**📌 Project Overview: SIM Purchase Order System**

A comprehensive Spring Boot microservices application designed to manage SIM card purchases, customer accounts, and subscription plans with robust CRUD operations and event-driven architecture.

**✅ Core Functionalities**

* **Customer Management**: Create, verify, activate, update, and delete customer accounts.
* **Plan Management**: Support for monthly and yearly plans.
* **SIM Card Operations**: Activation, suspension, and cancellation with start/end date tracking.

**🧩 Architecture & Design**

* **Microservices**:
  + Discovery Server (Eureka)
  + Sim-Purchase Service
  + Customer Service
  + Plan Service
  + Spring Cloud Config Server for centralized configuration
* **Service Discovery**: Eureka Server & Clients
* **Configuration Management**: Spring Cloud Config Server

**🔁 Event-Driven Architecture**

* **Apache Kafka Integration**:
  + Topics: Purchase, Activate, Suspend, Cancel
  + Multiple consumers for each topic
  + Retry mechanism with **Dead Letter Topics (DLT)**

**💡 Advanced Features**

* **MapStruct**: For DTO-Entity mapping across all services
* **Custom Responses**: Tailored messages for end-users and internal operations
* **Validation**: Input validation before processing
* **Exception Handling**: Custom error responses
* **Logging**: Activity tracking across services
* **Testing**: Unit and integration tests for all endpoints

**🔐 Security & Authentication**

* **Spring Security**: Username/password authentication
* **JWT Tokens**: Secure API access
* **BCrypt**: Password encoding

**🛠️ Performance & Reliability**

* **Retryable Pattern**: For SIM ordering when downstream services are unavailable
* **Atomic Transactions**: Ensures rollback on payment failure
* **Caching**: Reduces DB load and improves response time
* **Batch Processing**: Bulk SIM card insertion using Spring Batch
* **Scheduled Jobs**: Daily sales report generation with ShedLock to prevent duplicates

**📊 Monitoring & Documentation**

* **Spring Boot Actuator**: Health checks and metrics
* **Swagger**: Interactive API documentation
* **SonarLint & SonarCloud**: Code quality checks and static analysis

**🔄 Database Migration**

* Migrated from **MongoDB** to **PostgreSQL** for enhanced relational data handling