Architecture Overview

Drop Architecture Documentation

Project: Drop -- Fintech Payment App (Remittance + QR Payments) **Model:** PSD2 Pass-through (AISP + PISP) -- Drop never holds customer funds **Last updated:** 2026-02-21

Reading Guides by Audience

Executive Summary

Start here for a high-level understanding of Drop's architecture and regulatory position.

1. System Context (C4 Level 1) -- Drop and all external actors (banks, BankID, regulators)
2. Security Architecture -- Security controls and threat model
3. ADR-003: PSD2 Pass-through Model -- Why Drop never holds money

Architect Track

For a complete understanding of system design and key decisions.

1. **High-Level Design (C4)**
   - System Context (C4 Level 1) -- External actors and trust boundaries
   - Container Diagram (C4 Level 2) -- Internal containers and data flow
   - Component Overview (C4 Level 3) -- Module-level breakdown
2. **Cross-Cutting Concerns**
   - Security Architecture -- Auth, encryption, input validation
   - Data Architecture -- Data flow, storage, and processing
   - Deployment Architecture -- AWS App Runner, Cloudflare, CI/CD
3. **Architecture Decision Records**

- ADR Index -- All 12 ADRs with template

4. **Integration Specifications**
   - Open Banking AISP/PISP -- PSD2 banking integration
   - BankID OIDC Integration -- Norwegian eID authentication
   - Sumsub KYC Integration -- Identity verification
   - Payment Processing -- Remittance and QR payment flows
   - Sentry Observability -- Error tracking and monitoring

Developer Track

For implementing features and understanding code flows.

1. **Low-Level Designs (User Flows)**
   - Login Authentication -- BankID login flow (frontend)
   - Login Authentication Backend -- Auth backend implementation
   - Registration & Onboarding -- New user registration
   - Bank Account Linking -- AISP consent and account linking
   - Remittance -- International money transfer flow
   - QR Payment -- Merchant QR payment flow
   - Transaction History -- Transaction listing and filtering
   - Merchant Onboarding -- Business registration flow
   - Profile & Settings -- User preferences management
   - Notifications -- Push notification flow
2. **Database**
   - Database Design -- Schema rationale and entity relationships
   - Migration Strategy -- SQLite to PostgreSQL migration
   - Data Lifecycle -- Retention, archival, and deletion policies
   - Audit Architecture -- Compliance audit trail design
   - Indexing Strategy -- Database performance optimization
3. **Key ADRs for Developers**
   - ADR-004: JWT httpOnly Cookies -- Token storage security
   - ADR-006: SQLite to PostgreSQL -- **SUPERSEDED** by ADR-014
   - ADR-008: Hono API Framework -- Mobile API choice
   - ADR-009: Feature Flag System -- Runtime feature toggles
   - ADR-010: Dual Database Driver -- **SUPERSEDED** by ADR-014

- ADR-014: PostgreSQL-Only Architecture -- PostgreSQL 16 + Drizzle ORM (all environments)

Compliance Track

For regulatory compliance review and audit preparation.

1. Security Architecture -- Authentication, authorization, data protection
2. KYC/AML Flow -- Customer due diligence, transaction monitoring, STR filing
3. Data Lifecycle -- GDPR retention policies, right to erasure
4. Audit Architecture -- Audit trail for regulatory reporting
5. Registration & Onboarding -- Consent collection, age verification
6. ADR-003: PSD2 Pass-through -- Regulatory model decision
7. ADR-007: BankID OIDC Auth -- SCA compliance

Document Index

High-Level Design (HLD)

Document	Description	Key Diagrams
System Context (C4 L1)	Drop and all external actors, trust boundaries, compliance zones	C4 context diagram, trust boundary diagram
Container Diagram (C4 L2)	Internal containers: Next.js BFF, Hono API, databases, edge	C4 container diagram, data flow diagram
Component Overview (C4 L3)	Module-level breakdown: auth, transactions, merchants, compliance	Component diagrams per module
Security Architecture	Auth, encryption, input validation, rate limiting, headers	Security layer diagram, threat model
Data Architecture	Data flow, storage tiers, processing pipeline	Data flow diagram, storage topology
Deployment Architecture	AWS App Runner, Cloudflare, CI/CD pipeline, environments	Deployment topology, CI/CD pipeline

Architecture Decision Records (ADR)

ADR	Title	Category	Status

ADR-001	Consolidate to Single Backend	Architecture	Accepted
ADR-002	Separate FontelePay from Drop	Architecture	Accepted
ADR-003	PSD2 Pass-through Model (No Wallet)	Architecture	Accepted
ADR-004	JWT in httpOnly Cookies	Security	Accepted
ADR-005	Monolith-First Architecture	Architecture	Accepted
ADR-006	SQLite Dev, PostgreSQL Prod	Database	Superseded by ADR-014
ADR-007	BankID as Sole Auth Provider	Security	Accepted
ADR-008	Hono v4 for Mobile API	Backend	Accepted
ADR-009	Custom Feature Flag System	Backend	Accepted
ADR-010	Dual Database Driver Abstraction	Database	Superseded by ADR-014
ADR-011	Expo SDK 54 for Mobile	Mobile	Accepted
ADR-012	AWS App Runner Deployment	Infrastructure	Accepted
ADR-014	PostgreSQL-Only Architecture (Drizzle ORM)	Database	Accepted

Integration Specifications

Document	Description	External System
Open Banking AISP/PISP	PSD2 account access and payment initiation	Nordic banks (DNB, SpareBank1, Nordea)
BankID OIDC Integration	Norwegian eID authentication and SCA	BankID Norge
Sumsub KYC Integration	Document verification and screening	Sumsub
Payment Processing	Remittance and QR payment orchestration	Banks, SEPA, SWIFT
Sentry Observability	Error tracking and performance monitoring	Sentry

Low-Level Design (LLD) -- User Flows

Document	Description	Key Endpoints
Login Authentication	BankID OIDC login (frontend)	`/api/auth/bankid`, `/api/auth/bankid/callback`
Login Auth Backend	Auth backend: JWT, sessions, revocation	`lib/auth.ts`, `middleware/auth.ts`, `middleware/rate-limit.ts`
Registration & Onboarding	New user registration, consent, bank linking	`/api/auth/bankid`, `/api/consents`
Bank Account Linking	AISP consent and account sync	Open Banking AISP APIs
Remittance	International money transfer	`POST /api/transactions/remittance`
QR Payment	Merchant QR scan and pay	`POST /api/transactions/qr-payment`
Transaction History	Transaction listing with filters	`GET /api/transactions`
KYC/AML	KYC verification and AML monitoring	Sumsub webhooks, `aml_alerts`, `str_reports`
Merchant Onboarding	Business registration and QR setup	`POST /api/merchants/register`
Profile & Settings	User preferences and account management	`GET/PATCH /api/settings`
Notifications	Push notification delivery	`GET/PATCH /api/notifications`

Database Architecture

Document	Description
Database Design	Schema rationale, entity relationships, 19-table overview
Migration Strategy	Historical SQLite to PostgreSQL migration record (completed; see ADR-014)
Data Lifecycle	GDPR retention, AML record keeping, deletion policies
Audit Architecture	`audit_log` table design and compliance trail
Indexing Strategy	Performance indexes for queries, compliance, and reporting

---

Related Documentation

Document	Location	Description
Main Architecture Document	`project/architecture/`	Original architecture overview (v1.1)

Document	Location	Description
API Reference	`docs/backend/`	All 24+ API endpoints
Database Schema	`docs/backend/`	Full table definitions
Authentication System	`docs/backend/`	BankID OIDC implementation
Middleware	`docs/backend/`	Auth, rate limiting, validation
Feature Flags	`docs/backend/`	Runtime feature toggles
Security Architecture	`docs/security/`	Security controls reference
Compliance Status	`docs/security/`	Regulatory compliance tracking
Roadmap	Root	Product roadmap and phases

Architecture Principles

1. **Pass-through model:** Drop never holds customer funds. All payments initiated via PISP from user's bank.
2. **Security-first:** httpOnly cookies, parameterized SQL, BankID SCA, rate limiting on all public endpoints.
3. **Compliance by design:** Audit logging, consent tracking, AML monitoring built into the data model.
4. **Monolith-first:** Simple deployment, extract services only when scaling demands it.
5. **PostgreSQL-only:** PostgreSQL 16 in all environments (dev, CI, staging, production) via Drizzle ORM. See ADR-014.
6. **Feature-flagged:** Unreleased features (cards) safely gated; critical features can be killed instantly.

Revision #10
Created 2026-02-21 05:58:46 UTC by John
Updated 2026-05-23 10:51:28 UTC by John