# Drop Srbija

Serbian market fintech payment platform — architecture, compliance, and operations

# Architecture Overview

# Architecture Overview

## System Design

Drop Srbija is a fintech payment app for the Serbian market, enabling phone-based remittance and instant transfers via NBS IPS (Narodna Banka Srbije instant payment system).

**Core principle:** Money never touches Drop — it stays in user's bank account. Drop facilitates transfers only.

## Technology Stack

| Component | Choice | Rationale |
|---|---|---|
| **Language (Backend)** | Kotlin | ALAI standard 2026-03-17; full type safety + Ktor native support |
| **Framework** | Ktor 3.1.2 | Lightweight, async-first, Production-ready with netty transport |
| **Database** | PostgreSQL 16 | ALAI mandate; full ACID, timezone support, JSON operators |
| **ORM** | Exposed (Kotlin DSL) | Kotlin-native, composable, maps to SQL closely |
| **Migrations** | Flyway | Version control for schema; integration with Ktor startup |
| **Connection Pool** | HikariCP | Battle-tested, fast, connection reuse |
| **Auth** | JWT (HS256) | Stateless, easy to validate on backend + frontend |
| **Language (Frontend)** | TypeScript + Next.js 15 | React 19, server components, app router |
| **Styling** | Tailwind CSS 4 + shadcn/ui | ALAI standard; component library prebuilt |
| **Icons** | Lucide React | ALAI standard across all products |

| Component | Choice | Rationale |
|---|---|---|
| **Testing (Backend)** | Kotest + Testcontainers | Kotlin-native, containerized DB for integration tests |
| **Testing (Frontend)** | Vitest + Playwright | Unit + E2E, headless browser automation |

# Module Structure

```
DropSrbija/
├── backend/                    # Kotlin + Ktor 3.1.2
│   ├── src/main/kotlin/no/alai/dropsrbija/
│   │   ├── Application.kt      # Main entry point
│   │   ├── plugins/            # Ktor configuration
│   │   │   ├── Database.kt
│   │   │   ├── Routing.kt
│   │   │   ├── Authentication.kt
│   │   │   └── RateLimit.kt
│   │   ├── auth/               # JWT services
│   │   ├── models/             # Database tables (Exposed DSL)
│   │   └── modules/            # Business logic
│   │       ├── auth/           # Phone OTP flow
│   │       ├── user/           # User management
│   │       ├── transactions/   # Transaction handling
│   │       ├── recipients/     # Recipient management
│   │       └── ips/            # NBS IPS integration
│   └── src/main/resources/db/migration/  # Flyway migrations
└── frontend/                   # Next.js 15 + React 19
    └── src/app/
        ├── layout.tsx
        ├── page.tsx            # Landing page (Serbian)
        └── (app)/              # Authenticated app
```

# Database Schema (Migrations V1-V9)

## V1: Core Tables (Users, PhoneVerifications, Transactions, Recipients, NbsIpsLogs,

# Merchants, Settings)

**Users Table:**

- `id` — UUID (primary key)
- `phone` — +381XXXXXXXXX (unique)
- `first_name`, `last_name` — User identity
- `email` — optional
- `kyc_status` — pending | verified | rejected
- `role` — user | merchant | admin
- `phone_verified` — boolean
- `deleted_at` — soft delete support

**PhoneVerifications Table:**

- `id` — UUID (primary key)
- `phone` — +381XXXXXXXXX
- `otp` — 6-digit hash (bcrypt)
- `attempts` — failed attempts counter (max 5)
- `expires_at` — 10-minute expiry
- `verified_at` — timestamp of successful verification

**Transactions Table:**

- `id` — UUID (primary key)
- `user_id` — references users
- `type` — phone_transfer | qr_payment | bank_transfer
- `status` — processing | completed | failed
- `amount` — integer RSD
- `currency` — RSD | EUR | USD
- `recipient_phone` — +381XXXXXXXXX (if phone-based)
- `nbs_ips_id` — NBS transaction ID
- `nbs_ips_status` — PENDING | ACCEPTED | REJECTED | SETTLED

**NbsIpsLogs Table:**

- `id` — UUID (primary key)
- `transaction_id` — references transactions
- `request_type` — initiate | check_status | cancel
- `request_body` — JSON payload sent to NBS
- `response_body` — JSON response from NBS
- `response_status` — HTTP status code
- `error_message` — any error text

# V2: ISO 20022 Message Support

- Added `iso20022_message_id`, `iso20022_end_to_end_id` columns to `nbs_ips_logs`

# V3: Linked Bank Accounts

- New table: `linked_accounts` (user_id, bank_id, iban, account_name, verified_at)

# V4: Transaction Idempotency

- Added `idempotency_key_hash` (SHA-256 hash, unique constraint) to `transactions`

# V5: KYC Sessions

- New table: `kyc_sessions` (user_id, status, jmbg_verified, biometric_data, completed_at)

# V6: National ID Support

- Added `national_id_hash`, `national_id_encrypted` to `users` table

# V7: AML Flags

- New table: `aml_flags` (user_id, flag_type, reason, severity, created_by, resolved_at)

# V8: User Disclosure Acknowledgment

- Added `disclosure_acknowledged_at` to `users` table

# V9: Complaints System

- New table: `complaints` (user_id, category, description, status, resolved_at)

# Ports & Services

| Service | Port (Local) | Description |
|---|---|---|
| PostgreSQL | 5434 | Drop Srbija database (separate from Drop Norway) |
| Redis | 6380 | Rate limiting and caching |
| Backend API | 3003 | Ktor HTTP server |

| Service | Port (Local) | Description |
|---|---|---|
| Frontend | 3000 | Next.js dev server |

# Payment Flow Diagram

```
flowchart LR
    A[User Enters Phone + OTP] -->|POST /auth/verify-otp| B{OTP Valid?}
    B -->|Yes| C[JWT Issued]
    B -->|No| D[Increment Attempts]
    C --> E[User Initiates Payment]
    E -->|POST /v1/ips/initiate| F[Create Transaction Record]
    F --> G[Call NbsIpsService]
    G -->|ISO 20022 Message| H[NBS IPS API]
    H -->|ACCP/RJCT| I[Log Response to NbsIpsLogs]
    I -->|Async| J{Status?}
    J -->|ACCP| K[Update Transaction: completed]
    J -->|RJCT| L[Update Transaction: failed]
    K --> M[Recipient Receives RSD]
    L --> N[Notify User of Failure]
```

# Branch Model

Feature branches are created off the previous feature branch, following a linear progression:

```
feat/drop-srbija-models (T1)
    ↓
feat/drop-srbija-otp (T2)
    ↓
feat/drop-srbija-jwt (T3)
    ↓
feat/drop-srbija-ips (T6)
    ↓
...
feat/drop-srbija-disclosure-complaints (T13)
    ↓
feat/drop-srbija-docs (T29 — this documentation)
```

**Merge strategy:** TBD — will be established when merging to main/production branch.

# Key Architectural Decisions

1. **Bank Partner Adapter Pattern (Task 6):** NBS IPS is ISO 20022 mTLS, not REST; single NBS direct integration impossible without bank license. Solution: Partner with licensed Serbian bank that provides IPS gateway access.
2. **Agent Model Year 1 → Own PI License Year 2:** Start as registered agent under Serbian bank (Article 24 of Payment Services Law), then pursue own Payment Institution license once proven.
3. **Idempotency via SHA-256 Hash (Task 4):** Store hash of idempotency key, not the raw key itself, to minimize PII surface area.
4. **Lock Threshold 5 vs Spec 6:** Industry-standard practice is 5 failed OTP attempts before account lock, rather than 6 mentioned in some specs.

# Security Considerations

1. **PII Encryption:**
   - National ID stored encrypted (KMS key rotation)
   - Phone numbers indexed as hashes for lookups
   - Passwords NOT stored (auth via OTP only)
2. **OTP Security:**
   - 6-digit, 10-minute expiry
   - Max 5 attempts per verification
   - Rate limit: 3 requests per minute per phone
   - SMS delivery via Twilio (TLS, no logging)
3. **JWT Validation:**
   - HS256 with strong secret (64+ bytes)
   - Issuer: "dropsrbija-api", Audience: "dropsrbija"
   - httpOnly cookies (frontend sets automatic)
   - 24-hour expiry (refresh token TBD)
4. **NBS IPS Integration:**
   - mTLS for NBS API calls
   - Request signing (HMAC-SHA256)
   - Audit log for all NBS transactions
   - Retry logic with exponential backoff
5. **Rate Limiting:**
   - Global: 1000 req/min per IP
   - Auth: 10 OTP requests/hour per phone
   - Transaction: 50 transactions/hour per user
   - NBS API: Respect rate limits from NBS docs

# Monitoring & Observability

## Logs

- Structured JSON logs (Logback)
- Sensitive fields redacted (phone last 4 digits only)
- Log level: INFO (debug only in dev)

## Metrics

- Transaction latency (p50, p95, p99)
- NBS API response time
- OTP verification success rate
- Error rates by endpoint

## Alerts

- NBS IPS unavailable (down 5 min → page on-call)
- Database connection pool exhausted
- Unusual transaction volume

---

**Last Updated:** 2026-04-16
**Status:** Architecture scaffold complete — ready for Phase 2 (Frontend Onboarding UI)

# Regulatory Compliance

# Regulatory Compliance

## Overview

Drop Srbija operates within the Serbian regulatory framework for payment services, data protection, and anti-money laundering. This document outlines the key legal requirements and compliance obligations.

# NBS (Narodna Banka Srbije) — Payment Institution Licensing

## Licensing Path

**Year 1 (Recommended):** Operate as **registered agent** under Article 24 of the Law on Payment Services through partnership with a licensed Serbian bank.

**Year 2:** Pursue **Payment Institution (PI) license** directly from NBS once market validation is proven.

## Payment Institution License Requirements

If pursuing own PI license:

**Minimum Capital Requirement:** EUR 125,000 (or RSD equivalent)

**Required Documents for NBS Authorization:**

1. **Business Plan (3-year projection)**
   - Market analysis (Serbian remittance + domestic transfer market)
   - Revenue model (transaction fees, FX spreads)
   - Risk assessment (operational, financial, fraud, AML/CFT)
   - Financial projections (P&L, balance sheet, cash flow)
2. **AML/CFT Programme**

- Customer due diligence procedures
- Transaction monitoring rules (thresholds, alerts)
- Sanctions screening process
- Suspicious transaction reporting protocol
- USPNFT eUprava integration for STR filing

3. **IT Security and Business Continuity**
   - System architecture diagram
   - Data protection measures (encryption, access control, audit logs)
   - Incident response plan
   - Disaster recovery and backup procedures
   - Penetration testing schedule
4. **Organizational Structure**
   - Org chart with key personnel
   - CVs and credentials of directors and compliance officers
   - Proof of fit-and-proper assessment (criminal record check, financial solvency)
   - Compliance Officer appointment (AML/CFT specialist)
   - Data Protection Officer (DPO) appointment
5. **Proof of Share Capital**
   - Bank statement showing EUR 125,000 deposited
   - Shareholder agreements
   - Proof of source of funds

**Timeline:** 9-14 months from application submission to license issuance (optimistic).

**NBS Contact:**

- Email: platne.institucije@nbs.rs
- Phone: +381 11 3027 100
- Address: Nemanjina 17, 11000 Belgrade, Serbia

**Legal Reference:** nbs-pisp-license-requirements.md

---

# ZPNFTM (Zakon o spre?avanju pranja novca) — AML/CFT Framework

## Law on Prevention of Money Laundering and Terrorist Financing

**Official Gazette:** 113/2017, 91/2019, 153/2020

**Regulatory Authority:** Administration for the Prevention of Money Laundering (APML)

# Key Obligations

1. **Customer Due Diligence (CDD)**
   - Verify identity using government-issued ID (JMBG validation)
   - Collect name, address, date of birth, national ID number
   - Verify beneficial ownership (for legal entities)
   - Enhanced due diligence for high-risk customers (PEPs, high-value transactions)
2. **Transaction Monitoring**
   - Threshold: RSD 15,000 (~EUR 130) for identification requirement
   - High-value threshold: EUR 15,000 for enhanced monitoring
   - Pattern detection: Structuring, unusual activity, cross-border remittances
3. **Suspicious Transaction Reporting (STR)**
   - Report to APML via **USPNFT eUprava portal** (https://euprava.gov.rs)
   - No de minimis threshold — any suspicious activity must be reported
   - Prohibition on tipping off the customer
4. **Record Retention**
   - 5 years minimum for transaction data (Article 60)
   - 10 years for high-risk transactions
   - Must be readily accessible for APML audits
5. **Sanctions Screening**
   - Check all customers and transactions against:
     - **UN Consolidated List** (https://www.un.org/securitycouncil/sanctions/list)
     - **EU Restrictive Measures** (https://sanctionsmap.eu/)
     - **Serbian Government Sanctions** (Official Gazette)
   - **NOTE:** There is NO "NBS SDN list" — Serbia does not maintain a separate sanctions list. Use UN + EU + Serbian government sources only.
6. **PEP Screening**
   - Politically Exposed Persons (domestic and foreign)
   - Family members and close associates
   - Enhanced due diligence required

# USPNFT eUprava Integration

**Portal:** https://euprava.gov.rs/usluge/uspnft
**What it does:** Electronic submission of Suspicious Transaction Reports (STRs) to APML

**Drop Srbija Implementation:**

- Compliance Officer has eUprava account with STR permissions
- STR submission within 3 business days of detection
- System generates draft STR from `aml_flags` table entries
- Manual review by Compliance Officer before submission

**Legal Reference:** framework-contract-payment-users-sr.md

# ZZPL (Zakon o zaštiti podataka o li?nosti) — Data Protection

## Law on Personal Data Protection

**Official Gazette:** 87/2018

**Regulatory Authority:** Poverenik za informacije od javnog značaja i zaštitu podataka o ličnosti (Commissioner for Information of Public Importance and Personal Data Protection)

**Contact:**

- Email: office@poverenik.rs
- Phone: +381 11 3408 900
- Address: Bulevar kralja Aleksandra 15, 11000 Belgrade, Serbia

# Key Principles

Serbia's ZZPL is modeled on GDPR but with some differences. Key provisions:

1. **Legal Bases for Processing (Article 12)**
   - **(a) Consent:** Marketing communications, optional features
   - **(b) Contract Performance:** Account creation, payment processing
   - **(c) Legal Obligation:** AML/CFT compliance, incident reporting to NBS
   - **(d) Vital Interests:** Fraud prevention
   - **(e) Public Interest:** Not applicable for Drop Srbija
   - **(f) Legitimate Interest:** Product improvements, analytics
2. **Data Categories Processed by Drop Srbija**
   - **Identity:** Name, JMBG (national ID), date of birth, address
   - **Contact:** Phone number, email
   - **Financial:** IBAN, transaction history
   - **Biometric (if KYC implemented):** Facial recognition data for identity verification
   - **Device:** IP address, device ID, app version
3. **Data Subject Rights**
   - **Right of Access (Article 23):** User can request copy of all personal data
   - **Right to Rectification (Article 24):** User can correct inaccurate data
   - **Right to Erasure (Article 25):** "Right to be forgotten" — must delete unless legal obligation to retain (AML 5-year retention overrides)

- **Right to Data Portability (Article 27):** Export data in machine-readable format (JSON)
- **Right to Object (Article 28):** User can object to processing based on legitimate interest

4. **Data Breach Notification**
   - **To Poverenik:** Within 72 hours of breach discovery (Article 54)
   - **To Users:** Without undue delay if high risk to rights and freedoms
   - **Breach Definition:** Unauthorized access, data loss, data exposure, ransomware, etc.
5. **Data Retention Policy**
   - **Active users:** Retain while account is active
   - **Inactive users:** After 2 years of inactivity, anonymize or delete (unless AML retention applies)
   - **AML data:** 5 years from transaction date (overrides erasure requests)
   - **Marketing consent:** Until withdrawn
6. **Cross-Border Data Transfers**
   - Drop Srbija infrastructure: AWS EU (Frankfurt or Stockholm region)
   - Serbia is an EU candidate country — adequacy decision expected during EU accession
   - Current status: Transfers to EU/EEA allowed under ZZPL Article 63 (adequate protection)

# Data Protection Impact Assessment (DPIA)

**Required for:** Biometric KYC verification (facial recognition for JMBG validation)

**Document:** dpia-kyc-biometric.md

**Key Findings:**

- High risk: Biometric data is special category (Article 17)
- Mitigation: Encryption, access control, retention limits (delete after verification)
- Legal basis: Legal obligation (AML/CFT) + contract performance
- Approved by: ALAI Lexicon (awaiting Serbian DPO review)

# Privacy Policy

**Location:** privacy-policy-sr.md

**Publication Requirements:**

- Must be in Serbian (official version)
- Published on Drop Srbija website before launch
- In-app display during onboarding (user must accept before account creation)

- Updated whenever processing activities change

**Content Includes:**

- Data controller: Drop Srbija d.o.o. (legal entity TBD)
- Legal bases for each processing activity
- Data categories and retention periods
- Third-party processors (AWS, SMS gateway)
- User rights and how to exercise them
- DPO contact details

---

# Incident Reporting — NBS and Poverenik

## Three-Track Notification System

Drop Srbija has three parallel incident notification obligations:

### Track 1: NBS Initial Notification (Within 4 Hours)

**Trigger:** Significant operational or security incident affecting payment services

**Examples:**

- Service outage >2 hours
- Cyberattack or data breach
- Fraudulent transaction pattern
- NBS IPS integration failure
- Critical system failure

**Contact:** platne.institucije@nbs.rs, +381 11 3027 100

**Format:** Brief email alert with:

- Incident type and time of detection
- Preliminary impact assessment
- Immediate actions taken
- Estimated resolution time

### Track 2: NBS Detailed Report (Within 72 Hours)

**Follow-up to Track 1** with comprehensive analysis:

**Required Content:**

- Root cause analysis
- Full impact assessment (customers affected, transaction volume, financial loss)
- Timeline of events
- Preventive measures implemented
- Lessons learned

**Format:** PDF document, 5-15 pages, Serbian language

**Submission:** Email to platne.institucije@nbs.rs

## Track 3: Poverenik Data Breach Notification (Within 72 Hours)

**Trigger:** Personal data breach (unauthorized access, data exposure, data loss)

**Examples:**

- Database leak
- Phishing attack exposing customer data
- Employee unauthorized access
- Ransomware encryption of customer records

**Contact:** office@poverenik.rs, +381 11 3408 900

**Format:** Breach notification form (available on Poverenik website)

**Required Content:**

- Nature of breach (type of data, number of individuals affected)
- Likely consequences
- Measures taken to mitigate
- DPO contact details

**IMPORTANT:** This is a separate notification from NBS reporting. Personal data breaches must be reported to BOTH NBS (if affecting payment services) AND Poverenik (for data protection compliance).

# User Notification

**Trigger:** Data breach likely to result in high risk to user rights and freedoms

**Timeline:** Without undue delay (typically within 72 hours)

**Method:** SMS + in-app notification + email

**Template:** incident-notification-procedure.md contains user notification templates in Serbian.

---

# Serbian Bank Partnership

Drop Srbija Year 1 strategy relies on partnership with a licensed Serbian bank to access NBS IPS as a registered agent.

# Legal Framework: Article 24 (Agent Registration)

Under the Law on Payment Services, payment institutions and banks can appoint **registered agents** to provide payment services on their behalf.

**Requirements:**

- Agent must be registered with NBS
- Principal (bank) remains responsible for agent's actions
- Written agent agreement required
- Agent must comply with all AML/CFT obligations

# Bank Partnership Pitch

**Target Banks:** Raiffeisen Banka, Erste Bank, Banca Intesa, OTP Banka, Mobi Banka (digital-first)

**Value Proposition:**

- Drop brings new digital-native customers to the bank
- Increased IPS transaction volume
- Revenue share on transaction fees
- Co-branded offering (optional)

**What Drop Needs:**

- IPS gateway API access
- Registered agent status (Article 24)
- Technical integration support
- Bank account for settlement

**Document:** serbian-bank-partnership-pitch.md

**Timeline:** 6-7 months from first contact to launch (optimistic)

# Sanctions Sources (CORRECTED)

**Drop Srbija screens against three sources:**

1. **UN Consolidated List**
   - URL: https://www.un.org/securitycouncil/sanctions/list
   - Format: XML/PDF download
   - Update frequency: Weekly
2. **EU Restrictive Measures (Sanctions Map)**
   - URL: https://sanctionsmap.eu/
   - Format: JSON API available
   - Update frequency: Daily
3. **Serbian Government Sanctions**
   - Source: Official Gazette of the Republic of Serbia
   - Implementation: Serbia adopts UN sanctions, occasionally imposes additional measures
   - No centralized API — manual monitoring required

**CRITICAL CORRECTION:** There is NO "NBS SDN list." NBS does not maintain a separate sanctions list. Earlier references to "NBS SDN" were an error. Use only the three sources above.

---

# Compliance Roles

Drop Srbija must appoint the following officers before launch:

| Role | Responsibility | Qualifications |
|---|---|---|
| **Data Protection Officer (DPO)** | ZZPL compliance, data subject requests, breach notification | Legal/IT background, ZZPL expertise |
| **AML/CFT Compliance Officer** | Transaction monitoring, STR filing, sanctions screening | ACAMS certification or equivalent, Serbian language |
| **Risk Officer** | Operational and financial risk management | Can be CTO initially, fintech risk experience |

**Hiring Status:** TBD — awaiting Drop Srbija d.o.o. incorporation

---

# Legal Document Index

All legal and compliance documents are located in `/comms/decisions/`:

- nbs-pisp-license-requirements.md — NBS PI license application guide
- serbian-bank-partnership-pitch.md — Bank partnership proposal template
- privacy-policy-sr.md — ZZPL-compliant privacy policy (Serbian)
- privacy-policy-drop-srbija-draft.md — Privacy policy (English draft)
- incident-notification-procedure.md — Three-track incident reporting protocol
- dpia-kyc-biometric.md — DPIA for biometric KYC
- framework-contract-payment-users-sr.md — Framework contract for payment service users
- recommendation-year1-vs-year2.md — Finverge analysis: Agent model vs PI license
- serbian-banks-api-landscape.md — Serbian banking API research
- nbs-pi-license-application-package.md — Complete NBS application package guide

---

**DISCLAIMER:**

All documents are DRAFT status and require Serbian legal counsel review before use. Drop Srbija must engage a Serbian law firm specializing in fintech/payment services for:

1. Validation of all legal and regulatory statements
2. Review and finalization of all documents
3. Entity structure and licensing strategy advice
4. Drafting final agreements and regulatory submissions

**Budget Estimate:** EUR 5,000-10,000 for initial legal review.

---

**Last Updated:** 2026-04-16
**Next Review:** After Serbian legal counsel engagement

# Developer Onboarding

# Developer Onboarding

## Welcome to Drop Srbija

This guide will help you set up the Drop Srbija development environment and get started contributing to the codebase.

## Prerequisites

Ensure you have the following installed before proceeding:

| Tool | Version | Installation |
|---|---|---|
| **JDK** | 21 | `brew install openjdk@21` (macOS) or download from Adoptium |
| **Gradle** | 9.x | Included via Gradle Wrapper ( `./gradlew`) — no manual install needed |
| **Node.js** | 20+ | `brew install node@20` or nvm |
| **Docker Desktop** | Latest | Download from Docker |
| **Git** | Latest | `brew install git` (macOS) or pre-installed on most systems |
| **PostgreSQL Client** | 16+ | `brew install postgresql@16` (for `psql` CLI, optional) |

**Verify installations:**

```
java -version        # Should show OpenJDK 21
node -v              # Should show v20.x or higher
docker --version     # Should show Docker version 20+
git --version        # Any recent version
```

# First-Time Setup

## 1. Clone the Repository

```
cd ~/ALAI/products
git clone <repository-url> DropSrbija
cd DropSrbija
```

**Note:** Repository URL will be provided by ALAI (likely GitHub private repo).

## 2. Environment Variables

Copy the example environment file and customize:

```
cp .env.example .env
```

**Edit `.env` with your local settings:**

```
# Database
DATABASE_URL=postgresql://dropsrbija:dev_only_not_a_secret@localhost:5434/dropsrbija_dev
DATABASE_USER=dropsrbija
DATABASE_PASSWORD=dev_only_not_a_secret

# API
PORT=3003
JWT_SECRET=dev_only_not_a_secret_jwt_replace_in_production
JWT_EXPIRY_SECONDS=86400

# NBS IPS (stub for local dev)
NBS_IPS_ENDPOINT=https://ips.nbs.rs/api/v1
NBS_IPS_API_KEY=dev-stub-key

# Redis
REDIS_URL=redis://localhost:6380

# Frontend
NEXT_PUBLIC_API_URL=http://localhost:3003
NEXT_PUBLIC_APP_LANGUAGE=sr
```

**CRITICAL:** Never commit `.env` to version control. It's already in `.gitignore`.

# 3. Start Docker Services

Drop Srbija uses Docker Compose for local development infrastructure (PostgreSQL + Redis):

```
docker compose up -d
```

**What this does:**

- Starts PostgreSQL 16 on port `5434`
- Starts Redis on port `6380`
- Creates `dropsrbija_dev` database
- Runs in background (`-d` flag)

**Verify services are running:**

```
docker compose ps
```

Expected output:

```
NAME                    SERVICE     STATUS      PORTS
dropsrbija-postgres-1   postgres    running     0.0.0.0:5434->5432/tcp
dropsrbija-redis-1      redis       running     0.0.0.0:6380->6379/tcp
```

# 4. Build Backend

Navigate to the backend directory and build:

```
cd backend
./gradlew build
```

**What this does:**

- Downloads dependencies (Ktor, Exposed, Flyway, Kotest, etc.)
- Compiles Kotlin source code
- Runs Flyway migrations (creates all database tables)
- Runs unit tests

**First build takes 2-5 minutes.** Subsequent builds are faster (Gradle caches dependencies).

# 5. Seed Database (Optional)

For local development, you may want sample data:

```
./gradlew seedDatabase
```

**What this creates:**

- 3 test users (`+381601234567`, `+381602345678`, `+381603456789`)
- 5 sample transactions
- 2 recipients per user

**NOTE:** Seeding is optional and only for local dev. Production databases should NEVER be seeded.

# 6. Run Backend

Start the Ktor server:

```
./gradlew run
```

**Expected output:**

```
[main] INFO  Application - Application started in 0.234s
[main] INFO  Application - Responding at http://0.0.0.0:3003
[main] INFO  Application - Database migrations applied successfully
```

**Verify backend is running:**

```
curl http://localhost:3003/health
```

Expected response:

```
{
  "status": "healthy",
  "version": "0.1.0",
  "timestamp": 1713280800000
}
```

# 7. Run Frontend

Open a new terminal tab/window:

```
cd frontend
npm install          # First time only
```

```
npm run dev
```

**Frontend runs on:** http://localhost:3000

**Expected output:**

```
▲ Next.js 15.0.0
- Local:        http://localhost:3000
- Ready in 1.2s
```

# Running Tests

## Backend Tests (Kotest + Testcontainers)

**Unit Tests:**

```
cd backend
./gradlew test
```

**Integration Tests:**

Integration tests use Testcontainers (spins up real PostgreSQL in Docker):

```
RUN_INTEGRATION_TESTS=true ./gradlew integrationTest
```

**Why separate?** Integration tests are slower (~30s) because they start/stop Docker containers. Unit tests run in <5s.

**Test Reports:**

After running tests, view HTML report at:

```
backend/build/reports/tests/test/index.html
```

## Frontend Tests (Vitest + Playwright)

**Unit Tests (Vitest):**

```
cd frontend
npm run test
```

**E2E Tests (Playwright):**

```
cd frontend
npx playwright install     # First time only (installs browsers)
npx playwright test
```

**View Playwright Report:**

```
npx playwright show-report
```

**Headless vs Headed:**

By default, Playwright runs headless (no browser window). To see the browser:

```
npx playwright test --headed
```

# Branch Model

Drop Srbija follows a **linear feature branch strategy**:

```
main (protected)
    ↓
feat/drop-srbija-models (T1)
    ↓
feat/drop-srbija-otp (T2)
    ↓
feat/drop-srbija-jwt (T3)
    ↓
feat/drop-srbija-ips (T6)
    ↓
...
feat/drop-srbija-disclosure-complaints (T13)
    ↓
feat/drop-srbija-docs (T29)
```

**Rules:**

1. **Create new feature branch off the previous feature branch**, not off `main`
2. Branch naming: `feat/drop-srbija-<task-name>` (lowercase, kebab-case)
3. One feature per branch (corresponds to one Mission Control task)
4. Commit frequently with descriptive messages

5. **DO NOT** force push or rebase after pushing to remote

**Example workflow:**

```
# You're on feat/drop-srbija-otp and just finished Task 2
git checkout -b feat/drop-srbija-jwt
# Make changes for Task 3
git add .
git commit -m "Add JWT service and authentication plugin"
git push -u origin feat/drop-srbija-jwt
```

**Merge Strategy:** TBD — will be defined when first feature is ready for production merge.

# Common Commands (Makefile)

Drop Srbija provides a `Makefile` for common tasks:

| Command | Description |
|---|---|
| `make start` | Start all services (Docker + backend + frontend) |
| `make stop` | Stop all services |
| `make test` | Run all tests (backend + frontend) |
| `make lint` | Run linters (ktlint + eslint) |
| `make clean` | Clean build artifacts |
| `make logs` | Tail Docker logs |
| `make db-shell` | Open psql shell to database |

**Example:**

```
make start    # Starts everything
make test     # Run all tests
make stop     # Stop everything
```

# Environment Variables Checklist

Before running the app, verify these are set in `.env`:

**Backend:**

- [ ] `DATABASE_URL` — PostgreSQL connection string
- [ ] `DATABASE_USER` — Database username
- [ ] `DATABASE_PASSWORD` — Database password
- [ ] `PORT` — API port (default: 3003)
- [ ] `JWT_SECRET` — Secret for JWT signing (64+ random characters in production)
- [ ] `JWT_EXPIRY_SECONDS` — JWT lifetime (default: 86400 = 24 hours)
- [ ] `NBS_IPS_ENDPOINT` — NBS IPS API URL (stub for dev)
- [ ] `NBS_IPS_API_KEY` — NBS API key (stub for dev)

**Frontend:**

- [ ] `NEXT_PUBLIC_API_URL` — Backend API URL (default: http://localhost:3003)
- [ ] `NEXT_PUBLIC_APP_LANGUAGE` — UI language (default: sr for Serbian)

**Optional (Production only):**

- [ ] `REDIS_URL` — Redis connection string (for rate limiting)
- [ ] `TWILIO_ACCOUNT_SID` — Twilio account ID (for SMS OTP)
- [ ] `TWILIO_AUTH_TOKEN` — Twilio auth token
- [ ] `TWILIO_PHONE_NUMBER` — Twilio sender phone number

# Database Migrations (Flyway)

Drop Srbija uses Flyway for version-controlled schema changes.

**Migration files location:**

```
backend/src/main/resources/db/migration/
├── V1__init.sql
├── V2__nbs_ips_logs_iso20022.sql
├── V3__linked_accounts.sql
├── V4__transaction_idempotency.sql
├── V5__kyc_sessions.sql
├── V6__users_jmbg.sql
├── V7__aml_flags.sql
├── V8__disclosure_acknowledged.sql
└── V9__complaints.sql
```

**Naming convention:** `V<number>__<description>.sql` (double underscore after version number)

**Migrations run automatically** when you start the backend with `./gradlew run`.

**Manual migration:**

```
cd backend
./gradlew flywayMigrate
```

**Check migration status:**

```
./gradlew flywayInfo
```

**CRITICAL:** Never modify an already-applied migration. Create a new migration file instead.

# Connecting to Database

**Via psql (command line):**

```
psql -h localhost -p 5434 -U dropsrbija -d dropsrbija_dev
```

Password: `dev_only_not_a_secret`

**Via GUI tools:**

Use any PostgreSQL client (DBeaver, pgAdmin, TablePlus, etc.):

- **Host:** localhost
- **Port:** 5434
- **Database:** dropsrbija_dev
- **Username:** dropsrbija
- **Password:** dev_only_not_a_secret

# Code Style & Linting

## Backend (Kotlin)

Drop Srbija uses **ktlint** for Kotlin code formatting.

**Auto-format code:**

```
cd backend
./gradlew ktlintFormat
```

**Check for style violations:**

```
./gradlew ktlintCheck
```

**ktlint runs automatically** during `./gradlew build`.

# Frontend (TypeScript)

Drop Srbija uses **ESLint** + **Prettier** for TypeScript/React formatting.

**Lint code:**

```
cd frontend
npm run lint
```

**Auto-fix issues:**

```
npm run lint:fix
```

**Format code:**

```
npm run format
```

# Troubleshooting

## Port Already in Use

**Symptom:** `Address already in use: bind` error when starting backend

**Solution:**

```
# Find process using port 3003
lsof -i :3003

# Kill process
kill -9 <PID>
```

## Database Connection Refused

**Symptom:** `Connection to localhost:5434 refused` error

**Solution:**

```
# Check if Docker is running
docker compose ps

# Restart Docker services
docker compose down
docker compose up -d
```

# Flyway Migration Failed

**Symptom:** `Migration V3__linked_accounts.sql failed` error

**Solution:**

```
# Check Flyway status
cd backend
./gradlew flywayInfo

# Repair (if checksum mismatch)
./gradlew flywayRepair

# Manual rollback (use with caution)
psql -h localhost -p 5434 -U dropsrbija -d dropsrbija_dev
DELETE FROM flyway_schema_history WHERE version = '3';
```

# Gradle Build Slow

**Symptom:** `./gradlew build` takes >5 minutes

**Solution:**

```
# Increase Gradle heap size
export GRADLE_OPTS="-Xmx2g"

# Use Gradle daemon (should be on by default)
echo "org.gradle.daemon=true" >> ~/.gradle/gradle.properties
```

# Frontend "Module not found" Error

**Symptom:** `Cannot find module '@/components/ui/button'`

**Solution:**

```
cd frontend
rm -rf node_modules package-lock.json
npm install
```

# Next Steps

Now that your environment is set up:

1. **Read the Architecture Overview** — 01-architecture-overview.md
2. **Review the Regulatory Compliance guide** — 02-regulatory-compliance.md
3. **Check the Runbook for NBS IPS outage handling** — 04-runbook-nbs-ips-outage.md
4. **Explore the Decision Log** — 05-decision-log.md
5. **Pick a task from Mission Control** — `node ~/system/tools/mc.js list --product drop-srbija`

# Getting Help

**Technical Questions:**

- CodeCraft team (backend): Petter Graff, Martin Kleppmann
- Vizu team (frontend): Brad Frost, Lea Verou

**Compliance/Legal Questions:**

- Lexicon (ALAI Legal & Compliance)

**General Questions:**

- John (AI Director) — orchestrator for all ALAI operations

---

**Welcome aboard!** □□

---

**Last Updated:** 2026-04-16
**Next Review:** When onboarding feedback is received from first new developer

# Runbook: NBS IPS Outage

# Runbook: NBS IPS Outage

## Purpose

This runbook provides step-by-step procedures for detecting, triaging, and responding to NBS IPS (Narodna Banka Srbije Instant Payment System) outages or degraded performance.

## Trigger

Any of the following conditions indicate potential NBS IPS outage:

1. **High rejection rate:** >10% of payment initiation requests returning `RJCT` status
2. **Timeout spike:** >5 consecutive timeout errors (>30s response time)
3. **HTTP 5xx errors:** NBS IPS API returning 500/502/503/504
4. **NBS status page alert:** Official communication from NBS about system maintenance or outage

## Preconditions

Before following this runbook, ensure:

- ☐ You have access to production logs (CloudWatch, Datadog, or local Docker logs)
- ☐ You have access to NBS IPS status page: https://www.nbs.rs (check payment system availability)
- ☐ You have NBS contact details: platne.institucije@nbs.rs, +381 11 3027 100
- ☐ You have access to Drop Srbija admin dashboard (to pause outbound payments if needed)
- ☐ You have access to status page publishing tool (to notify users)

## Step-by-Step Response

# Step 1: Verify the Issue (ETA: 2 minutes)

**Check Docker logs for NBS IPS errors:**

```
docker logs dropsrbija-api | grep NbsIpsLog | tail -50
```

**Look for patterns:**

- Multiple `RJCT` (rejection) responses with error code `AM05` (duplicate submission)
- Multiple `RJCT` with error code `TECH` (technical error)
- Timeout messages: `java.net.SocketTimeoutException: Read timed out`
- HTTP 503 Service Unavailable
- HTTP 500 Internal Server Error

**Example of normal log:**

```
2026-04-16T10:15:32Z [NbsIpsLog] transaction_id=abc123 request_type=initiate
response_status=200 nbs_ips_status=ACCP
```

**Example of outage log:**

```
2026-04-16T10:45:12Z [NbsIpsLog] transaction_id=def456 request_type=initiate
response_status=503 error_message="Service Temporarily Unavailable"
2026-04-16T10:45:23Z [NbsIpsLog] transaction_id=ghi789 request_type=initiate
response_status=500 error_message="Internal Server Error"
```

**Check metrics dashboard** (if available):

- Transaction success rate (should be >95%)
- Average NBS IPS response time (should be <2s)
- Error rate by HTTP status code

# Step 2: Check NBS IPS Status Page (ETA: 1 minute)

**Official NBS status page:**

1. Go to https://www.nbs.rs
2. Navigate to: **Payment Systems → IPS → System Availability**
3. Check for announcements:
   - Scheduled maintenance windows
   - Incident notifications
   - System degradation alerts

**If NBS confirms outage:**

- Note the estimated resolution time (ERT)
- Proceed to Step 4 (Customer Communication)

**If NBS shows "All Systems Operational":**

- Outage may be isolated to Drop Srbija's connection
- Proceed to Step 3 (Technical Diagnosis)

# Step 3: Technical Diagnosis (ETA: 5 minutes)

**Possible causes of isolated failures:**

1. **Network issue between Drop Srbija and NBS:**
   - Check VPN/VPC connectivity (if applicable)
   - Verify mTLS certificates haven't expired
   - Test network path: `curl -I https://ips.nbs.rs`
2. **Rate limiting:**
   - NBS may throttle requests if Drop exceeds transaction quota
   - Check `nbs_ips_logs` table for HTTP 429 (Too Many Requests)
   - Solution: Implement exponential backoff (already in adapter)
3. **Authentication failure:**
   - mTLS client certificate may have expired
   - API key rotation (if NBS uses API keys)
   - Check for HTTP 401/403 errors in logs
4. **Partner bank integration issue:**
   - If Drop operates as agent under Article 24, outage may be at partner bank's IPS gateway
   - Contact bank technical support: [Partner Bank Support Number]

**Run diagnostic test transaction:**

```
# Send test payment (100 RSD to known-good recipient)
curl -X POST http://localhost:3003/v1/ips/initiate \
  -H "Authorization: Bearer <admin_jwt>" \
  -H "Content-Type: application/json" \
  -d '{
    "recipientPhone": "+381601234567",
    "amount": 100,
    "description": "IPS diagnostic test"
  }'
```

**Expected responses:**

- **Success:** `{"transactionId": "...", "status": "PENDING"}` → NBS IPS is working
- **Timeout:** No response after 30s → Network/connectivity issue
- **RJCT:** `{"status": "failed", "error": "TECH"}` → NBS technical error
- **HTTP 503:** NBS IPS is down

# Step 4: Customer Communication (ETA: 3 minutes)

**If outage is confirmed (NBS or Drop technical issue):**

**1. Post status page update** (https://status.dropsrbija.rs or in-app banner):

**Serbian template:**

```
□□Problemi sa trenutnim plaćanjima

NBS instant plaćanja su trenutno nedostupna zbog [razloga].
Radimo na rešavanju problema.

Očekivano vreme povratka: [ETA ili "u najkraćem roku"]

Vaša sredstva su sigurna. Pokušajte ponovo za nekoliko minuta.

Ažurirano: [timestamp]
```

**English translation (for reference):**

```
□□Issues with instant payments

NBS instant payments are currently unavailable due to [reason].
We are working on resolving the issue.

Expected resolution time: [ETA or "as soon as possible"]

Your funds are safe. Please try again in a few minutes.

Updated: [timestamp]
```

**2. Send SMS to active users** (optional, for extended outages >30 min):

**SMS template:**

```
Drop Srbija: Instant plaćanja trenutno nedostupna zbog problema sa NBS sistemom. Vaša sredstva su sigurna. Pokušajte ponovo za 30 min. Info: dropsrbija.rs/status
```

**3. Email to high-value users** (optional, for outages >2 hours):

Subject: `Obaveštenje o trenutnim problemima sa plaćanjima`

Body: [See incident-notification-procedure.md for full email template]

# Step 5: Escalate to NBS (if needed) (ETA: 5 minutes)

**When to escalate:**

- Outage duration >15 minutes AND NBS status page shows "operational"
- You suspect issue is specific to Drop Srbija's integration
- Multiple banks reporting similar issues (check Serbian fintech community channels)

**NBS Contact:**

- **Email:** platne.institucije@nbs.rs
- **Phone:** +381 11 3027 100 (business hours: 08:00-16:00 CET)
- **Emergency Phone:** [TBD — request during onboarding]

**Escalation email template:**

```
Subject: IPS Payment Failures – Drop Srbija (Urgent)

Poštovani,

Prijavljujemo tehničke probleme sa IPS plaćanjima preko Drop Srbija platforme.

Simptomi:
- Vreme početka: [timestamp]
- Procenat neuspelih transakcija: [X%]
- HTTP status kodovi: [500/503/timeout]
- Broj pogođenih korisnika: [Y]

Dijagnostika:
- NBS status stranica pokazuje "operativno"
- mTLS sertifikati validni do: [expiry date]
```

```
- Test transakcije vraćaju: [error message]

Molimo za hitnu pomoć u dijagnostici problema.

Kontakt: [your name], [phone], [email]

Drop Srbija d.o.o.
[Company registration details]
```

# Step 6: Implement Workaround (if available) (ETA: 10 minutes)

**Workaround Option 1: Switch to Alternate Bank Adapter**

If Drop has partnerships with multiple banks:

```
# Update environment variable (requires deployment)
export BANK_PARTNER=alternate_bank_id

# Restart API
docker compose restart api
```

**Workaround Option 2: Queue Transactions for Retry**

Backend already implements exponential backoff retry for failed transactions:

- First retry: 30 seconds
- Second retry: 2 minutes
- Third retry: 10 minutes
- After 3 failures: Mark transaction as `failed`, user receives notification

**No manual intervention needed** — adapter handles retries automatically.

**Workaround Option 3: Pause Outbound Payments**

If outage is prolonged (>2 hours) and retries are causing cascading failures:

```
# Via admin API (requires admin JWT)
curl -X POST http://localhost:3003/admin/payments/pause \
  -H "Authorization: Bearer <admin_jwt>" \
  -d '{"reason": "NBS IPS outage", "estimatedResumptionTime": "2026-04-16T14:00:00Z"}'
```

**Effect:**

- New payment requests return HTTP 503 with message: "Plaćanja su privremeno nedostupna"
- Existing pending transactions continue retry attempts
- Users see in-app banner: "Trenutno ne primamo nova plaćanja"

**Resume payments:**

```
curl -X POST http://localhost:3003/admin/payments/resume \
  -H "Authorization: Bearer <admin_jwt>"
```

# Step 7: Monitor Recovery (ETA: Ongoing)

**Once NBS IPS is back online:**

1. **Check transaction backlog:**

```
# Count transactions in "processing" state
psql -h localhost -p 5434 -U dropsrbija -d dropsrbija_prod -c \
  "SELECT COUNT(*) FROM transactions WHERE status = 'processing' AND created_at > NOW() -
INTERVAL '2 hours';"
```

2. **Verify retry processing:**

Backend retries failed transactions automatically. Monitor logs:

```
docker logs dropsrbija-api | grep "RetryProcessor" | tail -20
```

Expected output:

```
[RetryProcessor] Retrying transaction abc123 (attempt 1/3)
[NbsIpsLog] transaction_id=abc123 response_status=200 nbs_ips_status=ACCP
[RetryProcessor] Transaction abc123 succeeded on retry
```

3. **Update status page:**

```
 Problemi rešeni

NBS instant plaćanja su ponovo dostupna. Sva odložena plaćanja će biti procesirana automatski.

Hvala na strpljenju.
```

```
Ažurirano: [timestamp]
```

4. **Post-incident review** (within 24 hours):

- Total downtime duration
- Number of affected transactions
- Number of users impacted
- Root cause (NBS outage vs Drop technical issue)
- Lessons learned
- Action items (e.g., multi-bank redundancy, better monitoring)

# Step 8: Report to NBS (if required) (ETA: 72 hours)

**If outage meets NBS incident reporting criteria:**

- Duration >2 hours, OR
- > 1000 failed transactions, OR
- Security/data breach involved

**Follow incident notification procedure:**

See incident-notification-procedure.md for full 3-track reporting protocol:

- **Track 1:** NBS initial notification (within 4 hours)
- **Track 2:** NBS detailed report (within 72 hours)
- **Track 3:** Poverenik data breach notification (if applicable)

# Expected Outcome

**Success criteria:**

- □ Outage detected within 5 minutes of occurrence
- □ Root cause identified (NBS outage vs Drop technical issue)
- □ Users notified within 15 minutes (status page update)
- □ NBS contacted (if Drop-specific issue)
- □ Workaround implemented (if available)
- □ Service restored (or ETA communicated)
- □ Post-incident review completed

**Metrics to track:**

- **MTTD (Mean Time to Detect):** <5 minutes
- **MTTR (Mean Time to Resolve):** <30 minutes for Drop issues, variable for NBS outages
- **User notification latency:** <15 minutes

# Escalation Path

**Level 1 (On-call engineer):** Follow Steps 1-6 above

**Level 2 (Technical Lead):** If unresolved after 30 minutes

- Contact: Petter Graff (CodeCraft, petter-graff@alai.no)
- Decide: Implement workaround, escalate to NBS, or wait for NBS resolution

**Level 3 (CEO):** If outage >2 hours OR customer impact >10,000 users

- Contact: Alem Basic (alem@alai.no, +47 404 74 251)
- Decide: Public communication strategy, regulatory notification (NBS + Poverenik)

# Post-Outage Actions

**Mandatory:**

1. **Update runbook** — If new failure mode discovered, add to Step 3 diagnostic checklist
2. **Improve monitoring** — Add alert for specific error pattern that triggered outage
3. **Document lessons learned** — Add entry to Decision Log

**Optional (if pattern recurs):**

4. **Implement multi-bank redundancy** — Partner with 2+ banks for IPS access
5. **Pre-queue transactions** — Buffer transactions locally during known NBS maintenance windows
6. **Enhanced monitoring** — Set up synthetic transaction every 5 minutes to detect outages faster

# Related Documents

- Incident Notification Procedure — Full NBS/Poverenik reporting protocol
- Architecture Overview — NBS IPS integration design

- Decision Log — Historical incident post-mortems

---

**Last Updated:** 2026-04-16
**Next Review:** After first real NBS IPS outage (to validate runbook effectiveness)

# Decision Log

# Decision Log

## Purpose

This document records all significant architectural, technical, and regulatory decisions made during Drop Srbija development. Each entry includes context, decision, rationale, and alternatives considered.

## Format

Each decision follows this structure:

- **Date:** When decision was made
- **Decision ID:** Unique identifier (task number if applicable)
- **Title:** Brief description
- **Context:** Why this decision was needed
- **Decision:** What was decided
- **Rationale:** Why this option was chosen
- **Alternatives Considered:** What other options were evaluated
- **Consequences:** Trade-offs and implications
- **Status:** Active | Superseded | Deprecated

---

## Decisions

### D1: Bank Partner Adapter Pattern (Task 6)

**Date:** 2026-04-16
**Decision ID:** T6
**Status:** Active

**Context:**

Drop Srbija needs to integrate with NBS IPS (Narodna Banka Srbije Instant Payment System) to facilitate instant RSD transfers. NBS IPS uses ISO 20022 messaging over mTLS, not a REST API. Direct integration with NBS IPS requires a banking license or Payment Institution (PI) license from NBS.

**Decision:**

Implement a **bank partner adapter pattern** where Drop Srbija partners with a licensed Serbian bank that provides IPS gateway access. Drop operates as a **registered agent** under Article 24 of the Law on Payment Services.

**Rationale:**

1. **Faster time-to-market:** Agent registration takes 2-3 months vs 9-14 months for own PI license
2. **Lower upfront capital:** No EUR 125,000 capital requirement
3. **Reduced regulatory burden:** Bank partner handles NBS compliance, Drop focuses on customer experience
4. **Market validation:** Prove product-market fit before committing to full PI license

**Alternatives Considered:**

1. **Apply for PI license immediately:**
   - **Pros:** Full control, higher margins, regulatory independence
   - **Cons:** 9-14 month timeline, EUR 125,000 capital, extensive regulatory overhead
   - **Why rejected:** Too slow and capital-intensive for MVP phase
2. **Use third-party payment aggregator:**
   - **Pros:** Fastest integration (1-2 months)
   - **Cons:** High transaction fees (2-3%), less control over user experience, aggregator can change terms
   - **Why rejected:** Unsustainable economics, loss of differentiation
3. **Build without IPS (bank transfers only):**
   - **Pros:** Simpler integration, no licensing requirements
   - **Cons:** Transfers take 1-2 business days, poor user experience vs competitors
   - **Why rejected:** Instant payments are core value proposition

**Consequences:**

- ✅ Enables Year 1 launch with minimal regulatory overhead
- ✅ Drop can focus on product development, not licensing bureaucracy
- ❌ Bank partner takes 0.3-0.5% per transaction (reduces Drop's margin)
- ❌ Dependency on bank's IPS uptime and support quality
- ⚠ Transition to own PI license in Year 2 requires customer migration (account linking, re-KYC)

**Related Documents:**

- serbian-bank-partnership-pitch.md
- recommendation-year1-vs-year2.md

---

# D2: Agent Model Year 1 ? Own PI License Year 2

**Date:** 2026-04-16
**Decision ID:** Finverge Recommendation
**Status:** Active

**Context:**

After evaluating regulatory pathways, Drop Srbija must choose between:

1. Permanent agent model under bank partnership
2. Temporary agent model, transition to own PI license after market validation

**Decision:**

Operate as **registered agent Year 1**, then apply for **own Payment Institution license Year 2** once product-market fit is proven (target: 10,000+ active users, RSD 100M+ monthly transaction volume).

**Rationale:**

1. **De-risk upfront investment:** Avoid EUR 125,000 capital requirement until revenue is proven
2. **Regulatory learning:** Understand NBS compliance requirements while under bank's supervision
3. **Flexibility:** Can pivot business model or market positioning in Year 1 without sunk regulatory costs
4. **Stronger PI application:** NBS PI license approval more likely if applicant has proven track record

**Alternatives Considered:**

1. **Permanent agent model:**
   - **Pros:** No licensing costs ever, bank handles compliance
   - **Cons:** Permanent 0.3-0.5% fee to bank, limited negotiation leverage, bank can terminate partnership
   - **Why rejected:** Long-term economics don't work; Drop needs own license for sustainability
2. **PI license from Day 1:**

- **Pros:** Full control, no bank dependency, better margins
- **Cons:** 9-14 month delay to launch, EUR 125,000 upfront, high regulatory burden before proving PMF
- **Why rejected:** Too risky to commit capital and time before validating demand

**Consequences:**

- □ Minimize upfront capital and regulatory risk
- □ Learn NBS compliance requirements in Year 1
- □ Must re-KYC customers during transition (unless bank agreement allows customer transfer)
- □ Must maintain dual integrations during migration period
- ⚠ Year 2 timeline depends on NBS application backlog (9-14 months from submission)

**Trigger for Year 2 Transition:**

Start PI license application when Drop Srbija achieves:

- 10,000+ monthly active users, AND
- RSD 100M+ monthly transaction volume, AND
- Profitability at agent model margins (revenue > bank fees + operating costs)

**Related Documents:**

- recommendation-year1-vs-year2.md
- nbs-pisp-license-requirements.md

---

# D3: MTS Banka ? Banka Poštanska Štedionica (2021 Rebrand)

**Date:** 2026-04-16
**Decision ID:** Research Finding
**Status:** Historical Context

**Context:**

Early Drop Srbija research identified **MTS Banka** as a potential partner due to Telekom Srbija ownership and phone-to-IBAN synergy. However, MTS Banka ceased to exist as a standalone brand in 2021.

**Decision:**

Target **Banka Poštanska Štedionica** for partnership pitch, noting its legacy as MTS Banka and continued Telekom Srbija subsidiary status.

**Rationale:**

1. **Telekom Srbija ownership intact:** BPS is still a subsidiary of Telekom Srbija
2. **Phone-to-IBAN synergy preserved:** BPS has access to Telekom customer phone numbers for IBAN lookup
3. **Digital transformation appetite:** BPS is investing in digital banking (mobile app launched 2022)
4. **NBS IPS participant:** BPS is an active IPS participant

**Alternatives Considered:**

1. **Raiffeisen Banka:**
   - **Pros:** Largest bank by assets, strong digital banking, proven fintech partnerships
   - **Cons:** May have high partnership fees, slow decision-making (Austrian parent bureaucracy)
2. **Mobi Banka (digital-first bank):**
   - **Pros:** Agile, tech-forward, faster partnership negotiation
   - **Cons:** Smaller balance sheet, less brand trust, unknown IPS reliability

**Consequences:**

- □ Leverages existing Telekom Srbija phone database for phone-to-IBAN lookup
- □ Aligns with BPS digital transformation strategy
- □ BPS may lack fintech partnership experience compared to larger banks
- ⚠ BPS may require exclusivity clause (no partnerships with competing banks)

**Related Documents:**

- serbian-banks-api-landscape.md

---

# D4: Idempotency via SHA-256 Hash Only (No Raw Key Storage)

**Date:** 2026-04-16
**Decision ID:** T4
**Status:** Active

**Context:**

To prevent duplicate transaction submissions (e.g., user double-clicks "Pay" button), Drop Srbija must implement idempotency. Standard approach is to store a unique `idempotency_key` (UUID or user-generated string) in the database with a unique constraint.

**Decision:**

Store only the **SHA-256 hash of the idempotency key**, not the raw key itself. Add `idempotency_key_hash` column to `transactions` table with unique constraint.

**Rationale:**

1. **Privacy by design:** Raw idempotency key might contain PII (e.g., client-side key format: `user123-2026-04-16-transfer`)
2. **Minimize PII surface area:** Hash is irreversible; even if database is breached, attacker cannot recover original key
3. **ZZPL compliance:** Storing hashes instead of raw data aligns with data minimization principle (Article 5)
4. **Functional equivalence:** Idempotency only requires uniqueness check, not key retrieval

**Alternatives Considered:**

1. **Store raw idempotency key:**
   - **Pros:** Easier debugging (can see original key in logs)
   - **Cons:** PII risk, ZZPL non-compliance if key contains user data
   - **Why rejected:** Privacy risk outweighs debugging convenience
2. **No idempotency (rely on client-side debouncing):**
   - **Pros:** Simpler backend
   - **Cons:** Client-side debouncing unreliable (network issues, multiple devices)
   - **Why rejected:** High risk of duplicate charges, poor user experience

**Consequences:**

- ✅ ZZPL-compliant, minimal PII storage
- ✅ Prevents duplicate transactions
- ❌ Cannot retrieve original idempotency key for debugging (hash is one-way)
- ⚠️ Client must send same idempotency key for retries (backend cannot generate it)

**Implementation:**

```
// In TransactionService.kt
val idempotencyKeyHash = MessageDigest.getInstance("SHA-256")
    .digest(request.idempotencyKey.toByteArray())
    .joinToString("") { "%02x".format(it) }

// Check for duplicate
val existingTx = Transactions.select {
    Transactions.idempotencyKeyHash eq idempotencyKeyHash
}.singleOrNull()

if (existingTx != null) {
```

```
    return existingTx  // Return existing transaction, do not create new one
}
```

**Related Documents:**

- Migration: `V4__transaction_idempotency.sql`

---

# D5: OTP Lock Threshold: 5 Failed Attempts (Not 6)

**Date:** 2026-04-16
**Decision ID:** T2
**Status:** Active

**Context:**

Drop Srbija uses phone OTP for authentication. To prevent brute-force attacks, failed attempts must be limited. Some specs reference 6 attempts, industry standard is 5.

**Decision:**

Lock account (or require CAPTCHA) after **5 failed OTP attempts**, not 6.

**Rationale:**

1. **Industry standard:** Most banks and fintech apps use 5 attempts (Google, PayPal, Venmo, Cash App)
2. **Security vs UX balance:** 5 attempts is generous for legitimate users (allows for 4 typos), while preventing brute-force (6-digit OTP = 1M combinations, 5 attempts = 0.0005% success rate)
3. **Consistency:** ALAI's other products (Drop Norway, Bilko) use 5 attempts

**Alternatives Considered:**

1. **6 failed attempts:**
   - **Pros:** Slightly more user-friendly (one extra chance)
   - **Cons:** Marginal UX benefit, inconsistent with industry standard
   - **Why rejected:** 5 is already generous, no strong reason to deviate
2. **3 failed attempts:**
   - **Pros:** More secure (even harder to brute-force)
   - **Cons:** Too strict, frustrates legitimate users (e.g., mistyping OTP on small keyboard)
   - **Why rejected:** Poor UX, excessive security for low-risk authentication

**Consequences:**

- ✅ Aligns with industry best practices
- ✅ Prevents brute-force while allowing genuine user errors
- ✅ Users who fail 5 times must request new OTP (adds friction)
- ⚠ Support team must handle "locked out" users (reset mechanism needed)

**Implementation:**

```
// In PhoneOtpService.kt
if (verification.attempts >= 5) {
    throw IllegalStateException("Too many failed attempts. Request a new OTP.")
}
```

**Related Documents:**

- Test coverage: `PhoneOtpServiceTest.kt` (Task 20)

---

# D6: No Separate "Pending" Table for Transactions

**Date:** 2026-04-16
**Decision ID:** Architecture Decision
**Status:** Active

**Context:**

Some payment systems use a two-table approach:

1. `pending_transactions` (temporary, deleted after settlement)
2. `completed_transactions` (permanent, for historical records)

**Decision:**

Use a **single `transactions` table** with a `status` column (`processing | completed | failed`).

**Rationale:**

1. **Simplicity:** One source of truth for all transactions
2. **Auditability:** Full transaction lifecycle in one table (easier for compliance audits)
3. **Query performance:** PostgreSQL handles 1M+ rows efficiently with indexes; no need for table partitioning at MVP scale
4. **Retry logic:** Easier to implement (update status, don't move between tables)

**Alternatives Considered:**

1. **Two-table approach (pending + completed):**
   - **Pros:** Smaller "hot" table (faster queries on active transactions)
   - **Cons:** Complexity (must move records between tables), harder to audit full transaction history
   - **Why rejected:** Premature optimization; PostgreSQL can handle Drop's Year 1 volume in single table
2. **Event sourcing (immutable transaction events):**
   - **Pros:** Full audit trail, supports complex state machines
   - **Cons:** Overkill for MVP, steep learning curve, slower reads (must replay events)
   - **Why rejected:** Too complex for current requirements; revisit if/when Drop builds advanced fraud detection

**Consequences:**

- □ Simpler codebase, easier debugging
- □ Full transaction history in one query
- □ Table grows indefinitely (must implement archival policy in future)
- ⚠ If volume exceeds 10M+ transactions, may need table partitioning (by month)

**Related Documents:**

- Schema: `V1__init.sql` (transactions table)

---

# D7: Serbian Language UI (sr-RS Locale)

**Date:** 2026-04-16
**Decision ID:** Product Requirement
**Status:** Active

**Context:**

Drop Srbija targets Serbian market. UI language must be Serbian (sr-RS), but codebase must support internationalization (i18n) for potential future expansion (Bosnia, Croatia, Montenegro).

**Decision:**

- **Primary UI language:** Serbian (sr-RS)
- **Fallback language:** English (en-US) for error messages and developer logs
- **i18n library:** next-intl (Next.js 15 recommended)
- **Translation keys location:** `frontend/src/locales/sr.json`

**Rationale:**

1. **User trust:** Financial apps must be in local language (research shows 87% of Balkan users abandon apps not in their language)

2. **Regulatory requirement:** ZZPL and NBS mandate that user-facing documents (privacy policy, terms of service) are in Serbian
3. **Future-proof:** i18n structure allows adding Bosnian/Croatian/Montenegrin later (Serbo-Croatian mutual intelligibility)

**Alternatives Considered:**

1. **English-only:**
   - **Pros:** Easier development (no translation management)
   - **Cons:** Severely limits user adoption, non-compliant with ZZPL
   - **Why rejected:** Not viable for consumer fintech in Serbia
2. **Bilingual (Serbian + English):**
   - **Pros:** Appeals to expats and international users
   - **Cons:** Adds complexity, doubles translation effort, most users don't need it
   - **Why rejected:** Year 1 focus is local Serbian users; add English in Year 2 if data shows demand

**Consequences:**

- ✅ Maximizes user adoption in Serbian market
- ✅ ZZPL and NBS compliant
- ✅ All developers must work with Serbian text (use translation keys, not hardcoded strings)
- ⚠ Translation workflow needed (developer writes key → translator provides Serbian text)

**Implementation:**

```
// In app/layout.tsx
import { NextIntlClientProvider } from 'next-intl';
import sr from '@/locales/sr.json';

export default function RootLayout({ children }: { children: React.ReactNode }) {
  return (
    <html lang="sr">
      <body>
        <NextIntlClientProvider locale="sr" messages={sr}>
          {children}
        </NextIntlClientProvider>
      </body>
    </html>
  );
}
```

**Translation key example:**

```
// frontend/src/locales/sr.json
{
  "auth.otp.title": "Unesite OTP kod",
  "auth.otp.description": "Poslali smo vam 6-cifreni kod na {phone}",
  "auth.otp.submit": "Potvrdi",
  "auth.otp.resend": "Pošalji ponovo"
}
```

**Related Documents:**

- UI mockups: `frontend/src/app/(app)/page.tsx`

---

# D8: PostgreSQL 16 Only (No SQLite)

**Date:** 2026-04-16
**Decision ID:** ALAI Standard (2026-03-17)
**Status:** Active

**Context:**

Some developers prefer SQLite for local development (simpler setup, no Docker required). However, PostgreSQL and SQLite have different SQL dialects and feature sets.

**Decision:**

Use **PostgreSQL 16 for both local development and production**. No SQLite.

**Rationale:**

1. **ALAI mandate:** All ALAI products use PostgreSQL (CEO decision 2026-03-17)
2. **Dev-prod parity:** Eliminates "works on my machine" issues due to SQL dialect differences
3. **Feature requirements:** Drop Srbija uses PostgreSQL-specific features (JSONB operators, `TIMESTAMP WITH TIME ZONE`, full-text search)
4. **Test reliability:** Integration tests use Testcontainers (real PostgreSQL in Docker), ensuring 100% parity

**Alternatives Considered:**

1. **SQLite for dev, PostgreSQL for production:**
   - **Pros:** Faster local setup (no Docker)
   - **Cons:** SQL dialect differences cause bugs, harder to debug production issues locally
   - **Why rejected:** ALAI standard prohibits SQLite; dev-prod parity is critical
2. **MySQL/MariaDB:**

- **Pros:** Widely supported, familiar to many developers
- **Cons:** Weaker JSON support, timezone handling issues, not ALAI standard
- **Why rejected:** PostgreSQL is ALAI standard; no reason to deviate

**Consequences:**

- □ 100% dev-prod parity
- □ Leverages PostgreSQL advanced features (JSONB, RLS, full-text search)
- □ Requires Docker for local development (adds setup step)
- ⚠ Developers must learn PostgreSQL-specific SQL (not transferable to SQLite/MySQL)

**Related Documents:**

- ALAI Tech Stack Standard
- Docker setup: `docker-compose.yml`

---

# Future Decisions (Pending)

## FD1: Merge Strategy for Linear Feature Branches

**Status:** Pending
**Context:** Drop Srbija uses linear feature branch model (feat/A → feat/B → feat/C). Need to decide merge strategy when first feature is production-ready.

**Options:**

1. **Squash and merge:** Collapse all commits into one
2. **Rebase and merge:** Keep commits, rewrite history to be linear
3. **Merge commit:** Preserve branch history

**Recommendation:** TBD (pending first production merge)

---

## FD2: Multi-Bank Redundancy Strategy

**Status:** Pending
**Context:** Year 1 relies on single bank partner for IPS access. If bank's IPS gateway goes down, Drop is offline.

**Options:**

1. **Partner with 2+ banks:** Automatic failover if primary bank's IPS is down
2. **Pre-queue transactions:** Buffer locally during outages, process when bank is back
3. **Accept single point of failure Year 1:** Add redundancy in Year 2 after own PI license

**Recommendation:** TBD (pending first bank partnership negotiations)

---

# Change Log

| Date | Decision ID | Change |
|---|---|---|
| 2026-04-16 | D1-D8 | Initial decision log created |

---

**Last Updated:** 2026-04-16
**Next Review:** After first major architectural change or regulatory update