

Praktikum Rekayasa Perangkat Lunak

Pertemuan 10

→ Hari ini belajar apa??

Arsitektur Perangkat Lunak

Apa itu arsitektur perangkat lunak

Package Diagram

Apa itu package diagram? Apa gunanya?



1. Arsitektur Perangkat Lunak

Definisi, aspek, dan Jenis-jenis

Definisi

Beberapa *software architect* mendefinisikan arsitektur perangkat lunak sebagai ***blueprint sistem***, sementara yang lain mendefinisikannya sebagai ***roadmap untuk mengembangkan sistem***.

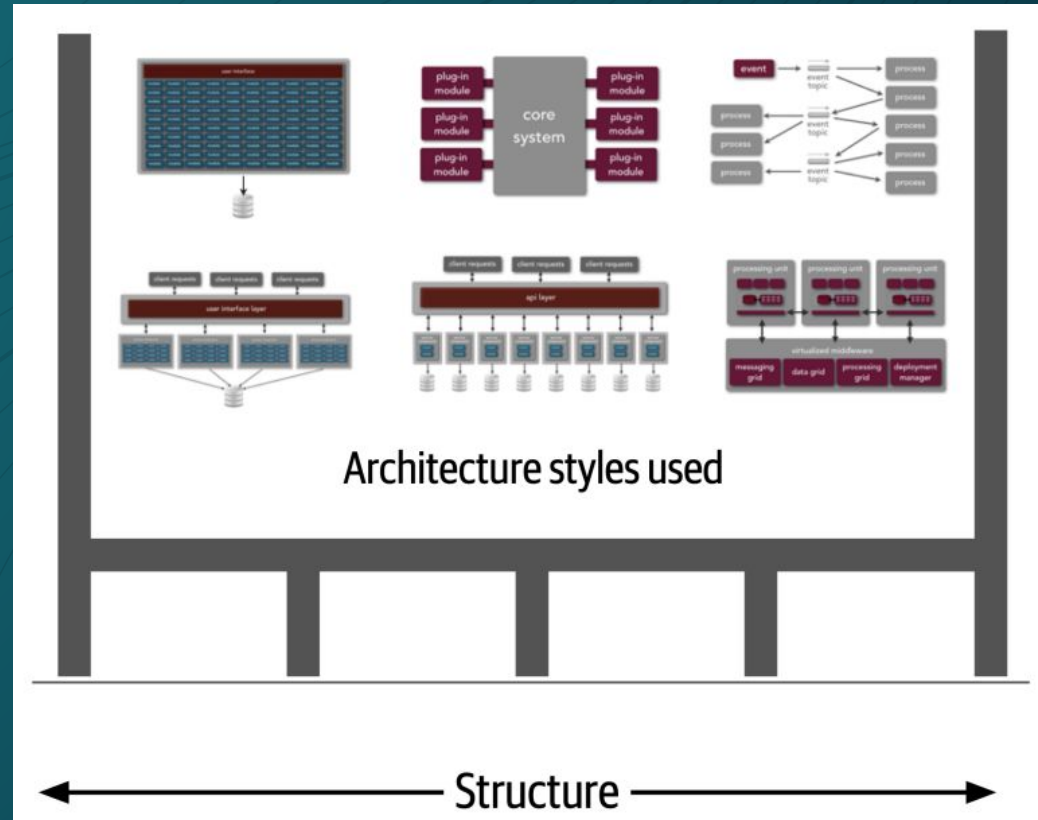
Arsitektur perangkat lunak **membahas mengenai struktur sistem, karakteristik arsitektur, keputusan arsitektur, dan prinsip desain.**

Aspek Arsitektur Perangkat Lunak

- **Struktur sistem** : Jenis bentuk struktur sistem yang diimplementasikan (*layered, microservices, microkernel*).
- **Karakteristik arsitektur** : *Success criteria* / fungsionalitas sistem (*availability, reliability, scalability, security, dll.*)
- **Keputusan arsitektur** : Aturan bagaimana sistem harus dibangun
- **Prinsip desain** : Panduan bagaimana sistem harus dibangun

Jenis-jenis Struktur Sistem

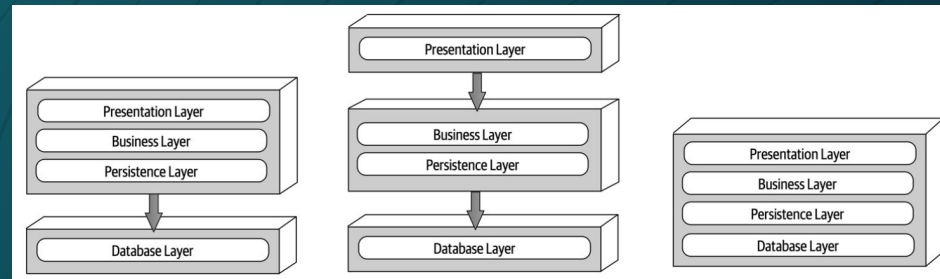
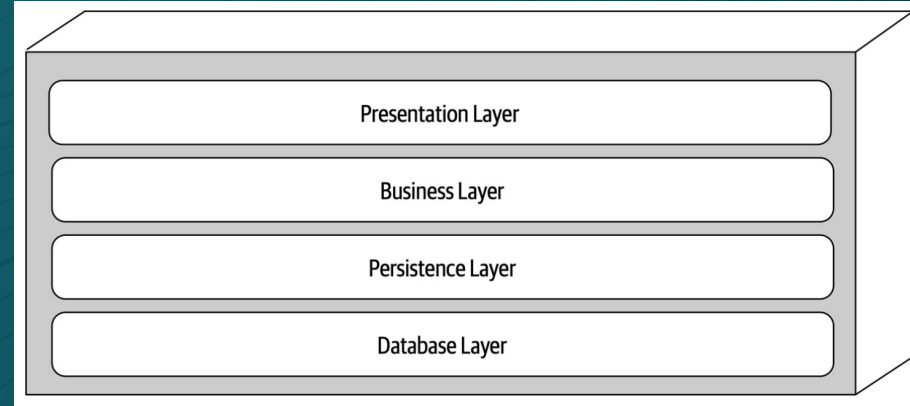
Disebut juga sebagai pola arsitektur, mendeskripsikan hubungan antar komponen yang mencakup berbagai karakteristik arsitektur.



Layered / n-tiered architecture

Komponen dalam layered architecture diatur ke dalam lapisan logis secara horizontal, dengan setiap lapisan melakukan peran tertentu dalam aplikasi.

Meskipun tidak ada batasan khusus dalam hal jumlah dan jenis lapisan yang harus ada, sebagian besar arsitektur berlapis terdiri dari empat lapisan, yaitu : *presentation*, *business*, *persistence*, and *database*.



Kelebihan dan Kekurangan

Kelebihan :

- Tergolong simpel
- Banyak developer yang familiar
- Cocok jika budget dan waktu sangat terbatas

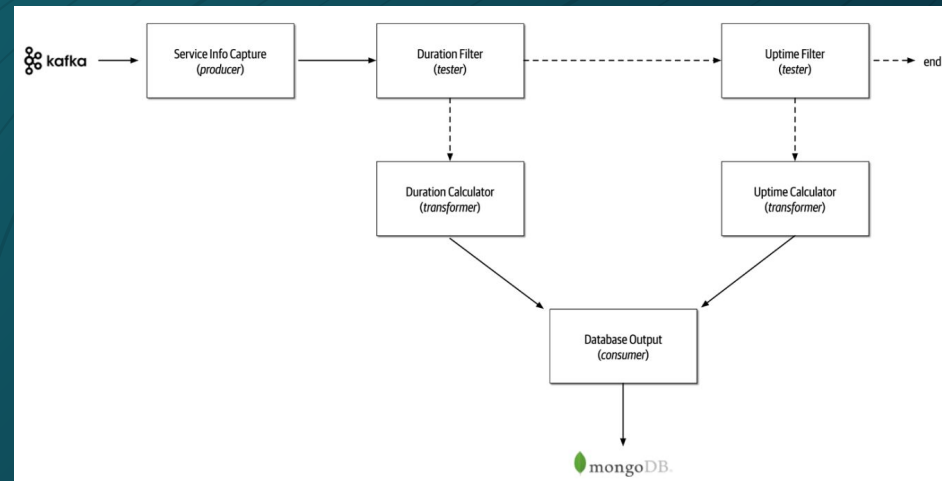
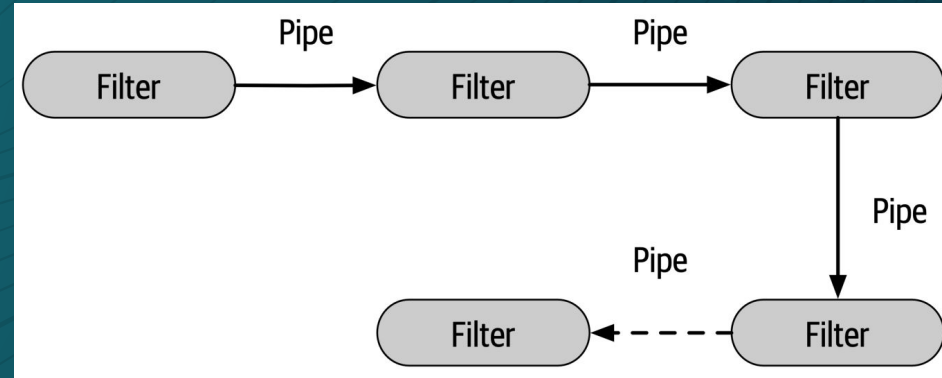
Kekurangan :

- Tidak scalable
- Kurang reliable
- Tidak modular
- Effort untuk deploy tinggi

Pipeline Architecture

Terdiri atas *pipes* dan *filters* dimana *pipe* membentuk komunikasi antar filter, biasanya secara searah.

Filters sendiri merupakan komponen yang terisolasi dan saling independen yang mana masing-masing hanya melakukan satu tugas saja.



Kelebihan dan Kekurangan

Kelebihan :

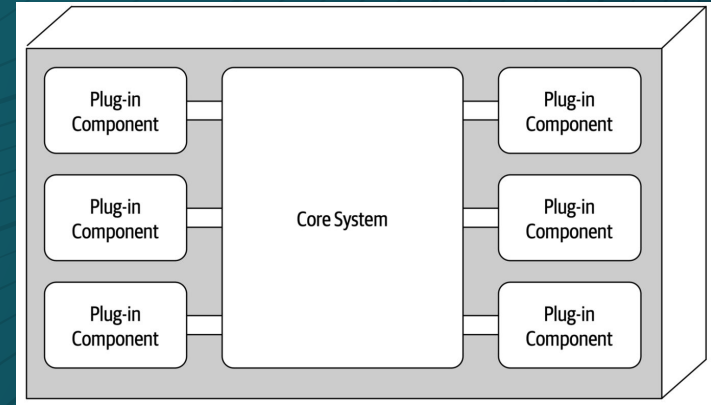
- Tergolong simpel
- Lebih modular ketimbang monolitik biasa dan layered
- Cost-effective

Kekurangan :

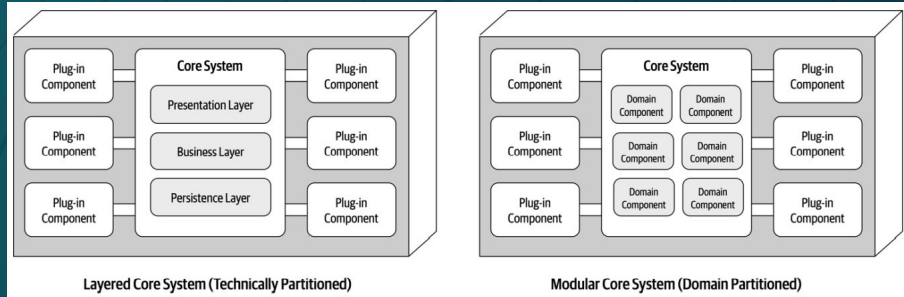
- Tidak scalable
- Kurang reliable
- Effort untuk deploy dan test tinggi

Microkernel Architecture

Gaya arsitektur mikrokernel adalah arsitektur monolitik yang relatif sederhana yang terdiri dari dua komponen arsitektur: sistem inti dan komponen plug-in.



Logika aplikasi dibagi antara komponen plug-in independen dan sistem inti dasar, menyediakan ekstensibilitas, kemampuan beradaptasi, dan isolasi fitur aplikasi dan logika pemrosesan kustom.



Kelebihan dan Kekurangan

Kelebihan :

- Lebih mudah dites dari arsitektur monolitik yang lain
- Lebih modular dari arsitektur monolitik yang lain
- Cost-effective
- Support domain partitioning

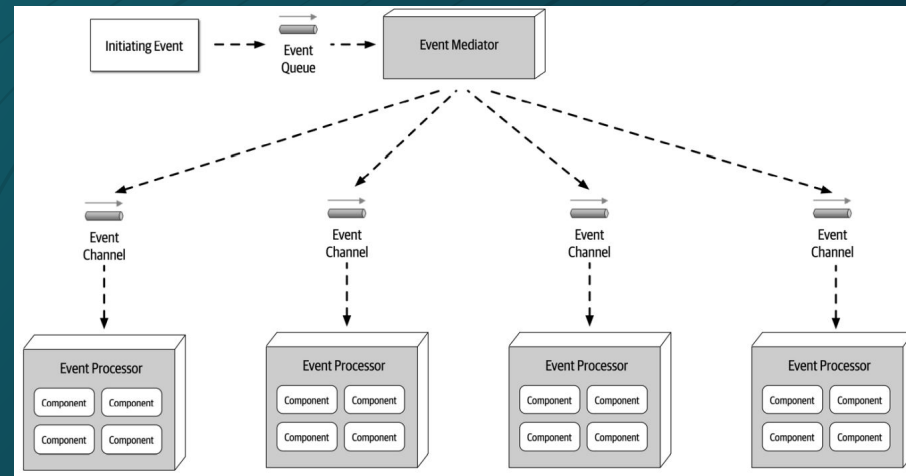
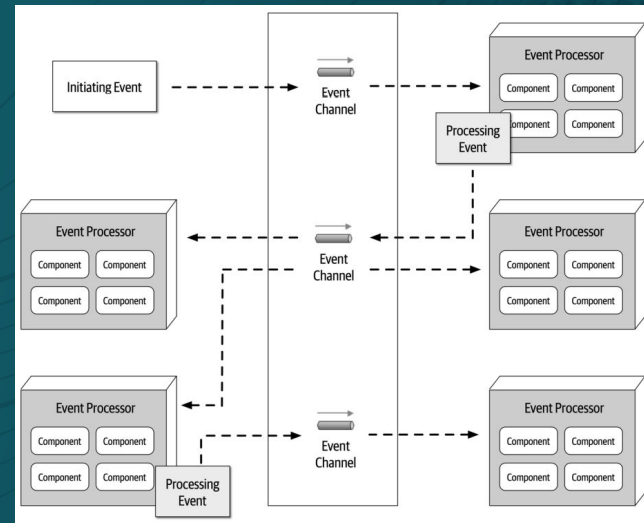
Kekurangan :

- Tidak scalable
- Kurang reliable
- Effort untuk deploy dan test tinggi

Event-driven Architecture

Event-driven Architecture terdiri dari komponen pemrosesan *events* yang dipisahkan (*decoupled*) yang secara asinkronus menerima dan memproses *events*.

Arsitektur ini dapat digunakan sebagai gaya arsitektur mandiri atau disematkan dalam gaya arsitektur lain (seperti arsitektur *event-driven microservices*).



Kelebihan dan Kekurangan

Kelebihan :

- Scalable
- Lebih modular dari arsitektur monolitik
- Fault-tolerant
- Support *asynchronous processing*

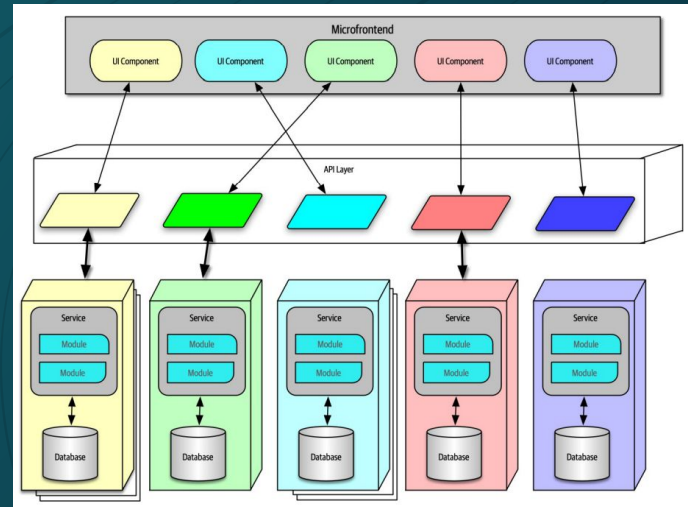
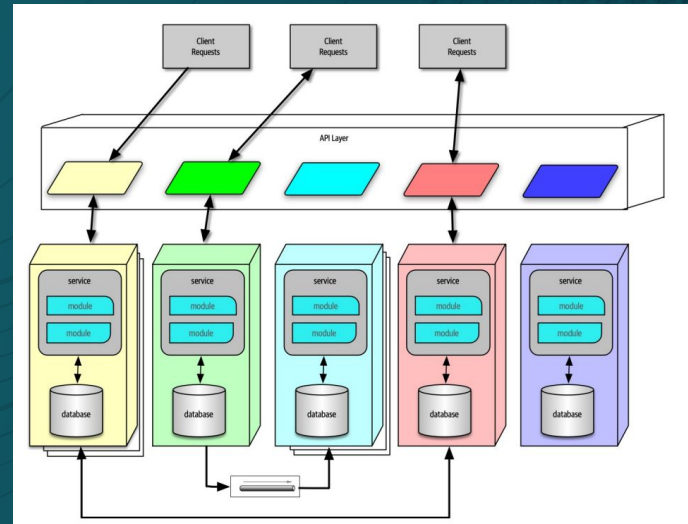
Kekurangan :

- Kompleks
- Tidak begitu mudah untuk dites

Microservices Architecture

Microservices architecture membentuk arsitektur terdistribusi: setiap layanan berjalan dalam prosesnya sendiri, yang awalnya menggunakan komputer fisik tetapi dengan cepat berkembang menjadi *virtual machines* dan *containers*.

Memisahkan layanan ke tingkat ini memungkinkan solusi sederhana untuk masalah umum dalam arsitektur yang banyak menampilkan infrastruktur multitenant untuk aplikasi hosting.



Kelebihan dan Kekurangan

Kelebihan :

- Scalable
- Modular
- Fault-tolerant
- Support *asynchronous processing*
- Mudah di tes dan deploy

Kekurangan :

- Kompleks
- Mahal



2. Package Diagram

Definisi, Kegunaan

Definisi

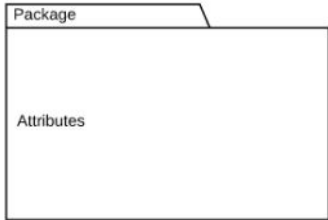

Package diagram merupakan salah satu dari jenis UML (Unified Modeling Language) yang digunakan untuk mengelompokkan elemen-elemen model dari use case dan class diagram.

Package Diagram memisahkan tampilan, domain, dan akses data ke dalam paket yang terpisah. Dengan adanya diagram ini, dapat mempermudah pembuatan sistem dengan cara mengumpulkan atribut-atribut yang sejenis.

Tujuan

- Memberikan pandangan yang jelas tentang struktur hierarkis dari berbagai elemen UML dalam sistem yang diberikan.
- Menyederhanakan *class* diagram yang kompleks menjadi visual yang tertata dengan baik.
- Menawarkan visibilitas tingkat tinggi yang berharga ke dalam proyek dan sistem skala besar.
- Memperjelas berbagai proyek dan sistem secara visual.
- Paket digambarkan sebagai folder file dan dapat digunakan pada salah satu diagram UML.

Komponenten

Symbol Image	Symbol Name	Description
	Package	Groups common elements based on data, behavior, or user interaction
	Dependency	Depicts the relationship between one element (package, named element, etc) and another

Penjelasan

- Package : Ruang nama yang digunakan untuk mengelompokkan elemen yang terkait secara logis dalam suatu sistem. Setiap elemen yang terkandung dalam paket harus menjadi elemen yang dapat dikemas dan memiliki nama yang unik.
- Packageable element : Sebuah elemen bernama, mungkin dimiliki langsung oleh sebuah paket. Ini dapat mencakup peristiwa, komponen, kasus penggunaan, dan paket itu sendiri. Elemen yang dapat dikemas juga dapat dirender sebagai persegi panjang dalam sebuah paket, diberi label dengan nama yang sesuai.

Penjelasan

- Dependensi : Representasi visual tentang bagaimana satu elemen (atau kumpulan elemen) bergantung pada atau memengaruhi yang lain. Dependensi dibagi menjadi dua kelompok: dependensi akses dan impor. (Lihat bagian selanjutnya untuk info lebih lanjut.)
- Element import : Hubungan terarah antara namespace yang diimpor dan elemen packageable yang diimpor. Ini digunakan untuk mengimpor elemen individual tertentu tanpa menggunakan import paket dan tanpa membuatnya publik di dalam namespace.

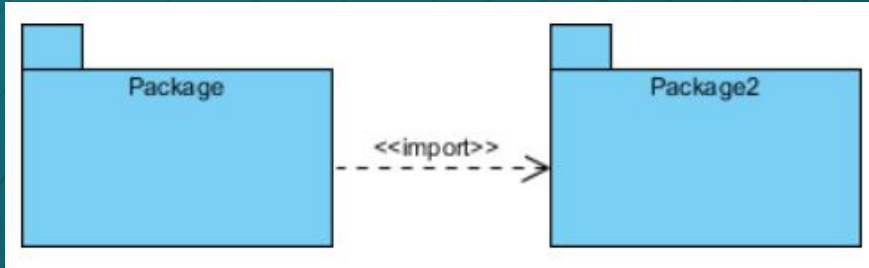
Penjelasan

- Package import : Hubungan terarah antara/dan namespace yang mengimport dan paket yang diimpor. Jenis hubungan terarah ini menambahkan nama anggota paket yang diimpor ke namespace-nya sendiri
- Package merge : Hubungan terarah di mana isi dari satu paket diperluas oleh konten yang lain. Pada dasarnya, isi dari dua paket digabungkan untuk menghasilkan paket baru.

Jenis-jenis Dependencies

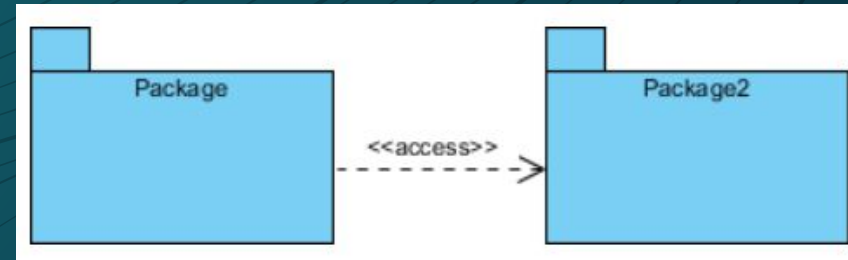
Import

Package meng-import fungsionalitas dari package lain.



Access

Package membutuhkan bantuan dari package lain.

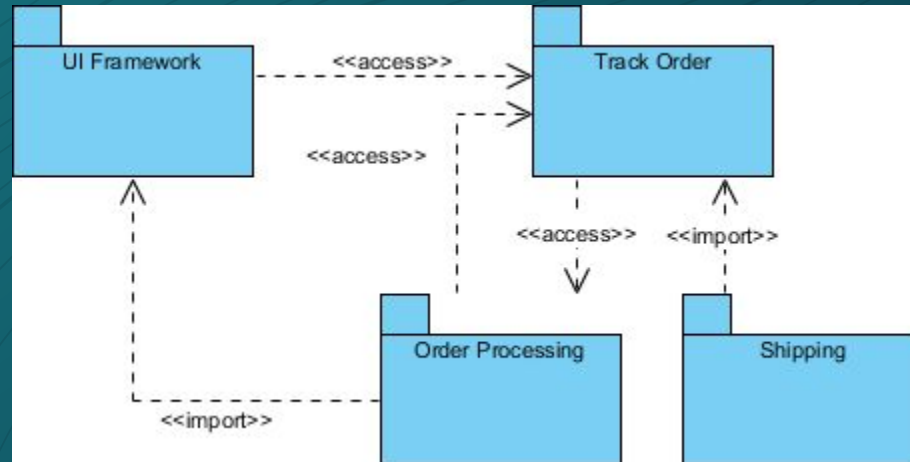


Catatan mengenai dependencies

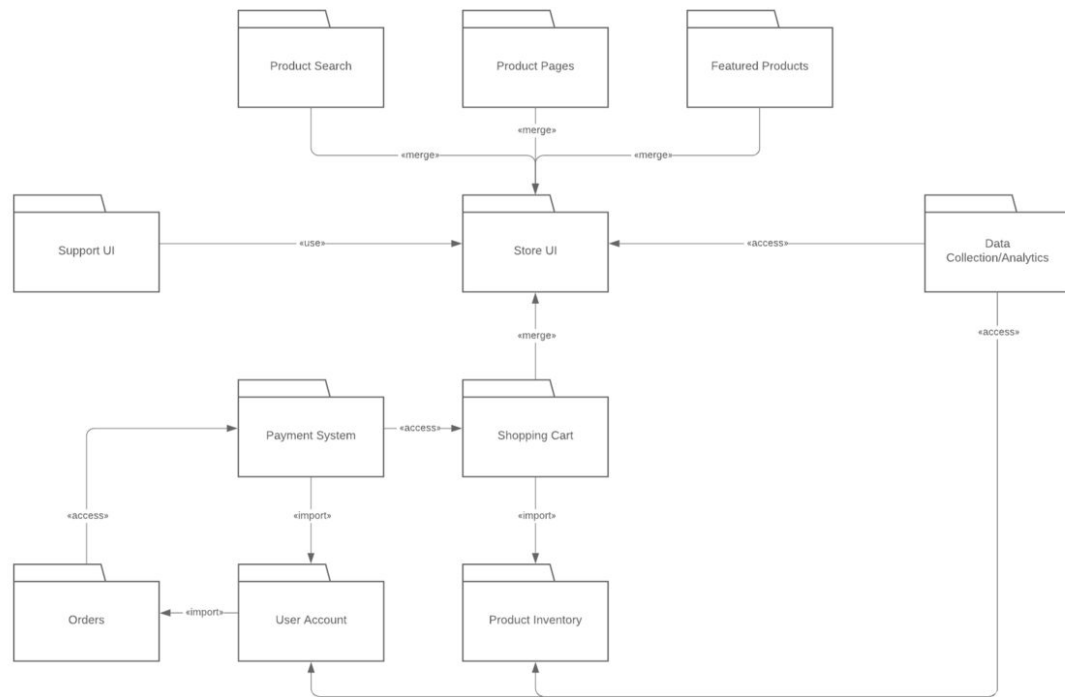
➤ Dependensi juga dapat dipecah lebih lanjut ke dalam kategori berikut:

- *Usage*: Terjadi ketika elemen bernama tertentu memerlukan elemen lain untuk definisi dan penerapan penuhnya. Contoh: klien dan pemasok.
- Abstraksi: Menghubungkan dua elemen yang mewakili konsep yang sama pada tingkat abstraksi yang berbeda dalam sistem (biasanya hubungan antara klien dan *supplier*).
- Deployment: Menggambarkan penyebaran artefak ke target penyebaran.

Contoh



Contoh





Tugas

udah ketebak lah ya

Tugas

Buat **package diagram** aplikasi sesuai proposal kelompok yang telah dibuat dan berikan penjelasannya.

Kumpulkan di Google Classroom

Format:

PackageDiagram_namaprojek.pdf

Deadline : H-1 Praktikum Berikutnya, 23.59 **(KUMPUL PERWAKILAN SAJA!)**

Terima kasih

Kalo ada yang mau ditanya, ya tanya aja