Repo Template and Environment Variable Matrix

Purpose

This document defines the required repository structure, scripts, environment variables, and operational conventions for all repositories in the suite.

It exists to:

- Prevent drift across repositories
- Reduce cognitive load
- Ensure consistent CI and runtime behaviour
- Enforce predictable boot, migration, and health semantics

All new repositories must follow this template unless explicitly approved otherwise.

Repository Types

The suite currently recognises three repository types:

1. Internal Service
2. Public Edge Service
3. Web Client

Each type has mandatory structure and environment requirements.

---

1. Internal Service Template

Required Directory Structure

```
src/
  http/
  domain/
  db/
  auth/
  observability/
  errors/
  config/
  routes/
  pagination/
  logging/

migrations/
tests/
  unit/
  integration/
openapi/
scripts/
```

Notes

- `src/pagination/` must include cursor signing and validation helpers.
- `src/logging/` must include canonical request completion logging helpers.

Required package.json Scripts

```
"scripts": {
  "dev": "...",
  "build": "tsc -p tsconfig.json",
  "start": "node dist/index.js",
  "typecheck": "tsc --noEmit",
  "lint": "eslint .",
  "test": "vitest run",
  "test:watch": "vitest",
  "test:contract": "vitest run tests/contract",
```

```
    "migrate": "node-pg-migrate up",
    "migrate:down": "node-pg-migrate down",
    "openapi:generate": "node scripts/openapi/validate.js",
    "openapi:check": "node scripts/openapi/validate.js"
  }
```

Exact dev runner may vary, but:

- build must fail on type errors
- test must run unit and integration tests
- migrate must apply migrations deterministically
- openapi:check must fail if the committed OpenAPI is invalid

Notes

- `src/errors/` must include:
  - error type definitions
  - error code constants
  - mapping helpers (Zod, Auth, DB, unknown)
  - envelope builder
- For now, the suite uses a hand maintained OpenAPI file as the source of truth.
- openapi:generate and openapi:check are intentionally the same command:
  - generation is a no op
  - validation is mandatory

Shared Error Module Convention

- Each repo must have a single entry point to create error objects and a single entry point to serialise them into the HTTP envelope.
- Prohibit route-local error response shaping.

Canonical Request Completion Logging

Each request must emit exactly one structured completion log event.

Framework default request completion logs must be disabled if they would produce duplicate completion events.

The canonical completion log must include:

- request_id
- method
- path
- status_code
- duration_ms
- caller_service_id when available
- actor_user_uuid when applicable
- organisation_uuid when applicable
- error_code when present
- language, locale, timezone when supplied via headers

If a request fails before route handlers execute (for example auth failure or validation failure), the completion log must still be emitted by the HTTP boundary.

Tests must assert:

- exactly one completion log per request
- required fields present
- error_code present when non 2xx

Cursor Pagination Helpers

Any service implementing cursor pagination must use shared helpers that enforce:

- limit default 50
- limit max 200
- cursor validation before query execution

- HMAC signed opaque cursor format per Principles for All Internal Services
- limit plus 1 fetch rule
- stable deterministic ordering with strict tuple comparison

Cursor payloads must contain only ordering tuple values and must not include internal identifiers.

Required Environment Variables

Core

- NODE_ENV
- PORT

Database

- DATABASE_URL

  Full Postgres connection string (Neon during Replit phase)
- DB_SCHEMA

  Schema owned by this service
- DB_POOL_SIZE
- DB_CONNECTION_TIMEOUT_MS
- DB_IDLE_TIMEOUT_MS

Rules:

- SSL required.
- Each service must use a dedicated DB user restricted to its schema.
- Cross-schema access is prohibited.

Auth0

- AUTH0_ISSUER
- AUTH0_AUDIENCE

- AUTH0_JWKS_URI

If service uses machine to machine:

- AUTH0_M2M_CLIENT_ID
- AUTH0_M2M_CLIENT_SECRET

Observability

- OTEL_EXPORTER_OTLP_ENDPOINT
- OTEL_SERVICE_NAME

Pagination

- CURSOR_HMAC_SECRET

Rules:

- CURSOR_HMAC_SECRET must never be shared across services.
- CURSOR_HMAC_SECRET must never be logged.

Service Specific

Only explicitly documented additional variables permitted.

Considered Respons

- CONSIDERED_RESPONSE_BASE_URL (required) - Base URL for considered-response HTTP API.

Database Invariants and Trigger Policy

If a service spec requires any database level invariant, it must be enforced in migrations.

This includes, but is not limited to:

- updated_at management
- canonicalisation rules (for example lower trim email)
- uniqueness rules beyond basic primary keys

- foreign key integrity beyond application checks

Rules:

- Do not rely on application code as the only enforcement mechanism for these invariants.
- If the spec requires updated_at, migrations must create an UPDATE trigger that sets updated_at = now().
- If the spec requires canonicalisation, migrations must enforce it using one of:
  - trigger enforced rewrite
  - generated canonical column plus constraint
  - check constraints that prevent non canonical values being stored

CI must validate migrations against a clean database, and any required triggers must exist after migration.

---

2. Public Edge Service Template

Inherits Internal Service template plus:

Additional Required Directory Considerations

If real time is enabled:

```
src/
  realtime/
```

Additional Environment Variables

Ably

- ABLY_API_KEY

  Server side only
- ABLY_ENVIRONMENT (optional if using non default)

Rules:

- ABLY_API_KEY must never be exposed to the browser.
- Real time publication must occur only in this service.

CORS

- CORS_ALLOWED_ORIGINS

---

3. Web Client Template

Required Directory Structure

```
src/
  components/
  pages/
  routes/
  services/
  auth/
  hooks/
  state/
  config/

tests/
```

Required Tooling

- React
- Vite
- React Router
- TanStack Query
- Zod
- Tailwind

- Vitest
- Playwright (recommended)

Required package.json Scripts

```
"scripts": {
  "dev": "vite",
  "build": "vite build",
  "preview": "vite preview",
  "typecheck": "tsc --noEmit",
  "lint": "eslint .",
  "test": "vitest run",
  "test:watch": "vitest"
}
```

Required Environment Variables

Auth0

- VITE_AUTH0_DOMAIN
- VITE_AUTH0_CLIENT_ID
- VITE_AUTH0_AUDIENCE

API

- VITE_API_BASE_URL

Ably (client side)

- No server API keys permitted.
- If required:
  - VITE_ABLY_PUBLIC_ENVIRONMENT

Ably client tokens must be obtained from polite-intervention.

Configuration Validation

All services must:

- Validate required environment variables at boot.
- Fail fast if any required variable is missing.
- Never log environment variable values.
- Never log raw thrown errors to clients; only the standard error envelope is returned, and full diagnostics remain in structured logs.

Web clients must:

- Validate required VITE_ variables at build time.
- Fail build if missing.

Health Endpoints (Services Only)

All services must expose:

- GET /health/live
- GET /health/ready

Readiness must verify:

- DB connectivity (if applicable)
- Required configuration present

Migration Policy

- Migrations must be committed.
- Migrations must apply only to the service schema.
- No automatic schema sync at runtime.
- CI must validate migrations against a clean database.

- Migrations must include any required triggers and invariant enforcement declared by the service spec.

OpenAPI Policy (Services Only)

- OpenAPI specification must be maintained in openapi/openapi.yaml.
- OpenAPI file must be committed to repository.
- openapi:check must validate the committed file and fail if invalid.
- CI must fail if openapi:check fails.

OpenAPI Error Schema File

- `openapi/components/errors.yaml` (or an equivalent path) must define:
  - `ErrorEnvelope`
  - `ErrorObject`
- All endpoints must reference it.

Secrets Policy

- No secrets in source control.
- No secrets in client code.
- Server side secrets stored in environment configuration.
- Tokens and secrets must never be logged.

Naming Conventions

- Service repositories: kebab-case
- Schema names: snake_case matching service name
- Environment variables: UPPER_SNAKE_CASE

- UUIDs used for all public identifiers

Change Discipline

Any deviation from this template must:

- Be explicitly documented
- Be approved at architectural level
- Include reasoning

Required repo documentation artifact

Each repository must include a top level README.md that is implementation aligned and kept current.

Purpose:

Make the repo self describing for a new engineer without needing external context, while enforcing suite invariants and preventing unsafe shortcuts.

Rules:

- Must be derived from the code and the referenced suite documents for this repo
- Must not claim behavior not present in the implementation
- Must state network boundary, auth model, and leakage rules explicitly
- Must include a concrete local run path and required environment variables
- Must document health endpoints and their exact semantics
- Must document observability wiring and how to validate it locally
- Must document database ownership and migration procedures when applicable

Standard README.md outline

1. Service summary
    - What this repo is
    - Primary responsibilities
    - Non responsibilities
    - Link to the canonical service overview doc in this repo
2. Architecture placement
    - Where it sits in the suite
    - Upstream callers
    - Downstream dependencies
3. Network boundary
    - Public vs internal
    - Who may call it
    - Who must never call it
4. Authentication and authorization
    - Token types accepted
    - Audience, issuer expectations
    - For edge services: end user token validation and downstream token exchange flow
    - For internal services: M2M token validation and any caller allowlisting rules
    - Delegated actor context headers, if used
    - What this service does and does not authorize
5. API surface
    - Link to OpenAPI file path
    - How to view or generate docs
    - Any versioning rules
    1. Error handling and leakage rules

- Response envelope shape
- JSON only
- No stack traces or internal details to clients
- Logging redaction rules

6. Environment variables
    - Table of variables, required vs optional
    - Which are required at boot
    - Which are migration only
    - Validation and fail fast behavior
7. Database
    - Does this service own a schema
    - Schema name
    - Connection modes: pooled vs direct
    - Migration commands
    - Seed commands
    - search_path policy if relevant
8. Observability
    - Logs, metrics, traces
    - OpenTelemetry config
    - Grafana Cloud destinations
    - How to confirm locally
9. Health endpoints
    - Liveness path and behavior
    - Readiness path and behavior
    - Exact dependencies checked by readiness
10. Local development

- Install
- Configure env
- Migrate and seed
- Run
- Smoke test commands

11. Security and operational invariants
    - Rate limiting and timeouts where applicable
    - Idempotency behavior where applicable
    - Prohibited behaviors list
12. Contributing
    - Test commands
    - Lint and typecheck commands