LAPORAN UJIAN AKHIR SEMESTER (UAS) - BACKEND DEVELOPMENT

Nama: [Nama Anda]
NIM: [NIM Anda]
Kelas: [Kelas Anda]
Mata Kuliah: Backend Development (DT199)
Topik Pilihan: Sistem Manajemen Rental PlayStation (Refactored to Microservices)

---

1. Ringkasan Proyek

Proyek ini adalah transformasi dari aplikasi monolitik **Rental_PS** menjadi arsitektur **Mikroservis**. Aplikasi ini mengelola penyewaan PlayStation, data game, dan transaksi pelanggan. Dalam arsitektur baru ini, logika bisnis dipisahkan ke dalam layanan independen yang berkomunikasi secara asinkron menggunakan RabbitMQ, sementara Laravel tetap digunakan sebagai antarmuka pengguna (Frontend).

2. Diagram Arsitektur Full-Stack

Sistem terdiri dari komponen-komponen berikut yang diorkestrasi menggunakan Docker Compose:

- **Frontend (Laravel)**: Menangani GUI dan interaksi pengguna. Berkomunikasi dengan mikroservis melalui API REST (Axios).
- **User Service (NestJS)**: Mengelola pendaftaran dan autentikasi pengguna. Mengirimkan event `user.created` ke RabbitMQ.
- **Order Service (NestJS)**: Mengelola transaksi penyewaan. Mendengarkan event dari RabbitMQ untuk sinkronisasi data.
- **RabbitMQ**: Broker pesan untuk komunikasi event-driven antar servis.
- **Database (MySQL)**: Menyimpan data persisten untuk aplikasi.

3. Daftar File Kode Penting

File	Fungsi Utama
`services/user-service/src/app.controller.ts`	Menangani registrasi dan mengirim event ke RabbitMQ.
`services/order-service/src/app.controller.ts`	Menangani pembuatan order dan mendengarkan event `user.created`.
`app/Http/Controllers/TransaksiController.php`	Controller Laravel yang telah direfaktor untuk memanggil API Order Service.
`docker-compose.yml`	Konfigurasi orkestrasi seluruh layanan.
`.github/workflows/ci.yml`	Pipeline CI/CD otomatis untuk build dan testing.

4. Contoh Kode dan Penjelasan

DTO (Data Transfer Object) - User Service

```
// services/user-service/src/dto/register.dto.ts
export class RegisterDto {
  @IsEmail()
  @IsNotEmpty()
  email: string;
  @IsNotEmpty()
  @MinLength(6)
  password: string;
}
```

Penjelasan: Digunakan untuk validasi input pada saat registrasi user sesuai kriteria soal.

RabbitMQ Handler - Order Service

```
// services/order-service/src/app.controller.ts
@EventPattern('user.created')
async handleUserCreated(data: any) {
  console.log('Order Service received user.created event:', data);
}
```

Penjelasan: Menangkap event asinkron dari User Service untuk memproses data user baru.

5. Kesimpulan

Implementasi ini berhasil memisahkan logika monolitik menjadi mikroservis yang skalabel tanpa merusak tampilan GUI asli. Penggunaan RabbitMQ memastikan komunikasi antar servis berjalan secara asinkron (event-driven), dan Docker mempermudah deployment di berbagai lingkungan. Tantangan utama adalah sinkronisasi data antar database servis, yang diselesaikan melalui pola event-driven.

Status CI/CD: Terkonfigurasi via GitHub Actions.
Status Kontainer: Terkonfigurasi via Docker Compose.