1. Product & Order Service (port 3001)
2. Customer Service (port 3002)

Microservice 1: Product & Order Service

Folder structure

* apps/product-order/src/products — Product CRUD
* apps/product-order/src/orders — Order creation and event publishing

Endpoints (Port: 3001)

Products CRUD

|  |  |  |
| --- | --- | --- |
| Method | Endpoint | Description |
| GET | /products | List all products |
| GET | /products/:id | Get product by ID |
| POST | /products | Create a new product |
| PUT | /products/:id | Update a product |
| DELETE | /products/:id | Delete a product |

|  |  |  |
| --- | --- | --- |
| Method | Endpoint | Description |
| GET | /orders | List all orders |
| GET | /orders/:id | Get order by ID |
| POST | /orders | Create new order |
| PUT | /orders/:id | Update an order |
| DELETE | /orders/:id | Delete an order |

Sample Create Request:

POST /orders

{

"customerId": 1,

"items": [

{ "productId": 2, "quantity": 3 }

]

}

Sample Create Request:

POST /products

{

"name": "Phone",

"price": 799,

"stock": 10

}

Orders CRUD

This emits a RabbitMQ event order.created to Customer Service.

Microservice 2: Customer Service

Core Folders

* apps/customer/src/customers — Customer CRUD and event handling
* apps/customer/src/orders — Handles incoming order events

Endpoints (Port: 3002)

Customers CRUD

|  |  |  |
| --- | --- | --- |
| Method | Endpoint | Description |
| GET | /customers | List all customers |
| GET | /customers/:id | Get customer by ID |
| POST | /customers | Create a new customer |
| PUT | /customers/:id | Update a customer |
| DELETE | /customers/:id | Delete a customer |

Sample Create Request:

POST /customers

{

"name": "Alice",

"email": "alice@example.com"

}

How to Run the Backend

Code Walkthrough

Product Service

* Controller: products.controller.ts
* Service: products.service.ts
* Uses TypeORM for DB access

Order Service

* Creates order, then emits order.created to RabbitMQ
* @Client({ transport: Transport.RMQ }) connects to RabbitMQ

Customer Service

* Listens for order.created
* If customer exists, appends order; else creates new customer first

1. Start Docker services:

docker-compose up -d

1. Install dependencies:

npm install

1. Start services:

npm run start:dev:product-order

npm run start:dev:customer