High-Level Design (HLD)

System-level architecture diagrams and design

- System Context
- Container Diagram
- Component Overview
- Deployment Architecture
- Security Architecture
- Data Architecture

System Context

System Context Diagram (C4 Level 1)

Document: HLD-001 **Status:** Approved **Last updated:** 2026-02-21 **Author:** Standards Architect
Applies to: Drop v1.0 (PSD2 pass-through model)

Overview

This document describes the C4 Level 1 system context for Drop, showing Drop as the central system and all external actors, systems, and regulatory bodies it interacts with. Drop operates as a PSD2 pass-through payment application -- it **never holds customer funds**. User money remains in their bank account at all times.

System Context Diagram

```
graph TB
    subgraph actors["External Actors"]
        sender["Sender<br/>(Norwegian Resident, 18+)<br/>Sends money abroad via PISP"]
        receiver["Receiver<br/>(30+ countries)<br/>Receives remittance"]
        merchant["Merchant<br/>(Norwegian Business)<br/>Accepts QR payments"]
    end

    subgraph drop_system["Drop Payment System"]
        drop["Drop<br/>Next.js 15 + Hono v4<br/>PSD2 Pass-through App<br/>(AISP + PISP)"]
    end

    subgraph banking["Banking & Open Banking"]
        bankid["BankID Norway<br/>OIDC Identity Provider<br/>Strong Customer Authentication"]
        nordic_banks["Nordic Banks<br/>(DNB, SpareBank1, Nordea)<br/>Open Banking
APIs<br/>AISP: Read balance<br/>PISP: Initiate payment"]
```

```
        payment_rails["Payment Rails<br/>SEPA (EEA)<br/>SWIFT (non-EEA)<br/>Remittance
corridors"]
    end

    subgraph compliance["Compliance & KYC"]
        sumsub["Sumsub<br/>KYC/AML Provider<br/>Document verification<br/>PEP/sanctions
screening"]
        finanstilsynet["Finanstilsynet<br/>Norwegian FSA<br/>PISP/AISP
registration<br/>Regulatory oversight"]
        okokrim["Okokrim / EFE<br/>Financial Intelligence Unit<br/>STR/SAR filing"]
    end

    subgraph infrastructure["Infrastructure"]
        aws["AWS App Runner<br/>Container hosting<br/>Auto-scaling"]
        cloudflare["Cloudflare<br/>CDN, DDoS protection<br/>DNS, TLS termination"]
        sentry["Sentry<br/>Error tracking<br/>Performance monitoring"]
    end

    %% Actor interactions
    sender -->|"BankID login\nView balance (AISP)\nSend money (PISP)\nQR payments"| drop
    receiver -.->|"Receives funds\n(via bank transfer)"| payment_rails
    merchant -->|"Register business\nView dashboard\nGenerate QR code"| drop

    %% Banking integrations
    drop -->|"OIDC authorize\nID token verification\nAge/identity check"| bankid
    drop -->|"AISP: GET /accounts\nAISP: GET /balances\nPISP: POST /payments"| nordic_banks
    drop -->|"PISP payment routing\nSEPA for EEA\nSWIFT for non-EEA"| payment_rails

    %% Compliance integrations
    drop -->|"Applicant creation\nDocument upload\nWebhook results"| sumsub
    drop -.->|"License registration\nRegulatory reporting\nCompliance audits"| finanstilsynet
    drop -.->|"STR filing\n(hvitvaskingsloven)"| okokrim

    %% Infrastructure
    drop -->|"Deploy containers\nAuto-scale"| aws
    drop -->|"DNS routing\nTLS, WAF\nDDoS protection"| cloudflare
    drop -->|"Error events\nPerformance traces"| sentry

    %% Bank to payment rails
    nordic_banks -->|"Execute transfers"| payment_rails
```

```
    classDef actorStyle fill:#E3F2FD,stroke:#1565C0,stroke-width:2px,color:#0D47A1
    classDef systemStyle fill:#0B6E35,stroke:#064E25,stroke-width:3px,color:#FFFFFF
    classDef bankingStyle fill:#FFF3E0,stroke:#E65100,stroke-width:2px,color:#BF360C
    classDef complianceStyle fill:#FCE4EC,stroke:#C62828,stroke-width:2px,color:#B71C1C
    classDef infraStyle fill:#F3E5F5,stroke:#6A1B9A,stroke-width:2px,color:#4A148C

    class sender,receiver,merchant actorStyle
    class drop systemStyle
    class bankid,nordic_banks,payment_rails bankingStyle
    class sumsub,finanstilsynet,okokrim complianceStyle
    class aws,cloudflare,sentry infraStyle
```

Trust Boundaries

```
graph TB
    subgraph tb_user["TRUST BOUNDARY: User Device (Untrusted)"]
        browser["Web Browser<br/>(Next.js SSR + CSR)"]
        mobile["Mobile App<br/>(Expo SDK 54)"]
    end

    subgraph tb_drop["TRUST BOUNDARY: Drop Application (Controlled)"]
        subgraph dmz["DMZ — Edge"]
            cf["Cloudflare<br/>WAF + CDN + DDoS"]
        end
        subgraph app["Application Layer"]
            nextjs["Next.js BFF<br/>Web auth, SSR"]
            hono["Hono API<br/>Mobile auth, REST"]
        end
        subgraph data["Data Layer"]
            pg["PostgreSQL<br/>(production)"]
            sqlite["SQLite<br/>(development)"]
        end
    end

    subgraph tb_banking["TRUST BOUNDARY: Banking Partners (External Trusted)"]
        bankid_tb["BankID OIDC"]
        openbanking["Open Banking APIs"]
```

```
    end

    subgraph tb_compliance["TRUST BOUNDARY: Compliance Partners (External Trusted)"]
        sumsub_tb["Sumsub KYC"]
    end

    subgraph tb_regulator["TRUST BOUNDARY: Regulatory (Government)"]
        fsa["Finanstilsynet"]
        efe["Okokrim / EFE"]
    end

    browser --> cf
    mobile --> cf
    cf --> nextjs
    cf --> hono
    nextjs --> pg
    nextjs --> sqlite
    hono --> pg
    hono --> sqlite
    nextjs --> bankid_tb
    hono --> bankid_tb
    nextjs --> openbanking
    hono --> openbanking
    nextjs --> sumsub_tb
    hono --> sumsub_tb
    nextjs -.-> fsa
    nextjs -.-> efe

    classDef untrusted fill:#FFCDD2,stroke:#C62828,stroke-width:2px
    classDef controlled fill:#C8E6C9,stroke:#2E7D32,stroke-width:2px
    classDef external fill:#FFF9C4,stroke:#F9A825,stroke-width:2px
    classDef regulator fill:#E1BEE7,stroke:#6A1B9A,stroke-width:2px

    class browser,mobile untrusted
    class cf,nextjs,hono,pg,sqlite controlled
    class bankid_tb,openbanking,sumsub_tb external
    class fsa,efe regulator
```

External Actors

End Users

Actor	Description	Authentication	Data Exchanged
Sender	Norwegian resident (18+) who sends money abroad or pays merchants via QR	BankID OIDC (SCA)	Personal data, bank account info (AISP), payment instructions (PISP)
Receiver	Person in 30+ countries who receives remittance	None (indirect)	Receives bank transfer via payment rails
Merchant	Norwegian business accepting QR payments	BankID OIDC + merchant registration	Business details, org number, transaction data, payout info

Banking & Payment Systems

System	Protocol	Data Flow	Trust Level
BankID Norway	OIDC 2.0 (authorize, token, JWKS endpoints)	ID tokens with `pid` (national ID), name, DOB	High -- Norwegian government-backed eID
Nordic Banks (DNB, SpareBank1, Nordea)	PSD2 Open Banking REST APIs	AISP: account list, balances, transactions; PISP: payment initiation, status	High -- regulated financial institutions
SEPA (Single Euro Payments Area)	SEPA Credit Transfer (SCT)	EEA remittance transfers (1-2 business days)	High -- ECB-regulated
SWIFT	SWIFT gpi	Non-EEA remittance transfers (2-4 business days)	High -- SWIFT-regulated

Compliance & Regulatory

System	Integration	Data Flow	Cadence
Sumsub	REST API + Webhooks	Applicant data, document images, verification results, PEP/sanctions matches	On registration + ongoing monitoring
Finanstilsynet	Regulatory portal	License applications, compliance reports, incident notifications	Quarterly + ad hoc
Okokrim / EFE	AltInn reporting	STR/SAR filings per hvitvaskingsloven	As triggered by AML alerts

Infrastructure

System	Role	Protocol	Data Flow
AWS App Runner	Container hosting and auto-scaling	HTTPS, Docker	Application containers, environment variables, logs
Cloudflare	Edge security and CDN	DNS, HTTPS, WebSocket	HTTP traffic, TLS termination, DDoS filtering, WAF rules
Sentry	Error tracking and APM	HTTPS (SDK)	Error events, performance traces, session replays

---

Compliance Zone Mapping

PSD2 (Betalingstjenesteloven)

Requirement	Drop Component	External System	Status
Strong Customer Authentication (SCA)	Auth flow (`/api/auth/bankid/`)	BankID OIDC	Implemented
Dynamic linking (amount + payee tied to auth)	Payment confirmation screen	BankID SCA challenge	Phase 2
AISP consent and access	Bank account linking flow	Nordic bank Open Banking APIs	Phase 2
PISP payment initiation	Remittance + QR payment flows	Nordic bank Open Banking APIs	Phase 2
Framework agreement (vilkar)	`landing/pages/vilkar.html`	--	Draft exists
Pre-transaction fee disclosure	`POST /api/transactions/disclosure`	--	Implemented

GDPR (Personopplysningsloven)

Requirement	Drop Component	Implementation
Lawful basis for processing	`consents` table	Consent tracking with IP + timestamp
Right to access (Art. 15)	`GET /api/user/data-export`	Full data export in JSON
Right to erasure (Art. 17)	`DELETE /api/user/account`	Soft delete, 5yr AML retention
Data minimization (Art. 5)	Schema design	Only necessary fields stored

Requirement	Drop Component	Implementation
Data portability (Art. 20)	`GET /api/user/data-export`	Machine-readable JSON export
Processing register (Art. 30)	`data_access_requests` table	Tracks all DSAR requests
DPIA (Art. 35)	`legal/dpia-vurdering.md`	Draft completed

AML / KYC (Hvitvaskingsloven)

Requirement	Drop Component	External System
Customer Due Diligence (CDD)	User registration + KYC flow	Sumsub (document verification)
Enhanced Due Diligence (EDD)	`screening_results` table	Sumsub (PEP/sanctions screening)
Transaction monitoring	`aml_alerts` table	Internal rules engine
Suspicious Transaction Reporting	`str_reports` table	Okokrim / EFE via AltInn
Record keeping (5 years)	All compliance tables	PostgreSQL with retention policies
Risk assessment	`users.risk_level` field	Sumsub risk scoring

DORA (Digital Operational Resilience Act)

Requirement	Drop Component	Implementation
ICT risk management	`legal/ikt-sikkerhetspolicy.md`	Policy drafted
Incident reporting	`legal/hendelseshaandtering.md`	Incident handling procedure
Resilience testing	Planned penetration test	Phase 3
Third-party risk management	`legal/utkontraktering-policy.md`	Outsourcing policy drafted
Business continuity	`legal/beredskapsplan.md`	BCP drafted

Data Flow Summary

Flow	Source	Destination	Data	Protocol	Encryption
User authentication	Browser/Mobile	BankID	OIDC auth request, state, nonce	HTTPS	TLS 1.3
Identity verification	Drop	BankID	Authorization code exchange	HTTPS	TLS 1.3
Balance read (AISP)	Drop	Nordic Bank	Account ID, consent token	PSD2 Open Banking API	TLS 1.3 + OAuth2

Flow	Source	Destination	Data	Protocol	Encryption
Payment initiation (PISP)	Drop	Nordic Bank	Amount, recipient, consent	PSD2 Open Banking API	TLS 1.3 + OAuth2 + SCA
KYC verification	Drop	Sumsub	Applicant data, documents	REST API + Webhooks	TLS 1.3 + API key
STR filing	Drop	Okokrim	Suspicious transaction report	AltInn portal	TLS 1.3 + certificate
Error tracking	Drop	Sentry	Error events, stack traces	HTTPS SDK	TLS 1.3 + DSN token
Web traffic	User	Cloudflare -> Drop	HTTP requests/responses	HTTPS	TLS 1.3 (edge + origin)

Cross-References

- Container Diagram (C4 Level 2) -- Internal container breakdown
- Security Architecture -- Detailed security controls
- BankID OIDC Integration -- Authentication integration spec
- Open Banking AISP/PISP -- Banking integration spec
- Sumsub KYC Integration -- KYC provider integration
- ADR-003: PSD2 Pass-through Model -- Foundational architecture decision
- ADR-007: BankID OIDC Auth -- Authentication provider decision

Container Diagram

C4 Level 2 — Container Diagram

> Drop fintech platform container architecture showing all runtime containers, their responsibilities, communication patterns, and the middleware chain that governs every API request.

Container Diagram

```
C4Container
  title Drop — Container Diagram (C4 Level 2)

  Person(user, "End User", "Norwegian resident 18+, authenticated via BankID")
  Person(merchant, "Merchant", "Business owner receiving QR payments")

  System_Boundary(drop, "Drop Platform") {
    Container(web, "drop-web", "Next.js 15, React 19, Tailwind v4", "Server-side rendered web application. Handles login redirect, dashboard, send money, QR scan, bank accounts, transaction history, notifications, settings, merchant dashboard.")
    Container(api, "drop-api", "Hono v4, Node.js 22", "REST API server. 26+ endpoints under /v1/. BankID OIDC callback, transaction processing, recipient management, merchant registration, GDPR compliance, admin operations.")
    Container(mobile, "drop-mobile", "Expo SDK 54, React Native", "Native mobile app for iOS and Android. BankID auth via expo-web-browser deep linking. AsyncStorage for token persistence. No offline support.")
    ContainerDb(db, "Database", "PostgreSQL 16 (all environments)", "19 tables: 12 core (users, transactions, bank_accounts, sessions, merchants, recipients, etc.) + 7 compliance (audit_log, aml_alerts, str_reports, screening_results, consents, data_access_requests, complaints). Drizzle ORM.")
```

```
  }

  System_Ext(bankid, "BankID OIDC", "Norwegian eID provider. OIDC authorize/token/JWKS
endpoints for Strong Customer Authentication.")
  System_Ext(sumsub, "Sumsub", "KYC/AML identity verification. WebSDK (web), React Native SDK
(mobile), webhooks for status updates.")
  System_Ext(openbanking, "Open Banking APIs", "PSD2 AISP (read balances) and PISP (initiate
payments) via licensed provider.")
  System_Ext(sepa, "SEPA/SWIFT Networks", "International payment rails for remittance
settlement to 30+ countries.")

  Rel(user, web, "HTTPS", "Browser")
  Rel(user, mobile, "HTTPS", "Native app")
  Rel(merchant, web, "HTTPS", "Merchant dashboard")

  Rel(web, api, "HTTPS REST", "/v1/* endpoints, JSON, Bearer token or httpOnly cookie")
  Rel(mobile, api, "HTTPS REST", "/v1/* endpoints, JSON, Bearer token")

  Rel(api, db, "SQL", "Type-safe queries via Drizzle ORM (src/shared/db/)")
  Rel(api, bankid, "OIDC", "Authorization code flow, JWKS token verification")
  Rel(api, sumsub, "REST + Webhooks", "Applicant creation, document checks, status webhooks")
  Rel(api, openbanking, "PSD2 API", "AISP balance reads, PISP payment initiation with SCA")
  Rel(api, sepa, "ISO 20022", "Remittance settlement via banking partner")
```

Container Responsibilities

Container	Technology	Responsibilities	Port
drop-web	Next.js 15, React 19, Tailwind v4	SSR web app, BankID redirect initiation, UI rendering for all 10 screens (Login, Onboarding, Dashboard, SendMoney, BankAccounts, TransactionHistory, ScanQR, Profile, Notifications, MerchantDashboard)	3000

Container	Technology	Responsibilities	Port
drop-api	Hono v4, Node.js 22 Alpine	REST API, BankID OIDC callback handling, JWT session management, transaction processing, GDPR endpoints, admin operations, audit logging	3001
drop-mobile	Expo SDK 54, React Native	iOS/Android native app, BankID via `expo-web-browser` + deep link (`drop://auth/callback`), AsyncStorage for token persistence, push notifications	N/A
Database	PostgreSQL 16 (all environments)	19 tables, foreign keys enforced. Drizzle ORM schema in `src/shared/db/schema.ts`. Local: Docker port 5433. Production: AWS RDS.	5432

Request Lifecycle

```
sequenceDiagram
    participant Client as Client (Web/Mobile)
    participant CORS as CORS Middleware
    participant ReqID as Request ID Middleware
    participant IP as Client IP Middleware
    participant RL as Rate Limiter
    participant Auth as Auth Middleware
    participant Route as Route Handler
    participant DB as Database
    participant ErrH as Error Handler

    Client->>+CORS: HTTPS Request
    CORS->>CORS: Validate Origin against allowlist
    CORS->>+ReqID: Pass if origin allowed
    ReqID->>ReqID: Extract x-request-id or generate UUID
    ReqID->>ReqID: Set x-request-id response header
    ReqID->>+IP: Forward request
    IP->>IP: Extract IP from x-real-ip / x-forwarded-for
    IP->>IP: Set clientIp context variable
```

```
    alt Rate-limited endpoint
        IP->>+RL: Forward to rate limiter
        RL->>DB: SELECT count, reset_at FROM rate_limits WHERE key = ?
        DB-->>RL: Current count
        alt Under limit
            RL->>RL: UPDATE count + 1
            RL->>+Auth: Forward request
        else Over limit
            RL-->>Client: 429 Too Many Requests
        end
    else Non-rate-limited endpoint
        IP->>+Auth: Forward request
    end

    alt Authenticated endpoint
        Auth->>Auth: Extract token (Bearer header or drop_token cookie)
        Auth->>Auth: Verify JWT (jose, HS256/RS256)
        Auth->>DB: SELECT session (check revoked = 0, expires_at > now)
        DB-->>Auth: Session record
        Auth->>DB: SELECT user WHERE id = ? AND deleted_at IS NULL
        DB-->>Auth: User record
        Auth->>Auth: Set user context variable
        Auth->>+Route: Forward authenticated request
    else Public endpoint
        IP->>+Route: Forward directly
    end

    Route->>DB: Business logic queries (parameterized)
    DB-->>Route: Query results
    Route-->>Client: JSON response { data: {...} }

    Note over ErrH: On any unhandled error
    Route-->>ErrH: Error thrown
    ErrH->>ErrH: Log error, capture in Sentry
    ErrH-->>Client: { error: "internal_error", message: "..." }
```

Middleware Chain

The Hono v4 API (`drop-api`) applies middleware in the following order for every request:

Order	Middleware	Source	Purpose
1	**CORS**	`hono/cors` in `app.ts:23-30`	Validates `Origin` header against allowlist (`localhost:3000`, `localhost:3001`, `APP_URL`). Sets `credentials: true` for cookie transport.
2	**Request ID**	`app.ts:33-38`	Reads `x-request-id` header or generates `crypto.randomUUID()`. Sets on context and response header for distributed tracing.
3	**Client IP**	`app.ts:41-47`	Extracts IP from `x-real-ip` then `x-forwarded-for` (first in chain), falls back to `127.0.0.1`. Stored in context for rate limiting and audit.
4	**Rate Limiter**	`middleware/rate-limit.ts`	Per-IP rate limiting backed by `rate_limits` DB table. Configurable limit and window per route. Cleans expired entries every 100 calls.
5	**Auth**	`middleware/auth.ts`	Extracts JWT from `Authorization: Bearer` header or `drop_token` cookie. Verifies signature (jose HS256/RS256), checks session not revoked, loads user record.
6	**Merchant**	`middleware/auth.ts:21-29`	Standalone middleware that independently verifies auth (calls `extractToken` and `verifyAndGetUser`) and checks `user.role === 'merchant'`. Does NOT extend or chain authMiddleware. Returns 403 if not merchant.
7	**Global Error Handler**	`middleware/error-handler.ts`	Catches all unhandled errors. HTTPException returns structured JSON with status. Other errors return 500, log to stdout, and capture in Sentry.

Rate Limit Configuration

Endpoint Group	Limit	Window	Source
BankID initiate	10 req	60s	`routes/auth.ts:19`
BankID callback	10 req	60s	`routes/auth.ts:43`
Remittance	10 req/60s per-IP + 3 req/60s per-user	60s	`routes/transactions.ts`
QR Payment	10 req/60s per-IP + 3 req/60s per-user	60s	`routes/transactions.ts`
Exchange rates	120 req	60s	`routes/rates.ts`

Communication Patterns

Web Client to API

The Next.js web app communicates with the Hono API over HTTPS REST:

- **Authentication:** httpOnly cookie (`drop_token`) set on BankID callback redirect. Cookie attributes: `HttpOnly`, `Path=/`, `Max-Age=604800` (7 days), `SameSite=Lax`.
- **CSRF protection:** CORS origin validation + `SameSite` cookie attribute.
- **Content type:** `application/json` for all request/response bodies.
- **Error envelope:** `{ error: "code", message: "human-readable", details: [...] }`.

Mobile Client to API

The Expo mobile app uses Bearer token authentication:

- **Token storage:** `AsyncStorage` (React Native encrypted storage).
- **Auth header:** `Authorization: Bearer <jwt>`.
- **BankID flow:** `expo-web-browser` opens BankID authorize URL, redirects back via deep link `drop://auth/callback?code=&state=`.
- **Token refresh:** `POST /v1/auth/refresh` — revokes old sessions, issues new JWT, sets cookie (web) and returns token in body (mobile).

API to Database

- **Abstraction layer:** `db.ts` provides `query()`, `getOne()`, `run()`, `runIgnore()`, `runUpsert()`, `transaction()`.

- **Driver detection:** `DATABASE_URL` env var present = PostgreSQL via `pg.Pool`, absent = SQLite via `better-sqlite3`.
- **SQL compatibility:** Automatic conversion of SQLite dialect to PostgreSQL (placeholders `?` to `$N`, `datetime('now')` to `CURRENT_TIMESTAMP`, `INSERT OR IGNORE` to `ON CONFLICT DO NOTHING`).
- **Transaction isolation:** SQLite uses `BEGIN/COMMIT/ROLLBACK` on the single connection. PostgreSQL uses pool client with explicit transaction.

API to External Services

Service	Protocol	Authentication	Data Flow
BankID OIDC	HTTPS (OpenID Connect)	Client ID + Client Secret	Auth code exchange, JWKS token verification, pid extraction
Sumsub KYC	REST + Webhooks	API key + HMAC signature	Applicant creation, document verification, status webhooks
Open Banking	PSD2 REST API	OAuth2 (provider-specific)	AISP balance reads (cached in `bank_accounts.balance`), PISP payment initiation
SEPA/SWIFT	ISO 20022 (via banking partner)	Banking partner credentials	Remittance settlement to 30+ countries

---

Cross-References

- **API endpoints:** API-REFERENCE.md — Full endpoint documentation with request/response examples
- **Database schema:** DATABASE-SCHEMA.md — All 19 tables with column definitions
- **Authentication:** AUTHENTICATION.md — BankID OIDC flow, JWT structure, session management
- **Middleware:** MIDDLEWARE.md — Detailed middleware documentation
- **Security:** SECURITY-ARCHITECTURE.md — Threat model, security headers, input validation
- **Deployment:** deployment-architecture.md — AWS + Cloudflare topology, CI/CD pipeline
- **Feature flags:** FEATURE-FLAGS.md — Runtime feature gating system

Component Overview

Component Overview (C4 Level 3)

Document: HLD-002 **Version:** 1.0 **Date:** 2026-02-21 **Author:** Frontend Architect (AI Agent)
Status: Draft **Scope:** Frontend component architecture for web and mobile applications

---

1. Purpose

This document provides a C4 Level 3 component view of the Drop frontend, covering both the Next.js web application and the Expo mobile application. It maps the component tree, shared component library, page composition patterns, and design system integration.

---

2. Web Application Component Architecture

The web application is built with Next.js 15 (App Router) and React 19, using Tailwind CSS v4 for styling and shadcn/ui (Radix UI primitives) for the component library.

2.1 Component Diagram — Web App Structure

```
graph TD
    subgraph "Next.js 15 App Router"
        RootLayout["RootLayout<br/>(app/layout.tsx)"]
        RootLayout --> CookieConsent["CookieConsent"]
        RootLayout --> PWARegister["PWARegister"]

        subgraph "Public Pages (No Auth)"
            Landing["/ Landing<br/>(Server Component)"]
```

```
        LoginPage["/login LoginPage"]
        RegisterPage["/register RegisterPage"]
        TermsPage["/terms TermsPage"]
        PrivacyPage["/privacy PrivacyPage"]
        FeesPage["/fees FeesPage"]
        WithdrawalPage["/withdrawal WithdrawalPage"]
    end

    subgraph "Authenticated Pages (useAuth)"
        Dashboard["/dashboard Dashboard"]
        SendMoney["/send SendMoney"]
        ScanQR["/scan ScanQR"]
        Accounts["/accounts BankAccounts"]
        Transactions["/transactions TransactionHistory"]
        Profile["/profile ProfileHub"]
        Notifications["/notifications NotificationCenter"]
        Complaints["/complaints ComplaintForm"]
    end

    subgraph "Profile Sub-Pages"
        ProfilePersonal["/profile/personal"]
        ProfileSecurity["/profile/security"]
        ProfileNotifications["/profile/notifications"]
        ProfileLanguage["/profile/language"]
    end

    subgraph "Feature-Flagged Pages"
        Cards["/cards CardManagement<br/>(FUTURE)"]
    end

    Profile --> ProfilePersonal
    Profile --> ProfileSecurity
    Profile --> ProfileNotifications
    Profile --> ProfileLanguage
end

subgraph "Shared Components"
    BottomNav["BottomNav<br/>(5 tabs)"]
    DropLogo["DropLogo / DropWordmark /<br/>DropLogoFull / DropAppIcon"]
```

```
        PrePaymentDisclosure["PrePaymentDisclosure<br/>(PSD2 modal)"]
    end

    subgraph "shadcn/ui Primitives"
        Button["Button"]
        Card["Card"]
        Dialog["Dialog"]
        Tabs["Tabs"]
        ScrollArea["ScrollArea"]
        Input["Input"]
        Select["Select"]
        Badge["Badge"]
        Skeleton["Skeleton"]
        Sheet["Sheet"]
        Separator["Separator"]
        Avatar["Avatar"]
        Alert["Alert"]
        Sonner["Sonner (Toast)"]
    end

    Dashboard --> BottomNav
    Dashboard --> DropLogo
    Dashboard --> ScrollArea
    Transactions --> BottomNav
    Transactions --> Tabs
    ScanQR --> BottomNav
    Accounts --> BottomNav
    Accounts --> Card
    Profile --> BottomNav
    Notifications --> BottomNav
    SendMoney --> PrePaymentDisclosure
    Landing --> DropLogoFull["DropLogoFull"]
```

2.2 Page Composition Pattern

Every authenticated page follows a consistent composition:

```
+----------------------------------+
|  Header (back nav + title)       |
+----------------------------------+
```

```
|                                  |
|  Page Content                    |
|  (scrollable area)               |
|                                  |
|                                  |
+----------------------------------+
|  BottomNav (fixed, 5 tabs)       |
+----------------------------------+
```

BottomNav Tabs (Web):

Tab	Label	Route	Icon
1	Hjem	`/dashboard`	Home (lucide)
2	Aktivitet	`/history`	Clock (lucide)
3	Skann	`/scan`	IconQrScan (custom)
4	Kontoer	`/accounts`	Landmark (lucide)
5	Profil	`/profile`	User (lucide)

---

3. Mobile Application Component Architecture

The mobile application is built with React Native (Expo SDK) using Expo Router for file-based navigation.

3.1 Component Diagram — Mobile App Structure

```
graph TD
    subgraph "Expo Router (React Native)"
        RootStack["Root Stack Layout<br/>(app/_layout.js)"]

        subgraph "Auth Screens (No Auth)"
            Welcome["index.js<br/>Welcome Screen"]
            MobileLogin["login.js<br/>Login Screen"]
            MobileRegister["register.js<br/>Registration (BankID)"]
```

```
        end

        subgraph "Tab Navigator (4 tabs)"
            TabLayout["(tabs)/_layout.js"]
            MobileDashboard["(tabs)/index.js<br/>Dashboard / Home"]
            MobileSend["(tabs)/send.js<br/>Send Money"]
            MobileScan["(tabs)/scan.js<br/>QR Scanner"]
            MobileProfile["(tabs)/profile.js<br/>Profile & Settings"]
        end

        subgraph "Modal Screens"
            MobileHistory["history.js<br/>Transaction History"]
        end

        RootStack --> Welcome
        RootStack --> MobileLogin
        RootStack --> MobileRegister
        RootStack --> TabLayout
        RootStack --> MobileHistory
        TabLayout --> MobileDashboard
        TabLayout --> MobileSend
        TabLayout --> MobileScan
        TabLayout --> MobileProfile
    end

    subgraph "Shared Libraries"
        APIClient["lib/api.js<br/>Fetch wrapper + Bearer auth"]
        Theme["lib/theme.js<br/>Colors, fonts, spacing"]
    end

    MobileDashboard --> APIClient
    MobileSend --> APIClient
    MobileScan --> APIClient
    MobileProfile --> APIClient
    MobileHistory --> APIClient
```

Tab Bar (Mobile):

Tab	Label	Icon	Screen
1	Hjem	House (Unicode)	Dashboard

Tab	Label	Icon	Screen
2	Send	Arrow (Unicode)	Send money
3	QR	QR (Unicode)	QR scanner
4	Profil	Person (Unicode)	Profile

---

4. Shared Component Library

4.1 Custom Drop Components

Component	File (Web)	Mobile Equivalent	Purpose
BottomNav	`components/bottom-nav.tsx`	`(tabs)/_layout.js` Tab Bar	Primary navigation
DropLogo	`components/drop-logo.tsx`	Inline SVG in Welcome	Brand mark (green "d" + gold arrow)
DropWordmark	`components/drop-logo.tsx`	Fraunces `<Text>`	"drop" text in Fraunces font
DropLogoFull	`components/drop-logo.tsx`	N/A	Mark + wordmark combined
DropAppIcon	`components/drop-logo.tsx`	N/A	App launcher icon
CookieConsent	`components/cookie-consent.tsx`	N/A (not applicable)	GDPR consent banner
PrePaymentDisclosure	`components/pre-payment-disclosure.tsx`	N/A (inline)	PSD2 fee disclosure modal
PWARegister	`components/pwa-register.tsx`	N/A	Service Worker registration

4.2 Custom Icons (`drop-icons.tsx`)

Icon	Usage	Shared Props
IconSendMoney	Send money action button	`{ size?: number; className?: string }`
IconQrScan	QR scan action / BottomNav tab	Same
IconVirtualCard	Card feature (FUTURE)	Same
IconShield	Trust/security sections	Same
IconFastTransfer	Marketing feature highlight	Same
IconCorridors	Corridor/globe feature	Same

4.3 shadcn/ui Components (Web Only)

All shadcn/ui components live in `components/ui/` and are built on Radix UI primitives with Tailwind styling via CSS variables in `globals.css`.

Component	Radix Primitive	Used By
Button	`@radix-ui/react-slot`	All pages
Card	div-based	Accounts, Dashboard
Dialog	`@radix-ui/react-dialog`	CookieConsent, Cards
Tabs	`@radix-ui/react-tabs`	Transactions
ScrollArea	`@radix-ui/react-scroll-area`	Dashboard
Input	native input	Login, Register, Send
Select	`@radix-ui/react-select`	Complaints, Settings
Badge	cva variants	Accounts, Profile
Skeleton	div + pulse animation	Loading states
Sheet	`@radix-ui/react-dialog`	Side panels
Separator	`@radix-ui/react-separator`	Profile sections
Avatar	`@radix-ui/react-avatar`	Profile, Dashboard
Alert	div-based	Accounts (PSD2 banner)
Sonner	sonner library	Toast notifications

---

5. Design System Integration

5.1 Design Token Reference

Colors

Token	Hex	Usage
Primary Green	`#0B6E35`	Buttons, active states, BottomNav active
Primary Green Dark	`#095C2C`	Hover/pressed states
Primary Green Light	`#E8F5E9`	Light backgrounds
Gold Accent	`#D4A017`	Logo accent, QR scanner brackets, pending status

Token	Hex	Usage
Text Primary	`#1A1A1A` (web) / `#1E293B`	Headings, body text
Text Secondary	`#6B7280` (mobile) / `#64748B` (web)	Descriptions, labels
Text Muted	`#9CA3AF` (mobile) / `#94A3B8` (web)	Timestamps, hints
Background	`#FAFCF8` (mobile) / `#F8FAFC` (web)	Page backgrounds
Card	`#FFFFFF`	Card surfaces
Border	`#E5E7EB` (mobile) / `#E2E8F0` (web)	Dividers, input borders
Error	`#EF4444`	Error states
Success	`#10B981`	Success indicators

Typography

Role	Web	Mobile
Display / Headings	Fraunces (via CSS)	Fraunces_700Bold / Fraunces_600SemiBold
Body	System / Inter	DMSans_400Regular
Body Medium	System / Inter Medium	DMSans_500Medium
Body Bold	System / Inter Bold	DMSans_700Bold

Spacing (Mobile)

Token	Value
xs	4px
sm	8px
md	16px
lg	24px
xl	32px
xxl	48px

Border Radius

Token	Value
sm	8px
md	12px
lg	16px
xl	24px (rounded-2xl)

Token	Value
full	9999px (circular)

5.2 Figma-to-Code Pipeline

```
Figma Design (Make Export)
        |
        v
Figma Make Export (Vite + React)
  mockups/figma-make-export/src/app/screens/
  10 screens: Login, Onboarding, Dashboard, SendMoney,
  BankAccounts, TransactionHistory, ScanQR, Profile,
  Notifications, MerchantDashboard
        |
        v
Implementation (Next.js / Expo)
  - Web: src/drop-app/src/app/
  - Mobile: src/drop-mobile/app/
        |
        v
Visual Validation
  Screenshot vs Figma reference comparison
```

Source of truth: `mockups/figma-make-export/src/components/` contains the canonical UI for all 10 core screens. Before any UI change, the corresponding Make component must be read first.

---

6. Web vs Mobile Feature Matrix

Feature	Web (Next.js)	Mobile (Expo)
Auth storage	httpOnly cookie (`drop_token`)	Bearer token (in-memory + AsyncStorage)
Navigation	App Router (file-based)	Expo Router (Stack + Tabs)
Send money flow	4 steps (recipient, amount, review, success)	2 steps (recipient+currency, amount+confirm)
Registration	BankID-only (auto-creation on first login; POST /register returns 410)	BankID-only (auto-creation on first login)
QR scanner	Simulated camera viewfinder	Simulated camera placeholder

Feature	Web (Next.js)	Mobile (Expo)
Bottom nav	5 tabs (Hjem, Aktivitet, Skann, Kontoer, Profil)	4 tabs (Hjem, Send, QR, Profil)
Cards page	Yes (feature-flagged, default off)	No
Merchant dashboard	Yes (role-gated)	No
Bank accounts	Dedicated `/accounts` page	Balance shown on dashboard
Notifications	Dedicated `/notifications` page	Not implemented
Profile sub-pages	4 (personal, security, notifications, language)	Inline settings
Feature flags	Environment variables	Not implemented
Legal pages	Terms, Privacy, Fees, Withdrawal, Complaints	Not implemented (links only)
Offline support	PWA Service Worker registration	No offline support
Deep linking	N/A	Not configured
Push notifications	N/A	Not implemented
Biometric auth	N/A	Not implemented
UI framework	shadcn/ui (Radix) + Tailwind v4	React Native StyleSheet
State management	useState + useAuth hook	useState + api module

---

7. Accessibility Considerations (WCAG 2.1 AA)

Area	Web Implementation	Mobile Implementation
Color contrast	Primary green (#0B6E35) on white meets 4.5:1	Same color tokens, system font rendering
Focus management	Radix UI provides built-in focus trapping for Dialog, Sheet	Expo Router handles screen focus
Screen reader	Semantic HTML via shadcn/ui, lucide icons with aria-hidden	React Native accessibility props needed
Touch targets	Buttons min 44px height (py-3 = 48px)	Tab bar height 60px, buttons styled per platform
Motion	Tailwind `transition-colors` only, no complex animation	Minimal animation (SplashScreen)
Language	`lang="nb"` on html element (Norwegian)	Not configured

Area	Web Implementation	Mobile Implementation
Keyboard nav	Radix handles arrow keys, Escape, Tab	N/A (touch-first)

---

8. Cross-References

- **API endpoints consumed by frontend:** See API Reference
- **Database schema behind API responses:** See Database Schema
- **Authentication flow (BankID OIDC):** See Authentication
- **Page specifications:** See PAGES.md
- **Figma exports (UI source of truth):** `mockups/figma-make-export/src/app/screens/`
- **Login authentication flow:** See flow-login-authentication.md
- **QR payment flow:** See flow-qr-payment.md
- **Transaction history flow:** See flow-transaction-history.md
- **Notification flow:** See flow-notifications.md
- **Merchant onboarding flow:** See flow-merchant-onboarding.md

Deployment Architecture

Deployment Architecture

> AWS deployment topology, Cloudflare edge layer, Docker multi-stage build, CI/CD pipeline, environment strategy, auto-scaling, health checks, and rollback procedures for the Drop fintech platform.

Deployment Topology

> **Note:** AWS App Runner is the PLANNED production deployment target. Current deployment uses Docker Compose only (`docker-compose.yml` and `docker-compose.production.yml`). No CI/CD pipeline, ECR, or GitHub Actions are configured yet.

```
graph TB
    subgraph Internet
        User[End Users]
        Mobile[Mobile App]
    end

    subgraph Cloudflare["Cloudflare Edge"]
        DNS[DNS<br/>getdrop.no]
        CDN[CDN<br/>Static assets cache]
        WAF[WAF Rules<br/>Rate limiting, bot protection,<br/>geo-blocking, OWASP rules]
        DDoS[DDoS Protection<br/>L3/L4/L7 mitigation]
    end

    subgraph AWS["AWS eu-north-1 (Stockholm)"]
        subgraph AppRunner["AWS App Runner (PLANNED)"]
            WebContainer[drop-web<br/>Next.js 15 standalone<br/>Node.js 22 Alpine<br/>Port
```

```
3000]
            APIContainer[drop-api<br/>Hono v4<br/>Node.js 22 Alpine<br/>Port 3001]
        end

        subgraph DataLayer["Data Layer"]
            RDS[(RDS PostgreSQL 16<br/>db.t3.micro → db.r6g.large<br/>Multi-AZ
failover<br/>Automated backups)]
        end

        subgraph Supporting["Supporting Services"]
            ECR[ECR<br/>Container Registry<br/>Image scanning enabled]
            SM[Secrets
Manager<br/>JWT_SECRET<br/>BANKID_CLIENT_SECRET<br/>DATABASE_URL<br/>SENTRY_DSN]
            CW[CloudWatch<br/>Logs + Metrics<br/>Alarm triggers]
        end
    end

    User -->|HTTPS| DNS
    Mobile -->|HTTPS| DNS
    DNS --> CDN
    CDN --> WAF
    WAF --> DDoS
    DDoS -->|Origin pull| WebContainer
    DDoS -->|Origin pull| APIContainer
    WebContainer --> RDS
    APIContainer --> RDS
    AppRunner --> ECR
    AppRunner --> SM
    AppRunner --> CW
```

CI/CD Pipeline

```
flowchart LR
    subgraph Trigger
        Push[git push]
        PR[Pull Request]
    end
```

```
    subgraph Build["Build Stage"]
        Checkout[Checkout code]
        Deps[npm ci]
        TypeCheck[tsc --noEmit]
        Lint[eslint]
        Test[vitest run]
    end

    subgraph Package["Package Stage"]
        DockerBuild[Docker multi-stage build]
        ImageScan[ECR image scan]
        PushECR[Push to ECR]
    end

    subgraph DeployStaging["Deploy: Staging"]
        DeployStagingEnv[Deploy to App Runner<br/>staging service]
        SmokeTest[Smoke test<br/>GET /v1/health]
        E2ETest[E2E test suite]
    end

    subgraph DeployProd["Deploy: Production"]
        Approval[Manual approval gate]
        BlueGreen[Blue/green swap<br/>App Runner traffic shift]
        HealthVerify[Health check verification<br/>3 consecutive passes]
        Rollback{Healthy?}
    end

    Push --> Checkout
    PR --> Checkout
    Checkout --> Deps --> TypeCheck --> Lint --> Test
    Test --> DockerBuild --> ImageScan --> PushECR
    PushECR --> DeployStagingEnv --> SmokeTest --> E2ETest
    E2ETest --> Approval --> BlueGreen --> HealthVerify --> Rollback
    Rollback -->|Yes| Done[Production Live]
    Rollback -->|No| RollbackAction[Revert to previous revision]
```

Pipeline Stages Detail

Stage	Tool	Timeout	Failure Action
Checkout	`actions/checkout@v4`	1m	Fail pipeline
Install deps	`npm ci`	5m	Fail pipeline
TypeScript check	`tsc --noEmit`	3m	Fail pipeline
Lint	`eslint .`	2m	Fail pipeline
Unit tests	`vitest run`	5m	Fail pipeline
Docker build	Multi-stage (4 stages: deps, test, builder, runner)	10m	Fail pipeline
Image scan	ECR vulnerability scan	5m	Warn on HIGH, block on CRITICAL
Push to ECR	`docker push`	3m	Fail pipeline
Deploy staging	App Runner update	10m	Fail pipeline
Smoke test	`curl /v1/health`	1m	Rollback staging
Manual approval	GitHub environment protection	24h	Pipeline expires
Production deploy	App Runner traffic shift	10m	Auto-rollback
Health verification	3x `GET /v1/health` at 10s intervals	1m	Auto-rollback

Docker Multi-Stage Build

Source: `src/drop-app/Dockerfile`

```
┌──────────────────────────────────────────────┐
│ Stage 1: deps (node:22-alpine)               │
│                                              │
│ • Install python3, make, g++ (native deps)   │
│ • COPY package*.json                         │
│ • npm ci (production + dev deps)             │
│ • Output: /app/node_modules                  │
├──────────────────────────────────────────────┤
│ Stage 2: test (node:22-alpine)               │
│                                              │
│ • COPY node_modules from deps                │
│ • COPY source code                           │
│ • Run vitest + coverage checks               │
```

```
│ • Mandatory test gate — blocks build on     │
│   failure                                   │
├─────────────────────────────────────────────┤
│ Stage 3: builder (node:22-alpine)           │
│                                             │
│ • COPY node_modules from deps               │
│ • COPY source code                          │
│ • npm run build (Next.js standalone output) │
│ • Output: .next/standalone, .next/static    │
├─────────────────────────────────────────────┤
│ Stage 4: runner (node:22-alpine)            │
│                                             │
│ • Non-root user: nextjs (UID 1001)          │
│ • Install python3, make, g++ (native deps)  │
│ • COPY public/ from builder                 │
│ • COPY .next/standalone from builder        │
│ • COPY .next/static from builder            │
│ • Data dir: /app/data (owned by nextjs)     │
│ • No source code                            │
│ • CMD: node server.js                       │
└─────────────────────────────────────────────┘
```

Security features in runner stage:

- Non-root user `nextjs` (UID 1001, GID `nodejs` 1001)
- **Note:** Runner stage currently includes `python3`, `make`, `g++` (installed via `apk add` for native dependency rebuilds). These should be removed in a future optimization.
- No source code — only compiled standalone output
- Data directory `/app/data` owned by `nextjs:nodejs`

Environment Configuration

Variable	Dev	Staging	Production	Source
`NODE_ENV`	`development`	`production`	`production`	Dockerfile ENV
`JWT_SECRET`	Dev fallback (static string `'dev-secret-change-in-production'`)	Secrets Manager	Secrets Manager	`auth.ts:8`

Variable	Dev	Staging	Production	Source
`DATABASE_URL`	Not set (SQLite)	RDS connection string	RDS connection string	Secrets Manager
`BANKID_CLIENT_ID`	Not set	BankID test env	BankID prod env	Secrets Manager
`BANKID_CLIENT_SECRET`	Not set	BankID test env	BankID prod env	Secrets Manager
`BANKID_MOCK`	`true`	`false`	`false`	App Runner env
`BANKID_CALLBACK_URL`	`http://localhost:3000/api/auth/bankid/callback`	`https://staging.getdrop.no/...`	`https://getdrop.no/...`	App Runner env
`NEXT_PUBLIC_SERVICE_MODE`	`demo`	`mock` or `live`	`live`	Build-time env
`SEED_DEMO`	implicit (non-prod)	`true`	Not set	App Runner env
`SENTRY_DSN`	Not set	Sentry staging project	Sentry prod project	Secrets Manager
`APP_URL`	`http://localhost:3000`	`https://staging.getdrop.no`	`https://getdrop.no`	App Runner env
`PORT`	`3000`	`3000`	`3000`	App Runner default

Environment Strategy

Environment	Purpose	Database	BankID	Data
Development	Local development, `docker compose up`	SQLite at `./data/drop.db`	Mock (`BANKID_MOCK=true`)	Demo seed data
Staging	Pre-release validation, QA, E2E tests	RDS PostgreSQL (separate instance)	BankID test environment	Demo seed data (`SEED_DEMO=true`)
Production	Live service	RDS PostgreSQL (Multi-AZ, automated backups)	BankID production	Real user data only

---

Scaling Configuration

App Runner Auto-Scaling

Parameter	Web Container	API Container
Min instances	1	1
Max instances	5	10

Parameter	Web Container	API Container
Concurrency target	50 req/instance	100 req/instance
Scale-up cooldown	30s	30s
Scale-down cooldown	300s	300s
CPU	1 vCPU	1 vCPU
Memory	2 GB	2 GB

Scaling Triggers

Metric	Threshold	Action
Concurrent requests per instance	> 80% of target	Scale up
Concurrent requests per instance	< 25% of target for 5m	Scale down
Response time p95	> 500ms for 3m	Scale up + CloudWatch alarm
Error rate (5xx)	> 5% for 2m	CloudWatch alarm, no auto-scale
CPU utilization	> 80% for 3m	Scale up

RDS PostgreSQL Scaling

Parameter	Staging	Production
Instance class	`db.t3.micro`	`db.t3.medium` (initial) → `db.r6g.large`
Storage	20 GB gp3	100 GB gp3, auto-scaling to 500 GB
Multi-AZ	No	Yes
Read replicas	0	0 (add when needed)
Backup retention	7 days	30 days
Maintenance window	Sunday 03:00 UTC	Sunday 03:00 UTC

Health Check Endpoints

API Health Check

Endpoint: `GET /v1/health` (Hono API) **Source:** `routes/health.ts`

```
// Success (200)
{
  "status": "ok",
  "version": "0.1.0",
  "uptime": 3600,
  "db": "connected",
  "dbLatencyMs": 1,
  "timestamp": "2026-02-21T12:00:00.000Z"
}

// Failure (503)
{
  "status": "error",
  "db": "disconnected",
  "timestamp": "2026-02-21T12:00:00.000Z"
}
```

Health check performs: `SELECT 1 as ok` query to verify database connectivity and measure latency.

Health Check Configuration

Component	Interval	Timeout	Retries	Grace Period
App Runner (web)	10s	5s	3	30s
App Runner (API)	10s	5s	3	30s
Docker Compose (dev)	30s	10s	3	10s
Cloudflare origin health	60s	10s	2	N/A

Blue/Green Deployment (Aspirational)

> **Note:** App Runner does NOT have built-in blue/green deployment (see ADR-012). The following describes an aspirational traffic-shifting strategy that would need custom implementation. App Runner performs rolling updates by default.

```
1. New revision deployed alongside current (blue)
2. New revision (green) starts and passes health checks
3. Traffic gradually shifted: 0% → 10% → 50% → 100%
4. If health checks pass for 60s at 100% → old revision drained
5. If health checks fail → immediate rollback to blue
```

Deployment Checklist

1. All CI checks pass (TypeScript, lint, tests)
2. Docker image built and scanned (no CRITICAL vulnerabilities)
3. Image pushed to ECR
4. Staging deployment succeeds
5. Smoke tests pass (`GET /v1/health` returns 200)
6. Manual approval (production deployments only)
7. Production deployment with health verification
8. Post-deployment monitoring (15 minutes)

Rollback Procedures

Automatic Rollback

App Runner automatically rolls back if:

- New revision fails health checks within grace period
- Health check failure rate exceeds threshold during traffic shift
- Container crashes on startup (exit code != 0)

Manual Rollback

```
# List recent revisions
aws apprunner list-operations --service-arn $SERVICE_ARN

# Rollback to previous revision
aws apprunner update-service \
  --service-arn $SERVICE_ARN \
  --source-configuration '{"ImageRepository":{"ImageIdentifier":"<previous-ecr-image>"}}'

# Verify rollback
```

```
curl https://getdrop.no/v1/health
```

Database Rollback

For database schema changes, migrations are forward-only. In case of issues:

1. **SQLite (dev/staging):** Restore from backup (see DEPLOYMENT.md backup section)
2. **PostgreSQL (prod):** RDS point-in-time recovery to any second within retention window (30 days)

```
# RDS point-in-time restore
aws rds restore-db-instance-to-point-in-time \
  --source-db-instance-identifier drop-prod \
  --target-db-instance-identifier drop-prod-restored \
  --restore-time "2026-02-21T11:00:00Z"
```

Cloudflare Configuration

Feature	Configuration	Purpose
DNS	`getdrop.no` → App Runner CNAME (proxied)	Domain routing
SSL/TLS	Full (strict)	End-to-end encryption
CDN	Cache static assets (`/_next/static/*`, `/public/*`)	Performance
WAF	OWASP Core Rule Set, rate limiting rules	Security
DDoS	L3/L4/L7 auto-mitigation	Availability
Bot management	Challenge mode for suspicious traffic	Security
Geo-blocking	Allow: NO, SE, DK, FI (Scandinavia) + test regions	Compliance
Page rules	`/*` → SSL always, HSTS	Security

Cloudflare WAF Rules

Rule	Action	Purpose
OWASP Core Rule Set	Block	SQL injection, XSS, path traversal

Rule	Action	Purpose
Rate limit: `/v1/auth/*`	Challenge at 20 req/10s	Auth endpoint abuse prevention
Rate limit: `/v1/transactions/*`	Block at 30 req/10s	Transaction abuse prevention
Country block: Sanctioned countries	Block	OFAC/UN sanctions compliance
Bot score < 30	Challenge	Bot traffic mitigation

Cross-References

- **Container diagram:** container-diagram.md — C4 Level 2 container architecture
- **Deployment guide:** DEPLOYMENT.md — Docker compose, backup/restore procedures
- **Security architecture:** SECURITY-ARCHITECTURE.md — Security headers, CSRF, rate limiting
- **Feature flags:** FEATURE-FLAGS.md — Environment variable driven feature gating
- **Database schema:** DATABASE-SCHEMA.md — All 19 tables

Security Architecture

Security Architecture — High-Level Design

Version: 1.0 **Date:** 2026-02-21 **Author:** Banking Architecture Team **Status:** Approved **Applies to:** Drop — Security Threat Model & Controls

---

1. Overview

Drop is a PSD2-regulated fintech application that processes financial transactions (remittance, QR payments) without holding customer funds. This document defines the security architecture: trust boundaries, threat model (STRIDE), SCA implementation, fraud detection, AML screening, data classification, and encryption strategy.

Security posture summary:

- All authentication via BankID OIDC (SCA by default)
- All payment SCA delegated to ASPSP (user's bank)
- JWT tokens in httpOnly cookies (web) or AsyncStorage (mobile)
- Parameterized SQL queries (no string concatenation)
- Input sanitization on all user-facing endpoints
- Compliance tables for audit, AML, STR, screening, consents, GDPR

---

2. Trust Boundaries

```
graph TB
    subgraph Internet["Internet (Untrusted)"]
        Browser["Web Browser"]
        Mobile["Mobile App (Expo)"]
        Attacker["Potential Attacker"]
    end
```

```
    subgraph CDN["CDN / Edge (Cloudflare)"]
        WAF["WAF + DDoS Protection"]
        TLS["TLS Termination"]
    end

    subgraph AppTier["Application Tier (AWS App Runner)"]
        subgraph NextJS["Next.js BFF"]
            WebRoutes["Web API Routes<br/>/api/auth/*, /api/transactions/*"]
            Middleware["Auth Middleware<br/>Rate Limiter<br/>CSRF Validator<br/>Input
Sanitizer"]
        end
        subgraph Hono["Hono API"]
            MobileRoutes["Mobile API Routes<br/>/v1/auth/*, /v1/transactions/*"]
            HonoMiddleware["Auth Middleware<br/>Rate Limiter"]
        end
    end

    subgraph DataTier["Data Tier (Private Subnet)"]
        SQLite["SQLite / PostgreSQL<br/>19 tables (12 core + 7 compliance)"]
    end

    subgraph ExternalServices["External Services (Trusted Partners)"]
        BankID["BankID OIDC<br/>(auth.bankid.no)"]
        ASPSP["ASPSPs<br/>(DNB, SpareBank 1, Nordea)"]
        FX["FX Rate Provider"]
        KYC["KYC Provider<br/>(Sumsub - future)"]
    end

    Browser -->|"HTTPS<br/>TB1: Internet→Edge"| WAF
    Mobile -->|"HTTPS<br/>TB1: Internet→Edge"| WAF
    Attacker -.->|"Blocked by WAF"| WAF
    WAF -->|"TB2: Edge→App"| Middleware
    WAF -->|"TB2: Edge→App"| HonoMiddleware
    Middleware --> WebRoutes
    HonoMiddleware --> MobileRoutes
    WebRoutes -->|"TB3: App→Data"| SQLite
    MobileRoutes -->|"TB3: App→Data"| SQLite
    WebRoutes -->|"TB4: App→External<br/>mTLS"| BankID
    WebRoutes -->|"TB4: App→External<br/>eIDAS cert"| ASPSP
```

```
    MobileRoutes -->|"TB4: App→External"| BankID

    style Internet fill:#ff6b6b,stroke:#333,color:#fff
    style CDN fill:#ffd93d,stroke:#333
    style AppTier fill:#6bcb77,stroke:#333
    style DataTier fill:#4d96ff,stroke:#333,color:#fff
    style ExternalServices fill:#845ec2,stroke:#333,color:#fff
```

Trust Boundary Definitions

Boundary	From	To	Protection
TB1: Internet to Edge	Browser/Mobile	Cloudflare	TLS 1.3, WAF rules, DDoS mitigation
TB2: Edge to Application	Cloudflare	Next.js/Hono	HTTPS, auth middleware, rate limiting
TB3: Application to Data	API layer	SQLite/PostgreSQL	Parameterized queries, file permissions
TB4: Application to External	API layer	BankID/ASPSP	mTLS (eIDAS QWAC), JWKS verification

---

3. STRIDE Threat Model

3.1 Threat Matrix

Component	Spoofing	Tampering	Repudiation	Info Disclosure	DoS	Elevation
BankID Auth	L: BankID handles identity	L: JWKS signature verification	L: Audit log + session tracking	M: pid hash exposure risk	M: Rate limit 10/min	L: Role check on every request
JWT Tokens	M: Token theft via XSS	L: HS256 signature	L: Session table tracks all JWTs	M: Payload contains userId	L: 7d expiry	M: Role claim in JWT
PISP Payments	L: SCA required per payment	M: Amount/payee tampering	L: Audit log + idempotency_key	L: Disclosure before payment	M: Rate limit 10/min	L: KYC check before remittance
AISP Balance	L: Consent required	L: Read-only from ASPSP	L: balance_synced_at tracking	M: Cached balance visible	L: Max 4 reads/day	N/A

Component	Spoofing	Tampering	Repudiation	Info Disclosure	DoS	Elevation
Database	L: No direct access	M: SQL injection risk	L: audit_log table	H: PII in users table	L: Rate limiting	L: User-scoped queries
API Endpoints	M: CSRF on web	M: Input manipulation	L: Audit logging	M: Error message leakage	H: Unthrottled endpoints	M: IDOR if user_id not checked

Risk levels: L = Low (mitigated), M = Medium (partial mitigation), H = High (needs attention), N/A = Not applicable

3.2 Detailed Threat Analysis

S — Spoofing

Threat	Attack Vector	Mitigation	Status
Identity spoofing	Stolen credentials	BankID OIDC (SCA: possession + knowledge)	Implemented
Session hijacking	Token theft	httpOnly + secure + sameSite=Lax cookies	Implemented
CSRF	Forged cross-origin request	State parameter (OIDC), Origin header validation	Implemented
Replay attack	Reuse old auth code	Nonce in OIDC flow, one-time code exchange	Implemented

T — Tampering

Threat	Attack Vector	Mitigation	Status
SQL injection	Malicious input in queries	Parameterized queries (all 24 endpoints)	Implemented
XSS	Script injection in fields	React auto-escaping, CSP headers, sanitizeText()	Implemented
Payment amount tampering	Modified request body	Server-side validation, SCA dynamic linking	Implemented
JWT modification	Altered token claims	HS256 signature verification	Implemented

R — Repudiation

Threat	Attack Vector	Mitigation	Status
Deny transaction	User claims they didn't authorize	BankID SCA log + audit_log table	Partial (audit_log exists, SCA tracking needed)

Threat	Attack Vector	Mitigation	Status
Deny consent	User claims no consent given	consents table with IP address + timestamp	Implemented
Admin action denial	Unauthorized changes	audit_log with user_agent and ip_address	Implemented

I — Information Disclosure

Threat	Attack Vector	Mitigation	Status
PII exposure	Database breach	Encryption at rest (planned), PID hashed with SHA-256	Partial
Card data exposure	API response leakage	Masked to last 4 digits, CVV hidden	Implemented
Bank account exposure	API response leakage	Masked to last 4 digits in recipient list	Implemented
Error message leakage	Verbose error responses	Centralized error handler, generic messages	Implemented

D — Denial of Service

Threat	Attack Vector	Mitigation	Status
API flooding	High request volume	Rate limiting (10-120/min per endpoint)	Implemented
Auth brute force	Repeated login attempts	BankID handles (locks after failures)	Implemented
Database exhaustion	Large data queries	Pagination (max 50/page), query limits	Implemented
Resource exhaustion	Large payloads	Input length limits (sanitizeText)	Implemented

E — Elevation of Privilege

Threat	Attack Vector	Mitigation	Status
IDOR	Access other user's data	`AND user_id = ?` on all queries	Implemented
Role escalation	Modify role claim	Server-side role check, role in DB not just JWT	Implemented

Threat	Attack Vector	Mitigation	Status
Merchant impersonation	Access merchant dashboard	`role = 'merchant'` check on merchant routes. **Note:** merchant role currently grants admin access (audit, screening, STR) via `isAdmin(role) === role === 'merchant'` in `admin.ts`	Implemented
KYC bypass	Skip verification	`kyc_status = 'approved'` check before remittance	Implemented

---

4. SCA Implementation

4.1 Two-Level SCA

Drop implements SCA at two levels:

Level	Purpose	Provider	Method
App Authentication	Login to Drop	BankID OIDC	BankID app (possession) + code/biometrics (knowledge/inherence)
Payment Authorization	Approve PISP payment	ASPSP via BankID	BankID at bank (dynamic linking: amount + payee)

4.2 SCA Factors

Factor Type	BankID Implementation
Knowledge	Personal code / PIN
Possession	Mobile device with BankID app / code generator
Inherence	Biometrics (fingerprint/face on mobile BankID)

PSD2 RTS Art. 4: At least 2 of 3 factors required. BankID provides 2 by default (possession + knowledge or inherence).

4.3 Dynamic Linking (PISP)

For every PISP payment, PSD2 RTS Art. 97(2) requires:

1. User sees **exact amount** and **payee name** during SCA

2. Authentication code is **cryptographically bound** to amount + payee
3. Any change to amount or payee **invalidates** the authentication

This is handled by the ASPSP's BankID integration — Drop passes `instructedAmount` and `creditorName` in the PISP API call, and the bank displays these during BankID authentication.

---

5. Fraud Detection Pipeline

```
flowchart TD
    A[Transaction Request] --> B[Pre-Transaction Checks]

    B --> C{User KYC Status}
    C -->|pending/rejected| D[REJECT: kyc_required]
    C -->|approved| E[Amount Validation]

    E --> F{Amount in range?}
    F -->|No| G[REJECT: validation_error]
    F -->|Yes| H[Velocity Check]

    H --> I{Exceeds daily/weekly limit?}
    I -->|Yes| J[FLAG: velocity_alert<br/>Insert into aml_alerts<br/>severity: medium]
    I -->|No| K[Pattern Analysis]

    K --> L{Structuring detected?<br/>Multiple txns just below threshold}
    L -->|Yes| M[FLAG: structuring_alert<br/>Insert into aml_alerts<br/>severity: high]
    L -->|No| N[Corridor Risk Check]

    N --> O{High-risk corridor?}
    O -->|Yes| P[Enhanced due diligence<br/>FLAG if first-time corridor]
    O -->|No| Q[Recipient Screening]

    Q --> R{Recipient on sanctions list?}
    R -->|Yes| S[BLOCK: sanctions_match<br/>Insert into screening_results<br/>result: match]
    R -->|No| T[APPROVE: Proceed to PISP]

    J --> T
    M --> U[Escalate to compliance officer<br/>Insert into str_reports<br/>status: draft]
    P --> T
```

```
    style D fill:#ff6b6b,color:#fff
    style G fill:#ff6b6b,color:#fff
    style S fill:#ff6b6b,color:#fff
    style T fill:#6bcb77,color:#fff
    style J fill:#ffd93d
    style M fill:#ffd93d
    style U fill:#ff9f43
```

5.1 Detection Rules

Rule	Trigger	Severity	Action
Velocity limit (`checkVelocity`)	> 5 transactions in 1 hour	Medium	`aml_alerts` record, continue with flag
Structuring detection (`checkStructuring`)	3+ transactions in 24h totaling > 50,000 NOK	High	`aml_alerts` + `str_reports` draft
High-value single (`checkHighAmount`)	Single transaction > 100,000 NOK	High	Enhanced monitoring, `aml_alerts` record
High-risk corridor (`checkHighRiskCorridor`)	Country on FATF grey/black list	High	Enhanced due diligence required
Unusual pattern (`checkUnusualPattern`)	Transaction amount > 5x user's average	Medium	`aml_alerts` record
Sanctions match	Recipient matches sanctions list	Critical	Block transaction, escalate
PEP match	User matches PEP database	High	Enhanced due diligence

These rules are implemented in `transaction-monitor.ts` and run on each remittance creation.

5.2 AML Screening Tables

Table	Purpose	Key Columns
`aml_alerts`	Transaction monitoring flags	`alert_type`, `severity`, `status` (open/investigating/resolved/escalated/filed)
`str_reports`	Suspicious Transaction Reports to authorities	`report_type`, `status` (draft/submitted/acknowledged), `reference_number`
`screening_results`	PEP/sanctions/adverse media checks	`screening_type`, `result` (clear/match/potential_match/error)

---

6. Data Classification

6.1 Classification Levels

Level	Description	Examples	Storage	Access
CRITICAL	Financial credentials, encryption keys	JWT_SECRET, BANKID_CLIENT_SECRET, eIDAS private keys	Vaultwarden only	Application runtime only
RESTRICTED	PII subject to GDPR	name, email, phone, date_of_birth, national_id_hash	Encrypted at rest (planned), DB access layer	Authenticated user (own data only)
CONFIDENTIAL	Financial data	transactions, bank balances, exchange rates, fees	DB with user-scoped access	Authenticated user (own data only)
INTERNAL	Operational data	audit_log, rate_limits, sessions	DB	System processes, compliance officers
PUBLIC	Non-sensitive	exchange rates (GET /api/rates), health check	DB / API	Unauthenticated

6.2 Data Classification by Table

Table	Classification	PII Fields	Encryption at Rest	Retention
`users`	RESTRICTED	email, first_name, last_name, phone, date_of_birth, national_id_hash	Planned	5 years post-deletion (AML)
`bank_accounts`	RESTRICTED	account_number, iban	Planned	Active + 5 years
`transactions`	CONFIDENTIAL	amount, recipient details	Planned	5 years (AML/tax)
`recipients`	RESTRICTED	name, bank_account	Planned	Active + 5 years
`sessions`	INTERNAL	token_hash	N/A (hash only)	30 days
`audit_log`	INTERNAL	ip_address, user_agent	Planned	5 years
`aml_alerts`	CONFIDENTIAL	details	Planned	5 years
`str_reports`	CONFIDENTIAL	details, reference_number	Planned	10 years
`screening_results`	CONFIDENTIAL	match_details	Planned	5 years

Table	Classification	PII Fields	Encryption at Rest	Retention
`consents`	RESTRICTED	ip_address	Planned	Until withdrawn + 5 years
`merchants`	CONFIDENTIAL	None (business data)	Planned	Active + 5 years
`cards`	RESTRICTED	last_four, token_ref	Planned	Active + 5 years
`data_access_requests`	INTERNAL	None (metadata only)	N/A	5 years
`complaints`	INTERNAL	None (user text)	Planned	5 years
`notifications`	INTERNAL	None	N/A	90 days
`settings`	INTERNAL	None (preferences)	N/A	Active
`spending_limits`	INTERNAL	None	N/A	Active
`exchange_rates`	PUBLIC	None	N/A	Indefinite
`rate_limits`	INTERNAL	None	N/A	Transient

---

7. Encryption

7.1 Encryption in Transit

Connection	Protocol	Certificate
Browser to Drop	TLS 1.3 (Cloudflare)	Cloudflare managed
Mobile to Drop	TLS 1.3	Cloudflare managed
Drop to BankID	TLS 1.2+	BankID server cert
Drop to ASPSP	mTLS (eIDAS QWAC)	Qualified Website Authentication Certificate
Drop to Database	N/A (SQLite local) / TLS (PostgreSQL)	PostgreSQL server cert

7.2 Encryption at Rest

Data	Current	Target
PostgreSQL 16 (all environments)	AWS RDS encryption (AES-256, TLS 1.3)	Active
Secrets (JWT_SECRET, etc.)	Vaultwarden	Vaultwarden + AWS Secrets Manager
Backups	Not encrypted	AES-256 encrypted backups
Logs	Plain text	Encrypted log storage

7.3 Key Management

Key	Purpose	Storage	Rotation
`JWT_SECRET`	Sign Drop JWTs	Vaultwarden / env var	Every 90 days
`BANKID_CLIENT_SECRET`	BankID OIDC client auth	Vaultwarden / env var	Per BankID policy
eIDAS QWAC private key	mTLS to ASPSPs	HSM (planned)	Per certificate lifecycle
eIDAS QSeal private key	Sign API requests	HSM (planned)	Per certificate lifecycle
`qr_hmac_key` (merchants)	HMAC for QR code verification	DB (`merchants` table)	Per merchant, on creation

7.4 Hashing

Data	Algorithm	Purpose	Source
Passwords	bcrypt (cost 12)	Password verification	`utils-server.ts:8-16`
National ID (pid)	SHA-256	User deduplication	`bankid.ts:211`
JWT tokens	SHA-256	Session lookup	`auth.ts:59`
PIN codes	bcrypt	Card PIN verification	`cards/[id]/pin/route.ts`

---

8. Security Controls Summary

8.1 Application Security

Control	Implementation	Source
Authentication	BankID OIDC (SCA)	`bankid.ts`, `auth.ts`
Authorization	JWT + role check + user_id scoping	`middleware/auth.ts`
Input validation	sanitizeText, validateName, validateAmount, etc.	`middleware/validation.ts`
SQL injection prevention	Parameterized queries (all endpoints)	`db.ts`
XSS prevention	React auto-escaping + CSP + sanitization	`next.config.ts`, `validation.ts`
CSRF prevention	Origin validation + sameSite=Lax cookies	`app.ts:23-30` (CORS)
Rate limiting	Per-IP, persistent (SQLite-backed)	`middleware/rate-limit.ts`

Control	Implementation	Source
Session management	Server-side tracking with revocation	`sessions` table, `auth.ts`

8.2 Infrastructure Security

Control	Implementation	Status
TLS 1.3	Cloudflare edge	Active (landing page)
WAF	Cloudflare WAF rules	Active (landing page)
DDoS protection	Cloudflare automatic	Active
HSTS	`max-age=63072000; includeSubDomains; preload`	Configured (`next.config.ts`)
X-Frame-Options	`DENY`	Configured
X-Content-Type-Options	`nosniff`	Configured
Referrer-Policy	`strict-origin-when-cross-origin`	Configured
Permissions-Policy	Camera (self), microphone (none), geolocation (self)	Configured

8.3 Compliance Controls

Control	Implementation	Table
Audit trail	All significant actions logged	`audit_log`
AML monitoring	Transaction pattern detection	`aml_alerts`
STR filing	Suspicious transaction reports	`str_reports`
PEP/sanctions screening	Automated list checking	`screening_results`
GDPR consent tracking	Consent grant/withdraw with IP	`consents`
Data access requests	GDPR Art. 15-17	`data_access_requests`
Complaint handling	Finansavtaleloven compliance	`complaints`

---

9. Security Audit Results

9.1 Pre-Hardening (2026-02-12)

Severity	Count

CRITICAL	4
HIGH	5
MEDIUM	6
LOW	4

9.2 Post-Hardening (2026-02-13)

Severity	Count	Details
CRITICAL	0	All resolved
HIGH	0	All resolved
MEDIUM	2	CSP tightening (nonce-based), proxy config
LOW	4	Acknowledged, out of scope for MVP

9.3 Key Remediations

Finding	Fix	Source
C1: Card data stored in plain	Now stores only `last_four` + `token_ref`	Schema change
C2: Demo credentials in production	Gated behind `NODE_ENV !== 'production'` (note: `SEED_DEMO=true` can override this check)	`db.ts:241`
C4: SHA-256 password hashes	Removed entirely, bcrypt only	`utils-server.ts`
C6/H1: No session revocation	Implemented in `sessions` table	`auth.ts:56-65`
H4: No input sanitization	sanitizeText() on all text fields	`validation.ts`
M5: Notification ID injection	Validated format + max 100 per request	`notifications/route.ts`
M6: Settings value injection	Currency/language whitelists	`settings/route.ts`

---

10. Cross-References

- **Existing Security Docs:** ../../security/SECURITY-ARCHITECTURE.md — Detailed implementation-level security
- **Compliance Status:** ../../security/COMPLIANCE.md — Regulatory readiness assessment
- **BankID OIDC:** ../integration/bankid-oidc-integration.md — Authentication flow details

- **Open Banking:** ../integration/open-banking-aisp-pisp.md — ASPSP SCA, consent security
- **Payment Processing:** ../integration/payment-processing.md — Transaction integrity, idempotency
- **Database Schema:** ../../backend/DATABASE-SCHEMA.md — All 19 tables including compliance tables
- **API Reference:** ../../backend/API-REFERENCE.md — Endpoint security requirements
- **Authentication:** ../../backend/AUTHENTICATION.md — JWT, session, rate limiting details

Data Architecture

Data Architecture

Version: 1.0 **Date:** 2026-02-21 **Status:** Approved **Owner:** Database Architect

---

Overview

Drop's data architecture supports a PSD2 pass-through fintech application with two core functions: international remittances and QR merchant payments. The system manages 19 tables across 5 domains, backed by PostgreSQL 16 (all environments: development, CI, staging, production) via Drizzle ORM. See ADR-014.

Drop never holds customer funds. The `bank_accounts.balance` field is a cached AISP read from the user's real bank account -- not a Drop-held balance.

Domain Model

The 19 tables are organized into 5 logical domains:

```
erDiagram
    %% User Domain
    users ||--o{ bank_accounts : "links"
    users ||--o{ cards : "owns"
    users ||--o{ recipients : "saves"
    users ||--o{ transactions : "initiates"
    users ||--o{ sessions : "authenticates"
    users ||--o{ notifications : "receives"
    users ||--|| settings : "configures"
    users ||--o{ spending_limits : "sets"

    %% Financial Domain
    transactions }o--o| recipients : "sends to"
    transactions }o--o| merchants : "pays"
```

```
users ||--o{ merchants : "registers as"
cards ||--o{ spending_limits : "limited by"

%% KYC/AML Domain
users ||--o{ screening_results : "screened"
users ||--o{ aml_alerts : "flagged"
aml_alerts ||--o{ str_reports : "escalated to"
transactions ||--o{ aml_alerts : "triggers"

%% GDPR Domain
users ||--o{ consents : "grants"
users ||--o{ data_access_requests : "submits"
users ||--o{ complaints : "files"

%% System Domain
users ||--o{ audit_log : "generates"

users {
    text id PK "usr_ prefix"
    text email UK
    text password_hash
    text first_name
    text last_name
    text phone
    text date_of_birth
    text kyc_status "pending|approved|rejected"
    text role "user|merchant"
    text risk_level "low|medium|high"
    text pep_status
    text national_id_hash
    text deleted_at
    text created_at
}

bank_accounts {
    text id PK
    text user_id FK
    text bank_name
    text account_number
    text iban
```

```
    integer balance "cached AISP read"
    text currency
    integer is_primary
}

transactions {
    text id PK
    text user_id FK
    text type "remittance|qr_payment"
    text status "processing|completed|failed"
    integer amount
    text currency
    integer fee
    text recipient_id FK
    text merchant_id FK
    real exchange_rate
    text idempotency_key UK
}

recipients {
    text id PK
    text user_id FK
    text name
    text country
    text currency
    text bank_account
}

merchants {
    text id PK
    text user_id FK
    text business_name
    text org_number UK
    text bank_account
    real fee_rate
    text qr_hmac_key
}

sessions {
    text id PK
```

```
    text user_id FK
    text token_hash
    text expires_at
    integer revoked
}

notifications {
    text id PK
    text user_id FK
    text type
    text title
    text body
    integer read
}

settings {
    text user_id PK
    text currency
    text language
    integer push_enabled
    integer email_enabled
}

exchange_rates {
    integer id PK
    text from_currency
    text to_currency
    real rate
}

cards {
    text id PK
    text user_id FK
    text type "virtual|physical"
    text last_four
    text status "active|frozen|cancelled"
}

spending_limits {
    text id PK
```

```
    text user_id FK
    text card_id FK
    text limit_type
    integer amount
}

audit_log {
    text id PK
    text user_id FK
    text action
    text resource_type
    text resource_id
    text ip_address
}

aml_alerts {
    text id PK
    text user_id FK
    text alert_type
    text severity "low|medium|high|critical"
    text transaction_id FK
    text status "open|investigating|resolved|escalated|filed"
}

str_reports {
    text id PK
    text user_id FK
    text alert_id FK
    text report_type
    text status "draft|submitted|acknowledged"
}

screening_results {
    text id PK
    text user_id FK
    text screening_type "pep|sanctions|adverse_media"
    text result "clear|match|potential_match|error"
}

consents {
```

```
        text id PK
        text user_id FK
        text consent_type
        integer granted
        text ip_address
    }

    data_access_requests {
        text id PK
        text user_id FK
        text request_type "export|erasure|rectification|restriction"
        text status "pending|processing|completed|rejected"
    }

    complaints {
        text id PK
        text user_id FK
        text category
        text subject
        text status "received|investigating|resolved|escalated"
    }

    rate_limits {
        text key PK
        integer count
        integer reset_at
    }
```

Domain Groupings

1. User Domain (4 tables)

Core identity, authentication, and preferences.

Table	Purpose	Record Growth
`users`	User accounts with KYC/AML fields, BankID identity	1 per registered user

Table	Purpose	Record Growth
`settings`	Per-user preferences (currency, language, notifications)	1 per user (1:1)
`sessions`	JWT session tracking with revocation support	Multiple per user, prunable
`notifications`	In-app notification delivery	High volume, prunable

Key relationships: `users` is the central entity. Every other user-scoped table references `users(id)` via foreign key. `settings` has a 1:1 relationship using `user_id` as its primary key.

2. Financial Domain (7 tables)

Transaction processing, bank account linkage, exchange rates, and payment cards.

Table	Purpose	Record Growth
`transactions`	All financial operations (remittance + QR payment)	High volume, append-only
`bank_accounts`	Linked bank accounts with cached AISP balance	Few per user
`recipients`	Saved remittance recipients	Few per user
`merchants`	Registered merchant profiles	1 per merchant user
`exchange_rates`	NOK-to-foreign currency rates	6 corridor records, updated periodically
`cards`	Virtual/physical payment cards (FUTURE, feature-flagged)	Few per user
`spending_limits`	Card spending limits (FUTURE)	Few per card

Key relationships: `transactions` polymorphically references either `recipients` (for remittances) or `merchants` (for QR payments) -- never both simultaneously. `bank_accounts.balance` is a cached read-only value from AISP, not a Drop-held balance.

3. KYC/AML Domain (3 tables)

Anti-money laundering monitoring and regulatory screening.

Table	Purpose	Record Growth
`aml_alerts`	Flagged suspicious transaction patterns	Event-driven, low volume
`str_reports`	Suspicious Transaction Reports filed with Økokrim	Rare, legally retained

Table	Purpose	Record Growth
`screening_results`	PEP/sanctions/adverse media screening results	Per user, periodic rescreens

Key relationships: `aml_alerts` links to a triggering `transaction`. `str_reports` escalates from an `aml_alert`. Both reference the `user` under investigation.

4. GDPR/Compliance Domain (3 tables)

Data subject rights, consent management, and complaint handling.

Table	Purpose	Record Growth
`consents`	GDPR consent records (terms, privacy, marketing, cookies)	Few per user
`data_access_requests`	DSAR tracking (export, erasure, rectification, restriction)	Rare
`complaints`	Customer complaints per Finansavtaleloven section 3-53	Low volume

Key relationships: All reference `users(id)`. Consent withdrawal triggers downstream processing (e.g., marketing opt-out).

5. System Domain (2 tables)

Operational infrastructure for audit trails and rate limiting.

Table	Purpose	Record Growth
`audit_log`	User action audit trail for compliance	Very high volume
`rate_limits`	IP-based rate limiting counters	Ephemeral, auto-cleaned

Key relationships: `audit_log` optionally references `users(id)` (some system events are unauthenticated). `rate_limits` is standalone with no foreign keys.

Data Classification

Each table is classified by sensitivity level for security controls, encryption, and access policies.

Classification uses the 5-level taxonomy defined in [security-architecture.md](): CRITICAL, RESTRICTED, CONFIDENTIAL, INTERNAL, PUBLIC.

Table	Classification	PII	Financial	Compliance	Rationale
`users`	RESTRICTED	Yes	No	Yes	Contains name, email, phone, DOB, national ID hash
`bank_accounts`	RESTRICTED	Yes	Yes	Yes	Bank account numbers, IBAN, cached balance
`transactions`	CONFIDENTIAL	No	Yes	Yes	Financial records, amounts, exchange rates
`recipients`	RESTRICTED	Yes	Yes	No	Names and foreign bank account numbers
`merchants`	CONFIDENTIAL	No	Yes	No	Business details, org numbers, bank accounts
`sessions`	INTERNAL	No	No	No	Token hashes enabling authentication bypass if leaked
`cards`	RESTRICTED	Yes	Yes	Yes	Card last-four, token refs, PINs (FUTURE)
`aml_alerts`	CONFIDENTIAL	No	No	Yes	Regulatory investigation data
`str_reports`	CONFIDENTIAL	No	No	Yes	Filed with Økokrim, legally protected
`screening_results`	CONFIDENTIAL	No	No	Yes	PEP/sanctions match data
`audit_log`	INTERNAL	Partial	No	Yes	IP addresses, user agents, action descriptions
`consents`	RESTRICTED	Partial	No	Yes	IP addresses, consent timestamps
`data_access_requests`	INTERNAL	No	No	Yes	DSAR metadata and download URLs
`complaints`	INTERNAL	No	No	Yes	User-submitted text content

Table	Classification	PII	Financial	Compliance	Rationale
`notifications`	INTERNAL	No	No	No	Display text, no sensitive content
`settings`	INTERNAL	No	No	No	UI preferences only
`exchange_rates`	PUBLIC	No	No	No	Public market data
`spending_limits`	INTERNAL	No	No	No	User-configured limits
`rate_limits`	INTERNAL	No	No	No	Ephemeral IP counters

Data Flow

Remittance Flow

```
sequenceDiagram
    participant U as User (Mobile/Web)
    participant API as Hono API / Next.js
    participant Auth as Auth Middleware
    participant DB as Database (SQLite/PostgreSQL)
    participant AISP as Open Banking AISP
    participant PISP as Open Banking PISP

    U->>API: POST /transactions/remittance
    API->>Auth: Verify JWT + session
    Auth->>DB: SELECT sessions WHERE token_hash = ? AND revoked = 0
    Auth-->>API: userId, role

    API->>DB: SELECT * FROM recipients WHERE id = ? AND user_id = ?
    DB-->>API: Recipient (country, currency, bank_account)

    API->>DB: SELECT rate FROM exchange_rates WHERE to_currency = ?
    DB-->>API: Exchange rate

    API->>DB: SELECT * FROM bank_accounts WHERE user_id = ? AND is_primary = 1
    DB-->>API: Bank account (balance check)
```

```
    Note over API,DB: Atomic transaction begins
    API->>DB: UPDATE bank_accounts SET balance = balance - ? WHERE balance >= ?
    API->>DB: INSERT INTO transactions (type='remittance', status='processing', ...)
    API->>DB: INSERT INTO audit_log (action='transaction.create', ...)
    API->>DB: INSERT INTO notifications (type='transaction', ...)
    Note over API,DB: Atomic transaction commits

    API->>PISP: Initiate payment from user's bank (production)
    API-->>U: 201 { transaction details, ETA }
```

QR Payment Flow

```
sequenceDiagram
    participant U as User (Mobile)
    participant API as Hono API / Next.js
    participant DB as Database
    participant M as Merchant

    U->>U: Scan QR code (drop://pay/{merchantId})
    U->>API: POST /transactions/qr-payment { merchantId, amount }
    API->>DB: Verify JWT session
    API->>DB: SELECT * FROM merchants WHERE id = ?
    DB-->>API: Merchant details (fee_rate, bank_account)

    API->>DB: SELECT * FROM bank_accounts WHERE user_id = ? AND is_primary = 1
    DB-->>API: Primary bank account

    Note over API,DB: Atomic transaction
    API->>DB: UPDATE bank_accounts SET balance = balance - (amount + fee)
    API->>DB: INSERT INTO transactions (type='qr_payment', status='completed')
    API->>DB: INSERT INTO audit_log (action='qr_payment.create')
    API->>DB: INSERT INTO notifications (title='Betaling registrert')
    Note over API,DB: Commit

    API-->>U: 201 { payment confirmation }
```

Caching Strategy

Data	Cache Location	TTL	Invalidation	Rationale
Exchange rates	`exchange_rates` table	Updated periodically (external feed in production)	Table update replaces rows	Rates change infrequently; per-request DB lookup is sufficient
Bank account balance	`bank_accounts.balance` column	`balance_synced_at` tracks freshness	Re-synced via AISP on dashboard load	Cached AISP read; Drop never modifies this value except through sync
User session validity	`sessions` table lookup	Until `expires_at`	Set `revoked = 1` on logout	Every authenticated request checks session table
Rate limit counters	`rate_limits` table	`reset_at` Unix timestamp (60s window)	Auto-cleaned every 100 rate limit checks	Expired entries deleted in `middleware/rate-limit.ts`
JWT payload	In-cookie (client-side)	7d (all clients)	Cookie cleared on logout, session revoked server-side	Stateless token; server validates against sessions table
Feature flags	In-memory (process)	Process lifetime	Restart or env var change	Read from environment variables at startup

No external cache layer (Redis/Memcached): At current scale, PostgreSQL 16 with Drizzle ORM handles the expected query volume without an external cache. A caching layer will be evaluated when query volume exceeds PostgreSQL connection pool capacity (max 20 connections per App Runner instance).

Data Access Layer (Drizzle ORM)

> **NOTE:** The dual-driver abstraction (`db.ts`, `USE_PG`) was removed per ADR-014 (2026-03-03). The data access layer is now Drizzle ORM exclusively.

The database access layer (`src/shared/db/schema.ts` + Drizzle ORM) provides type-safe access to PostgreSQL 16:

Pattern	How

SELECT queries	`db.select().from(table).where(...)`
Single row SELECT	`db.select().from(table).limit(1)`
INSERT/UPDATE/DELETE	`db.insert(table).values(...)`, `db.update()`, `db.delete()`
Upsert	`db.insert(table).values(...).onConflictDoUpdate(...)`
Atomic operations	`db.transaction(async (tx) => { ... })`
Row locking	`db.select().from(table).for('update')`
Raw SQL escape hatch	`db.execute(sql\`SELECT ...\`)`

Connection string: `DATABASE_URL=postgresql://...` (required in all environments).

Cross-References

- **Schema details:** DATABASE-SCHEMA.md
- **Database design rationale:** database-design.md
- **Migration strategy:** migration-strategy.md
- **Data lifecycle and GDPR:** data-lifecycle.md
- **Audit architecture:** audit-architecture.md
- **Indexing strategy:** indexing-strategy.md
- **API reference:** API-REFERENCE.md
- **Security architecture:** SECURITY-ARCHITECTURE.md
- **Authentication:** AUTHENTICATION.md