# WardPass performance and Drizzle implementation

Date: 2026-10-30

## Scope and result

This change starts the Prisma-to-Drizzle migration without applying a database migration or changing the Neon schema. WardPass now uses Drizzle for Better Auth and the vault, item, and admin-session database paths. The existing tables, columns, indexes, foreign keys, cascades, and user data remain in place.

Drizzle is a reasonable fit for this small, explicit data layer. It does not make a database request inherently faster. The performance work here reduces duplicate client requests, limits the data returned to navigation components, combines the active-vault view into one database operation, and moves Argon2 work off the browser's main thread. A before/after authenticated browser measurement is still needed to prove the effect in the deployment environment.

## What changed

### Database and authentication

- Added `drizzle/schema.ts` for the six live PostgreSQL tables: `user`, `session`, `account`, `verification`, `vault`, and `vaultItem`. Its names, column types, nullability, indexes, foreign keys, and cascade behavior were compared with a fresh Neon

introspection. Timestamp columns use JavaScript `Date` values in application code.
- Added `drizzle/client.ts` using Drizzle's Neon HTTP driver and `DATABASE_URL`. Added `drizzle.config.ts` and `db:pull`, `db:generate`, and `db:migrate` package scripts; Drizzle Kit uses `DIRECT_URL` for schema introspection and migration tooling.
- Added `drizzle/migrations/0000_broad_spiral.sql` and its snapshot as an introspected baseline. The generated SQL is commented out. No DDL or data change was applied to Neon. `db:generate` now reports "No schema changes".
- Switched Better Auth from its Prisma adapter to `drizzleAdapter`, while preserving the existing table names and additional admin fields. The installed Better Auth version's types do not support the proposed `advanced.database.joins` option, so it was not enabled.
- Removed Prisma application packages and the old Prisma client, schema, config, and client singleton. Prisma strings remaining in `package-lock.json` are transitive optional peer metadata from Better Auth's ecosystem, not a WardPass Prisma runtime import.

## Authorization and sensitive data

- Added server-only session helpers that derive the user and admin role from Better Auth's current request session.
- Migrated vault and vault-item reads/writes to Drizzle. Queries now scope reads, updates, deletes, trash, and restore operations to the authenticated owner. Item creation validates its type and checks the server-side item quota; vault creation checks its server-side quota.
- Added an admin-session guard to direct admin session listing queries. Session tokens are omitted from regular session, admin session, and user-session-search responses. Revocation now accepts a session ID and resolves the underlying token only on the server.
- Reused the request session in the authenticated layout and prefetched the narrow vault summary there. Sidebar and vault selection consume the hydrated TanStack Query cache. Vault summaries contain only id, name, slug, icon, and icon color.

### Vault loading and browser work

- Replaced the vault page's active-items request plus separate trash request with `getVaultPageData`. It returns active item records and a trash count, using a filtered join and correlated count in one database round trip. It does not return trashed ciphertext to the active-vault screen.
- Removed forced `refetchOnMount` behavior from the session, vault, and relevant query hooks. Cached data remains fresh for two minutes; mutations invalidate the relevant cache keys.
- Moved Argon2id derivation to a module Web Worker where available. Memory, time, parallelism, salt, and key length parameters are unchanged. The existing synchronous fallback remains for environments without Worker support.
- Kept React Query Devtools development-only and dynamically loaded it.

## Performance evidence

The earlier local production-build profile is a baseline, not a before/after result. It measured public routes and a database connectivity query; it did not exercise an authenticated user flow.

| Measurement | Existing baseline |
| --- | --- |
| Warm Prisma/Neon `SELECT 1` median | about 57.6 ms |
| First `/` TTFB | 369 ms |
| Warm `/` TTFB samples | 19–52 ms |
| `/password-generator` TTFB samples | 18–39 ms |
| `/sign-in` TTFB samples | 7–29 ms |
| Public-route HTML size | about 33–60 KB |

| | |
|---|---|
| Referenced JavaScript | 21–22 chunks, about 1.01–1.09 MB raw script bodies |

The JavaScript figure is an approximate sum of raw chunk bodies from the local server; it is not compressed transfer size. The `SELECT 1` result measures the full Prisma-to-Neon path, including network and service latency. It does not isolate Prisma overhead, and this change has not claimed or demonstrated a Drizzle query-time speedup.

Authenticated route timings, LCP, compressed transfer size, browser main-thread time, and before/after database query counts remain unmeasured. The available localhost browser session did not allow a reliable authenticated profiling run in this task. Repeat that profile with the same production build and test account after browser access is available.

## Validation performed

- `npx tsc --noEmit` — passed.
- `npm run build` — passed, including the production compile, TypeScript phase, and static generation. The build needs network access to fetch the existing Geist, Inter, and JetBrains Mono Google Fonts; the restricted run failed at font download, and the network-enabled run passed.
- `npm run db:generate` — passed and reported no schema changes against the introspected snapshot.
- `npm run db:pull` — read-only Neon introspection succeeded: 6 tables, 60 columns, 8 indexes, and 4 foreign keys. It did not apply DDL.
- Targeted ESLint on the changed server, auth, Drizzle, and worker modules — passed.
- Full `npm run lint` — still fails on existing lint findings: 12 errors and 14 warnings across UI components and pages (including `any` types, effect state updates, an impure render call, and unused values). These should be fixed in a separate cleanup pass; no broad lint suppression was added.
- No staging database or test account was available for sign-up,

sign-in, email, Google OAuth, admin, or cross-user integration flows. Those checks remain required before deployment.

## Risks and remaining work

1. **Quota checks can race.** Vault and item counts are checked before their insert in separate Neon HTTP requests. This prevents ordinary client bypasses, but two concurrent creates could both observe a count below the limit. If strict concurrent enforcement is required, add a database-level quota strategy or a transaction-capable/pool-backed path and test it on a Neon branch.
2. **The migration has not been deployed.** Prisma and Drizzle were pointed at the same existing database during the transition, and no dual writes were introduced. Before deploying this working tree, validate the Drizzle adapter and every auth flow against an isolated Neon branch; the local codebase no longer contains Prisma as a fallback.
3. **Full browser profiling is pending.** Measure TTFB, LCP, compressed JavaScript transfer, long tasks/main-thread time, route transitions, database request timings, and query counts for both public and authenticated routes on the same build.
4. **Build depends on Google Fonts availability.** Next.js downloads the three configured font families during the build. A self-hosted font setup would make builds independent of Google Fonts network access; decide this separately because it changes asset management and potentially the resulting typography files.
5. **Quota wording in the product plan is inaccurate.** The three-vault and three-item rules are storage quotas, not request rate limits. The limits appear to be product policy and should be configurable or documented as such.

## Plan-document review

The supplied plan files are product/design context, not instructions that override the implementation request.

- The main plan describes the 3-vault/3-item cap as a “rate limit”; it is a quota. The implementation enforces the check server-side, with the concurrency caveat above.
- The design plan and developer guide call for sharp 0px corners and minimal shadows, while much of the current UI uses rounded corners and shadow utilities. Decide whether the guide or the implementation is the source of truth and align the UI or guide.
- The design language uses “military-grade” and claims adherence to security standards. Keep external security claims tied to a documented threat model and reviewed evidence; UI styling and algorithm choice alone do not establish a security certification.
- The typography plan deliberately uses Google Fonts. That adds an external build dependency, described above; self-hosting is a practical reliability improvement if builds must work without external network access.

## Suggested next steps

1. Create an isolated Neon branch and run sign-up, sign-in, verification, password reset, session revocation, Google sign-in, admin operations, vault/item CRUD, trash/restore, cross-user denial, and quota checks against it.
2. Decide how to make the quotas atomic under concurrent creates.
3. Profile the same authenticated production routes before and after, then compare browser, database, and query-count results.
4. Fix the full-project lint findings in focused changes and update the three project plan documents to match the shipped schema, quota behavior, and design direction.
5. Only then schedule production rollout, with a rollback path and a verified database backup. Prisma has already been removed from this working tree, so the pre-migration release is the available application rollback until a Drizzle-backed release is proven.