Service-Based Architecture

Software Architecture

Richard Thomas

March 11, 2024

> *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.

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

User Interface
from Basic Service-Based Architecture
[Container]
Delivers the system's user interface.

User Interface
[Deployment Node]

Service 1 API

Service 2 API

Service 3 API

API Façade 1
from Basic Service-Based Architecture
[Container]

API Façade 2
from Basic Service-Based Architecture
[Container]

API Façade 3
from Basic Service-Based Architecture
[Container]

Characteristic 1
from Basic Service-Based Architecture
[Container]

Characteristic 2
from Basic Service-Based Architecture
[Container]

Characteristic 3
from Basic Service-Based Architecture
[Container]

Characteristic 4
from Basic Service-Based Architecture
[Container]

Characteristic 5
from Basic Service-Based Architecture
[Container]

Characteristic 6
from Basic Service-Based Architecture
[Container]

Service 1
[Deployment Node]

Service 2
[Deployment Node]

Service 3
[Deployment Node]

Store & Retrieve

Store & Retrieve

Store & Retrieve

Store & Retrieve

Store & Retrieve

Store & Retrieve

Database
from Basic Service-Based Architecture
[Container]
Application's database.

DBMS
[Deployment Node]

Database Server
[Deployment Node]

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.

What are the consequences of having a shared
database?

What are the consequences of having a shared database?

Increased *data coupling*.

Logical Partitioning of Persistent Data

Separate Databases

Shared Entities Schema
from Separate Databases Example
[Container]

Shared DBMS
[Deployment Node]

Shared Database Server
[Deployment Node]

Shared Entities
from Separate Databases Example
[Container]
Persistent entities that are shared between services.

Service 1 Entities
from Separate Databases Example
[Container]
Entities that are unique to Service 1.

Service 1
[Deployment Node]

Server 1
[Deployment Node]

Shared Entities
from Separate Databases Example
[Container]
Persistent entities that are shared between services.

Service 2 Entities
from Separate Databases Example
[Container]
Entities that are unique to Service 2.

Service 2
[Deployment Node]

Server 2
[Deployment Node]

Shared Entities
from Separate Databases Example
[Container]
Persistent entities that are shared between services.

Service 3 Entities
from Separate Databases Example
[Container]
Entities that are unique to Service 3.

Service 3
[Deployment Node]

Server 3
[Deployment Node]

Shared Entities
from Separate Databases Example
[Container]
Persistent entities that are shared between services.

Service 4 Entities
from Separate Databases Example
[Container]
Entities that are unique to Service 4.

Service 4
[Deployment Node]

Service 1 Entities Schema
from Separate Databases Example
[Container]

Service 2 Entities Schema
from Separate Databases Example
[Container]

DBMS 1
[Deployment Node]

Database Server 1
[Deployment Node]

Service 3 Entities Schema
from Separate Databases Example
[Container]

DBMS 2
[Deployment Node]

Database Server 2
[Deployment Node]

Service 4 Entities Schema
from Separate Databases Example
[Container]

DBMS 3
[Deployment Node]

Server 4
[Deployment Node]

Separate UIs

Mobile User Interface
from Separate User Interfaces
[Container]
Delivers a mobile interface for users.

Mobile Device
[Deployment Node]

Service 1 API

API Façade 1
from Separate User Interfaces
[Container]

Service 1
[Deployment Node]

Service 1 API

Web User Interface
from Separate User Interfaces
[Container]
Delivers a web interface for users.

Web Server
[Deployment Node]

Service 2 API

API Façade 2
from Separate User Interfaces
[Container]

Service 2
[Deployment Node]

Service 2 API

Separate UIs

Mobile User Interface
from Separate User Interfaces
[Container]
Delivers a mobile interface for users.

Mobile Device
[Deployment Node]

Service 1 API

API Façade 1
from Separate User Interfaces
[Container]

Service 1
[Deployment Node]

Service 2 API

API Façade 2
from Separate User Interfaces
[Container]

Service 2
[Deployment Node]

Service 1 API

Web User Interface
from Separate User Interfaces
[Container]
Delivers a web interface for users.

Web Server
[Deployment Node]

Service 2 API

Admin User Interface
from Separate User Interfaces
[Container]
Delivers the user interface for admin functionality.

Admin UI Platform
[Deployment Node]

Admin Service API

Admin API Façade
from Separate User Interfaces
[Container]

Admin Service
[Deployment Node]

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

- 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

Web Application
[Container: J2EE Server]
Delivers the web front-end for the on-line store.

Find & Retrieve Product Details
[RMI]

Suggest Products
[Component: Java]
Suggest products user may be interested in.

Search for Product
[Component: Java]
Finds products based on search criteria.

Product Browsing Façade
[Component: JSF]
Provides interface to retrieve product information based on search criteria

Browse for Product
[Component: Java]
Finds products based on user navigation.

Shopping Cart Management
[Component: Java]
Add and remove items from user's shopping cart.

Data Mining Service
[Software System]
Collects customer visit data and provides product recommendations.

Application Database
[Container: MySQL]
Stores customer credentials, products and orders.

Get Product Suggestions
[REST API JSON/HTTPS]

Customer Browsing History
[REST API JSON/HTTPS]

Uses

Uses

Uses

Uses

Finds Product
[JPA]

Finds Product
[JPA]

Updates Cart
[JPA]

Customer Browsing History
[REST API JSON/HTTPS]

Record Cart Updates
[REST API JSON/HTTPS]

Product Browsing
[Container]

Product Browsing Service API

Search https://api.sahara.com/v1/search?keywords=...
Browse https://api.sahara.com/v1/browse?category=...
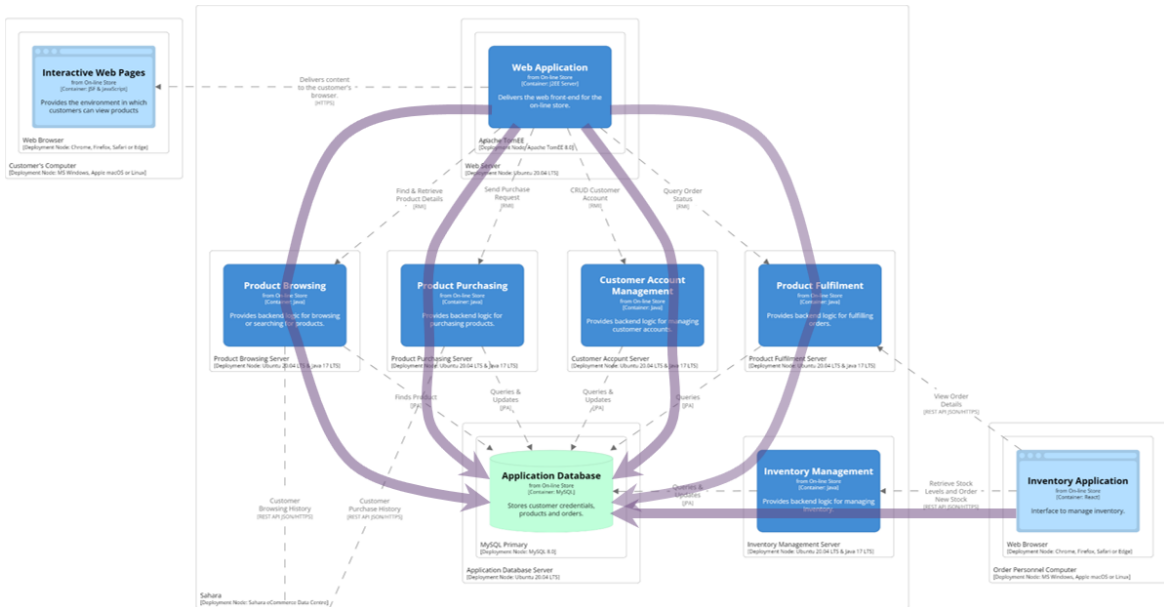**Add to Cart** https://api.sahara.com/v1/cart

- JSON to pass data
- JSF action controller handles request

Sahara: Deployment Diagram

Web Browser
[Deployment Node: Chrome, Firefox, Safari or Edge]

Interactive Web Pages
from On-line Store
[Container: JSP & JavaScript]

Provides the environment in which customers can view products

Customer's Computer
[Deployment Node: MS Windows, Apple macOS or Linux]

Delivers content to the customer's browser.
[HTTPS]

Web Application
from On-line Store
[Container: JEE Server]

Delivers the web front-end for the on-line store.

Apache TomEE
[Deployment Node] Apache TomEE 8.0]

Web Server
[Deployment Node: Ubuntu 20.04 LTS]

Find & Retrieve Product Details
[RMI]

Send Purchase Request
[RMI]

CRUD Customer Account
[RMI]

Query Order Status
[RMI]

Product Browsing
from On-line Store
[Container: Java]

Provides backend logic for browsing or searching for products.

Product Browsing Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]

Product Purchasing
from On-line Store
[Container: Java]

Provides backend logic for purchasing products.

Product Purchasing Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]

Customer Account Management
from On-line Store
[Container: Java]

Provides backend logic for managing customer accounts.

Customer Account Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]

Product Fulfilment
from On-line Store
[Container: Java]

Provides backend logic for fulfilling orders.

Product Fulfilment Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]

Finds Product
[JPA]

Queries & Updates
[JPA]

Queries & Updates
[JPA]

Queries
[JPA]

View Order Details
[REST API JSON/HTTPS]

Customer Browsing History
[REST API JSON/HTTPS]

Customer Purchase History
[REST API JSON/HTTPS]

Application Database
from On-line Store
[Container: MySQL]

Stores customer credentials, products and orders.

MySQL Primary
[Deployment Node: MySQL 8.0]

Application Database Server
[Deployment Node: Ubuntu 20.04 LTS]

Queries & Updates
[JPA]

Inventory Management
from On-line Store
[Container: Java]

Provides backend logic for managing inventory.

Inventory Management Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]

Retrieve Stock Levels and Order New Stock
[REST API JSON/HTTPS]

Inventory Application
from On-line Store
[Container: React]

Interface to manage inventory.

Web Browser
[Deployment Node: Chrome, Firefox, Safari or Edge]

Order Personnel Computer
[Deployment Node: MS Windows, Apple macOS or Linux]

Sahara
[Deployment Node: Sahara eCommerce Data Centre]

Sahara: Concurrent Access

What happens if a service goes down?

Question

What happens if a service goes down?

Answer

Need to manage timeouts, retries, graceful 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 😟