

# Service-Based Architecture

CSSE6400

Richard Thomas

March 21, 2022

### Definition 1. Distributed System

A system with multiple components located on different machines that communicate and coordinate actions in order to appear as a single coherent system to the end-user.

## Quote

A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.

– Leslie Lamport [Turing Award, 2013]

### Definition 2. Service-Based Architecture

System is partitioned into business domains that are deployed as distributed services. Functionality is delivered through a user interface that interacts with the domain services.

# Service-Based Architecture



## Terminology

**User Interface** Provides access to system functionality

**Services** Implement functionality for a single,  
independent business process

**Service APIs** Communication mechanism between UI  
and each service

**Database** Stores persistent data for the system

### Definition 3. API Abstraction Principle

Services should provide an API that hides implementation details.

#### Definition 4. Façade Design Pattern

Provide a simple, abstract interface to use a service domain's functionality. A component within the service coordinates how to deliver the requested functionality with the service's internal components.



### Definition 5. Independent Service Principle

Services should be independent, with no dependencies on other services.

### Question

What are the consequences of having a shared database?

### Question

What are the consequences of having a shared database?

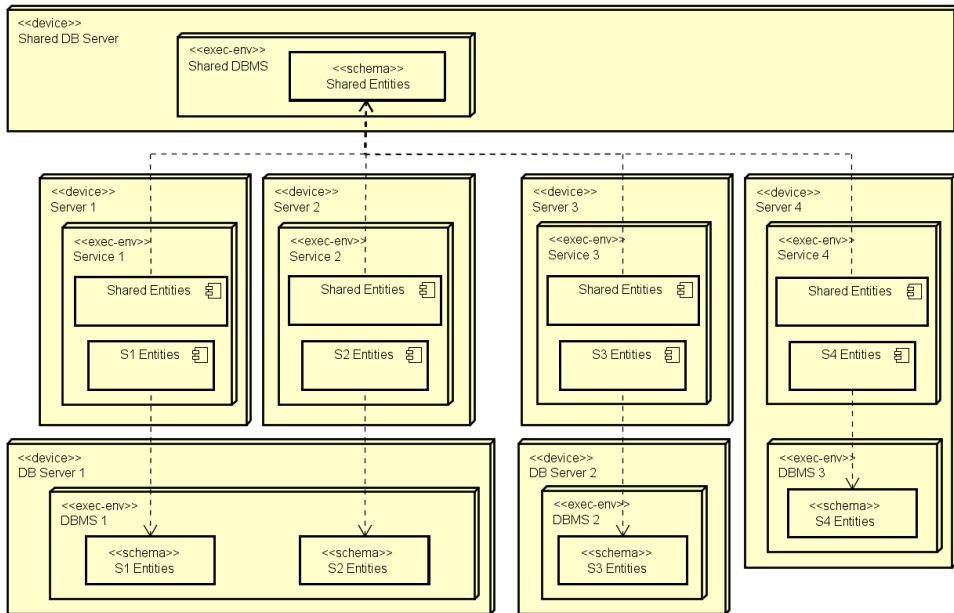
### Answer

Increased *data coupling*.

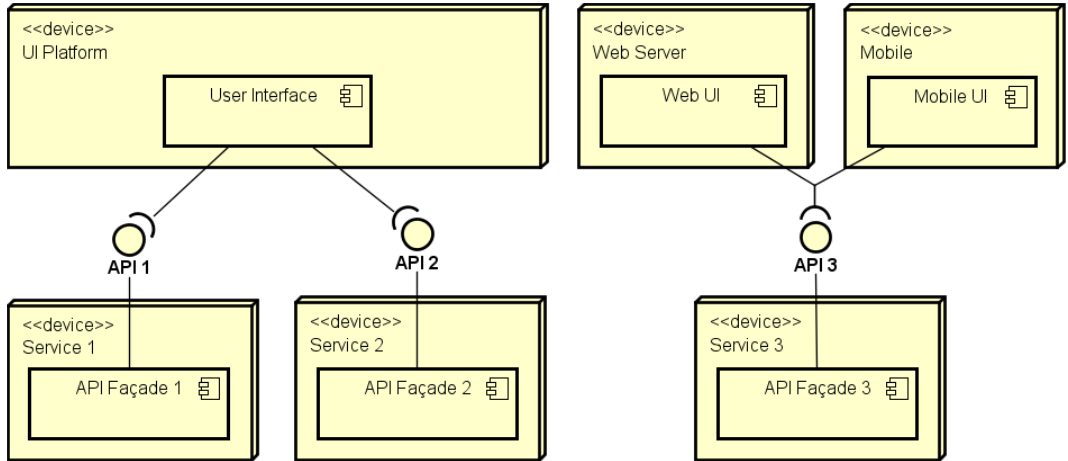
# Logical Partitioning of Persistent Data



## Separate Databases



# Separate UIs



# Sahara: Context Diagram



## On-line Store Service Domains

**Browsing** Customers can find products & add to cart

**Purchasing** Customers can purchase products in cart

**Fulfilment** Customers & staff can track order fulfilment

**Account Management** Customers can manage their  
account details

**Inventory Management** Staff can view stock levels and  
order new stock



Partitioning

Services are defined by domain partitioning

## Coarse Services

- Domains are large
  - *Coarse-grained* services
- Each service will have an internal architecture
  - Technical or domain partitioning

# Sahara: On-line Store Container Diagram



# Sahara: Product Browsing Component Diagram



## Product Browsing Service API

**Search** <https://api.sahara.com/v1/search?keywords=...>

**Browse** <https://api.sahara.com/v1/browse?category=...>

**Add to Cart** <https://api.sahara.com/v1/cart>

- JSON to pass data
- JSF action controller handles request

# Sahara: Deployment Diagram



# Sahara: Concurrent Access



Question

What happens if a service goes down?



### Question

What happens if a service goes down?

### Answer

Need to manage timeouts, retries, graceful failure, ...

## Consider Network Failure

If customer tried to add product to cart:

- What happens if Product Browsing didn't receive it?
- What happens if UI didn't get a response?
- What happens if Database wasn't updated?

# API Layer



## API Layer Advantages

- Acts as a reverse proxy or gateway to services
- Hides internal network structure
- Easier to implement *cross-cutting* concerns
  - e.g. security policies
- Allows service discovery
  - Interface to register service
  - Clients can find out what services are available

## Pros & Cons

**Simplicity** For a distributed system



**Modularity** Services



**Extensibility** New services



**Deployability** Independent services



**Testability** Independent services



**Security** API layer



**Reliability** Independent services



**Interoperability** Service APIs



**Scalability** Coarse-grained services

