Middleware

By
Jitendra Singh Tomar || Jeetu

Content

- Introduction to Middleware
- Middleware Architectures
- Common Middleware Services
- Middleware Security
- Middleware Integration Patterns

What is middleware?

- Middleware is software that different applications use to communicate with each other.
- It provides functionality to connect applications intelligently and efficiently so that you can innovate faster.
- Middleware acts as a bridge between diverse technologies, tools, and databases so that you can integrate them seamlessly into a single system.

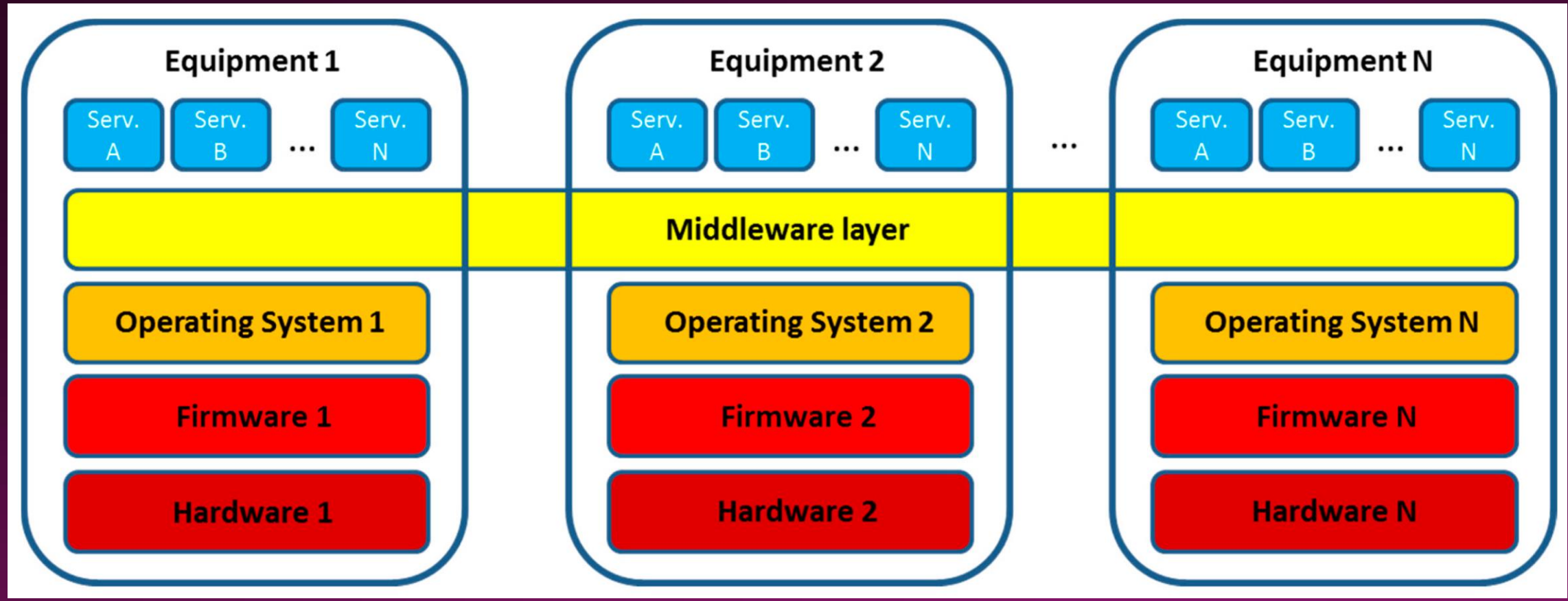
What is middleware?

- Developers use middleware to support application development and simplify design processes.
- Without middleware, developers would have to build a data exchange module for each software component that connects to the application.

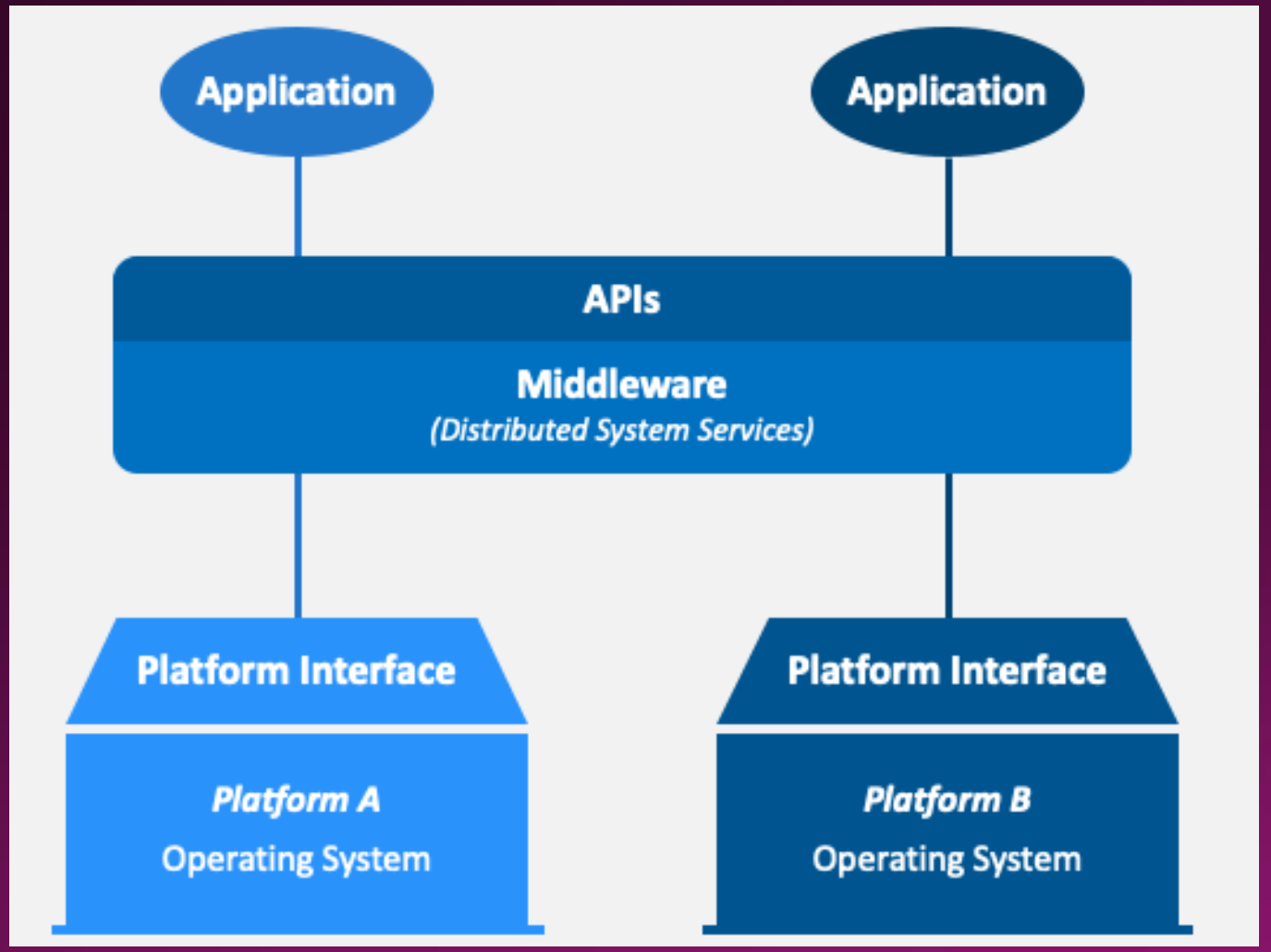
Use cases of middleware

- Game development
- Electronics
- Software development
- Data transmission
- Distributed applications

Middleware architecture



Middleware architecture



What is platform middleware?

- Platform middleware supports application development by providing a system of managed tools and resources.
- Developers use platform middleware to share or transfer resources between applications.
- Following are some examples of platform middleware resources:
  - Content management systems
  - Containers
  - Runtime environments
  - Web servers

Types of Middleware

- Message-Oriented Middleware (MOM)
- Object Request Brokers (ORB)
- Database Middleware
- Remote Procedure Call (RPC) Middleware
- Web Middleware
- Transaction Processing Middleware
- Integration Middleware

Message-Oriented Middleware (MOM)

- MOM facilitates asynchronous communication (replies can be delayed) between distributed components by enabling the exchange of messages.
- It typically provides features such as message queuing, publish-subscribe messaging, and message routing.
- Examples include Apache Kafka, RabbitMQ, and IBM MQ.

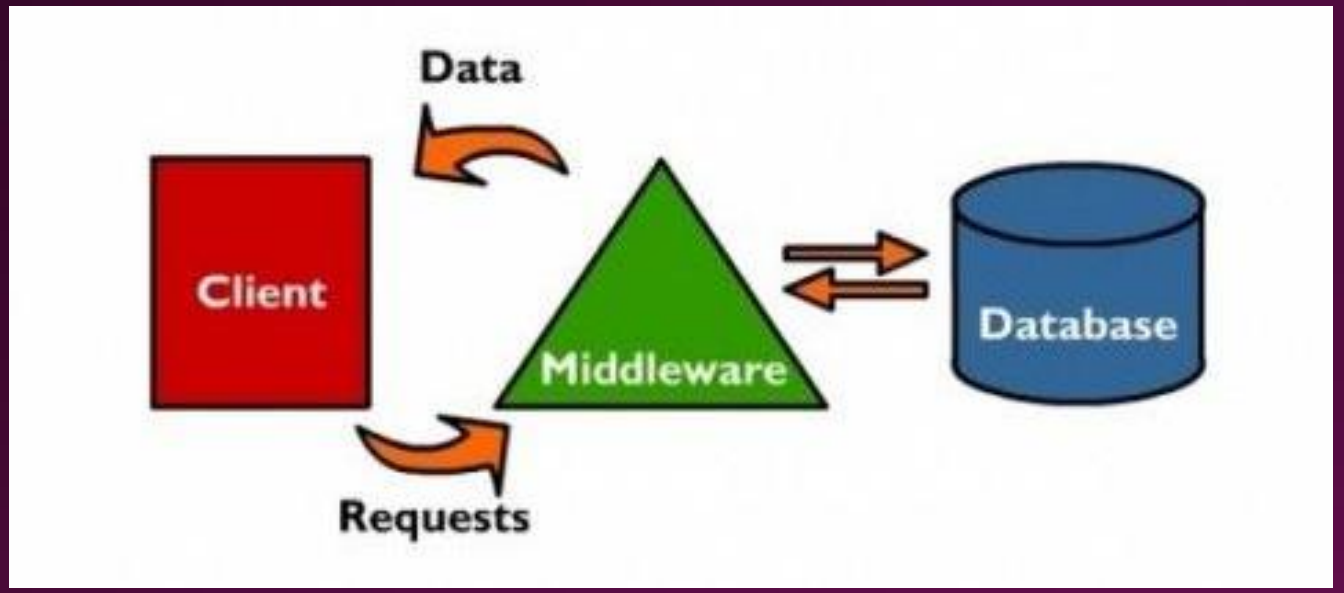
Object Request Brokers (ORB)

- ORBs facilitate communication between distributed objects or components in a distributed computing environment.
- They enable remote method invocation (RMI) by handling requests and responses between client and server objects transparently.
- Examples include CORBA (Common Object Request Broker Architecture) and Java RMI (Remote Method Invocation).

Database Middleware

- Database middleware provides an abstraction layer between applications and databases, allowing them to interact with databases without needing to understand the underlying database management system (DBMS) or SQL.
- It often includes features such as connection pooling, data caching, and object-relational mapping (ORM).

Database Middleware

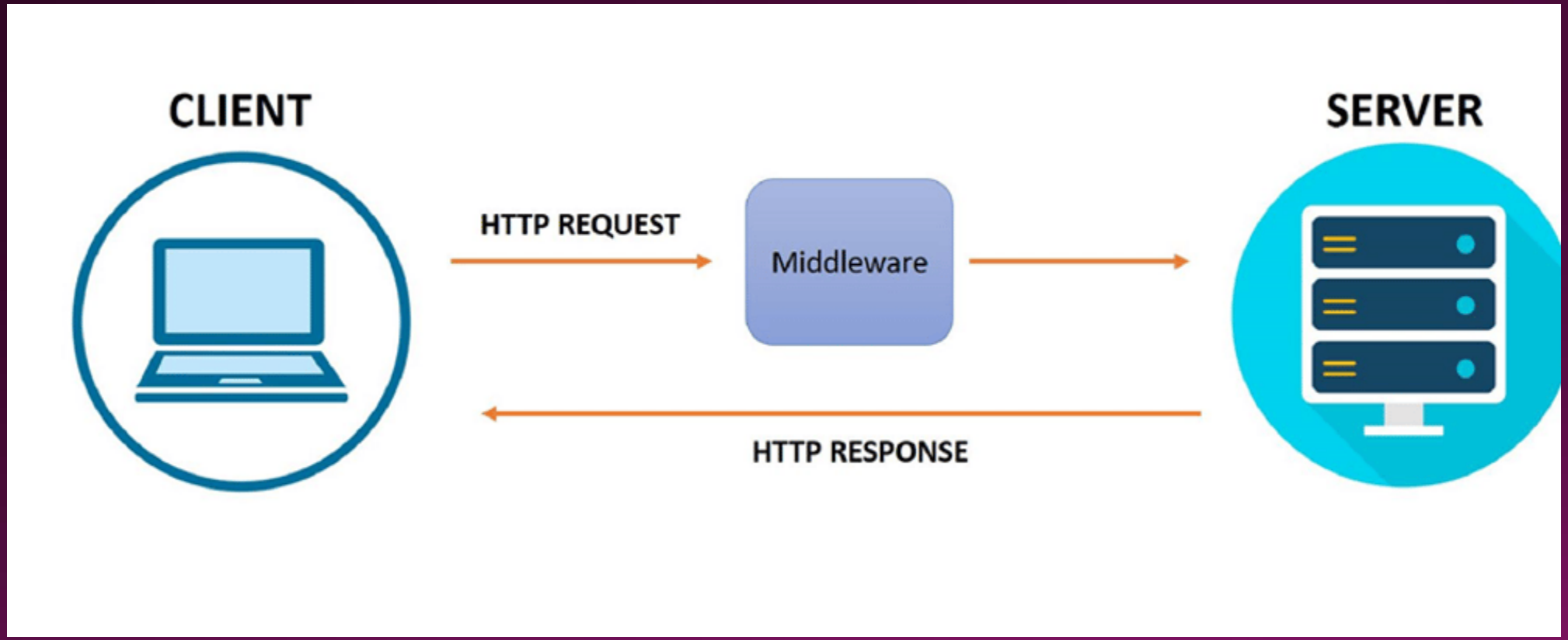


- Examples include JDBC (Java Database Connectivity) for Java applications and ODBC (Open Database Connectivity) for various platforms.

Remote Procedure Call (RPC) Middleware

- RPC middleware enables communication between distributed processes or components by allowing one process to invoke procedures or functions on another process located on a remote machine.
- It abstracts away the complexities of network communication and marshaling/un-marshaling parameters.
- Examples include gRPC, XML-RPC, and JSON-RPC.

Web Middleware



Web Middleware

- Web middleware provides services and frameworks for developing web applications, managing HTTP requests, and handling web-related concerns such as session management, authentication, and authorization.
- Examples include web servers like Apache HTTP Server and application servers like Tomcat, JBoss, and Microsoft IIS.

Transaction Processing Middleware

- Transaction processing middleware manages distributed transactions across multiple resources or services, ensuring atomicity, consistency, isolation, and durability (ACID properties).
- It coordinates transactional operations and handles concurrency control and recovery in distributed environments.
- Examples include Java Transaction API (JTA) and Microsoft Distributed Transaction Coordinator (MSDTC).

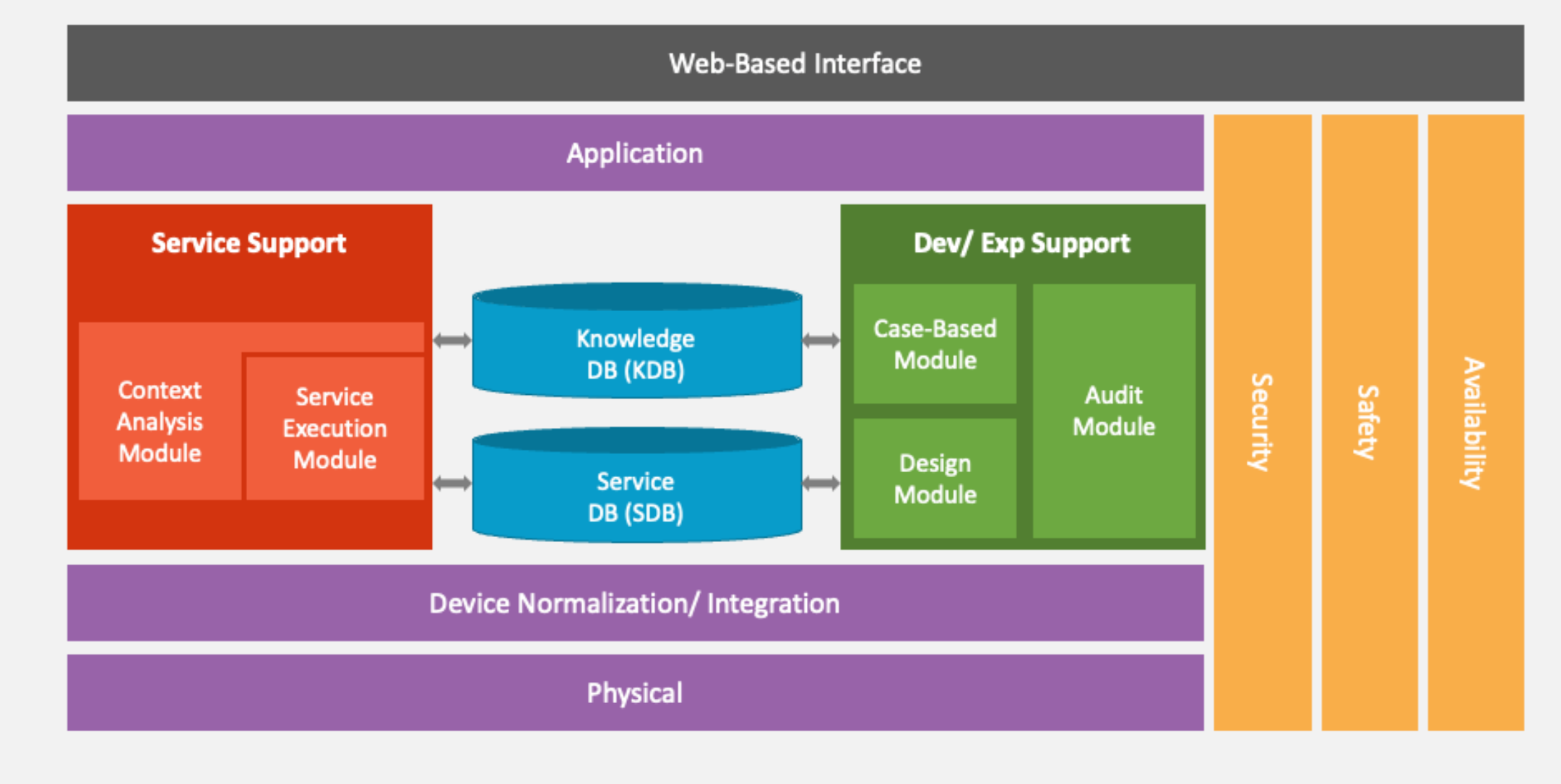
Integration Middleware

- Integration middleware facilitates the integration of disparate systems, applications, and data sources by providing tools and services for data transformation, protocol mediation, and message routing.
- It enables seamless communication and interoperability between different systems.
- Examples include Apache Camel, MuleSoft Anypoint Platform, and IBM Integration Bus.

Components of Middleware Architectures

- Communication Protocols
- Message Brokers
- Middleware Services
- Integration Adapters
- Distributed Components
- Middleware Containers
- Middleware APIs
- Management and Monitoring Tools

Components of Middleware Architectures



Security Challenges in Middleware

- Authentication and Authorization
- Data Confidentiality and Integrity
- Message Security
- Identity Management
- Secure Configuration and Deployment
- Denial-of-Service (DoS) Attacks
- Monitoring and Logging
- Secure Interoperability

Secure Communication Protocols

- TLS (Transport Layer Security)
  - The successor of SSL.
- IPsec (Internet Protocol Security)
  - It provides mechanisms for authentication, encryption and integrity check.
- SSH (Secure Shell)
  - Used for secure login, command execution & file transfer.

Secure Communication Protocols

- SFTP (SSH File Transfer Protocol)
  - It provides encrypted file transfer and remote file management capabilities.
- AMQP/TCP (Advanced Message Queuing Protocol over TCP)
  - AMQP is a messaging protocol that supports secure communication between message brokers and clients.

Secure Communication Protocols

- HTTPS (HTTP Secure)
  - It encrypts HTTP traffic using TLS, protecting sensitive information exchanged over the web.
- MQTT over TLS (MQTT Secure)
  - It ensures the confidentiality and integrity of messages exchanged in IoT (Internet of Things) and messaging applications within middleware architectures.

Middleware Integration Patterns

- Middleware integration patterns are architectural blueprints that define how different systems, components, or applications can communicate and exchange data within a distributed environment.
- Different types of patterns are:
  - Point-to-Point Integration
  - Publish-Subscribe Model
  - Message Brokers and Queues

Patterns - Point-to-Point Integration

- In this, two or more systems communicate directly with each other to exchange data or messages.
- Point-to-point integration is straightforward and suitable for simple communication scenarios but can become complex and difficult to manage as the number of connections increases.

Patterns - Publish-Subscribe Model

- The publish-subscribe model involves a communication pattern where publishers produce messages and subscribers receive messages based on their interests or subscriptions.

Patterns - Message Brokers

- Message brokers act as intermediaries between producers and consumers, facilitating the exchange of messages within a distributed system.
- They typically provide features such as message queuing, routing, transformation, and delivery assurance.

Patterns - Message Queues

- Message queues are data structures that hold messages sent from producers until they are consumed by consumers.
- Message queues typically support features such as message persistence, prioritization, and expiration, ensuring reliable message delivery even in the presence of network failures or system downtime.
- Examples of message queues include Amazon SQS (Simple Queue Service) and Azure Service Bus.