FÜR IT-PARTNER UND ENTWICKLER

Lynk – Technische Übersicht für Ihren *IT-Partner*

Architektur, Technik, Sicherheit, Deployment, Betrieb und Erweiterung – damit Sie entscheiden können, ob Sie Lynk im eigenen Cloud-Konto betreiben und erweitern können. Jede Aussage ist im Quell-Repository nachprüfbar.

6	57	~10k	34	9
DIENSTE	API-MODULE	AUTOMATISIERTE TESTS	CI-WORKFLOWS	SPRACHEN

HP Capital GmbH · Mayrwiesstraße 11 · 5300 Hallwang (Salzburg) · Österreich

Repository-Stand: 24.09.2026

Inhalt

01	Kurzfassung (für die Geschäftsführung)	3
02	Architektur	4
03	Technologie und Versionen	5
04	Daten, Queues und Speicher	5
05	Externe Integrationen	6
06	KI-Schicht: Anbieter, Modellstufen, Nachvollziehbarkeit	6
07	Sicherheit, Zugriff und Mandantentrennung	7
08	Deployment im eigenen Konto	7
09	Systemanforderungen (Schätzung)	8
10	Konfiguration	9
11	Betrieb: Monitoring, Backup, Updates	10
12	Tests und CI	10
13	Lynk erweitern	11
14	Sprachen	11
15	Status: Was es heute gibt	12
16	Grenzen und Einschränkungen	13
17	Vor dem Go-live richten wir ein und prüfen	13
18	Übergabepaket	14

Alle Pfade beziehen sich auf das Lynk-Quell-Repository (Stand 24.09.2026). Dieses Dokument enthält keine Secrets oder Zugangsdaten.

01 Kurzfassung (für die Geschäftsführung)

- **Was es ist:** eine Webanwendung aus sechs Diensten (Python und Node.js) plus den Datenbanken MongoDB und Redis und einem S3-kompatiblen Dateispeicher. Verbreitete Open-Source-Technik – nichts Exotisches.
- **Wo es läuft:** in Ihrem eigenen Cloud- oder Serverkonto (Linux-Server oder Kubernetes-Cluster), in einem Rechenzentrum Ihrer Wahl – für EU-Agenturen empfehlen wir die EU. Wir richten es dort beim Onboarding ein und übergeben alles; wir hosten es nicht. Docker-Images und ein Helm-Chart sind Teil des Repositorys.
- **Was Ihnen gehört:** der vollständige Quellcode, die Deployment-Dateien, die Tests und die Runbooks. Ihr IT-Partner kann Lynk ohne uns betreiben, ändern und erweitern.
- **Was von außen nötig ist:** eigene Konten für Server oder Cloud, Unipile (LinkedIn-Anbindung), einen KI-(LLM-)Anbieter, einen E-Mail-Server und einen Dateispeicher – von Ihnen beauftragt und bezahlt (grobe Kosten in Abschnitt 9). Optional: Microsoft-Login, Stripe.
- **Wie ehrlich dieses Dokument ist:** Jede Aussage bezieht sich auf Dateien im Repository. Wo etwas nicht fertig ist, steht es da (Abschnitte „Status“ und „Vor dem Go-live“).

Diese Übersicht beschreibt den Repository-Stand vom 24. September 2026. Dateipfade in `Monospace` beziehen sich auf dieses Repository; die sechs Dienste liegen unter `lynk-code-backup-20260702-084926/<dienst>__container-0/` (ein historischer Verzeichnisname, bewusst beibehalten – siehe ADR-000 in `docs/adr/`). Pfade unten sind auf den Dienstnamen gekürzt.

02 Architektur

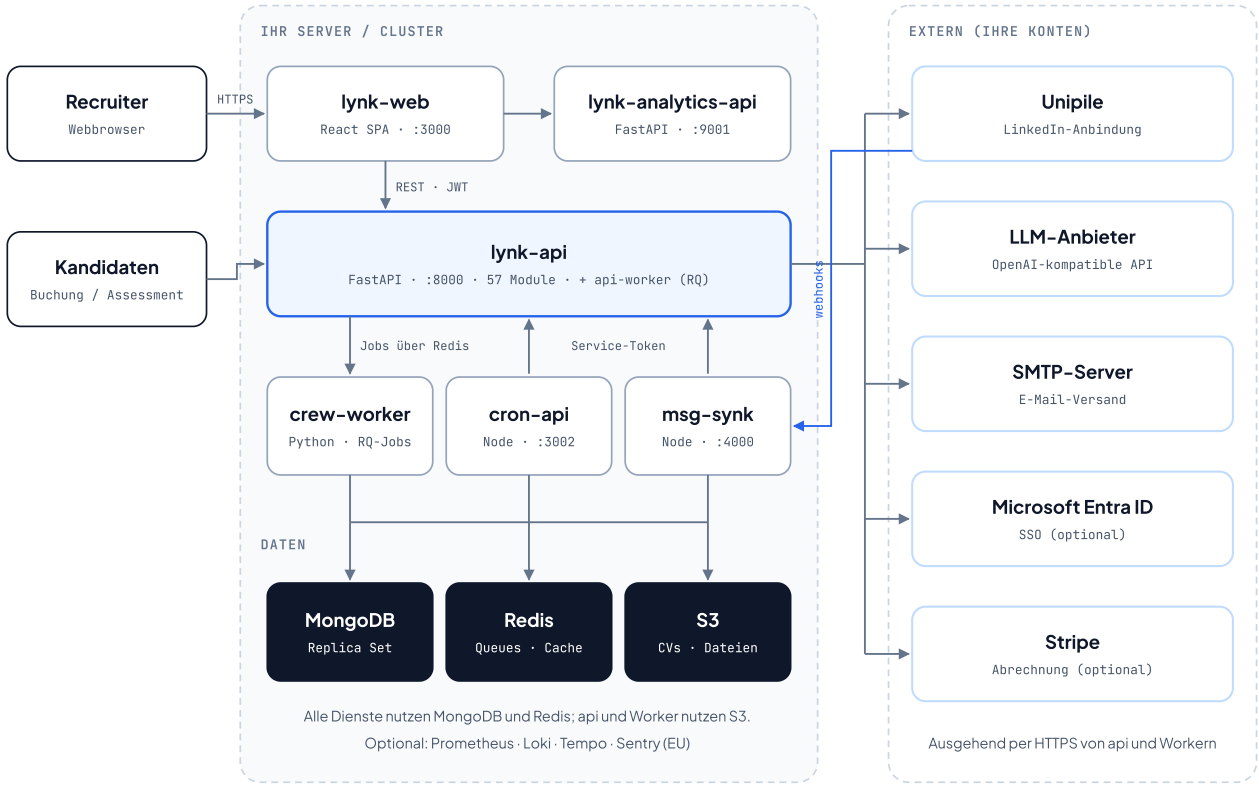


Abbildung 1 – Dienste, Datenspeicher und externe Dienste. Ports wie in docker-compose.yml und im Helm-Chart festgelegt.

DIENST	TECHNIK	AUFGABE
`lynk-web`	React 19 · Vite 6 · TypeScript	Browser-Oberfläche (Single-Page-App) auf :3000. Spricht mit lynk-api und lynk-analytics-api; Microsoft-Login über MSAL.
`lynk-api`	Python · FastAPI	Zentrale REST-API (:8000): Auth, Workspaces, Jobs, Kandidaten, Matching, Nachrichten, Assessments, Buchung, Abrechnung. 57 Fachmodule. Dasselbe Image läuft auch als `lynk-api-worker` (RQ-Queue `api_tasks`).
`lynk-crew-worker`	Python · RQ · CrewAI	Hintergrund-Jobs aus priorisierten Redis-Queues: LinkedIn-Suche und -Import, CV-/Profil-Extraktion, KI-Profilanalyse, Job-Profil-Aufgaben.
`lynk-analytics-api`	Python · FastAPI	Auswertungen, Aktivierungs- und Kohorten-Kennzahlen (:9001).
`lynk-cron-api`	Node 20 · TypeScript · Express	Scheduler (croner): Follow-up-Zeitpläne und periodische Aufgaben; ruft lynk-api mit Service-Token auf (:3002).
`lynk-msg-synk`	Node 20 · TypeScript · Express	Nachrichten-Sync: empfängt Unipile-Webhooks (HMAC-geprüft), speichert LinkedIn-Nachrichten und Verbindungsereignisse (:4000).

Quellen: `DEVELOPMENT.md`, `docker-compose.yml`, `deploy/helm/lynk/values.yaml`, `Dockerfile` und `package.json` / `pyproject.toml` der Dienste.

03 Technologie und Versionen

EBENE	TECHNOLOGIE	VERSION / FESTLEGUNG
Backend (Python)	FastAPI, Pydantic 2, Beanie ODM + mongoengine, RQ, structlog, OpenTelemetry	Python 3.10.12, festgelegt über `uv` (`pyproject.toml`, `uv.lock`); FastAPI 0.139.2 (api) / 0.115.8 (analytics)
KI-Orchestrierung	OpenAI Python SDK, LangChain core, CrewAI	`openai ⩾ 2.46`, `crewai ⩾ 1.15.3`
Backend (Node)	Express 4, Mongoose 8, ioredis, pino, prom-client	Node 20, TypeScript 5
Frontend	React 19, Vite 6, Ant Design 5, Tailwind CSS 3, Redux Toolkit, TanStack Query 5, React Router 6, i18next	TypeScript 5.9, Node-20-Build-Image
Daten	MongoDB, Redis	`mongo:8`, `redis:7-alpine` in Compose; MongoDB 8.0 auf Staging
Container	Docker (multi-stage, non-root user), Helm 3-compatible chart, ArgoCD	Images pro Git-Commit in GHCR (`build-push-ghcr.yml`)

Es gibt kein serverseitiges HTML-Rendering; die Oberfläche ist ein statisches Bundle. Das api-Image enthält ein Headless-Chromium (Playwright) zum Rendern externer Stellenseiten und ist deshalb größer als die übrigen Images.

04 Daten, Queues und Speicher

SPEICHER	VERWENDUNG	HINWEISE FÜR DEN BETRIEB
MongoDB	Alle Anwendungsdaten: Nutzer, Workspaces, Jobs, Kandidaten, Nachrichten, Scores, Entscheidungsprotokoll, LLM-Nutzung, Feature-Flags. 71 Dokumentmodelle im asynchronen ODM registriert (`src/db/beanie_connect.py`).	In Produktion als **Replica Set** betreiben: Zahlungsrelevante Schreibvorgänge nutzen Transaktionen und verweigern auf einem Einzelserver (`src/db/tx.py`). Managed (z. B. MongoDB Atlas, EU-Region) oder selbst betrieben ist beides möglich.
Redis	RQ-Job-Queues (nach Priorität), Scheduler-Locks, Cache, clusterweite Rate-Limit-Zähler.	Persistenz (AOF) aktivieren, damit Jobs einen Neustart überleben; Wiederherstellung in `RQ-RECOVERY.md`.
S3-kompatibler Objektspeicher	CVs und hochgeladene Dokumente.	Jede S3-API (AWS, Cloudflare R2, MinIO, Hetzner …) über `AWS_ENDPOINT`. Bucket privat halten.

Schemaänderungen laufen über einen eingebauten Migrations-Runner (`lynk-api/migrations/`, `python -m migrations.runner status|up`); angewandte Versionen stehen in der Collection `_migrations`, jede läuft genau einmal (`docs/migrations-runbook.md`). Der Code wird schrittweise von mongoengine auf das asynchrone Beanie-ODM umgestellt (ADR-002); heute sind beide Schichten im Einsatz.

05 Externe Integrationen

DIENST	ZWECK	ANBINDUNG
Unipile (LinkedIn)	LinkedIn-Suche, Profilimport, Nachrichten, Einladungen. Jeder Recruiter verbindet sein eigenes LinkedIn-Konto.	REST (`LINKEDIN_*`-Variablen) aus api/Workern; eingehende Webhooks an msg-synk, HMAC-signiert (`WEBHOOK_HMAC_ENFORCE` standardmäßig an). Sendelimits pro Konto in Lynk.
LLM-Anbieter	CV-Extraktion, Profilanalyse, Match-Begründungen, Nachrichtenentwürfe.	Jede OpenAI-kompatible HTTP-API; drei Modellstufen (Abschnitt 6).
SMTP	Transaktions-Mails: Einladungen, OTP-Codes, Terminbestätigungen mit .ics.	Standard-SMTP (`EMAIL_*`). SPF/DKIM für die Absenderdomain einrichten.
S3-Speicher	CVs, Dokumente.	`AWS_*` / `AWS_ENDPOINT`
Microsoft Entra ID	Single Sign-on für Recruiter (optional).	MSAL im Browser (`VITE_CLIENT_ID`, `VITE_TENANT_ID`); Token-Prüfung serverseitig über OIDC-Discovery/JWKS. Ein generischer OIDC-Weg existiert hinter dem Laufzeit-Flag `sso.oidc`.
Ihr ATS / CRM	Import von Stellen, Bewerbungen und Kandidaten; Zurückschreiben, soweit Ihr System und Ihr Tarif es erlauben. Eine Anbindung ist im 8-Wochen-Programm enthalten (eingerichtet und getestet); jede weitere 2.500 € einmalig netto. Lese-/Schreibumfang wird vor dem Kauf schriftlich vereinbart. Bullhorn nur auf Anfrage und nach Freigabe durch Bullhorn für KI-Anbieter.	Konnektor-Framework in lynk-api (`ats_connect/`: Fähigkeitsmodell, Registry, Sync, Ingest). Vorhandene Adapter: d.vinci, Personio, Teamtailor, SAP SuccessFactors. Weitere Systeme werden pro Kunde über die API des Anbieters oder, wo sinnvoll, über einen Integrationsdienst angebunden. Sie verbinden mit Ihrem eigenen Konto beim Anbieter.
Stripe	Code für Abo-Abrechnung vorhanden (Checkout, Webhooks, Mahnwesen).	Für eine eigene Installation optional; `STRIPE_*`.

06 KI-Schicht: Anbieter, Modellstufen, Nachvollziehbarkeit

Alle LLM-Aufrufe laufen über einen Client (`lynk-api/src/common/openai_llm.py`), der die OpenAI-kompatible Chat-API spricht. Der Anbieter ist Konfiguration, kein Code: `LLM_BASE_URL` auf einen kompatiblen Endpunkt zeigen lassen (kommerzieller EU-Anbieter, Azure OpenAI oder selbst betriebener vLLM-/Ollama-Server). Jeder Aufruf hat einen *Zweck*, jeder Zweck eine Modellstufe mit eigenem Modell, Endpunkt und Schlüssel:

STUFE	VERWENDUNG	VARIABLEN
FAST	mechanische Ausgaben (Zahlen, Flags, kurze Listen)	`LLM_MODEL_FAST`, `LLM_BASE_URL_FAST`, `LLM_API_KEY_FAST`
QUALITY	Extraktion (CVs, Stellenbeschreibungen)	`LLM_MODEL_QUALITY`, `…_QUALITY`
PREMIUM	kandidatenseitige Texte (Nachrichtenentwürfe)	`LLM_MODEL_PREMIUM`, `…_PREMIUM`

Nicht gesetzte Stufen fallen auf `LLM_MODEL` / `LLM_BASE_URL` zurück.

- **Nutzungsmessung:** Jeder Aufruf schreibt einen `LLMUsage`-Datensatz (Modell, Zweck, Tokens, Kostenschätzung aus `llm_pricing.json`) – auch Aufrufe aus dem crew-worker.
- **Konfigurations-Fingerprint:** Bei jedem Start schreibt jeder Dienst eine Logzeile `LLM_CONFIG_FINGERPRINT` und ein Dokument `LLMConfigFingerprint` (SHA-256 über Anbieter und Modell/Host je Stufe; nie der Schlüssel, nur der Hostname). Damit lässt sich beantworten, welches Modell einen Kandidaten bewertet hat.
- **Modell-Changelog:** Jede Modell- oder Endpunktänderung braucht eine Zeile in `MODEL_CHANGELOG.md`. Das bereitet die Aufzeichnungspflichten des EU AI Act vor (Recruiting-Scoring ist in Anhang III als Hochrisiko gelistet).
- **Mensch entscheidet:** Der Match-Score ist als automatisierte Einschätzung gekennzeichnet, nicht als Einstellungsentscheidung; Nachrichten sind Entwürfe, die ein Recruiter prüft und sendet; Assessments werden nicht von KI bewertet.

07 Sicherheit, Zugriff und Mandantentrennung

BEREICH	UMSETZUNG
Authentifizierung	Kurzlebige JWT-Access-Tokens (HS256) plus Refresh-Tokens; ein fehlendes Secret führt zur Verweigerung, und lynk-analytics-api lehnt Platzhalter-Secrets oder solche unter 32 Zeichen ab. Login per E-Mail-OTP möglich.
SSO	Microsoft (Entra ID) über MSAL/OIDC; generisches OIDC hinter einem Flag.
MFA	TOTP (Authenticator-Apps), Secrets verschlüsselt gespeichert mit Feld-Verschlüsselungsschlüssel (`src/modules/mfa/`).
Mandanten	Alle Daten gehören zu einem **Workspace**; Hilfsfunktionen in `src/common/tenancy.py` filtern Abfragen und liefern 404 für IDs fremder Mandanten. Isolationstests laufen in der CI.
Dienst-zu-Dienst	Signierte Service-Tokens („Service Principals") zwischen cron-api / msg-synk und lynk-api (`SERVICE_PRINCIPALS_ENABLED`, `SERVICE_JWT_SECRET`).
Härtung	Rate-Limiting (Redis), Größenlimits für Requests, SSRF-Schutz für ausgehende Abrufe, Security-Header, Container ohne Root, HMAC auf eingehenden Webhooks, Secrets nur serverseitig (`VITE_*`-Werte sind bewusst öffentlich).
Lieferkette	Die CI führt Secret-Scanning (Gitleaks), SAST (Semgrep), Container-Scans (Trivy), Dependency-Audit + Dependabot aus und erzeugt pro Main-Build eine SPDX-SBOM (Syft).
DSGVO	Aufbewahrungs-CronJob (`deploy/gdpr/`), Export-/Lösch-Endpunkte (Modul `gdpr`), Marketing-Double-Opt-in hinter einem Flag.

Meldung von Schwachstellen und Reaktionszeiten: `SECURITY.md` (Bestätigung binnen 3 Werktagen, Ziel für Fix oder Abhilfe 30 Tage).

08 Deployment im eigenen Konto

Lynk läuft in **Ihrem eigenen Cloud- oder Serverkonto**, eröffnet auf den Namen Ihrer Agentur bei einem Anbieter Ihrer Wahl. Beim Onboarding installieren wir es dort mit befristetem Admin-Zugang, verbinden Ihre Anbieter, testen mit Ihnen und übergeben alle Zugangsdaten; danach entfernen wir unseren Zugang. Den laufenden Betrieb übernimmt Ihr IT-Team oder IT-Partner. Wir hosten Lynk nicht.

OPTION	WAS IM REPOSITORY LIEGT	GEEIGNET FÜR
Docker Compose	`docker-compose.yml`: MongoDB 8, Redis 7 und alle sechs Dienste aus ihren Dockerfiles, mit Health-Checks.	Lokale Entwicklung und Evaluierung. Die Datei ist als „nicht für Produktion“ gekennzeichnet (einzelnes MongoDB, kein TLS, keine Backups).
Kubernetes (Helm)	`deploy/helm/lynk/`: Umbrella-Chart für alle Dienste inkl. `lynk-api-worker`; Werte je Umgebung, HPA/KEDA, PodDisruptionBudgets, Probes. `deploy/argocd/`: GitOps-Applications (manueller Sync). `infra/`: Network Policies, Monitoring, Backup-Zeitpläne, External Secrets.	Produktion. Referenzumgebung: Single-Node-k3s auf einer Hetzner-VM (Staging). Jedes konforme Kubernetes sollte funktionieren.
Einzelne VM mit Containern	Dieselben Docker-Images; Konfiguration über env-Dateien.	Kleine Installationen. Ein produktionstaugliches Setup (MongoDB als Replica Set, TLS-Reverse-Proxy, Backups) wird beim Onboarding eingerichtet; heute gibt es dafür keine fertige Datei im Repository.

Secrets werden nie eingecheckt. Das Referenz-Setup nutzt Doppler → External Secrets Operator → ein Kubernetes-Secret, das alle Dienste einbinden (`deploy/secrets/`); jeder Secret-Store, der Umgebungsvariablen befüllen kann, funktioniert. **Health-Endpunkte:** api `/health` (Liveness) und `/ready` (503 mit Aufschlüsselung je Abhängigkeit), analytics `/health`, web `/`, msg-synk `/`.

09 Systemanforderungen (Schätzung)

Schätzung – kein gemessener Benchmark. Grundlage sind die Ressourcen-Einstellungen im Helm-Chart und Beobachtungen auf Staging; mit eigener Last validieren.

SZENARIO	RECHENLEISTUNG	DATEN
Evaluierung / Entwicklung (Docker Compose)	1 VM, 4 vCPU, 8 GB RAM, 40 GB SSD	Lokale MongoDB- und Redis-Container
Produktion, eine Agentur (ca. 5–20 Recruiter)	Kubernetes mit 1–3 Knoten à 4 vCPU / 8 GB oder eine VM mit 4–8 vCPU / 16 GB	MongoDB-Replica-Set mit 3 Mitgliedern (managed oder selbst betrieben), anfangs 20–50 GB; Redis mit AOF, 1 GB; S3-Bucket, wächst mit dem CV-Volumen

Zum Vergleich: Die Helm-Standardwerte fordern für die sieben App-Deployments zusammen ca. 1,9 GB RAM und 0,7 vCPU an, mit Limits von ca. 4,5 GB und 7 vCPU (`deploy/helm/lynk/values.yaml`). lynk-api ist speichergebunden (ca. 470 MB auf Staging beobachtet). Monitoring (Prometheus, Grafana, Loki) im Cluster braucht zusätzlich etwa 1–2 GB. LLM-Inferenz ist nicht enthalten – sie läuft beim LLM-Anbieter, außer Sie betreiben ein Modell selbst; dafür ist GPU-Kapazität separat zu planen.

Was Sie direkt bezahlen

Diese Anbieter beauftragen Sie im eigenen Namen und bezahlen sie direkt, nicht uns. Grobe Richtwerte zum Stand dieses Dokuments; die Preise legen die Anbieter fest.

POSTEN	GROBE KOSTEN	HINWEIS
Server- oder Cloudkonto	≈ 50–150 € pro Monat	Abhängig von Anbieter, Region und Größe (siehe oben); inklusive Datenbank und Dateispeicher, wenn beim selben Anbieter.
KI-(LLM-)Anbieter	≈ 200–300 € pro Monat bei 10 Recruitern	Nutzungsabhängig; Lynk misst jeden Aufruf, so sehen Sie die Kosten je Aufgabe.
LinkedIn-Anbindung (Unipile)	≈ 5 € pro Recruiter und Monat	Ein verbundenes Konto je Recruiter.
E-Mail-Versand (SMTP)	gering	Abhängig von Anbieter und Volumen.

10 Konfiguration

Jeder Dienst hat eine `.env.example` mit allen Variablen, die der Code liest, und sicheren lokalen Standardwerten, wo möglich (`DEVELOPMENT.md`). Der Schlüsselsatz für Kubernetes ist in `infra/secrets/lynk-secrets.example.yaml` vorgegeben. Beide enthalten keine echten Werte. Variablengruppen:

GRUPPE	BEISPIELE (NUR NAMEN)
Server / Umgebung	`HOST` , `PORT` , `ENVIRONMENT` , `FRONTEND_URL` , `BACKEND_URL` , `CRON_SERVICE_URL`
MongoDB	`DATABASE_URI` (Python), `MONGO_URL` + `MONGODB_URI` (cron), `MONGO_URI` (msg-synk) – müssen alle auf dieselbe Datenbank zeigen
Redis	`REDIS_HOST` , `REDIS_PORT` , `REDIS_USERNAME` , `REDIS_PASSWORD`
Auth	`JWT_TOKEN_SECRET` , `ACCESS_TOKEN_TTL_MIN` , `REFRESH_TOKEN_EXPIRES_IN` , `FIELD_ENCRYPTION_KEY` , `SERVICE_PRINCIPALS_ENABLED` , `SERVICE_JWT_SECRET`
LLM	`LLM_PROVIDER` , `LLM_BASE_URL[_FAST|_QUALITY|_PREMIUM]` , `LLM_MODEL[_…]` , `LLM_API_KEY[_…]` , `LLM_API_TIMEOUT`
LinkedIn / Unipile	`LINKEDIN_API_KEY` , `LINKEDIN_BASE_URL` , `UNIPILE_WEBHOOK_SECRET` , `WEBHOOK_CALLBACK_URL`
S3	`AWS_ACCESS_KEY_ID` , `AWS_SECRET_ACCESS_KEY` , `AWS_REGION` , `AWS_ENDPOINT` , `AWS_STORAGE_BUCKET_NAME`
E-Mail (SMTP)	`EMAIL_HOST` , `EMAIL_PORT` , `EMAIL_USER` , `EMAIL_PASS` , `EMAIL_SENDER_EMAIL` , `EMAIL_SENDER_NAME`
Stripe	`STRIPE_SECRET_KEY` , `STRIPE_WEBHOOK_SECRET`
Web-Build (öffentlich)	`VITE_API_URL` , `VITE_ANALYTICS_API_URL` , `VITE_CLIENT_ID` , `VITE_TENANT_ID` , `VITE_MS_REDIRECT_URL`
Observability	`SENTRY_DSN` , `OTEL_EXPORTER_OTLP_ENDPOINT` , `LOG_LEVEL`
Feature-Schalter	`*_ENABLED` -Umgebungsschalter, siehe Abschnitt 15

Feature-Flags gibt es auf zwei Ebenen: Umgebungsschalter zum Deploy-Zeitpunkt (`*_ENABLED` , `VITE_*_ENABLED` für die Oberfläche) und Laufzeit-Flags in MongoDB (Collection `FeatureFlag` , 30 Sekunden Cache). Fällt der Flag-Speicher aus, gilt immer „aus“ (`src/common/feature_flags.py`). So sind Deployment und Freischaltung eines Features getrennt.

11 Betrieb: Monitoring, Backup, Updates

Logging und Monitoring

Python-Dienste loggen JSON über structlog, Node-Dienste über pino, jeweils nach stdout (bereit für Loki oder jeden Log-Shipper). Prometheus-`/metrics`-Endpunkte, OpenTelemetry-Traces (OTLP, z. B. Tempo) und Sentry-Fehlerberichte sind eingebaut und optional. `infra/monitoring/` enthält Prometheus-Alarm- und SLO-Regeln (mit Unit-Tests), Grafana-Dashboards und Alertmanager-Konfiguration; `uptime-check.yml` prüft von außerhalb des Clusters. Der LLM-Konfigurations-Fingerprint (Abschnitt 6) gehört zu dieser Spur.

Backup und Notfallwiederherstellung

`deploy/backup/mongo-backup-cronjob.yaml` erstellt nächtlich einen `mongodump`, verschlüsselt ihn gegen ein öffentliches Zertifikat (RSA-4096 / AES-256; der Cluster kann seine eigenen Backups nicht entschlüsseln), schreibt ein SHA-256–Manifest und lädt in S3–kompatiblen Speicher hoch (Standard-Aufbewahrung 14 Tage); ein Erfolgszeitstempel speist einen Alarm „kein aktuelles Backup“. Ein geplanter Restore-Drill-Job und `restore.sh` sind enthalten, ebenso Velero-Zeitpläne (`infra/backup/`). `DR-RUNBOOK.md` beschreibt Neuaufbau, Knotenausfall, Redis-Verlust, kompromittierte Zugangsdaten, fehlerhaftes Deployment und Datenbankkorruption und kennzeichnet jeden Schritt als erprobt, unverifiziert oder Lücke. Zusätzlich empfehlen wir ein managed MongoDB mit Point-in-Time-Recovery.

Updates und Upgrade-Pfad

1. Updates kommen als Commits/Tags in Ihr Repository, mit Changelog.
2. Die CI baut pro Commit einen Image-Satz (`build-push-ghcr.yml`), getaggt mit dem Git-SHA.
3. Zuerst auf Staging deployen; die Smoke-Suite (`tests/staging/run.sh baseline`) muss grün sein.
4. Denselben SHA in Produktion übernehmen (`promote-prod.yml` setzt `values-prod.yaml`; ein Paritätscheck prüft, dass Staging ihn lief), dann in ArgoCD synchronisieren.
5. Ausstehende Datenbank-Migrationen ausführen (`python -m migrations.runner up`).
6. Rollback: vorherigen SHA erneut setzen (`docs/prod-promote-rollback.md`).

Wenn Sie den Code selbst ändern, übernehmen Sie unsere Updates wie jeden Upstream-Branch; die Modulstruktur (Abschnitt 13) hält Konflikte lokal.

Updates sind 12 Monate ab Go-live enthalten. Danach gibt es einen optionalen Wartungsvertrag (Updates und Support), Preis auf Anfrage; ohne ihn läuft Ihre Installation unverändert weiter.

12 Tests und CI

ART	WO	CA. ANZAHL
Python-Unit- und Integrationstests (pytest)	`lynk-api/tests`, `lynk-crew-worker/tests`, `lynk-analytics-api/tests`	~7.600 Testfunktionen in ~550 Dateien (mehr Fälle durch Parametrisierung)
TypeScript-Tests (Vitest)	lynk-web, lynk-cron-api, lynk-msg-synk	~2.600 Tests in ~275 Dateien
Container-Tests (Testcontainers)	`container-tests.yml`	echtes MongoDB/Redis

ART	WO	CA. ANZAHL
Staging-Smoke / End-to-End / visuell	`tests/staging`, `tests/e2e`, `tests/visual`	Marker: smoke, writes, live_only, perf, destructive

34 GitHub-Actions-Workflows (`.github/workflows/`): Lint und Typprüfung, Unit- und Container-Tests, OpenAPI-Drift und -Vollständigkeit, Aktualität generierter API-Typen, Bundle-Größen- und Web-Qualitätsbudgets, Barrierefreiheits-/Visual-Checks, Compose-Smoke, Infrastruktur-Validierung, Security-Scanner (Gitleaks, Semgrep, Trivy, Dependency-Audit), SBOM, Image-Build/-Push und -Aufbewahrung, Staging-Smoke und -Rollback, Prod-Promotion, nächtliche E2E-Tests, Lasttest, Flake-Report und Uptime-Check. Insgesamt rund 10.000 automatisierte Tests.

13 Lynk erweitern

lynk-api ist in Fachmodule unter `src/modules/<name>/` gegliedert. Beim Start bindet `load_routers()` (`src/__init__.py`) jedes Modulpaket mit einer `router.py` ein; Module ohne Router sind Bibliotheken. Ein typisches Modul:

```
src/modules/myFeature/
  __init__.py      # kennzeichnet das Paket
  router.py        # FastAPI-APIRouter - HTTP-Schnittstelle, Auth-Abhängigkeiten
  handler.py       # Geschäftslogik
  schema.py        # Pydantic-Request-Modelle
  responses.py     # typisierte Response-Modelle (für die OpenAPI-Checks)
  model.py         # Beanie-Dokument(e) - in src/db/beanie_connect.py registrieren
  tasks.py         # RQ-Hintergrund-Jobs (optional)
tests/test_my_feature.py
```

- **Wo ergänzen:** Neue Funktionen kommen in ein neues Modul, gemeinsame Hilfsfunktionen nach `src/common/`. Jede Abfrage auf den Workspace beschränken (`tenancy.py`). Neue Features hinter ein Flag legen, damit sie unsichtbar ausgeliefert werden können.
- **Hintergrundarbeit:** als RQ-Job einreihen; lang laufende KI-Aufgaben gehören in den crew-worker, API-nahe Aufgaben in die Queue `api_tasks`.
- **Frontend:** Seiten in `lynk-web/src/Pages`, API-Aufrufe in `src/API`, State in `src/Slices` / `src/Stores`, Texte in den i18n-Katalogen (alle 9 Sprachen, per Paritätstest erzwungen).
- **Konventionen** (`CLAUDE.md`): nie direkt auf `main` committen – Branch → Pull Request → Review; Titel nach Conventional Commits (`feat:`, `fix:`, `refactor:`, `chore:`, `test:`, `docs:`, `ci:`, `perf:`); Code und Doku auf Englisch; nie Assertions abschwächen oder Tests überspringen, um grün zu werden; instabile Tests sind Befunde; ruff und mypy für Python, ESLint und `tsc` für TypeScript; LLM-Modellwechsel brauchen eine Zeile in `MODEL_CHANGELOG.md`.
- **Lokaler Start:** `docker compose up mongo redis`, dann einen Dienst mit `uv run fastapi dev src` oder `yarn dev` starten; `tests/staging/ci_seed.py` legt zwei Test-Mandanten an und gibt ein gültiges JWT aus.

14 Sprachen

Oberfläche und Anschreiben-Vorlagen gibt es in **9 Sprachen**: Englisch, Deutsch, Französisch, Spanisch, Italienisch, Niederländisch, Polnisch, Portugiesisch (Portugal) und Portugiesisch (Brasilien). Das Sprachregister

ist `lynk-api/src/config/locales.json`; sieben weitere Codes sind registriert, für die Oberfläche aber ausgeschaltet. Die Web-Kataloge nutzen i18next; Tests erzwingen gleiche Schlüssel und Pluralformen über alle Sprachen. Eine neue Sprache bedeutet: Katalog ergänzen und Sprache einschalten – ohne Codeänderung in den Modulen. Automatische Maschinenübersetzung (DeepL) ist nicht angebunden.

15 Status: Was es heute gibt

LIVE – ALLE FUNKTIONEN ZUM GO-LIVE EINGESCHALTET

- LinkedIn-Suche und Ein-Klick-Import über Unipile (Verbindung je Recruiter)
- CV-Parsing; Dublettenerkennung (überspringt Dubletten, führt nicht zusammen)
- Match-Score aus 5 Faktoren mit Aufschlüsselung (Skills 40 %, Erfahrung 25 %, Titel 15 %, Ort 10 %, Ausbildung 10 %)
- KI-Entwürfe für Erstnachrichten, geplante Follow-ups, Posteingang, Sendelimits je Konto
- Assessments, Buchungslinks mit .ics, Entscheidungsprotokoll, Auswertungen
- Workspaces, Microsoft-SSO, MFA, 9 Oberflächensprachen
- Jedes OpenAI-kompatible LLM, Modellstufen, Messung je Aufruf, Konfigurations-Fingerprint
- Schriftliche Begründung zu jedem Score (Match v2), jobübergreifendes Re-Matching
- Freigabe-Queue & Sammelaktionen, Pause je Job, Heute-Liste
- ATS-/CRM-Anbindung (eine inklusive, Abschnitt 5)
- E-Mail-Eingang, Outlook-Kalender, Status-Seite für Kandidaten, KI-Hinweis an Kandidaten

BEIM GO-LIVE GESETZTE SCHALTER (FRÜHER HINTER FEATURE-FLAGS)

- Match-v2-Rubrik mit schriftlicher Begründung `MATCH_V2_RUBRIC_API_ENABLED`
- Jobübergreifendes Re-Matching `MATCH_V2_CROSS_JOB_ENABLED`
- Freigabe-Queue & Sammelaktionen `BULK_ACTIONS_ENABLED`
- Pause je Job `AUTOMATION_CONTROL_ENABLED`
- Heute-Liste `VITE_WORK_QUEUE_ENABLED`
- ATS-/CRM-Konnektor-Framework und Sync `ATS_CONNECTIONS_API_ENABLED`, `ATS_SYNC_ENABLED`
- Outlook-Kalender-Push `booking.ms_graph`
- E-Mail-Eingang `EMAIL_INTAKE_ENABLED`
- Kandidatenportal `CANDIDATE_PORTAL_ENABLED`
- KI-Hinweis an Kandidaten `AIA_DISCLOSURE_ENABLED`

NICHT ENTHALTEN

- Weitere ATS-/CRM-Anbindungen (je 2.500 €); Zurückschreiben, soweit nicht vereinbart
- Microsoft Teams
- DeepL-Maschinenübersetzung
- Automatisches Zusammenführen von Dubletten
- Kundenportal

Alle Funktionen der mittleren Spalte sind gebaut und werden in jeder Installation zum Go-live nach den Abnahmetests eingeschaltet. Die Schalternamen stehen hier, damit Ihr IT-Partner weiß, wo sie liegen; manche Features haben zusätzlich einen passenden `VITE_*`-Schalter für die Oberfläche.

16 Grenzen und Einschränkungen

- **LinkedIn** wird nur über Unipile und das eigene Konto des Recruiters erreicht. Es gelten die Nutzungsgrenzen von LinkedIn; Lynk ergänzt Sendelimits je Konto, kann die von LinkedIn aber nicht aufheben.
- **KI-Ergebnisse** hängen vom konfigurierten Modell ab. Scores sind Einschätzungen zur Unterstützung, keine Entscheidungen; es gibt keine automatische Absage.
- **ATS/CRM:** Lesen oder auch Zurückschreiben hängt von System und Tarif ab (Personio z. B. nur Lesen); Umfang vor dem Kauf schriftlich bestätigt, Bullhorn nur mit dessen Freigabe.
- **Compose** ist für die Entwicklung; einzelne Dockerfiles brauchen ggf. kleine Anpassungen an Ihre Registry und Basis-Images (`DEVELOPMENT.md`).
- **Skalierung:** Die meisten Dienste skalieren horizontal; `lynk-cron-api` läuft bewusst mit einer Instanz (geplante Jobs dürfen nur einmal laufen).
- **Repository-Struktur** behält die historischen Snapshot-Verzeichnisnamen; zwei ODM-Schichten (mongoengine, Beanie) bestehen nebeneinander, bis die Migration abgeschlossen ist.
- **Keine Performance-Garantie** über die Schätzung in Abschnitt 9 hinaus; ein Lasttest-Workflow und eine Baseline existieren (`loadtest.yml`, `docs/perf-baseline-2026-08-27.md`).

17 Vor dem Go-live richten wir ein und prüfen

Jede Lizenznehmerin betreibt ihre eigene Instanz im eigenen Cloud-Konto; wir betreiben keine Kundeninstanzen. Release-Stand und Installationsreferenzen auf Anfrage. Jede Installation durchläuft mit Ihrem IT-Partner diese Checkliste:

- [ ] Cloud- oder Serverkonto auf den Namen Ihrer Agentur eröffnen und uns befristeten Admin-Zugang geben; Instanz dort aus einem getaggten Release des übergebenen Repositorys ausrollen; Health-Endpunkte grün; Staging-Smoke-Suite läuft dagegen erfolgreich.
- [ ] LLM-Anbieter wählen (EU-Hosting empfohlen) und mit ihm einen **Auftragsverarbeitungsvertrag (AVV, Art. 28 DSGVO)** abschließen; Modelle je Stufe setzen; in `MODEL_CHANGELOG.md` eintragen; Fingerprint-Zeile im Log prüfen.
- [ ] Dienst-zu-Dienst-Auth aktivieren: `SERVICE_PRINCIPALS_ENABLED` und ein starkes `SERVICE_JWT_SECRET` für api, cron-api und msg-synk – **Voraussetzung für den Versand geplanter Follow-ups**.
- [ ] Neue Secrets erzeugen (JWT ≥ 32 Zeichen, Feld-Verschlüsselungsschlüssel, Webhook-Secrets); nur im eigenen Secret-Manager ablegen.
- [ ] MongoDB als Replica Set (Transaktionen); Redis mit Persistenz und Passwort; privater S3-Bucket.
- [ ] Nächtliches verschlüsseltes Backup läuft; privater Backup-Schlüssel an zwei getrennten Orten verwahrt; **ein Restore-Test durchgeführt und dokumentiert**.
- [ ] Unipile-Konto und Webhook-URL; jeder Recruiter verbindet sein LinkedIn-Konto; Sendelimits geprüft.
- [ ] SMTP-Absenderdomain mit SPF/DKIM; Microsoft-SSO-App-Registrierung, falls genutzt.
- [ ] ATS/CRM über Ihr eigenes Konto verbunden; vereinbarter Umfang getestet.
- [ ] Alarm-Empfänger und Dashboard-Zugang eingerichtet; externer Uptime-Check auf Ihre URLs.
- [ ] Alle Funktionen nach den Abnahmetests eingeschaltet; Übergabe aller Zugangsdaten; unser befristeter Admin-Zugang entfernt.

18 Übergabepaket

Übliche Übergabe-Checklisten für Individualsoftware verlangen Quellcode mit vollständiger Historie, eine Architekturübersicht, Deployment- und Betriebs-Runbooks, Umgebungsvorlagen und eine Liste der Drittkomponenten mit Lizenzen. Das erhalten Sie:

BESTANDTEIL	INHALT	STATUS
Quellcode	Git-Repository mit vollständiger Historie: sechs Dienste, `deploy/`, `infra/`, `tests/`, `scripts/`	vorhanden
Dokumentation	Diese Übersicht; `DEVELOPMENT.md`, `SECURITY.md`, 5 Architekturentscheidungen (ADRs), Observability, Mandantentrennung, Migrationen, LLM-Anbieter-Doku, `MODEL_CHANGELOG.md`	vorhanden
Runbooks	`DR-RUNBOOK.md`, Deploy/Promotion/Rollback, Staging-Rollout, Backup/Restore, Redis-Job-Wiederherstellung (`RQ-RECOVERY.md`)	vorhanden
Umgebungsvorlagen	`.env.example` je Dienst; Kubernetes-Secret-Vorlage – nur Namen, keine Werte	vorhanden
Deployment	Dockerfiles, `docker-compose.yml`, Helm-Chart, ArgoCD-Applications, Backup- und DSGVO-CronJobs, Monitoring-Regeln und Dashboards	vorhanden
CI	34 GitHub-Actions-Workflows (mit Aufwand auf andere CI-Systeme übertragbar)	vorhanden
SBOM	SPDX-Software-Stückliste pro Main-Build (`sbom.yml`)	vorhanden
Open-Source-Lizenzliste	Lizenzübersicht aller Drittkomponenten, abgeleitet aus der SBOM	**wird vor der Übergabe erstellt**
Testdaten	Seed-Skript für zwei getrennte Test-Mandanten (`tests/staging/ci_seed.py`)	vorhanden
Nicht enthalten	Konten und Zugangsdaten bei Drittanbietern (Unipile, LLM-Anbieter, SMTP, Speicher, Stripe, Microsoft): Diese laufen auf Ihren Namen und bleiben bei Ihnen.	–

Fragen Ihres IT-Partners beantworten wir gern vor jeder Entscheidung. Kontakt: HP Capital GmbH, Paul Hagler und Philipp Habring · hello@lynk-recruiting.com