Maintenance

- Backup et recovery
- Mise à jour des dépendances
- Logs
- Monitoring
- Troubleshooting

Backup et recovery

Backup et recovery

> *Dernière mise à jour : 2026-05-10*

⚠ À COMPLÉTER

Aucune stratégie de backup applicative n'est implémentée dans le code (pas de script `scripts/backup.sh`, pas de cron dans `src/cron/`, pas de doc dans ARCHITECTURE.md / CONTRIBUTING.md).

La protection des données dépend actuellement entièrement de la stratégie Unraid native (RAID + parity + snapshots ZFS si activé sur le pool `appdata`).

À clarifier / décider :

- ☐ **Pool storage** : `appdata` est-il sur cache SSD ZFS avec snapshots, ou sur array (parity + array disks) ?
- ☐ **Snapshots ZFS** : si oui, fréquence ? Rétention ?
- ☐ **Backup off-site** : il y a-t-il un backup vers un autre Unraid / cloud ?
- ☐ **Périodicité testée** : as-tu déjà testé un restore ? Quand ?

Données à protéger

Quoi	Où	Volume	Critique ?
SQLite (`database.db`)	`/mnt/user/appdata/boardgame-referee/data/database.db`	~10-50 MB	Élevé : users, jeux, questions, feedbacks
PDFs uploadés	`/mnt/user/appdata/boardgame-referee/pdfs/*.pdf`	100 MB - quelques GB	Élevé : irrécupérables si perdus (à moins d'avoir l'original)

Quoi	Où	Volume	Critique ?
PNG rendus	`/mnt/user/appdata/boardgame-referee/pdfs/images/`	Quelques GB	Faible : régénérables via `pdftoppm`
Caches JSON	`/mnt/user/appdata/boardgame-referee/data/{ocr,contexts,conflicts}-*-<slug>.json`	Quelques MB	Moyen : régénérables via Claude (mais cher en quota)
Data sources cartes	`/mnt/user/appdata/boardgame-referee/data/{magic,lorcana,terraforming-mars,ark-nova}-cards/`	Quelques GB	Moyen : re-téléchargeables sauf TM/Ark Nova qui sont locaux statiques
Card image cache	`/mnt/user/appdata/boardgame-referee/data/card-images-cache/`	Quelques GB	Faible : régénérable sharp depuis CDN
Qdrant collections	`/mnt/user/appdata/boardgame-referee/qdrant/storage/`	Quelques GB	Élevé : ré-ingestion = qq heures + quota Claude
Clé SSH oracle	`/mnt/user/appdata/boardgame-referee/ssh/id_ed25519`	<1 KB	Critique : sans elle, plus d'oracle SSH
NPM config	`/mnt/user/appdata/nginx-proxy-manager/`	Quelques MB	Élevé : recréer manuellement long

Recommandations (à valider)

Niveau minimum

1. **SQLite** : `sqlite3 .backup` quotidien vers un autre disque ou cloud
2. **PDFs** : rsync vers un autre disque ou cloud (ils sont irrécupérables)
3. **Clé SSH oracle** : copie sur un support sécurisé (Bitwarden chiffré ou USB)

Niveau confortable

4. **Qdrant snapshots** : pris au moment des grosses ré-ingestions (cf. `deploiement/rollback.md` pour les commandes)
5. **Caches JSON** : optionnel, ils valent juste ce qu'ils ont coûté en quota Claude
6. **NPM config** : backup `database.sqlite` NPM quotidien

Snapshots Qdrant

Manuel pour l'instant :

```
# Snapshot d'une collection
curl -X POST http://192.168.10.100:6333/collections/<collection>/snapshots
# Renvoie un nom de snapshot

# Liste les snapshots
curl http://192.168.10.100:6333/collections/<collection>/snapshots

# Télécharger un snapshot pour archive
curl -O http://192.168.10.100:6333/collections/<collection>/snapshots/<snapshot-name>
```

Restore SQLite

```
# Stop le container
docker compose -f /mnt/user/appdata/boardgame-referee/docker-compose.yml stop app

# Restore depuis backup
cp /backup/path/database.db.bak /mnt/user/appdata/boardgame-referee/data/database.db
# Ou si WAL : restore aussi .db-wal et .db-shm

# Restart
docker compose -f /mnt/user/appdata/boardgame-referee/docker-compose.yml up -d app
```

Restore Qdrant

```
# Re-create la collection depuis le snapshot
curl -X PUT http://192.168.10.100:6333/collections/<collection>/snapshots/recover \
  -H "Content-Type: application/json" \
  -d '{"location": "<snapshot-url>"}'
```

Test de restore

⚠ **Une procédure de restore non testée n'existe pas.** Quand tu as une stratégie de backup en place, force-toi à tester un restore au moins une fois par trimestre :

1. Sur un container test sur Unraid avec un volume vide

2. Restore SQLite + Qdrant
3. Vérifier qu'on peut login + voir les jeux + reposer une question

---

Questions à toi (Thymon)

- **Quel est ton workflow Unraid actuel pour `appdata` ?** (RAID? Snapshots? Off-site?)
- **As-tu déjà perdu un fichier sur ce projet ?** (incident antérieur qui a forcé un fix)
- **Veux-tu qu'on ajoute un script de backup applicatif** (sqlite3 .backup + rsync PDFs) en cron sur Unraid ?

Mise à jour des dépendances

Mise à jour des dépendances

> *Dernière mise à jour : 2026-05-10*

Politique : manuel

Pas de Renovate / Dependabot configurés. Mises à jour manuelles, en bloc, quand tu en as envie.

Pourquoi pas Renovate ?

- Volume de PR auto trop élevé sur un projet d'un seul mainteneur
- Préférence pour batch les mises à jour (1× / 2-3 mois)
- Évite la friction "PR Renovate qui casse la CI le mardi matin"

Si tu changes d'avis : Renovate self-hosted via Docker, conf simple `renovate.json` à la racine.

Workflow de batch update

Backend

```
# 1. Voir les outdated
npm outdated

# 2. Patch updates (safe — bug fixes)
npm update
```

```
# 3. Minor / major sélectifs
npm install hono@latest
npm install drizzle-orm@latest
npm install zod@latest
npm install @types/node@latest

# 4. Vérifier
npm run build
npm test

# 5. Tester en dev
npm run dev
# Aller sur /play, poser quelques questions, vérifier RAG OK

# 6. Si tout OK : commit + push → CI build + push image → pull en prod
git add package.json package-lock.json
git commit -m "chore(deps): bump backend deps"
git push
```

Frontend

```
cd frontend
npm outdated
npm update

# Major upgrades
npm install vue@latest vue-router@latest pinia@latest
npm install tailwindcss@latest @tailwindcss/vite@latest
npm install marked@latest

npm run build
npm test
npm run dev
# Tester l'UI

cd ..
git add frontend/package.json frontend/package-lock.json
git commit -m "chore(deps): bump frontend deps"
```

```
git push
```

Cas spéciaux

`@flesh-and-blood/cards` + `@flesh-and-blood/types`

Ces deux packages contiennent les **données cartes FAB**. Mettre à jour = nouveau set FAB. Workflow détaillé : `tcg-integrations/mettre-a-jour-cartes.md`.

```
npm update @flesh-and-blood/cards @flesh-and-blood/types
# Build + push obligatoire pour que le container voie les nouvelles cartes
```

`mana-font` (frontend)

Webfont Andrew Gioia pour symboles MTG. Update rare (pas de nouveau symbole MTG depuis longtemps). Si nouvelle release :

```
cd frontend
npm install mana-font@latest
# Vérifier visuellement qu'aucun symbole existant ne casse
```

`pdfjs-dist`

Couche d'extraction PDF. Updates régulières mais souvent breaking. Tester sur quelques PDFs de jeux différents avant de push.

`better-sqlite3`

Driver SQLite synchrone. Update prudemment — il y a souvent des prebuilds Node-version-specific. Si update casse en CI :

- Vérifier compat Node 22
- Si Alpine cible : prebuild musl absent → bloque tout, voir `pipeline-cicd.md` pour le contexte

Audit de sécurité

```
npm audit
cd frontend && npm audit
```

Souvent du bruit (vulnérabilités CVSS bas dans des transitive deps). Filtrer :

```
npm audit --audit-level=high
```

Update Node 22 → 23 / 24

Pas urgent. Mais quand tu décides :

1. Modifier `engines.node` dans `package.json`
2. Modifier le base image dans `Dockerfile` (`FROM node:24-bookworm-slim`)
3. Modifier le base image dans `.gitea/workflows/build.yml` (container test)
4. `npm ci` localement avec Node 24, vérifier que tout compile / test

Pas de major version pinning

`package.json` utilise `^` partout (= compatible minor). Pas de pinning strict. Si tu veux pin une version pour reproduire un bug :

```
npm install hono@4.7.6 --save-exact
```

`--save-exact` retire le `^`.

Logs

Logs

> *Dernière mise à jour : 2026-05-10*

Logger central

Tous les logs serveur passent par `src/lib/logger.ts`. Conventions :

- **Aucun** `console.*` dans `src/` (sauf `src/config.ts` qui boot avant le logger)
- Préfixer le scope : `logger.info('[meta-sync] ...')`, `logger.error('[ssh] ...')`
- Niveaux : `debug` / `info` / `warn` / `error` (filtrable via `LOG_LEVEL`)
- Format prod : JSON ligne (parseable via `jq`)
- Format dev : pretty print (lisible sans outil)

Locations

- **Stdout / stderr Docker** : `docker logs boardgame-referee`
- **Fichier persistant** : `/app/data/logs/server.log` (volume Unraid `data/`)

Rotation

Géré par `entrypoint.sh` au boot :

- Si `server.log > 50 Mo` → archive en `server-YYYYMMDD-HHMMSS.log`
- Purge automatique des archives `> 30 jours`

Pas de rotation runtime (only at boot). Si tu fais beaucoup de DEBUG en prod, pense à restart le container ou logrotate manuellement.

Filtrer

```
# Live tail
docker exec boardgame-referee tail -f /app/data/logs/server.log

# Live tail avec scope filter (jq pour le JSON)
docker exec boardgame-referee tail -f /app/data/logs/server.log | jq 'select(.scope == "rag")'

# Grep classique
docker exec boardgame-referee grep "claude-quota" /app/data/logs/server.log | tail -50

# Toutes les erreurs des 24 dernières heures
docker exec boardgame-referee grep -E '"level":"error"' /app/data/logs/server.log | tail -100
```

Niveaux par défaut

Environnement	`LOG_LEVEL`
Dev	`debug`
Prod	`info`

Pour passer en debug en prod (temporaire) :

```
# Modifier .env Unraid : LOG_LEVEL=debug
# Restart container :
docker compose -f /mnt/user/appdata/boardgame-referee/docker-compose.yml restart app
```

⚠ N'oublie pas de remettre en `info` après — debug est verbeux et remplit vite les 50 Mo.

Logs notables à surveiller

Pattern	Sens
`[claude-quota] ClaudeQuotaError`	Pause auto enclenchée, vérifier `resetAt`
`[ssh] command rejected`	Wrapper oracle a bloqué une commande (debug `/tmp/oracle-debug.log` si nécessaire)

Pattern	Sens
`[ingest] stage error`	Stage d'ingestion en échec
`[meta-sync] purge`	Purge hebdo des chunks méta périmés
`[forum-sync]`	Cron forums BGG (3h du matin par défaut)
`[hyde] timeout`	HyDE Haiku a dépassé 25s, fallback question brute
`[contextual-cache] flushed`	Cache JSON sauvé sur disque (avant quota error ou checkpoint)
`[rag] retrieve`	Diagnostics retrieval (timings, scores) en debug

Pas de log aggregation externe

Pas de Loki / Grafana / ELK / Sentry actuellement. Tout reste dans le fichier Unraid + Docker logs.

Si un jour tu veux ajouter Sentry ou un log shipper :

- Ajouter une lib (`@sentry/node`)
- Wrapper le logger pour aussi pousser vers Sentry sur `level === 'error'`
- Configurer `SENTRY_DSN` env var
- Le code reste centralisé dans `src/lib/logger.ts` — les autres fichiers n'ont pas à savoir qu'on log ailleurs

Debug d'une question

Pour relire les diagnostics d'une question donnée (lecture seule) :

```
docker exec boardgame-referee npm run debug:question -- --id <questionId>
```

Affiche : question, réponse, chunks retrouvés, HyDE, timings — équivalent CLI du panneau `/admin/feedback/<id>`.

Monitoring

Monitoring

> *Dernière mise à jour : 2026-05-10*

État actuel : minimal

L'app expose un seul endpoint health, qui sert au healthcheck Docker. Pas de Prometheus / Grafana / OpenTelemetry / Sentry actuellement.

`GET /api/health`

```
{
  "status": "ok",
  "timestamp": "2026-05-10T12:34:56.789Z"
}
```

Public (pas d'auth). Utilisé par le healthcheck Docker (`docker-compose.yml`) :

```
healthcheck:
  test: ["CMD", "wget", "-qO-", "http://localhost:3000/api/health"]
  interval: 30s
  timeout: 5s
  retries: 3
  start_period: 10s
```

Si 3 échecs consécutifs → container marqué `unhealthy`. Pas de restart auto en place — c'est manuel.

`GET /api/admin/health` (admin only)

Plus riche, utilisé par la page `/admin` :

```
{
  "qdrant": { "ok": true, "collections": 14, "stats": {...} },
  "tei": { "ok": true, "model": "BAAI/bge-m3" },
  "reranker": { "ok": true },
  "claude_ssh": { "ok": true, "latency_ms": 5234 },
  "smtp": { "ok": true, "configured": true },
  "stats": {
    "totalGames": 23,
    "totalQuestions": 542,
    "totalUsers": 3
  }
}
```

Si tu vois Qdrant/TEI/Reranker `ok: false`, le RAG est cassé. Investigation :

- `docker logs <service>` côté Unraid
- Tester le service directement : `curl http://192.168.10.100:8099/health` (TEI)

Ajouter un monitoring externe

Suggestions (non implémentées) :

Uptime Kuma

- Le plus simple. Container Unraid à part.
- Monitor : HTTP GET `https://rules.thymon.fr/api/health` toutes les 60s
- Notifications : ntfy / email / Telegram
- Permet de monitorer aussi NPM, Qdrant, TEI, etc. depuis un seul endroit.

Prometheus + Grafana

- Plus poussé. Hono peut exposer des métriques `/metrics` (lib `@hono/prometheus-metrics` ou équivalent).

- Métriques utiles : latence RAG p50/p99, count erreurs Claude, queue ingestion, etc.
- Container Prometheus + Grafana sur Unraid.

Sentry

- Erreurs côté backend + frontend
- DSN à mettre dans `SENTRY_DSN` env var
- Wrapper `src/lib/logger.ts` pour pousser sur `level === 'error'`

Alerting

Aucun alerting actuellement. Si l'app meurt à 3h du matin, tu le vois quand tu poses ta première question le lendemain.

Pour ajouter du basique :

- Uptime Kuma + notif ntfy / Telegram
- ou cron qui ping `/api/health` toutes les 5 min et envoie un mail si fail (script externe)

Métriques applicatives à logger

Si tu veux instrumenter sans monitoring formel :

- Logger `info` les timings RAG (`[rag] question_id=X total=4521ms hyde=2103ms retrieve=854ms rerank=121ms llm=1443ms`)
- Logger `info` les ingestion durées et tailles (`[ingest] game=X chunks=234 ocr_pages=12 duration_s=380`)
- Logger `warn` les fallbacks (HyDE timeout, retrieval vide, reranker down…)

Ensuite tu peux faire `grep + jq` sur `/app/data/logs/server.log` pour des stats ad hoc.

Troubleshooting

Troubleshooting

> *Dernière mise à jour : 2026-05-10*

Pièges connus déjà rencontrés, à actualiser au fil des incidents.

Build / CI

Le job `test` échoue avec une erreur `better-sqlite3`

- **Cause** : tu as changé l'image de base CI vers Alpine. Pas de prebuild musl pour `better-sqlite3`.
- **Fix** : remettre `node:22-bookworm-slim` (Debian Slim glibc).

`vue-tsc --noEmit` passe en local mais foire en CI

- **Cause** : la CI fait `vue-tsc --build` (compile full), pas `--noEmit` (type-check seul).
- **Fix** : tester avec `npm run build` (qui appelle `vue-tsc --build`) avant de push.

Backend runtime

`Boolean("false") === true` — env var qui ne fait pas ce qu'on attend

- **Symptôme** : `VISION_ENABLED=false` est interprété comme `true`.
- **Cause** : `z.coerce.boolean()` en JS — toute chaîne non vide est truthy.
- **Fix** : utiliser le helper `envBool()` défini en haut de `src/config.ts`. Comprend `true/false/1/0/yes/no/on/off`.

`process.env.X` retourne undefined en prod

- **Cause** : la var n'a pas été ajoutée dans `unraid/boardgame-referee.xml`, donc l'admin Unraid ne l'a pas setée dans l'UI.
- **Fix** : règle des 3 fichiers — `src/config.ts` + `.env.example` + `unraid/boardgame-referee.xml`. Sinon la var est silencieusement absente.

Une regex `\b` ne match pas un nom finissant par `é`

- **Cause** : `\b` sans flag `u` ne reconnaît que les boundaries `[A-Za-z0-9_]`.
- **Fix** : lookahead Unicode-aware avec flag `u` : `(?=\\s|$|[^\\p{L}\\p{N}_])`. Piège historiquement déclenché sur les mentions `@` Lorcana.

Claude répond "je ne vois aucune image"

- **Cause** : `claude-ssh.ts` lance le CLI avec `--system-prompt` au lieu de `--append-system-prompt`. Le contrat d'utilisation des outils est écrasé.
- **Fix** : toujours `--append-system-prompt`.

`ClaudeQuotaError` non détectée → "answer empty" générique

- **Cause** : le wording du message de quota a changé côté Anthropic.
- **Fix** : étendre `QUOTA_MARKERS` dans `claude-quota.ts`. Cf. `pipeline-rag/pause-quota-claude.md`.

Retrieval / RAG

Citation `[BASE p.4 : "..."]` ouvre la mauvaise page

- **Cause** : plusieurs livrets indexés partagent la page 4 (base + extension). La regex de citation ne capturait pas le nom du livret.
- **Fix** : cf. ARCHITECTURE.md § "Citations cliquables — format nom du livret". Le format est désormais `[<nom du livret>, p.X : "..."]` avec résolution `nameToId` côté frontend.

Image PNG envoyée à Claude mais c'est la page 4 du mauvais livret

- **Cause** : groupage par `page` au lieu de `(livret, page)` dans `answer.ts`.
- **Fix** : cf. `pipeline-rag/vision-inline.md` — granularité composite obligatoire.

Claude cite un livret en `[EXT p.4]` au lieu de `[Heavy Rain, p.4]`

- **Cause** : ancien format de citation pré-2026-04-11.
- **Fix** : c'est rétrocompatible côté frontend (regex avec fallback). Mais à terme migrer le SYSTEM_PROMPT pour imposer le nouveau format.

Reranker retourne ~0 sur du contenu technique abstrait → enterre le bon chunk

- **Cause** : le reranker bge-v2-m3 est peu confiant sur du jargon technique.
- **Fix** : c'est précisément la motivation du blending position-aware (RAG Fusion v2). Cf. ADR-005. Si tu vois encore des cas, considérer ajuster les ratios (`75/25` → `85/15` sur top-3).

Frontend

`display: flex` dans une CSS scoped écrase le `hidden lg:flex` Tailwind

- **Cause** : la spécificité CSS scoped batt utilities.
- **Fix** : préférer Tailwind utilities partout. Si tu dois absolument scoped, utiliser `!important` ou `@apply`.

Card autocomplete `@` ne matche pas un nom japonais / accentué

- **Cause** : `\b` regex Unicode-incompatible (cf. backend ci-dessus, même piège).
- **Fix** : `useArbiterMarkdown.ts:74-81` — lookahead `(?=\\s|$|[^\\p{L}\\p{N}_])` avec flag `u`.

CardZoomModal ne s'ouvre pas après reload

- **Cause** : `cardLookup` n'a pas été reconstruit depuis les messages persistés en localStorage.
- **Fix** : `usePlaySession` appelle `rebuildFromMessages()` on mount. Vérifier que ce path n'a pas été cassé.

SSE casse à 60s pile et UI affiche network error

- **Cause** : NPM `proxy_read_timeout 60s` + heartbeat backend pas émis.
- **Fix** : vérifier que `setInterval` heartbeat 8s est bien armé dans le service streamSSE. Pattern dans `routes/ask.ts`.

Infra

TEI répond `000` (curl no response)

- **Cause** : port custom Unraid (8099 au lieu de 80, 8990 au lieu de 8090).

- **Fix** : suspecter d'abord le port. Cf. mémoire `reference_unraid_ports.md`.

Container `oracle` SSH timeout

- **Cause** : VM oracle down ou auth ed25519 cassée.
- **Fix** :
  - `ssh -i /app/ssh/id_ed25519 oracle@<host> "cd /home/oracle/work && claude -p" <<< "test"`
  - Si timeout : SSH server down, restart la VM
  - Si `Permission denied` : vérifier `authorized_keys` côté VM, vérifier que la clé privée n'a pas été régénérée

Wrapper oracle rejette toutes les commandes

- **Cause** : préfixe strict du wrapper modifié sans aligner les autres endroits.
- **Fix** : 3 endroits doivent matcher (`/home/oracle/bin/oracle-claude.sh` PREFIX, `CLAUDE_SSH_WORKDIR` env, `permissions.allow`/`additionalDirectories` du settings.json oracle). Cf. `securite/ssh-oracle.md`.

Logs spam "EADDRINUSE :::3000"

- **Cause** : container restart trop rapide, l'ancien process pas encore mort.
- **Fix** : `docker compose down && docker compose up -d` (au lieu de restart).

BDD

Migration Drizzle qui ne s'applique pas

- **Cause** : `drizzle-kit generate` n'a pas détecté le diff (rare). Ou la table `__drizzle_migrations__` est désynchronisée.
- **Fix** : vérifier le contenu de `migrations/` vs `src/schema.ts`. Si nécessaire, `npm run db:migrate` manuel pour forcer.

`database.db is locked`

- **Cause** : un autre process / container tient le fichier.

- **Fix** : `lsof /app/data/database.db` (depuis Unraid CLI) pour identifier. Souvent un ancien container qui n'est pas vraiment mort.