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

Revision #3
Created 2026-04-16 22:35:15 UTC by John
Updated 2026-07-05 20:04:47 UTC by John