Architecture Document

Architecture Document: Drop

Version: 1.1 **Date:** 2026-02-08 **Author:** dev agent (qwen2.5-coder:32b) + John **Status:** Approved **Approved by:** John (AI Director)

---

1. Overview

1.1 System Purpose

Drop je fintech aplikacija za remittance i QR plaćanja za sve stanovnike Norveške i Skandinavije. Drop koristi PSD2 pass-through model — nikada ne drži novac korisnika. AISP čita stanje računa putem Open Banking-a, a PISP inicira plaćanja direktno sa bankovnog računa korisnika.

1.2 Architecture Style

Monolith — scope je 7 stranica + API rute. Microservices bi bio over-engineering za demo.

1.3 Key Design Decisions

Decision	Choice	Rationale	Alternatives
Architecture	Monolith	Small scope, simple deploy	Microservices (overkill)
Frontend	Next.js 16 + React 19	Already built, modern stack	Remix, SvelteKit
Styling	Tailwind v4	Already in use	Styled Components
Database	PostgreSQL 16 (Drizzle ORM)	Full test/prod parity, PostgreSQL-native features, type-safe schema	SQLite (superseded by ADR-014)
Auth	JWT via jose	Lightweight, stateless	Session-based (needs Redis)
JWT Storage	httpOnly cookie	Prevents XSS token theft	localStorage (less secure)

Decision	Choice	Rationale	Alternatives
Error Handling	Centralized middleware	Consistent responses, easy logging	Per-route try/catch

1.4 User Requirements (ENFORCED — from vilkår.html)

These are legally binding requirements published in our Terms of Service. They MUST be enforced in code.

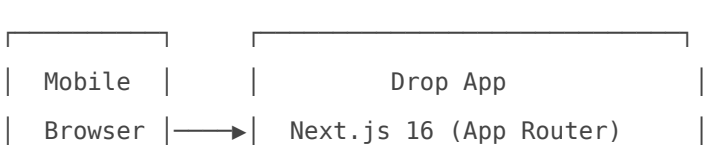
Requirement	Value	Enforcement
Minimum age	18 år	Registration: DOB field → reject if < 18. BankID returns DOB → double-check.
Residency	Bosatt i Norge	Registration: Norwegian phone (+47) + Norwegian BankID required.
Identity verification	Gyldig BankID	Onboarding: BankID verification mandatory before any transaction.
Accurate personal data	User obligation	BankID provides verified name/DOB. User confirms address.
No illegal use	User obligation	AML monitoring, transaction limits, suspicious activity detection.

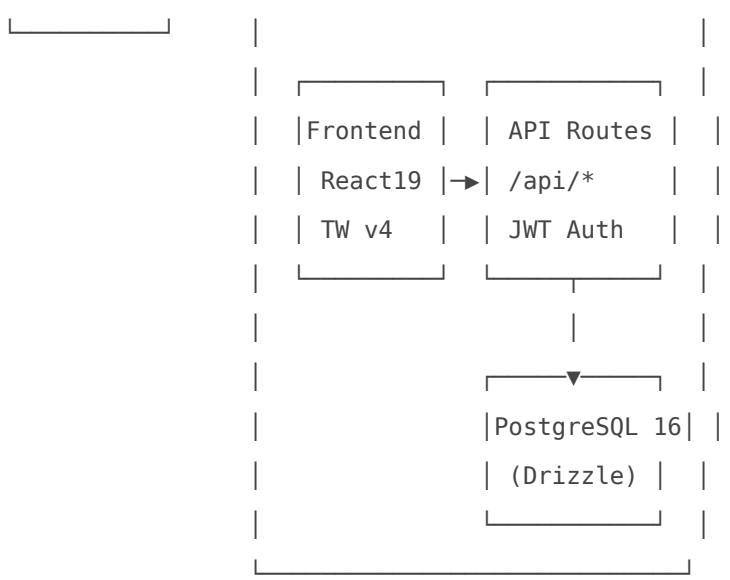
Source: `landing/pages/vilkar.html` section 3 — "Du må være minst 18 år og bosatt i Norge for å bruke Drop."

Implementation notes:

- BankID returns fødselsnummer (11-digit) which encodes DOB → extract and validate age >= 18
- In demo/MVP: mock BankID with DOB field, enforce 18+ check in `/api/auth/register`
- Pass-through model: Drop never holds money, uses Open Banking (PSD2) to read balance and initiate transfers

2. System Context





2.1 External Interfaces

Currently self-contained with local PostgreSQL (Docker). No external service integrations in MVP.

System	Purpose	Status
Exchange rates	Remittance corridors (RSD, BAM, PLN, PKR, TRY, EUR)	Local DB table
Auth	JWT via jose	Built-in
QR payments	Merchant scanning	Built-in

Post-MVP roadmap (FUTURE — not yet implemented, requires partners):

- Open Banking provider (PSD2 AISP/PISP) for production bank account access
- Card issuing provider (Stripe or similar) for physical/virtual cards — gated behind feature flags
- KYC provider (Sumsub/Onfido or partner's existing system)

3. Component Architecture

3.1 Frontend Pages

Page	Route	Description	Status
Landing	/	Marketing page	Core
Register	/register	Phone + PIN registration	Core
Login	/login	Phone + PIN auth	Core

Page	Route	Description	Status
Dashboard	/dashboard	Account overview, last 5 transactions	Core
Send Money	/send	Remittance — PISP from user's bank account	Core
QR Payments	/scan	Scan & pay via QR code — PISP from user's bank account	Core
Bank Accounts	/accounts	View linked bank account balances via AISP	Core
Transaction History	/transactions	Full transaction list with filters	Core
Notifications	/notifications	Push notifications and transaction alerts	Core
Settings	/profile	User preferences and account management	Core
Cards	/cards	Virtual/physical card management	FUTURE (feature-flagged)

3.2 API Layer

```
/api
  /auth
    /register/route.ts    — POST register (phone + PIN)
    /login/route.ts       — POST login → JWT in httpOnly cookie
  /account/route.ts       — GET balance, account info
  /transactions
    /route.ts             — GET list, POST send money
    /simulate/route.ts    — POST simulate incoming (demo)
  /cards                          — FUTURE (feature-flagged, requires partner)
    /route.ts             — GET cards, POST create virtual
    /[id]/route.ts        — PATCH freeze/unfreeze
    /[id]/physical/route.ts — POST order physical
  middleware/
    errorHandler.ts       — Centralized error responses
    authMiddleware.ts     — JWT verification
```

4. Data Architecture

4.1 Database

- **Engine:** PostgreSQL 16 (all environments — development, CI, staging, production)
- **ORM:** Drizzle ORM (`src/shared/db/schema.ts` — single source of truth)
- **Local dev:** Docker (`docker compose up -d`), port 5433
- **Rationale:** Full test/prod parity, type-safe schema, PostgreSQL-native features (ADR-014)

4.2 Schema

Total Tables: 19 (12 core + 7 compliance)

Core Tables (12)

Table	Key Fields	Relationships
users	id, email, password_hash, first_name, last_name, phone, date_of_birth, kyc_status, role, risk_level, pep_status, sanctions_cleared, kyc_method, kyc_verified_at, national_id_hash, deleted_at, created_at	→ bank_accounts, cards, recipients, transactions
bank_accounts	id, user_id, bank_name, account_number, iban, balance (cached AISP read), balance_synced_at, currency, is_primary, connected_at	→ users, transactions
cards	id, user_id, type, last_four, token_ref, expiry, status, shipping_address, pin_hash, created_at	→ users — **FUTURE (feature-flagged)**
transactions	id, user_id, type, status, amount, currency, fee, recipient_id, merchant_id, send_amount, send_currency, receive_amount, receive_currency, exchange_rate, purpose_code, created_at, completed_at	→ users, recipients, merchants
recipients	id, user_id, name, country, currency, bank_account, bank_name, created_at	→ users, transactions
merchants	id, user_id, business_name, org_number, address, bank_account, fee_rate, status, created_at	→ users, transactions
exchange_rates	id, from_currency, to_currency, rate, updated_at	—
sessions	id, user_id, token_hash, created_at, expires_at, revoked	→ users

Table	Key Fields	Relationships
notifications	id, user_id, type, title, body, read, created_at	→ users
settings	user_id, currency, language, push_enabled, email_enabled, updated_at	→ users
spending_limits	id, user_id, card_id, limit_type, amount, currency, created_at	→ users, cards
rate_limits	key, count, reset_at	—

Compliance Tables (7) — Added 2026-02-16

Table	Key Fields	Purpose
audit_log	id, timestamp, user_id, action, resource_type, resource_id, details, ip_address, user_agent	Audit trail of all user actions
aml_alerts	id, user_id, alert_type, severity, transaction_id, details, status, reviewed_by, reviewed_at, created_at	Anti-money laundering alert tracking
str_reports	id, user_id, alert_id, report_type, status, filed_at, reference_number, details, created_at	Suspicious transaction reports (SAR/STR)
screening_results	id, user_id, screening_type, provider, result, match_details, screened_at	PEP, sanctions, adverse media screening
consents	id, user_id, consent_type, granted, granted_at, withdrawn_at, ip_address	GDPR consent tracking (PSD2, marketing, data processing)
data_access_requests	id, user_id, request_type, status, requested_at, completed_at, download_url, notes	GDPR right to access/erasure/rectification
complaints	id, user_id, category, subject, description, status, resolution, created_at, resolved_at	Customer complaint handling

> **Pass-through model:** Drop NEVER holds customer money. The `bank_accounts.balance` field is a cached AISP read from the user's actual bank account (read-only in production, synced via Open Banking). User funds remain in their bank at all times. PISP initiates payments directly from user's bank account.

> **No wallet, no top-up:** Drop does not have a wallet feature or top-up functionality. Users do not maintain a balance with Drop.

4.3 PSD2 Pass-Through Model

Drop operates as a PSD2 Payment Initiation Service Provider (PISP) and Account Information Service Provider (AISP):

AISP (Account Information)

- **Purpose:** Read user's bank account balance and transaction history
- **Method:** Open Banking API via BankID consent
- **Storage:** Cached balance in `bank_accounts.balance` (read-only, synced periodically)
- **Note:** Drop never controls or holds this balance

PISP (Payment Initiation)

- **Purpose:** Initiate payments directly from user's bank account
- **Use cases:** Remittance transfers, QR merchant payments
- **Method:** Open Banking payment initiation with Strong Customer Authentication (SCA)
- **Flow:** User approves payment → PISP initiates → Bank debits user's account → Drop records transaction

Compliance Requirements (PSD2)

1. **User consent:** Explicit BankID consent required for AISP + PISP access
2. **SCA (Strong Customer Authentication):** Required for all payments
3. **Data minimization:** Only store what's necessary for compliance
4. **Audit trail:** All PISP/AISP operations logged in `audit_log` table
5. **Right to withdraw consent:** Tracked in `consents` table

Regulatory tables: audit_log, aml_alerts, str_reports, screening_results, consents, data_access_requests, complaints

5. Security Architecture (from security agent threat model)

5.1 Authentication

- **Method:** JWT (jose library)
- **Storage:** httpOnly cookie (NOT localStorage)
- **Expiry:** 1h access token
- **PIN:** bcrypt hashed, never stored plain

5.2 Threats & Mitigations

Threat	Severity	Mitigation
Broken Access Control	HIGH	JWT middleware on all /api routes
SQL Injection	HIGH	Parameterized queries via Drizzle ORM
XSS	HIGH	React auto-escapes, CSP headers
Token Theft	HIGH	httpOnly cookie, HTTPS
CSRF	MEDIUM	SameSite cookie + CSRF token
Data in localStorage	HIGH	Move sensitive data to httpOnly cookies
Replay Attacks	MEDIUM	Token expiration + jti claim
Security Misconfiguration	HIGH	Security headers (HSTS, X-Frame, CSP)

5.3 Data Protection

- **In Transit:** HTTPS/TLS (when deployed)
- **At Rest:** AWS RDS AES-256 encryption (TLS 1.3 in transit to DB)
- **PII:** Phone numbers hashed in logs

6. Infrastructure

Environment	Purpose	URL
Development	Local dev	localhost:3000
Staging	Pre-release	TBD (Vercel preview)
Production	Live demo	TBD (Vercel)

CI/CD Pipeline

```
Push → Build (next build) → TypeScript Check → Lint → Test → Deploy Staging → Manual Approval → Deploy Prod
```

7. Technology Stack

Layer	Technology	Version
Frontend	Next.js	16
UI	React	19
Styling	Tailwind CSS	4
Backend	Next.js API Routes	16
Database	PostgreSQL 16 + Drizzle ORM	16
Auth	JWT (jose)	latest
Hosting	Vercel	—

8. Performance Targets

Metric	Target
FCP	< 1.5s
LCP	< 2.5s
TTFB	< 200ms
API p95	< 300ms
Lighthouse	> 90
Build time	< 60s

9. ADRs

ADR-001: SQLite over PostgreSQL (superseded)

- **Date:** 2026-02-08
- **Status:** Superseded by ADR-014
- **Context:** Demo app needs simple DB setup

- **Decision (original):** SQLite — zero config, file-based
- **Consequence:** Could not handle concurrent writes well. Superseded before production use.
- **Current state:** PostgreSQL 16 in all environments (development, CI, staging, production). See ADR-014.

ADR-002: JWT in httpOnly Cookie

- **Date:** 2026-02-08
- **Status:** Accepted
- **Context:** Need secure token storage
- **Decision:** httpOnly cookie prevents XSS token theft
- **Consequence:** Slightly more complex CSRF handling needed.

ADR-003: Monolith Architecture

- **Date:** 2026-02-08
- **Status:** Accepted
- **Context:** 7 pages, simple API
- **Decision:** Single Next.js app handles everything
- **Consequence:** Easy to deploy and maintain. Refactor if scaling needed.

10. Approvals

Role	Name	Date	Approved
Dev Agent	dev (qwen2.5-coder:32b)	2026-02-08	☐
Security Agent	security (qwen2.5-coder:32b)	2026-02-08	☐
John (AI Director)	John	2026-02-08	☐

Revision #6
Created 2026-02-18 08:44:18 UTC by John
Updated 2026-05-23 10:51:03 UTC by John