Backend Architecture Document

Backend Architecture Document

> **Project:** Drop **Version:** 0.1.0 **Date:** 2026-02-23 **Author:** Platform Architect (AI)
> **Status:** In Review **Reviewers:** Alem Bašić (CEO)

Document History

Version	Date	Author	Changes
0.1	2026-02-23	Platform Architect (AI)	Initial draft from source code analysis

---

1. Architecture Pattern

Pattern: Modular Monolith — Next.js App Router with co-located API routes

Pattern Considered	Pros	Cons	Decision
Monolith (Next.js all-in-one)	Simple deploy, single codebase, lowest latency	Scaling bottleneck, frontend/backend coupled	Selected
Modular Monolith	Module isolation within single deploy	Extra abstraction overhead for small team	Partially adopted (lib/ modules)
Microservices	Independent scaling per service	Operational complexity, too expensive for MVP	Rejected

Rationale: Drop is a two-person MVP (Alem + AI). The Next.js App Router pattern co-locates API routes (`app/api/`) with the frontend, enabling full-stack development in a single TypeScript codebase with zero additional runtime complexity. The `src/lib/` directory provides module isolation (db, middleware, services, features) without microservice overhead. App Runner handles scaling concerns at the infrastructure level.

Pass-Through Model (Critical Architecture Constraint): Drop NEVER holds customer money. All payments are pass-through via PSD2:

- **AISP** (Account Information): reads bank balance from user's real bank account
- **PISP** (Payment Initiation): initiates transfers directly from user's bank account
- `bank_accounts.balance` = last AISP-read value from external bank (cached for UI display, NOT a Drop-held balance)

---

2. Technology Stack

Layer	Technology	Version	Notes
Runtime	Node.js	22 (Alpine)	LTS, Dockerfile base image
Framework	Next.js (App Router)	16.1.6	Standalone output for Docker
Frontend	React	19.2.3	
Language	TypeScript	^5	Strict mode
Database (production)	PostgreSQL (via `pg`)	16 (RDS)	`pg ^8.18.0`
Database (MVP/staging)	SQLite (via `better-sqlite3`)	^12.6.2	Auto-detected when no `DATABASE_URL`
Auth	JWT (via `jose`)	^6.1.3	HS256, httpOnly cookie
Password hashing	bcryptjs	^3.0.3	Legacy — BankID replaces email/password
Identity (eID)	BankID OIDC	Norwegian eID	Mandatory for all users
KYC	Sumsub WebSDK + API	Production-ready	Only connected external service
Open Banking	TBD (Swan deprecated)	—	AISP/PISP provider selection pending
Styling	Tailwind CSS	^4	
UI Components	Radix UI	^1.4.3	Accessible, unstyled primitives
Icons	Lucide React	^0.563.0	
Theme	next-themes	^0.4.6	Dark/light mode
Toasts	Sonner	^2.0.7	
Testing (unit)	Vitest	^4.0.18	
Testing (E2E)	Playwright	^1.58.2	
Linting	ESLint	^9	

---

3. Application Structure

```
src/drop-app/
├── src/
│   ├── app/                    # Next.js App Router
│   │   ├── api/                # API route handlers (REST endpoints)
│   │   │   ├── auth/           # BankID OIDC + session management
│   │   │   │   ├── bankid/     # initiate + callback endpoints
│   │   │   │   ├── me/         # current user + bank accounts
│   │   │   │   ├── logout/     # session revocation
│   │   │   │   └── refresh/    # token refresh
│   │   │   ├── transactions/   # remittance, qr-payment, history, disclosure, receipt
│   │   │   ├── recipients/     # recipient CRUD
│   │   │   ├── rates/          # exchange rates (public)
│   │   │   ├── merchants/      # merchant registration + dashboard
│   │   │   ├── notifications/  # push notification management
│   │   │   ├── settings/       # user preferences
│   │   │   ├── consents/       # GDPR consent management
│   │   │   ├── complaints/     # Finansavtaleloven §3-53 complaints
│   │   │   ├── cards/          # [FUTURE] feature-flagged card management
│   │   │   ├── user/           # GDPR data export + account deletion
│   │   │   └── health/         # health check endpoint
│   │   └── (frontend pages)    # Next.js pages
│   └── lib/                    # Shared application library
│       ├── db.ts               # Database abstraction (PostgreSQL + SQLite)
│       ├── middleware.ts        # Auth, rate limiting, CSRF, session revocation
│       ├── middleware/          # Modular middleware library
│       │   ├── auth-middleware.ts  # Bearer token auth (mobile)
│       │   ├── error-handler.ts   # AppError class + error response formatting
│       │   └── validation.ts      # Input validation functions
│       ├── alerts.ts           # Slack alerting + error spike detection
│       ├── secrets.ts          # Pluggable secrets provider (env / Doppler / AWS SM)
│       ├── feature-flags.ts    # Environment-variable-based feature flags
│       ├── features.ts         # Feature tracking system (dev tool)
│       └── services/           # External service integrations
│           ├── index.ts        # Service initialization
│           ├── mock-sumsub.ts  # Sumsub KYC (production-ready)
│           ├── mock-swan.ts    # Swan Open Banking (DEPRECATED)
│           └── mock-stripe.ts  # Stripe Issuing (mock only, FUTURE)
```

```
├── tests/                    # Test suite
│   ├── setup.ts              # Vitest setup (NODE_ENV=test, in-memory DB)
│   ├── *.test.ts             # Unit + integration tests
│   └── e2e/                  # Playwright E2E tests
│       ├── user-flows.spec.ts
│       ├── full-flows.spec.ts
│       └── input-chaos.spec.ts
└── scripts/
    ├── backup.sh             # SQLite backup script
    └── qa-report.js          # QA metrics generator
```

---

4. Database Layer

4.1 Dual-Database Architecture

Drop auto-detects the database driver at startup:

- `DATABASE_URL` set → PostgreSQL (`pg` driver)
- `DATABASE_URL` not set → SQLite (`better-sqlite3`)

Source: `src/lib/db.ts`

4.2 Key Database Tables

Table	Purpose	Notes
`users`	User accounts, KYC status, BankID linkage	`kyc_status`: pending/approved/rejected; `national_id_hash`: SHA-256 of BankID pid
`sessions`	JWT session tracking + revocation	SHA-256 hash of JWT, `revoked` flag
`bank_accounts`	Linked bank accounts (AISP data)	`balance` = last AISP read (NOT Drop-held funds)
`transactions`	All payments (remittance + QR)	`type`: remittance/qr_payment; `status`: processing/completed/failed
`recipients`	Saved international recipients	Bank account masked in API responses
`merchants`	Merchant profiles, QR data	`org_number` unique (9 digits, Norwegian)

Table	Purpose	Notes
`notifications`	User notifications	Feature-flagged
`rate_limits`	IP-based rate limiting (persistent)	`key`: IP address, window-based counter
`audit_log`	Security + compliance audit trail	`action`, `resource_type`, `resource_id`, `details`
`aml_alerts`	AML/financial crime alerts	`status`: open/closed/filed
`str_reports`	Suspicious Transaction Reports	Filed with Finanstilsynet
`consents`	GDPR consent records	`consent_type`: terms/privacy/marketing/cookies_analytics/cookies_marketing
`data_access_requests`	GDPR export/erasure requests	`type`: export/erasure
`complaints`	User complaints (Finansavtaleloven §3-53)	15-business-day response SLA
`exchange_rates`	NOK → destination currency rates	Updated externally
`feature_flags`	Runtime feature flag overrides	Complement to env-var flags
`cards`	[FUTURE] Virtual/physical cards	Feature-flagged, all flags default false

4.3 Data Auto-Detection (db.ts)

```
// Auto-detects driver based on DATABASE_URL env var
const driver = process.env.DATABASE_URL ? 'pg' : 'sqlite';
```

---

5. Authentication Architecture

5.1 BankID OIDC Flow (Primary Auth)

Auth method: Norwegian BankID — mandatory for all users. Email/password auth deprecated (returns 410 Gone).

Token: JWT (HS256), stored in `drop_token` httpOnly cookie (web) or Authorization Bearer header (mobile).

Token lifetime: 24h (web), 7d (mobile)

BankID Web Flow:

1. `GET /api/auth/bankid` → generate state + nonce, set `bankid_state` cookie, return redirect URL
2. User authenticates with BankID at provider
3. `GET /api/auth/bankid/callback?code=&state=` → verify state, exchange code for tokens, verify JWKS signature, parse `pid`, hash pid → SHA-256, find/create user, issue JWT cookie

User creation on first BankID login:

- Parse pid (Norwegian national ID, 11 digits) from ID token
- Hash pid with SHA-256 → `national_id_hash` column
- KYC status automatically `approved` (BankID = verified identity)
- Password set to sentinel `'EIDONLY'` — no password login possible

Age verification: pid encodes date of birth — must be >= 18 years old.

5.2 Session Management

```
Login  → Create session record (SHA-256 of JWT) in sessions table
Request → requireAuth() checks: cookie present + JWT valid + session not revoked
Logout → revokeAllSessions(userId) — sets revoked=1 on all user sessions
```

5.3 CSRF Protection

- **Web:** State parameter in BankID OIDC flow (httpOnly cookie)
- **API:** Origin header validation in `requireAuth()` against allowed origins
- **Mobile:** N/A (Bearer token, no cookies)

---

6. Middleware Stack

Middleware	Function	Applied To
`requireAuth()`	CSRF check → cookie extraction → JWT verify → user lookup → session revocation check	All protected routes
`requireMerchant()`	`requireAuth()` + role check (`role === 'merchant'`)	Merchant-only routes
`rateLimit(ip, limit)`	Persistent IP-based counter via `rate_limits` DB table, 60s window	Auth endpoints (10/min), public rates (120/min)
`getClientIp()`	Extract IP from `x-forwarded-for`	All rate-limited routes
`jsonError()`	Standardized JSON error response	All routes

Middleware	Function	Applied To
`featureGate(flag)`	Returns 404 if feature flag disabled	Cards, spending limits, notifications
Input validation	`validateEmail`, `validatePhone`, `validateAmount`, `validateName`, `sanitizeText`, etc.	All mutation endpoints
Error handler	`AppError` class with predefined constructors	All routes via `createErrorResponse()`

---

7. API Design Principles

1. **Consistent response envelope:**
   - Success: `{ "data": { ... } }` or `{ "data": [...], "pagination": { ... } }`
   - Error: `{ "error": "code", "message": "...", "details": [...] }`
2. **No wallet model:** Drop never holds funds. `bank_accounts.balance` is AISP-read cache only.
3. **KYC gate:** Remittance requires `kyc_status === 'approved'` — enforced in route handler.
4. **Atomic transactions:** Balance deduction and transaction creation in a single DB transaction.
5. **Data masking:** Bank account numbers masked in responses (`*****5678`), card numbers PCI-masked.
6. **GDPR by design:** Data export, account deletion (soft delete), consent records all implemented.
7. **Compliance-first:** STR reports, AML alerts, audit log, complaint system (Finansavtaleloven §3-53), PITR retention (5 years per hvitvaskingsloven).

---

8. Security Architecture

Control	Implementation
Auth	BankID OIDC (Norwegian eID) — mandatory
Session tokens	httpOnly, secure, sameSite=strict JWT cookies
Rate limiting	Persistent DB-backed per-IP (10/min auth, 120/min public)
Input validation	Custom validators (no external dep) — email, phone, amount, IBAN, name (XSS-resistant)
SQL injection	Parameterized queries via `pg` / `better-sqlite3`
XSS	CSP headers (strict production — no unsafe-eval) + HTML sanitization in `sanitizeText()`
CSRF	Origin header validation + BankID state parameter

Control	Implementation
Secrets	AWS Secrets Manager / Fly.io secrets — never in code or .env
Error masking	`createErrorResponse()` masks internal errors in production
Password hashing	bcryptjs (legacy users)
Card data	PCI-masked (never expose full card number or CVV)
Audit trail	`audit_log` table — all sensitive actions logged
AML	`aml_alerts` + `str_reports` tables — compliance framework

9. External Service Integrations

Service	Status	Purpose
Sumsub	PRODUCTION (only connected external service)	KYC/identity verification — WebSDK + webhook
BankID OIDC	PRODUCTION	Norwegian eID authentication
Open Banking (AISP/PISP)	TBD — provider selection pending	Bank balance read + payment initiation
Swan Open Banking	DEPRECATED	Was planned, no longer selected
Stripe Issuing	MOCK (future)	Card issuance — no SDK, no API keys
Slack	PRODUCTION	Operational alerting via webhook
BetterStack	PRODUCTION	External uptime monitoring

10. Related Documents

- API Reference
- Middleware Design
- Service Design
- External Services Integration
- Deployment Architecture
- Authentication Source

Approval

Role	Name	Date	Signature
Author	Platform Architect (AI)	2026-02-23	
Reviewer			
Approver	Alem Bašić		

Revision #5
Created 2026-02-23 12:03:10 UTC by John
Updated 2026-05-31 20:02:58 UTC by John