

Logical Architecture View

[Author Name]

Université de Toulon, LIS UMR CNRS 7020

2026-10-03

Table des matières

1 Purpose of this Document	1
2 System Structure	1
3 Key Domain Concepts	1
4 Architecture Layers	2
5 Functional Decomposition	2
6 Module Organization	2
7 Key Interfaces	2
8 System Behaviors	3
9 Data Flow Overview	3
10 Architectural Decisions	3
11 Quality Attributes	3
12 Template Validation Checklist	3
13 Related Views	3

Pattern: [Design pattern]
Responsibility: [Key functional area]

Backend:

Type: [Technology/framework]
Pattern: [Design pattern]
Responsibility: [Key functional area]

Data:

Type: [Technology approach]
Stores: [Key data repositories]
Patterns: [Data access patterns]

1 Purpose of this Document

This document provides a logical view of the [System Name] architecture, focusing on the system's functional structure and organization independent of technical implementation details.

2 System Structure

Architectural Style: [Layered|Microservices|Monolithic|etc.]
Primary Patterns: [List of key architectural patterns used]

Components:

Frontend:

Type: [Technology/framework]

3 Key Domain Concepts

Core Entities:

- Name: [Entity name]
Purpose: [Brief description]
Relationships: [How it relates to other entities]

- Name: [Entity name]

Purpose: [Brief description]

Relationships: [How it relates to other entities]

Business Processes:

- Name: [Process name]

Steps: [Key process steps]

Actors: [Who/what is involved]
Data Flow: [Key data elements]

4 Architecture Layers

```
graph TD
    subgraph "Frontend Layer"
        A[Web App]
        B[Mobile App]
    end

    subgraph "Backend Layer"
        C[API Gateway]
        D[Services]
    end

    subgraph "Data Layer"
        E[Primary DB]
        F[Cache]
    end

    A --> C
    B --> C
    C --> D
    D --> E
    D --> F
```

5 Functional Decomposition

```
graph TD
    A[System] --> B[Function 1]
    A --> C[Function 2]
    A --> D[Function 3]

    B --> B1[Sub-function 1.1]
    B --> B2[Sub-function 1.2]

    C --> C1[Sub-function 2.1]
    C --> C2[Sub-function 2.2]

    D --> D1[Sub-function 3.1]
    D --> D2[Sub-function 3.2]
```

6 Module Organization

```
@startuml
package "Core Modules" {
    [Security]
    [Logging]
    [Configuration]
}

package "Feature Modules" {
    [Feature A]
    [Feature B]
    [Feature C]
}

package "Support Modules" {
    [Utilities]
    [Common]
    [Testing]
}

[Feature A] --> [Core Modules]
[Feature B] --> [Core Modules]
[Feature C] --> [Core Modules]
[Feature A] --> [Support Modules]
@enduml
```

7 Key Interfaces

External Interfaces:

- Name: [Interface name]
Purpose: [What it does]
Exposed To: [Which systems/users]
Method: [How it's accessed]

Internal Interfaces:

- Name: [Interface name]
Purpose: [What it does]
Provider: [Which component provides it]
Consumers: [Which components use it]
Style: [Interface style]

8 System Behaviors

```
@startuml
actor User
participant "Frontend" as FE
participant "Backend Service" as BE
database "Database" as DB

User -> FE: Request action
activate FE
FE -> BE: API call
activate BE
BE -> DB: Query data
activate DB
DB --> BE: Return data
deactivate DB
BE --> FE: Response
deactivate BE
FE --> User: Display result
deactivate FE
@enduml
```

9 Data Flow Overview

```
graph LR
    A[Input] --> B[Process 1]
    B --> C[Process 2]
    C --> D[Storage]
    D --> E[Process 3]
    E --> F[Output]
    B --> G[External System]
    G --> E
```

10 Architectural Decisions

Decision	Rationale	Alternatives Considered
[Decision]	[Why this approach]	[Other options]
[Decision]	[Why this approach]	[Other options]

11 Quality Attributes

```
Performance:
- Requirement: [Specific requirement]
  Strategy: [How architecture supports this]

Scalability:
- Requirement: [Specific requirement]
  Strategy: [How architecture supports this]

Security:
- Requirement: [Specific requirement]
  Strategy: [How architecture supports this]

Maintainability:
- Requirement: [Specific requirement]
  Strategy: [How architecture supports this]
```

12 Template Validation Checklist

- ☐ System structure documented
- ☐ Component relationships defined
- ☐ Module boundaries established
- ☐ Layer responsibilities specified
- ☐ Key interfaces identified
- ☐ System behaviors described
- ☐ Data flows mapped
- ☐ Quality attributes addressed
- ☐ Team review completed

13 Related Views

- System Context - System boundaries and external interfaces

- Component Architecture - Detailed component design
- Data Architecture - Data structures and flows