E-Commerce Microservices Platform: Comprehensive Technical & Functional Documentation

Knowledge Base for AI Assistant

January 3, 2026

Contents

1 Introduction

This document serves as the primary knowledge base for the RAG-based AI assistant. It details the architecture, service logic, data models, and user workflows of the E-Commerce Microservices Platform. The platform is designed for high scalability, fault tolerance, and secure transaction management.

2 System Architecture & Infrastructure

The platform follows a classic microservices pattern using the Spring Cloud ecosystem.

2.1 Infrastructure Services

- **Discovery Service (Netflix Eureka)**:
  - **Port**: 8761.
  - **Role**: Service registry. Every microservice instances register their network location on startup.
  - **Client Configuration**: Services use `spring-cloud-starter-netflix-eureka-client`.
- **Config Service**:
  - **Port**: 9999.
  - **Role**: Centralized configuration. It pulls configuration files from the `config-repo` directory.
  - **Import Mechanism**: Services use `spring.config.import=configserver:http://localhost:9999`.
- **API Gateway (Spring Cloud Gateway)**:
  - **Port**: 8888.
  - **Role**: Single entry point. It handles load balancing (`lb://`) and cross-cutting concerns like security verification.

2.2 Message Broker (Kafka)

The system uses Kafka for asynchronous communication and real-time analytics.

- **Broker**: Runs on port 9092.
- **Topics**: Specifically handles topics for `BillEvents`.

3 Security Implementation

3.1 Identity and Access Management (Keycloak)

Keycloak manages authentication and authorization.

- **Realm**: `ecom-realm`.
- **Resource Server**: The Gateway validates JWT tokens issued by Keycloak.

3.2 User Roles

- **ADMIN**: Access to all management endpoints, including customer lists and inventory editing.
- **CUSTOMER**: Restricted to browsing products, managing their cart, and viewing their own bills.

4 Microservice Deep-Dive

4.1 Customer Service (Port 8081)

- **Entity**: `Customer` (id, name, email).
- **Logic**: Uses Spring Data REST to automatically expose repositories.
- **Projections**: Defines `CustomerProjection` and `CustomerProjectionEmail` for tailored API responses.
- **Endpoints**:
  - `GET /api/customers`: Retrieves all customers.
  - `GET /api/customers/{id}`: Retrieves a specific customer.

4.2 Inventory Service (Port 8082)

- **Entity**: `Product` (id [String], name, price, quantity).
- **Logic**: Manages stock. In real-world scenarios, it would decrease quantity upon order fulfillment.
- **Endpoints**:
  - `GET /api/products`: Standard product catalog.
  - `PUT/PATCH /api/products/{id}`: Used by admins to update stock or details.

4.3 Billing Service (Port 8083)

The core transactional service.

- **Logic**: Orchestrates order creation.
- **Order Creation Workflow**:
  1. Receives a `POST /api/bills/full` with `OrderRequest`.
  2. Validates the request (non-empty items).
  3. Saves a new `Bill` entity.
  4. Iterates through items to create `ProductItem` records linked to the bill.
  5. **Enrichment**: Uses OpenFeign clients to fetch customer and product details from their respective services to return a "full" object to the user.
- **Feing Clients**: `CustomerRestClient` and `ProductRestClient`.

4.4 Analytics Service

Real-time stream processing service.

- **Technology**: Spring Cloud Stream with Kafka Streams binder.
- **Logic**: Implements `billCounter` function.
- **Process**:
  - Consumes `BillEvent` records.
  - Groups them by `customerId`.
  - Maintains a persistent state store named `bill-counts` for live tracking.

5 Frontend Application (Angular)

Detailed breakdown of the user interface functionality.

5.1 Navigation Component (`app.html`)

The application layout uses a dynamic sidebar and a primary content area.

- **Sidebar Features**:
  - **Visibility**: Only shown to `isLoggedIn` users.
  - **Collapsible**: Users can toggle the sidebar width.
  - **Identity**: Displays the current user's profile and identifies their role (ADMIN/CUSTOMER).

5.2 Route Protection and Navigation

- **Products (`/products`)**: Publicly browsable catalog.
- **Customers (`/customers`)**: **Admin only** route.
- **Bills (`/bills`)**: Shows order history.
- **Cart (`/cart`)**: **Customer only** route for managing selections.

6 Practical User Guide: How-To Steps

6.1 Starting a Session

1. Click "Login" in the top bar or landing hero. 2. You will be redirected to Keycloak for secure authentication. 3. Upon return, the sidebar will appear, providing access to authorized services.

6.2 Managing Products (For Admins)

- Navigate to "Products". - Use the provided forms to update stock quantities or modify product pricing.

6.3 Ordering Workflow (For Customers)

1. **Selection**: Browse products and add desired items to the cart. 2. **Cart Review**: Go to the "Cart" page. Verify quantities and the total amount. 3. **Checkout**: Proceed with the order. This triggers the Billing Service transaction. 4. **Confirmation**: Navigate to "Bills" to see your newly generated invoice.

6.4 Analyzing Data (Internal)

- Access the Analytics service endpoints to view aggregated bill counts per customer. - Use the Eureka dashboard (`:8761`) to ensure all services are healthy.

7 Troubleshooting and Configuration

- **Service Down**: Check the Eureka dashboard. If a service is missing, ensure it's started and pointing to the correct Config Server.
- **401/403 Errors**: Verify the Keycloak token. Session expiry might require a re-login.
- **Empty Bills**: Check if the Inventory or Customer service is reachable from the Billing service via Feign.