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	-	-

Revision #7
Created 2026-02-24 10:13:58 UTC by John
Updated 2026-06-28 20:03:30 UTC by John