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.

Introduce idea of distributed systems and then move on to service-based being an simple approach.

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.

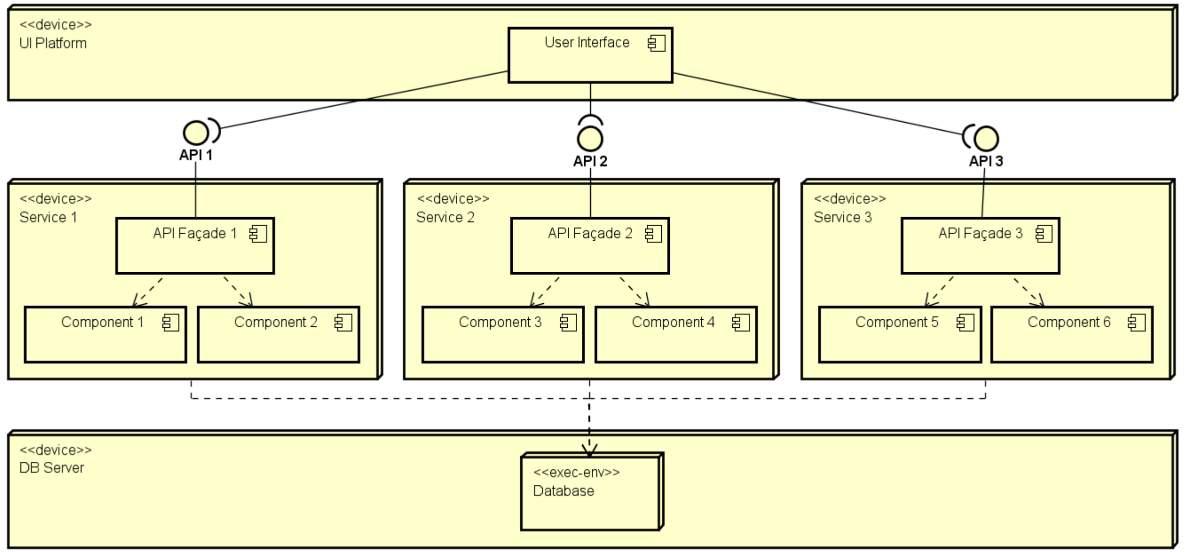
– Leslie Lamport [Turing Award, 2013]

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.

Explain why this leads to a fairly simple distributed architecture.

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

- Explain that the Service APIs are communication protocols and data formats, not just a Java-style interface.
- Usually all Service APIs use the same communication protocol (e.g. REST).
- Also point out that messages between the UI and services will typically be asynchronous.

Definition 3. API Abstraction Principle
Services should provide an API that hides implementation details.

- Each service publishes its own API.
- Hides service implementation details, reducing coupling between UI and service.
- Makes it easier to reuse service across systems or by supporting service (e.g. auditing).

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.

Summarise Façade Design Pattern and how it is used in a service-based architecture. Mention its from the GoF book.

Definition 5. Independent Service Principle

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

- Explain consequences of dependencies between services.
- Services can't easily be deployed separately if they depend on other services.
- They would require interfaces between services, increasing coupling.

Question

What are the consequences of having a shared database?

Question

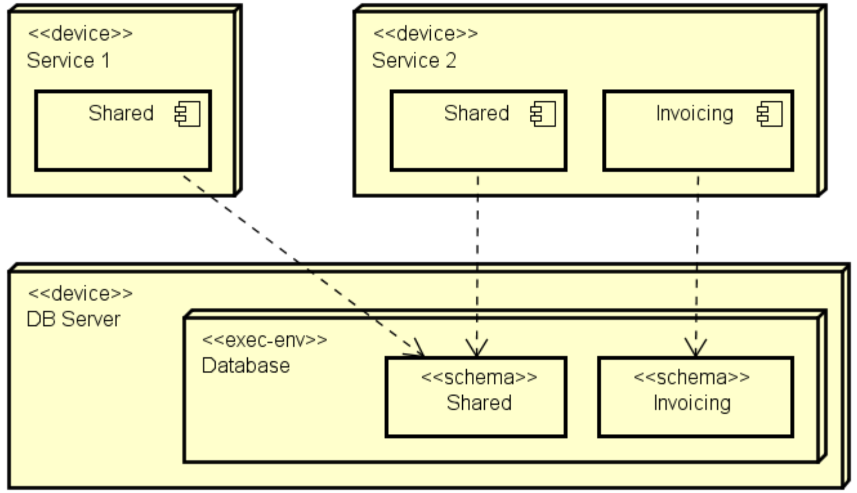
What are the consequences of having a shared database?

Answer

Increased *data coupling*.

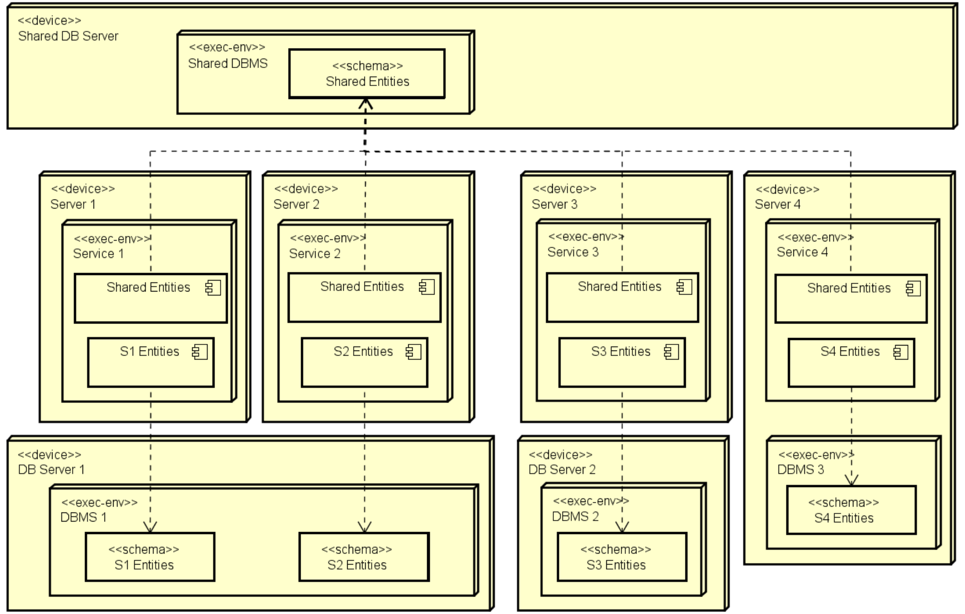
- If a row of a database is locked and another service wants to use it, it is blocked. Losing efficiency benefits of a distributed system.
- If one service changes the structure of its persistent data, all services using that data need to be updated and tested.
- If one service changes how it uses persistent data, all other services using the same data need to be retested.

Logical Partitioning of Persistent Data



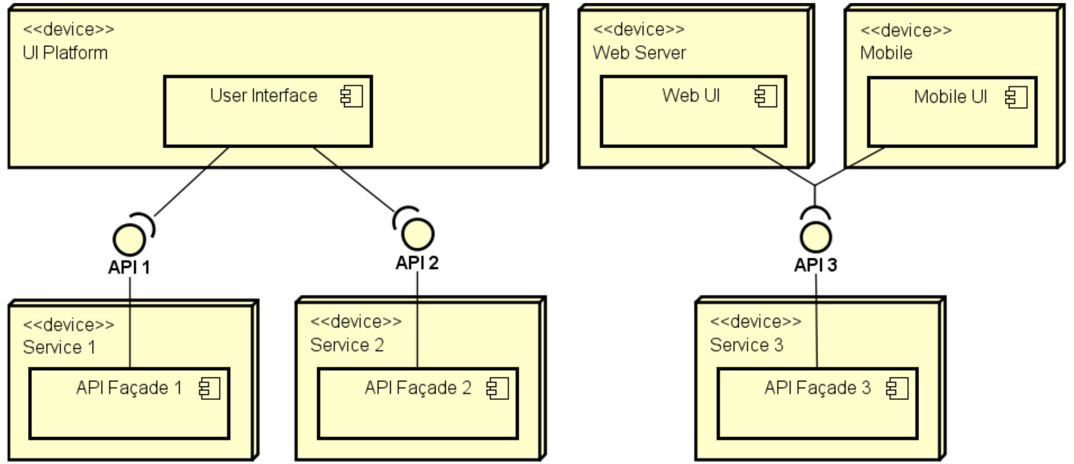
- Define a minimal set of shared persistent objects.
- Create a shared library to access these objects.
- Changes to shared persistent objects are restricted as they require changes to other services.
- Each service may have its own persistent objects stored in tables that are not shared with other services.

Separate Databases



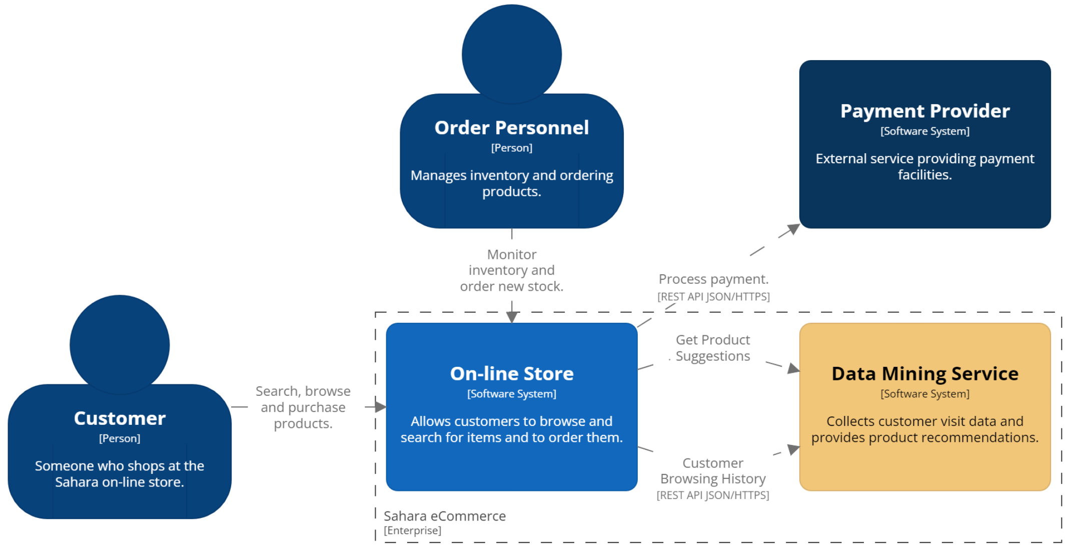
Discuss options of separate DB servers, share DBs on one server, DBs embedded in application.

Separate UIs



- UI Platform could be desktop, web or mobile app.
- This allows multiple concurrent users, even through one user interface.

Sahara: Context Diagram



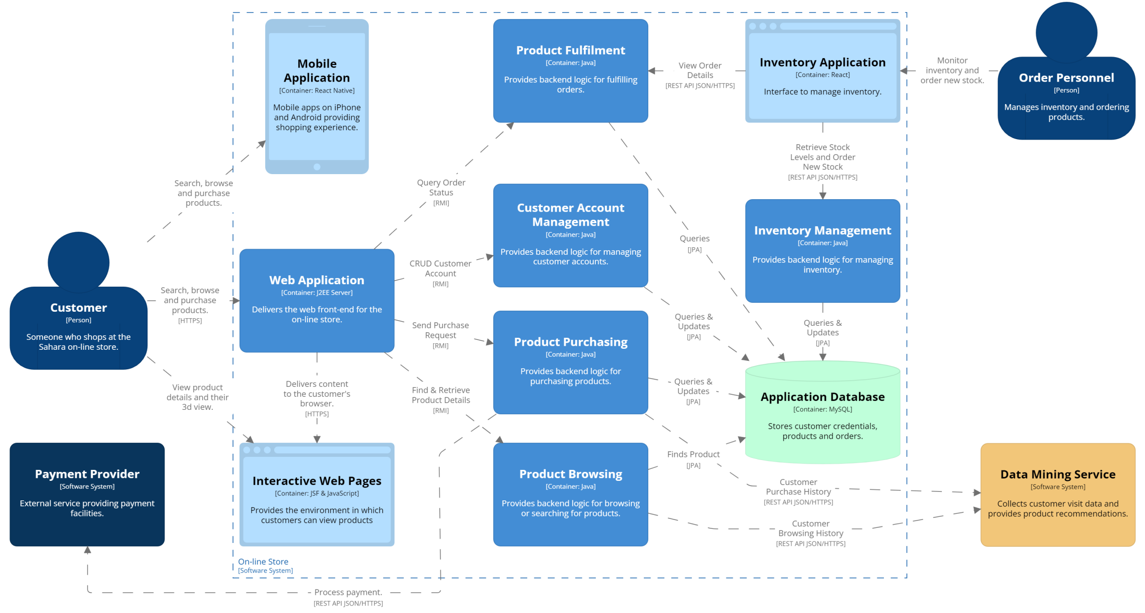
Order Personnel
[Person]
Manages inventory and ordering products.
Payment Provider
[Software System]
External service providing payment facilities.
Monitor inventory and order new stock.
Process payment.
[REST API JSON/HTTPS]
Customer
[Person]
Someone who shops at the Sahara on-line store.
Search, browse and purchase products.
On-line Store
[Software System]
Allows customers to browse and search for items and to order them.
Get Product Suggestions
Data Mining Service
[Software System]
Collects customer visit data and provides product recommendations.
Customer Browsing History
[REST API JSON/HTTPS]
Sahara eCommerce
[Enterprise]

- Summarise Sahara eCommerce example.
- Order Personnel & Payment Provider added to this example.

Oi!

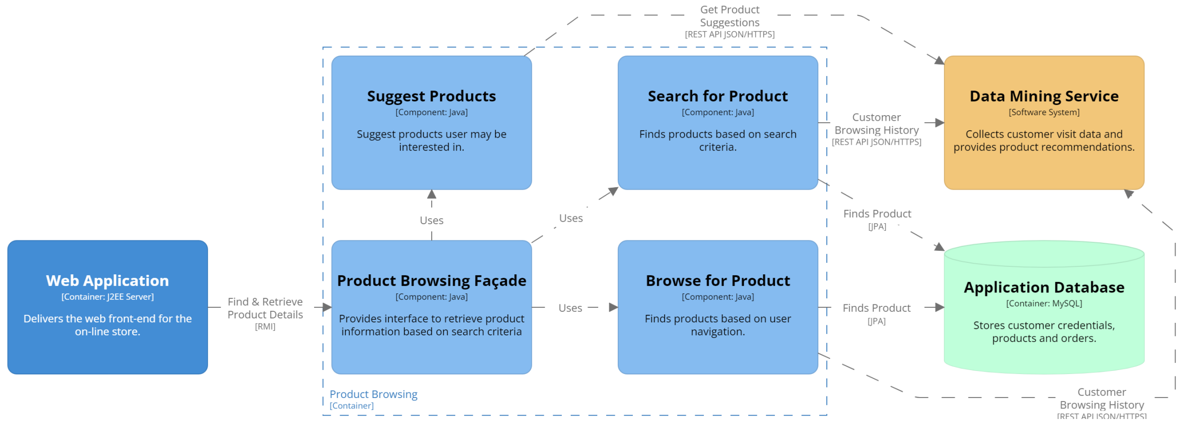
Slide to introduce domain services for Sahara example.

Sahara: On-line Store Container Diagram


Mobile Application
[Container: React Native]
Mobile apps on iPhone and Android providing shopping experience.
Product Fulfilment
[Container: Java]
Provides backend logic for fulfilling orders.
View Order Details
[REST API JSON/HTTPS]
Inventory Application
[Container: React]
Interface to manage inventory.
Monitor inventory and order new stock.
Order Personnel
[Person]
Manages inventory and ordering products.
Search, browse and purchase products.
Query Order Status
[RMI]
Retrieve Stock Levels and Order New Stock
[REST API JSON/HTTPS]
Customer Account Management
[Container: Java]
Provides backend logic for managing customer accounts.
Queries
[JPA]
Inventory Management
[Container: Java]
Provides backend logic for managing inventory.
CRUD Customer Account
[RMI]
Customer
[Person]
Someone who shops at the Sahara on-line store.
Search, browse and purchase products.
[HTTPS]
Web Application
[Container: J2EE Server]
Delivers the web front-end for the on-line store.
Queries & Updates
[JPA]
Queries & Updates
[JPA]
Send Purchase Request
[RMI]
Product Purchasing
[Container: Java]
Provides backend logic for purchasing products.
Queries & Updates
[JPA]
Application Database
[Container: MySQL]
Stores customer credentials, products and orders.
View product details and their 3d view.
Delivers content to the customer's browser.
[HTTPS]
Find & Retrieve Product Details
[RMI]
Finds Product
[JPA]
Payment Provider
[Software System]
External service providing payment facilities.
Interactive Web Pages
[Container: JSF & JavaScript]
Provides the environment in which customers can view products
Product Browsing
[Container: Java]
Provides backend logic for browsing or searching for products.
Customer Purchase History
[REST API JSON/HTTPS]
Data Mining Service
[Software System]
Collects customer visit data and provides product recommendations.
Customer Browsing History
[REST API JSON/HTTPS]
On-line Store
[Software System]
Process payment.
[REST API JSON/HTTPS]

- Review Domain Services in context of the overall system.
- Mobile App relationships not shown to reduce clutter.
- Repeat idea that Service APIs mean you may have multiple UIs (Web, Mobile, Inventory Apps).

Sahara: Product Browsing Component Diagram

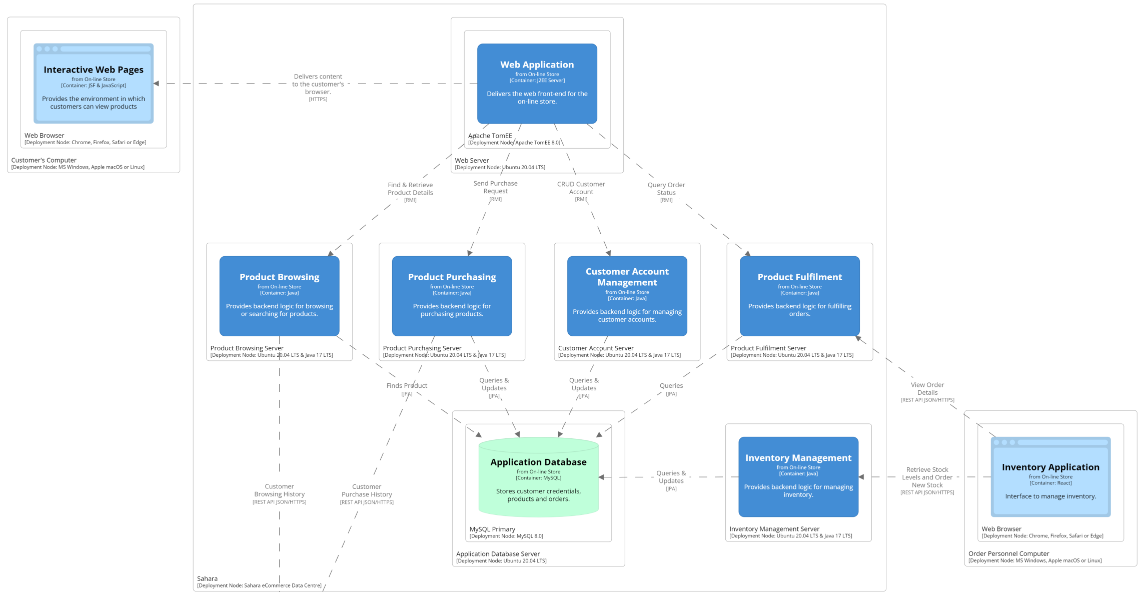

Get Product Suggestions
[REST API JSON/HTTPS]
Suggest Products
[Component: Java]
Suggest products user may be interested in.
Search for Product
[Component: Java]
Finds products based on search criteria.
Customer Browsing History
[REST API JSON/HTTPS]
Data Mining Service
[Software System]
Collects customer visit data and provides product recommendations.
Uses
Uses
Finds Product
[JPA]
Web Application
[Container: J2EE Server]
Delivers the web front-end for the on-line store.
Find & Retrieve Product Details
[RMI]
Product Browsing Façade
[Component: Java]
Provides interface to retrieve product information based on search criteria
Uses
Browse for Product
[Component: Java]
Finds products based on user navigation.
Finds Product
[JPA]
Application Database
[Container: MySQL]
Stores customer credentials, products and orders.
Product Browsing
[Container]
Customer Browsing History
[REST API JSON/HTTPS]

- Summarise the components making up the key parts of the Product Browsing Service (container).
- Product Browsing Façade provides the Service API.

Oi!

REST API for Product Browsing?

Sahara: Deployment Diagram


Interactive Web Pages
from On-line Store
[Container: JSF & JavaScript]
Provides the environment in which customers can view products
Web Browser
[Deployment Node: Chrome, Firefox, Safari or Edge]
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: J2EE 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]
View Order Details
[REST API JSON/HTTPS]
Finds Product
[JPA]
Queries & Updates
[JPA]
Queries & Updates
[JPA]
Queries
[JPA]
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]

- UI Platform could be desktop, web or mobile app.
- This allows multiple concurrent users, even through one user interface.

Oi!

Issue of failures in distributed systems?
Show service architecture for Sahara and summarise parallel flows through services.
Aside about stateless services and possible multi-threaded web pages (e.g. JSF).
Comment about Services being domain partitioning, but Service internals may be technical or domain paritioned.

Pros & Cons

Simplicity	Core system & Plug-in interface	👍
Extensibility	Plug-ins	👍
Interoperability	Plug-ins	👍
Scalability		👎
Reliability		👎