codingstudio

DAY 8

ARSITEKTUR APLIKASI PROFESIONAL

BY ALFADJRI DWI FADHILAH

codingstudio

Timeline

1 VERSION CONTROLL SYSTEM (VCS)

2 APPLICATION DESIGN PATTERNS

3 ARSITEKTUR AND STRUCTURE PROJECT

codingstudio

VERSION CONTROLL SYSTEM (VCS)

codingstudio

VERSION CONTROL

Version Control adalah sistem yang merekam semua perubahan pada kode kita dari waktu ke waktu yang memungkinkan untuk melihat kembali versi sebelumnya dan berkolaborasi dengan aman

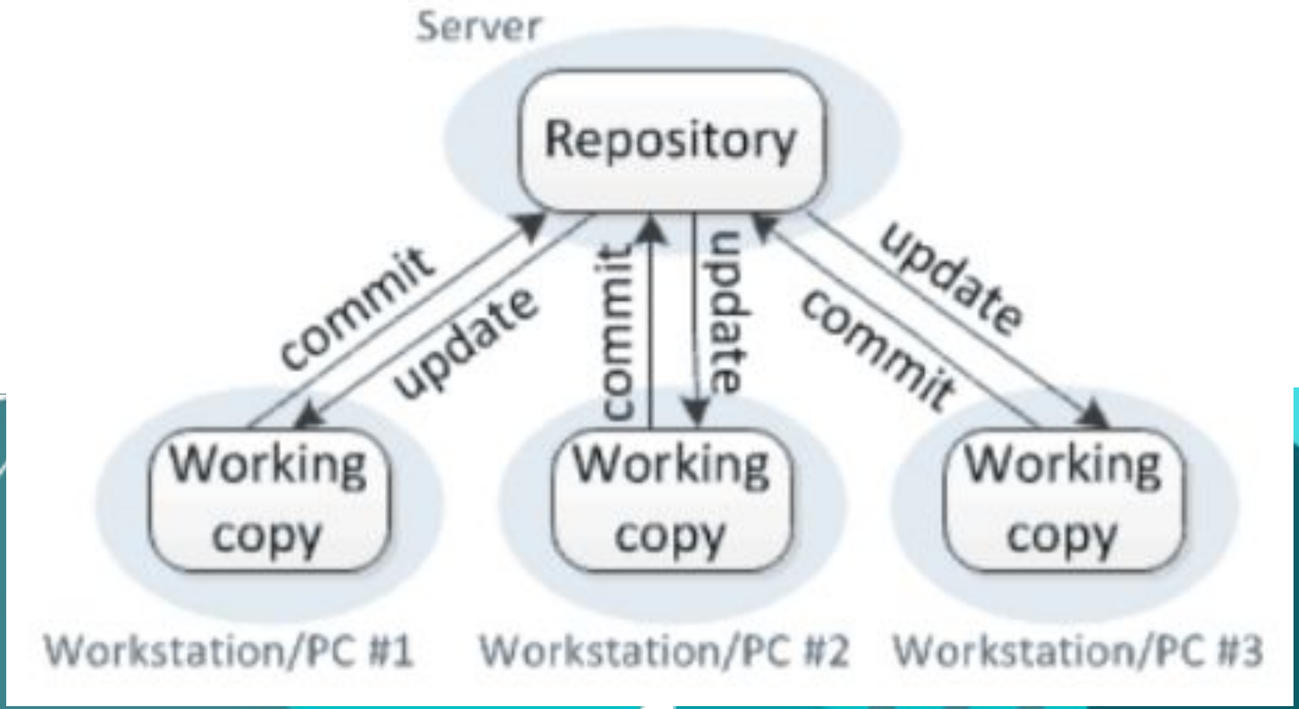
codingstudio

Manfaat Menggunakan Version Control

1. Mesin waktu untuk kode : memudahkan melihat riwayat atau checkpoint dan kembali versi mana pun jika terjadi kesalahan
2. Kolaborasi Tim Tanpa Konflik : Membantu time bekerja pada file yang sama menggabungkan perubahan, dan mengelola konflik secara sistematis
3. Cadangan (Backup) yang Terdistribusi : Setiap Anggota tim memiliki salinan lengkap dari riwayat project, sehingga lebih aman
4. Manajemen Rilis Profesional : Memudahkan pemberian label versi pada aplikasi yang sudah jadi

codingstudio

CENTRALISED VERSION CONTROL SYSTEM (CENTRALISED VCS)



codingstudio

Centralised Version Control System (Centralised VCS)

Dalam model ini terdapat satu server pusat yang bertindak sebagai satu-satunya repository dan menyimpan seluruh riwayat version project

Cara Kerjanya adalah

Check-out : Pengembang mengambil atau “Check-out” salinan kerja (working copy) dari repository pusat untuk memulai mengerjakan kode

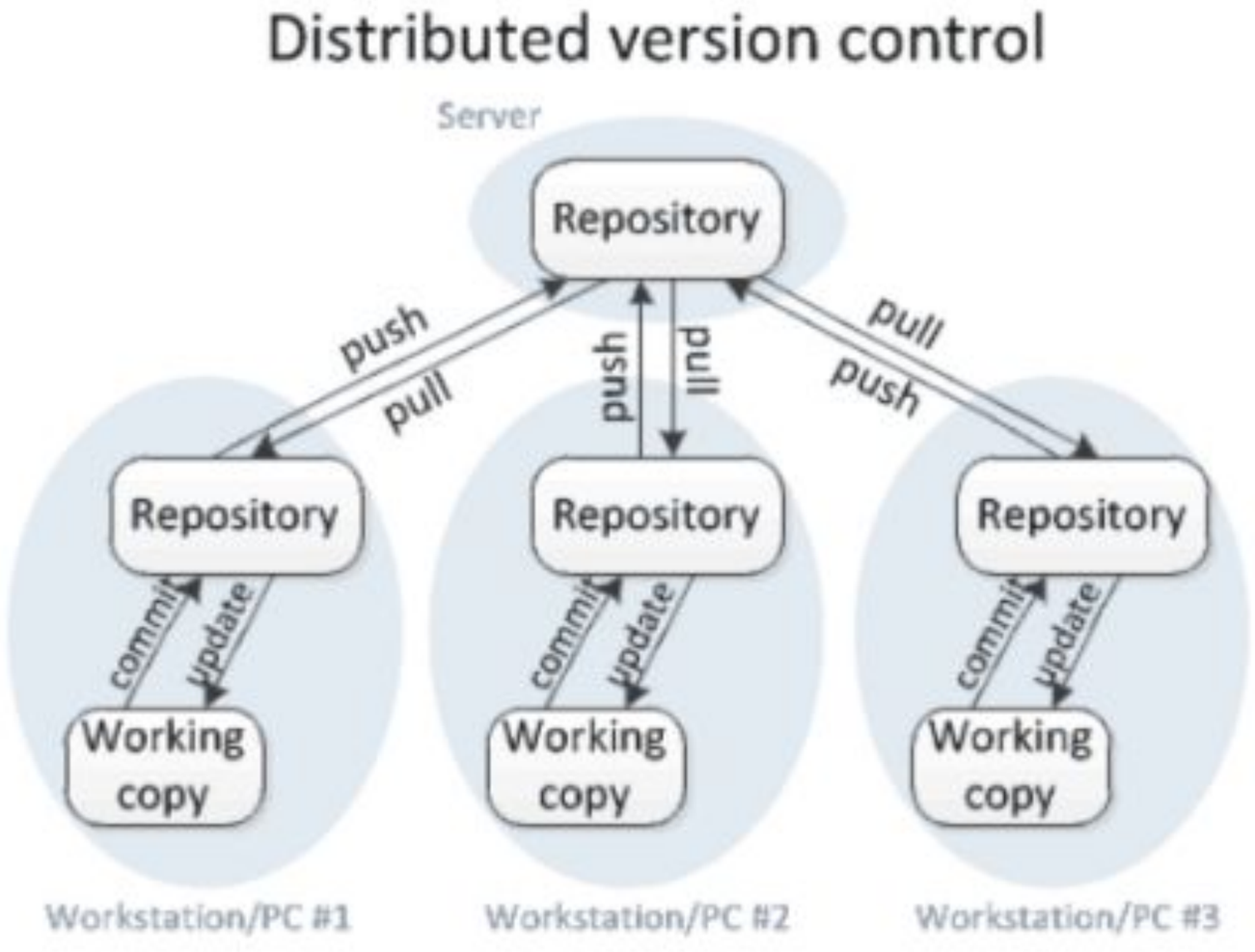
codingstudio

Centralised Version Control System (Centralised VCS)

Commit : Setelah selesai , pengembang mengirimkan kembali hasil perubahannya (commit) ke repository pusta agar tercatat dan bisa dilihat oleh anggota tim lainnya

codingstudio

DISTRIBUTED VERSION CONTROL SYSTEM (DISTRIBUTED VCS)



codingstudio

Distributed Version Control System (Distributed VCS)

Dalam model ini , setiap pengembangan menyalin atau clone keseluruhan repository ke komputer lokal mereka

Kerja Local & Offline : Pengembangn dapat bekerja secara independen, melalui commit dan membuat cabang (branch) tanpa harus terhubung ke pusat server

Sinkronisasi: Koneksi ke server hanya diperlukan saat ingin mengirim (push) perubahan ke tim atau mengambil (pull) pembaruan dari tim

codingstudio

GIT

Git adalah sebuah perangkat lunak Distributed Version Control System (DVCS) yang bersifat gratis dan open source. Saat ini, git merupakan standar industri yang digunakan oleh jutaan developer di seluruh dunia untuk berkolaborasi dalam pengembagnan aplikasi.

codingstudio

Kenapa Git Begitu Populer

Gratis dan Open Source : Siapapun dapat menggunakan dan berkontribusi pada pengembangan git tanpa biaya

Branching yang kuat : Git memiliki fitur pembuatan cabang atau branch yang sangat mudah dan efisien

Kecepatan dan Kinerja : Karena sifatnya yang berdistribusi hampir semua operasi seperti commit dan melihat riwayat dilakukan secara lokal dan terasa instan

Keamanan & Integritas Data : Git menjamin riwayat project aman. Setiap perubahan dicatat dengan cryptographic hash, sehingga hampir tidak memungkinkan mengubah riwayat tanpa terdeteksi.

TIGA AREA KERJA UTAMA

codingstudio

Untuk memahami Git, kita harus paham bahwa setiap proyek git memiliki tiga “area” utama :

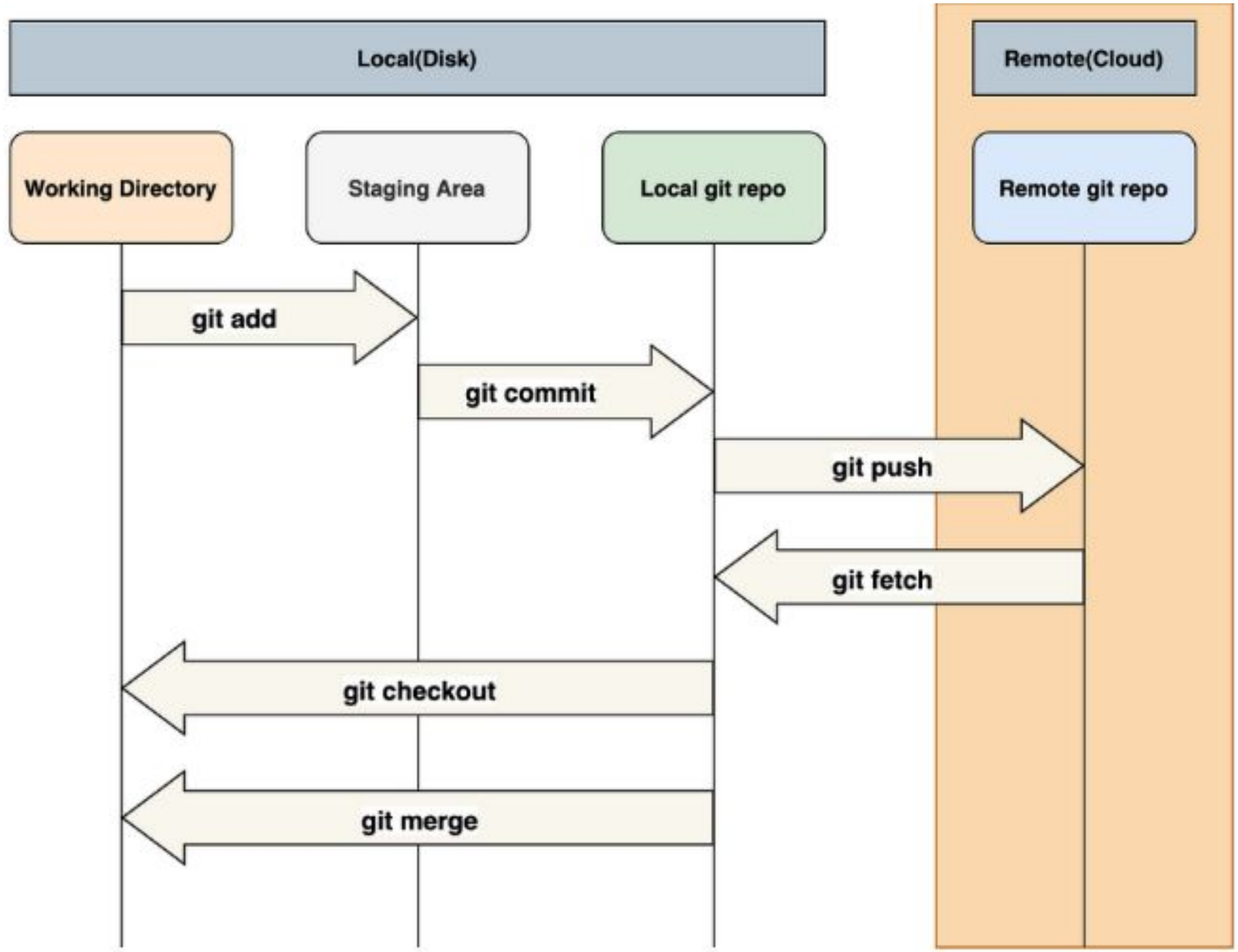
Working Directory (Area kerja) adalah folder proyek yang kamu lihat dan bisa kamu edit isinya secara langsung Tempat di mana kamu menulis dan mengubah kode

Staging Area (Area Tunggu) Sebuah area untuk mendaftarkan atau mempersiapkan perubahan mana saja dari Working Directory yang akan dimasukan ke dalam commit (snapshot) berikutnya.

Git Directory (Repository) Merupakan Sebuah folder tersembunyi bernama “.git” yang berisi seluruh riwayat commit , branch dan semua informasi lain tentang proyek

CARA KERJA GIT

codingstudio



codingstudio

Hand On

Set up Git

Inisialisasi repository atau download repository

bagaimana cara melihat log

Membuat branch dan merubah branch

melihat apa yang diubah pada pada log sebelum dan yang terbaru

mengirimkan perubahan

mengirimkan pesan perubahan

menggabungkan perubahan ke branch yang ada

memperbaiki branch yang konflik

codingstudio

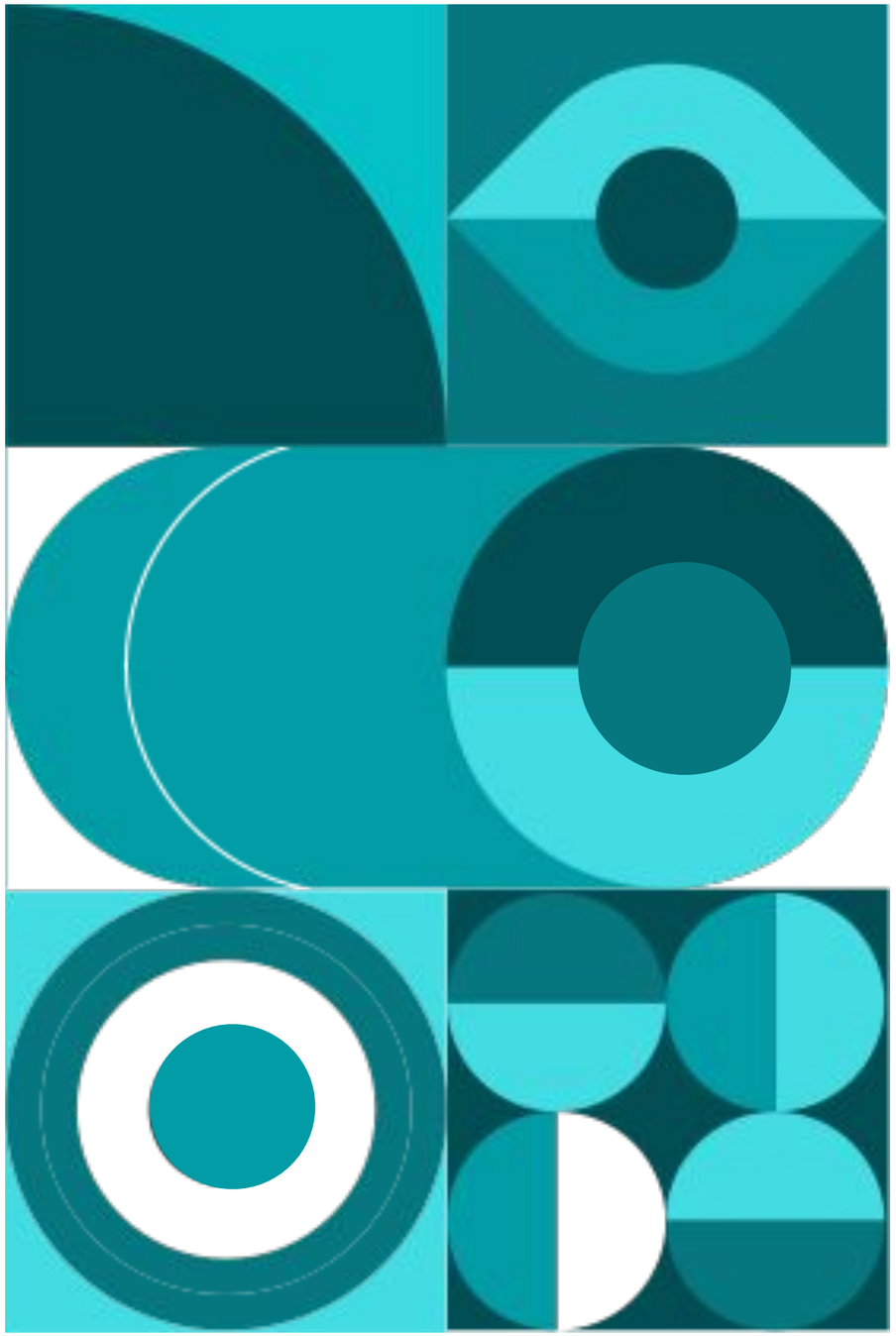
PERINTAH DASAR GIT

git init : digunakan untuk membuat project baru yang akan di simpan di git

git clone <repository-url> : digunakan untuk mendownload ke local atau mengcopy repository untuk saling bekerja sama atau hanya menggunakan saja

git add <file> : untuk memberitahukan snapshot atau file yang akan mengalami perubahan

git commit -m “commit message” ; Memberikan pesan pada staging area apa yang sudah dilakukan



codingstudio

PERINTAH DASAR GIT

git status : menampilkan status yang ada apa yang sudah di modifikasi atau dilakukan

git log : menampilkan riwayat commit pada repository

git branch : menampilkan semua cabang yang ada pada repository

git branch <branch-name> : membuat cabang dengan spesifikasi

git checkout < branch-name> : menukar ke cabang yang di inginkan

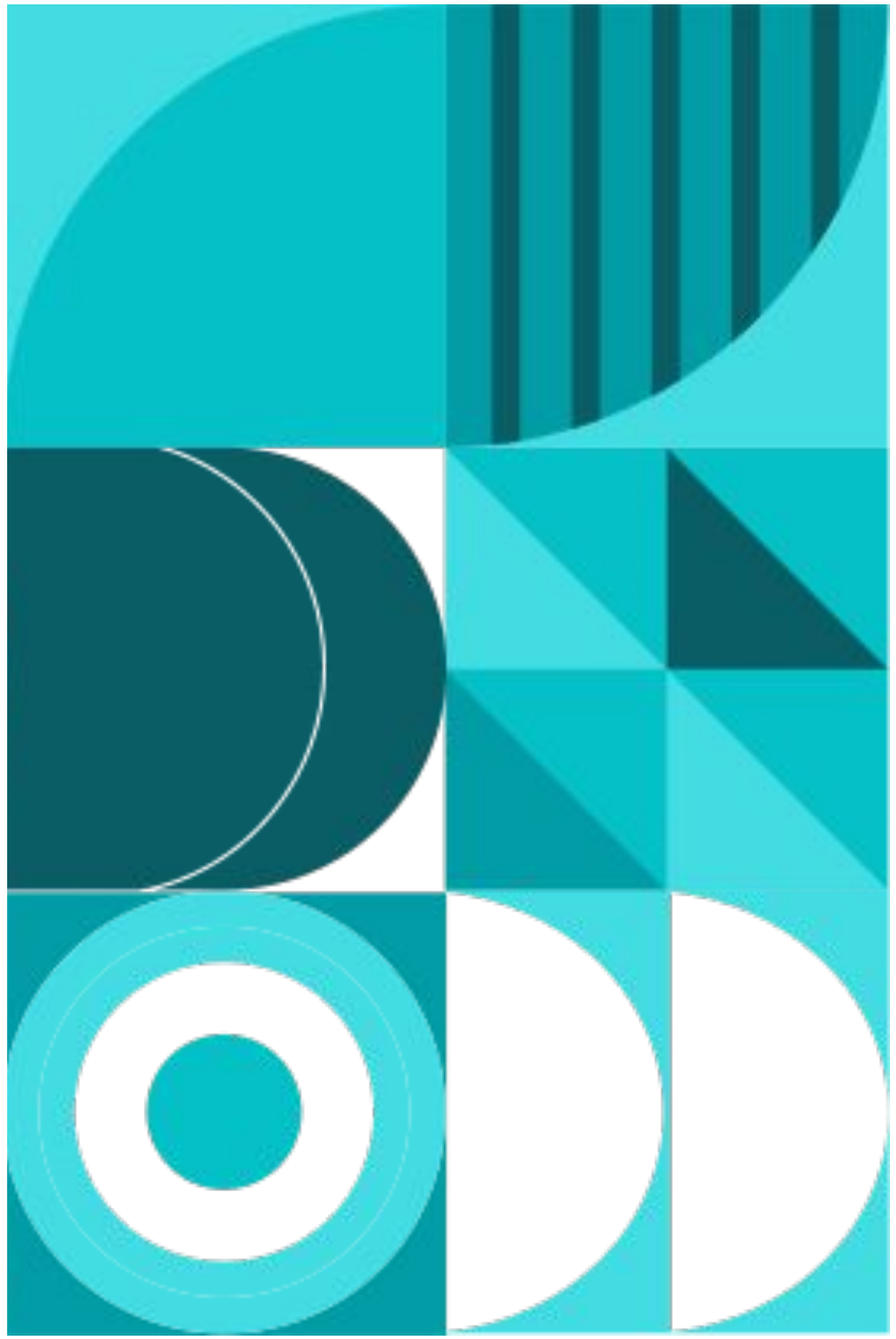
codingstudio

PERINTAH DASAR GIT

git merge < branch-name> : menggabungkan cabang dengan repository yang dituju

git push : upload perubahan yang ada pada local ke repository yang ada di internet

git pull : Mendownload atau mencocokan perubahan yang terjadi di repository internet dengan file yang ada di local



codingstudio

APPLICATION DESIGN PATTERNS

codingstudio

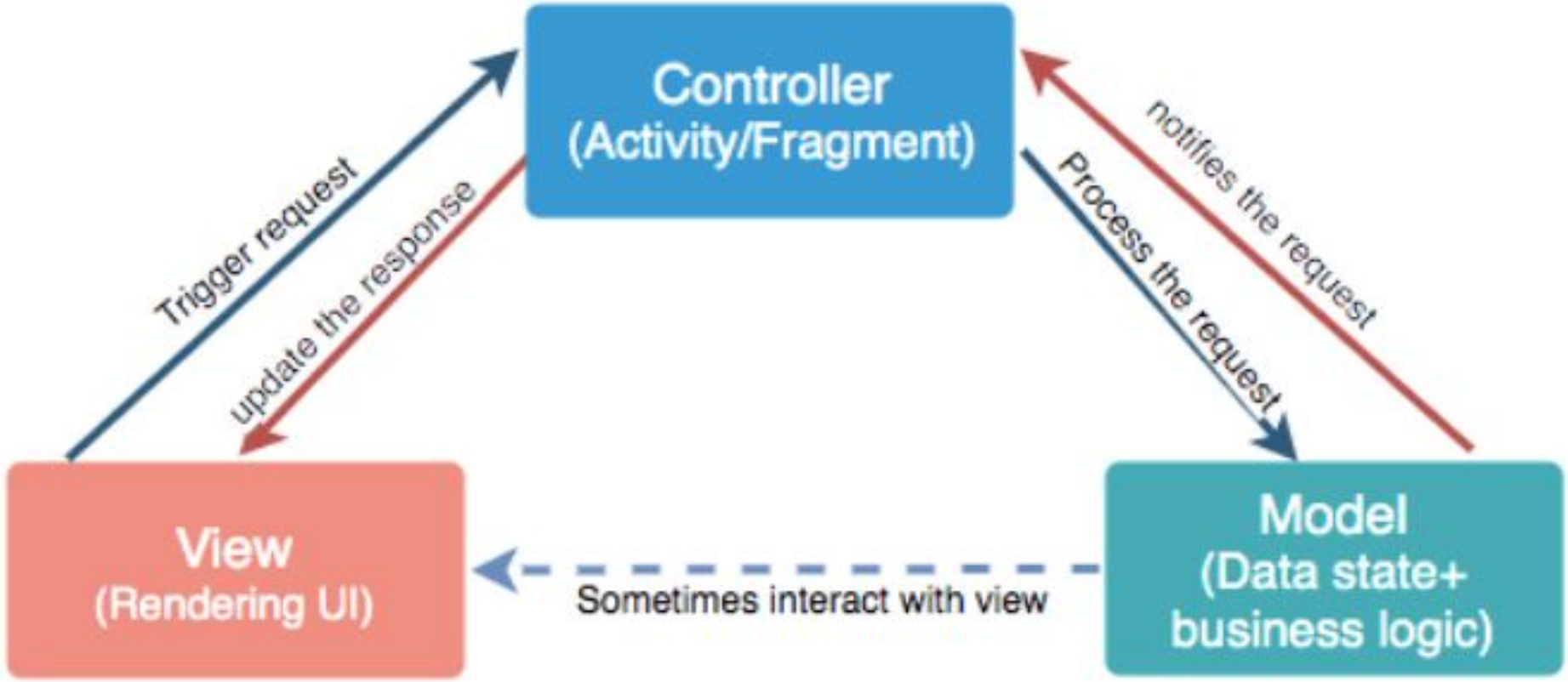
APPLICATION DESIGN PATTERN

Struktur project UI adalah kerangka kerja organisasi kode yang memecah tanggung jawab antara model (data) , View (tampilan) dan Perantara (Controller / Presenter/ View Model) untuk meningkatkan keterbacaan, keterujian dan pemeliharan kode.

- Adapun beberapa variasi yang bisa digunakan
  - MVC (Model View Controller)
  - MVP (Model View Presenter)
  - MVVM (Model View Viewmodel)

MVC (MODEL VIEW CONTROLLER)

codingstudio



codingstudio

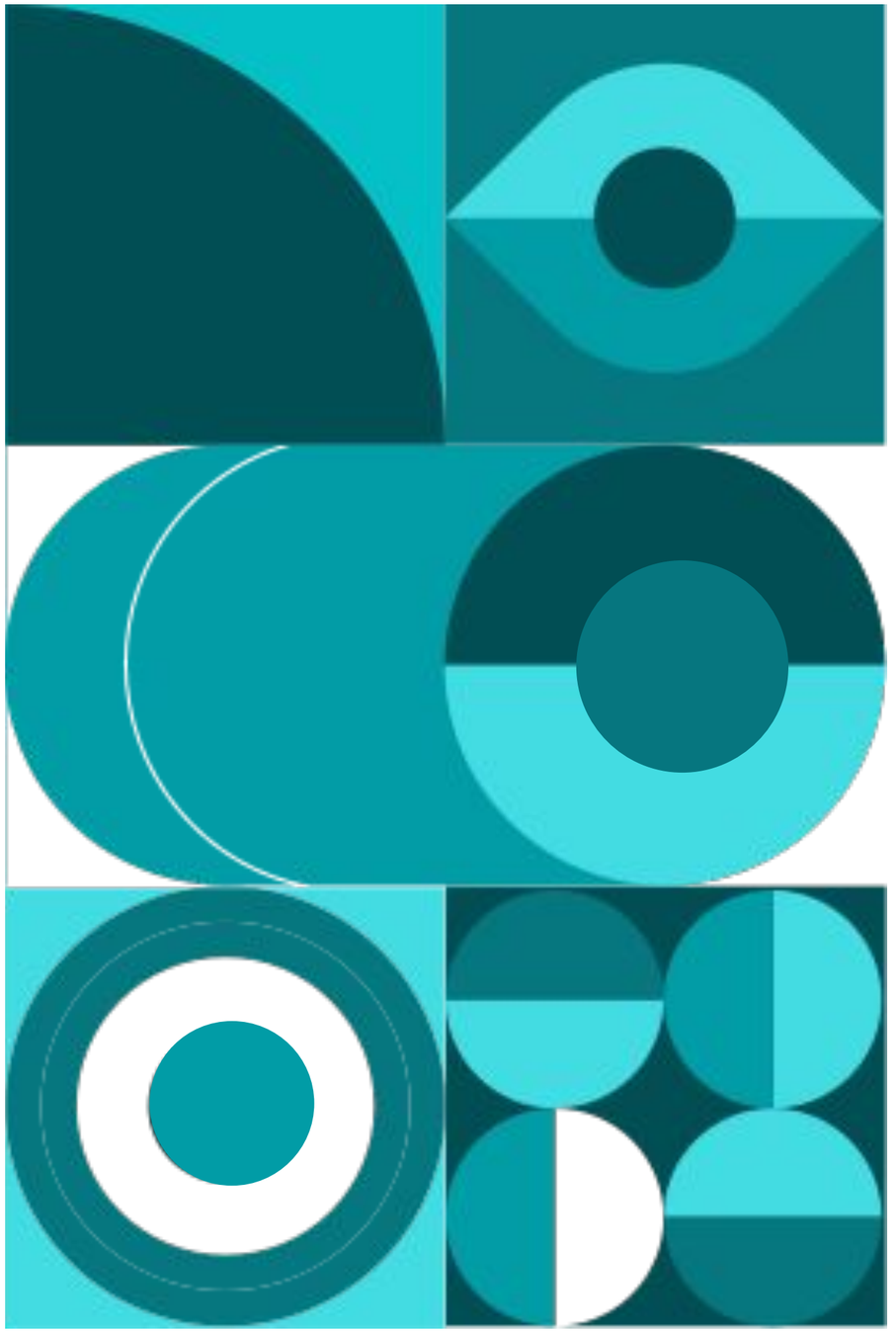
Tugas MVC (Model View Controller)

- **Model** : Menentukan jenis data dan business logic.
- **View** : Merupakan hasil yang dilihat.
- **Controller** : Bertindak sebagai penghubung antara model dan view

codingstudio

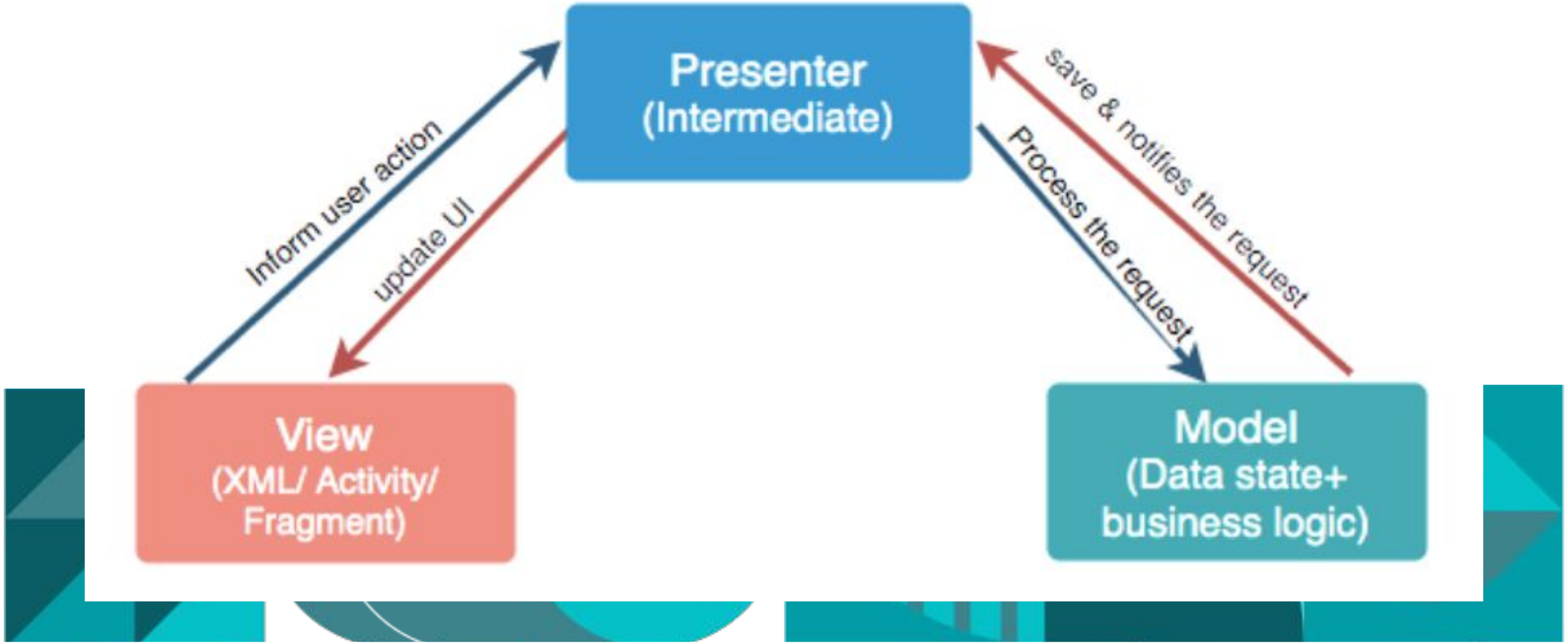
Cara Kerja MVC (Model View Controller)

- **Aksi pengguna** : Pengguna berinteraksi dengan view
- **View ke Controller** : View memberitahukan controller tentang aksi tersebut
- **Controller ke Model** : Controller memperbarui Model berdasarkan input
- **Model ke View** : Model (atau Controller) memberitahu View untuk memperbarui tampilan
- **Kunci** : View dan Controller saling mengetahui.Logika presentasi bisa tersebar



MVP (MODEL VIEW PRESENTER)

codingstudio



codingstudio

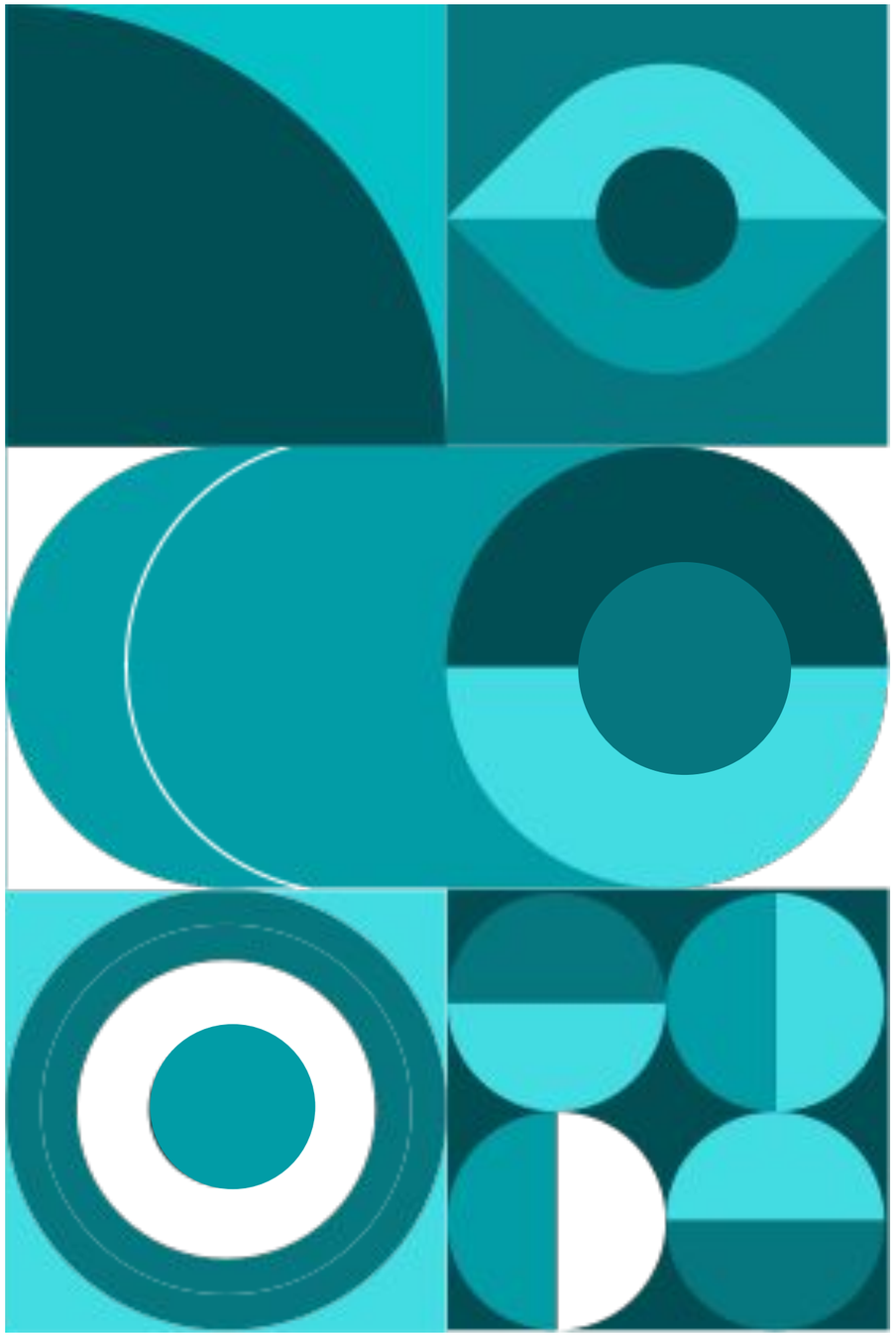
Tugas MVP (Model View Presenter)

- **Model** : Menentukan jenis data dan business logic
- **View** : Bertanggung jawab untuk menampilkan data kepada pengguna
- **Presenter** : Bertindak sebagai perantara antara model dan view yang bertanggung jawab untuk logika presentasi

codingstudio

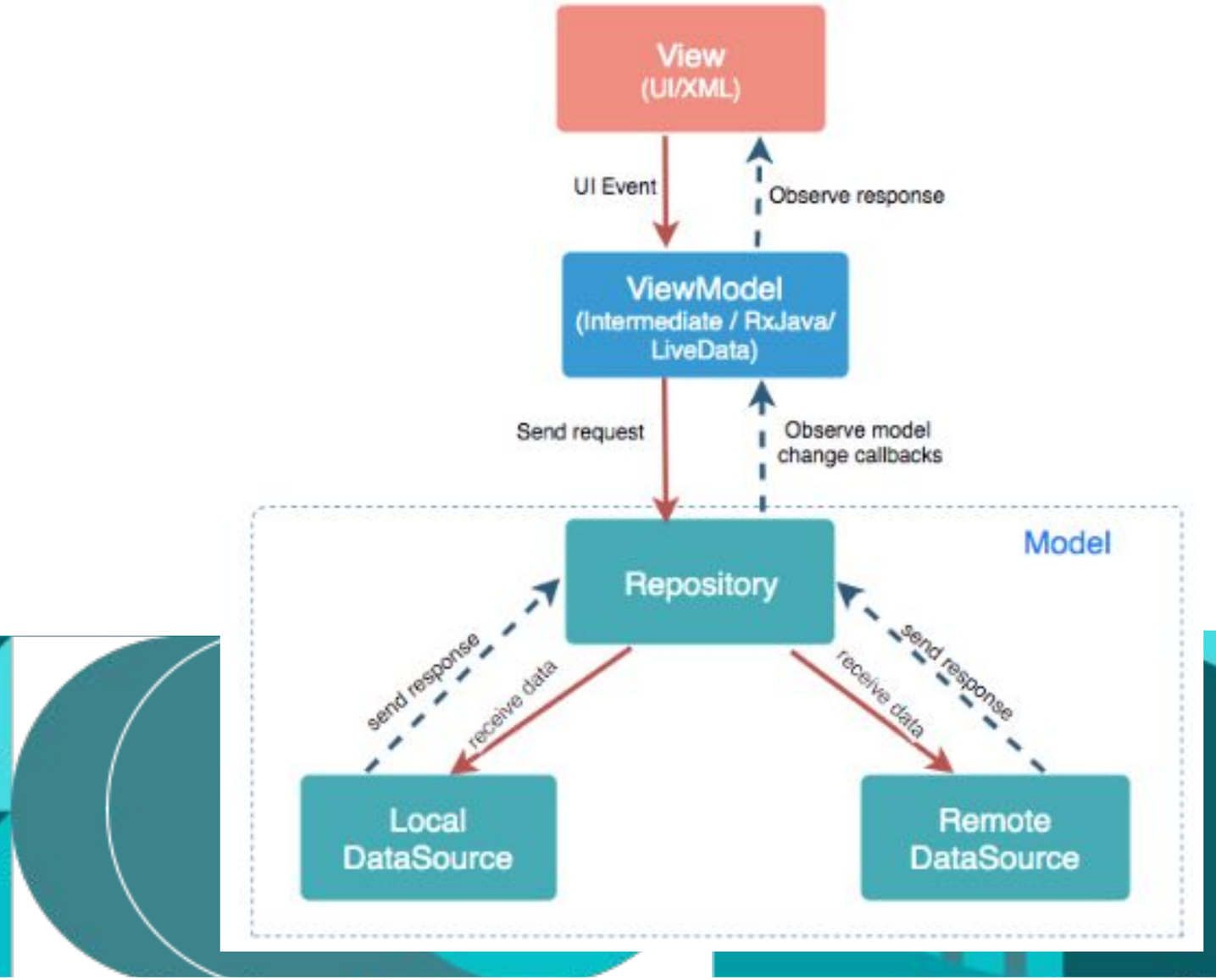
Cara kerja MVP (Model View Presenter)

- Pengguna berinteraksi dengan view
- View mendelegasikan Tindakan pengguna ke Presenter
- Presenter berinteraksi dengan model untuk mengambil atau memanipulasi data
- Model memberitahu Presenter tentang perubahan data
- Presenter menyiapkan data untuk ditampilkan dan memperbarui view



MVVM (MODEL VIEW VIEWMODEL)

codingstudio



codingstudio

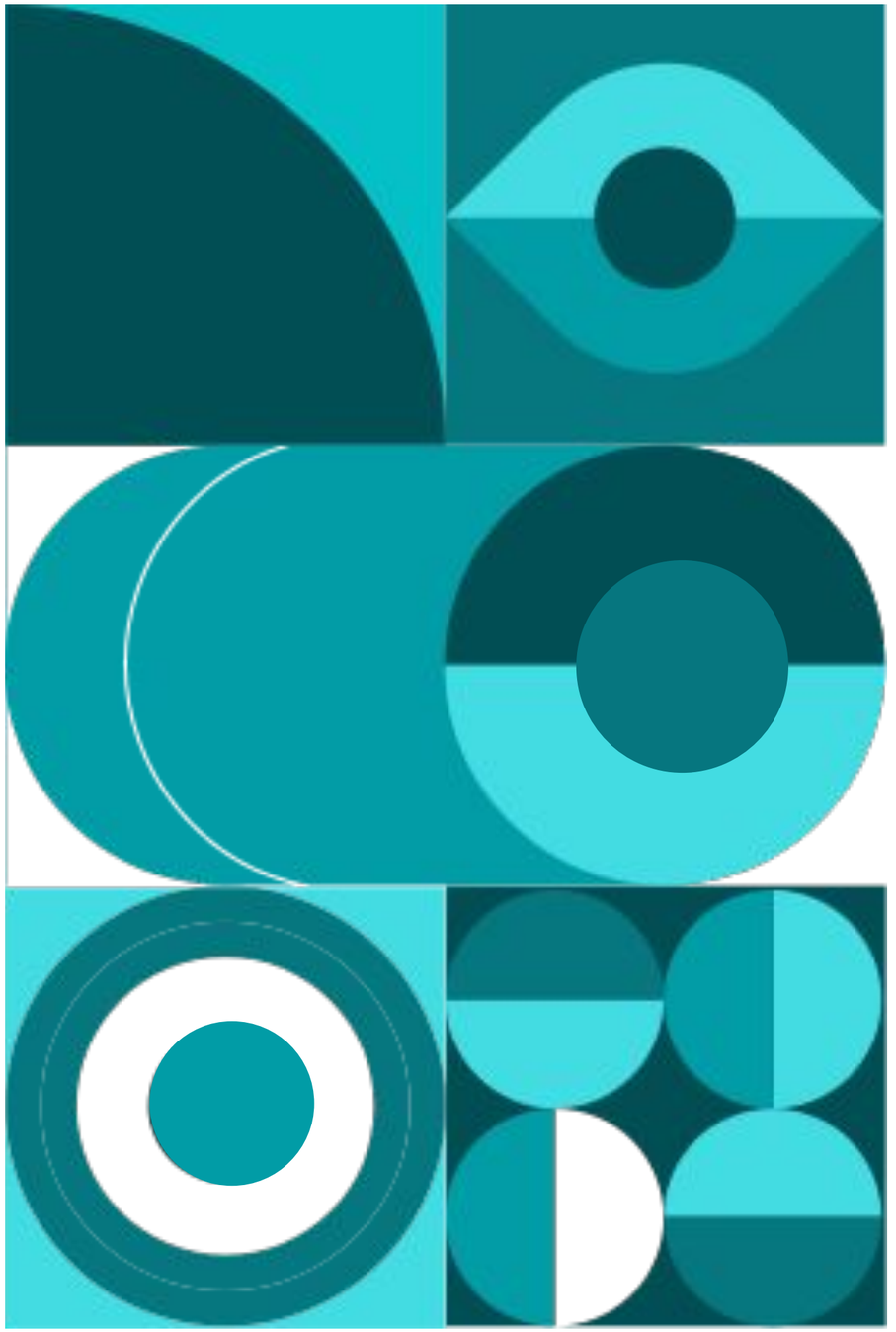
Tugas MVVM (Model View ViewModel)

- **Model** : Menentukan jenis data dan business logic
- **View** : Bertanggung jawab untuk menampilkan data kepada pengguna
- **ViewModel** : Menyediakan data yang akan ditampilkan oleh View dan perintah yang dapat dieksekusi oleh View

codingstudio

MVVM (Model View ViewModel)

- View terikat (blind) ke properti di ViewModel. Ketiak property di Viewmodel berubah, view secara otomatis diperbarui
- Pengguna berinteraksi dengan View (misalnya , mengetik di teksbox, mengklik tombol)
- Tindakan view terikat (bind) ke perintah (commands) di ViewModel
- View Model menerima perintah dan berinteraksi dengan model untuk memperbarui data
- Model memperbarui data dan memberi tahu ViewModel tentang perubahan.
- ViewModel memperbarui propertinya, dan View secara otomatis diperbarui melalui data binding.



codingstudio

ARSITEKTUR AND STRUCTURE PROJECT

codingstudio

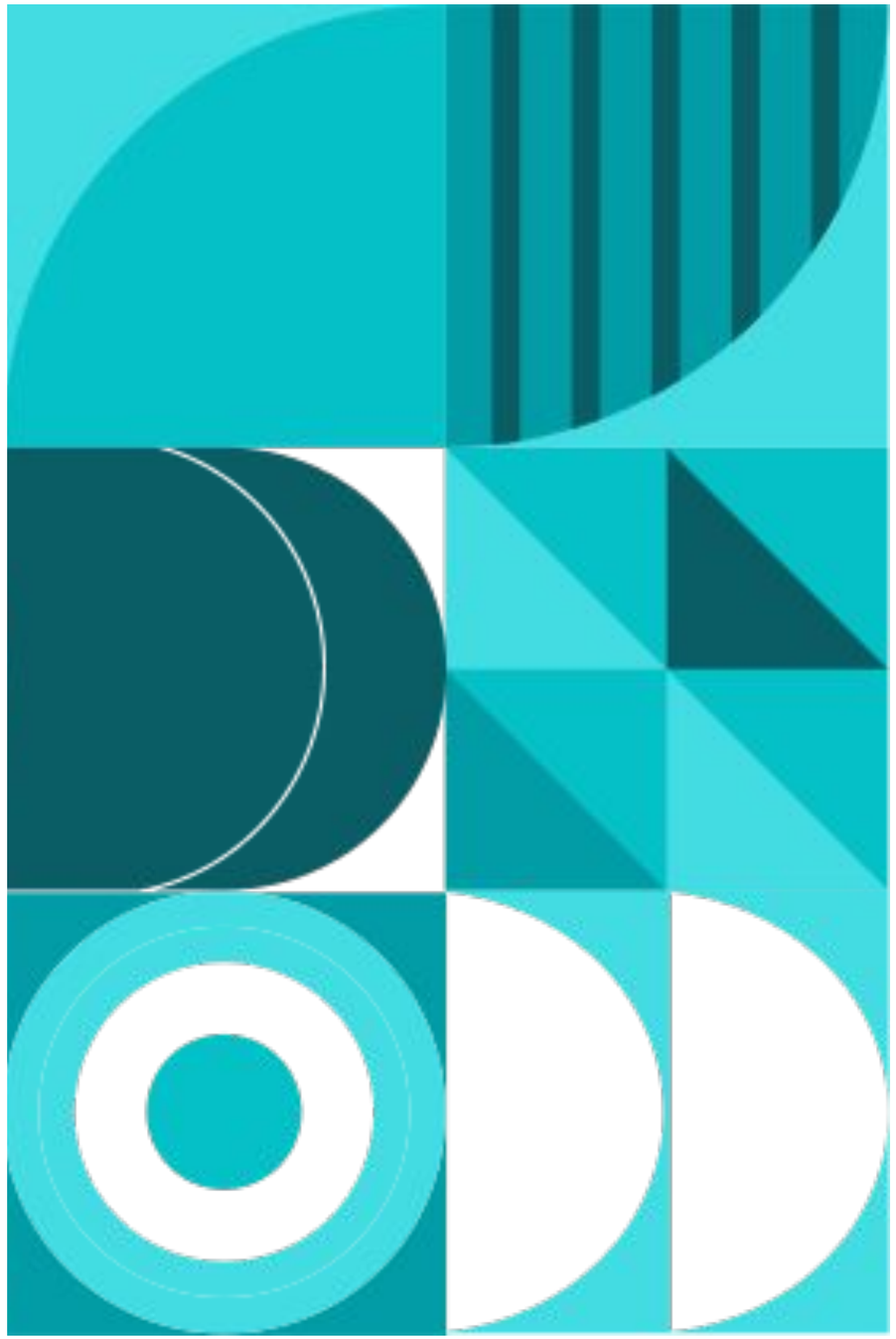
ARCHITECTURE APPLICATION

- Architecture Application adalah cetak biru atau struktur konseptual yang mendefinisikan komponen-komponen aplikasi , bagaimana komponen tersebut berinteraksi satu sama lain, dan prinsip yang memandu desain dan evolusi aplikasi dari waktu ke waktu
- yang bertujuan untuk Skalabilitas, pemeliharaan , keandalan , kinerja , dan kemampuan suatu aplikasi

codingstudio

FAKTOR YANG MEMPENGARUHI PEMILIHAN ARSITEKTUR

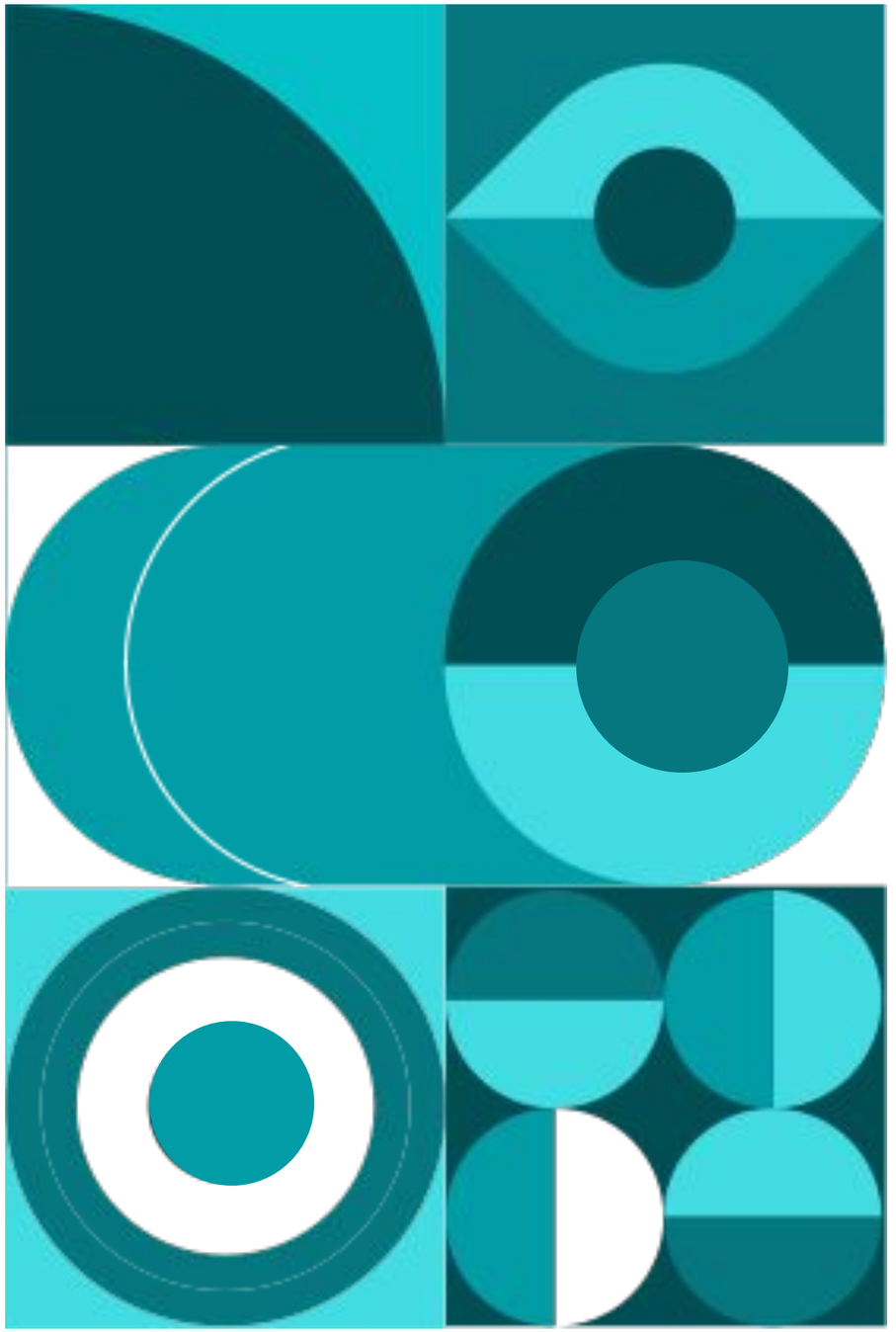
- Kebutuhan Bisnis
- Persyaratan Non-Functional
- Ukuran dan kompleksitas Aplikasi
- Tim Pengembang
- Anggaran dan Waktu
- Teknologi yang tersedia



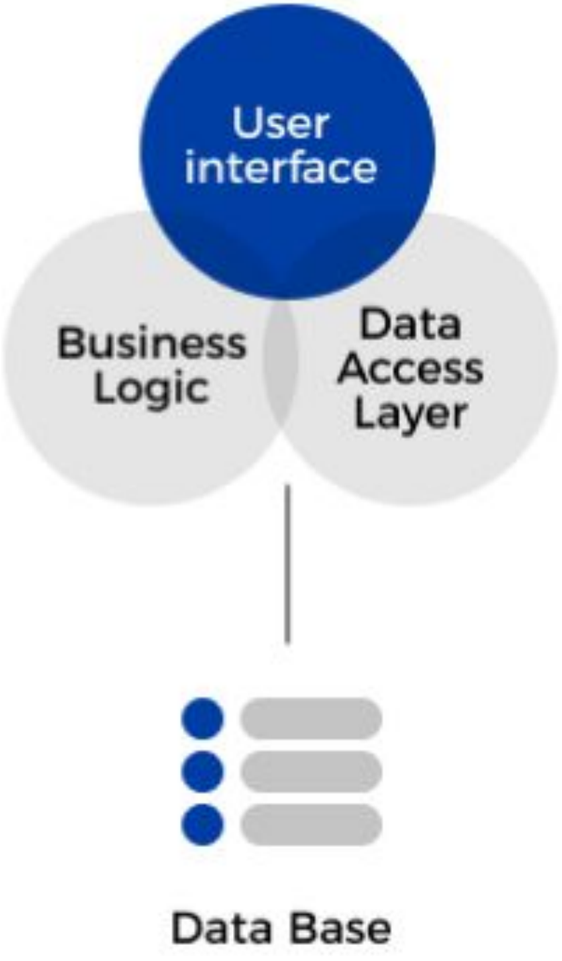
codingstudio

Pola Arsitektur Aplikasi

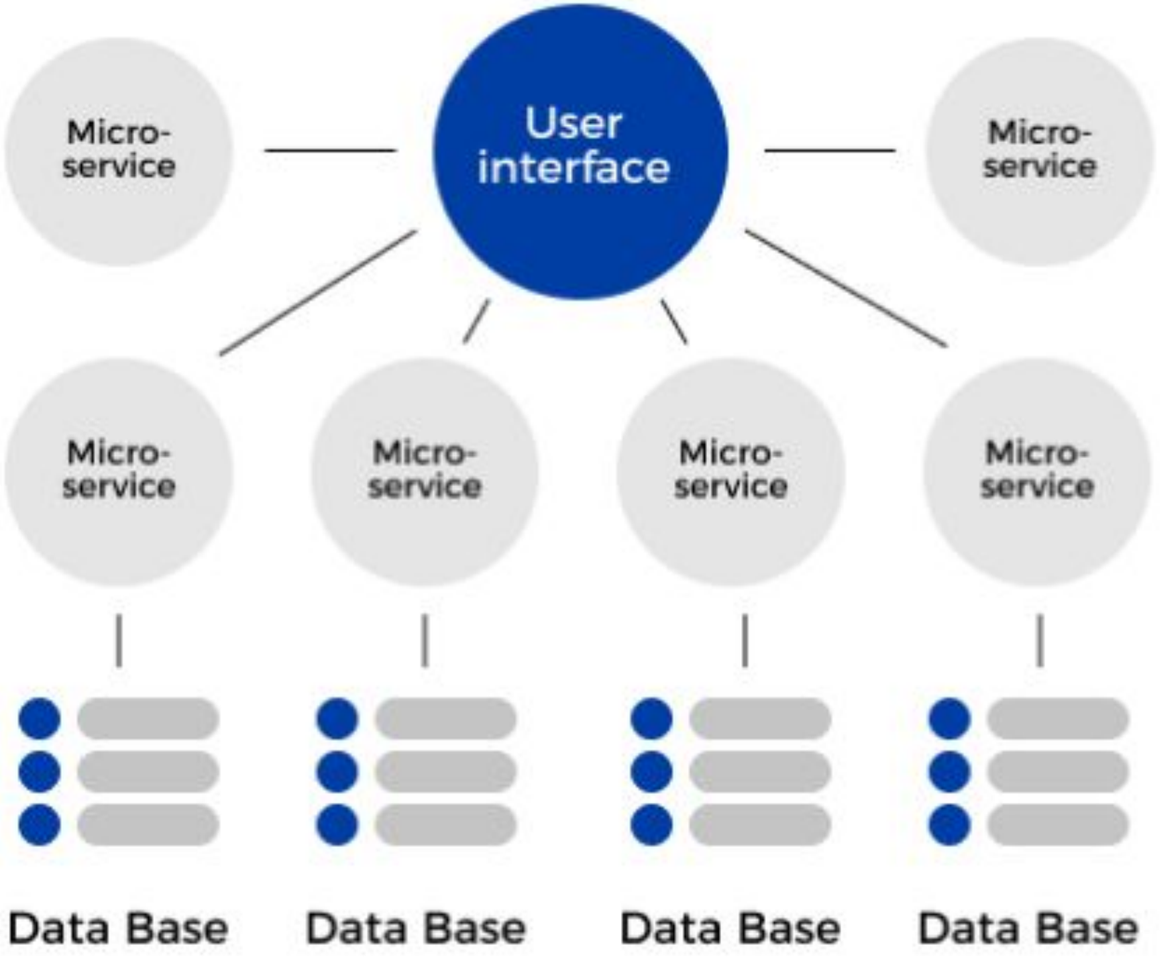
- Monolitik (Dibahas)
- Microservice (dibahas)
- Service Oriented Architecture SOA
- Arsitektur Berlapis
- Event Driven Architecture
- Client-server



MONOLITHIC ARCHITECTURE



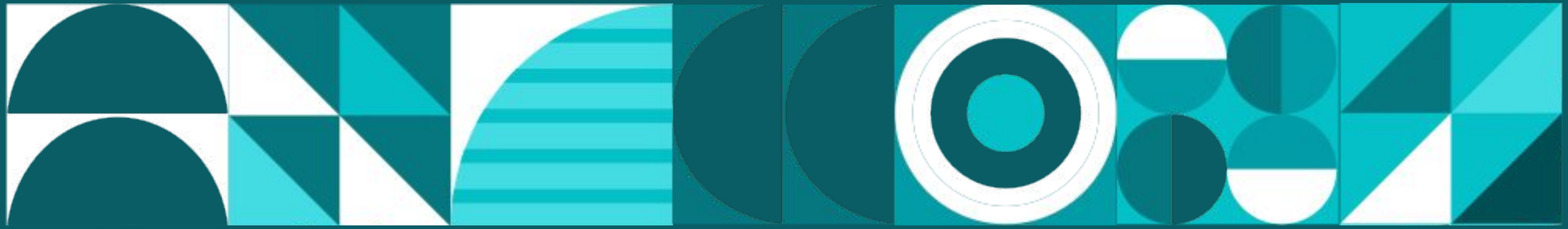
MICROSERVICE ARCHITECTURE

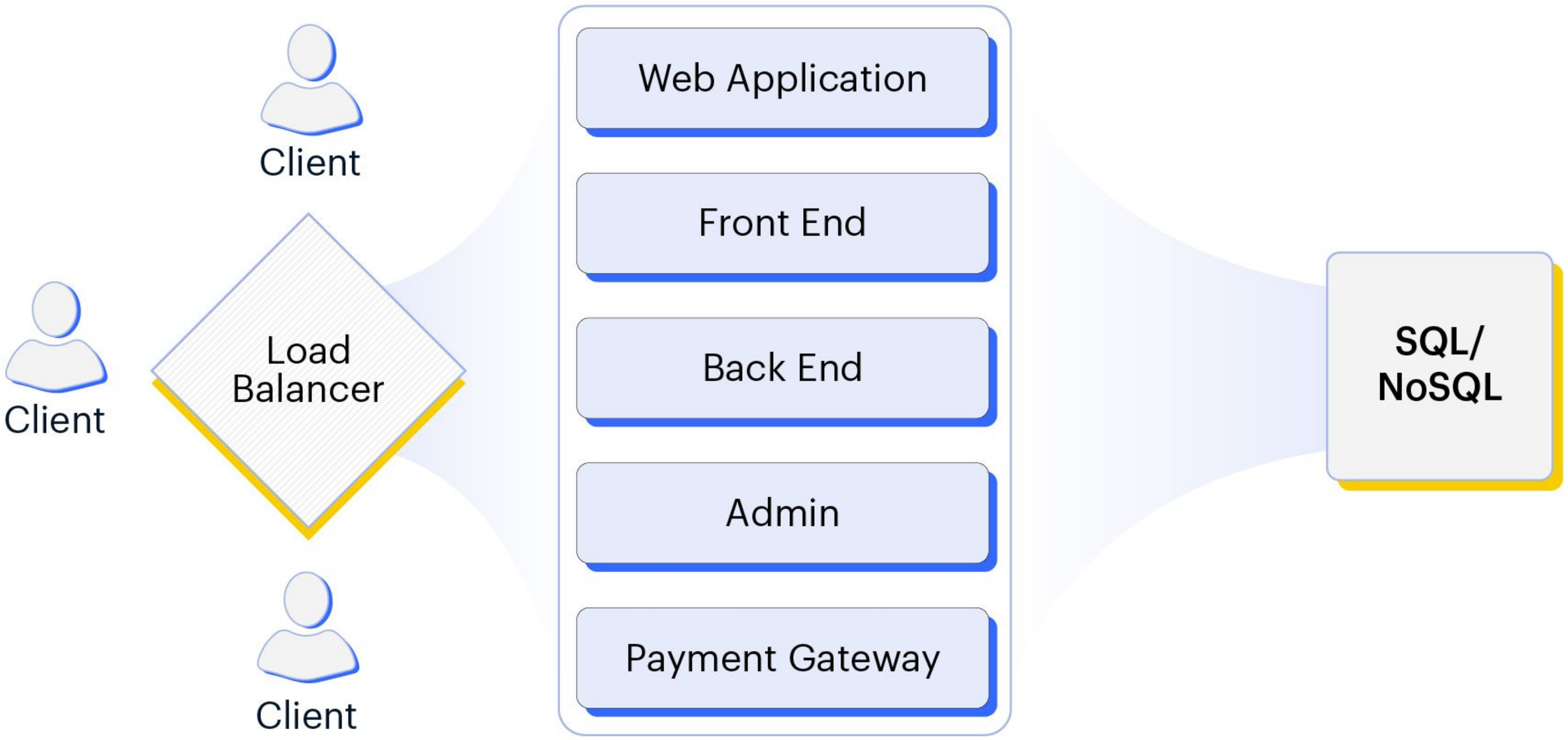


codingstudio

ARSITEKTUR MONOLITIK

Arsitektur Monolitik adalah pendekatan tradisional dimana seluruh komponen aplikasi mulai dari antarmuka pengguna (User Interface), logic bisnis (business logic), hingga lapisan akses data (data access layer) dibangun dan di deploy sebagai satu unit tunggal




Client
Client
Load Balancer
Client
Web Application
Front End
Back End
Admin
Payment Gateway
SQL/ NoSQL

codingstudio

Kelebihan dan Kekurangan Monolitik

Kelebihan

Sederhana untuk dimulai : lebih mudah dan cepat untuk dikembangkan, diuji dan di deploy pada tahap awal

Performa tinggi Komunikasi antar komponen sangat cepat karena terjadi di dalam satu proses yang sama

Manajemen lebih mudah karena hanya ada satu codebase atau basis kode yang perlu dikelola

Kelebihan dan Kekurangan Monolitik

codingstudio

Kekurangan

Sulit untuk dikembangkan skalanya Jika hanya satu fitur yang butuh sumber daya lebih, kamu harus mendeploy ulang dan men-scale keseluruhan aplikasi

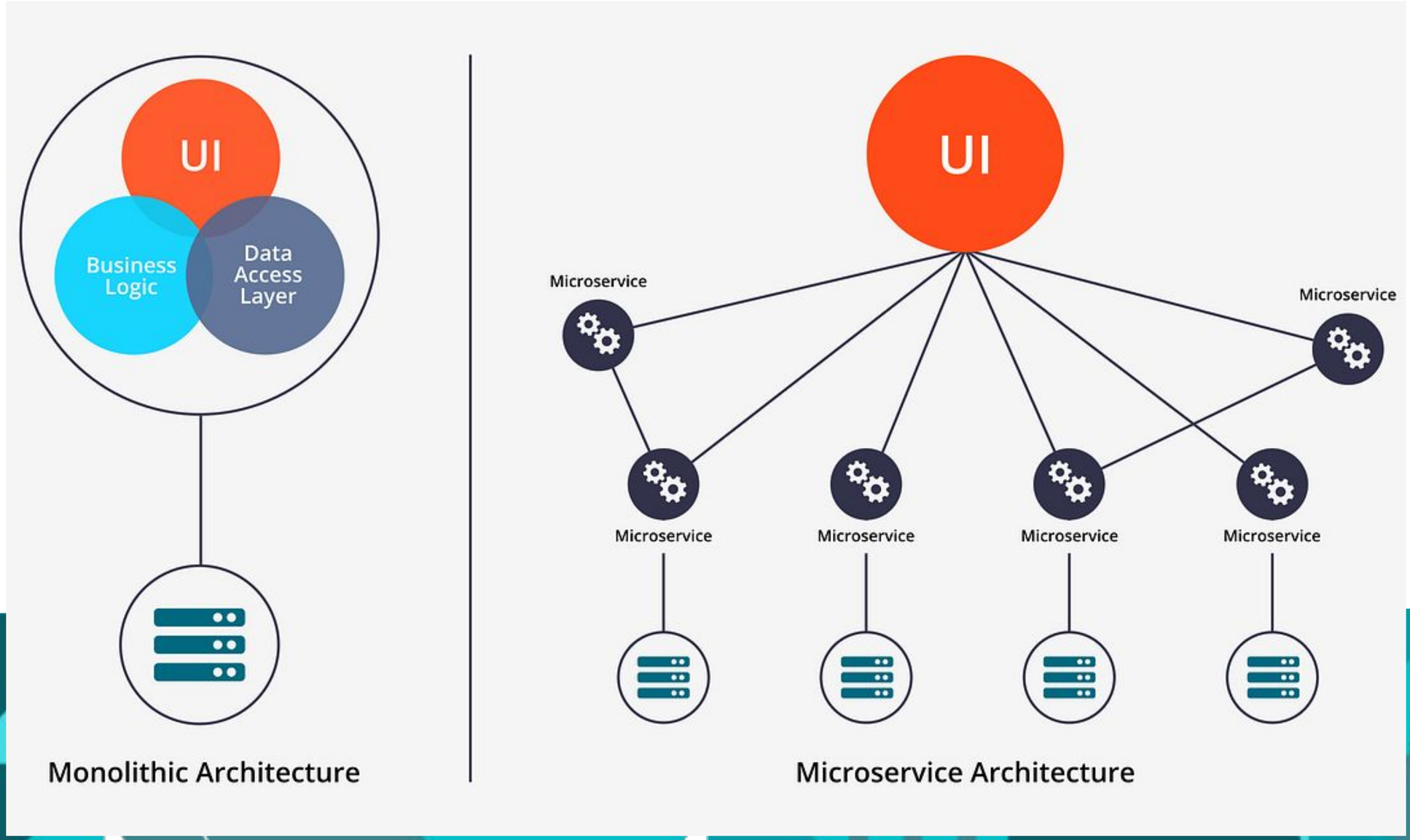
Teknologi Terkunci : sulit untuk mengadopsi bahasa pemrograman atau teknologi baru. Seluruh aplikasi terikat pada satu tumpukan teknologi

Rentan Gagal : jika satu komponen kecil mengalami eror, seluruh aplikasi berisiko ikut mati (down)

codingstudio

ARSITEKTUR MICROSERVICES

Arsitektur Microservices adalah pendekatan yang memecah aplikasi menjadi kumpulan layanan-layanan kecil yang independen. Setiap layanan memiliki fungsi bisnis spesifik, basis kodenya sendiri, dan dapat di-deploy secara terpisah


UI
Business Logic
Data Access Layer
Monolithic Architecture
UI
Microservice
Microservice
Microservice
Microservice
Microservice
Microservice
Microservice Architecture

codingstudio

Kelebihan dan Kekurangan Microservices

Kelebihan

Skalabilitas Fleksibel : kamu bisa men-scale hanya layanan yang membutuhkan,sehingga lebih efisien

Fleksibilitas Teknologi : Setiap layanan bisa menggunakan teknologi atau bahasa pemrograman yang paling sesuai untuk tugas

Lebih tahan banting : Jika satu layanan gagal, layanan lainnya tetap bisa berjalan.

Pengembangn oleh tim terpisah : tim kecil bisa fokus mengembgankan dan men-deploy layanan sendiri dengan cepat

codingstudio

Kelebihan dan Kekurangan Microservices

Kekurangan

Sangat kompleks lebih sulit untuk di deploy, di monitoring dan dikelola secara keseluruhan.

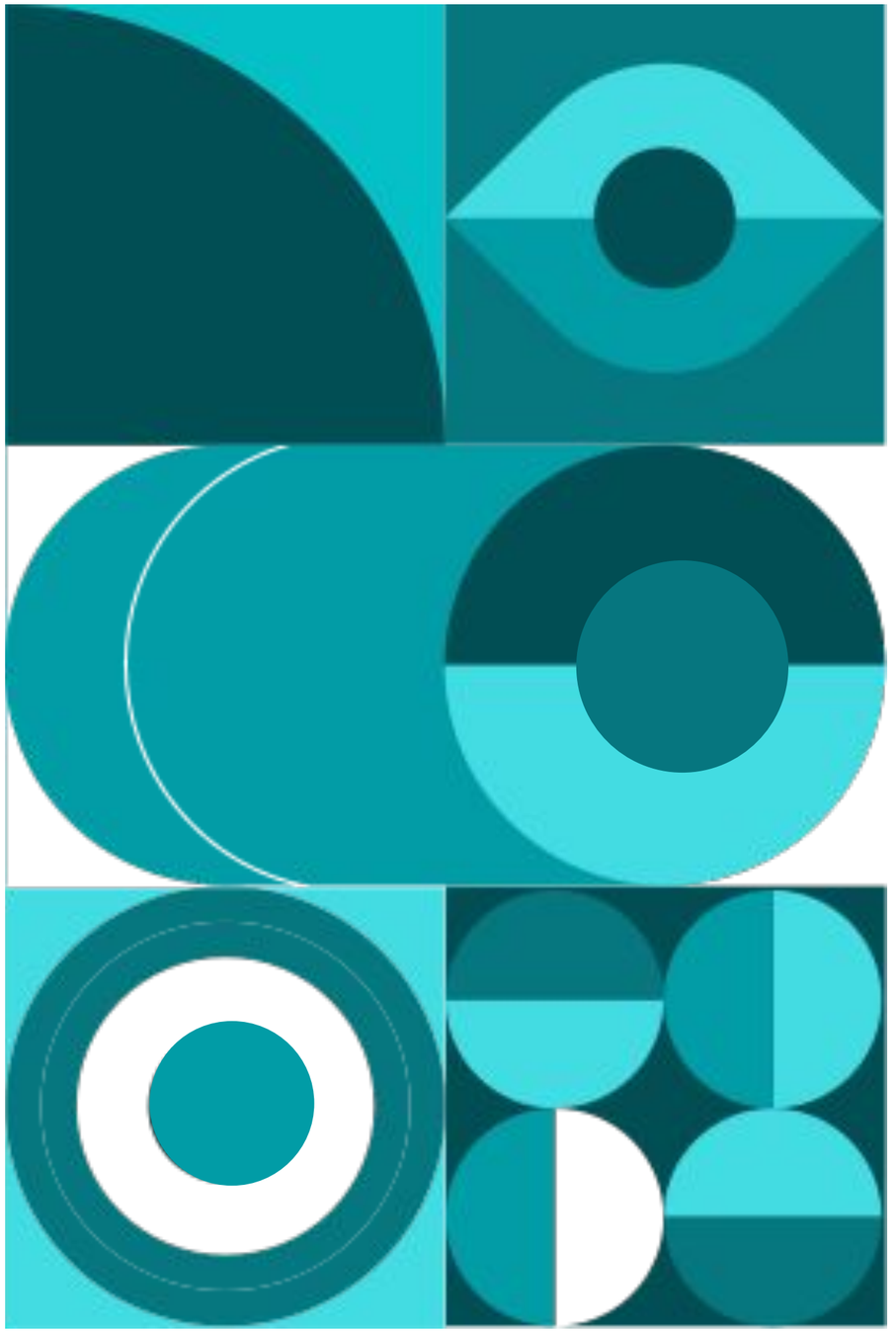
Tantangan Jaringan : komunikasi antar layanan melalui jaringan bisa lebih lambat dan menjadi sumber masalah baru.

Konsistensi Data : Menjaga agar data tetap konsisten di berbagai layanan yang memiliki database sendiri dan tantangan besar

codingstudio

HANDS ON

Implementasikan code yang sudah dibuat
sebelumnya dengan metode OOP
pecah menjadi konsep MVC saja yang paling dasar



THANK YOU

codingstudio