Glossaire

Glossaire

Glossaire

> *Dernière mise à jour : 2026-05-10*

A

ADR (Architecture Decision Record) : note courte qui documente une décision d'architecture (contexte, décision, conséquences). Voir le chapitre `Adrs`.

Anchor cards : cartes obligatoires dans une decklist (mode deckbuilding). Le prompt Opus a une règle stricte « tu DOIS inclure ces cartes ».

Argon2id : algorithme de hash recommandé pour les mots de passe (lib `argon2`). Utilisé pour `users.password_hash`.

B

Banc d'éval RAG : suite de questions test exécutées contre l'API, avec un judge Haiku qui compare la réponse à un attendu. Subjectif. Voir `tests/rag/` + `npm run test:rag`.

bge-m3 : modèle d'embeddings (1024 dim) servi par TEI. Bilingue, support sparse BM25 natif Qdrant.

bge-reranker-v2-m3 : cross-encoder qui re-classe les top-K candidats du retrieval. Servi par TEI sur `:8990`.

BM25 : algorithme de sparse retrieval keyword-based. Implémenté nativement par Qdrant.

C

`canAddGames` : permission booléenne sur user. Quand `true`, l'utilisateur a accès à `/add-game`. Admins bypass.

Chunk : morceau de texte indexé dans Qdrant (~paragraphe). Préfixé par son contexte LLM (Contextual Retrieval B).

Chunking : étape d'ingestion qui découpe le texte d'un PDF en chunks sémantiquement cohérents.

ClaudeQuotaError : exception levée par `claude-quota.ts` quand le CLI Claude renvoie un message de saturation quota. Contient `resetAt: Date`.

Conflict detect : étape d'ingestion (extensions seulement) qui détecte si un chunk d'extension `replaces` / `modifies` / `extends` un chunk du jeu de base.

Contextual Retrieval B : technique d'amélioration retrieval. Pour chaque chunk, Claude génère 1-2 phrases de contexte (avec le doc complet en input). Le contexte préfixe le chunk avant embedding.

CORS : Cross-Origin Resource Sharing. Whitelist `CORS_ORIGIN` du backend Hono.

D

Deckbuilding (intent) : 4e intent à côté de `rules` / `synergy` / `meta`. Déclenche un pipeline dédié pour produire des decklists structurées exécutables.

Deck import (FAB) : modal qui parse un decklist Fabrary collé en texte → preview → attache au chat (sticky cap 80).

Decompose-query : étape qui appelle Haiku pour décomposer une question synergy/deckbuilding en JSON structuré (couleurs, format, types, etc.). Multi-query par TCG.

DotGG : ancien fournisseur de méta-game pour Lorcana. Data figée 21 nov 2025, inutilisable depuis.

Drizzle ORM : ORM TypeScript type-safe. Migration : `drizzle-kit generate` puis `migrate`.

E

ed25519 : algorithme de clé SSH moderne. Utilisé pour la clé dédiée oracle (`/app/ssh/id_ed25519`).

`envBool()` : helper dans `src/config.ts` pour parser les booléens d'env. Évite le piège `Boolean("false") === true`.

F

FAB (Flesh and Blood) : TCG. Données via package npm `@flesh-and-blood/cards` bundlé dans l'image Docker.

Fabrary : builder de deck communautaire FAB. Source du deck import (texte "Copy as Text").

ForceCommand : option SSH qui force l'exécution d'un wrapper, peu importe la commande envoyée par le client. Utilisé pour `/home/oracle/bin/oracle-claude.sh`.

Fusion RRF v2 : variante du RRF avec pondération question×2 + top-rank bonus + blending position-aware. Cf. ADR-005.

H

Haiku : modèle Claude rapide. Utilisé pour HyDE, decompose-query, classify intent, contextual, conflict detect (`claude-haiku-4-5-20251001`).

Heartbeat (SSE) : event `{ type: 'heartbeat' }` émis toutes les 8s. Empêche NPM de fermer la connexion à `proxy_read_timeout 60s`. Filtré côté client par `useEventStream.skipHeartbeat`.

Hierarchy LLM : étape d'ingestion qui demande à Claude d'analyser le doc complet et produire l'arbre chapter > section > sub-section. Chaque chunk reçoit `hierarchy_path`.

Hono : framework web TypeScript moderne (cf. ADR-001). Backend de l'app.

HyDE (Hypothetical Document Embedding) : technique RAG où Haiku génère un passage hypothétique du livret pour la question, puis on embed ce passage. Améliore le recall.

I

Idempotent : opération qu'on peut répéter sans effet de bord supplémentaire. Le client BookStack `ensure-shelf` / `upsert-page` est idempotent (le push ne duplique pas).

Ingest : pipeline d'ingestion d'un PDF (PDF → chunks → Qdrant). 11 étapes, voir `pipeline-rag/flux-ingestion.md`.

Intent : classification d'une question (`rules` / `synergy` / `meta` / `deckbuilding`). Détermine le pipeline RAG.

L

Lorcana : TCG Disney/Ravensburger. Données via LorcanaJSON (MIT). Méta DotGG figée.

M

Méta-game : informations sur le métagame compétitif (tier list, archétypes, decklists tournois). Indexé en chunks `[META]`.

MTG (Magic: The Gathering) : TCG. Données via Scryfall bulk + traduction Haiku.

N

NPM (Nginx Proxy Manager) : reverse proxy sur Unraid. Gère TLS Let's Encrypt + routes vers `rules.thymon.fr`.

O

OCR (Optical Character Recognition) : reconnaissance de texte sur image. Phase 1 : tesseract auto sur PDFs scannés (`OCR_ENABLED=true` par défaut). Phase 2 prévue avec Claude vision en fallback.

Opus : modèle Claude le plus puissant. Utilisé pour les réponses RAG et le mode deckbuilding.

Oracle (VM) : VM SSH dédiée qui exécute Claude Code CLI. Verrouillage 4 couches.

P

`pdftoppm` : binaire (paquet `poppler-utils`) qui rend les pages PDF en PNG. Utilisé à 300 DPI pour la vision Claude.

pdfjs-dist : lib JS d'extraction texte PDF. Utilisée à l'étape 1 du pipeline d'ingestion.

Pinia : state management Vue 3. Stores : `auth`, `games`, `session`.

`pending` **(role)** : statut user en attente de validation admin. Bloque `/play`, `/history`, etc.

Q

Qdrant : vector DB. Une collection par jeu (`rules_<slug>`) + collections par TCG.

Quota Claude : limite usage du compte Pro/Team (fenêtre 5h glissante). Géré par `claude-quota.ts` + reprise auto via scheduler.

R

RAG (Retrieval-Augmented Generation) : architecture où on récupère des chunks pertinents avant de générer la réponse LLM. Cœur du projet.

Reranker : modèle qui re-classe les candidats du retrieval. bge-reranker-v2-m3 servi par TEI.

Resync (admin) : action `/admin` qui compare la collection Qdrant à la source courante et applique les diffs. Ne télécharge PAS la nouvelle source — c'est aux scripts npm.

Retrieval : étape qui récupère les top-K chunks pertinents pour une question. Hybrid (dense + sparse) + RRF + rerank.

Result (discriminé) : pattern de retour de handler `{ ok: true; ... } | { ok: false; status; error }`. TypeScript force à gérer tous les cas.

Riftbound : TCG Riot Games (univers League of Legends). Données via API card-gallery live.

RRF (Reciprocal Rank Fusion) : algorithme de fusion de listes classées. Score = `Σ 1/(60 + rank)` (variante v2 utilisée ici, cf. ADR-005).

S

Scryfall : source data MTG. Bulk JSON `all_cards.json` téléchargé localement.

SSE (Server-Sent Events) : protocole streaming HTTP unidirectionnel. Utilisé par `/api/ask/stream`, `/api/games/ingest`, `/api/admin/games/:id/sync-cards`.

Sticky mentions : cartes citées aux tours précédents, accumulées côté frontend pour contexte multi-tours. Cap FIFO 20 (80 si deck attaché).

T

TCG (Trading Card Game) : jeu de cartes à collectionner. 6 supportés actuellement.

TEI (Text Embeddings Inference) : service HuggingFace qui sert les modèles d'embeddings. Tournant sur la RTX 3060.

tesseract : OCR open source. Wrapper `services/ocr/tesseract.ts` (CLI invoqué via `execFile`).

Tier list : classement des archétypes du métagame (S/A/B tier). Source typique : Mobalytics, MTGGoldfish.

V

`validateModel()` : valide qu'un nom de modèle Claude matche `/^[a-z0-9-]{1,40}$/`. Anti-injection shell.

Vision inline : Claude lit le PNG d'une page directement via son outil `Read` au moment de la question (pas pendant l'ingestion). Cf. ADR-004.

W

WAL (Write-Ahead Log) : mode SQLite par défaut. Permet writes concurrents avec reads. Fichiers `.db` + `.db-wal` + `.db-shm`.

`withTimeout` : wrapper de promesse `(promise, ms, label) => Promise<T>`. Évite les blocages de boot et de RAG.

Z

Zod : lib de validation TypeScript. Utilisée pour tous les inputs API + les env vars (`src/config.ts`).