Mode deckbuilding

Mode deckbuilding

> *Dernière mise à jour : 2026-05-10*

Depuis 2026-04-24 (Axe 2 Phase 1), un 4e intent `'deckbuilding'` s'ajoute à `'rules' | 'synergy' | 'meta'`. Il déclenche un pipeline dédié pour produire des **decklists structurées exécutables** (60 cartes MTG, 40+12+3 Riftbound, 60+sideboard FAB) là où le RAG classique produisait de la prose d'analyse.

Flux (`src/services/rag/deckbuilding/`)

1. Classification (`classify.ts`)

Haiku retourne `'deckbuilding'` quand la question exige une **liste exhaustive avec un nombre de cartes précis** ("decklist complète", "40 + 12 runes", "avec sideboard").

À distinguer de `synergy` qui reste une discussion d'archétype en prose.

Ordre de priorité : `deckbuilding > meta > synergy > rules`.

2. Dispatch (`answer.ts`)

Early-return vers `askDeckbuilding()` juste après résolution de l'intent. Le retrieval standard est jeté, askDeckbuilding relance son propre retrieval avec `intent='synergy'` forcé.

3. Spec Haiku (`deckbuilding/spec.ts`)

Un appel Haiku produit un `DeckSpec` (Zod) structuré :

- format
- tailles (mainboard, sideboard, runes, battlefields, equipment)
- max copies (généralement 4 MTG, 3 Riftbound, etc.)
- rôles cibles (`creatures`, `spells`, `lands`, `equipment`...)
- anchor cards (cartes obligatoires)

Fallback defaults par jeu si Haiku renvoie un JSON invalide :

- MTG Standard : `60 + 15, max 4`
- Riftbound : `40 + 0 + 3 + 12r + 3bf`
- FAB CC : `60 + 0, max 3`

4. Anchor cards

Les fresh + sticky mentions (@-cartes de l'Axe 1) deviennent **obligatoires** dans la decklist. Le prompt Opus a une règle stricte « tu DOIS inclure ces cartes ». Log warning serveur si une anchor est absente de la réponse finale.

5. Retrieval synergy

`retrieveChunks()` est appelé avec `intent='synergy'` → déclenche l'expansion mécanique (`retrieve/synergy-expansion.ts`). `retrieveChunks` accepte `'deckbuilding'` mais le mappe vers `'synergy'` en interne (fallback safe).

6. Prompt Opus spécialisé (`prompt-deckbuilding.ts`)

`DECKBUILDING_SYSTEM_PROMPT` impose :

- total exact par section
- max copies non-basiques (basic lands FR/EN whitelistés)
- anchor cards obligatoires
- pool strict (uniquement cartes du contexte)
- format markdown par rôles : `### Créatures (24)` + `4× [[card:Nom]]`

7. userPrompt assemblage

Ordre des blocs :

1. `SPEC DU DECK` (JSON Haiku)
2. `CARTES OBLIGATOIRES (ANCHORS)` (anchors fresh + sticky)
3. `CARTES CITÉES PAR LE JOUEUR (ce tour)` (full)
4. `CARTES ÉVOQUÉES DANS LA CONVERSATION` (sticky abrégées)
5. `DECKLISTS DE TOURNOI` (mainboard + sideboard complets, jusqu'à 5)
6. `POOL DE CARTES ÉLIGIBLES` (organisé par rôle, cap 30 par rôle)
7. `CARTES CANDIDATES` (chunks `[CARD]` + `[META]`, pas `[BASE]`/`[EXT]`)

8. Génération Opus

`promptStream(DECKBUILDING_SYSTEM_PROMPT, userPrompt, 'opus')` streamé. Post-processing : `autoWrapCardMentions` + `buildWrapAliases` + enrichissement extraCards.

9. Pool exhaustif + decklists d'inspiration (Phase 2, 2026-04-24)

- `fetchEligibleCards()` (`deckbuilding/pool.ts`) scroll Qdrant avec filtre structurel :
  - MTG : `card_mtg_legal_formats` + post-filtre `card_mtg_color_identity ⊆ spec.colors`
  - FAB : `card_legal_heroes`
  - Riftbound : sans filtre Qdrant (collection mono-format) + post-filtre `card_domains ⊆ spec.domains`
- `fetchInspirationDecklists()` : hybrid search (dense+sparse) sur `meta_type='tournament_deck'` filtré par `meta_format` — jusqu'à 5 decklists similaires (infra : MTGTop8 pour MTG, RiftboundStats pour Riftbound, fabtcg.com pour FAB)
- `selectRoleCandidates()` : pré-filtre chaque rôle de la spec à **cap 30 par rôle**, priorité aux cartes déjà vues dans les decklists d'inspiration
- TM / Ark Nova : pas de champs structurels → retombent silencieusement sur Phase 1 (retrieval synergy seul)
- `scrollAllPoints()` accepte `opts.filter` et `opts.withVector` (défaut true — Phase 2 passe `false` pour économiser ~120 MB sur MTG 30k cartes)

10. Validation structurelle + retry auto (Phase 3, 2026-04-24)

- `parseDeckResponse()` : extrait les cartes par section (main / sideboard / runes / battlefields) via regex stricte avec `×/x/X` obligatoire — les lignes de prose ne matchent pas
- `validateDeck()` :
  - Total exact par section

    - Max copies non-basiques (basic lands whitelist FR/EN : Plains/Island/Swamp/Mountain/Forest/Wastes + Plaine/Île/Marais/Montagne/Forêt)
    - Anchor cards présentes (≥1 copie)
    - Cartes inconnues du catalogue → warning non bloquant
- Si validation échoue : `formatRetryPrompt()` construit nouveau userPrompt listant les issues + contraintes strictes
- Opus relancé jusqu'à **2 retries** (3 tentatives max)
- Pendant retry : token séparateur visible streamé pour transparence — `done.fullAnswer` écrase le content côté frontend, l'utilisateur final voit uniquement la version corrigée
- Si après 3 tentatives la decklist reste invalide : on garde la dernière + warning log serveur (pas d'échec total)

11. Frontend

Aucun changement. Le rendu markdown gère titres + listes + `Nx [[card:X]]` (texte standard). `CardZoomModal` fonctionne automatiquement.

Hors scope Phases 1/2/3

- Pas de composant `<DeckListView>`
- Pas d'export `.dec` ni bouton Copier
- Pas de badge de validation côté frontend (verdict logué serveur, pas exposé via `done`)
- Pas de support TM / Ark Nova (pas d'équivalent format/archétype dans leurs données)

Helpers exportés

`autoWrapCardMentions`, `buildWrapAliases`, `renderFullCard`, `renderShortCard` sont exportés depuis `answer.ts` pour réutilisation par `deckbuilding/ask.ts`.

Revision #1
Created 2026-05-10 15:20:14 UTC by thymon
Updated 2026-05-10 15:20:14 UTC by thymon