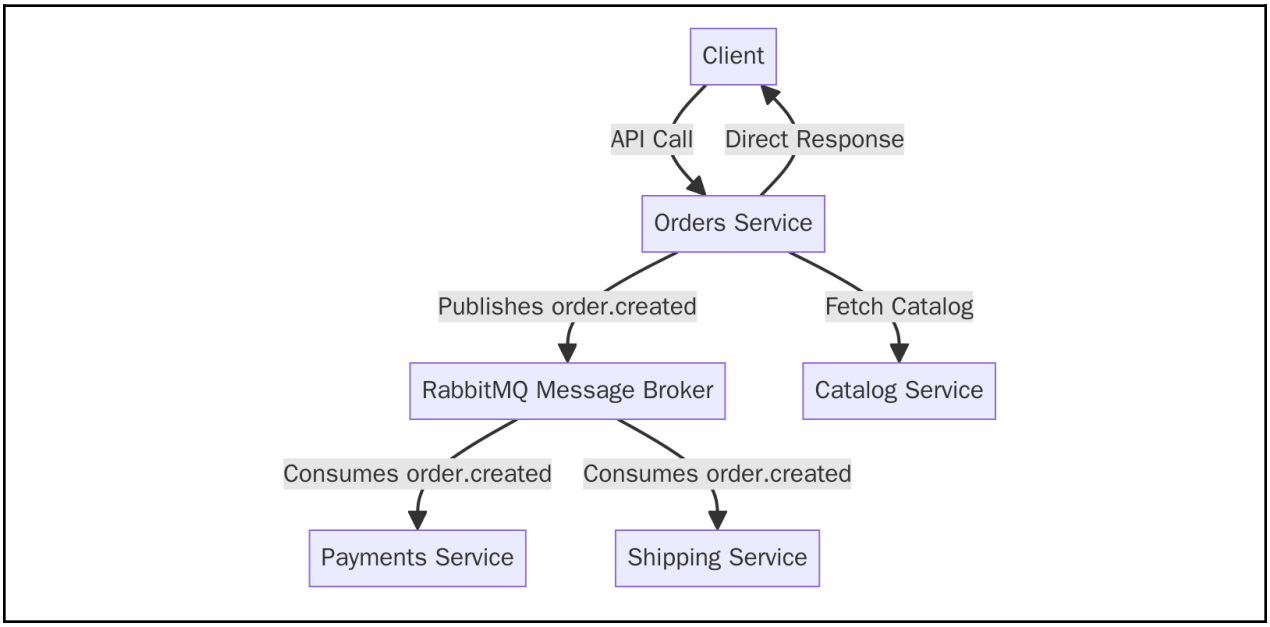
Architecture Diagram

The GlobalBooks system is designed as a distributed, Service-Oriented Architecture (SOA) that decomposes a traditional monolith into four independent microservices. The architecture utilizes a hybrid communication model to balance the need for real-time user feedback with the resilience of asynchronous background processing.



Key Components and Interactions:

Client: Represents any end-user application, such as a web front-end or mobile app. For this project, the client is simulated using API testing tools like Postman and SoapUI.

Orders Service (REST API):

- This is the primary synchronous entry point into the system.
- The client interacts with this service via direct API calls (e.g., POST /orders) to manage the order lifecycle.
- It provides immediate, synchronous responses to the client (e.g., 201 Created), confirming that a request has been successfully accepted.

RabbitMQ Message Broker:

- This is the central hub for all asynchronous communication, ensuring that services are loosely coupled.
- The Orders Service acts as a Producer, publishing event messages (like order.created and order.paid) to a topic exchange within RabbitMQ.

Payments & Shipping Services (Asynchronous Consumers):

- These are headless background services that act as Consumers.
- The Payments Service subscribes to order.created messages, initiating payment processing as soon as a new order is logged.
- The Shipping Service subscribes to order.paid messages, ensuring that the shipping process is only triggered after an order has been successfully paid for.
- This event-driven model allows these long-running tasks to execute independently without blocking the Orders Service or the client.

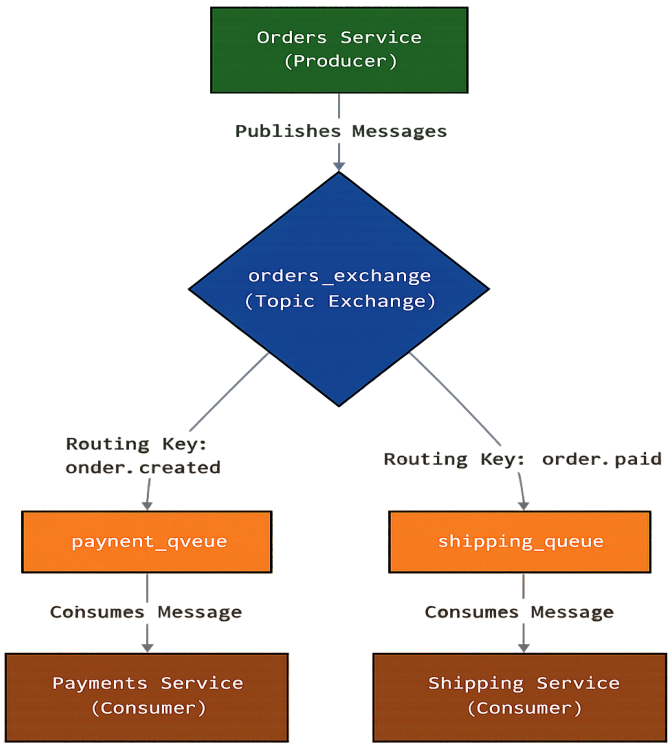
Catalog Service (SOAP API):

- This is a synchronous, standalone data service.
- It is designed to be called by other internal services or orchestration processes (like the defined BPEL workflow) to "Fetch Catalog" data, such as retrieving the official price or stock level of a book by its ISBN.

In summary, the architecture leverages a synchronous REST API for its primary command interface and an asynchronous, event-driven model for orchestrating internal business processes, resulting in a scalable, resilient, and maintainable system.

Asynchronous Messaging Architecture (RabbitMQ)

The asynchronous communication within the GlobalBooks architecture is managed by a RabbitMQ message broker. The diagram below illustrates the topology used to ensure messages are routed correctly and that business processes are decoupled and resilient.



Key Components:

Producer: The Orders Service is the sole producer of business event messages.

Topic Exchange: All messages are published to a central topic exchange named orders_exchange. This exchange acts as a smart router, directing messages based on a "routing key".

Routing Keys & Queues: The system uses two distinct routing keys to represent different business events:

1. order.created: This key is used when a new order is created. The exchange is bound to route these messages exclusively to the payment_queue.
2. order.paid: This key is used when an order's status is updated to PAID. The exchange is bound to route these messages exclusively to the shipping_queue.

Consumers: Two background services act as consumers, each listening to its dedicated queue:

1. The Payments Service consumes messages from the payment_queue.
2. The Shipping Service consumes messages from the shipping_queue.