollte eine Lizenz definiert werden.

README & Projektstatus

- Das Root-`README.md` beschreibt das Projekt, listet die Microservices, Infrastruktur und Governance-Dokumente auf und enthält einen aktuellen Fortschritts-Bericht (Issues/Services/Tests/Monitoring) [25] [26]. Der Bericht ist sehr umfangreich, aber auf deutsche Nutzer*innen ausgerichtet; eine englische Zusammenfassung könnte die Zugänglichkeit erhöhen.
- Der README-Statusblock zeigt 65 % Projekt-Reife mit 1607 Python-Dateien, 140+ Commits und 247 Tests [27]. Diese Informationen sind hilfreich für neue Mitarbeitende.

CI/CD, GitHub Hygiene & Milestones

- Laut `MILESTONES.md` wurde die Phase M1 („GitHub & CI Baseline“) als abgeschlossen markiert. Sie zielte auf Repository-Hygiene, aktives CI/CD, strukturierte Labels und Automatisierungen [28]. In dieser Phase wurden ein Stale Bot, Auto-Labeler und ein Label-System eingeführt – das legt eine solide Basis für das Issue-Management.
- Die nächsten Meilensteine (M2 bis M9) sind geplant oder in Arbeit; u. a. „Infra & Security Hardening“, „Observability“, „Persistenz“, „Testnet“, „Security Review“ und „Release 1.0“ [29] [30]. Diese Roadmap sollte mit den hier vorgeschlagenen Audit-Maßnahmen abgeglichen werden.

Sicherheitslage & Legacy

- Das `SECURITY_BASELINE.md` dokumentiert, dass bekannte CVEs (u. a. pip CVE-2025-8869) behoben wurden und alle Python-Services auf pip 25.3 aktualisiert wurden [31]. Darüber hinaus wurden Sicherheitsmaßnahmen wie non-root-Images, Read-only-Dateisysteme und wöchentliche Scans umgesetzt [32].

- Es wird ein **Accept-Risk** für embedded gosu-Binaries (kritische CVEs in den Redis/Postgres-Images) beschrieben. Die Attack-Surface-Analyse bewertet das Risiko als niedrig und definiert eine Mitigation mit Pinning, CI-Gates und Monitoring [33].
- Das `LEGACY_FILES.md` listet zahlreiche veraltete Compose-Dateien (`docker-compose.yml`, `docker-compose.base.yml`, `docker-compose.dev.yml`) sowie unsichere Praktiken wie plaintext-Passwort-Variablen und lokale `.secrets`-Verzeichnisse. Es beschreibt Migrationspfade zu den neuen Dateien unter `infrastructure/compose` und zur Nutzung von Docker-Secrets [34] [35]. Die Entfernung der Legacy-Dateien ist für Phase 3 geplant [36].

Hinweise zu Commit-Hooks & Secret-Scanning

- Unter `tools/` gibt es Skripte wie `install-git-hooks.ps1` und `enforce-root-baseline.ps1`. Dies deutet auf eine vorbereitete Infrastruktur für Git-Hooks zur Durchsetzung von Commit-Standards und Root-Baseline-Checks. Eine Prüfung der installierten Hooks und deren Wirksamkeit sollte Teil des Audits sein.
- Es existiert eine `.gitleaksignore`-Datei, die offenkundig für das Secret-Scanning mit **gitleaks** genutzt wird. Es sollte überprüft werden, ob gitleaks in der CI-Pipeline aktiviert ist und ob das `gitleaks.toml`-Konfigurationsfile existiert.

Fehlende oder verbesserungswürdige Punkte

1. **Lizenz & Rechtliches:** Es fehlt eine Lizenzdatei, sodass unklar ist, unter welchen Bedingungen der Code genutzt werden darf. Eine OS-Lizenz wie MIT, Apache 2.0 oder GPL sollte gewählt und implementiert werden. Ebenfalls fehlt ein `CODE_OF_CONDUCT.md` und ein `CONTRIBUTING.md`, die für die Community-Governance hilfreich sind.
2. **Test-Abdeckung & Qualitätsmetriken:** Laut README existieren 247 Testfunktionen bei 1607 Python-Dateien [27]. Eine Coverage-Analyse (z.B. mittels `pytest-cov`) ist nicht dokumentiert; hierfür sollten Metriken definiert und CI-Gates gesetzt werden.
3. **Secret-Management:** Obwohl die Legacy-Files einen Umzug zu Workspace-Secrets und den Verzicht auf `.env` propagieren [37], sollte überprüft werden, ob wirklich keine `.env`-Dateien im Repo verbleiben und ob secrets nicht versehentlich in den Code gelangen. Der Infra-Policy zufolge sind Secrets im Repo strikt verboten [12].
4. **Ergänzende Governance-Checks:** Der Policy-Stack legt strenge Regeln für Änderungen fest. Eine Prüfung der vorhandenen Pull-Requests sollte sicherstellen, dass sie stets den Proposal-Review-Approval-Flow durchlaufen und dass Delivery-Gate-Checks (`DELIVERY_APPROVED.yaml`) eingehalten werden.
5. **Code-Qualität & Style:** Das Repo-Guideline verlangt konventionelle Commit-Nachrichten und deterministischen Python-Code [22] [38]. Die Einrichtung von Pre-Commit-Hooks (flake8, black, mypy) könnte die Einhaltung automatisiert absichern.
6. **Dokumentation:** Das README ist umfangreich, jedoch könnte eine englische Version die internationale Zusammenarbeit verbessern. Zudem fehlen Hinweise zur lokalen Einrichtung (Requirements, Virtualenv) und zur Nutzung von Make-Targets aus den Repo-Guidelines [39].

Roadmap zur Repo-Hygiene & Governance

Die folgenden Schritte bauen aufeinander auf und orientieren sich an den Milestones M2–M8. Sie sind in kurz-, mittel- und langfristige Maßnahmen gegliedert.

Phase 1: Sofortmaßnahmen (nächste 2 Wochen)

Maßnahme	Beschreibung	Begründung und Quelle
Lizenz auswählen und hinzufügen	Wähle eine Open-Source-Lizenz (z.B. MIT oder Apache 2.0) passend zum Projektziel; lege sie als `LICENSE` ins Root und verweise im README darauf.	Aktuell fehlt eine Lizenz; ohne klare Lizenz ist die Nutzung unklar.
CODE_OF_CONDUCT & CONTRIBUTING	Erstelle `CODE_OF_CONDUCT.md` und `CONTRIBUTING.md` mit Verweis auf Rollen und Prozesse aus der Governance.	Fördert Community-Hygiene und definiert Beiträge; aktuell nicht vorhanden.
Legacy-Dateien entfernen	Entferne `docker-compose.yml`, `docker-compose.base.yml`, `docker-compose.dev.yml` und andere als **deprecated** markierte Dateien, nachdem das Team umgestellt hat [36]. Füge `.gitignore`-Einträge hinzu, um Wiedereinchecken zu verhindern.	Die Migration ist in Phase 3 geplant; Entfernen verhindert Verwechslungen und „Repo-Verschmutzung“.
Secret-Scanning konfigurieren	Prüfe, ob gitleaks in der CI-Pipeline läuft; richte `gitleaks.toml` mit den aktuellen Secrets-Regeln ein und verifiziere `.gitleaksignore`. Ergänze GitHub-Secret-Scanning.	Die Tresor- und Infra-Policies untersagen Secrets im Repo [12] [17].
Pre-Commit-Hooks etablieren	Nutze die vorhandenen Skripte (`install-git-hooks.ps1`) zur Installation von Hooks für Code-Formatierung, Linting (flake8/ black) und Commit-Message-Linting (Conventional Commits). Dokumentiere die Verwendung im README.	Unterstützt deterministischen Coding-Style [38].
CI-Pipeline ergänzen	Integriere Test-Coverage-Checks (z.B. `pytest-cov`) und Linting-Jobs in GitHub Actions. Definiere Schwellenwerte (z.B. >80 % Coverage) und mache sie zu Merge-Blockern.	Derzeit sind nur Test- und Lint-Jobs erwähnt; Coverage fehlt.

Phase 2: Mittelfristige Maßnahmen (Q1 2026)

Maßnahme	Beschreibung	Begründung und Quelle
Secrets-Management konsolidieren	Implementiere den im Legacy-Guide beschriebenen Weg: Secrets nur über Docker Secrets (`../.cdb_local/.secrets`), Nutzung eines Secret-Managers (Vault / Sealed Secrets) gemäß Infra-Policy [12] [37]. Entferne `.env`-Files und dokumentiere den lokalen Setup-Prozess.	Vermeidet Leaks und sorgt für künftige Kubernetes-Readiness.
Infrastruktur Hardening (Milestone M2)	Migriere alle Compose-Dateien in `infrastructure/compose` und arbeite mit `stack_up.ps1` laut Legacy-Guide [34]. Implementiere Secrets, Health-Checks und network isolation. Ergänze TLS/SSL für externe Verbindungen.	Erfüllt Infra-Policy (IaC, GitOps, Secrets) [9] [12].
Governance-Gate Enforcement	Überprüfe, ob der Delivery-Gate (`DELIVERY_APPROVED.yaml`) korrekt gesetzt ist; CI/Jira sollten Blockieren, wenn `delivery.approved: false` [4]. Automatisiere die Prüfung in der Pipeline.	Verhindert unautorisierte Mutationen im Working Repo.
Testnetz & Persistenz (Milestones M5 & M7)	Entwickle Replay-fähige End-to-End-Tests, Performance-Tests und Persistenz-Integration. Ergänze Tools zur deterministischen Replay-Funktion.	Policies verlangen deterministische Replays und Auditierbarkeit [40].
Internationalisierung der Dokumentation	Ergänze das README um eine englische Version und extrahiere den Fortschritts-Block in ein automatisch generiertes Dashboard. Verweise auf das Docs Hub für vertiefte Informationen.	Verbessert Lesbarkeit für internationale Stakeholder.
Community-Kommunikation	Definiere eine wöchentliche Governance-Review (Policy-Stack Review) und protokolliere sie im Docs Hub.	Die Governance verlangt regelmäßige Reviews und klare Rollenverteilung [3].

Phase 3: Langfristige Maßnahmen (Q2 2026 → Release 1.0)

Maßnahme	Beschreibung	Begründung
Kubernetes-Readiness & GitOps	Packe alle Services in stateless Containers mit ConfigMaps/Secrets, definiere Resource-Limits und Liveness-Probes. Implementiere FluxCD oder ähnliches für GitOps-Reconcile.	Erfüllt die Infra-Policy (Kubernetes-Readiness [11], GitOps [10]).
Event-Driven Backbone	Migriere von Redis als temporärem Transport zu JetStream/Kafka; implementiere Dual-Write und Replay-Vergleich wie in der Infra-Policy beschrieben [41].	Erhöht Persistenz, Reproduzierbarkeit und Auditierbarkeit.
RL-Safety & Kill-Switch	Implementiere den Risk-Layer mit deterministischen Guardrails und Action-Masking. Richte das Kill-Switch-System ein und dokumentiere Tests zur Verifikation [42] [13].	Erfordert strikte Einhaltung der RL-Safety-Policy.
Penetration Testing & Compliance	Beauftrage ein externes Security-Team (Milestone M8). Prüfe OWASP Top 10, Container-Scanning, Netzwerk-Isolation und Secrets-Management. Schließe offene CVEs.	Erfüllt Milestone M8 und stärkt Sicherheit.
Tresor-Zone Implementieren	Technische Trennung der Tresor-Zone (Keys, Limits, Governance-Docs) vom Trading-System. Stelle offline-Access und Auditing sicher [43] [16].	Schutz vor Verlusten und Missbrauch.
Release-Prozess & Incident-Response	Erstelle einen Release-1.0-Checklisten-Prozess (Milestone M9) mit Security-Sign-off, Monitoring, SLAs und Rollback-Plan.	Finalisiert die Governance-Überarbeitung und bereitet Go-Live vor.

Abschluss und Ausblick

Das Working Repo von Claire de Binare zeigt bereits eine starke Trennung von Code und Wissen, eine aktive CI-Pipeline und einen klaren Milestone-Plan. Die Governance-Dokumente im Docs Hub bilden eine robuste Grundlage, die deterministische Abläufe und strikte Sicherheitsgarantien vorschreibt. Fehlende Elemente wie Lizenz, Verhaltens- und Beitragsrichtlinien, vollständiges Secret-Management und die Entfernung von Legacy-Dateien sollten kurzfristig behoben werden.

Die vorgeschlagene Audit-Roadmap orientiert sich an den bestehenden Policies und Milestones und ergänzt sie um konkrete Maßnahmen zur Repo-Hygiene, Sicherheits-Härtung und Governance-Compliance. Durch regelmäßige Reviews, automatisierte Prüfungen und eine klare Rollentrennung wird gewährleistet, dass das Projekt deterministisch, auditierbar und sicher bleibt.

1 2 3 40 CDB_CONSTITUTION.md
https://github.com/jannekbuengener/Claire_de_Binare_Docs/blob/main/knowledge/governance/CDB_CONSTITUTION.md

4 5 CDB_POLICY_STACK_MINI.md
https://github.com/jannekbuengener/Claire_de_Binare_Docs/blob/main/knowledge/governance/CDB_POLICY_STACK_MINI.md

6 7 8 CDB_AGENT_POLICY.md
https://github.com/jannekbuengener/Claire_de_Binare_Docs/blob/main/knowledge/governance/CDB_AGENT_POLICY.md

9 10 11 12 41 CDB_INFRA_POLICY.md
https://github.com/jannekbuengener/Claire_de_Binare_Docs/blob/main/knowledge/governance/CDB_INFRA_POLICY.md

13 14 42 CDB_RL_SAFETY_POLICY.md
https://github.com/jannekbuengener/Claire_de_Binare_Docs/blob/main/knowledge/governance/CDB_RL_SAFETY_POLICY.md

15 16 17 43 CDB_TRESOR_POLICY.md
https://github.com/jannekbuengener/Claire_de_Binare_Docs/blob/main/knowledge/governance/CDB_TRESOR_POLICY.md

18 22 38 39 CDB_REPO_GUIDELINES.md
https://github.com/jannekbuengener/Claire_de_Binare_Docs/blob/main/knowledge/governance/CDB_REPO_GUIDELINES.md

19 20 21 CDB_REPO_STRUCTURE.md
https://github.com/jannekbuengener/Claire_de_Binare_Docs/blob/main/knowledge/governance/CDB_REPO_STRUCTURE.md

23 24 DOCS_MOVED_TO_DOCS_HUB.md
https://github.com/jannekbuengener/Claire_de_Binare/blob/main/DOCS_MOVED_TO_DOCS_HUB.md

25 26 27 README.md
https://github.com/jannekbuengener/Claire_de_Binare/blob/main/README.md

28 29 30 MILESTONES.md
https://github.com/jannekbuengener/Claire_de_Binare/blob/main/.github/MILESTONES.md

31 32 33 SECURITY_BASELINE.md
https://github.com/jannekbuengener/Claire_de_Binare/blob/main/docs/security/SECURITY_BASELINE.md

34 35 36 37 LEGACY_FILES.md
https://github.com/jannekbuengener/Claire_de_Binare/blob/main/LEGACY_FILES.md