TECHNICAL AND GOVERNANCE MANUAL

Technical Documentation

Derived directly from the current markdown manual suite with governed football stationery and DBG publishing conventions.

Source: 08_technical_documentation.md

CAF and FIFA references are contextual standards markers only. Federation-authoritative operational data remains runtime truth.

FECOFA TECHNICAL DOCUMENTATION

1. OVERVIEW

FECOFA is a Next.js 16 application backed by Prisma and PostgreSQL for managing football federation structures, organizations, members, competitions, passports, and operational workflows.

The platform currently supports a federation-aware operating model with:

- federation switching
- federation-scoped data access
- modular Prisma seeding
- local Docker-backed development
- digital passport and compliance workflows

This document is the technical reference for developers working on the application.

---

2. CORE STACK

- Next.js 16
- React 19
- TypeScript
- Prisma 5
- PostgreSQL
- Redis
- Tailwind CSS 4
- Vitest
- Docker Compose

Key package scripts include:

- `npm run dev`
- `npm run dev:local`
- `npm run validate`
- `npm run validate:release`
- `npm run build`
- `npm test -- --run`
- `npm run db:setup:dev` (development only)

- `npm run db:migrate:deploy`
- `npm run db:seed`
- `npm run db:setup`
- `npm run docker:dev:deps`
- `npm run docker:dev:seed`
- `npm run docker:dev:bootstrap`

---

3. PRODUCT MODEL

The normalized internal hierarchy is:

Federation -> Province -> Cercle -> Organization -> Team -> Member

> **IMPORTANT**
> IMPORTANT

- these are normalized schema nouns
- federation-specific UI labels such as County, District, Region, Branch, or Sub-County are presentation labels only
- business logic and database queries must continue to use the normalized Prisma model

Main entities:

- **Federation**: country-level football authority
- **Province**: federation-scoped territorial node
- **Cercle**: federation-scoped local node under a province
- **Organization**: club or academy
- **Member**: player, coach, referee, or staff
- **Competition**: federation-scoped competition
- **Team**: organization-level playing group
- **CPERecord**: compliance/professional education tracking

---

4. FEDERATION ARCHITECTURE

4.1 ACTIVE FEDERATION RESOLUTION

Federation selection is user-driven, but database truth remains authoritative.

Current model:

- the browser stores the requested federation selector in a cookie
- runtime federation resolution is performed server-side
- the active federation must exist in the database and be active
- config is used for branding, locale, timezone, and UI label mapping
- config is not the source of database truth

Relevant runtime files:

- `lib/federation/config.ts`
- `lib/federation/getActiveFederation.ts`
- `lib/federation/scope.ts`
- `app/actions/setFederation.ts`

4.2 FEDERATION SCOPE SAFETY

Federation safety is enforced through centralized scope helpers. These helpers are used to verify that an entity belongs to the active federation before reads or writes continue.

Examples include:

- `getActiveFederationContext()`
- `resolveActiveFederation()`
- `assertCercleInFederation()`
- `assertOrganizationInFederation()`
- `assertCompetitionInFederation()`
- `assertMemberInFederation()`
- `assertMemberPassportInFederation()`
- `assertTeamInFederation()`
- `resolveScopedOrganization()`

Use centralized helpers instead of:

- fetching by ID globally and comparing country codes later
- hand-rolling federation checks in each page or server action
- trusting UI federation labels for business logic

4.3 FEDERATION BRANDING

Federation config currently carries:

- code
- display name
- official full name
- short name
- official website URL metadata
- official logo metadata
- optional totem metadata
- locales
- default locale
- timezone
- UI hierarchy labels
- verification preferences
- branding colors

Configured CAF federation identity rollout:

- `CD`, `KE`, and `GH` remain the reference entries with approved logo and totem assets.
- the remaining configured CAF federation codes are now promoted into explicit config entries instead of being synthesized from a placeholder generator.
- explicit entries now define official federation name, short identity label, timezone, locale policy, and staged asset state even when `logoPath` is intentionally `null`.
- generated fallback remains only as a neutral safety net for unknown or future federation codes outside the explicit registry.

Official naming policy:

- use each federation's real official federation full name in `officialName`
- `name` mirrors `officialName` for shared UI consistency
- avoid placeholder identities such as `<code> Football Federation` for CAF federations already modeled in config
- `shortName` remains optional and can be acronym/code for compact UI areas

Branding is injected into the UI through CSS custom properties at the root layout level.

Brand identity model:

- federation identity remains the primary operational brand across the suite
- DBG / Societe Debonhomme SARL is the owner and publisher layer and should appear elegantly in shell footers, legal notices, and manual exports
- AU-inspired watermarking is an atmospheric decorative layer only and must stay visually restrained
- CAF and FIFA references are contextual standards markers only and must stay visually bounded
- country flags now act as the lead federation mark across shared portal and high-visibility shell surfaces for this branding mission

- official federation logos remain optional secondary marks where helpful and authorized
- optional totem/emblem is secondary and decorative
- flag-derived federation colors power identity accents (headers, selectors, hero accents)

Brand hierarchy rules:

1. Federation identity
2. DBG owner / publisher layer
3. AU atmospheric layer
4. CAF / FIFA contextual reference layer

Brand governance source of truth:

- `lib/branding/governance.ts` is the formal code-backed source of truth for hierarchy, surface policy, legal copy, watermark ceilings, and manual/PDF stationery text
- `components/branding/BrandGovernancePrimitives.tsx` contains the reusable governed primitives for identity marks, governance ribbons, legal notices, portal cards, and bounded contextual fallback callouts
- `BRANDING.md` is the maintainers' playbook for hierarchy, surface rules, lead-mark policy, and anti-pattern review

Brand governance matrix summary:

- shell and portal surfaces are federation-led, flag-first, and may only use DBG, AU, and CAF/FIFA as bounded secondary context
- footer and legal surfaces are DBG-led for publishing legitimacy, but must not imply runtime ownership over federation operations
- manuals and PDFs remain federation-led, allow logo-first treatment when an approved official logo exists, and keep AU stationery below print readability risk thresholds
- compact selector chips, empty states, and operational callouts must remain text-led and must not introduce decorative or third-party football clutter

LEGAL AND RUNTIME SAFEGUARDS

LEGAL AND RUNTIME SAFEGUARDS

- branding metadata remains presentation-only
- translated or branded copy must not become routing, RBAC, scope, or persistence truth
- FIFA language should be framed as an international regulatory reference, not as the literal runtime source of truth
- CAF language should be framed as a continental governance context, not a business-logic switch
- do not hardcode unapproved third-party football marks into random components

Portal-level flag strategy:

- the root federation portal uses ISO country-code derived Unicode flag marks as the lead federation indicator
- shared page headers and shell chrome may also use flag-first presentation for stronger national identity signaling

- if a federation code cannot produce a safe flag mark, render a neutral federation-code token instead
- never reuse another federation's flag, logo, or totem as a fallback
- flag rendering is presentation-only and never participates in federation resolution, routing, RBAC, or DB truth

Official federation website policy:

- official federation website links are stored as presentation metadata in `lib/federation/config.ts`
- show the link only when a valid `http` or `https` URL is configured
- never fabricate placeholder or guessed URLs in the UI
- missing website metadata must degrade cleanly with no broken link rendering

AU watermark policy:

- AU-inspired color watermarking and football line art may be used as low-opacity background treatment on high-visibility surfaces
- watermarking must remain subtle, enterprise-grade, and accessibility-safe
- watermarking must never sit behind dense text with harmful contrast loss
- mobile surfaces should use further reduced AU opacity ceilings
- printable exports should keep AU line art and accent opacity below the governed print ceiling
- watermarking is decorative only and must not affect logic or navigation

Accessibility and color semantics policy:

- federation identity colors remain presentation-only brand accents
- semantic status meaning must use dedicated accessible status treatments instead of raw federation colors
- critical meaning must never depend on color alone; pair status color with text labels, icons, counts, or patterned fills
- shared semantic utilities live in `app/globals.css` and should be preferred for status chips, callouts, and progress bars
- informational links must not be styled as success states unless they truly represent successful operational outcomes
- focus states must remain highly visible against both dark and light surfaces
- decorative overlays, watermarks, and glow effects must yield to readability on dense operational surfaces

Manual and PDF accessibility rules:

- manual exports must preserve high-contrast body text as the default reading mode
- important notes, warnings, and operational boundaries should render as boxed callouts with explicit labels
- decorative football stationery must remain low-opacity and must not compete with paragraph readability
- generated PDFs must keep federation branding presentation-only and separate from runtime operational truth
- federation identity should remain the title and masthead lead, while DBG remains the publisher and footer legal identity
- CAF and FIFA references must remain contextual support text only, never the lead document owner

Accessibility governance contract:

- the source of truth lives in `lib/accessibility/governance.ts`
- future status surfaces should prefer shared primitives in `components/accessibility/SemanticPrimitives.tsx`
- the contract must remain durable across dark, light, branded, dashboard, manual, and PDF surfaces
- engineering reviews should reject any feature that restores color-only meaning, unreadable decorative overlays, or unlabeled analytics bars

Accessibility governance matrix:

Surface	Required controls	Prohibited regressions
Shell and navigation	visible focus ring; readable contrast over brand surfaces; branding remains presentation-only	hidden focus state; decorative overlay behind dense navigation copy
Cards and heroes	high-contrast text blocks; subdued decoration; outcome labels stated in text	glow-only meaning; busy watermark behind body copy
Badges and status chips	semantic tone; explicit label text; optional icon/count companion	raw federation colors as status truth; unlabeled colored pills
Alerts and callouts	semantic tone; boundary label; readable border and background contrast	decorative-only banner; muted text on tinted surface
Forms and validation	textual validation state; visible focus; error or success summary beyond color alone	green/red only messaging; low-contrast helper text
Passport and eligibility	eligibility wording; payment/compliance companion label; progressbar semantics	decorative framing without explanation; hidden eligibility meaning
Analytics and charts	direct values on every point; patterned fills; accessible labels for each measure	hue-only comparison; silent bars without numeric context
Manuals and PDFs	readable default body text; boxed operational callouts; presentation-only branding disclaimer	dense watermark under paragraphs; brand color as instructional meaning
Empty states and zero-data views	explicit empty-state wording; neutral semantic treatment; next-step guidance	blank panels; unexplained color cue

Chart accessibility rules:

- label every chart or trend point with a direct value
- expose an accessible name for each chart measure or progress treatment
- prefer patterned fills or a second cue in addition to tone

Brand versus accessibility enforcement:

- federation colors can frame identity, but they cannot become success, warning, or failure truth on their own
- semantic outcomes must use the governed tone system even when brand palettes are visually present nearby
- manual and PDF branding must stay subordinate to readability and operational comprehension

Switcher option model:

- shared federation switchers should receive a presentation-ready option shape:

- `code`
- `name`
- `shortName`
- `officialName`
- `logoPath`
- optional `totemPath`
- switcher triggers should be logo-first and show compact federation text without relying on raw config internals

Explicit registry helpers:

- `listExplicitFederationConfigs()` returns the first-class configured federation entries.
- `isExplicitFederationConfig(code)` identifies whether a federation belongs to the explicit registry.
- `listPendingFederationLogoCodes()` identifies explicit entries still in staged logo rollout.

Asset conventions:

- official logos: `/public/federations/<code>-logo.svg`
- optional totems: `/public/totems/<code>-totem.svg` (or existing federation-specific totem file)

Fallback behavior:

- the app must never fall back to another country's logo/totem (for example DRC leopard) when active federation assets are missing
- when `logoPath` is missing, render a neutral federation-code placeholder mark
- missing optional totem must not block rendering
- unknown federation codes must resolve to a neutral config (no inherited DRC logo/totem identity)

Safety rules:

- branding metadata remains presentation-only
- business logic, federation scope enforcement, RBAC decisions, and persistence rules must not depend on logo/totem assets
- locale and i18n behavior remains independent from branding assets

Staged rollout guidance:

- logos can be onboarded incrementally per federation using `logoPath`
- keep `logoPath: null` until an authorized official asset is available
- do not fabricate or substitute another federation's logo while waiting for approved assets
- explicit federation identity does not require immediate logo completion; locale, naming, timezone, and palette can be promoted first while assets remain staged

How to add a new explicit federation entry:

1. add the federation code and metadata in lib/federation/config.ts
2. set `officialName`, `shortName`, timezone, and any hierarchy / verification overrides needed

3. add an approved logo at `/public/federations/<code>-logo.svg` only when the authorized asset is available

4. optionally add a totem asset under `/public/totems/`

5. keep `logoPath: null` and `totemPath: null` during staged rollout instead of reusing another federation's identity assets

6. verify tests covering locale policy and asset existence still pass

Explicit vs fallback config:

- explicit federation config is a curated presentation record for known federation codes

- fallback federation config is a neutral safety net for unknown or future codes

- neither explicit nor fallback config changes federation-scoped DB truth, RBAC, or business rules

4.4 LOCALE AND DICTIONARY RESOLUTION

The platform uses one centralized dictionary layer.

Primary files:

- `lib/i18n/dictionaries.ts`

- `lib/i18n/locales.ts`

- `lib/i18n/getDictionary.ts`

- `app/actions/setLocale.ts`

- `lib/federation/config.ts`

Supported platform locales:

- `en`

- `fr`

- `ar`

- `pt`

- `es`

Federation locale policy model:

- `locales`: supported locales for that federation

- `defaultLocale`: default locale for the federation

- `localeDisplayOrder`: preferred UI ordering for locale selector display

Federation policy is runtime presentation policy only; it must not change business, RBAC, federation scope, or persistence rules.

Resolution order:

1. `NEXT_LOCALE` cookie (only if supported by the active dictionary and allowed by federation locale policy)

2. federation `defaultLocale` from active federation config

3. first supported locale from federation `localeDisplayOrder` / supported locale list

4. explicit global fallback locale (`en`)

> **IMPORTANT**
> safeguards:

- missing or unsupported locale codes never crash rendering
- dictionary lookup always degrades to a supported fallback
- locale controls are presentation-only and must not affect business logic or federation scope checks

Dictionary conventions:
- keep copy grouped by domain (`auth`, `adminUsers`, `monitoring`, `analytics`, `registration`, etc.)
- prefer adding keys to centralized dictionaries instead of embedding repeated strings in pages/components
- keep keys stable and descriptive; avoid ad-hoc string bags
- when adding a new locale, add translation patch coverage for high-value operator/admin surfaces first
- dictionary expansion must preserve key-shape consistency across locales

Language switcher rules:
- only locales supported by the active federation are shown
- unsupported locale requests are normalized to federation policy before persisting cookie state
- operators should never be offered locale options outside federation policy

What must remain non-localized:
- internal security/audit event names (for example `ACTION_FORBIDDEN`, `ANALYTICS_VIEW_BLOCKED`)
- internal metric keys and operational identifiers consumed by monitoring/reporting
- data model enums and normalized hierarchy/business identifiers

---

5. KEY APPLICATION SURFACES

Main pages currently include:
- `/` federation portal landing page
- `/[province_slug]`
- `/[province_slug]/[cercle_slug]`
- `/[province_slug]/[cercle_slug]/[org_slug]`
- `/registration`
- `/competitions`
- `/match-reports`
- `/analytics`

- `/medical`
- `/referees`
- `/sporting`
- `/intradesk`
- `/my-football`
- `/planning/agenda`
- `/passport/[id]`
- `/tms`

When editing these surfaces:

- prefer federation-aware service/query helpers over inline Prisma logic
- avoid repeating relational federation filters in every page
- keep page components focused on rendering prepared view data where practical

5.1 FEDERATION PORTAL HOMEPAGE

The root page now acts as a federation portal instead of a generic dashboard-first entry.

Current root-page responsibilities:

- list all active federations from the DB-backed active federation registry
- render one clickable federation portal tile per federation
- use country flags as the lead mark on that page only
- surface official federation website links only where valid metadata exists
- keep selection cookie-backed and server-resolved through the existing federation selection action
- expose quick links for the currently selected federation context
- expose a bottom PDF manual library derived from the markdown manual suite

> **IMPORTANT**
> routing and selection rules:

- the root page must not invent a second federation-resolution system
- federation portal tiles must reuse the selection mechanism in `app/actions/setFederation.ts`
- active federation still resolves from the browser cookie plus server-side DB validation in `lib/federation/getActiveFederation.ts` and `lib/federation/scope.ts`
- translated display names, flag marks, and manual labels must never become routing keys or business identifiers

Manual PDF surface:

- the root page renders grouped PDF links for shared, operator, stakeholder, and technical manuals
- PDF links are generated from the manual catalog in lib/manuals/catalog.ts

- PDFs are served by the bounded on-demand route [app/manual-pdfs/[slug]/route.ts](app/manual-pdfs/[slug]/route.ts)
- route responses use predictable filenames via `Content-Disposition`
- PDF rendering applies premium stationery, DBG legal footer treatment, football line art, and bounded CAF/FIFA contextual framing

5.2 STAKEHOLDER PRODUCT LAYER

The stakeholder surface extends the core member model without collapsing the distinction between PlatformUser and Member.

Primary files:

- `app/my-football/page.tsx`
- `components/stakeholder/StakeholderPortfolio.tsx`
- `lib/stakeholder/access.ts`
- `lib/stakeholder/portfolio.ts`
- `lib/stakeholder/metrics.ts`

Product rules:

- stakeholder identity is anchored on `PlatformUser` through `StakeholderProfile`
- linked player/family/guardian/representative access is bounded by `MemberStakeholderRelationship`
- premium visibility is consent-led and federation-scoped
- self-service player access remains available through `SELF` relationships without requiring third-party consent approval
- medical, payment, selection, and transfer visibility must be individually gated

Operational guidance:

- do not expose passport identifiers, compliance state, medical cases, invoices, or transfer signals to non-self stakeholders when consent is pending, expired, suspended, or revoked
- keep stakeholder portfolio reads scoped to the active federation
- use derived access helpers instead of duplicating consent logic inside UI components

---

6. SERVER ACTIONS

Server actions under app/actions/ are federation-sensitive. They must not perform unscoped reads or writes.

When updating or creating an action:

1. resolve the active federation centrally
2. scope entity lookup by federation in the query itself where possible

3. use shared scope assertions for org, competition, member, team, or cercle ownership
4. fail fast if the target entity is outside the active federation
5. do not fetch globally and compare federation after the fact unless absolutely necessary

Typical federation-sensitive operations include:

- registration verification
- organization activation
- competition creation
- match reporting
- match events
- transfers
- payment processing
- injury prediction
- scouting
- telemetry
- compliance updates

---

7. PRISMA AND DATABASE

7.1 SCHEMA

Prisma schema is defined in:

- `prisma/schema.prisma`

The app uses PostgreSQL as the primary database.

7.2 LOCAL BOOTSTRAP REALITY

Build and test success do not guarantee database readiness.

These commands do not create DB tables:

- `npm test -- --run`
- `npm run build`

The Prisma schema must be applied before runtime pages can query tables like Province.

7.3 LOCAL BOOTSTRAP FLOW

Recommended manual flow:

docker compose -f docker-compose.yml -f docker-compose.dev.yml up -d db redis

npm run db:setup

npm run dev

Convenience flow:

npm run dev:local

Docker-assisted bootstrap flow:

npm run docker:dev:bootstrap

7.4 MISSING TABLE DEVELOPER GUIDANCE

The home page includes developer-safe handling for missing Prisma tables in non-production so local bootstrap issues produce a precise setup message instead of a vague crash.

This is a development aid only and should not be used to hide real production failures.

---

8. ENVIRONMENT HANDLING

8.1 HOST-RUN NEXT.JS

When running the app directly on your machine with npm run dev, use:

DATABASE_URL="postgresql://postgres:postgres@localhost:5432/fecofa?schema=public"

8.2 CONTAINER-RUN SERVICES

Inside Docker Compose, services use Docker DNS hostnames, for example:

- `db:5432`
- `redis:6379`

Do not mix the two contexts silently.

Use .env.example as the local baseline.

8.3 PRODUCTION RUNTIME VALIDATION (FAIL FAST)

Runtime configuration is validated centrally in:

- `lib/config/runtime.ts`

The runtime will fail fast on startup in production mode when enforcement is enabled and critical settings are unsafe or missing.

Primary validation targets:

- `DATABASE_URL` must be set and use PostgreSQL protocol
- `SESSION_TOKEN_PEPPER` must be present and high entropy in production
- `ALLOW_DEV_AUTH_BYPASS` must never be enabled in production
- session/token TTL values must be positive numbers

Enforcement behavior:

- production runtime enforcement is enabled by default in container runtime
- enforcement is intentionally skipped during build phase so CI builds can complete before runtime wiring

8.4 REQUIRED PRODUCTION ENVIRONMENT VARIABLES

Minimum required production variables:

NODE_ENV=production

DATABASE_URL=postgresql://...

SESSIONTOKENPEPPER=<high-entropy-secret-at-least-32-characters>

FECOFAENFORCECONFIG=1

ALLOWDEVAUTH_BYPASS=0

Optional:

SESSIONCOOKIEDOMAIN=.example.org

SESSIONTTLHOURS=12

CREDENTIALTOKENTTL_HOURS=2

Never enable in production:

- `ALLOW_DEV_AUTH_BYPASS=1`

- weak or placeholder `SESSION_TOKEN_PEPPER`

8.5 COOKIE AND SESSION RUNTIME SAFETY

Session cookies are set with production-safe defaults:

- `httpOnly: true`
- `secure: true` in production
- `sameSite: lax`
- `priority: high`
- explicit `path: /`
- optional domain via `SESSION_COOKIE_DOMAIN`

Session and auth token hashes are peppered using validated runtime secret material (SESSIONTOKENPEPPER) so raw token reuse is hardened against hash-table style attacks.

---

9. HEALTH AND READINESS

Operational endpoints:

- `GET /health/live` (liveness)
- `GET /health/ready` (readiness including DB query check)

Behavior:

- `/health/live` returns service health heartbeat
- `/health/ready` returns HTTP `200` only when database query succeeds
- readiness returns HTTP `503` when DB is unavailable

These endpoints are used by Docker health checks and are safe for orchestrator probing.

---

10. SEEDING ARCHITECTURE

The seeding system has been modularized.

Entry points:

- `prisma/seed.ts`
- `prisma/seeding/orchestrator.ts`

Main seeding layers:

- `prisma/seeding/configs/`
- `prisma/seeding/core/`
- `prisma/seeding/stages/`
- `prisma/seeding/runners/`

The seeding system supports presets and country selection behavior through the orchestrator and preset config.

Current design goals:
- keep `prisma/seed.ts` thin
- drive federation expansion through config
- support DRC territorial seeding separately from broader CAF federation packs
- preserve federation-safe seeding assumptions

10.1 SEEDING COMMANDS

```
npm run db:seed
npm run db:setup
npm run docker:dev:seed
```

10.2 SEEDING GUIDANCE

When extending seeding:
- do not return to one giant `prisma/seed.ts`
- keep static country and territory data in config files
- use centralized factories/helpers for IDs, organizations, teams, members, and competitions
- keep federation semantics aligned between runtime config and seeded DB records

---

11. TESTING AND VALIDATION

Primary checks:

```
npm test -- --run
npm run build
```

Recommended local verification after infrastructure/bootstrap changes:

docker compose -f docker-compose.yml -f docker-compose.dev.yml up -d db redis
npm run db:setup
npm run dev

Then verify:
- homepage loads successfully
- federation switching works
- competition and organization data stay federation-scoped
- no page regresses due to missing schema or broken scope checks

Disaster recovery verification command:

npm run ops:verify:recovery

---

12. DISASTER RECOVERY AND BACKUP/RESTORE OPERATIONS

Operational recovery tooling is available through npm scripts:

npm run ops:backup
npm run ops:restore -- --file <backup.dump> --confirm
npm run ops:verify:recovery

Restore safety controls:
- restore requires explicit `--confirm`
- production restore additionally requires `--allow-production`
- optional `--reset-target` supports reset-and-restore workflows

Recovery scope checks include:
- DB connectivity
- critical table existence
- active federation presence
- liveness/readiness endpoint responses

Full operator runbook:
- `DISASTER_RECOVERY.md`

---

13. PRODUCTION RELEASE AUTOMATION (BOUNDED)

Release helper commands:

npm run ops:preflight -- --mode production

npm run ops:deploy -- --base-url https://your-hostname

npm run ops:verify:release -- --base-url https://your-hostname

Preflight goals:

- validate runtime env safety assumptions
- verify required release/runtime artifacts exist
- enforce production safety gate expectations (`ALLOW_DEV_AUTH_BYPASS=0`, strong `SESSION_TOKEN_PEPPER`)

Deploy helper goals:

- keep startup ordering explicit (db/redis before web)
- apply schema update (`db:migrate:deploy`) in runtime context before web rollout
- run post-deploy verification before considering rollout healthy

Post-deploy verification goals:

- DB connectivity
- active federation presence
- `/health/live`
- `/health/ready`
- `/sign-in` reachability

Primary release runbook:

- `RELEASE_PROCEDURES.md`

---

14. CURRENT ENGINEERING CHECKPOINTS

Recent hardening milestones include:

- local Prisma bootstrap hardening
- developer-safe missing-table guidance on the homepage
- federation-safe action scoping centralization
- stronger scope helpers in `lib/federation/scope.ts`
- modular seeding orchestration

- persistent monitoring and bounded operational metrics
- production runtime configuration validation and health/readiness endpoints

These changes should be treated as architectural baselines, not temporary patches.

---

15. ENGINEERING RULES FOR FUTURE CHANGES

1. Do not bypass centralized federation scope helpers in high-risk reads or writes.
2. Do not use UI hierarchy labels as business-logic truth.
3. Do not assume `build` or `test` implies DB readiness.
4. Do not reintroduce monolithic seeding.
5. Prefer scoped DB lookups over fetch-then-compare federation checks.
6. Keep host-local and container DB connection assumptions explicit.
7. Keep page components thinner over time by moving reusable scoped queries into services/helpers.
8. Keep development bypass flags disabled in production at all times.
9. Treat health/readiness endpoints as operational interfaces and keep them lightweight.

---

16. SUGGESTED NEXT DOCUMENTATION ADDITIONS

Future docs worth adding:
- `ARCHITECTURE.md` for deeper system architecture
- `SEEDING.md` for country presets and federation rollout rules
- `ACTIONS.md` for server-action safety conventions
- `DEPLOYMENT.md` for production Docker and environment guidance

For end-user operational guidance, use the role-based manual suite:
- `USER_MANUAL.md` as the master index
- `manuals/00-overview.md` for shared orientation
- `manuals/*.md` for role-specific guides

16.1 ROLE-BASED USER MANUAL GOVERNANCE

The FECOFA user-manual system is now structured as a suite instead of one generic end-user document.

Current manual architecture:

- `USER_MANUAL.md`: master index, audit summary, and role chooser
- `manuals/00-overview.md`: shared orientation and identity rules
- `manuals/01-federation-leadership.md`
- `manuals/02-provincial-official.md`
- `manuals/03-cercle-administrator.md`
- `manuals/04-club-academy-operator.md`
- `manuals/05-referee-match-official.md`
- `manuals/06-player.md`
- `manuals/07-parent.md`
- `manuals/08-legal-guardian.md`
- `manuals/09-manager-representative.md`

Manual PDF delivery architecture:

- the markdown suite remains the source of truth
- homepage PDF links are produced from the shared manual catalog in lib/manuals/catalog.ts
- PDF bytes are generated on demand from markdown by [app/manual-pdfs/[slug]/route.ts](app/manual-pdfs/[slug]/route.ts)
- PDF covers and interior page chrome use a shared premium stationery system with AU-inspired watermark treatment, football line art, and DBG publisher footer lines
- use predictable slugs and filenames when adding a new manual PDF entry
- do not hand-wire homepage PDF links independently from the manual catalog

Maintenance rules:

1. keep route references aligned to real App Router routes
2. do not describe a role as having access to a screen that the current product does not justify
3. keep federation leadership-only surfaces such as `/admin/users` and `/admin/monitoring` out of local or stakeholder manuals unless the role truly owns them
4. keep stakeholder documentation aligned with the consent-led access rules in `lib/stakeholder/access.ts`
5. keep passport and identity guidance consistent with the identity architecture in this document
6. prefer updating the relevant role manual instead of expanding `USER_MANUAL.md` back into a generic all-purpose guide
7. keep DBG ownership language elegant and restrained in markdown manuals and PDF exports
8. keep CAF/FIFA references contextual and never imply they override federation-authoritative runtime data

When adding a new manual:

1. confirm the role or persona is justified by the current product, routes, and access model
2. add the new file under `manuals/`
3. register the manual in lib/manuals/catalog.ts so homepage PDF delivery stays in sync

4. add the manual to `USER_MANUAL.md`
5. update this governance section if the suite structure changes
6. verify that route references, identity guidance, and scope boundaries remain truthful

End-user manuals versus technical docs:

- user manuals explain how a role uses FECOFA screens
- technical documentation explains architecture, runtime, development, and engineering constraints
- release, backup, restore, health, and disaster recovery procedures should stay in technical or runbook documentation, not in normal role manuals

---

17. IDENTITY ARCHITECTURE

14.1 PURPOSE

The identity stack separates internal system identity from human-facing football identity and government identity.

14.2 IDENTITY STACK

1. `Member.id`
- Internal database primary key.

2. `Member.digitalPassportId`
- Internal durable passport token.
- Used for joins and system-safe references.

3. `Member.careerNumber`
- Human-facing immutable 9-digit football lifetime number.
- Globally unique across the platform.

4. `Member.nationalId`
- Optional external government identity.

14.3 CAREER NUMBER RULES

- Exactly 9 digits (`^\d{9}$`).
- Opaque and non-semantic (no federation, location, role, or personal data encoded).
- Assigned once at first registration.

- Never changed, regenerated, or recycled.
- Must survive transfers, organization changes, and role evolution.

Examples:

- `1` -> `000000001`
- `25` -> `000000025`
- `120045` -> `000120045`

14.4 ISSUANCE AND BACKFILL

- Issuance is sequence-backed via centralized helper logic.
- Existing members are backfilled through `prisma/backfillCareerNumbers.ts`.
- `npm run db:setup` delegates to `npm run db:setup:dev` for local development and executes:

1. schema push
2. seed
3. career-number backfill

- Production release and recovery paths must use `npm run db:migrate:deploy` instead of schema push or seed commands.

14.5 OPERATIONAL IDENTIFIER POLICY

- Public/operational identifier: `careerNumber`
- Internal system token: `digitalPassportId`
- Government verification identifier: `nationalId`

Lookup workflows may accept any of:

- `careerNumber`
- `digitalPassportId`
- `nationalId`

PREMIUM MANUAL DOCUMENT STANDARD

All user and role manuals within the FECOFA Digital Hub must adhere to the premium blueprint standard:

- **Reference Blueprint:** The unified premium layout is required across the board. Every manual must include confidentiality, source authority, audience, surfaces, operational scope, alerts, escalation paths, and documentary governance sections.
- **Role Accuracy:** Formatting is standardized, but capabilities must remain specific to the role. Stakeholder manuals must not imply administrator powers. Operator manuals must state their exact Province, Cercle,

Organization, Member, passport, CPE, payment, and routing boundaries.

- **Metadata:** Manuals must be registered in `lib/manuals/catalog.ts` under the correct grouping so the homepage catalog and PDF route remain synchronized.

- **PDF Rendering:** The PDF generation operates through the shared renderer in `lib/manuals/pdf/render.ts`, served by `app/manual-pdfs/[slug]/route.ts`. Do not create bespoke PDF designs. Update markdown predictably so every manual benefits from the global stationery, watermark, accessibility callout, and legal footer.

- **PDF Safety:** Manual sources must remain safe for the current standard-font PDF renderer. Avoid emojis, smart punctuation, and unsupported Unicode until the font stack is upgraded. Validate with `npm run manuals:sanitize` and the manual catalog tests.

2026 MANUAL UPDATE & GOVERNANCE

- The FECOFA Premium Manual defines the structural standard.

- All generated manuals must adhere to a shared corporate blueprint enforcing confidentiality, audience scope, explicit permissions, daily operating cycles, identity preservation (`careerNumber`, `digitalPassportId`, `nationalId`), escalation discipline, and DBG documentation standards.

DOCUMENT GOVERNANCE

Federation identity remains the primary operational brand. DBG establishes publisher legitimacy and document polish. AU atmosphere stays subdued. CAF and FIFA references remain contextual and restrained.

DBG SARL - Societe Debonhomme SARL

Siege social : 44, Avenue du Livre, Gombe, Kinshasa, Republique Democratique du Congo