ADR-001: Consolidate Backends

ADR-001: Consolidate to Single Backend

Status: Accepted **Date:** 2026-02-12 **Deciders:** John (AI Director), Alem (CEO) **Category:** Architecture

---

Context

The Drop codebase contained two competing middleware implementations creating confusion about which was authoritative:

1. **`lib/middleware.ts`** -- A simple, monolithic file used by all 24 API routes. Provided cookie-based JWT auth (`requireAuth`, `requireMerchant`), basic rate limiting via SQLite-backed `rate_limits` table, IP extraction from `X-Forwarded-For`, and standardized JSON error responses (`jsonError`).
2. **`lib/middleware/` directory** -- A more robust, modular implementation with `auth-middleware.ts` (Bearer token auth, in-memory rate limiting with `X-RateLimit-*` headers), `error-handler.ts` (typed `AppError` class with predefined error constructors), and `validation.ts` (comprehensive input validation: phone, amount, IBAN, PIN, email, currency, sanitization). This directory was **completely unused** by any route.

Additionally, **FontelePay** (an earlier project iteration) resided at `src/fontelepay/` with its own `.git/`, `node_modules/`, and independent codebase, blurring project boundaries.

The `SOURCE-STATUS.md` analysis confirmed that Drop source code had never been properly committed to version control. A clean rebuild using build artifacts as specification was the recommended path.

```
graph LR
    subgraph before["Before (Dual Middleware)"]
        routes["24 API Routes"] --> mw1["lib/middleware.ts<br/>(cookie JWT, basic rate limit)"]
```

```
        unused["UNUSED"] --> mw2["lib/middleware/<br/>(Bearer JWT, typed errors, validation)"]
    end

    subgraph after["After (Consolidated)"]
        routes2["24 API Routes"] --> mw3["lib/middleware.ts<br/>(cookie JWT, persistent rate limit,<br/>CSRF, session revocation)"]
        routes2 --> val["lib/middleware/validation.ts<br/>(input validation, sanitization)"]
    end

    before -->|"ADR-001"| after
```

Decision

1. **Consolidate middleware** into a single implementation, incorporating the best parts of both:
   - Cookie-based JWT auth from `lib/middleware.ts` (matches current route expectations)
   - Typed errors and input validation from `lib/middleware/` directory
   - Persistent rate limiting via `rate_limits` SQLite table (replacing in-memory `Map`)
2. **Remove duplicate code** after consolidation to eliminate confusion about which implementation is authoritative.
3. **Separate FontelePay** from Drop's `src/` directory (see ADR-002).
4. **Remove dead code** including any orphaned backend variants and unused files.

Consequences

Positive

- Single source of truth for middleware behavior
- Eliminates developer confusion about which middleware to use
- Typed errors (`AppError`, `Errors.*`) improve debugging and error reporting
- Input validation (`validateName`, `sanitizeText`, `validateAmount`) applied consistently
- Clean separation between Drop and FontelePay projects
- Reduced codebase size and maintenance burden

Negative

- All 24 API routes needed import path updates during rebuild
- Risk of regression if middleware behavior changed subtly during consolidation
- FontelePay separation required updating any shared references

Risks

- **Behavioral drift:** Consolidated middleware may behave differently from original in edge cases. Mitigation: test suite covering auth, transactions, cards, and merchants.
- **Import breakage:** Route import paths change. Mitigation: rebuild approach writes routes fresh against consolidated middleware.

References

- ADR-002: Separate FontelePay -- Companion decision for FontelePay extraction
- Middleware Documentation -- Current middleware reference
- Security Architecture -- Security controls using consolidated middleware
- Original source: `comms/decisions/ADR-001-consolidate-backends.md`

Revision #4
Created 2026-02-21 05:58:56 UTC by John
Updated 2026-05-23 10:56:30 UTC by John