LAB 8 REPORT: DEPLOYMENT VIEW & ATAM ANALYSIS

Project Name: Movie Ticket Booking System

Member Name: Nguyễn Đình Quyền - 23010485

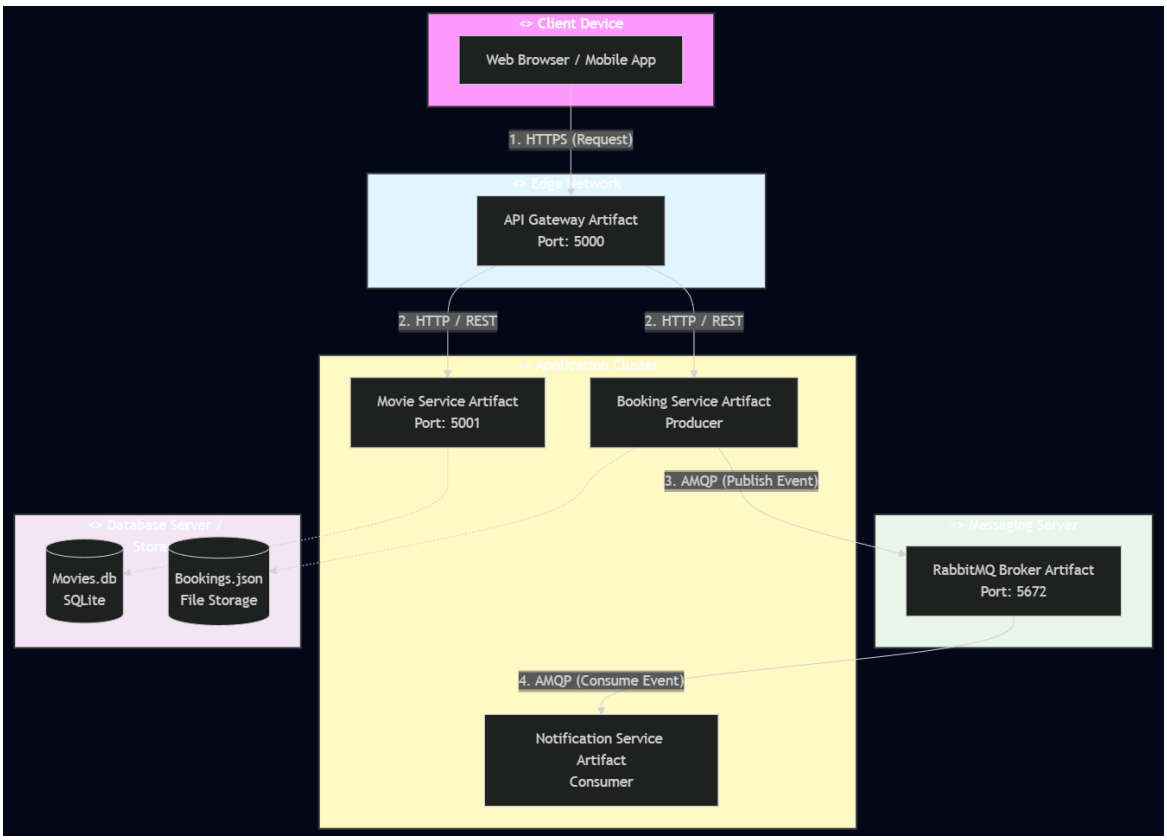
Trần Hữu Kiên - 23010258

1. OBJECTIVES

The goal of this lab is to document the physical deployment of the Microservices Architecture and evaluate its quality attributes against the Monolithic approach.

- **Deployment View:** Visualize how services (Product, Order, Notification, Gateway) are deployed on physical nodes.
- **ATAM Analysis:** Evaluate Scalability and Availability using specific scenarios.

2. UML DEPLOYMENT DIAGRAM



Components & Nodes Description:

1. **Nodes (Hình hộp 3D):**
    - **Client Device:** Represents the user's browser or mobile app.
    - **Edge Network:** Contains the Load Balancer/API Gateway.
    - **Application Cluster:** The main execution environment for Microservices.
    - **Messaging Server:** Dedicated server for RabbitMQ.
    - **Database Server:** Hosts the SQLite/Database instances.
2. **Artifacts (Các file/service bên trong Node):**
    - Inside **Edge Network**: `API Gateway Artifact` (Lab 6).
    - Inside **Application Cluster**:
        - `Movie Service Artifact` (Lab 5).
        - `Booking Service Artifact` (Lab 7 - Producer).
        - `Notification Service Artifact` (Lab 7 - Consumer).
    - Inside **Messaging Server**: `RabbitMQ Broker Artifact`.
    - Inside **Database Server**: `Movies DB`, `Bookings DB`.
3. **Communication Paths (Mũi tên kết nối):**
    - **Client Device** $\leftrightarrow$ **Edge Network** (Protocol: HTTPS).
    - **Edge Network** $\leftrightarrow$ **Application Cluster** (Protocol: HTTP/REST).
    - **Application Cluster (Booking Service)** $\rightarrow$ **Messaging Server** (Protocol: AMQP).
    - **Messaging Server** $\rightarrow$ **Application Cluster (Notification Service)** (Protocol: AMQP).

[INSERT YOUR UML DEPLOYMENT DIAGRAM HERE]

(Chèn hình ảnh sơ đồ Deployment bạn đã vẽ)

---

3. ARCHITECTURE TRADE-OFF ANALYSIS (ATAM)

This section evaluates how the architecture handles specific quality attribute scenarios.

3.1. Scenarios Definition

- **Scalability Scenario (SS1):** During a "Black Friday" movie release event, the system experiences a 10x spike in users searching for movies and viewing details.
- **Availability Scenario (AS1):** The Notification Service crashes or becomes unavailable due to a bug while users are booking tickets.

3.2. ATAM Analysis Table

Quality Attribute	Scenario	Monolithic (Layered) Approach	Microservices Approach (Current)
Scalability	**SS1** (10x Traffic Spike on Movie Search)	**Inefficient:** We must scale the entire application (including Booking, User, Payment modules) even though only the "Movie Search" feature is under heavy load. This wastes resources (CPU/RAM).	**Efficient:** We can scale *only* the **Movie Service** instances (e.g., spin up 5 more containers). The Booking and Notification services remain untouched. Resources are optimized.
Availability	**AS1** (Notification Service Fails)	**Low Fault Isolation:** Since the notification logic is tightly coupled, a failure in sending emails could cause the entire booking transaction to fail or hang (Timeouts), leading to lost revenue.	**High Fault Isolation:** Thanks to **Event-Driven Architecture (Lab 7)**, the Booking Service places the message in RabbitMQ and finishes the transaction. The email will be sent later when the Notification Service recovers. The user is not affected.

---

4. TRADE-OFF STATEMENT

Based on the analysis above, here are the identified architectural trade-offs:

Conclusion:

Moving to Microservices Architecture offers superior **Scalability** (granular scaling of Movie Service) and **Availability** (Fault isolation via RabbitMQ). However, it introduces significant **Complexity** in deployment and operations. We now need to manage multiple services (Gateway, Movie, Booking, Notification), a Message Broker, and network communications, whereas the Monolith was a single deployable unit.

This trade-off is acceptable for the **Movie Ticket Booking System** because the high traffic demand during movie releases requires the system to be highly scalable and resilient, which justifies the operational complexity.