ORDERING GATEWAY SERVICE

Full Technical & Architectural Specification

Project Name: Autonomous Grocery Network
Module: API Gateway / Orchestrator
Language: Golang (Go) 1.21+
Documentation Type: System Design Document (SDD)

Confidential Engineering Core

This specification covers the comprehensive logic for JWT authentication, database persistence, ZeroMQ analytics broadcasting, and gRPC orchestration between inventory and pricing sub-systems.

Document Version: 3.0.4
February 13, 2026

Contents

1. Operational Overview

The Ordering Microservice serves as the primary gateway for the Auto Grocery system. It is responsible for client authentication, order lifecycle management, and providing a unified REST API for both smart devices and supply-chain trucks.

2. Database Schema Details

The service maintains strict relational integrity within a PostgreSQL environment.

2.1 Smart Device Registry

Table 1: Table Definition: smart_devices

Field	Type	Description
`id`	SERIAL (PK)	Internal unique sequence.
`device_id`	VARCHAR(100)	Hardware-specific UUID for robotic units.
`password`	TEXT	Bcrypt-hashed credentials.
`email`	VARCHAR(255)	Digital receipt delivery address.
`phone`	VARCHAR(20)	SMS alert contact.

2.2 Order Lifecycle Management

Orders transition through the following states stored in the `grocery_orders` table:

- **PREVIEWED:** User has requested a quote; stock is soft-locked.
- **PROCESSING:** Robots have been dispatched via ZMQ.
- `COMPLETED`: Webhook received from Inventory confirming pick-up.

3. Authentication & Security Middleware

The service utilizes a stateless JWT-based authentication mechanism.

3.1 Token Verification Workflow

Every incoming request to the `/api/client` prefix passes through the `AuthMiddleware`. 1. The middleware extracts the ‘Bearer‘ token. 2. It validates the signature against the `JWT_SECRET`. 3. It injects the `device_id` and `user_id` into the request context.

4. Endpoint Specification (Comprehensive)

4.1 Client Login Handler (`login.go`)

Method: `POST /api/client/login`
Logic:

1. Retrieves the record from the `smart_devices` table.
2. Performs a `bcrypt.CompareHashAndPassword` check.
3. Generates a standard JWT Access Token.

4.2 Robot Confirmation (`confirm.go`)

This is the center of the orchestration. It performs a gRPC call to the Inventory Service's `AssignRobots` method.

```
// Logic inside ConfirmOrderHandler
resp, err := inventoryClient.AssignRobots(ctx, &pb.RobotRequest{
    OrderId: orderId,
    Items: items,
})
```

5. Analytics Broadcaster (ZeroMQ)

To prevent the main database from becoming a bottleneck, order metrics are broadcasted to an external analytics engine using ZeroMQ.

5.1 Flatbuffers Serialization

Metrics are packed into the `OrderMetric` Flatbuffers format. This ensures that the Ordering Service can publish thousands of metrics per second with near-zero CPU overhead.

6. Internal Webhooks

The `/internal/webhook/update-order` endpoint is restricted to internal traffic from the Inventory Service. It triggers the final update to the order status when robots complete their task.

7. Conclusion & Future Roadmap

The Ordering Microservice architecture provides the scalability required for a high-volume autonomous mart. Future updates will include mTLS between microservices and more granular permission scopes for different robot classes.