ARSITEKTUR SISTEM ePOLIMART

E-COMMERCE KAMPUS (Platform Pemesanan Makanan & Barang di Lingkungan Kampus)

Disusun untuk: Pengembangan Sistem e-commerce kampus

1. Pendahuluan

1.1 Latar Belakang

Di lingkungan kampus, kebutuhan mahasiswa terhadap makanan, minuman, dan perlengkapan perkuliahan sering muncul secara berkala dan mendesak. Akses fisik ke kantin, koperasi, atau penjual mandiri kadang memakan waktu dan menimbulkan antrean. ePolimart dikembangkan sebagai platform e-commerce internal kampus untuk memudahkan mahasiswa dan civitas akademika melakukan pemesanan dan pembelian barang atau jasa di area kampus.

1.2 Tujuan

- Menyediakan platform digital bagi mahasiswa untuk memesan makanan dan membeli barang kebutuhan kampus secara online.
- Mempermudah pengelolaan order bagi kantin, koperasi, dan penjual mahasiswa.
- Meningkatkan efisiensi distribusi barang di lingkungan kampus serta mendorong kegiatan kewirausahaan mahasiswa.
- Mengintegrasikan sistem pembayaran digital kampus (cashless) dan manajemen logistik internal.

1.3 Ruang Lingkup

Sistem mencakup frontend web & mobile untuk pengguna, back-end microservices untuk manajemen katalog, keranjang, checkout, pembayaran, manajemen stok, dan admin dashboard untuk operasional kampus.

2. Deskripsi Umum Sistem

2.1 Gambaran Umum

ePolimart adalah platform e-commerce kampus (web + mobile) yang memungkinkan:
- Pemesanan makanan dari kantin dan layanan antar internal kampus.
- Pembelian barang kebutuhan seperti alat tulis, perlengkapan praktikum, dan produk jualan mahasiswa.
- Integrasi pembayaran melalui dompet kampus, e-wallet, maupun transfer bank.
- Pelacakan pesanan dan manajemen stock yang realtime untuk petugas kantin/koperasi.

2.2 Aktor / Pengguna Sistem

1. Mahasiswa (pembeli)
2. Penjual (kantin, koperasi, mahasiswa wirausaha)
3. Kurir / Petugas Antar Kampus (opsional)

4. Admin Kampus / Operator Sistem

3. Fitur Utama

- Registrasi & Login (sso kampus / email kampus) dengan peran (student, seller, admin).
- Katalog produk & menu kantin (kategori, opsi ukuran/variasi).
- Keranjang belanja persisten (tersinkron antar perangkat).
- Checkout dengan pilihan jadwal pengantaran / pick-up.
- Integrasi pembayaran: campus-wallet, e-wallet, transfer bank, dan pembayaran tunai saat pickup.
- Manajemen stok & reservasi stock ketika checkout dimulai.
- Dashboard seller untuk melihat order, menyiapkan pesanan, dan update status.
- Notifikasi real-time (push, email, SMS) untuk status order.
- Rating & review produk/kantin.
- Laporan penjualan untuk admin kampus dan seller.

4. Non-Functional Requirements

- Availability: target 99.5% (dengan maintenance terjadwal di luar jam sibuk).
- Performance: response katalog <200ms, latency checkout <2s (pada kondisi normal).
- Scalability: autoscaling untuk service frontend dan API.
- Security: TLS, proteksi OWASP Top 10, dan kebijakan privasi data mahasiswa.
- Maintainability: service terpisah (microservices) dan CI/CD untuk deployment.
- Localization: Bahasa Indonesia (default) dan format waktu/wilayah kampus.

5. High-Level Architecture

Pendekatan arsitektur: Microservices (direkomendasikan). Setiap domain bisnis dipisah dalam layanan mandiri sehingga mudah diskalakan dan di-deploy.

Diagram Arsitektur (ringkas):

Clients (Web SPA / Mobile App)
  ↓
CDN / Load Balancer
  ↓
API Gateway (Auth, Rate Limit, Routing)
  ↓
Microservices: Auth | Product | Cart | Order | Payment | Inventory | Notification | Review | Reporting
  ↓
Data Stores: PostgreSQL (transactions) | Redis (cache, session) | Elasticsearch (search) | S3/MinIO (images) | Kafka/RabbitMQ (event bus)

6. Komponen Sistem & Tanggung Jawab

- API Gateway: Routing, autentikasi awal, rate limiting, logging request

- Auth Service: SSO kampus, JWT issuance, manajemen user & role

- Product Service: CRUD produk, kategori, gambar, push index ke search

- Cart Service: Manage cart (Redis + persistent DB), cart merging saat login

- Order Service: Pembuatan order, perhitungan total, apply diskon/voucher, publish event order.created

- Inventory Service: Reservasi stock, commit/rollback stock, threshold alerts

- Payment Service: Integrasi payment gateway, handle webhook pembayaran

- Notification Service: Kirim email, push notification (FCM), SMS gateway

- Seller Dashboard: UI untuk seller manage produk, order, laporan

- Admin Dashboard: Manajemen user, approval seller, laporan dan monitoring

- Reporting / Analytics: Ingest event ke data warehouse untuk laporan penjualan

- Delivery Service (opsional): Manajemen kurir/petugas antar internal kampus

7. Data Model (Ringkasan Tabel & Keterangan)

Berikut tabel utama dan kolom penting. Tipe data disederhanakan; implementasi bisa disesuaikan pada DBMS (Postgres).

Tabel	Kolom Utama	Keterangan
users	id_user (PK), nama, email, password, role, no_hp, alamat	Menyimpan data user pembeli, penjual, admin
toko	id_toko (PK), id_user (FK), nama_toko, deskripsi, lokasi	Informasi toko yang dikelola penjual
produk	id_produk (PK), id_toko (FK), nama_produk, deskripsi, harga, stok	Daftar produk yang dijual
keranjang	id_keranjang (PK), id_user (FK), tanggal_dibuat	Keranjang belanja user
detail_keranjang	id_detail_keranjang (PK), id_keranjang (FK), id_produk (FK), qty, subtotal	Daftar produk dalam keranjang
pesanan	id_pesanan (PK), id_user (FK), total_harga, status_pesanan, tanggal_pesanan	Data pesanan user
detail_pesanan	id_detail_pesanan (PK), id_pesanan (FK), id_produk (FK), qty, subtotal	Rincian produk dalam pesanan
pembayaran	id_pembayaran (PK), id_pesanan (FK), metode_pembayaran, status_pembayaran, tanggal_pembayaran	Mencatat status pembayaran
riwayat_stok	id_riwayat (PK), id_produk (FK), perubahan, keterangan, tanggal	Mencatat perubahan stok produk

8. Alur Proses Utama

8.1 Sequence: Checkout (Mahasiswa Memesan Makanan)

1. Mahasiswa memilih item di katalog dan menambahkan ke keranjang.
2. Mahasiswa membuka halaman checkout dan memilih metode pengambilan (antar

kampus / pickup) dan metode pembayaran.
3. Order Service menerima request checkout: memvalidasi cart dan memanggil Inventory Service untuk melakukan reservasi stock.
4. Jika reservasi berhasil, Order Service membuat record order dengan status 'pending_payment' dan memanggil Payment Service untuk membuat payment intent.
5. Mahasiswa menyelesaikan pembayaran melalui payment gateway / campus-wallet.
6. Webhook dari payment gateway menginformasikan Payment Service tentang sukses pembayaran -> Payment Service mengupdate payment record dan publish event 'payment.succeeded'.
7. Order Service menangani event tersebut: ubah status order menjadi 'paid' dan publish 'order.paid'. Inventory Service commit reservation (kurangi stock).
8. Notification Service mengirim konfirmasi ke mahasiswa, dan seller/kantin menerima notifikasi pesanan baru.
9. Seller menyiapkan pesanan -> ubah status ke 'packed' / 'ready for pickup' -> kurir internal atau mahasiswa mengambil pesanan.
10. Setelah pesanan diterima, status diupdate menjadi 'completed' dan laporan penjualan diupdate di Reporting Service.

8.2 Handling Timeout & Rollback

- Jika pembayaran tidak selesai dalam batas waktu TTL reservasi (mis: 15 menit), Inventory Service akan otomatis rollback reservasi dan order akan dibatalkan (status: cancelled) atau diberi notifikasi untuk retry.
- Untuk concurrency pada stock, gunakan mekanisme optimistic locking (versioning) atau distributed locks untuk memastikan konsistensi stock.

9. Integrasi Pembayaran & Pengiriman

Pembayaran:
- Integrasikan dengan payment gateway lokal (mis: Midtrans/Xendit) dan campus-wallet (saldo mahasiswa).
- Gunakan tokenisasi untuk menyimpan metode pembayaran jika diperlukan, dan jangan menyimpan nomor kartu secara langsung.

Pengiriman / Pengambilan:
- Opsi pickup di kantin atau pengantaran oleh petugas/kru kampus.
- Integrasikan modul Delivery Service sederhana untuk assign kurir, estimasi waktu, dan pelacakan status antar.

10. Keamanan & Privasi

- Semua komunikasi lewat HTTPS/TLS.
- Password di-hash dengan bcrypt/argon2.
- Gunakan JWT untuk autentikasi; refresh token disimpan di DB/Redis.
- Validasi dan sanitasi semua input untuk mencegah SQL Injection, XSS.
- Role-based access control (RBAC) untuk endpoint admin/seller.

- Simpan secrets di Secret Manager / KMS; enkripsi data sensitif.
- Jika menyimpan data mahasiswa, patuhi kebijakan privasi kampus dan GDPR-like aturan jika berlaku.

11. Infrastruktur & Deployment

Rekomendasi deployment di cloud atau private cloud kampus (on-premise) menggunakan container orchestration (Kubernetes). Berikut mapping arsitektur ke komponen infra:

- CDN: CloudFlare / CloudFront untuk static assets dan caching

- Load Balancer & API Gateway: NGINX/ALB + Kong/Traefik untuk routing, TLS termination

- Container Platform: Kubernetes (EKS/GKE/AKS) atau on-prem k8s

- Database: Postgres (Primary + Read Replicas), dengan backup & PITR

- Cache: Redis (Cluster) untuk sesi & cache

- Object Storage: S3 / MinIO untuk image & invoice

- Search: Elasticsearch / OpenSearch untuk pencarian produk

- Message Broker: Kafka / RabbitMQ untuk event-driven architecture

- Monitoring: Prometheus, Grafana, ELK Stack (Elasticsearch, Logstash, Kibana), Sentry

- CI/CD: GitHub Actions / GitLab CI -> Build Docker -> Push -> Helm deploy

Pertimbangan tambahan:
- Gunakan multi-AZ for DB untuk high availability.
- Setup autoscaling untuk pods dan workers.
- Implementasikan backup rutin dan DR plan.

12. Observability & Monitoring

- Centralized logging (Fluentd/Logstash → Elasticsearch/Kibana).
- Metrics (Prometheus) dan dashboards (Grafana) untuk health checks dan performance.
- Distributed tracing (OpenTelemetry + Jaeger) untuk debugging flow checkout.
- Error tracking (Sentry) untuk exception monitoring.
- Alerts (PagerDuty/Opsgenie) untuk insiden kritikal.

13. CI/CD, Testing & QA

- Pipelines: build, unit tests, integration tests, security scans (dependency & SCA), docker image build, deployment to staging, e2e tests, then deployment to production.
- Test types: unit, integration, contract tests (pact) for microservices, end-to-end (Playwright/Cypress) for frontend.
- Use feature flags for gradual rollout and A/B testing.

14. Operational Playbook (singkat)

- Runbook untuk skenario: payment gateway down, DB failover, high error rates, slow checkout.
- On-call rotation & escalation policy.
- Routine maintenance windows scheduled outside peak campus hours.
- Regular security audits & penetration testing.

15. Roadmap & Rencana Pengembangan

Phase 1 (MVP): Auth SSO, product catalog, cart, checkout manual, seller dashboard, basic payment (campus-wallet).
Phase 2: Integrasi payment gateway, search (Elasticsearch), notification real-time, reporting.
Phase 3: Scaling, optimization, personalized recommendation, third-party integrations (library, campus services).

16. Appendix: API Endpoint Contoh & SQL DDL (ringkas)

Contoh endpoint (RESTful):

- POST /api/v1/auth/login -> body: {campus_id, password} -> returns JWT

- GET /api/v1/products -> query: ?q=&category=&page=&size=

- GET /api/v1/products/{id} -> detail produk

- POST /api/v1/cart -> add item

- GET /api/v1/cart -> get cart

- POST /api/v1/checkout -> body: {cart_id, payment_method, delivery_option}

- POST /api/v1/webhooks/payment -> payment gateway webhook

Contoh SQL DDL ringkas untuk tabel users, products, orders:

```
CREATE TABLE users (
  id SERIAL PRIMARY KEY,
  campus_id VARCHAR(50) UNIQUE NOT NULL,
  name VARCHAR(150) NOT NULL,
  email VARCHAR(150),
  password_hash VARCHAR(255),
  role VARCHAR(20),
  phone VARCHAR(30),
  created_at TIMESTAMP DEFAULT NOW()
);
```

```
CREATE TABLE products (
  id SERIAL PRIMARY KEY,
  seller_id INTEGER REFERENCES users(id),
  name VARCHAR(255) NOT NULL,
  sku VARCHAR(100),
  description TEXT,
  price NUMERIC(12,2),
  stock INTEGER DEFAULT 0,
  status VARCHAR(20),
  created_at TIMESTAMP DEFAULT NOW()
);

CREATE TABLE orders (
  id SERIAL PRIMARY KEY,
  order_number VARCHAR(50) UNIQUE NOT NULL,
  user_id INTEGER REFERENCES users(id),
  total_amount NUMERIC(12,2),
  status VARCHAR(20),
  created_at TIMESTAMP DEFAULT NOW()
);
```

ERD