

Logical & Data Architecture

[Author Name]

Université de Toulon, LIS UMR CNRS 7020

2026-10-14

Table des matières

- 1 Purpose of this Document 1
- 2 Architectural Layers 1
- 3 Architectural Principles 1
- 4 Logical Component Model 2
- 5 Data Models 2
 - 5.1 Conceptual Data Model 2
 - 5.2 Logical Data Model 2
- 6 Layer Dependencies 3
- 7 Component Organization 3
- 8 Interface Definitions 4
- 9 Quality Attributes 4
- 10 Data Architecture 4
- 11 Data Flow 5
- 12 Data Privacy & Governance 5
- 13 Template Validation Checklist 5
- 14 Related Views 5

1 Purpose of this Document

This document describes the logical organization of the [System Name] system and its associated data architecture, providing a foundation for technical implementation decisions.

2 Architectural Layers

Application Layers:

Presentation:

Type: [UI framework, e.g., React, Angular]

Pattern: [MVC, MVVM, Flux, etc.]

Components: [Key UI components]

Business:

Type: [Service pattern, e.g., Microservices, Domain Services]

Style: [Architectural style, e.g., REST, Event-Driven]

Key Services: [Core business services]

Data:

Type: [Storage pattern, e.g., RDBMS, NoSQL]

Model: [Data model approach, e.g., Relational, Document]

Persistence: [Storage mechanism]

3 Architectural Principles

Principle	Description	Rationale
[Name]	[Brief description]	[Why this principle matters]
[Name]	[Brief description]	[Why this principle matters]
[Name]	[Brief description]	[Why this principle matters]

4 Logical Component Model

```
graph TD
    subgraph "Client Applications"
        A[Web Client]
        B[Mobile Client]
        C[Admin Portal]
    end

    subgraph "API Layer"
        D[API Gateway]
        E[BFF Services]
        F[Authentication]
    end

    subgraph "Business Services"
        G[Service 1]
        H[Service 2]
        I[Service 3]
    end

    subgraph "Data Layer"
        J[Repositories]
        K[Data Access]
        L[Caching]
    end

    A --> D
    B --> E
    C --> D
    D --> F
    D --> G
    E --> G
    E --> H
    G --> J
    H --> J
    I --> K
    J --> L
```

5 Data Models

5.1 Conceptual Data Model

```
erDiagram
    CUSTOMER ||--o{ ORDER : places
    ORDER ||--|{ LINE_ITEM : contains
    PRODUCT ||--o{ LINE_ITEM :
        "ordered in"

    CUSTOMER {
        string id
        string name
        string email
    }

    ORDER {
        string id
        date created_at
        string status
    }

    LINE_ITEM {
        string id
        int quantity
        decimal price
    }

    PRODUCT {
        string id
        string name
        decimal price
    }
```

5.2 Logical Data Model

```
@startuml
entity "Customer" as customer {
    * customer_id : UUID <<PK>>
    --
    * name : VARCHAR(100)
    * email : VARCHAR(100) <<unique>>
    * created_at : TIMESTAMP
    status : VARCHAR(20)
}
```

```

}

entity "Order" as order {
    * order_id : UUID <<PK>>
    --
    * customer_id : UUID <<FK>>
    * created_at : TIMESTAMP
    * status : VARCHAR(20)
    shipping_address : VARCHAR(200)
    total_amount : DECIMAL(10,2)
}

entity "LineItem" as lineitem {
    * line_item_id : UUID <<PK>>
    --
    * order_id : UUID <<FK>>
    * product_id : UUID <<FK>>
    * quantity : INT
    * unit_price : DECIMAL(10,2)
    total_price : DECIMAL(10,2)
}

entity "Product" as product {
    * product_id : UUID <<PK>>
    --
    * name : VARCHAR(100)
    * price : DECIMAL(10,2)
    description : TEXT
    category : VARCHAR(50)
    inventory_count : INT
}

customer ||--o{ order
order ||--|{ lineitem
product ||--o{ lineitem
@enduml

```

6 Layer Dependencies

```

@startuml
package "Presentation" {
    [UI Components]
    [View Models]

```

```

    [Navigation]
}

package "Business" {
    [Domain Services]
    [Business Rules]
    [Application Services]
}

package "Data" {
    [Repositories]
    [Data Access]
    [Caching]
}

[UI Components] --> [View Models]
[View Models] --> [Application Services]
[Application Services] --> [Domain Services]
[Domain Services] --> [Repositories]
[Repositories] --> [Data Access]
@enduml

```

7 Component Organization

Core Components:

- Name: [Component name]
- Purpose: [What it does]
- Dependencies: [What it relies on]
- Pattern: [Design pattern used]
- API Surface: [Primary interfaces]

Additional components as needed

Cross-Cutting:

- Security:
 - Responsibility: [What it handles]
 - Approach: [Implementation strategy]
- Logging:
 - Approach: [Implementation strategy]

```

strategy]
    Key Events: [What gets logged]

- Monitoring:
    Metrics: [Key measurements]
    Alerts: [When to alert]

- Configuration:
    Sources: [Config sources]
    Refresh: [Update strategy]

```

8 Interface Definitions

```

Internal APIs:
- Service: [Service name]
  Type: [Synchronous/Asynchronous]
  Protocol: [REST, GraphQL, gRPC,
etc.]
  Purpose: [What it does]
  Key Operations:
    - [Operation 1]
    - [Operation 2]
  Consumers: [Who uses it]

External APIs:
- Interface: [API name]
  Version: [Current version]
  Pattern: [REST, GraphQL, etc.]
  Contract: [Link to specification]
  Authentication: [Method]
  Rate Limits: [Constraints]

```

9 Quality Attributes

```

Performance:
  Response Time: [Target, e.g., "95%
of requests < 200ms"]
  Throughput: [Target, e.g., "1000
TPS"]
  Scalability: [Approach, e.g.,
"Horizontal scaling up to 100
instances"]

```

```

Reliability:
  Availability: [Target, e.g., "99.9%
uptime"]
  Fault Tolerance: [Strategy, e.g.,
"Circuit breaker pattern"]
  Recovery: [Plan, e.g., "Automated
failover within 30 seconds"]

Security:
  Authentication: [Method, e.g.,
"OAuth 2.0 with OIDC"]
  Authorization: [Approach, e.g.,
"RBAC"]
  Data Protection: [Standards, e.g.,
"Encryption in transit and at rest"]

```

10 Data Architecture

```

Domain Models:
- Entity: [Entity name]
  Purpose: [Business purpose]
  Attributes: [Key fields]
  Relations: [Connections to other
entities]
  Volume: [Expected data volume]
  Growth: [Expected growth rate]

Storage Types:
  Operational:
    - Type: [Database type, e.g.,
"PostgreSQL"]
    Purpose: [Use case]
    Scale: [Size/operations]
    Retention: [Data retention
policy]
    Backup: [Backup strategy]

  Analytical:
    - Type: [Database type, e.g.,
"Snowflake"]
    Purpose: [Use case]
    Scale: [Size/operations]

```

ETL: [Data loading approach]
Access: [Query patterns]

Requirements: [Key requirements]
Implementation: [How addressed]

11 Data Flow

```
@startuml
database "Operational DB" as odb
database "Analytics DB" as adb
queue "Message Queue" as mq
component "ETL Pipeline" as etl
component "Data API" as api

api --> odb : CRUD operations
odb --> mq : Change data capture
mq --> etl : Batch processing
etl --> adb : Data loading
adb --> api : Reporting queries
@enduml
```

12 Data Privacy & Governance

```
Data Classification:
- PII:
    Examples: [List of PII data elements]
    Handling: [Special procedures]
    Encryption: [Approach]

- Sensitive:
    Examples: [List of sensitive data elements]
    Handling: [Special procedures]
    Access: [Restrictions]

Compliance Requirements:
- Regulation: [e.g., GDPR, HIPAA]
  Requirements: [Key requirements]
  Implementation: [How addressed]

- Regulation: [Additional regulations]
```

13 Template Validation Checklist

- ☐ Architectural layers defined
- ☐ Logical component model established
- ☐ Data models documented
- ☐ Component dependencies mapped
- ☐ Interface definitions provided
- ☐ Quality attributes specified
- ☐ Data flows described
- ☐ Data privacy/governance addressed
- ☐ Team review completed

14 Related Views

- System Context - System boundaries
- Component Architecture - Detailed component design
- Infrastructure Architecture - Deployment view