TUGAS INDIVIDU PENGANTARAN REKAYASA PERANGKAT LUNAK
PERBEDAAN PADA TIPE SOFTWARE ARSITEKTUR

Dosen Pengampu:

DIAN ANGGRAINI, S.ST., M.T.

Oleh:

Nama: Keidjaru Axiro

NIM: 2511446

PROGRAM STUDI REKAYASA PERANGKAT LUNAK

KAMPUS UPI DI CIBIRU

UNIVERSITAS PENDIDIKAN INDONESIA

2025

1. **Desain**

- **Monolithic:** Semua komponen dijahit dalam satu blok besar. desainnya sederhana tapi agak kaku dan sedikit rumit untuk dikembangkan
- **Layered:** Desain dibagi berdasarkan tanggung jawab (presentasi, logika bisnis, data). lebih modular dan mudah dipahami
- **Tiered:** Hampir sama dengan layered tapi tiap lapisan bisa dijalankan di server berbeda. fokus ke pemisahan tanggung jawab secara fisik
- **Microservices:** Desain paling modular; tiap layanan berdiri sendiri. kompleks tapi sangat fleksibel dan skalabel (Mudah di kembangkan dan di improvisasi tanpa merubah bagian lain)

2. **Pengembangan**

- **Monolithic:** Cepat di awal karena satu basis kode. tapi makin lama makin berat, karena perubahan kecil bisa memengaruhi seluruh sistem
- **Layered:** Pengembangan lebih terstruktur, tiap tim bisa fokus pada lapisan tertentu
- **Tiered:** Tiap tier bisa dikembangkan paralel oleh tim berbeda, lebih mudah untuk proyek besar
- **Microservices:** Tiap layanan bisa dikembangkan dengan teknologi dan bahasa berbeda. ideal untuk tim besar yang independen, tapi butuh manajemen tinggi

3. **Penyebaran (Deployment)**

- **Monolithic:** Sekali build, sekali deploy. tapi kalau ada bug kecil, harus rilis ulang seluruh aplikasi (redeploy)
- **Layered:** Masih satu paket besar, tapi bisa diatur deploymentnya per layer jika sistemnya mendukung
- **Tiered:** Lebih fleksibel karena tiap tier dihosting di server atau mesin virtual terpisah
- **Microservices:** Setiap layanan bisa dideploy sendiri (continuous deployment friendly). cocok untuk CI/CD

4. **Debugging**

- **Monolithic:** Mudah di awal karena semua dalam satu proses. tapi kalau sistem sudah besar, melacak bug jadi mimpi buruk

- **Layered:** Lebih mudah melacak bug berdasarkan lapisan yang terlibat
- **Tiered:** Bisa dilakukan debugging per tier, tapi perlu komunikasi antar server yang jelas
- **Microservices:** Debugging lebih kompleks karena banyak layanan yang saling berinteraksi, tapi isolasi bug jadi lebih mudah dilakukan

5. Modifikasi

- **Monolithic:** Perubahan kecil bisa berdampak ke seluruh sistem. risiko tinggi
- **Layered:** Lebih aman memodifikasi satu layer tanpa menyentuh lainnya
- **Tiered:** Perubahan bisa dilakukan pada satu tier tanpa mengganggu yang lain, asal API tetap konsisten
- **Microservices:** Paling mudah dimodifikasi; tiap layanan bisa diganti total tanpa menurunkan seluruh sistem

6. Skala (Scalability)

- **Monolithic:** Sulit diskalakan, karena harus menduplikasi seluruh aplikasi
- **Layered:** Bisa diskalakan sebagian, tapi masih terbatas karena satu deployment besar
- **Tiered:** Lebih mudah diskalakan, misalnya hanya menambah server di tier data
- **Microservices:** Skalabilitas terbaik. setiap layanan bisa diskalakan secara independen sesuai kebutuhan beban kerja

Table Aspek:

Aspek	Monolithic	Layered	Tiered	Microservices
Desain	Sederhana tapi kaku	Modular	Modular fisik	Paling modular
Pengembangan	Cepat tapi berat di akhir	Terstruktur	Paralel	Independen
Penyebaran	Sekali deploy	Semiterpisah	Multiserver	Per layanan
Debugging	Sulit saat besar	Lebih mudah	Per-tier	Kompleks tapi terisolasi
Modifikasi	Risiko tinggi	Aman per layer	Aman per tier	Paling fleksibel
Skala	Lemah	Sedang	Baik	Terbaik