ADR-006 : Repository pattern (Phase 2)

ADR-006 : Repository pattern (Phase 2)

> *Date : 2026-Q1 (Phase 2 refactoring) — Statut : accepté*

Contexte

Au début du projet, les routes Hono importaient directement `db` et `drizzle-orm` :

```
// routes/games.ts
import { db } from '../db.js';
import { games } from '../schema.js';

app.get('/api/games', async (c) => {
  const allGames = await db.select().from(games).all();
  return c.json(allGames);
});
```

Au fil du temps, le même `db.select().from(games)` se retrouvait dans 20 fichiers (routes, services, scripts, crons). Conséquences :

- Quand on voulait ajouter un index sur une colonne, il fallait grep tous les `from(games)` pour comprendre l'impact
- Les requêtes se dupliquaient (3 endroits avec `where(eq(games.id, ...))`)

- Pas de naming métier — juste du Drizzle qui parlait techniquement

Décision

Repository pattern. Tout accès DB passe par `src/repositories/*.repo.ts`. Aucune autre couche n'importe `db`, `drizzle-orm` ou les tables du schéma.

```
// repositories/games.repo.ts
import { db } from '../db.js';
import { games } from '../schema.js';

export async function getById(id: string): Promise<Game | null> {
  return db.select().from(games).where(eq(games.id, id)).get() ?? null;
}

export async function listByParent(parentId: string): Promise<Game[]> {
  return db.select().from(games).where(eq(games.parentGameId, parentId)).all();
}

// + une trentaine de fonctions par repo
```

```
// routes/games.ts
import * as gamesRepo from '../repositories/games.repo.js';

app.get('/api/games/:id', async (c) => {
  const game = await gamesRepo.getById(c.req.param('id'));
  if (!game) return c.json({ error: 'not found' }, 404);
  return c.json(game);
});
```

Règle stricte :

- ❌ `db.select(...)` dans `routes/`, `services/`, `middleware/`, `cron/`, `scripts/`, `index.ts`
- ✅ `db.select(...)` uniquement dans `src/repositories/*.repo.ts`

Repos actuels

- `repositories/games.repo.ts` — listAll, getById, getByName, search, upsert, setIngestStatus, listByParent, etc.
- `repositories/questions.repo.ts` — CRUD questions + feedback
- `repositories/users.repo.ts` — auth + permissions

Conséquences

Bonnes

- **Lecture facilitée** : pour comprendre toutes les opérations sur `games`, j'ouvre `games.repo.ts` et c'est tout.
- **Refactor simple** : ajouter un index, changer un nom de colonne → un seul endroit à modifier (sauf si la modif change le contrat de retour, mais ça c'est attendu).
- **Naming métier** : `getByBggId(id)` parle, `db.select().from(games).where(eq(games.bggId, id)).get()` ne parle pas.
- **Tests** : mock du repo avec `vi.mock('../repositories/games.repo.js')` est trivial — pas besoin de mock toute la couche Drizzle.

Mauvaises

- **Indirection supplémentaire** : pour ajouter une nouvelle requête, il faut éditer le repo + le consommateur (au lieu d'un seul fichier).
- **Pas une vraie abstraction** : si on migre Drizzle → autre ORM, les repos restent à réécrire. Mais leur API reste stable, donc les consommateurs ne bougent pas.
- **Conventions à appliquer** : un nouveau dev pourrait être tenté de mettre `db.select()` directement dans une route. Documenter (cf. CONTRIBUTING.md) + grep régulier pour vérifier.

Vérification

```
# Doit retourner zéro résultat
grep -rn "from 'drizzle-orm'" src/ --include="*.ts" \
  | grep -v src/repositories/ \
  | grep -v src/db.ts \
  | grep -v src/schema.ts \
  | grep -v drizzle.config.ts
```

Alternative envisagée

- **Active Record style** (ex. Prisma) : modèles auto-générés, requêtes natives sur les modèles. Demande de changer d'ORM. Drizzle est bien, on garde + repo pattern manuel.
- **Service layer mais sans repo** : services qui contiennent leurs propres queries Drizzle. Mais le service mélange logique métier + accès DB, retombe dans le même problème (DB queries éparpillées).

Évolution future

Si on ajoute un cache (Redis) entre repo et consommateurs :

- Le repo reste l'API stable
- L'implémentation du repo lit/écrit Redis en surcouche
- Les consommateurs ne changent pas

C'est précisément le bénéfice de l'abstraction.

Revision #1
Created 2026-05-10 15:19:54 UTC by thymon
Updated 2026-05-10 15:19:54 UTC by thymon