Microservices Architecture

Software Architecture

Richard Thomas
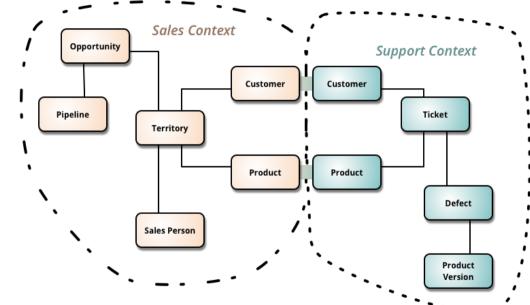
May 8, 2023

Microservices

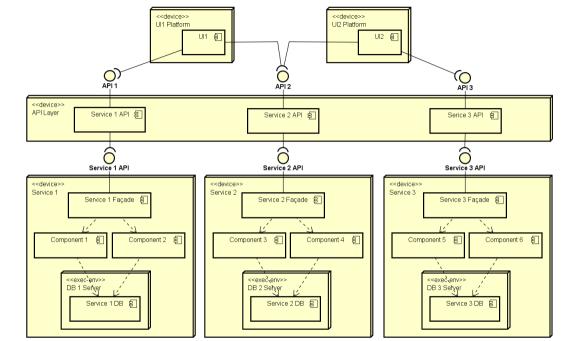# Inspired by DDD

Definition 1. Bounded Context

Logical boundary of a domain where particular terms and rules apply consistently.

Sales Context

Support Context

Opportunity

Pipeline

Territory

Customer

Product

Sales Person

Customer

Ticket

Product

Defect

Product Version

Definition 2. Service Cohesion Principle

Services are cohesive business processes. They are a bounded context.

Definition 3. Service Independence Principle

Services should not depend on the implementation of other services.

Corollary 1. Low Coupling

Services should have minimal coupling with other services.

Corollary 2. No Reuse

Services do not reuse components from other services, to avoid dependencies.

Choreography & Orchestration

Choreography Similar to event-driven *broker*

Orchestration Similar to event-driven *mediator*

"Place an order to purchase a book"

API Layer

Workflow owner

Order Placement
Module
Module
Database

pay_for_order

Payment
Module
Module
Database

update_inventory

Inventory
Module
Module
Database

"Place an order to purchase a book"

API Layer

Workflow owner

Place Order

Module

Module

Order Placement

Module

Module

Database

Inventory

Module

Module

Database

Payment

Module

Module

Database

How bad is the coupling with choreography or orchestration?

How bad is the coupling with choreography or orchestration?

For a very large system, very bad.

Inventory Application
from On-line Store
[Container: React]
Interface to manage inventory.
Web Browser
[Deployment Node: Chrome, Firefox, Safari or Edge]
Order Personal Computer
[Deployment Node: MS Windows, Apple macOS or Linux]

Mobile Application
from On-line Store
[Container: React Native]
Mobile apps on iPhone and Android providing shopping experience.
Customer's Mobile Device
[Deployment Node: Apple iOS or Android]

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

Payment Gateway
from Payment Provider
[Container]
Provides access to service providing payment facilities.
Payment Provider Gateway
[Deployment Node]

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]

Product Details Database
from On-line Store
[Container: MySQL]
Stores products details.
Inventory Database
[Deployment Node: MySQL 8.0]

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]

Fulfilment Database
from On-line Store
[Container: MySQL]
Stores customer credentials, products and orders.
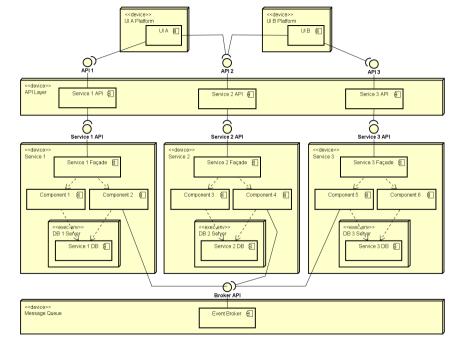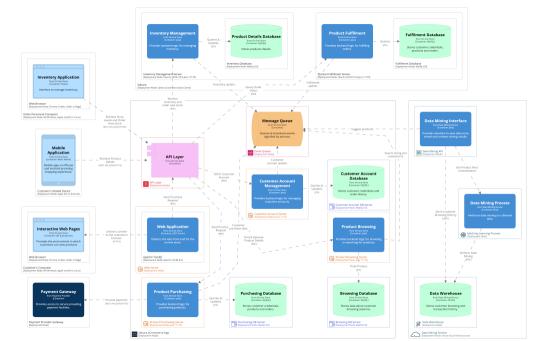Fulfilment Database
[Deployment Node: MySQL 8.0]

API Layer
from On-line Store
[Container]
API Layer
[Deployment Node]

Message Queue
from On-line Store
[Container: Java]
Queues & broadcast events signalled by services.
Event Queue
[Deployment Node]

Customer Account Management
from On-line Store
[Container: Java]
Provides backend logic for managing customer accounts.
Customer Account Server
[Deployment Node: Java 17 LTS]

Customer Account Database
from On-line Store
[Container: MySQL]
Stores customer credentials and order history.
Customer Account DB Server
[Deployment Node: MySQL 8.0]

Data Mining Interface
from Data Mining Service
[Container: Java]
Provides interface to save data to be mined and retrieve mining results.
Data Mining Node
[Deployment Node]

Data Mining Process
from Data Mining Service
[Container: Java]
Performs data mining on collected data.
Machine Learning Process
[Deployment Node]

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]

Product Browsing
from On-line Store
[Container: Java]
Provides backend logic for browsing or searching for products.
Product Browsing Server
[Deployment Node: Java 17 LTS]

Product Purchasing
from On-line Store
[Container: Java]
Provides backend logic for purchasing products.
Product Purchasing Server
[Deployment Node: Java 17 LTS]

Purchasing Database
from On-line Store
[Container: MySQL]
Stores customer credentials, products and orders.
Purchasing DB Server
[Deployment Node: MySQL 8.0]

Browsing Database
from On-line Store
[Container: MySQL]
Stores data about customer browsing sessions.
Browsing DB Server
[Deployment Node: MySQL 8.0]

Data Warehouse
from Data Mining Service
[Container: Oracle]
Stores customer browsing and transaction history.
Data Warehouse
[Deployment Node]

Sahara
[Deployment Node: Sahara eCommerce Data Centre]

Data Mining Service
[Deployment Node: Oracle Cloud Infrastructure]

Sahara eCommerce App
[Deployment Node]

Queries & Updates [JPA]
Queries & Updates [JPA]
Queries [JPA]
Inventory update.
Query Order Status [RMI]
Fulfilment update.
Monitor inventory and order new stock.
Retrieve Stock Levels and Order New Stock [REST API JSON/HTTPS]
Retrieve Product Details [REST API JSON/HTTPS]
Delivers content to the customer's browser [HTTPS]
Process payment. [REST API JSON/HTTPS]
Suggest products.
Customer account update.
CRUD Customer Account [RMI]
Queries & Updates [JPA]
Search string and customer id.
Get Product Recommendations
Store Customer Browsing History [JDBC]
Perform Data Mining [JDBC]
Send Purchase Request [RMI]
Send Purchase Request [RMI]
Customer purchase data.
Find & Retrieve Product Details [RMI]
Finds Product [JPA]
Queries & Updates [JPA]
Queries & Updates [JPA]
Data Mining API

Are *browsing* and *purchasing* separate contexts?

Are *browsing* and *purchasing* separate contexts?

- Are the a single business process or different processes?
- Do they share much or little data?

- What about *inventory management* and *browse*?
- How do they maintain a consistent product database?

Pros & Cons

Modularity 👍

Extensibility 👍

Reliability 👍

Interoperability 👍

Scalability 👍

Security 😐

Deployability 😐

Testability 😐

Simplicity 🙁