API & Data

- API Reference
- Authentication

API Reference

API Reference

> **Project:** {{PROJECT_NAME}} **Version:** {{VERSION}} **Date:** {{DATE}}
> **Author:** {{AUTHOR}} **Status:** Draft | In Review | Approved **Reviewers:** {{REVIEWERS}}

Document History

Version	Date	Author	Changes
0.1	{{DATE}}	{{AUTHOR}}	Initial draft

---

1. API Overview & Conventions

API style: `{{RESTful HTTP/JSON | GraphQL | gRPC}}` **API version strategy:** `{{URL versioning: /v1/ | Header versioning}}` **OpenAPI spec:** `{{https://api.domain.com/docs/openapi.json}}` **Interactive docs:** `{{https://api.domain.com/docs}}`

Design conventions:

- Resources named as plural nouns: `/users`, `/orders`, `/products`
- HTTP methods map to CRUD: GET (read), POST (create), PUT (replace), PATCH (update), DELETE (remove)
- Response format: always JSON
- Timestamps: ISO 8601 UTC (`2024-01-15T10:30:00Z`)
- IDs: UUID v4 strings
- Booleans: `true` / `false` (never `1` / `0`)
- Empty collections: `[]` (never `null`)
- Missing optional fields: omitted (never `null` unless semantically null)

---

2. Base URLs

Environment	Base URL
Development	`http://localhost:4000/v1`
Staging	`https://api-staging.{{domain.com}}/v1`
Production	`https://api.{{domain.com}}/v1`

---

3. Authentication

Method: Bearer Token (JWT)

Obtain tokens: `POST /auth/login` (see Auth section below)

Include in requests:

```
Authorization: Bearer <access_token>
```

Token lifetimes:

- Access token: 15 minutes
- Refresh token: 30 days (rotate on use)

Refresh tokens: `POST /auth/refresh` with `{ "refreshToken": "..." }` in body.

API Key authentication (machine-to-machine):

```
X-API-Key: <api_key>
```

API keys are scoped and managed at `{{https://dashboard.domain.com/api-keys}}`.

---

4. Common Request/Response Headers

4.1 Request Headers

Header	Required	Description
`Authorization`	Yes (auth routes)	`Bearer <token>`
`Content-Type`	Yes (POST/PUT/PATCH)	`application/json`
`Accept`	No	`application/json` (default)
`X-Request-ID`	No	Client-provided idempotency ID
`Accept-Language`	No	`en`, `nb`, etc. — affects response locale

4.2 Response Headers

Header	Description
`Content-Type`	`application/json; charset=utf-8`
`X-Request-ID`	Echo of client request ID (or server-generated)
`X-RateLimit-Limit`	Total requests allowed in window
`X-RateLimit-Remaining`	Remaining requests in current window
`X-RateLimit-Reset`	Unix timestamp when window resets
`Retry-After`	Seconds to wait (set when 429 returned)

---

5. Error Response Format

```
{
  "error": {
    "code": "VALIDATION_ERROR",
    "message": "Request validation failed",
    "details": [
      {
        "field": "email",
        "message": "Invalid email format",
        "code": "INVALID_FORMAT"
      }
    ],
    "requestId": "req_7f3a2b1c",
    "timestamp": "2024-01-15T10:30:00Z"
  }
}
```

Standard error codes:

HTTP Status	Error Code	Meaning
400	`VALIDATION_ERROR`	Request body / params failed validation
400	`BAD_REQUEST`	Malformed request
401	`UNAUTHORIZED`	Missing or invalid authentication
401	`TOKEN_EXPIRED`	JWT has expired — refresh required
403	`FORBIDDEN`	Authenticated but lacks permission
404	`NOT_FOUND`	Resource does not exist
409	`CONFLICT`	Resource already exists / version conflict
422	`UNPROCESSABLE`	Valid format but business rule violation
429	`RATE_LIMITED`	Too many requests
500	`INTERNAL_ERROR`	Unexpected server error
503	`SERVICE_UNAVAILABLE`	Temporary downtime

---

6. Resources

6.1 Authentication

POST /auth/login

Authenticate user and receive token pair.

Auth required: No

Request body:

```
{
  "email": "user@example.com",
  "password": "{{password}}"
}
```

Response `200 OK`:

```
{
  "accessToken": "eyJhbGciOiJIUzI1NiJ9...",
  "refreshToken": "dGhpcyBpcyBhIHJlZnJlc2g...",
  "expiresIn": 900,
  "user": {
    "id": "usr_01HX7...",
    "email": "user@example.com",
    "name": "John Doe",
    "role": "user"
  }
}
```

Error responses:

Status	Code	Condition
401	`INVALID_CREDENTIALS`	Wrong email or password
429	`RATE_LIMITED`	> 5 failed attempts in 15 min

POST /auth/refresh

Rotate access and refresh tokens.

Auth required: No

Request body:

```
{ "refreshToken": "dGhpcyBpcyBhIHJlZnJlc2g..." }
```

Response `200 OK`: Same as login response.

POST /auth/logout

Revoke refresh token.

Auth required: Yes (Bearer)

Request body:

```
{ "refreshToken": "dGhpcyBpcyBhIHJlZnJlc2g..." }
```

Response `204 No Content`

6.2 Users

Endpoints:

Method	Path	Description	Auth
`GET`	`/users`	List users (paginated)	Admin
`GET`	`/users/:id`	Get user by ID	Self or Admin
`POST`	`/users`	Create user	Admin
`PATCH`	`/users/:id`	Update user fields	Self or Admin
`DELETE`	`/users/:id`	Delete user (soft)	Admin
`GET`	`/users/me`	Get current user	Authenticated
`PATCH`	`/users/me`	Update current user	Authenticated

GET /users

List users with pagination and filtering.

Auth required: Admin

Query parameters:

Parameter	Type	Default	Description
`page`	integer	`1`	Page number
`pageSize`	integer	`25`	Items per page (max: 100)
`search`	string	—	Search name or email (min 2 chars)
`role`	string	—	Filter by role: `admin`, `user`, `viewer`
`status`	string	`active`	Filter by status: `active`, `inactive`, `all`
`sort`	string	`createdAt`	Sort field
`dir`	string	`desc`	Sort direction: `asc`, `desc`

Response `200 OK`:

```
{
  "data": [
    {
```

```
      "id": "usr_01HX7...",
      "email": "user@example.com",
      "name": "Jane Doe",
      "role": "user",
      "status": "active",
      "createdAt": "2024-01-15T10:30:00Z",
      "updatedAt": "2024-01-15T10:30:00Z"
    }
  ],
  "pagination": {
    "page": 1,
    "pageSize": 25,
    "total": 142,
    "totalPages": 6
  }
}
```

GET /users/:id

Auth required: Self or Admin

Path parameters:

Parameter	Type	Description
`id`	UUID	User ID

Response `200 OK`:

```
{
  "id": "usr_01HX7...",
  "email": "user@example.com",
  "name": "Jane Doe",
  "role": "user",
  "status": "active",
  "profile": {
    "avatarUrl": "https://cdn.domain.com/avatars/...",
    "bio": "Software developer"
  },
  "createdAt": "2024-01-15T10:30:00Z",
  "updatedAt": "2024-01-15T10:30:00Z"
```

```
}
```

Error responses:

Status	Code	Condition
404	`NOT_FOUND`	User does not exist
403	`FORBIDDEN`	Non-admin accessing another user

POST /users

Auth required: Admin

Request body:

```
{
  "email": "newuser@example.com",
  "name": "New User",
  "role": "user",
  "password": "{{password}}"
}
```

Response `201 Created`: Full user object (same as GET /users/:id)

PATCH /users/:id

Auth required: Self or Admin

Request body (all fields optional):

```
{
  "name": "Updated Name",
  "profile": {
    "bio": "Updated bio"
  }
}
```

Response `200 OK`: Updated user object.

DELETE /users/:id

Soft-deletes user (sets `deletedAt`, anonymizes PII).

Auth required: Admin

Response `204 No Content`

6.3 {{RESOURCE_NAME}}

Endpoints:

Method	Path	Description	Auth
`GET`	`/{{resource}}`	List `{{resource}}`	`{{Auth level}}`
`GET`	`/{{resource}}/:id`	Get by ID	`{{Auth level}}`
`POST`	`/{{resource}}`	Create	`{{Auth level}}`
`PATCH`	`/{{resource}}/:id`	Update	`{{Auth level}}`
`DELETE`	`/{{resource}}/:id`	Delete	`{{Auth level}}`

TODO: Document all endpoints for this resource following the pattern above.

---

7. Pagination Format

All list endpoints return the same pagination envelope:

```
{
  "data": [...],
  "pagination": {
    "page": 1,
    "pageSize": 25,
    "total": 142,
    "totalPages": 6,
    "hasNextPage": true,
    "hasPreviousPage": false
  }
}
```

Cursor pagination (high-performance, for infinite scroll):

```
GET /feed?cursor=eyJpZCI6MTIzfQ&pageSize=20
```

Response includes `nextCursor` — pass as `cursor` in next request.

8. Filtering & Sorting Conventions

Filter parameters:

```
GET /orders?status=pending&createdAt[gte]=2024-01-01&total[lte]=1000
```

Operator	Suffix	Example
Equals	(none)	`?status=active`
Greater than	`[gt]`	`?price[gt]=100`
Greater than or equal	`[gte]`	`?price[gte]=100`
Less than	`[lt]`	`?price[lt]=500`
Less than or equal	`[lte]`	`?price[lte]=500`
In list	`[in]`	`?status[in]=active,pending`
Not in list	`[nin]`	`?status[nin]=deleted`

Sort: `?sort=createdAt&dir=desc` (default: `createdAt desc`)

---

9. Webhooks Documentation

Webhook endpoint: Configured per-account at `{{https://dashboard.domain.com/webhooks}}`

Delivery: HTTP POST with JSON body, signed with HMAC-SHA256.

Signature verification:

```
const signature = req.headers['x-webhook-signature'];
const computed = crypto
  .createHmac('sha256', webhookSecret)
  .update(rawBody)
  .digest('hex');
const valid = crypto.timingSafeEqual(
  Buffer.from(signature), Buffer.from(computed)
);
```

Event envelope:

```
{
  "id": "evt_01HX7...",
  "type": "user.created",
  "data": { /* resource object */ },
  "timestamp": "2024-01-15T10:30:00Z",
  "version": "1"
}
```

Available events:

Event	Trigger
`user.created`	New user registered
`user.updated`	User profile changed
`user.deleted`	User account deleted
`{{resource}}.{{action}}`	`{{Description}}`

Retry policy: 5 retries with exponential backoff. Undeliverable after 24 hours → marked as failed.

---

10. Rate Limiting

Endpoint Group	Limit	Window
Public endpoints	100 req	1 minute
Authenticated endpoints	1000 req	1 minute
Admin endpoints	5000 req	1 minute
Auth endpoints (login)	5 req	15 minutes
Webhook delivery	N/A	—

Response when rate limited (`429 Too Many Requests`):

```
{
  "error": {
    "code": "RATE_LIMITED",
    "message": "Too many requests. Please retry after 60 seconds.",
    "retryAfter": 60
  }
}
```

11. Code Examples

cURL

```
# Login
curl -X POST https://api.domain.com/v1/auth/login \
  -H "Content-Type: application/json" \
  -d '{"email": "user@example.com", "password": "secret"}'

# Get users (with token)
curl -X GET "https://api.domain.com/v1/users?page=1&pageSize=10" \
  -H "Authorization: Bearer <access_token>"
```

JavaScript (fetch)

```
const response = await fetch('https://api.domain.com/v1/users', {
  headers: {
    'Authorization': `Bearer ${accessToken}`,
    'Content-Type': 'application/json',
  },
});

if (!response.ok) {
  const error = await response.json();
  throw new Error(error.error.message);
}

const { data, pagination } = await response.json();
```

Python

```
import httpx

client = httpx.Client(
    base_url="https://api.domain.com/v1",
    headers={"Authorization": f"Bearer {access_token}"}
```

```
)

response = client.get("/users", params={"page": 1, "pageSize": 10})
response.raise_for_status()
result = response.json()
```

---

12. SDK Availability

Language	Package	Repository	Status
TypeScript / JavaScript	`@{{company}}/api-client`	`{{URL}}`	`{{Available/Planned}}`
Python	`{{company}}-python`	`{{URL}}`	`{{Available/Planned}}`
Go	`{{company}}-go`	`{{URL}}`	`{{Planned}}`

Approval

Role	Name	Date	Signature
Author			
Backend Lead			
Tech Lead			
Product Owner			

Authentication

Drop Authentication System

> Sources: `src/drop-app/src/app/api/auth/bankid/`, `src/drop-api/src/lib/bankid.ts`, `src/drop-api/src/routes/auth.ts`

Overview

Drop uses **BankID OIDC** as the sole authentication method. Email/password login has been removed to comply with PSD2/SCA requirements.

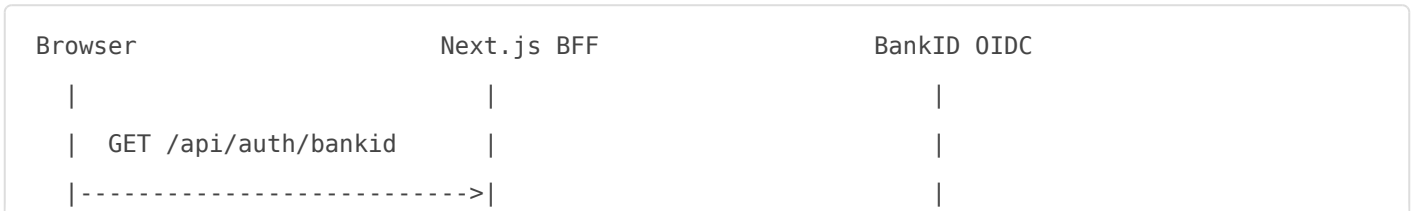
- **Auth method:** BankID OIDC (Norwegian eID)
- **JWT Algorithm:** HS256 (HMAC-SHA256), RS256 opt-in
- **Library:** `jose` (`SignJWT` / `jwtVerify`)
- **Token lifetime:** 24h (web cookie), 7d (mobile Bearer token)
- **Web cookie:** `drop_token` (httpOnly, secure, sameSite=strict)
- **Mobile:** Bearer token in Authorization header

Phase 2 (planned)

- Vipps Login — same OIDC pattern, user dedup by `national_id_hash`
- Idura aggregator optional (single integration point for BankID + Vipps)

Authentication Flow

BankID Login (Web)



```
  |                           |  1. Rate limit check         |
  |                           |  2. Generate state + nonce   |
  |                           |  3. Set bankid_state cookie  |
  |  { redirectUrl }          |                              |
  |<--------------------------|                              |
  |                           |                              |
  |  Browser redirects to BankID authorize URL               |
  |--------------------------------------------------------->|
  |                           |                              |
  |  User authenticates with BankID                          |
  |                           |                              |
  |  BankID redirects to /api/auth/bankid/callback?code=&state=
  |<---------------------------------------------------------|
  |                           |                              |
  |  GET /callback?code&state |                              |
  |-------------------------->|                              |
  |                           |  4. Verify state vs cookie   |
  |                           |  5. Exchange code for tokens |
  |                           |----------------------------->|
  |                           |  { id_token, access_token }  |
  |                           |<-----------------------------|
  |                           |  6. Verify ID token (JWKS)   |
  |                           |  7. Parse pid, verify age    |
  |                           |  8. Find/create user         |
  |                           |  9. Create session + cookie  |
  |  302 /dashboard           |                              |
  |<--------------------------|                              |
```

BankID Login (Mobile)

```
Mobile App                 Hono API                      BankID OIDC
  |                           |                              |
  |  GET /v1/auth/bankid/initiate?platform=mobile            |
  |-------------------------->|                              |
  |  { redirectUrl, state }   |                              |
  |<--------------------------|                              |
  |                           |                              |
  |  Open BankID in secure browser (expo-web-browser)        |
  |--------------------------------------------------------->|
```

```
|                           |                                |
|  User authenticates with BankID                            |
|                           |                                |
|  Redirect to drop://auth/callback?code=&state=             |
|<-----------------------------------------------------------|
|                           |                                |
|  POST /v1/auth/bankid/callback                             |
|  { code, state, platform }                                 |
|-------------------------->|                                |
|                           |  1. Exchange code for tokens   |
|                           |------------------------------->|
|                           |  { id_token }                  |
|                           |<-------------------------------|
|                           |  2. Verify ID token (JWKS)     |
|                           |  3. Parse pid, verify age      |
|                           |  4. Find/create user           |
|                           |  5. Create session             |
|  { token, data }          |                                |
|<--------------------------|                                |
|                           |                                |
|  Store token in AsyncStorage                               |
```

User Creation

BankID login automatically creates user accounts:

1. **Parse pid** from BankID ID token (Norwegian national ID, 11 digits)
2. **Hash pid** with SHA-256 for storage (`national_id_hash` column)
3. **Check existing** user by `national_id_hash`
4. **If new:** Create user with:
   - `kyc_status = 'approved'` (BankID = verified identity)
   - `kyc_method = 'bankid'`
   - `auth_provider = 'bankid'`
   - `password_hash = 'EIDONLY'` (sentinel — no password auth)
5. **Age check:** Must be >= 18 (parsed from pid birthdate)

JWT Structure

Payload

```
interface JwtPayload {
  userId: string;   // e.g., "usr_a1b2c3d4e5f6g7h8"
  email: string;    // e.g., "usr_xxx@bankid.drop.local"
  role: string;     // "user" or "merchant"
}
```

Claims

Claim	Value
`exp`	Current time + 24h (web) / 7d (mobile)
`iat`	Current time
`iss`	`drop-api` (Hono) / none (Next.js)
`aud`	`drop` (Hono) / none (Next.js)

Session Revocation

1. **On login:** `sessions` record created with SHA-256 hash of JWT
2. **On each request:** Verify session not revoked + not expired
3. **On logout:** All user sessions marked `revoked = 1`

CSRF Protection

- **Web:** State parameter in BankID OIDC flow (stored in httpOnly cookie)
- **API:** Origin header validation against allowed origins
- **Mobile:** N/A (Bearer token, no cookies)

Rate Limiting

Endpoint	Limit
BankID initiate	10/min per IP
BankID callback	10/min per IP

Endpoint	Limit
Auth me/logout/refresh	No additional limit (auth required)

Authorization

Role-Based Access

Two roles: `user` and `merchant`.

Route	Auth	Role
GET /auth/bankid/initiate	None	-
POST /auth/bankid/callback	None	-
GET /auth/me	Required	Any
POST /auth/logout	Required	Any
POST /auth/refresh	Required	Any
POST /merchants/register	Required	Any (upgrades to merchant)
GET /merchants/dashboard	Required	Merchant

Deprecated Endpoints

These endpoints return **410 Gone**:

Endpoint	Replacement
`POST /auth/login`	BankID OIDC flow
`POST /auth/register`	Automatic via BankID login
`POST /auth/verify-otp`	Not needed (BankID replaces OTP)

Environment Variables

Required (Production)

```
BANKID_CLIENT_ID          # BankID OIDC client ID
BANKID_CLIENT_SECRET      # BankID OIDC client secret
BANKID_CALLBACK_URL       # Web callback URL (e.g.,
https://getdrop.no/api/auth/bankid/callback)
BANKID_CALLBACK_URL_MOBILE # Mobile deep link (e.g., drop://auth/callback)
JWT_SECRET                # JWT signing secret (min 32 chars)
```

Optional

```
BANKID_AUTHORIZE_URL      # Default: BankID prod authorize endpoint
BANKID_TOKEN_URL          # Default: BankID prod token endpoint
BANKID_JWKS_URL           # Default: BankID prod JWKS endpoint
BANKID_ISSUER             # Default: BankID prod issuer
BANKID_MOCK=true          # Dev mode: mock OIDC flow (no real BankID needed)
JWT_ALGORITHM             # "HS256" (default) or "RS256"
JWT_EXPIRY                # Default: "24h"
```

Merchant Flow

Merchants use the same BankID login as regular users. After logging in:

1. Navigate to merchant registration
2. Fill in business details (business name, org number, bank account)
3. `POST /merchants/register` with auth token
4. User role upgraded from `user` to `merchant`
5. Merchant dashboard becomes accessible