Product Catalog System

Project Overview

A microservices-based banking product catalog system that manages master product templates and tenant-specific product instances (solutions). The system enables multi-tenant banks to browse, configure, and deploy banking products from a centralized catalog.

Architecture

Two-Tier Product Model

- **Product Catalog**: Master templates for banking products (e.g., "Premium Checking", "High-Yield Savings")
- **Solutions**: Tenant-specific instances configured from catalog templates

Microservices

Located in `backend/`:

1. **product-service** (port 8082): Manages product catalog and tenant solutions
2. **customer-service** (port 8083): Customer management
3. **account-service** (port 8084): Account operations
4. **transaction-service** (port 8085): Transaction processing
5. **notification-service** (port 8086): Notifications
6. **reporting-service** (port 8087): Analytics and reporting
7. **compliance-service** (port 8088): Compliance and auditing
8. **api-gateway** (port 8080): Entry point and routing

Technology Stack

- **Backend**: Spring Boot 3.2.0, Java 17
- **Database**: MongoDB (port 27018 in Docker)
- **Messaging**: Apache Kafka
- **Build**: Maven (multi-module project)
- **Deployment**: Docker Compose

Package Structure

```
com.bank.product
├── domain/
│   ├── catalog/         # Master product catalog
│   │   ├── model/
│   │   ├── repository/
│   │   ├── service/
│   │   └── controller/
```

```
│   └── solution/        # Tenant product instances
│       ├── model/
│       ├── repository/
│       ├── service/
│       └── controller/
├── config/              # Security, etc.
├── repository/          # Shared repositories
└── service/             # Shared services
```

Key Domain Models

Product Catalog (Master Template)

- Catalog product ID, name, description
- Product type (CHECKING, SAVINGS, LOAN, etc.)
- Pricing templates, rate tiers, fee templates
- Available features and configuration options
- Status: DRAFT, AVAILABLE, DEPRECATED, RETIRED

Solution (Tenant Instance)

- References catalog product ID
- Tenant-specific configuration
- Custom pricing, fees, terms
- Status: DRAFT, ACTIVE, SUSPENDED, RETIRED
- Approval workflow for changes

Development Commands

Build

```
# Build all services
mvn clean install

# Build specific service
cd backend/product-service
mvn clean package
```

Docker Deployment

```
# Start all services
docker-compose up -d

# View logs
docker-compose logs -f product-service
```

```
# Stop all services
docker-compose down
```

MongoDB Access

```
# Connect to MongoDB
docker exec -it mongodb mongosh -u admin -p admin123 --
authenticationDatabase admin

# Use product catalog database
use productcatalog
```

API Endpoints

Product Catalog (Master Templates)

- `GET /api/v1/catalog/available` - List available catalog products
- `GET /api/v1/catalog/{catalogProductId}` - Get catalog details
- `POST /api/v1/catalog` - Create catalog product (admin)
- `PUT /api/v1/catalog/{catalogProductId}` - Update catalog product

Solutions (Tenant Products)

- `GET /api/v1/solutions` - List tenant solutions
- `GET /api/v1/solutions/{solutionId}` - Get solution details
- `POST /api/v1/solutions/configure` - Configure new solution from catalog
- `PATCH /api/v1/solutions/{solutionId}/status` - Update solution status

Authentication

All endpoints use HTTP Basic Authentication:

- Admin: `admin:admin123` (ROLE_ADMIN, ROLE_USER)
- User: `catalog-user:catalog123` (ROLE_USER)

Important Conventions

Naming

- **Catalog**: Master product templates
- **Solution**: Tenant-specific product instances
- **Product**: The overarching domain (package name)

MongoDB Collections

- `product_catalog` - Master catalog templates
- `solutions` - Tenant product instances

- `categories` - Product categories
- `tenant_solution_config` - Tenant configurations
- `users` - Authentication

Multi-tenancy

- Header: `X-Tenant-ID` for tenant context
- Header: `X-User-ID` for user tracking

Recent Changes

Latest Refactoring (feature/refactor branch)

- Renamed package from `com.bank.productcatalog` to `com.bank.product`
- Renamed `catalog-service` to `product-service`
- Renamed `Product` model to `Solution` (tenant instances)
- Kept `ProductCatalog` name (master templates)
- Reorganized domain models into `catalog` and `solution` subdomains

Areas Requiring Attention

1. **Authentication**: Currently using basic auth; consider JWT for production
2. **Testing**: Test coverage needs improvement
3. **API Gateway**: Not yet fully integrated with services
4. **Kafka Integration**: Event publishing not yet implemented
5. **Documentation**: OpenAPI/Swagger documentation incomplete
6. **Workflows**: Temporal/Camunda style workflows to manage distributed processing

File Locations

Configuration

- `backend/product-service/src/main/resources/application.yml` - Service config
- `backend/product-service/src/main/resources/application-docker.yml` - Docker overrides
- `docker-compose.yml` - Container orchestration
- `backend/pom.xml` - Parent POM
- `init-mongo.js` - MongoDB initialization script

Domain Models

- Common models: `backend/common/src/main/java/com/bank/product/`
- Service-specific: `backend/product-service/src/main/java/com/bank/product/`

Git Workflow

- Main branch: `main`
- Feature branches: `feature/*`
- Current work: `feature/refactor` (package restructuring)