Services & Middleware

Services

Drop External Services

> Source: `src/drop-app/src/lib/services/`

Overview

Drop uses a **PSD2 pass-through model** — it never holds customer money. AISP reads bank balances via Open Banking, PISP initiates payments from the user's own bank account.

Drop integrates with external service providers. Each service has a different readiness level — see status tags below.

> **Legend:** `[PRODUCTION]` = real SDK, production-ready. `[MOCK/DEV]` = mock only, NOT connected to real APIs. `[PLANNED]` = future roadmap. `[DEPRECATED]` = no longer the chosen provider.

Service mode is controlled by `NEXT_PUBLIC_SERVICE_MODE` env var (default: `mock`).

Source: `services/index.ts:21-30`

```
export const config = {
  mode: (process.env.NEXT_PUBLIC_SERVICE_MODE || "mock") as "mock" | "production",
  endpoints: {
    sumsub: process.env.SUMSUB_API_URL || "https://api.sumsub.com",
  },
};
```

> **Note on Swan:** Swan was previously listed as the Open Banking provider but has been deprecated. The pass-through PSD2 model will use a different AISP/PISP provider (TBD).

> **Note on Stripe:** Card issuing is a future feature gated behind feature flags. No Stripe SDK is integrated — only a mock file exists.
>
> **Note on Vipps/Nets:** Sometimes mentioned in business discussions but have ZERO code in the codebase.

Swan — Open Banking / PSD2 Provider [DEPRECATED]

> ⚠ DEPRECATED: Swan is no longer the planned Open Banking provider. Mock code remains but will be removed.

File: `services/mock-swan.ts` **Production docs:** https://docs.swan.io/ **Status:** DEPRECATED mock — no production integration, no contract, no API keys.

Interfaces

Interface	Description
`SwanAccount`	Bank account with IBAN, BIC, balance, status
`SwanTransaction`	SEPA credit/debit with status tracking

Functions

Function	Signature	Description
`createAccount`	`(userId) → SwanAccount`	Create new bank account with IBAN
`getAccount`	`(accountId) → SwanAccount | null`	Retrieve account details
`getBalance`	`(accountId) → {available, pending}`	Get balance breakdown
`initiateTransfer`	`({fromAccountId, toIban, amount, ...}) → SwanTransaction`	Initiate SEPA credit transfer
`simulateIncoming`	`({toAccountId, amount, fromIban}) → SwanTransaction`	Simulate incoming transfer
`getTransactions`	`(accountId, limit?) → SwanTransaction[]`	List recent transactions
`onWebhook`	`(callback) → void`	Register webhook listener

Mock Behavior

- 200-800ms simulated latency per call
- IBAN generated in `BA` format (Bosnia mock)
- Transfers start as `Pending`, settle to `Booked` after 2 seconds
- State persisted to `localStorage` (browser) or in-memory (server)
- `_testHelpers.reset()` clears all mock data

Account Statuses

`Opened` | `Closing` | `Closed`

Transaction Types

`SepaCredit` | `SepaDebit` | `CardTransaction`

Transaction Statuses

`Pending` | `Booked` | `Rejected`

Stripe — Card Issuing [MOCK/DEV]

> ⚠ MOCK ONLY: No Stripe SDK installed. Mock file for UI development only.

File: `services/mock-stripe.ts` **Production docs:** https://stripe.com/docs/issuing **Status:** Mock implementation only — no real Stripe API calls, no SDK, no API keys.

Interfaces

Interface	Description
`StripeCard`	Card with type, brand, status, spending limits
`StripeCardDetails`	Full card number, CVC, expiry (virtual only)
`StripeAuthorization`	Card authorization with merchant info

Functions

Function	Signature	Description
`createVirtualCard`	`({cardholderName, spendingLimit?}) → StripeCard`	Issue virtual Visa card
`orderPhysicalCard`	`({cardholderName, shippingAddress}) → StripeCard`	Order physical card
`getCardDetails`	`(cardId) → StripeCardDetails`	Get full card details (virtual only)
`setCardStatus`	`(cardId, active) → StripeCard`	Freeze/unfreeze card
`updateSpendingLimit`	`(cardId, limit) → StripeCard`	Update spending limit
`getCards`	`() → StripeCard[]`	List all cards
`simulateAuthorization`	`({cardId, amount, merchant}) → StripeAuthorization`	Simulate card purchase
`getAuthorizations`	`(cardId?) → StripeAuthorization[]`	List authorizations

Mock Behavior

- Virtual cards created instantly with `active` status, expire in 3 years
- Physical cards start as `pending`, activate after 5 seconds (simulating shipping)
- Default spending limit: 5,000 (virtual), 10,000 (physical)
- Authorization declined if: card not active OR spending limit exceeded
- Brand is always `Visa`
- Mock card numbers: `4242 4242 4242 {last4}`

Card Statuses

`active` | `inactive` | `canceled` | `pending`

Authorization Statuses

`pending` | `approved` | `declined`

Sumsub — KYC/Identity Verification [PRODUCTION]

> PRODUCTION-READY: Sumsub is the only external service with real production API integration.

File: `services/mock-sumsub.ts` **Production docs:** https://docs.sumsub.com/ **Status:** Production-ready — real API calls, WebSDK integration, webhook handling.

Interfaces

Interface	Description
`SumsubApplicant`	KYC applicant with review status
`SumsubDocument`	Identity document (passport, ID card, etc.)
`SumsubVerificationResult`	Verification outcome with per-check breakdown

Functions

Function	Signature	Description
`createApplicant`	`({externalUserId, email?, phone?}) → SumsubApplicant`	Create KYC applicant
`getAccessToken`	`(applicantId) → {token, expiresAt}`	Get WebSDK token (30min)
`submitDocument`	`(applicantId, document) → void`	Submit ID document
`submitSelfie`	`(applicantId, selfieData) → void`	Submit selfie for liveness
`getApplicantStatus`	`(applicantId) → SumsubApplicant`	Check applicant status
`getVerificationResult`	`(applicantId) → SumsubVerificationResult`	Get verification details
`forceApprove`	`(applicantId) → void`	Force approve (testing only)
`onWebhook`	`(callback) → void`	Register webhook listener

Mock Behavior

- Verification completes after 3-second delay
- 90% approval rate in mock mode
- Risk score: 15 (approved) or 85 (rejected)
- Rejected with label `DOCUMENT_UNREADABLE`, type `RETRY`

Applicant Statuses

`init` | `pending` | `queued` | `completed` | `onHold`

Review Answers

`GREEN` (approved) | `RED` (rejected) | `RETRY`

Verification Checks

Check	Description
documentAuthenticity	Document is genuine
livenessCheck	Selfie is a real person
facematch	Selfie matches document photo
sanctionsCheck	Not on sanctions lists
pepCheck	Not a politically exposed person

Document Types

`PASSPORT` | `ID_CARD` | `DRIVERS` | `RESIDENCE_PERMIT`

Service Initialization

Source: `services/index.ts:36-48`

```
// Call on app startup
await initializeServices();

// Reset all mocks (testing)
resetMockServices();
```

Service Status Summary

Service	Status	Description
Sumsub	`[PRODUCTION]`	Real API integration, WebSDK, webhook handling — READY
Stripe	`[MOCK/DEV]`	Mock file only for UI development — NO SDK, NO API keys
Swan	`[DEPRECATED]`	No longer the planned Open Banking provider — mock will be removed
Vipps	`[PLANNED]`	Future consideration — ZERO code currently

Service	Status	Description
Nets	`[PLANNED]`	Future consideration — ZERO code currently

Important Notes

1. **Sumsub is the ONLY production-ready service** — all others are mocks or deprecated.
2. **Console warnings** are emitted on module load for mock services to make usage visible.
3. **Mock state** uses `localStorage` in browser, in-memory on server — resets on server restart.
4. **Production API endpoints** are configurable via environment variables.
5. **The current backend API routes do NOT call these service modules directly** — they use the database layer (`db.ts`) for all operations. The services are available for future integration when real providers are connected.

Middleware Design Document

Middleware Design Document

> **Project:** Drop **Version:** 0.1.0 **Date:** 2026-02-23 **Author:** Platform Architect (AI)
> **Status:** In Review **Reviewers:** Alem Bašić (CEO)

Document History

Version	Date	Author	Changes
0.1	2026-02-23	Platform Architect (AI)	Initial draft from source code analysis

---

1. Overview

Drop has two middleware layers:

1. **`src/lib/middleware.ts`** — The active middleware used by all API routes. Provides `requireAuth`, `requireMerchant`, `rateLimit`, `getClientIp`, `jsonError`, CSRF protection, and session revocation.
2. **`src/lib/middleware/`** — A modular middleware library with `auth-middleware.ts` (Bearer token for mobile), `error-handler.ts` (AppError class), and `validation.ts` (input sanitization functions).

Both layers are used in production. Routes import from `@/lib/middleware` (auth, rate limiting) and `@/lib/middleware/validation` (input validation).

---

2. Active Middleware (`lib/middleware.ts`)

2.1 `requireAuth(request?)`

Source: `middleware.ts:42–80`

Authenticates the current request via cookie-based JWT.

Returns: `{ user: User, error: null } | { user: null, error: NextResponse }`

Steps:

1. **CSRF origin check** — if `Origin` header present, must match allowed origins (`NEXT_PUBLIC_APP_URL`, `http://localhost:3000`, `http://localhost:3001`)
2. **Cookie extraction** — reads `drop_token` from request cookies
3. **JWT verification** — validates HS256 signature and expiry using `jose` library
4. **User lookup** — loads user from `users` table by `userId` from JWT payload
5. **Session revocation check** — verifies at least one non-revoked session exists for this user

Usage:

```
const { user, error } = await requireAuth(request);
if (error) return error;  // Returns NextResponse with JSON error
// user is guaranteed non-null here
```

Error responses:

- 401 `unauthorized` — missing cookie, invalid JWT, expired token, user not found, all sessions revoked

---

2.2 `requireMerchant(request?)`

Source: `middleware.ts:101–108`

Extends `requireAuth` with a merchant role check.

```
const { user, error } = await requireMerchant(request);
if (error) return error;  // 401 if not authenticated, 403 if not merchant
```

Returns 403 **`forbidden`** if user exists but `role !== 'merchant'`.

Applied to: `GET /api/merchants/dashboard`, `GET /api/merchants/qr`, `GET /api/merchants/transactions`

2.3 `rateLimit(ip, limit, windowMs?)`

Source: `middleware.ts:7–31`

Persistent IP-based rate limiter using the `rate_limits` database table.

Parameter	Default	Description
`ip`	—	Client IP address
`limit`	—	Max requests per window
`windowMs`	60,000ms	Window size in milliseconds

Returns: `boolean` — `true` if request is allowed, `false` if rate limited.

Implementation:

- Uses `runUpsert` for atomic counter creation/update
- Cleans expired entries on each call (removes rows where `expires_at < now`)
- Counter stored in `rate_limits` table: `(key, count, expires_at)`

Rate limit table schema:

```
CREATE TABLE rate_limits (
  key TEXT PRIMARY KEY,      -- IP address
  count INTEGER DEFAULT 1,
  expires_at INTEGER         -- Unix timestamp (ms)
);
```

Usage:

```
const ip = getClientIp(request);
if (!(await rateLimit(ip, 10))) {  // 10 req/min
  return jsonError("rate_limited", "Too many requests", 429);
}
```

Applied limits:

Endpoint	Limit	Window
`/api/auth/bankid/initiate`	10/min	60s
`/api/auth/bankid/callback`	10/min	60s
`/api/auth/register` (deprecated)	10/min	60s
`/api/auth/login` (deprecated)	10/min	60s

Endpoint	Limit	Window
`/api/transactions/remittance`	10/min	60s
`/api/transactions/qr-payment`	10/min	60s
`/api/rates`	120/min	60s
`/api/rates/[currency]`	120/min	60s

2.4 `getClientIp(request)`

Source: `middleware.ts:33–35`

Extracts the client's real IP address from the `x-forwarded-for` header (first IP in the chain — the originating client). Falls back to `'127.0.0.1'` if header not present.

Note: When behind App Runner (AWS managed proxy), `x-forwarded-for` is set automatically with the real client IP.

2.5 `jsonError(error, message, status, details?)`

Source: `middleware.ts:37–39`

Creates a standardized JSON error `NextResponse`.

```
return jsonError("validation_error", "Validation failed", 422, ["Email required"]);
// Response body: { "error": "validation_error", "message": "Validation failed", "details":
["Email required"] }
```

2.6 `revokeAllSessions(userId)`

Source: `middleware.ts:83–85`

Sets `revoked=1` on all sessions for a user. Called by `POST /api/auth/logout`.

```
UPDATE sessions SET revoked = 1 WHERE user_id = $1;
```

2.7 `generateCsrfToken() / validateCsrf(request, token)`

Source: `middleware.ts:88–99`

CSRF token generation (32 random bytes hex-encoded) and validation via `x-csrf-token` header.

Status: Implemented but not actively required on any route. CSRF protection is handled via:

- BankID OIDC state parameter (login flow)
- Origin header validation (in `requireAuth`)

---

3. Middleware Library (`lib/middleware/`)

3.1 Error Handler (`middleware/error-handler.ts`)

AppError class:

```
class AppError extends Error {
  constructor(
    public code: string,
    message: string,
    public status: number = 500,
    public details?: unknown
  ) {}
}
```

Predefined error constructors:

Constructor	Code	HTTP Status
`Errors.unauthorized(msg?)`	`UNAUTHORIZED`	401
`Errors.forbidden(msg?)`	`FORBIDDEN`	403
`Errors.notFound(resource)`	`NOT_FOUND`	404
`Errors.badRequest(msg, details?)`	`BAD_REQUEST`	400

Constructor	Code	HTTP Status
`Errors.conflict(msg)`	`CONFLICT`	409
`Errors.tooManyRequests(msg?)`	`RATE_LIMIT_EXCEEDED`	429
`Errors.internal(msg?)`	`INTERNAL_ERROR`	500

Error response format:

```
{
  "error": {
    "code": "BAD_REQUEST",
    "message": "Amount must be between 100 and 50000 NOK",
    "details": "validation_error"
  }
}
```

Production masking: `createErrorResponse()` masks internal error messages in production — only returns `"An unexpected error occurred"` for 500 errors.

3.2 Auth Middleware (`middleware/auth-middleware.ts`)

Alternative auth middleware for **mobile clients** using Bearer token pattern.

`requireAuth(request)`:

- Extracts JWT from `Authorization: Bearer <token>` header
- Verifies JWT signature + expiry
- Returns `userId` from payload

In-memory rate limiter (for Bearer token routes):

- `DEFAULT_RATE_LIMIT`: 100 req/min
- `STRICT_RATE_LIMIT`: 10 req/min
- Auto-cleanup every 5 minutes
- Rate limit headers: `X-RateLimit-Limit`, `X-RateLimit-Remaining`, `X-RateLimit-Reset`

`getClientIP(request)`: Checks `X-Forwarded-For` → `X-Real-IP` → falls back to `'unknown'`.

3.3 Validation (`middleware/validation.ts`)

Input validation functions — no external dependencies, all custom implementations.

Function	Description	Rules
`validatePhone(phone)`	International phone	Starts with `+`, 8–15 digits
`validateAmount(amount)`	Positive monetary amount	`> 0`, max 2 decimal places
`validateIBAN(iban)`	European IBAN	Country code + alphanumeric, mod-97 checksum
`validatePIN(pin)`	Card PIN	Exactly 4 digits
`validateEmail(email)`	Email address	Basic `x@y.z` pattern
`validateCurrency(currency)`	ISO 4217 code	Whitelist: EUR, USD, GBP, BAM, CHF, PLN, NOK, RSD, TRY, PKR
`validateDateISO(date)`	ISO 8601 date	Parseable by `Date.parse()`
`validateName(name)`	Name field	1–100 chars, at least one letter, XSS-safe
`validateLanguage(lang)`	Language code	Whitelist: nb, en, bs, sq
`sanitizeText(text, maxLength?)`	Text sanitization	Strips HTML tags + control chars, trims, enforces max length (default 500)
`validate(condition, msg)`	Assert helper	Throws `AppError` (400) if false
`required(value, name)`	Required field check	Throws `AppError` (400) if null/undefined

Security notes:

- `validateName` checks for: `<script`, `javascript:`, `onerror=`, `onclick=` — blocks XSS injection in name fields
- `sanitizeText` removes HTML tags via regex, strips control characters
- `validateIBAN` implements full mod-97 checksum algorithm
- `validateAmount` rejects `NaN`, `Infinity`, negative values

---

4. Security Headers (Next.js Config)

Applied to all responses via `next.config.ts`:

Header	Production Value	Development Value	Purpose
`Content-Security-Policy`	`default-src 'self'; script-src 'self'; style-src 'self' 'unsafe-inline'; img-src 'self' data: blob:; connect-src 'self'; frame-ancestors 'none'`	Adds `'unsafe-eval'` + `'unsafe-inline'` for HMR	XSS protection

Header	Production Value	Development Value	Purpose
`X-Frame-Options`	`DENY`	`DENY`	Clickjacking prevention
`X-Content-Type-Options`	`nosniff`	`nosniff`	MIME sniffing prevention
`Referrer-Policy`	`strict-origin-when-cross-origin`	Same	Referrer leakage prevention
`Permissions-Policy`	`camera=(self), microphone=(), geolocation=(self)`	Same	Feature restriction
`Strict-Transport-Security`	`max-age=63072000; includeSubDomains; preload`	Same	Force HTTPS (2-year HSTS)

---

5. Middleware Usage Matrix

Route	Rate Limit	`requireAuth`	`requireMerchant`	Feature Flag	Validation Functions
GET `/api/auth/bankid`	10/min	No	No	No	—
GET `/api/auth/bankid/callback`	10/min	No	No	No	state cookie
GET `/api/auth/me`	No	Yes	No	No	—
POST `/api/auth/logout`	No	Yes	No	No	—
POST `/api/auth/refresh`	No	Yes	No	No	—
GET `/api/transactions`	No	Yes	No	No	—
POST `/api/transactions/remittance`	10/min	Yes	No	No	`validateAmount`
POST `/api/transactions/qr-payment`	10/min	Yes	No	No	`validateAmount`
GET `/api/rates`	120/min	No	No	No	—
POST `/api/recipients`	No	Yes	No	No	`validateName`, country whitelist
POST `/api/merchants/register`	No	Yes	No	No	`validateName`, orgNumber
GET `/api/merchants/dashboard`	No	Yes	Yes	No	period whitelist

Route	Rate Limit	`requireAuth`	`requireMerchant`	Feature Flag	Validation Functions
GET `/api/notifications`	No	Yes	No	`notifications`	—
PATCH `/api/notifications`	No	Yes	No	`notifications`	ID format, max 100
PATCH `/api/settings`	No	Yes	No	No	currency/language whitelist
POST `/api/cards/[id]/physical`	No	Yes	No	`physicalCards`	address min 10 chars
POST `/api/cards/[id]/pin`	No	Yes	No	`cardPin`	`validatePIN`
GET/PUT `/api/cards/[id]/limits`	No	Yes	No	`spendingLimits`	limitType whitelist

---

6. Error Spike Detection

Implemented in `src/lib/alerts.ts` as a middleware-adjacent concern:

- Every HTTP 5xx response triggers `trackError()` (called in `jsonError()` middleware for 500 errors)
- Rolling 1-minute window of error timestamps maintained in-memory
- When count > 5 in 60 seconds → sends critical Slack alert to `#drop-ops`
- 10-minute cooldown per alert title prevents spam

Limitation: Error counter is in-memory only — resets on application restart. Redis-backed counter planned for v1.0.

Related Documents

- Backend Architecture
- API Reference
- Source: MIDDLEWARE.md

Approval

Role	Name	Date	Signature
Author	Platform Architect (AI)	2026-02-23	
Reviewer			
Approver	Alem Bašić		

Feature Flags

Drop Feature Flags

> Sources: `src/drop-app/src/lib/feature-flags.ts`, `src/drop-app/src/lib/features.ts`

Feature Flag System

Source: `feature-flags.ts`

Architecture

Feature flags are controlled via **environment variables** with the pattern:

```
NEXT_PUBLIC_FF_<SCREAMING_SNAKE_CASE>=true|false
```

The `NEXT_PUBLIC_` prefix ensures flags are available on both server and client (inlined at build time by Next.js).

Conversion example: `physicalCards` → `NEXT_PUBLIC_FF_PHYSICAL_CARDS`

Source: `feature-flags.ts:42-45`

Available Flags

Flag Name	Env Var	Default	Description
virtualCards	NEXT_PUBLIC_FF_VIRTUAL_CARDS	false	Virtual card issuance
physicalCards	NEXT_PUBLIC_FF_PHYSICAL_CARDS	false	Physical card ordering
cardDetails	NEXT_PUBLIC_FF_CARD_DETAILS	false	View full card details

Flag Name	Env Var	Default	Description
cardFreeze	NEXT_PUBLIC_FF_CARD_FREEZE	false	Card freeze/unfreeze
cardPin	NEXT_PUBLIC_FF_CARD_PIN	false	Card PIN management
spendingLimits	NEXT_PUBLIC_FF_SPENDING_LIMITS	false	Card spending limits
notifications	NEXT_PUBLIC_FF_NOTIFICATIONS	true	Push notifications
merchantDashboard	NEXT_PUBLIC_FF_MERCHANT_DASHBOARD	true	Merchant dashboard

Source: `feature-flags.ts:27-36`

Server-Side API

Function	Return Type	Description
`isEnabled(flag)`	`boolean`	Check if a flag is enabled
`getAllFlags()`	`FeatureFlags`	Get all flags with current values
`featureGate(flag)`	`NextResponse | null`	API middleware: returns 404 response if disabled, null if enabled

`featureGate` usage in routes:

```
// In any route handler:
const gate = featureGate("physicalCards");
if (gate) return gate;  // Returns 404 with "Feature not available"
```

Source: `feature-flags.ts:80-88`

Routes using `featureGate`:

Route	Flag
POST /api/cards/[id]/physical	`physicalCards`
POST /api/cards/[id]/pin	`cardPin`
GET /api/cards/[id]/limits	`spendingLimits`
PUT /api/cards/[id]/limits	`spendingLimits`
GET /api/notifications	`notifications`
PATCH /api/notifications	`notifications`

Client-Side API

Function	Return Type	Description
`useFeatureFlag(flag)`	`boolean`	React hook for a single flag
`useFeatureFlags()`	`FeatureFlags`	React hook for all flags

These work because `NEXT_PUBLIC_*` env vars are inlined at build time — no server roundtrip needed.

Source: `feature-flags.ts:94-114`

Feature Tracking System

Source: `features.ts`

A separate system for tracking **implementation progress** of Drop features. Not runtime flags — this is a development tracking tool.

Feature Interface

```
interface Feature {
  id: string;              // e.g., "auth-001"
  category: string;        // e.g., "Authentication"
  name: string;            // e.g., "User Registration"
  description: string;
  status: "pending" | "in_progress" | "passing" | "failing";
  priority: number;        // 1 = highest
  dependencies: string[];  // IDs of prerequisite features
  acceptanceCriteria: string[];
  implementedAt?: string;  // ISO date
  testedAt?: string;       // ISO date
}
```

Feature Categories and Status

Category	Total	Passing	Pending	Notes

Authentication	4	3	1 (Biometric Login)	
KYC	1	1	0	
Banking	6	5	1	bank-006 (Top-up via Card) is FUTURE — incompatible with pass-through model
Cards	4	4	0	FUTURE — all card features are gated behind feature flags (default: false)
Notifications	1	0	1 (Push Notifications)	

All Features

ID	Name	Status	Priority	Dependencies	Notes
auth-001	User Registration	passing	1	-	
auth-002	PIN Login	passing	1	auth-001	
auth-003	Logout	passing	2	auth-002	
auth-004	Biometric Login	pending	3	auth-002	
kyc-001	Identity Verification	passing	1	auth-001	
bank-001	IBAN Generation	passing	1	kyc-001	
bank-002	Balance Display	passing	1	bank-001	AISP read-only
bank-003	Send Money	passing	1	bank-002	PISP from user's bank
bank-004	Receive Money	passing	1	bank-001	
bank-005	Transaction History	passing	2	bank-003, bank-004	
bank-006	Top-up via Card	passing	2	bank-001	**FUTURE** — no wallet in pass-through model
card-001	Virtual Card Issuance	passing	1	kyc-001	**FUTURE** — feature-flagged
card-002	Card Freeze/Unfreeze	passing	2	card-001	**FUTURE** — feature-flagged
card-003	Card Transactions	passing	1	card-001	**FUTURE** — feature-flagged
card-004	Physical Card Order	passing	3	card-001	**FUTURE** — feature-flagged

ID	Name	Status	Priority	Dependencies	Notes
notif-001	Push Notifications	pending	3	auth-001	

Helper Functions

Function	Description
`getFeaturesByStatus(status)`	Filter features by status
`getFeaturesByCategory(category)`	Filter features by category
`getFeatureStats()`	Get counts: total, passing, pending, inProgress, failing, percentComplete
`getReadyFeatures()`	Features whose dependencies are all `passing`
`printFeatureReport()`	Formatted text report

Source: `features.ts:284-357`

Environment Variable Summary

Variable	Purpose	Default
NEXT_PUBLIC_FF_VIRTUAL_CARDS	Enable virtual cards	false
NEXT_PUBLIC_FF_PHYSICAL_CARDS	Enable physical cards	false
NEXT_PUBLIC_FF_CARD_DETAILS	Enable card detail view	false
NEXT_PUBLIC_FF_CARD_FREEZE	Enable card freeze	false
NEXT_PUBLIC_FF_CARD_PIN	Enable card PIN	false
NEXT_PUBLIC_FF_SPENDING_LIMITS	Enable spending limits	false
NEXT_PUBLIC_FF_NOTIFICATIONS	Enable notifications	true
NEXT_PUBLIC_FF_MERCHANT_DASHBOARD	Enable merchant dashboard	true
NEXT_PUBLIC_SERVICE_MODE	mock or production	mock
DATABASE_URL	PostgreSQL 16 connection string	Required (no SQLite fallback)
JWT_SECRET	JWT signing secret	dev-only fallback
NEXT_PUBLIC_APP_URL	App URL for CSRF	-
SEED_DEMO	Enable demo data in staging	-

Middleware

Drop Middleware

> Sources: `src/drop-app/src/lib/middleware.ts`, `src/drop-app/src/lib/middleware/`

Overview

Drop has two middleware layers:

1. **`lib/middleware.ts`** — The active middleware used by all API routes. Provides `requireAuth`, `requireMerchant`, `rateLimit`, `getClientIp`, `jsonError`, CSRF, and session revocation.
2. **`lib/middleware/`** directory — A modular middleware library with `auth-middleware.ts`, `error-handler.ts`, and `validation.ts`. Exported via barrel file `middleware/index.ts`.

The API routes import from both: `@/lib/middleware` (auth, rate limiting) and `@/lib/middleware/validation` (input validation).

Active Middleware (`middleware.ts`)

requireAuth(request?)

Source: `middleware.ts:42-80`

Authenticates the current request via cookie-based JWT. Returns `{ user, error }`.

Steps:

1. **CSRF origin check** — If `Origin` header present, must match allowed origins
2. **Cookie extraction** — Reads `drop_token` from cookies
3. **JWT verification** — Validates signature and expiry
4. **User lookup** — Loads user from `users` table
5. **Session revocation check** — Verifies at least one non-revoked session exists

Allowed origins: `NEXT_PUBLIC_APP_URL`, `http://localhost:3000`, `http://localhost:3001`

```
const { user, error } = await requireAuth(request);
if (error) return error;  // Returns NextResponse with error JSON
```

---

requireMerchant(request?)

Source: `middleware.ts:101-108`

Extends `requireAuth` with a role check: user must have `role === 'merchant'`. Returns 403 if not.

```
const { user, error } = await requireMerchant(request);
if (error) return error;
```

---

rateLimit(ip, limit, windowMs?)

Source: `middleware.ts:7-31`

Persistent IP-based rate limiter using the `rate_limits` database table.

```
if (!(await rateLimit(ip, 10))) {           // 10 requests per 60s window
  return jsonError("rate_limited", "Too many requests", 429);
}
```

- Default window: 60,000ms (1 minute)
- Cleans expired entries on each call
- Uses `runUpsert` for atomic counter creation/update

---

getClientIp(request)

Source: `middleware.ts:33-35`

Extracts client IP from `x-forwarded-for` header (first IP in chain), falls back to `127.0.0.1`.

---

jsonError(error, message, status, details?)

Source: `middleware.ts:37-39`

Creates a standardized JSON error response.

```
return jsonError("validation_error", "Validation failed", 422, ["Email required"]);
// → { "error": "validation_error", "message": "Validation failed", "details": ["Email
required"] }
```

---

revokeAllSessions(userId)

Source: `middleware.ts:83-85`

Sets `revoked=1` on all sessions for a user. Called during logout.

---

generateCsrfToken() / validateCsrf(request, token)

Source: `middleware.ts:88-99`

CSRF token generation (32 random bytes hex-encoded) and validation via `x-csrf-token` header. Available but not actively required on any route.

Middleware Library (`middleware/`)

Error Handler

Source: `middleware/error-handler.ts`

AppError class:

```
class AppError extends Error {
  constructor(code: string, message: string, status: number = 500, details?: unknown)
}
```

Predefined error constructors (`Errors.*`):

Constructor	Code	Status
`Errors.unauthorized(msg?)`	UNAUTHORIZED	401

Constructor	Code	Status
`Errors.forbidden(msg?)`	FORBIDDEN	403
`Errors.notFound(resource)`	NOT_FOUND	404
`Errors.badRequest(msg, details?)`	BAD_REQUEST	400
`Errors.conflict(msg)`	CONFLICT	409
`Errors.tooManyRequests(msg?)`	RATE_LIMIT_EXCEEDED	429
`Errors.internal(msg?)`	INTERNAL_ERROR	500

Error response format:

```
{
  "error": {
    "code": "BAD_REQUEST",
    "message": "...",
    "details": "..."
  }
}
```

`createErrorResponse(error)` handles `AppError`, standard `Error`, and unknown errors. In development, includes original error messages; in production, masks internal errors.

Auth Middleware

Source: `middleware/auth-middleware.ts`

Alternative auth middleware using Bearer token pattern (vs. cookie pattern in `middleware.ts`).

`requireAuth(request)` — Extracts JWT from `Authorization: Bearer <token>` header, verifies, returns userId.

In-memory rate limiter with:

- `DEFAULT_RATE_LIMIT`: 100 req/min
- `STRICT_RATE_LIMIT`: 10 req/min
- Auto-cleanup every 5 minutes
- Rate limit response headers (`X-RateLimit-*`)

`getClientIP(request)` — Checks `X-Forwarded-For`, then `X-Real-IP`, then falls back to `'unknown'`.

Validation

Source: `middleware/validation.ts`

Input validation functions (no external dependencies):

Function	Description	Rules
`validatePhone(phone)`	International phone format	Starts with `+`, 8-15 digits
`validateAmount(amount)`	Positive number	> 0, max 2 decimal places
`validateIBAN(iban)`	European IBAN format	Country code + digits + alphanumeric, mod-97 checksum
`validatePIN(pin)`	Card PIN	Exactly 4 digits
`validateEmail(email)`	Email address	Basic `x@y.z` pattern
`validateCurrency(currency)`	ISO 4217 code	Whitelist: EUR, USD, GBP, BAM, CHF, PLN, NOK, RSD, TRY, PKR
`validateDateISO(date)`	ISO 8601 date	Parseable by `Date.parse()`
`validateName(name)`	Name field	1-100 chars, at least one letter, no script/HTML injection
`validateLanguage(lang)`	Language code	Whitelist: nb, en, bs, sq
`sanitizeText(text, maxLength?)`	Text sanitization	Strips HTML tags, control chars, trims, enforces max length (default 500)
`validate(condition, msg)`	Assert helper	Throws `AppError` (400) if false
`required(value, name)`	Required check	Throws `AppError` (400) if null/undefined

Security notes:

- `validateName` checks for dangerous patterns: `<script`, `javascript:`, `onerror=`, `onclick=`
- `sanitizeText` removes HTML tags via regex, strips control characters
- IBAN validation implements the full mod-97 checksum algorithm

---

Middleware Usage by Route

Route	Rate Limit	Auth	Merchant	Feature Flag	Validation
POST /auth/register	10/min	-	-	-	email, name, phone, age
POST /auth/login	10/min	-	-	-	-
GET /auth/me	-	Yes	-	-	-

Route	Rate Limit	Auth	Merchant	Feature Flag	Validation
POST /auth/logout	-	Yes	-	-	-
POST /auth/refresh	-	Yes	-	-	-
GET /transactions	-	Yes	-	-	-
POST /transactions/remittance	10/min	Yes	-	-	amount range, decimal
POST /transactions/qr-payment	10/min	Yes	-	-	amount range, decimal
GET /rates	120/min	-	-	-	-
GET /rates/[currency]	120/min	-	-	-	-
POST /cards/[id]/physical	-	Yes	-	physicalCards	address min 10 chars
POST /cards/[id]/pin	-	Yes	-	cardPin	4-digit PIN
GET /cards/[id]/limits	-	Yes	-	spendingLimits	-
PUT /cards/[id]/limits	-	Yes	-	spendingLimits	limitType whitelist
GET /notifications	-	Yes	-	notifications	-
PATCH /notifications	-	Yes	-	notifications	ID format, max 100
PATCH /settings	-	Yes	-	-	currency/language whitelist
POST /recipients	-	Yes	-	-	name, country whitelist
POST /merchants/register	-	Yes	-	-	orgNumber 9 digits
GET /merchants/dashboard	-	Yes	Merchant	-	period whitelist
GET /merchants/qr	-	Yes	Merchant	-	-
GET /merchants/transactions	-	Yes	Merchant	-	-