# ****📌 Roteiro de Testes Manuais****

### ****Introdução****

Este roteiro documenta os passos necessários para testar manualmente a aplicação, garantindo que os serviços essenciais como MongoDB, Redis e RabbitMQ estejam funcionando corretamente.

## ****1️⃣ Verificar se Todos os Containers Estão Rodando****

docker ps

✅ **Verifique se os seguintes serviços estão rodando:**  
✔ order-api (porta 8080)  
✔ rabbitmq (porta 5672)  
✔ mongodb (porta 27017)  
✔ redis (porta 6379)

❌ **Se algum serviço não estiver rodando, reinicie todos os containers:**

docker-compose down && docker-compose up -d --build

## ****2️⃣ Testar Conexão com o MongoDB****

docker exec -it mongodb mongosh --eval "use orderdb; db.orders.find().pretty()"

## ****3️⃣ Testar Conexão com o Redis****

docker exec -it redis redis-cli keys "\*"

## ****4️⃣ Testar Publicação de Mensagem no RabbitMQ (Simulando Produto-Externo-A)****

docker exec -it rabbitmq rabbitmqadmin publish exchange=orders.generated.exchange routing\_key=orders.generated.key payload='{"orderId":"1234", "products": [{"name":"Produto Externo A","price":15.0,"quantity":2}] }'

## ****5️⃣ Testar o Consumo da Mensagem pela**** order-api

docker logs order-api --tail 50

## ****6️⃣ Testar o Consumo da Mensagem na Fila**** orders.processed.queue

docker exec -it rabbitmq rabbitmqadmin get queue=orders.processed.queue

## ****7️⃣ Testar a API Diretamente****

curl -X POST http://localhost:8080/orders \

-H "Content-Type: application/json" \

-d '{"orderId":"999", "products": [{"name":"Produto Teste","price":50.0,"quantity":2}] }'

----

## ****1️⃣ Verificar se Todos os Containers Estão Rodando****

docker ps

✅ Verifique se os seguintes serviços estão rodando: ✔ order-api (porta 8080) ✔ rabbitmq (porta 5672) ✔ mongodb (porta 27017) ✔ redis (porta 6379) ✔ prometheus (porta 9090) ✔ grafana (porta 3000) ✔ datadog-agent (porta 8126)

❌ Se algum serviço não estiver rodando, reinicie todos os containers:

docker-compose down && docker-compose up -d --build

## ****2️⃣ Testar Conexão com o MongoDB****

docker exec -it mongodb mongosh --eval "use orderdb; db.orders.find().pretty()"

## ****3️⃣ Testar Conexão com o Redis****

docker exec -it redis redis-cli keys "\*"

## ****4️⃣ Testar Publicação de Mensagem no RabbitMQ (Simulando Produto-Externo-A)****

docker exec -it rabbitmq rabbitmqadmin publish exchange=orders.generated.exchange routing\_key=orders.generated.key payload='{"orderId":"1234", "products": [{"name":"Produto Externo A","price":15.0,"quantity":2}] }'

## ****5️⃣ Testar o Consumo da Mensagem pela Order API****

docker logs order-api --tail 50

## ****6️⃣ Validar a Publicação do Pedido Processado (Fila orders.processed.queue)****

docker exec -it rabbitmq rabbitmqadmin get queue=orders.processed.queue

## ****7️⃣ Simular o Consumo da Mensagem pelo Produto-Externo-B****

docker exec -it rabbitmq rabbitmqadmin get queue=orders.processed.queue

## ****8️⃣ Testar a API Diretamente****

**Criar um Pedido**

curl -X POST http://localhost:8080/orders \

-H "Content-Type: application/json" \

-d '{"orderId":"999", "products": [{"name":"Produto Teste","price":50.0,"quantity":2}] }'

**Consultar Pedido Criado**

curl -X GET http://localhost:8080/orders/999

## ****9️⃣ Validar Monitoramento com Spring Boot Actuator****

**Verificar Health Check**

curl http://localhost:8080/actuator/health

**Listar Métricas Disponíveis**

curl http://localhost:8080/actuator/metrics

**Acessar Métricas Prometheus**

curl http://localhost:8080/actuator/prometheus

## ****🔟 Validar Monitoramento com Prometheus e Grafana****

**Verificar Status do Prometheus**

curl http://localhost:9090/-/ready

**Acessar Interface Web do Prometheus**

* URL: http://localhost:9090

**Acessar Interface Web do Grafana**

* URL: http://localhost:3000
* Usuário: admin
* Senha: admin

## ****1️⃣1️⃣ Validar Integração com Datadog****

**Verificar Logs e Métricas no Datadog**

1. Acesse o Datadog na URL: https://app.datadoghq.com
2. Verifique as métricas do Spring Boot na aba de Monitoramento
3. Acesse os logs e valide a ingestão correta de dados

## ****Conclusão****

Este roteiro garante que a API esteja operacional, os serviços estejam corretamente integrados e o monitoramento esteja configurado. Caso encontre falhas, consulte os logs e verifique as conexões dos containers.