Service Design

Service Design Document

> **Project:** {{PROJECT_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. Service Overview

Property	Value
Service name	`{{service-name}}`
Bounded context	`{{Domain / bounded context name}}`
Repository	`{{https://github.com/org/service-name}}`
Owner team	`{{Team Name}}`
On-call	`{{PagerDuty rotation / team contact}}`
Runbook	`{{https://wiki.domain.com/runbooks/service-name}}`
Tech stack	`{{Node.js 20 + NestJS + PostgreSQL + Redis}}`

Purpose:

> TODO: 2-3 sentences. What does this service do? What business capability does it own? What is explicitly OUT of scope?

Bounded context: This service owns the **{{DOMAIN}}** domain. It is the single source of truth for `{{entities owned}}`. Other services must NOT directly access this service's database — they must call its API or subscribe to its events.

---

2. Service Responsibility & Ownership

This service IS responsible for:

- `{{Primary responsibility 1}}`
- `{{Primary responsibility 2}}`
- `{{Primary responsibility 3}}`

This service is NOT responsible for:

- `{{Out-of-scope concern 1 — handled by service X}}`
- `{{Out-of-scope concern 2}}`

Data ownership:

- Owns: `{{users, user_profiles, user_preferences tables}}`
- Does NOT own: `{{orders (belongs to order-service)}}`

---

3. Interface Definition

3.1 REST API Endpoints

Method	Path	Description	Auth
`GET`	`/{{service}}/health`	Health check	None
`GET`	`/{{service}}/{{resource}}`	List `{{resources}}`	Bearer JWT
`GET`	`/{{service}}/{{resource}}/:id`	Get by ID	Bearer JWT
`POST`	`/{{service}}/{{resource}}`	Create	Bearer JWT
`PATCH`	`/{{service}}/{{resource}}/:id`	Update	Bearer JWT
`DELETE`	`/{{service}}/{{resource}}/:id`	Delete	Bearer JWT

Internal endpoints (service-to-service only, no external access):

Method	Path	Description	Auth
`GET`	`/internal/{{resource}}/:id`	Bulk lookup by IDs	Service API key

Full API reference: See `api-reference.md` or `{{OpenAPI URL}}`

3.2 gRPC Service Definition (if applicable)

```
// proto/{{service_name}}.proto
syntax = "proto3";

package {{service_name}};

service {{ServiceName}}Service {
  rpc Get{{Resource}} (Get{{Resource}}Request) returns ({{Resource}});
  rpc List{{Resources}} (List{{Resources}}Request) returns (List{{Resources}}Response);
  rpc Create{{Resource}} (Create{{Resource}}Request) returns ({{Resource}});
}

message {{Resource}} {
  string id = 1;
  string name = 2;
  string created_at = 3;
}

message Get{{Resource}}Request {
  string id = 1;
}
```

TODO: Remove or populate gRPC section based on actual communication protocol.

3.3 Events Published

Event Type	Trigger	Topic / Queue	Consumer(s)
`{{domain}}.{{entity}}.created`	Entity created	`{{topic-name}}`	`{{service-a, service-b}}`
`{{domain}}.{{entity}}.updated`	Entity updated	`{{topic-name}}`	`{{service-a}}`
`{{domain}}.{{entity}}.deleted`	Soft delete	`{{topic-name}}`	`{{service-b}}`

Example published event:

```
{
  "specversion": "1.0",
  "type": "{{domain}}.{{entity}}.created",
  "source": "{{service-name}}",
  "id": "evt_01HX7...",
  "time": "2024-01-15T10:30:00Z",
  "datacontenttype": "application/json",
  "data": {
    "id": "{{UUID}}",
    "{{field}}": "{{value}}"
  }
}
```

Full event schemas: See `event-schema-documentation.md`

3.4 Events Consumed

Event Type	Source Service	Handler Action
`{{domain}}.{{entity}}.created`	`{{source-service}}`	`{{Action this service takes}}`
`{{domain}}.{{entity}}.deleted`	`{{source-service}}`	`{{Action — e.g., cascade delete}}`

Consumer group: `{{service-name}}-consumer` **Idempotency:** All handlers are idempotent (duplicate events produce same result).

---

4. Database

4.1 Technology & Rationale

Property	Value
Database	`{{PostgreSQL 16}}`
ORM	`{{Prisma 5}}`
Rationale	`{{Why this DB was chosen}}`
Hosting	`{{AWS RDS / Supabase / Self-hosted}}`
Replication	`{{1 primary + 2 read replicas}}`

Property	Value
Backup	`{{Daily snapshot + WAL archiving}}`
Encryption	At rest and in transit

4.2 Schema Overview

```
erDiagram
    USERS {
        uuid id PK
        string email UK
        string name
        string role
        string status
        timestamp created_at
        timestamp updated_at
        timestamp deleted_at
    }

    USER_PROFILES {
        uuid id PK
        uuid user_id FK
        string avatar_url
        string bio
        jsonb settings
        timestamp updated_at
    }

    USER_SESSIONS {
        uuid id PK
        uuid user_id FK
        string refresh_token_hash
        string ip_address
        timestamp expires_at
        timestamp created_at
    }

    USERS ||--o| USER_PROFILES : has
    USERS ||--o{ USER_SESSIONS : has
```

TODO: Update schema to reflect actual tables. Add missing tables.

4.3 Data Ownership Boundaries

- **Read access:** Any service may query via this service's API
- **Write access:** ONLY this service writes to its tables
- **Direct DB access:** FORBIDDEN for all other services

Cross-service data pattern:

```
Service A needs user name:
  → GET /users/:id via HTTP (NOT direct DB query)
  → Or subscribe to user.updated events and cache locally
```

---

5. Dependencies

5.1 Upstream Services (Services This Depends On)

Service	Purpose	Criticality	Fallback
`{{auth-service}}`	JWT validation	Critical	Cache valid tokens 5 min
`{{notification-service}}`	Send emails	Non-critical	Queue for retry
`{{{{EXTERNAL_API}}}}`	`{{Purpose}}`	`{{Critical/Non-critical}}`	`{{Fallback strategy}}`

5.2 Downstream Services (Services That Depend On This)

Service	How it uses this service	Impact if this service is down
`{{order-service}}`	Validate user exists before creating order	Cannot create orders
`{{notification-service}}`	Resolve user email for delivery	Cannot send user notifications

---

5.3 External APIs & Third-Party

Service	Purpose	Rate Limit	Credentials
`{{SendGrid}}`	Transactional email	100 req/s	Vault: `sendgrid/api-key`
`{{Stripe}}`	Payment processing	—	Vault: `stripe/secret-key`

5.4 Dependency Diagram

```
graph LR
    ThisService["{{service-name}}"]

    subgraph "Upstream (depends on)"
        AuthService["auth-service"]
        ExternalAPI["external-api"]
    end

    subgraph "Downstream (depended on by)"
        OrderService["order-service"]
        NotifService["notification-service"]
    end

    AuthService --> ThisService
    ExternalAPI --> ThisService
    ThisService --> OrderService
    ThisService --> NotifService
```

6. Deployment Configuration

Property	Dev	Staging	Production
Replicas	1	2	`{{min: 3, max: 10}}`
CPU request	100m	250m	500m
CPU limit	500m	1000m	2000m
Memory request	128Mi	256Mi	512Mi
Memory limit	512Mi	1Gi	2Gi
Port	4000	4000	4000

Kubernetes manifest location: `{{k8s/{{service-name}}/}}` **Helm chart:** `{{charts/{{service-name}}/}}` **Docker image:** `{{registry.domain.com/service-name}}`

---

7. Scaling Strategy

Dimension	Strategy	Trigger
Horizontal (replicas)	HPA: CPU > 70% OR RPS > 1000	Automatic
Vertical (resources)	VPA recommendations reviewed monthly	Manual
Database	Read replicas for SELECT queries	Manual
Cache	Redis Cluster when > 10GB RAM	Manual

Stateless confirmation: This service stores NO session state in memory — safe to scale horizontally.

---

8. Health Check & Readiness Probes

```
livenessProbe:
  httpGet:
    path: /health/live
    port: 4000
  initialDelaySeconds: 30
  periodSeconds: 10
  failureThreshold: 3

readinessProbe:
  httpGet:
    path: /health/ready
    port: 4000
  initialDelaySeconds: 10
  periodSeconds: 5
  failureThreshold: 3

startupProbe:
  httpGet:
    path: /health/startup
```

```
    port: 4000
  failureThreshold: 30
  periodSeconds: 10
```

---

9. SLA Commitments

Metric	Target	Measurement Window
Availability	99.9% (8.7h downtime/year)	Rolling 30 days
P50 response time	< 50ms	1 hour
P95 response time	< 200ms	1 hour
P99 response time	< 500ms	1 hour
Error rate (5xx)	< 0.1%	1 hour

SLA breach escalation: Alert → PagerDuty `{{on-call rotation}}` → Incident declared at SLA breach risk.

---

10. Monitoring & Alerting Rules

Metric	Threshold	Alert Severity	Channel
Error rate (5xx)	> 1% for 5 min	P1	PagerDuty
P99 latency	> 1s for 5 min	P2	Slack `#alerts`
CPU utilization	> 85% for 10 min	P3	Slack `#alerts`
Memory utilization	> 80%	P3	Slack `#alerts`
DB connection pool	> 80%	P2	PagerDuty
Queue depth	> 10,000 items	P2	Slack `#alerts`

Dashboard: `{{https://monitoring.domain.com/dashboards/service-name}}`

---

11. Runbook Reference

Runbook location: `{{https://wiki.domain.com/runbooks/{{service-name}}}}`

Quick reference for common incidents:

Incident	Initial Response
High error rate	Check logs → identify error pattern → scale up if OOM
High latency	Check DB slow query log → check Redis hit rate → check upstream dependency
Pod crash loop	Check OOMKilled → check logs → check health probe thresholds
DB connection exhaustion	Check pool config → check idle connections → force disconnect

Approval

Role	Name	Date	Signature
Author			
Service Owner			
Architect			
SRE Lead			

Revision #3
Created 2026-02-24 14:53:26 UTC by John
Updated 2026-05-25 07:33:29 UTC by John