# Module Design

# Module Design Document

> **Project:** {{PROJECT_NAME}} **Module:** {{MODULE_NAME}} **Service:** {{SERVICE_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. Module Overview & Responsibility

**Module:** `{{MODULE_NAME}}` **Layer:** Domain | Application | Infrastructure | Presentation **Repository:** `{{REPO_OR_MONOREPO_PATH}}` **Team Owner:** {{TEAM_NAME}}

**Single Responsibility Statement:**

> {{THE_MODULE_IS_RESPONSIBLE_FOR_ONE_THING}}

**This module owns:**

- {{OWNED_RESOURCE_1}} (data + business logic)
- {{OWNED_RESOURCE_2}}

**This module does NOT own:**

- {{NOT_OWNED_1}} — owned by `{{OTHER_MODULE}}`

- {{NOT_OWNED_2}} — owned by `{{OTHER_MODULE}}`

**Why this is a separate module:** {{RATIONALE_FOR_SEPARATION}} — e.g., "Separate bounded context, different team ownership, different scaling requirements"

---

# 2. Interface Definition (Public API)

## 2.1 Exported Service Interface

```
// Public interface exported by this module
export interface I{{ModuleName}}Service {
  /**
   * {{METHOD_1_DESCRIPTION}}
   * @throws {ValidationError} if dto is invalid
   * @throws {ConflictError} if {{UNIQUE_FIELD}} already exists
   */
  create(dto: Create{{Entity}}Dto, context: RequestContext): Promise<{{Entity}}>;

  /**
   * {{METHOD_2_DESCRIPTION}}
   * @throws {NotFoundError} if not found
   */
  findById(id: string, context: RequestContext): Promise<{{Entity}}>;

  /**
   * {{METHOD_3_DESCRIPTION}}
   */
  findAll(filter: {{Filter}}Dto, context: RequestContext):
Promise<PaginatedResult<{{Entity}}>>;

  /**
   * {{METHOD_4_DESCRIPTION}}
   * @throws {NotFoundError} if not found
   * @throws {ForbiddenError} if user lacks permission
   */
  update(id: string, dto: Update{{Entity}}Dto, context: RequestContext): Promise<{{Entity}}>;

  /**
```

```
   * {{METHOD_5_DESCRIPTION}}
   * @throws {NotFoundError} if not found
   */
  delete(id: string, context: RequestContext): Promise<void>;
}

// DTOs exported for consumers
export type Create{{Entity}}Dto = { /* ... */ };
export type Update{{Entity}}Dto = { /* ... */ };
export type {{Filter}}Dto = { /* ... */ };
export type {{Entity}} = { /* ... */ };
```

# 2.2 HTTP Endpoints (if applicable)

| Method | Path | Auth | Description |
|---|---|---|---|
| `POST` | `/api/v{{V}}/{{resource}}` | JWT | Create {{entity}} |
| `GET` | `/api/v{{V}}/{{resource}}` | JWT | List {{entities}} |
| `GET` | `/api/v{{V}}/{{resource}}/:id` | JWT | Get by ID |
| `PUT` | `/api/v{{V}}/{{resource}}/:id` | JWT | Full update |
| `PATCH` | `/api/v{{V}}/{{resource}}/:id` | JWT | Partial update |
| `DELETE` | `/api/v{{V}}/{{resource}}/:id` | JWT | Soft delete |

# 2.3 Events Published

| Event | Topic/Queue | Schema | Triggered By |
|---|---|---|---|
| `{{entity}}.created` | `{{TOPIC}}` | See §5 | POST endpoint |
| `{{entity}}.updated` | `{{TOPIC}}` | See §5 | PUT/PATCH endpoint |
| `{{entity}}.deleted` | `{{TOPIC}}` | See §5 | DELETE endpoint |
| `{{entity}}.{{CUSTOM_EVENT}}` | `{{TOPIC}}` | See §5 | {{BUSINESS_TRIGGER}} |

---

# 3. Internal Structure

```
{{MODULE_NAME}}/
├── controllers/
│   └── {{entity}}.controller.ts      # HTTP request handling, input parsing
├── services/
│   └── {{entity}}.service.ts         # Business logic
├── repositories/
│   ├── {{entity}}.repository.ts      # Data access interface
│   └── {{entity}}.repository.pg.ts   # PostgreSQL implementation
├── domain/
│   ├── {{entity}}.entity.ts          # Domain entity / value objects
│   ├── {{entity}}.events.ts          # Domain events
│   └── {{entity}}.errors.ts          # Domain-specific errors
├── dto/
│   ├── create-{{entity}}.dto.ts
│   ├── update-{{entity}}.dto.ts
│   └── {{entity}}-filter.dto.ts
├── mappers/
│   └── {{entity}}.mapper.ts          # DB record ↔ domain entity ↔ DTO
├── __tests__/
│   ├── unit/
│   └── integration/
└── {{entity}}.module.ts              # Module registration / DI wiring
```

**Layer rules (enforced by linting):**

- Controllers only call Services (never Repositories directly)
- Services only call Repositories and publish Events
- Domain entities have no framework dependencies
- Mappers live at service layer — not in controllers

---

# 4. Database Schema

## Primary Table: `{{table_name}}`

| Column | Type | Nullable | Default | Constraints | Description |
|---|---|---|---|---|---|
| `id` | `UUID` | NO | `gen_random_uuid()` | PK | Surrogate key |
| `created_at` | `TIMESTAMPTZ` | NO | `NOW()` | | Immutable creation time |

| Column | Type | Nullable | Default | Constraints | Description |
|---|---|---|---|---|---|
| `updated_at` | `TIMESTAMPTZ` | NO | `NOW()` | | Auto-updated on write |
| `deleted_at` | `TIMESTAMPTZ` | YES | `NULL` | | Soft delete marker |
| `version` | `INTEGER` | NO | `1` | | Optimistic lock version |
| `{{FIELD_1}}` | `{{TYPE}}` | {{YES/NO}} | `{{DEFAULT}}` | {{CHECK/UNIQUE/FK}} | {{DESCRIPTION}} |
| `{{FIELD_2}}` | `{{TYPE}}` | {{YES/NO}} | `{{DEFAULT}}` | {{CHECK/UNIQUE/FK}} | {{DESCRIPTION}} |
| `{{FK_ID}}` | `UUID` | NO | | FK → `{{other_table}}(id)` | {{RELATIONSHIP}} |

**Indexes:**

```
CREATE INDEX CONCURRENTLY idx_{{table_name}}_{{column}} ON {{table_name}}({{column}})
    WHERE deleted_at IS NULL;
-- Rationale: {{WHY_THIS_INDEX}}

CREATE UNIQUE INDEX idx_{{table_name}}_{{unique_column}} ON {{table_name}}({{unique_column}})
    WHERE deleted_at IS NULL;
-- Rationale: Enforce uniqueness for active records only
```

**RLS (Row-Level Security):**

```
-- TODO: Enable if multi-tenant
-- ALTER TABLE {{table_name}} ENABLE ROW LEVEL SECURITY;
-- CREATE POLICY {{table_name}}_tenant_isolation ON {{table_name}}
--     USING (tenant_id = current_setting('app.current_tenant_id')::uuid);
```

---

# 5. API Endpoints (Detailed)

## `POST /api/v{{V}}/{{resource}}`

**Request:**

```
{
  "{{field1}}": "string (required, max 255 chars)",
  "{{field2}}": "number (required, > 0)",
  "{{field3}}": "string (optional, enum: [A, B, C])"
}
```

**Success `201`:**

```
{
  "id": "uuid",
  "{{field1}}": "value",
  "{{field2}}": 0,
  "createdAt": "ISO8601"
}
```

**Event published:** `{{entity}}.created`

```
{
  "specversion": "1.0",
  "type": "{{entity}}.created",
  "source": "/{{resource}}",
  "id": "{{EVENT_UUID}}",
  "time": "ISO8601",
  "data": {
    "entityId": "uuid",
    "tenantId": "uuid",
    "createdBy": "uuid",
    "{{field1}}": "value"
  }
}
```

---

# 6. Business Logic Specifications

## 6.1 Business Rules

| Rule ID | Rule | Enforced In | Error |
|---|---|---|---|
| BR-001 | {{RULE_1_DESCRIPTION}} | Service | `{{ERROR_CODE}}` |

| Rule ID | Rule | Enforced In | Error |
|---|---|---|---|
| BR-002 | {{RULE_2_DESCRIPTION}} | Domain Entity | `{{ERROR_CODE}}` |
| BR-003 | {{RULE_3_DESCRIPTION}} | Service + DB constraint | `ConflictError` |

## 6.2 Validation Rules

| Field | Type | Required | Validation | Error Message |
|---|---|---|---|---|
| `{{field1}}` | `string` | Yes | Min 1, Max 255 chars | "{{field1}} must be 1-255 characters" |
| `{{field2}}` | `number` | Yes | Positive integer | "{{field2}} must be a positive integer" |
| `{{field3}}` | `enum` | No | One of: [A, B, C] | "{{field3}} must be A, B, or C" |

## 6.3 Authorization Rules

| Operation | Required Role | Additional Conditions |
|---|---|---|
| Create | `{{ROLE}}` | — |
| Read own | Any authenticated user | `userId === resource.createdBy` |
| Read any | `{{ADMIN_ROLE}}` | — |
| Update | `{{ROLE}}` | `userId === resource.createdBy OR isAdmin` |
| Delete | `{{ADMIN_ROLE}}` | Soft delete only |
| Hard delete | `SUPER_ADMIN` | Requires 2FA confirmation |

---

# 7. Event Publishing / Consuming

## 7.1 Events Published

| Event | When | Payload Schema | Idempotency Key |
|---|---|---|---|
| `{{entity}}.created` | After successful create | `{entityId, tenantId, ...}` | `entityId` |
| `{{entity}}.updated` | After successful update | `{entityId, changes: {...}}` | `entityId + version` |
| `{{entity}}.deleted` | After soft delete | `{entityId, deletedAt}` | `entityId` |

## 7.2 Events Consumed

| Event | Source Module | Handler | Processing Guarantee |
|---|---|---|---|
| `{{other_entity}}.deleted` | `{{OTHER_MODULE}}` | Cascade soft-delete related records | At-least-once |
| `{{other_entity}}.updated` | `{{OTHER_MODULE}}` | Update denormalized cache | At-least-once |

**Idempotency strategy:** All consumers check `processed_events` table before processing. Duplicate events are logged and skipped.

---

# 8. Dependencies

## 8.1 Upstream (what this module depends on)

| Dependency | Type | Coupling | Reason |
|---|---|---|---|
| `AuthModule` | Internal module | Loose (interface) | JWT validation, user context |
| `{{OTHER_MODULE}}` | Internal module | Loose (events) | {{REASON}} |
| PostgreSQL | Infrastructure | Required | Primary storage |
| Redis | Infrastructure | Optional | Caching (degrades gracefully) |
| `{{EXTERNAL_SDK}}` | External library | Hard | {{REASON}} |

## 8.2 Downstream (what depends on this module)

| Consumer | What they use | Notes |
|---|---|---|
| `{{OTHER_MODULE}}` | `{{entity}}.created` events | Read-only consumer |
| `{{FRONTEND}}` | HTTP API | Via API gateway |

---

# 9. Error Handling & Recovery

| Error Scenario | Handling | User Impact | Recovery |
|---|---|---|---|
| DB connection lost | Retry 3x with backoff, then 503 | Request fails gracefully | Auto-recover when DB reconnects |
| External API timeout | Return cached data or 503 | Degraded feature | Async retry, alert on-call |

| Error Scenario | Handling | User Impact | Recovery |
|---|---|---|---|
| Duplicate submission | Detect via unique constraint, return 409 | Clear error message | None needed |
| Invalid state transition | Return 422 with state machine error | Clear error message | User corrects input |
| Event publish failure | Log to retry queue, return 202 | Async delay | Background retry |

---

# 10. Configuration & Feature Flags

## Environment Variables

| Variable | Type | Default | Description |
|---|---|---|---|
| `{{MODULE_NAME}}_CACHE_TTL` | `number` | `300` | Cache TTL seconds |
| `{{MODULE_NAME}}_MAX_LIST_SIZE` | `number` | `100` | Max items per page |
| `{{MODULE_NAME}}_RATE_LIMIT_RPM` | `number` | `60` | Rate limit per minute |

## Feature Flags

| Flag | Type | Default | Description |
|---|---|---|---|
| `{{MODULE_NAME}}_ENABLE_CACHE` | `boolean` | `true` | Toggle Redis caching |
| `{{MODULE_NAME}}_ENABLE_{{FEATURE}}` | `boolean` | `false` | Gradual rollout of {{FEATURE}} |

---

# 11. Monitoring & Health Checks

## Health Check Endpoint

`GET /health/{{module-name}}`

```
{
  "status": "healthy | degraded | unhealthy",
```

```
  "checks": {
    "database": "healthy",
    "cache": "healthy | degraded",
    "externalApi": "healthy | degraded | unhealthy"
  },
  "latency": {
    "database_ms": 5,
    "cache_ms": 1
  }
}
```

# Key Metrics

| Metric | Type | Alert Threshold | Dashboard |
|---|---|---|---|
| `{{module}}_requests_total` | Counter | — | {{DASHBOARD_LINK}} |
| `{{module}}_request_duration_ms` | Histogram | p99 > {{THRESHOLD}}ms | {{DASHBOARD_LINK}} |
| `{{module}}_errors_total` | Counter | Error rate > {{THRESHOLD}}% | {{DASHBOARD_LINK}} |
| `{{module}}_cache_hit_rate` | Gauge | < {{THRESHOLD}}% for 5min | {{DASHBOARD_LINK}} |
| `{{module}}_db_pool_exhausted` | Counter | Any occurrence | {{DASHBOARD_LINK}} |

---

# 12. Primary Flow — Sequence Diagram

```
sequenceDiagram
    autonumber
    participant C as Controller
    participant S as Service
    participant V as Validator
    participant R as Repository
    participant DB as PostgreSQL
    participant EB as Event Bus
    participant Cache as Redis

    C->>V: validate(dto)
    alt Invalid input
```

```
        V-->>C: ValidationError
        C-->>Client: 400 Bad Request
    end
    V-->>C: Validated DTO

    C->>S: create(dto, context)
    S->>S: checkBusinessRules(dto)
    alt Business rule violation
        S-->>C: BusinessRuleError
        C-->>Client: 422 Unprocessable
    end

    S->>DB: BEGIN TRANSACTION
    S->>R: create(entityData)
    R->>DB: INSERT INTO {{table_name}}
    DB-->>R: Inserted record
    R-->>S: {{Entity}} domain object

    S->>DB: COMMIT

    S->>EB: publish("{{entity}}.created", event)
    Note over EB: Async — does not block response

    S->>Cache: INVALIDATE related keys
    S-->>C: {{Entity}} DTO
    C-->>Client: 201 Created
```

# Approval

| Role | Name | Date | Signature |
|---|---|---|---|
| Author | | | |
| Module Owner | | | |
| Tech Lead | | | |
| Reviewer | | | |

Revision #5
Created 2026-02-24 14:52:28 UTC by John
Updated 2026-06-28 20:03:39 UTC by John