

MODUL PRAKTIKUM

APLIKASI BERBASIS PLATFORM

S1 INFORMATIKA



Published by School of Computing



LEMBAR PENGESAHAN

Saya yang bertanda tangan di bawah ini:

Nama : Shinta Yulia Puspitasari, S.T., M.T.
NIK : 18880124
Koordinator Mata Kuliah : Aplikasi Berbasis Platform
Prodi : S1 Informatika

Menerangkan dengan sesungguhnya bahwa modul ini digunakan untuk pelaksanaan praktikum di Semester Genap Tahun Ajaran 2021/2022 di Laboratorium Informatika Fakultas Informatika Universitas Telkom.

Bandung, Februari 2022



Shinta Yulia Puspitasari, S.T., M.T.

Dr. Erwin Budi Setiawan, S.Si., M.T.

Peraturan Praktikum Laboratorium Informatika 2021/2022

1. Praktikum diampu oleh dosen kelas dan dibantu oleh asisten laboratorium dan asisten praktikum.
2. Praktikum dilaksanakan di Gedung Kultubai Selatan (IFLAB 1 s/d IFLAB 5) dan Gedung Kultubai Utara (IFLAB 6 s/d IFLAB 7) sesuai jadwal yang ditentukan.
3. Praktikan wajib membawa modul praktikum, kartu praktikum, dan alat tulis.
4. Praktikan wajib mengisi daftar hadir *rooster* dan BAP praktikum dengan bolpoin bertinta hitam.
5. Durasi kegiatan praktikum S-1 = 1 SKS atau setara dengan 150 menit.
 - 15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
 - 45 menit untuk penyampaian materi
 - 90 menit untuk pengerjaan jurnal dan tes akhir
6. Jumlah pertemuan praktikum:
 - 14 kali di lab (praktikum rutin)
 - 2 kali di luar lab (terkait Tugas Besar atau COTS)
7. Praktikan wajib hadir minimal 75% dari seluruh pertemuan praktikum di lab. Jika total kehadiran kurang dari 75% maka nilai UAS/ Tugas Besar = 0.
8. Praktikan yang datang terlambat :
 - <= 30 menit : diperbolehkan mengikuti praktikum tanpa tambahan waktu Tes Awal
 - > 30 menit : tidak diperbolehkan mengikuti praktikum 9.
9. Saat praktikum berlangsung, asisten praktikum dan praktikan:
 - Wajib menggunakan seragam sesuai aturan institusi.
 - Wajib mematikan/ mengkondisikan semua alat komunikasi.
 - Dilarang membuka aplikasi yang tidak berhubungan dengan praktikum yang berlangsung.
 - Dilarang mengubah pengaturan *software* maupun *hardware* komputer tanpa ijin.
 - Dilarang membawa makanan maupun minuman di ruang praktikum.
 - Dilarang memberikan jawaban ke praktikan lain.
 - Dilarang menyebarkan soal praktikum.
 - Dilarang membuang sampah di ruangan praktikum.
 - Wajib meletakkan alas kaki dengan rapi pada tempat yang telah disediakan.
10. Setiap praktikan dapat mengikuti praktikum susulan maksimal dua modul untuk satu mata kuliah praktikum, dengan ketentuan:
 - Praktikan yang dapat mengikuti praktikum susulan hanyalah praktikan yang memenuhi syarat sesuai ketentuan institusi, yaitu: sakit (dibuktikan dengan surat keterangan medis), tugas dari institusi (dibuktikan dengan surat dinas atau dispensasi dari institusi), atau mendapat musibah atau kedukaan (menunjukkan surat keterangan dari orangtua/wali mahasiswa.)
 - Persyaratan untuk praktikum susulan diserahkan sesegera mungkin kepada asisten laboratorium untuk keperluan administrasi.
 - Praktikan yang diijinkan menjadi peserta praktikum susulan ditetapkan oleh Asman Lab dan Bengkel Informatika dan tidak dapat diganggu gugat.
11. Pelanggaran terhadap peraturan praktikum akan ditindak secara tegas secara berjenjang di lingkup Kelas, Laboratorium, Fakultas, hingga Universitas.

Tata cara Komplain Praktikum IFLab Melalui IGracias

1. Login IGracias
2. Pilih Menu **Masukan dan Komplain**, pilih **Input Tiket**



3. Pilih Fakultas/Bagian: **Bidang Akademik (FIF)**
4. Pilih Program Studi/Urusan: **Urusan Laboratorium/Bengkel/Studio (FIF)**
5. Pilih Layanan: **Praktikum**
6. Pilih Kategori: **Pelaksanaan Praktikum**, lalu pilih **Sub Kategori**.
7. Isi **Deskripsi** sesuai komplain yang ingin disampaikan.

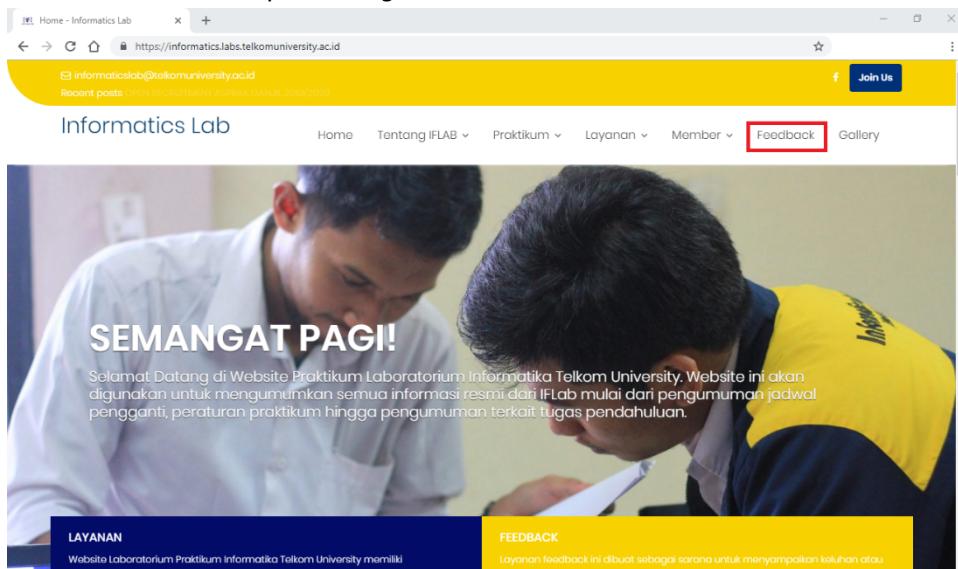
A screenshot of a web form titled 'Input Keluhan'. The form fields are:

- Fakultas / Bagian : BIDANG AKADEMIK (FIF)
- Program Studi / Urusan : URUSAN LABORATORIUM/BENGKEL/STUDIO (FIF)
- Pelapor : RIZQILLAH ZAHRA LESTARI
- Layanan : PRAKTIKUM
- Kategori : Pelaksanaan Praktikum
- Sub Kategori : Please Select...
- Tipe Masukan : Komplain Masukan
- Deskripsi : (Rich Text Editor area)

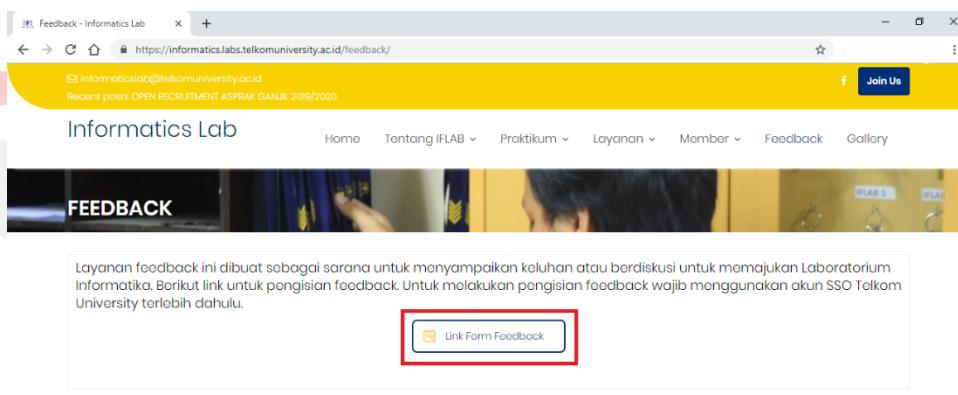
8. Lampirkan file jika perlu. Lalu klik Kirim.

Tata Cara Komplain Praktikum IFLAB Melalui Website

1. Buka website <https://informatics.labs.telkomuniversity.ac.id/> melalui browser.
2. Pilih menu **Feedback** pada navigation bar website.



1. Pilih tombol **Link Form Feedback**.



1. Lakukan *login* menggunakan akun **SSO Telkom University** untuk mengakses *form feedback*.
2. Isi *form* sesuai dengan *feedback* yang ingin diberikan.

DAFTAR ISI

LEMBAR PENGESAHAN	1
Peraturan Praktikum Laboratorium Informatika 2021/2022	2
Tata cara Komplain Praktikum IFLab Melalui iGracias	3
Tata Cara Komplain Praktikum IFLAB Melalui Website	4
DAFTAR ISI	5
DAFTAR GAMBAR	10
DAFTAR TABEL	13
MODUL 1. RUNNING MODUL	1
1.1. Git	1
1.2. Instalasi Apache, PHP, dan MySQL dengan XAMPP	3
1.3. Instalasi Composer dan Laravel	6
1.4. Instalasi Visual Studio Code	8
1.5. Instalasi JDK	9
1.6. Instalasi Flutter SDK	13
1.7. Instalasi Android Studio	18
1.8. Instalasi SDK Android	24
MODUL 2. HTML & GIT	27
2.1. Pengenalan HTML	27
2.1.1. Tag HTML	27
2.1.2. Elemen HTML	27
2.1.3. Atribut HTML	27
2.2. Dasar Sintaks HTML	28
2.3. Heading	28
2.4. Hyperlink	29
2.5. Tabel	29
2.6. Image	30
2.7. Audio / Video Elemen	31
2.8. Form	32
2.8.1. Elemen Form	32
2.8.2. Atribut Elemen Form	32
2.9. Penggunaan Git	35
2.9.1. Membuat repositori baru	35
2.9.2. Menambahkan isi repositori	35
2.9.3. Membuat repositori online	36
2.9.4. Menyimpan hasil pekerjaan di repositori online.	37
2.9.5. Clone repositori milik orang lain.	38

MODUL 3. CSS – Cascading Style Sheet	39
3.1. Pengenalan CSS	39
3.1.1. Cara Menyisipkan CSS	39
3.1.2. Selector	40
3.2. Font Properties	40
3.3. List Properties	40
3.4. Alignment of Text	42
3.5. Colors	43
3.6. Span & Div	43
3.7. Bootstrap	44
3.7.1. Pemasangan Bootstrap	44
3.7.2. Bootstrap Container	44
3.7.3. Bootstrap Grid	45
3.7.4. Text Style	46
3.7.5. Bootstrap Table, Image, & Button	46
3.7.6. Bootstrap Form	49
MODUL 4. JAVASCRIPT	51
4.1. Pengenalan Javascript	51
4.1.1. Sejarah Singkat Javascript	51
4.1.2. Prinsip Dasar Javascript	51
4.2. Sintaks Umum pada Javascript	51
4.2.1. Tipe data dasar	51
4.2.2. Variabel	52
4.2.3. Array	52
4.2.4. Pengendalian Struktur	52
4.3. Object Orientation pada Javascript	53
4.3.1 Pembuatan Object pada Javascript	53
4.3.2. Akses Nilai Property	54
4.3.3. Prototype pada Javascript	54
4.4. Function pada Javascript	54
4.4.1 Pembuatan Fungsi pada Javascript	54
4.4.2. Pemanggilan Fungsi	55
4.5. DOM Manipulation	56
4.6. jQuery	57
4.6.1. Efek hide/show	58
4.6.2. Efek animasi	58
4.6.3. DOM Manipulation	59
MODUL 5. PHP	61
5.1. Web Server dan Server Side Scripting	61
5.2. Pengenalan PHP	62
5.3. Variabel	62

5.4. Konstanta	63
5.5. Operator dalam PHP	63
5.6. Struktur Kondisi	64
5.7. Perulangan (Looping)	65
5.8. Function	66
5.9. Array	67
5.10. GET dan POST	68
5.10.1. Metode GET	68
5.10.2. Metode POST	69
5.11. XML	70
5.11.1. Pengertian XML	70
5.11.2. Sintaks XML	70
5.12. JSON	71
5.12.1. Pengertian JSON	71
5.12.2. Perbedaan JSON dan XML	72
5.13. AJAX	73
5.13.1. Apa Itu AJAX	73
5.13.2. Cara Kerja AJAX	73
5.13.3. Event Handling	73
MODUL 6. LARAVEL	77
6.1. Framework & MVC	77
6.2. Pengenalan Laravel	78
6.3. Cara Kerja Laravel	79
6.3.1. Routing	80
6.3.2. View	80
6.3.3. Controller	81
MODUL 7. LARAVEL LANJUT	84
7.1. CRUD	84
7.1.1. Konfigurasi dan Skema	84
7.1.2. Model	85
7.1.3. Controller	85
7.1.4. View	87
7.1.5. Tampilan Halaman	88
7.2. Templating Halaman	90
7.3. Form Validation	91
7.4. Session	92
7.5. Middleware	93
7.6 Model Relasi	94
MODUL 8. PENGENALAN DART	96
8.1. Pengenalan Dart	96
8.1.1. Variable	96

8.1.2. Statement Control	96
8.1.3. Looping	98
8.1.4. List	98
8.1.5. Fungsi	99
MODUL 9. FLUTTER LAYOUT DASAR	101
9.1. Pengenalan Widget	101
9.2. Container	101
9.3. GridView	102
9.4. ListView	104
9.4.1. ListView.builder	105
9.4.2. ListView.separated	105
9.5. Stack	106
MODUL 10. FLUTTER LAYOUT LANJUT	108
10.1. Row	108
10.2. Column	109
10.3. Nested Rows & Columns	111
10.4. CustomScrollView	116
MODUL 11. PACKAGES & USER INTERACTION	118
11.1. Packages	118
11.1.1. Pengenalan Packages	118
11.1.2. Penggunaan Packages	118
11.2. User Interaction	119
11.2.1. Stateful & Stateless	119
11.2.2. Form	119
11.2.3. Menu	120
11.2.3.1. Tab Bar	120
11.2.3.2. Bottom Navigation Bar	122
11.2.4. Buttons	124
11.2.4.1. ElevatedButton	124
11.2.4.2. TextButton	124
11.2.4.3. DropdownButton	125
MODUL 12. MODEL, NAVIGATION & NOTIFICATION	126
12.1. Model	126
12.1.1 Pengenalan Model	126
12.1.2 Membuat Model Class	126
12.2. Navigation	127
12.2.1 Navigation Pindah Halaman	127
12.2.2 Navigation Mengirim Data	127
12.3. Notification	128
MODUL 13. STATE MANAGEMENT & NETWORKING	133

13.1. State Management	133
13.1.1. Pengenalan State Management	133
13.1.2. State Management Provider	133
13.2. Networking	135
13.2.1. Fetch & Delete Data	136
13.2.2. Send & Update	137
13.2.3. Parse JSON	137
MODUL 14. MAPS, PLACES, CAMERA & MEDIA	138
14.1. Google Maps API	138
14.1.1. Menambahkan Packages Google Maps	140
14.1.2. Menambahkan Akses Lokasi Kita Pada Manifest	142
14.1.3. Menambahkan Marker pada Google Maps	142
14.2. Place Picker	143
14.3. Camera API	144
14.4. Media API	146
DAFTAR PUSTAKA	149



DAFTAR GAMBAR

Gambar 1.1 Instalasi Git	1
Gambar 1.2 Opsi instalasi pada git	1
Gambar 1.3 Pilihan editor	2
Gambar 1.4 Pilihan opsi environment	2
Gambar 1.5 Fitur opsional pada Git	2
Gambar 1.6 Mengecek versi Git	3
Gambar 1.7 Peringatan antivirus pada instalasi XAMPP	3
Gambar 1.8 Peringatan UAC pada instalasi XAMPP	3
Gambar 1.9 Jendela awal instalasi XAMPP	4
Gambar 1.10 Jendela pemilihan komponen pada XAMPP	4
Gambar 1.11 Jendela pemilihan lokasi instalasi	4
Gambar 1.12 Jendela instalasi Bitnami pada XAMPP	5
Gambar 1.13 Jendela konfirmasi instalasi	5
Gambar 1.14 Jendela proses instalasi	5
Gambar 1.15 Jendela instalasi XAMPP selesai	6
Gambar 1.16 Jendela ketika service pada XAMPP dijalankan	6
Gambar 1.17 Tampilan ketika localhost dijalankan pada <i>browser</i>	6
Gambar 1.18 Instalasi Composer	7
Gambar 1.19 Setting Path PHP	7
Gambar 1.20 Instalasi Laravel	7
Gambar 1.21 Menjalankan Laravel	8
Gambar 1.22 Welcome Page Laravel	8
Gambar 1.23 Tampilan download visual studio code	8
Gambar 1.24 Jendela setup visual studio code	9
Gambar 1.25 Tampilan download jdk	9
Gambar 1.26 Jendela awal instalasi JDK	10
Gambar 1.27 Jendela folder path pada instalasi JDK	10
Gambar 1.28 Jendela tunggu pada instalasi JDK	10
Gambar 1.29 Jendela finish instalasi pada JDK	11
Gambar 1.30 Proses mencopy path jdk dari folder instalasinya	11
Gambar 1.31 Melakukan search environment variables	11
Gambar 1.32 Jendela system properties	12
Gambar 1.33 Jendela Environment Variables	12
Gambar 1.34 Jendela New System Variable	12
Gambar 1.35 Jendela New System Variabel	13
Gambar 1.36 Jendela Environment Variables telah selesai diakses	13
Gambar 1.37 Menemukan berkas flutter_console.bat	14
Gambar 1.38 Tampilan flutter_console.bat	15
Gambar 1.39 Jendela system properties	15
Gambar 1.40 Jendela Environment Variables	16
Gambar 1.41 Jendela Edit User Variable	16
Gambar 1.42 Jendela Edit Environment Variable	17
Gambar 1.43 Jendela awal instalasi Android Studio	18
Gambar 1.44 Jendela Choose Components pada Instalasi Android Studio	19
Gambar 1.45 Jendela Configuration Settings pada Instalasi Android Studio	19
Gambar 1.46 Jendela Choose Start Menu Folder pada Android Studio	19
Gambar 1.47 Jendela proses instalasi pada Android Studio	20
Gambar 1.48 Proses Instalasi dan Jendela finish pada Android Studio	20
Gambar 1.49 Jendela menu installation folder	20

Gambar 1.50 Jendela proses tunggu masuk Android Studio	21
Gambar 1.51 Jendela Installation type	21
Gambar 1.52 Jendela Select UI Theme	21
Gambar 1.53 Jendela Verify Settings	22
Gambar 1.54 Jendela Downloading Components	22
Gambar 1.55 Jendela tampilan pertama Android Studio	22
Gambar 1.56 Memilih New Project	23
Gambar 1.57 Jendela Select a Project Template	23
Gambar 1.58 Jendela Configure Your Project	23
Gambar 1.59 Selesai mengatur untuk tampilan awal project	24
Gambar 1.60 Memilih SDK Manajer di Android Studio	24
Gambar 1.61 Tampilan SDK Tools	25
Gambar 1.62 Memilih tools yang dibutuhkan pada SDK Tools	25
Gambar 1.63 Notifikasi untuk keterangan perubahan	25
Gambar 1.64 Jendela Licence Agreement untuk download SDK Tools	26
Gambar 1.65 Jendela finish instalasi SDK Android	26
Gambar 2.1 Struktur dasar HTML	21
Gambar 2.2 Tingkatan heading	28
Gambar 2.3 Tabel pada HTML	29
Gambar 2.4 Penggunaan colspan dan rowspan pada tabel HTML	30
Gambar 2.5 Penggunaan image pada HTML	31
Gambar 2.6 Penggunaan audio dan video pada HTML	31
Gambar 2.7 Contoh form pendaftaran	35
Gambar 2.8 Inisialisasi repositori git	35
Gambar 2.9 Langkah membuat file	35
Gambar 2.10 Hasil file yang telah dibuat	36
Gambar 2.11 Hasil perintah dari git status	36
Gambar 2.12 File telah ditambahkan	36
Gambar 2.13 Setelah commit dan melihat status	36
Gambar 2.14 Tampilan membuat repositori pada Github	37
Gambar 2.15 Tampilan setelah repositori dibuat	37
Gambar 2.16 Tampilan repositori setelah di-push pada Github	38
Gambar 2.17 Tombol clone terlihat pada kotak merah	38
Gambar 2.18 URL repositori target yang akan di-clone	38
Gambar 3.1. Penulisan CSS	39
Gambar 3.2 Penerapan font properties	40
Gambar 3.3 Tampilan penggunaan list properties	41
Gambar 3.4 List properties dengan tambahan CSS	42
Gambar 3.5 Penerapan alignment of text	42
Gambar 3.6 Penerapan Span & Div	43
Gambar 3.7 Class container	45
Gambar 3.8 Hasil penerapan Bootstrap grid	46
Gambar 3.9 Hasil penerapan Class.table-hover	47
Gambar 3.10 Hasil penerapan Class.thead-dark	47
Gambar 3.11 Image class.thumbnail dan class.fluid	48
Gambar 3.12 Hasil penerapan class button	49
Gambar 3.13 Contoh Bootstrap form	50
Gambar 4.1 Struktur DOM	56
Gambar 5.1 Arsitektur web standar	61
Gambar 5.2. Arsitektur web dinamis	61
Gambar 5.3 Tampilan ketika menjalankan file PHP pada browser	62



Gambar 5.4 Hasil penggunaan IF-THEN	64
Gambar 5.5 Hasil penggunaan switch-case	65
Gambar 5.6 Hasil penggunaan perulangan	66
Gambar 5.7 Output URL dari Metode GET	68
Gambar 5.8 Output Data dari Metode GET	69
Gambar 5.9 Output dengan Metode POST	70
Gambar 5.10 Deklarasi XML	71
Gambar 5.11 Cara kerja AJAX	73
Gambar 5.12 Tampilan control panel XAMPP	74
Gambar 5.13 Folder penyimpanan file AJAX	74
Gambar 5.14 File ajax_info pada folder penyimpanan file AJAX	75
Gambar 5.15 Tampilan file AJAX ketika dieksekusi	75
Gambar 5.16 Tampilan file AJAX-2	75
Gambar 5.17 Parameter pada salah satu fungsi AJAX	76
Gambar 6.1 Cara kerja MVC	78
Gambar 6.2 Alur kerja Laravel	79
Gambar 6.3 Struktur Folder Laravel	79
Gambar 7.1 Tampilan halaman view	88
Gambar 7.2 Tampilan halaman form tambah produk	89
Gambar 7.3 Tampilan halaman view setelah tambah data	89
Gambar 7.4 Tampilan halaman form ubah data	89
Gambar 9.1 Tampilan Container	102
Gambar 9.2 Tampilan Penerapan GridView	103
Gambar 9.3 Tampilan Penerapan ListView	104
Gambar 9.4 Tampilan Penerapan ListView.Separated	106
Gambar 9.5 Tampilan Penerapan Stack	106
Gambar 9.6 Tampilan Background Gradient	107
Gambar 10.1 Tampilan Overflowed di Flutter	108
Gambar 10.2 Tampilan Penggunaan Expanded	109
Gambar 10.3 Tampilan Penerapan Column	110
Gambar 10.4 Tampilan Penerapan mainAxisAlignment dan mainAxisSize	111
Gambar 10.5 Tampilan Penerapan Nested Rows dan Column	111
Gambar 10.6 Tampilan Penerapan Nested Rows dan Column 2	112
Gambar 10.7 Tampilan Penerapan Nested Rows dan Column 3	112
Gambar 10.8 Widget Tree	113
Gambar 10.9 Widget Tree dengan 3 Column	114
Gambar 11.1 Google Fonts	118
Gambar 11.2 Yaml file	118
Gambar 11.3 Dependencies	118
Gambar 12.1 Model pada flutter	126
Gambar 13.1 State Management Provider	133
Gambar 14.1 google_maps_flutter	140
Gambar 14.2 Demo Gmaps pada Flutter	142
Gambar 14.3 Demo Pencarian pada Gmaps	144
Gambar 14.4 Membuka file .plist	146



DAFTAR TABEL

Tabel 2.1 Elemen form	32
Tabel 2.2 Atribut elemen form	32
Tabel 3.1 Tabel font properties	40
Tabel 3.2 CSS properties untuk elemen list	41
Tabel 3.3 Alignment of text	42
Tabel 3.4 CSS Colors	43
Tabel 3.5 Sistem grid Bootstrap	45
Tabel 3.6 Text style	46
Tabel 3.7 Elemen tabel	46
Tabel 3.8 Class button	48
Tabel 4.1 Daftar sintaks DOM	56
Tabel 4.2 Daftar Fungsi DOM Manipulation pada JQuery	59
Tabel 4.3 Contoh DOM Manipulation	59
Tabel 5.1 Operator pada PHP	63
Tabel 6.1 URL yang ditangani otomatis	82



MODUL 1. RUNNING MODUL

1.1. Git

Git adalah salah satu sistem pengontrol versi (*Version Control System*) pada proyek perangkat lunak yang diciptakan oleh Linus Torvalds. Pengontrol versi bertugas mencatat setiap perubahan pada *file* proyek yang dikerjakan oleh banyak orang maupun sendiri. Git dikenal juga dengan *distributed revision control* (VCS terdistribusi), artinya penyimpanan *database* Git tidak hanya berada dalam satu tempat saja.

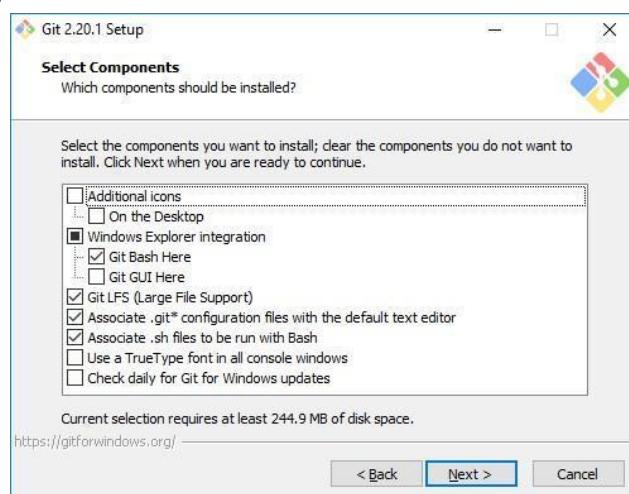
Untuk melakukan instalasi git pada computer Anda, lakukan langkah berikut ini:

1. Buka link berikut ini untuk mengunduh Git. <https://git-scm.com/download/win>
2. Klik dua kali pada file yang sudah diunduh.
3. Maka akan muncul informasi lisensi Git, klik *next* untuk melanjutkan.



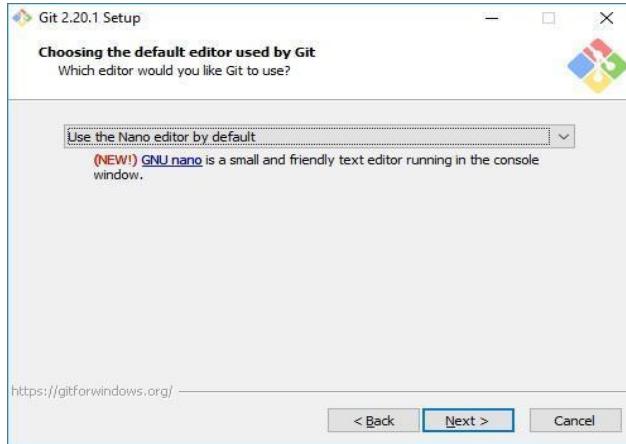
Gambar 1.1 Instalasi Git

4. Pada bagian ini, Anda dapat memilih komponen apa saja yang akan dipasang, jika sudah klik *next* untuk melanjutkan.



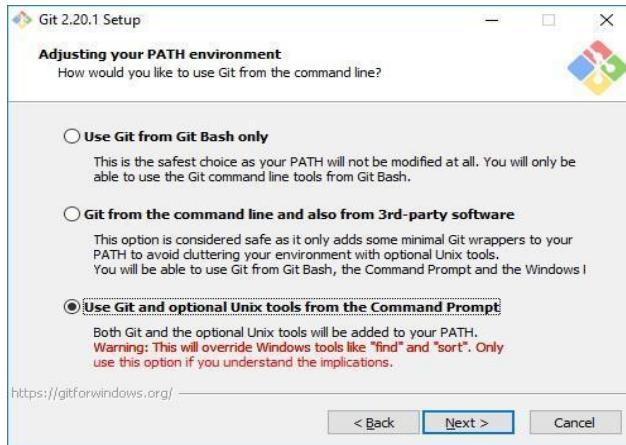
Gambar 1.2 Opsi instalasi pada git

5. Pilih editor yang akan digunakan secara *default* oleh Git. Gunakan Nano jika ingin editor yang lebih simpel untuk digunakan, atau Vim jika memang Anda menguasainya.



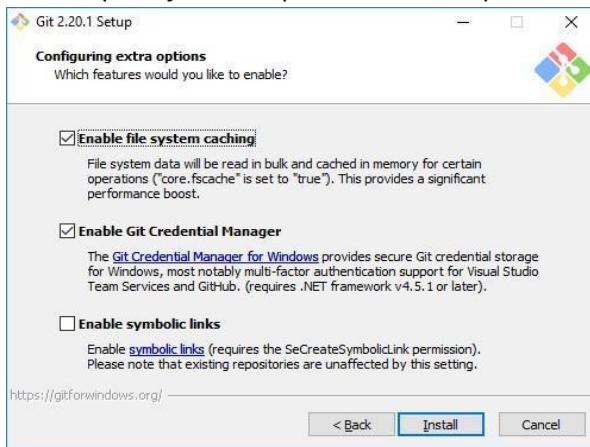
Gambar 1.3 Pilihan editor

- Pilih opsi ketiga agar *Command Prompt* dapat mengenali Git dan beberapa perintah UNIX lainnya.



Gambar 1.4 Pilihan opsi environment

- Untuk selanjutnya, gunakan opsi *default* sampai Anda berada pada tahap *install*. Lalu klik *install*.



Gambar 1.5 Fitur opsional pada Git

- Pastikan Git sudah terinstall dengan melakukan perintah `git --version` pada *command prompt*.

```
C:\>git --version  
git version 2.20.1.windows.1
```

Gambar 1.6 Mengecek versi Git

9. Lakukan konfigurasi awal dengan melakukan perintah

```
git config --global user.name "Nama Anda" git config --global user.email  
email.anda@contoh.com
```

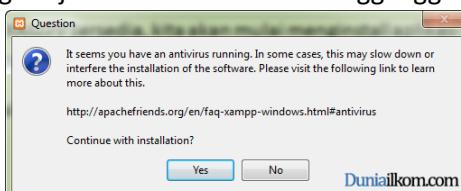
1.2. Instalasi Apache, PHP, dan MySQL dengan XAMPP

Proses instalasi Apache, PHP dan MySQL terkadang menjadi kendala tersendiri bagi yang ingin memulai belajar pemrograman *web*. Namun saat ini sudah tersedia banyak aplikasi paket yang mencakup ketiga aplikasi tersebut. Beberapa diantaranya adalah sebagai berikut:

1. XAMPP (versi windows) dan LAMPP (versi linux) yang dapat diunduh di <https://www.apachefriends.org/download.html>
2. WAMP Server
3. APPServ
4. PHPTriad

Pada modul ini, akan digunakan aplikasi paket XAMPP sebagai sarana pendukung dan pembelajaran pemrograman *web*.

1. Persiapan Instalasi
 - a. Pastikan komputer anda tidak terinstall *web server* lain seperti IIS atau PWS karena dapat menyebabkan bentrok dengan *web server* Apache yang akan dipasang. Namun jika anda tetap ingin mempertahankannya, setelah installasi *web server* Apache selesai, anda dapat mengkonfigurasinya secara manual untuk mengganti nomor *port* yang akan digunakan oleh Apache.
 - b. Download source XAMPP versi 5.6.32 (yang akan digunakan di modul ini) pada <https://www.apachefriends.org/download.html> dan tersedia untuk sistem operasi Windows, Linux dan Mac.
2. Proses Instalasi
 - a. Jalankan *file installer* XAMPP
 - b. Akan ditampilkan jendela instalasi XAMPP. Pilih **Yes**. Peringatan tersebut menunjukkan bahwa antivirus sedang berjalan dan tidak akan mengganggu proses instalasi.



Gambar 1.7 Peringatan antivirus pada instalasi XAMPP

- c. Setelah itu akan muncul jendela peringatan UAC (*User Account Control*). Klik **OK**.



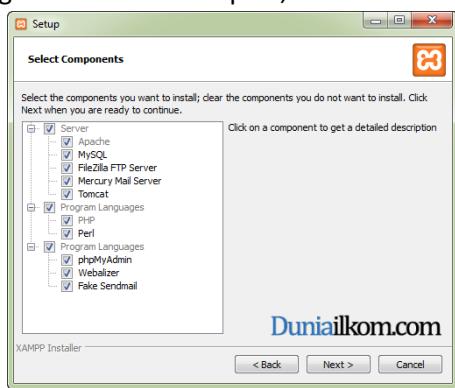
Gambar 1.8 Peringatan UAC pada instalasi XAMPP

- d. Akan muncul jendela awal instalasi. Klik **Next** untuk melanjutkan instalasi.



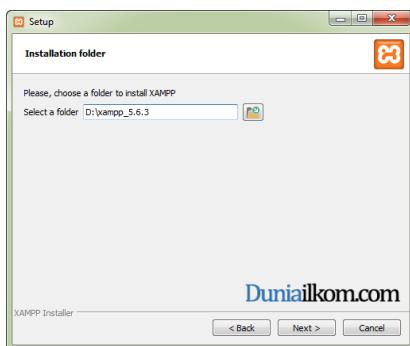
Gambar 1.9 Jendela awal instalasi XAMPP

- e. Jendela berikutnya adalah “**Select Component**”. Pada bagian ini kita bisa memilih aplikasi apa saja yang akan dipasang. Setelah selesai dipilih, klik **Next** untuk melanjutkan.



Gambar 1.10 Jendela pemilihan komponen pada XAMPP

- f. Selanjutnya adalah memilih lokasi dimana XAMPP akan dipasang. Jika sudah ditentukan lokasinya, klik next **Next**.



Gambar 1.11 Jendela pemilihan lokasi instalasi

- g. Tampilan berikutnya adalah jendela “**Bitnami for XAMPP**”. Hilangkan centang pada bagian “Learn more about Bitnami for XAMPP”. Klik **Next**.

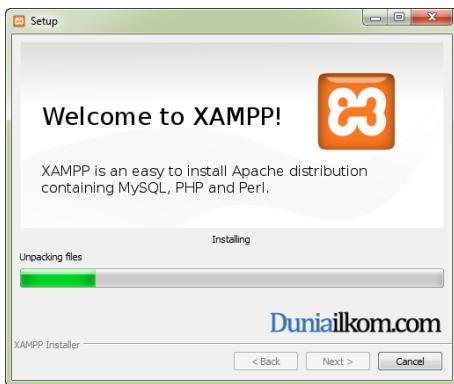


Gambar 1.12 Jendela instalasi Bitnami pada XAMPP

- h. Jendela berikutnya adalah konfirmasi untuk mulai memasang XAMPP. Klik **Next** dan proses instalasi akan berlangsung.



Gambar 1.13 Jendela konfirmasi instalasi



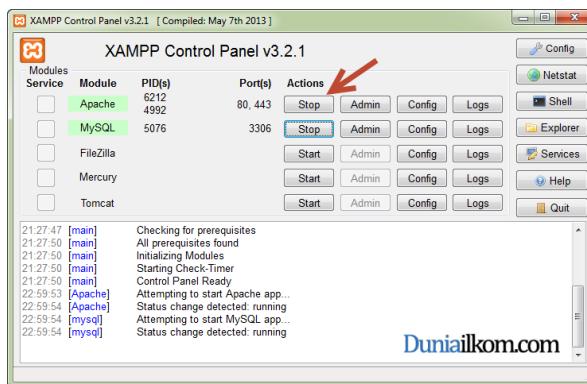
Gambar 1.14 Jendela proses instalasi

- i. Jika jendela “Completing the XAMPP Setup Wizard” telah tampil, maka proses instalasi XAMPP telah selesai. Centang bagian “Do you want to start the Control Panel now?” untuk menjalankan XAMPP.



Gambar 1.15 Jendela instalasi XAMPP selesai

- j. Akan muncul jendela Control Panel dari XAMPP. Pada bagian ini, kita bisa mengaktifkan *service* apa saja yang akan dijalankan dengan cara menekan tombol **Start** pada setiap *service* yang ada.



Gambar 1.16 Jendela ketika service pada XAMPP dijalankan

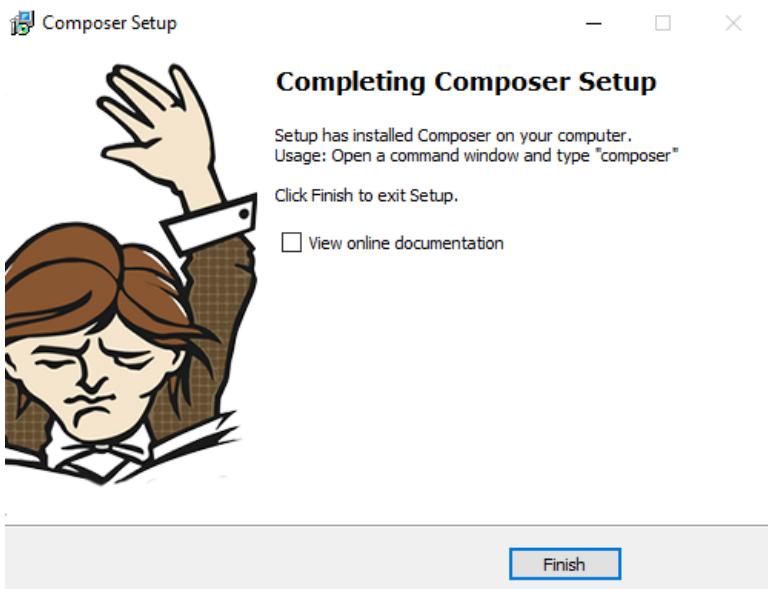
- k. Untuk mengujinya, buka *web browser* dan ketikkan alamat **localhost** pada *address bar*, kemudian tekan **enter**. Akan tampil jendela XAMPP yang menunjukkan bahwa proses instalasi dan *service* Apache berjalan dengan baik.



Gambar 1.17 Tampilan ketika localhost dijalankan pada browser

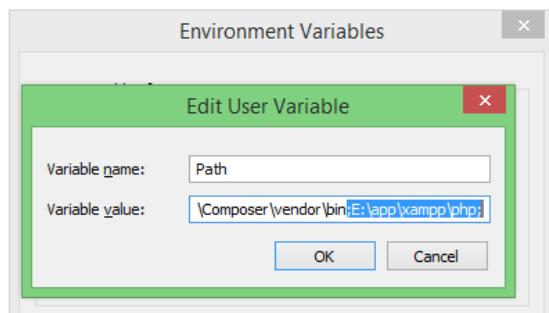
1.3. Instalasi Composer dan Laravel

Selain web server, instalasi Laravel membutuhkan aplikasi lain sebagai dependency manager yaitu Composer. Dengan adanya aplikasi ini, library-library yang diperlukan bisa seragam untuk semua tim dan dapat di-download dan diinstal secara mandiri secara lokal, sehingga tidak perlu dimasukkan ke dalam repository kode.



Gambar 1.18 Instalasi Composer

Setelah menginstal Composer, pastikan php.exe sudah dimasukkan ke PATH dalam environment variables pada sistem operasi. Contoh dapat dilihat pada gambar di bawah ini jika menggunakan Windows dan XAMPP. Tambahkan string “[path instalasi]/xampp/php”.



Gambar 1.19 Setting Path PHP

Kemudian, buka command prompt / console / terminal dan pindah ke folder dimana Laravel akan diinstal. Ketikkan perintah di bawah ini. Perintah ini akan men-download Laravel versi paling terakhir, ter-update, dan stable ke dalam folder “namafolderapp”.

```
composer create-project laravel/laravel namafolderapp
```

Jika tampilan seperti di bawah ini maka instalasi sudah berhasil.

```
> @php artisan key:generate --ansi
+ [32mApplication key set successfully. +[39m
E:\training>
```

Gambar 1.20 Instalasi Laravel

Masuk ke folder yang tadi di-define dalam perintah ketika instalasi (download), lalu ketikkan perintah di bawah ini untuk menjalankan aplikasi Laravel di web server.

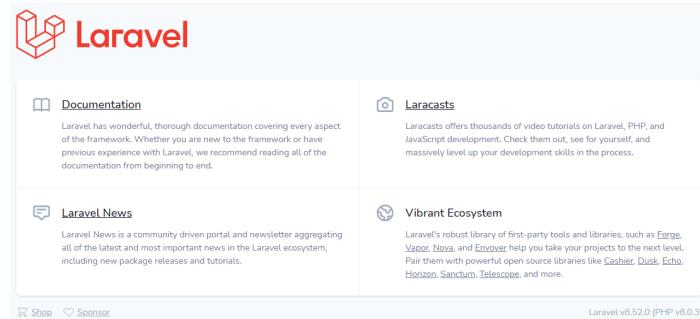
```
php artisan serve
```

Jika tampilan seperti di bawah ini maka Laravel sudah berhasil dijalankan di port default 8000.

```
E:\training>cd demo_app  
E:\training\demo_app>php artisan serve  
Starting Laravel development server: http://127.0.0.1:8000
```

Gambar 1.21 Menjalankan Laravel

Setelah itu buka browser dan akses URL <http://localhost:8000> atau <http://127.0.0.1:8000> untuk membuka welcome page default dari Laravel.

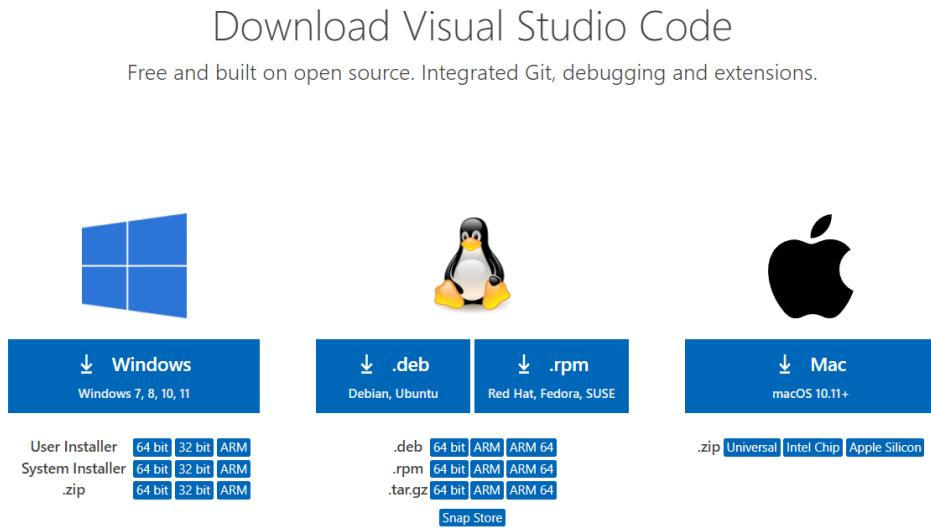


Gambar 1.22 Welcome Page Laravel

1.4. Instalasi Visual Studio Code

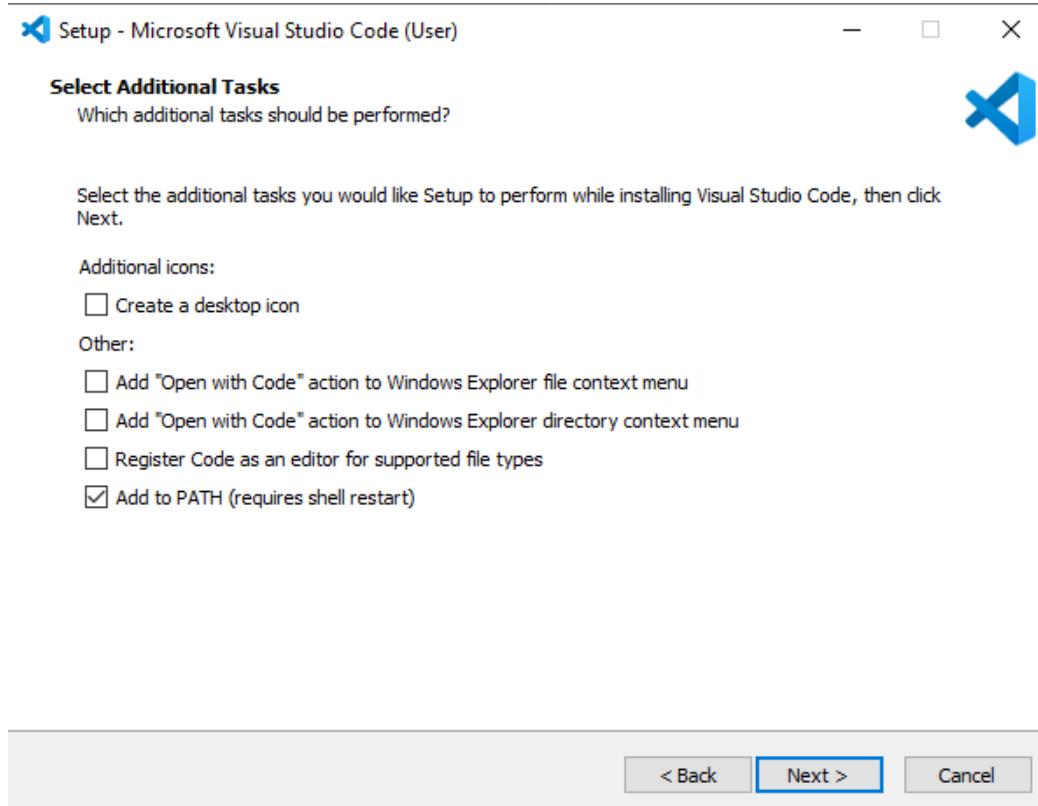
Pada kali ini, kita akan menggunakan visual studio code sebagai editor untuk membuat aplikasi menggunakan Flutter. Berikut langkah-langkah instalasinya:

1. Download visual studio code di <https://code.visualstudio.com/Download>
2. Pilih sistem operasi sesuai dengan device masing-masing



Gambar 1.23 Tampilan download visual studio code

3. Jika sudah di download, jalankan installer nya
4. Jika di tahap ini, biarkan default atau bawaan dari visual studio code nya



Gambar 1.24 Jendela setup visual studio code

5. Lalu pilih next, dan install
6. Tunggu hingga proses instalasi selesai

1.5. Instalasi JDK

JDK (Java Development Kit) merupakan perangkat yang digunakan untuk melakukan proses kompilasi dari kode java ke bytecode yang dapat dimengerti dan dapat dijalankan oleh JRE (Java Runtime Environment). Berikut merupakan tata cara dalam melakukan instalasi JDK:

1. buka link <https://www.oracle.com/java/technologies/javase-jdk15-downloads.html> untuk download, lalu pilih sesuai dengan sistem operasi pada perangkat yang digunakan.

Java SE Development Kit 15.0.2		
This software is licensed under the Oracle Technology Network License Agreement for Oracle Java SE		
Product / File Description	File Size	Download
Linux ALM 64 Bit RPM Package	141.82 MB	jdk-15.0.2_linux-x64_bin.rpm
Linux ALM 64 Compressed Archive	157 MB	jdk-15.0.2_linux-xarch64_bin.tgz
Linux x64 Debain Package	154.81 MB	jdk-15.0.2_linux-x64_bin.deb
Linux x64 RPM Package	162.03 MB	jdk-15.0.2_linux-x64_bin.rpm
Linux x64 Compressed Archive	179.35 MB	jdk-15.0.2_linux-x64_bin.tgz
macOS Installer	175.95 MB	jdk-15.0.2_macosx-x64_bin.dmg
macOS Compressed Archive	176.51 MB	jdk-15.0.2_macosx-x64_bin.tgz
Windows x64 Installer	193.71 MB	jdk-15.0.2_windows-x64_bin.exe

Gambar 1.25 Tampilan download jdk

2. Buka file instalasi yang telah didownload, lalu klik next.



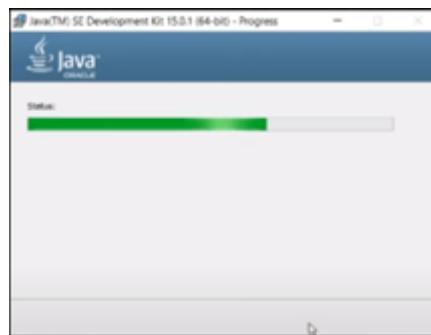
Gambar 1.26 Jendela awal instalasi JDK

3. Pilih folder path tempat menyimpan instalasi JDK, direkomendasikan sesuai dengan yang telah tertera pada proses instalasi, lalu klik next.



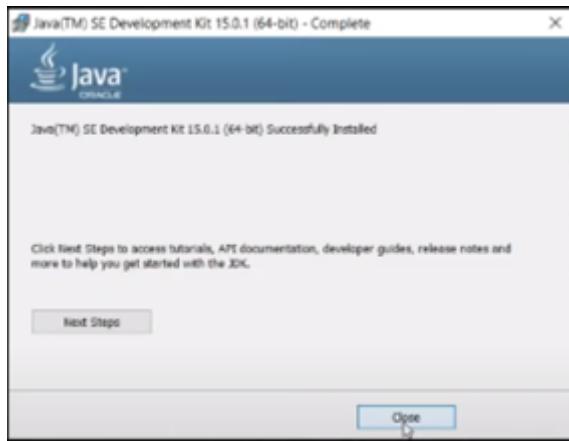
Gambar 1.27 Jendela folder path pada instalasi JDK

4. Tunggu instalasi hingga selesai.



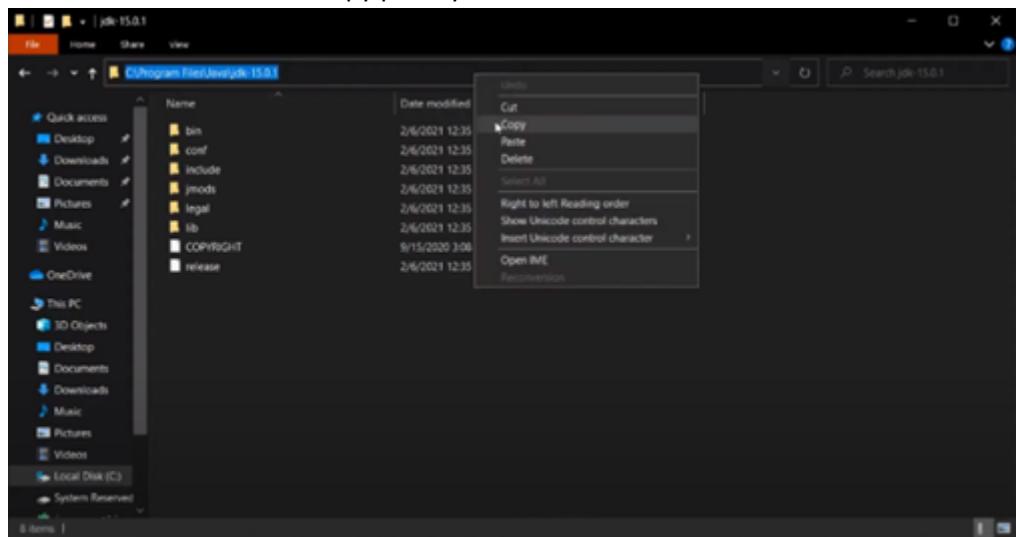
Gambar 1.28 Jendela tunggu pada instalasi JDK

5. Setelah proses instalasi selesai klik close.



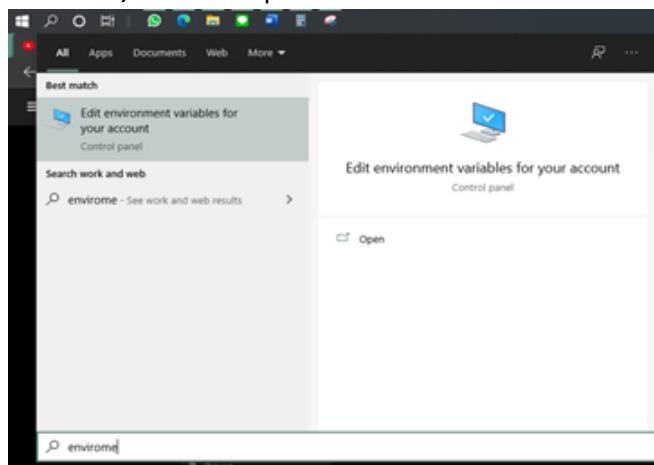
Gambar 1.29 Jendela finish instalasi pada JDK

6. Akses folder instalasi tadi lalu copy pathnya.



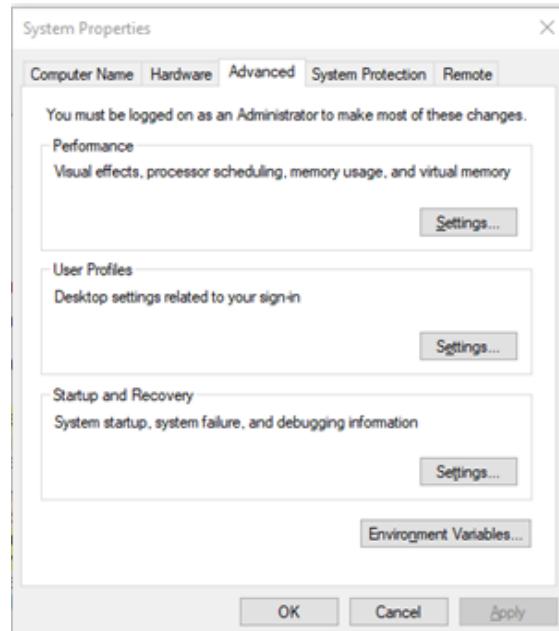
Gambar 1.30 Proses mencopy path jdk dari folder instalasinya

7. Search **environment variable**, lalu buka aplikasi tersebut.



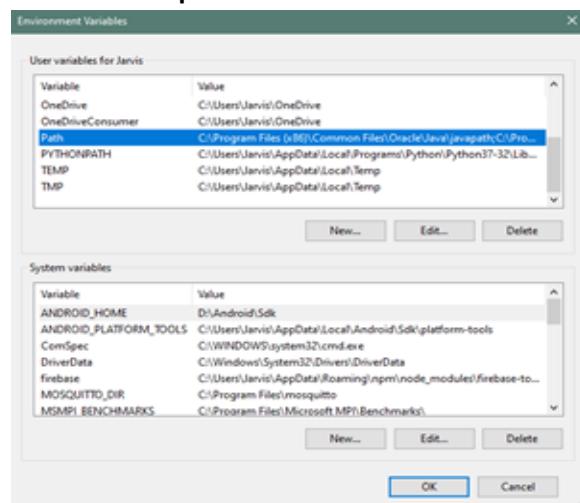
Gambar 1.31 Melakukan search environment variables

8. Pada tab advance klik **Environment Variables**



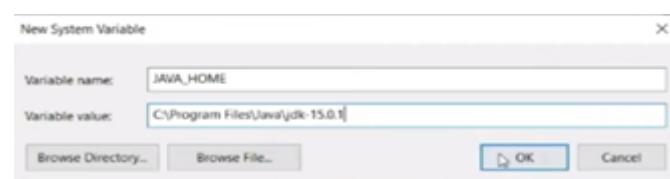
Gambar 1.32 Jendela system properties

9. Selanjutnya pada user variables klik path



Gambar 1.33 Jendela Environment Variables

10. Masukan informasi variable name "JAVA_HOME" lalu variable value (paste file path jdk tadi)



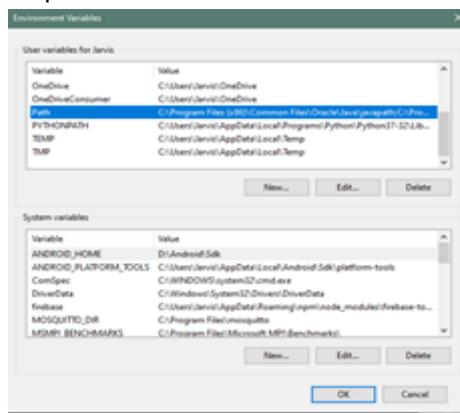
Gambar 1.34 Jendela New System Variable

11. Lakukan hal yang sama pada bagian **Sistem Variable**. Klik path -> masukan informasi yang sama



Gambar 1.35 Jendela New System Variabel

12. Klik **OK** untuk menyelesaikan proses instalasi.



Gambar 1.36 Jendela Environment Variables telah selesai diakses

1.6. Instalasi Flutter SDK

Sebelum melakukan instalasi Flutter, kita perlu menyiapkan dan menginstall tools yang dibutuhkan saat pengembangan aplikasi menggunakan Flutter. Berikut langkah-langkah instalasi Flutter:

1. Persyaratan Minimum

a. Windows

- Sistem Operasi Windows 7 SP1 atau lebih baru (64-bit), x86-64 based.
- Ruang Penyimpanan 1.64 GB (tidak termasuk IDE dan tools lainnya).
- Flutter bergantung pada tools yang ada pada environment:
 - Windows PowerShell 5.0 atau versi terbaru (sudah terdapat pada Windows 10). Bisa download pada link [ini](#).
 - Git for Windows 2.x, dengan opsi “Use Git from the Windows Command Prompt”. Dapat diunduh pada link [ini](#).

b. MacOs

- Sistem Operasi Mac OS 64-bit.
- Ruang penyimpanan 2.8 GB dan tidak termasuk IDE dan tools lainnya.
- Flutter tergantung pada command-line tools ini yang tersedia di environment:
 - bash
 - curl
 - git 2.x
 - Mkdir
 - rm
 - unzip
 - which

c. Linux

- Sistem Operasi Linux 64-bit.

- Ruang penyimpanan 1.8 GB dan tidak termasuk IDE dan tools lainnya.
- Flutter tergantung pada command-line tools ini yang tersedia di environment:
 - bash
 - curl
 - git 2.x
 - mkdir
 - rm
 - unzip
 - which
 - xz-utils

2. Instalasi Flutter SDK

Pada instalasi kali ini akan ditunjukkan langkah-langkah instalasi pada OS Windows. Untuk OS lainnya, bisa akses pada link [ini](#). Berikut langkah-langkah instalasinya:

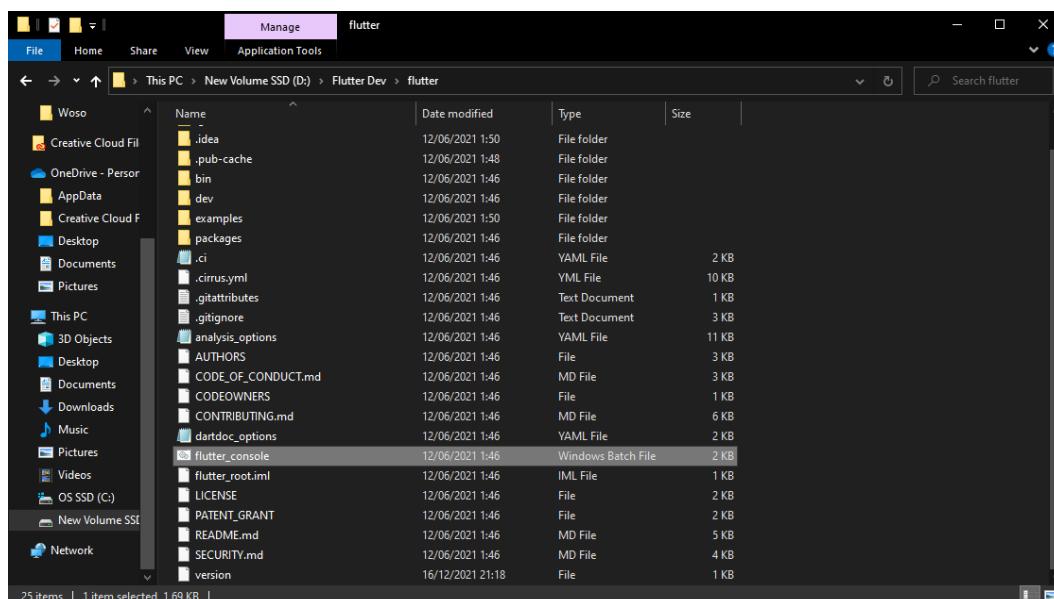
1. Unduh Flutter SDK pada link dibawah ini, dan pastikan unduh versi yang stabil dan yang terbaru dari Flutter. Sesuaikan juga dengan sistem operasi yang dimiliki. Flutter SDK dapat diunduh melalui link:

<https://flutter.dev/docs/development/tools/sdk/releases>

2. Ekstrak berkas zip dan tempatkan folder flutter pada lokasi instalasi yang diinginkan untuk Flutter SDK, misalnya C:\Development.

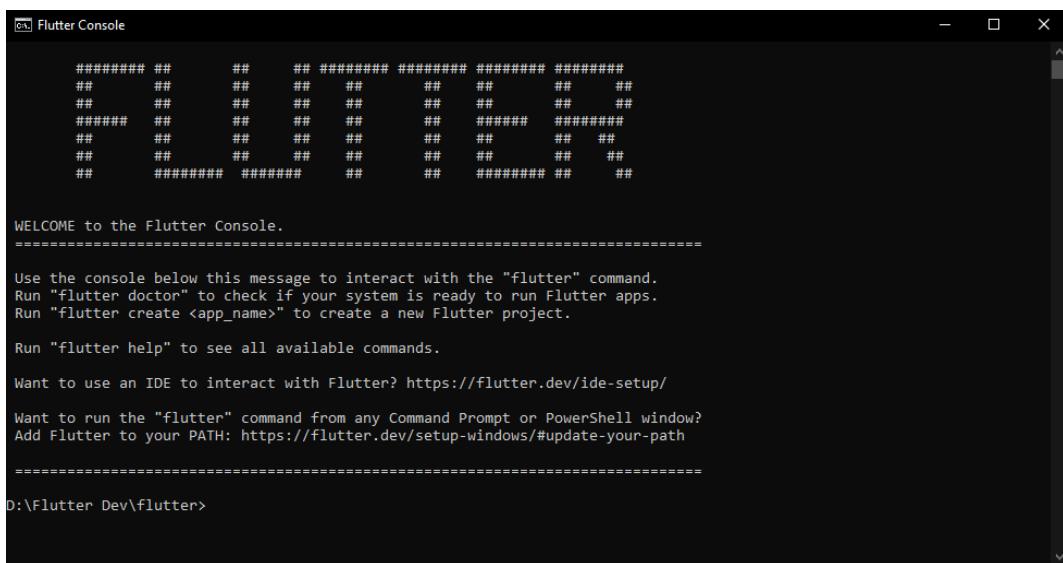
Catatan: Jangan pasang Flutter di direktori seperti C:\Program Files atau yang membutuhkan hak istimewa seperti administrator.

3. Temukan berkas flutter_console.bat di dalam direktori flutter tersebut. Mulai dengan klik dua kali atau jalankan script tersebut dan Anda sekarang siap untuk menjalankan perintah Flutter di Flutter Console.



Gambar 1.37 Menemukan berkas flutter_console.bat

4. Tampilan dari flutter_console.bat seperti di bawah ini:

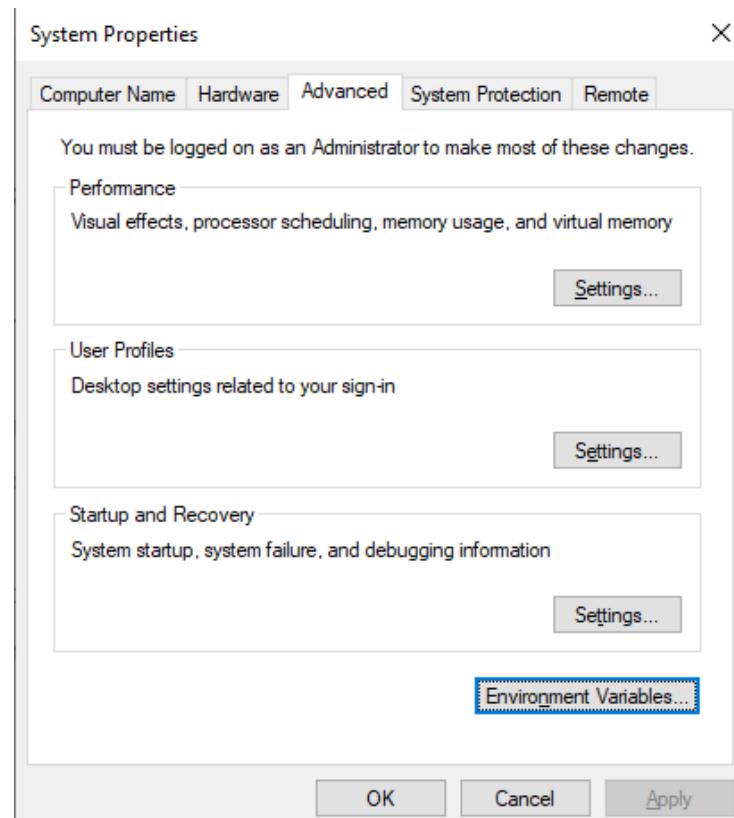


Gambar 1.38 Tampilan flutter_console.bat

3. Update Path

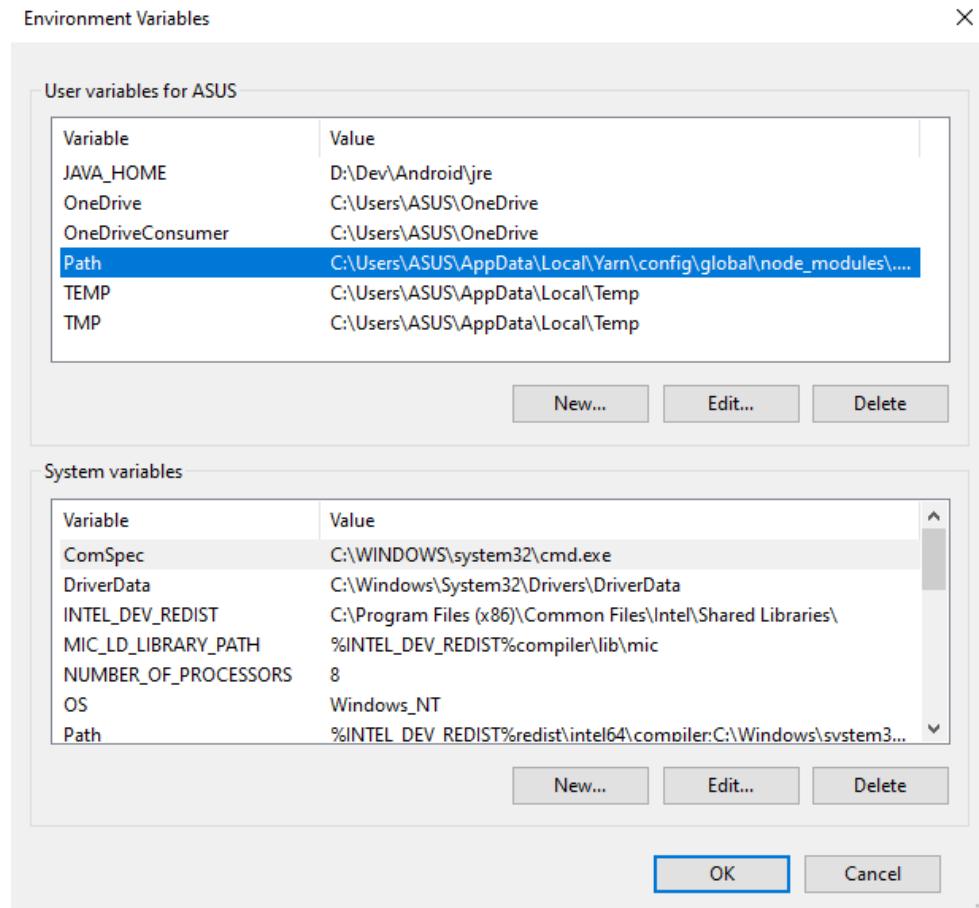
Langkah ini bertujuan agar perintah Flutter bisa digunakan pada command prompt/terminal. Berikut langkah-langkahnya:

1. Dari bar pencarian di Start menu, ketik 'env' dan pilih Edit Environment Variable untuk akun Anda.
 2. Klik pada tombol Environment Variables.



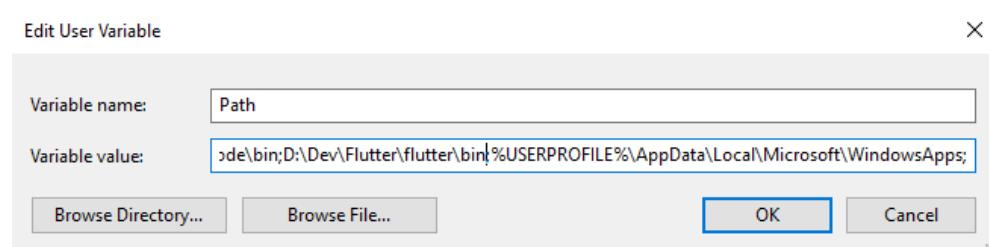
Gambar 1.39 Jendela system properties

3. Di bawah User variabel periksa apakah ada entri yang disebut PATH, jika ada maka pilih lalu edit, jika tidak ada maka buat baru dengan nama variabel Path.

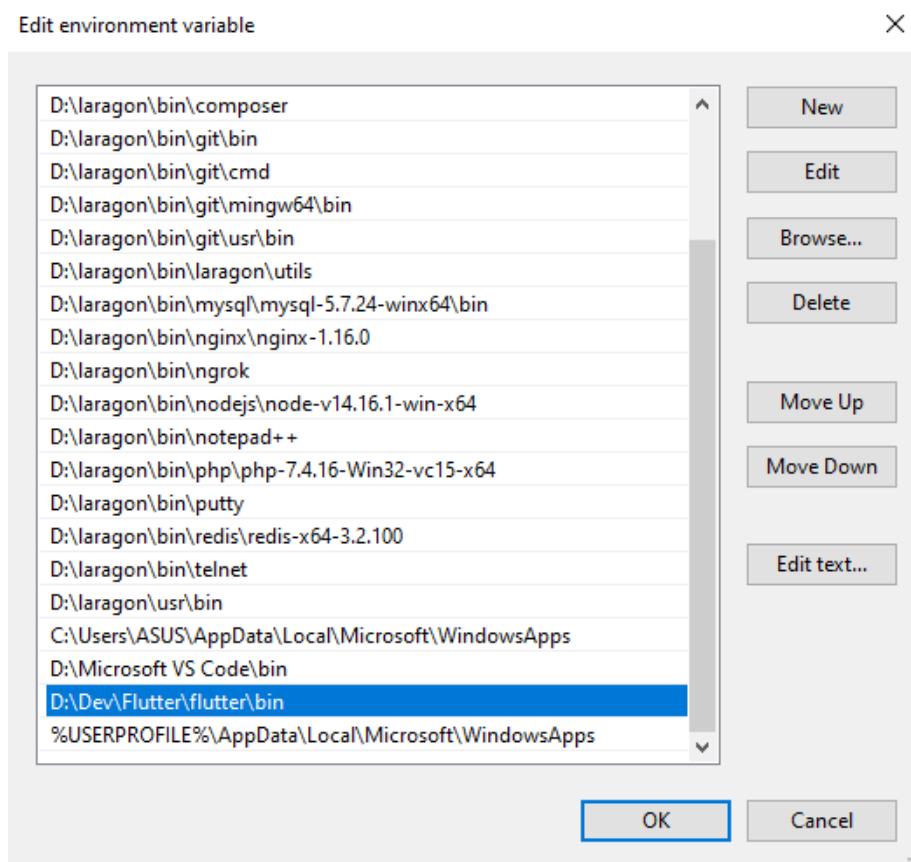


Gambar 1.40 Jendela Environment Variables

4. Edit atau tambahkan value-nya dengan direktori Flutter SDK.
- Jika terdapat entri, tambahkan path lengkap ke flutter\bin menggunakan tanda titik koma (;) sebagai pemisah dari nilai yang ada (jika menggunakan mode edit satu baris).
 - Jika entri tidak ditemukan, buat user variabel baru dan beri nama Path dan beri nilai flutter\bin sebagai nilainya.



Gambar 1.41 Jendela Edit User Variable



Gambar 1.42 Jendela Edit Environment Variable

Catatan: Anda harus menutup dan membuka kembali semua jendela konsol yang ada agar perubahan dapat terlihat.

4. Flutter Doctor

Flutter doctor merupakan perintah untuk mengecek kelengkapan framework flutter yang akan digunakan, seperti versi, Android SDK yang digunakan, iOS SDK yang digunakan (tersedia di MacOS), perangkat yang terhubung, dan lain-lain). Jalankan perintah berikut untuk membuka flutter doctor:

```
flutter doctor
```

Perintah ini memeriksa environment Anda dan menampilkan laporan ke jendela terminal. Pada Flutter SDK sudah terdapat Dart SDK, jadi Anda tidak perlu menginstal Dart secara terpisah. Periksa output dengan cermat untuk perangkat lunak lain yang mungkin perlu Anda instal atau melakukan sesuatu lebih lanjut (ditunjukkan dalam teks tebal).

Contoh:

```
[-] Android toolchain - develop for Android devices
  • Android SDK at D:\Android\sdk
** X Android SDK is missing command line tools; download from https://goo.gl/XxQghQ**
```

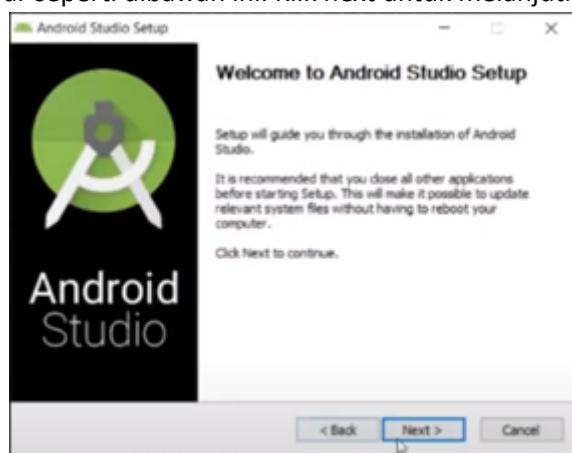
- Try re-installing or updating your Android SDK,
visit <https://flutter.io/setup/#android-setup> for detailed instructions.

Bagian tersebut menjelaskan cara menyelesaikan proses instalasi Flutter SDK. Setelah memasang dependensi yang hilang, jalankan perintah flutter doctor lagi untuk memverifikasi bahwa Anda telah mengatur semuanya dengan benar.

1.7. Instalasi Android Studio

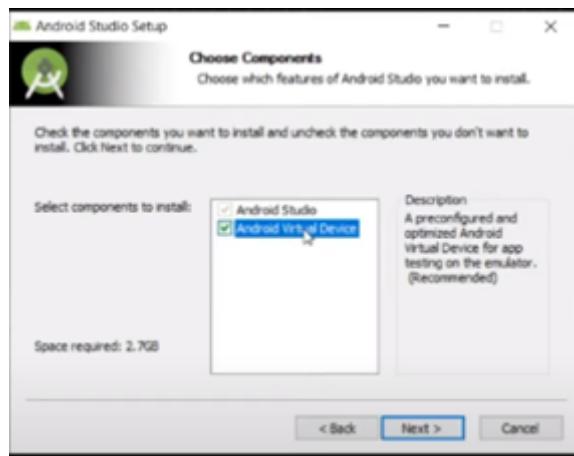
Android Studio merupakan IDE resmi dalam membangun aplikasi berbasis Android, proses instalasinya juga sederhana sehingga kita dapat dengan mudah membangun aplikasi menggunakan Android Studio. Pada modul ini kita akan menggunakan Android Studio sebagai IDE maka dari itu berikut hal-hal yang harus dipersiapkan dalam menginstall Android Studio:

1. Persiapan instalasi
 - a. Pastikan PC/Laptop yang digunakan memenuhi *minimum requirement* dalam menginstall aplikasi Android Studio seperti:
 - Microsoft® Windows® 7/8/10 (64-bit)
 - 4 GB RAM *minimum*, 8 GB RAM *recommended*
 - 2 GB of *available disk space minimum*,
 - 4 GB *Recommended* (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
 - b. Download aplikasi Android Studio terbaru pada <https://developer.android.com/studio>. Tersedia untuk Sistem Operasi Windows, Mac dan Linux.
2. Proses instalasi
 - a. Jika mendownload file .exe, *double-click* untuk menjalankan, Jika mendownload file .zip, buka paket ZIP, salin folder android-studio ke folder Program Files, lalu buka folder android-studio -> bin dan jalankan studio64.exe (untuk mesin 64-bit) atau studio.exe (untuk mesin 32-bit).
 - b. Akan tampil gambar seperti dibawah ini. Klik next untuk melanjutkan aplikasi.



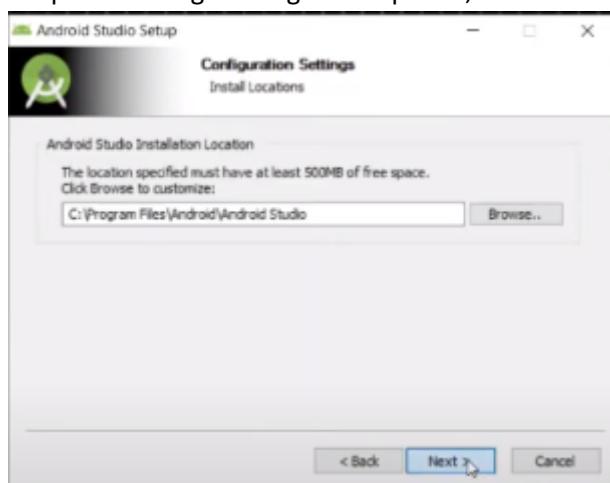
Gambar 1.43 Jendela awal instalasi Android Studio

- c. Pastikan **Android Virtual Device** ter-checklist untuk **optimasi virtual device** untuk testing pada emulator. Lalu klik next.



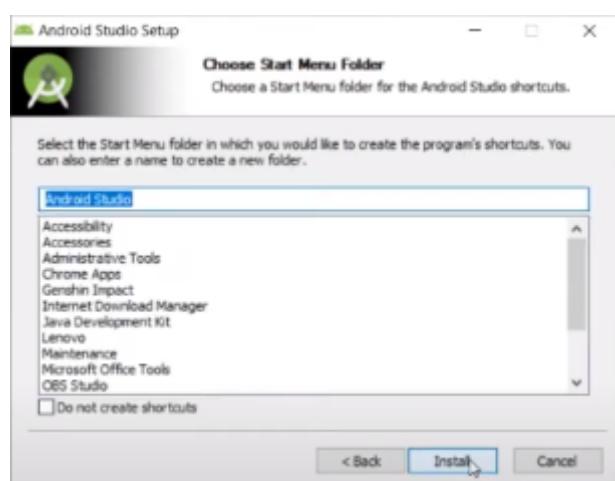
Gambar 1.44 Jendela Choose Components pada Instalasi Android Studio

- d. Pilih *folder path* tempat kamu ingin menginstall aplikasi, setelah itu klik next.



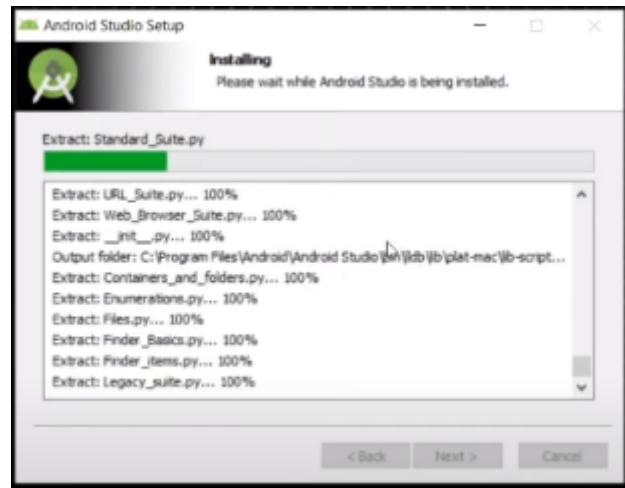
Gambar 1.45 Jendela Configuration Settings pada Instalasi Android Studio

- e. Pilih folder untuk membuat shortcut. Klik install untuk melanjutkan.



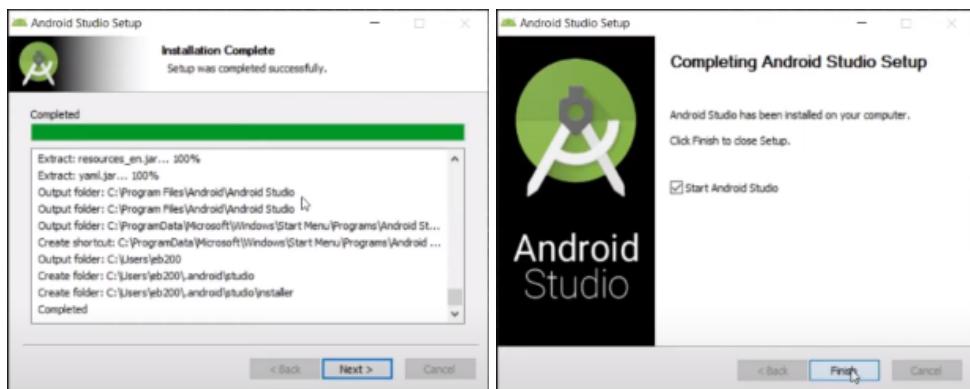
Gambar 1.46 Jendela Choose Start Menu Folder pada Android Studio

f. Tunggu proses instalasi hingga selesai



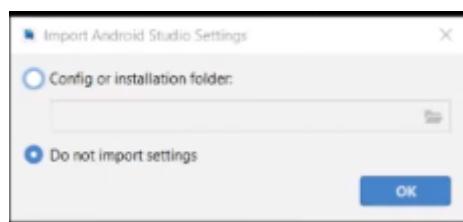
Gambar 1.47 Jendela proses instalasi pada Android Studio

g. Setelah proses instalasi selesai, klik next dan finish.



Gambar 1.48 Proses Instalasi dan Jendela finish pada Android Studio

h. Jika belum memiliki installation folder (biasanya pada pertama kali install) pilih **Do not import setting**.



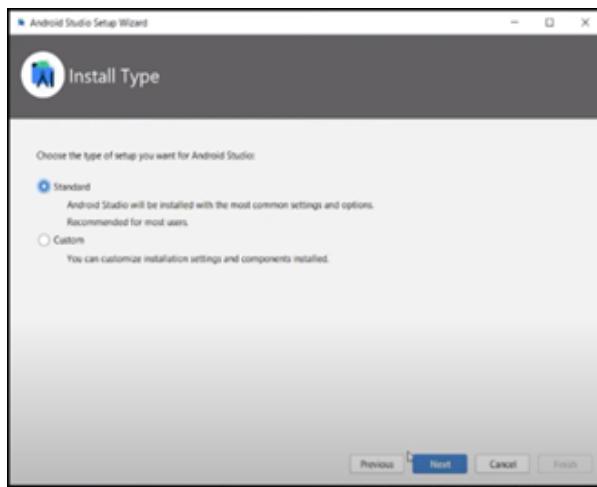
Gambar 1.49 Jendela menu installation folder

- i. Tunggu proses Android Studio hingga berjalan.



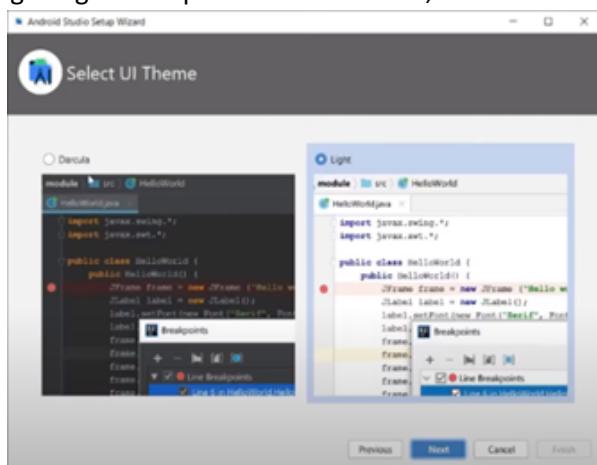
Gambar 1.50 Jendela proses tunggu masuk Android Studio

- j. Pilih *installation type* (*Standart is recomended*) untuk Android Studio, lalu klik next.



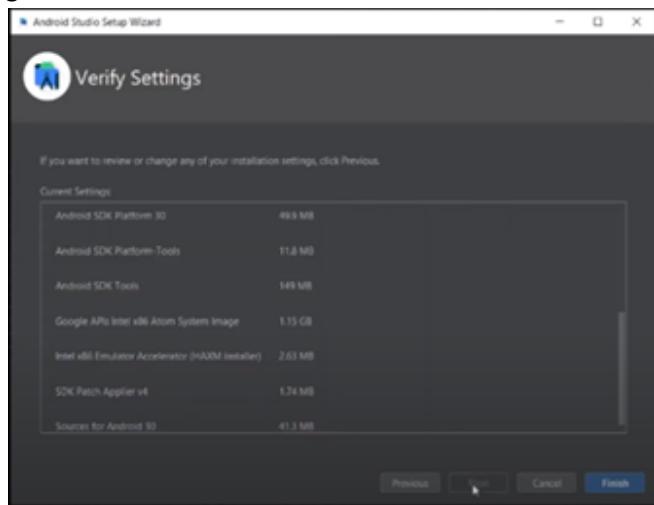
Gambar 1.51 Jendela Installation type

- k. Pilih tema yang ingin digunakan pada Android Studio, lalu klik next.



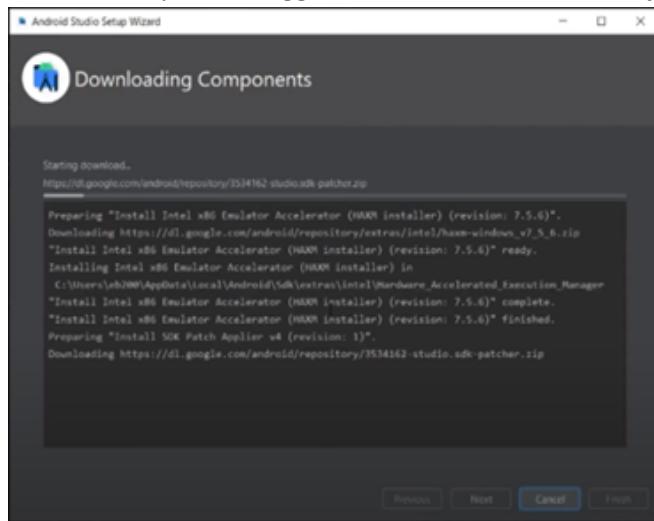
Gambar 1.52 Jendela Select UI Theme

- I. Kamu akan melihat daftar konfigurasi setting standart. Klik finish untuk mendownload komponen yang dibutuhkan.



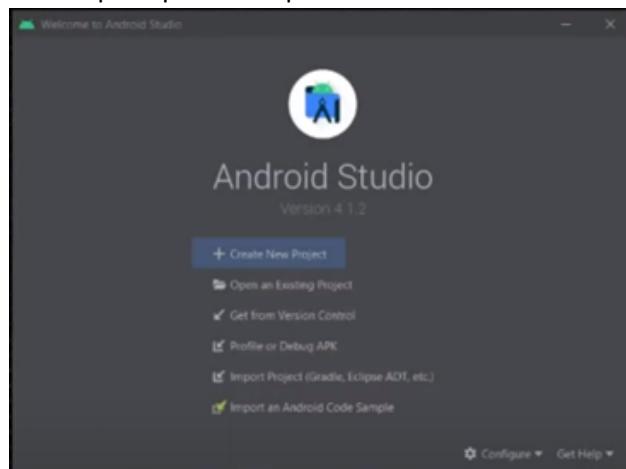
Gambar 1.53 Jendela Verify Settings

- m. Tunggu proses instalasi komponen hingga selesai. Setelah selesai klik *finish*.



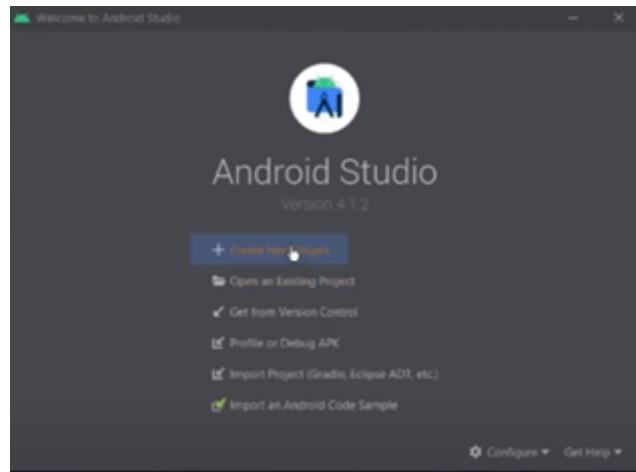
Gambar 1.54 Jendela Downloading Components

- n. Berikut merupakan tampilan pertama pada android studio setelah instalasi.



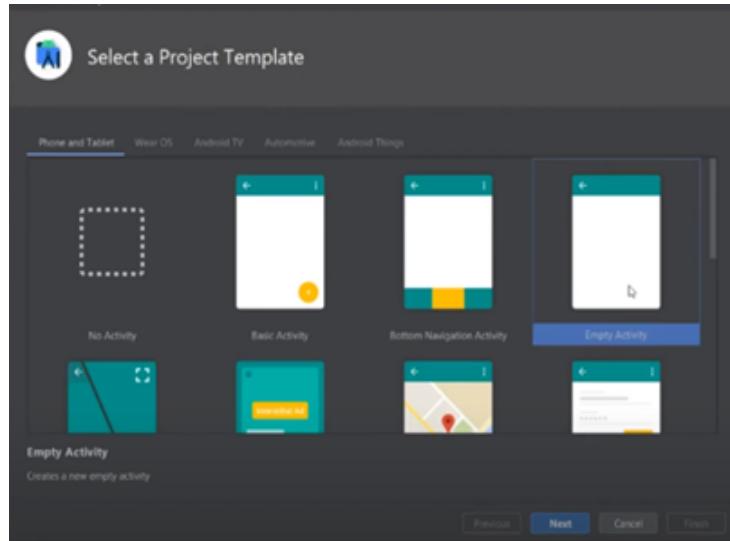
Gambar 1.55 Jendela tampilan pertama Android Studio

o. silahkan buat **New Project**.



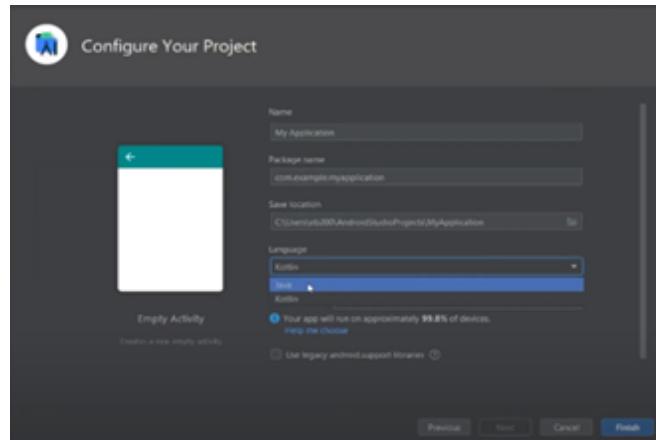
Gambar 1.56 Memilih New Project

p. Pilih Empty Activity lalu klik next.



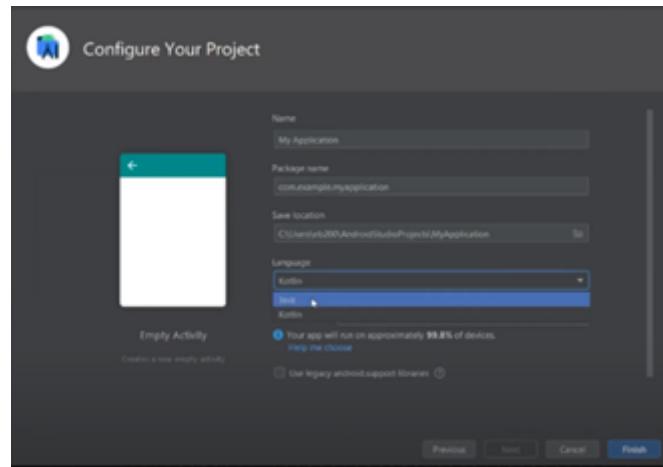
Gambar 1.57 Jendela Select a Project Template

q. Silahkan masukan nama aplikasi serta bahasa pemrograman yang diinginkan, pada modul ini kita akan menggunakan bahasa pemrograman Java, Klik finish.



Gambar 1.58 Jendela Configure Your Project

- r. Berikut merupakan tampilan awal pada aplikasi Android Studio setelah membuat projek baru.

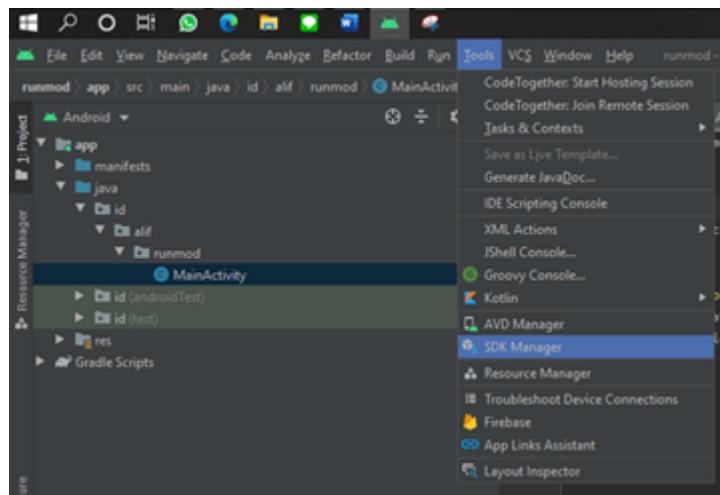


Gambar 1.59 Selesai mengatur untuk tampilan awal project

1.8. Instalasi SDK Android

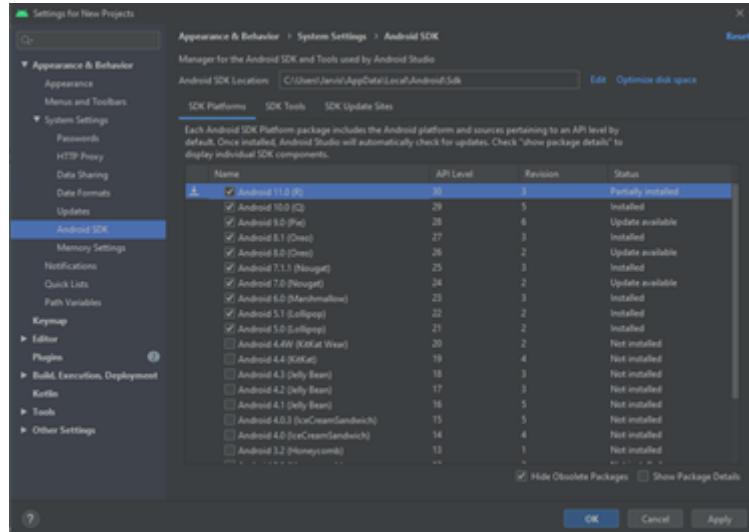
SDK (Standart Development Kit) merupakan kumpulan dari beberapa alat, komponen, juga platform untuk mengembangkan aplikasi berbasis android. SDK wajib ada pada Android Studio. Berikut merupakan cara instalasi SDK:

1. Klik **tools** pada menu bar lalu Klik **SDK Manager**.



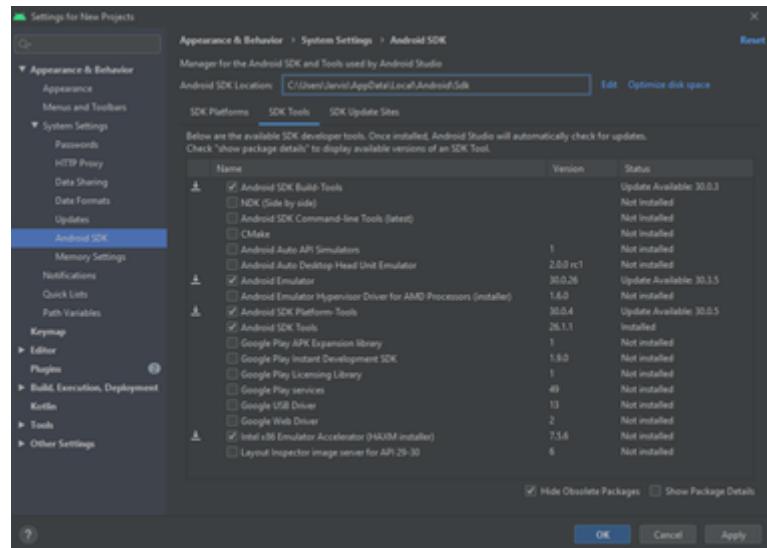
Gambar 1.60 Memilih SDK Manajer di Android Studio

2. Pada **SDK Platform** checklist SDK yang kamu butuhkan, lalu pindah ke tab **SDK Tools**.



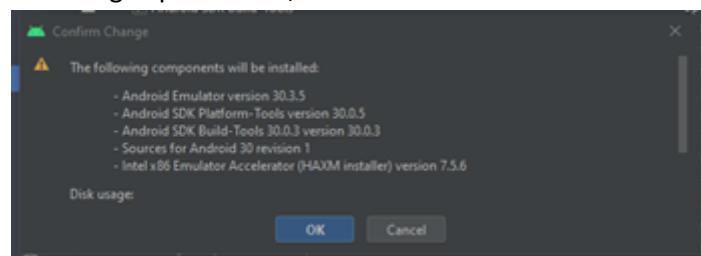
Gambar 1.61 Tampilan SDK Tools

- Pada **SDK Tools** checklist tools yang kamu butuhkan, setelah itu klik **OK**.



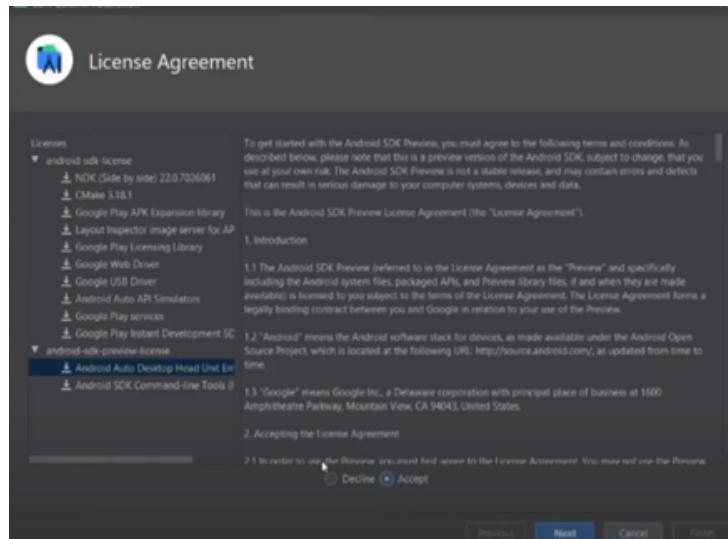
Gambar 1.62 Memilih tools yang dibutuhkan pada SDK Tools

- Lalu akan muncul keterangan perubahan, klik **OK**.



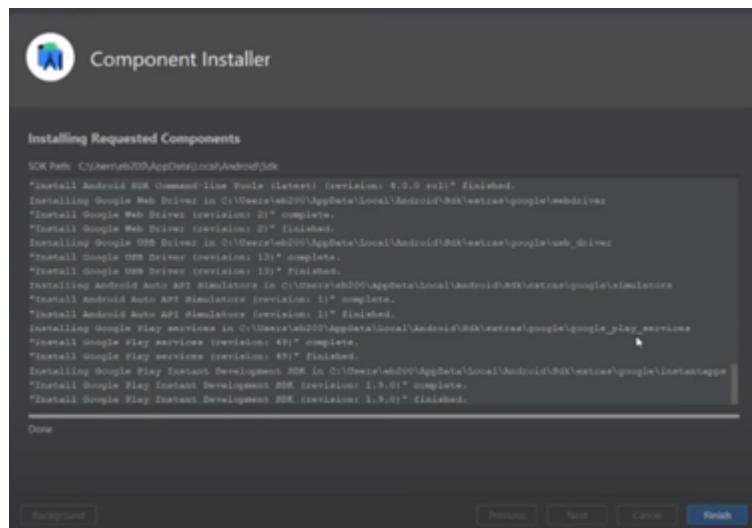
Gambar 1.63 Notifikasi untuk keterangan perubahan

- Accept setiap **Licence Agreement** untuk mendownload SDK tools yang dipilih, lalu klik next.



Gambar 1.64 Jendela Licence Agreement untuk download SDK Tools

6. Tunggu download dan instalasi hingga selesai, lalu klik Finish.



Gambar 1.65 Jendela finish instalasi SDK Android

MODUL 2. HTML & GIT

Tujuan Praktikum

1. Mahasiswa mampu memahami konsep dan implementasi HTML pada web.
2. Mahasiswa mampu memahami sintaks dan elemen-elemen HTML.
3. Mahasiswa mampu memahami penggunaan Git.

2.1. Pengenalan HTML

HTML atau *HyperText Markup Language* merupakan bahasa dasar yang digunakan untuk membangun sebuah *web* dimana HTML menangani elemen-elemen dasar pada pembangunan sebuah *website*. Struktur HTML paling dasar adalah sebagai berikut:



Gambar 2.1 Struktur dasar HTML

2.1.1. Tag HTML

Tag dalam HTML secara normal memiliki sepasang tag di mana tag pertama merupakan tag pembuka dan yang kedua merupakan tag penutup. Konten yang ingin ditampilkan pada laman *web* diletakkan di antara kedua tag tersebut.

`<nama_tag> letakkan konten di sini ... </nama_tag>`

Tag dalam HTML tidak semuanya berbentuk pasangan, ada beberapa tag yang hanya berdiri sendiri seperti tag "
" yang berguna untuk berpindah baris.

2.1.2. Elemen HTML

Elemen HTML merupakan tag HTML yang telah memiliki konten atau isi di antara kedua tag pembuka dan penutupnya. Elemen HTML dapat berupa teks atau juga dapat menyisipkan tag HTML lain pada elemen tersebut.

```
<!DOCTYPE html>
<html>
  <head> <!-- Contoh elemen berisi tag lain -->
    <title>Page Title</title>
  </head>                               asdff<body>
    <h1>My First Heading</h1> <!-- Contoh Elemen berisi Teks -->
    <p>My first paragraph.</p>
  </body>
</html>
```

2.1.3. Atribut HTML

Atribut HTML merupakan tambahan informasi dari sebuah tag HTML. Bentuk atribut untuk setiap tag HTML berbeda-beda sehingga kegunaan atribut juga berbeda seperti menambahkan informasi warna elemen, ukuran lebar, ukuran panjang dan lain-lain. Namun, mayoritas atribut yang sering muncul untuk setiap tag HTML adalah atribut "id" dan "class" karena kedua atribut ini berperan besar dalam

pengembangan laman *web* dengan CSS dan JavaScript. Atribut HTML dideklarasikan di dalam *tag* pembuka pada setiap elemen HTML dengan format **nama_atribut="value"**, setiap nilai atribut diapit oleh petik dua.

```
<a href="www.google.co.id" > Google.co.id </a>
<input type="button" id="btnSubmit" class="btnSubmit1" value="Kirim"/>
```

2.2. Dasar Sintaks HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>Page Title</title>
    </head>
    <body>
        <h1>My First Heading</h1>
        <p>My first paragraph.</p>
    </body>
</html>
```

Seperti yang sudah dijelaskan sebelumnya struktur dasar HTML antara lain berupa :

- Deklarasi `<! DOCTYPE html>` mendefinisikan dokumen menjadi HTML5
- Elemen `<html>` adalah elemen dasar dari halaman HTML
- Elemen `<head>` berisi informasi meta tentang dokumen
- Elemen `<title>` menentukan judul untuk dokumen
- Elemen `<body>` berisi konten halaman yang terlihat

2.3. Heading

Heading pada HTML merupakan *tag* yang berguna untuk menampilkan judul dari konten laman *web* yang dibangun. *Heading* dalam sebuah laman *web* berperan penting untuk aplikasi mesin pencarian karena sistem mesin pencarian bekerja dengan menggunakan *Heading* laman *web* kita sebagai *index* pencarian. Dalam HTML terdapat enam tingkatan *Heading* di mana semakin kecil nilai *heading* nya maka semakin penting dan semakin besar ukurannya pada laman *web*.

```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```

Heading 1

Heading 2

Heading 3

Heading 4

Heading 5

Heading 6

Gambar 2.2 Tingkatan heading

2.4. Hyperlink

Hyperlink dalam HTML memungkinkan halaman *web* berpindah laman atau bernavigasi menuju laman *web* yang lain. Tag yang digunakan adalah tag `<a>...`

<code> Visit Google </code>
Output :
<u>Visit Google</u>
Jika panah kursor kita arahkan pada teks tersebut, kursor akan berubah menjadi bentuk tangan.

Dalam tag *Hyperlink* pada HTML ada satu atribut yang harus digunakan agar konten yang ada di antara tag *hyperlink* berjalan dan dapat melakukan navigasi menuju laman *web* lain yaitu atribut `href`. Atribut ini bernilai *url* atau alamat dari laman *web* tujuan.

2.5. Tabel

Tabel pada HTML merupakan salah satu elemen penting khususnya digunakan untuk menampilkan data yang membutuhkan bentuk tabel. Tabel pada HTML didefinisikan dengan tag `<table></table>` dengan setiap pendefinisian baris menggunakan tag `<tr></tr>`, pendefinisian *heading* tabel menggunakan tag `<th></th>` dan pendefinisian kolom menggunakan tag `<td></td>`.

```
<table width="80%" height="50%" border="1">
    <tr>
        <th>Nama Lengkap</th>
        <th>Kota Kelahiran</th>
        <th>Age</th>
    </tr>
    <tr>
        <td>Budi</td>
        <td>Jakarta</td>
        <td>35</td>
    </tr>
    <tr>
        <td>Andi</td>
        <td>Semarang</td>
        <td>52</td>
    </tr>
    <tr>
        <td>Rasyid</td>
        <td>Surabaya</td>
        <td>22</td>
    </tr>
</table>
```

Nama Lengkap	Kota Kelahiran	Age
Budi	Jakarta	35
Andi	Semarang	52
Rasyid	Surabaya	22

Gambar 2.3 Tabel pada HTML

Dalam tabel HTML kita dapat melakukan operasi *Merge Cell* yang biasanya dapat dilakukan pada aplikasi perkantoran seperti Microsoft Word atau Excel dengan cara menambahkan atribut `colspan` dan `rowspan` pada tag pembuka kolom yaitu `<td>` nilai dari atribut tersebut berupa jumlah kolom atau baris yang akan dgabungkan.

```

<table width="80%" height="50%" border="1">
  <tr>
    <th rowspan="2">Nama Lengkap</th>
    <th colspan="2">Gelar Pendidikan</th>
    <th rowspan="2">Age</th>
  </tr>
  <tr>
    <th> Sarjana </th>
    <th> Magister </th>
  </tr>
  <tr>
    <td>Budi</td>
    <td>S.Kom</td>      <td>M.Sc</td>
    <td>35</td>
  </tr>
  <tr>
    <td>Andi</td>
    <td>S.SiKom</td>
    <td>M.T</td>
    <td>52</td>
  </tr>
</table>

```

Nama Lengkap	Gelar Pendidikan		Age
	Sarjana	Magister	
Budi	S.Kom	M.Sc	35
Andi	S.SiKom	M.T	52

Gambar 2.4 Penggunaan colspan dan rowspan pada tabel HTML

2.6. Image

Menampilkan gambar pada halaman *web* merupakan sebuah improvisasi dalam pembuatan desain sebuah *web* yang dapat memperindah tampilan *website*. Tag HTML yang digunakan adalah `` tag ini tidak memiliki pasangan penutup maka dari itu diakhir tag pembuka ditambahkan garis miring seperti di atas. Terdapat satu atribut wajib yang harus ditambahkan seperti atribut `href` pada tag *Hyperlink* yaitu atribut `src` yang bernilai alamat direktori gambar disimpan.

```


<p>Ini Gambar Kincoir Angin</p>

```



Ini Gambar Kincir Angin

Gambar 2.5 Penggunaan image pada HTML

2.7. Audio / Video Elemen

Sebelum berkembangnya teknologi HTML5, untuk menyisipkan audio atau video, diperlukan sebuah *plugin* seperti *Flash Player* namun sekarang dengan HTML5 memiliki *tag* yang dapat menyisipkan audio atau video ke dalam laman *web*. Untuk audio menggunakan *tag* `<audio>` untuk *tag* pembuka dan `<source>` untuk memanggil *url* atau alamat direktori *file*. Sedangkan untuk video menggunakan *tag* `<video>`.

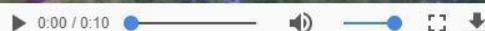
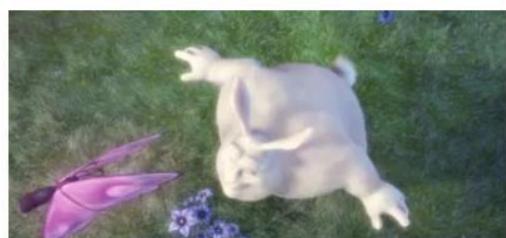
```

<audio controls>
  <source src="horse.ogg" type="audio/ogg">   <source src="horse.mp3"
  type="audio/mpeg"> Your browser does not support the audio element.
</audio>

<video width="400" controls>
  <source src="mov_bbb.mp4" type="video/mp4">   <source src="mov_bbb.ogv"
  type="video/ogg"> Your browser does not support HTML5 video.
</video>

<p>
Video courtesy of
<a href="https://www.bigbuckbunny.org/" target="_blank">Big Buck Bunny</a>. </p>

```



Video courtesy of [Big Buck Bunny](https://www.bigbuckbunny.org/).

Gambar 2.6 Penggunaan audio dan video pada HTML

2.8. Form

Form pada HTML digunakan sebagai wadah untuk menampung dan mengumpulkan data-data dari pengguna jika diperlukan untuk disimpan dalam sebuah *database*. Tag dasar untuk pemanggilan *form* adalah <form> ... </form> dan diantara tag *form* tersebut merupakan tempat mendefinisikan elemenelemen yang dibutuhkan *form* yang akan dibuat nantinya. Atribut utama dari tag *form* yaitu **action**, atribut ini bernilai tujuan data akan diolah dengan bahasa pemrograman *web* saat tombol "Submit" ditekan, selain itu atribut **method** yang hanya bernilai POST atau GET ini juga sangat dibutuhkan untuk pengolahan data dengan bahasa pemrograman *web*. Pembahasan lebih lanjut ada pada modul PHP.

2.8.1. Elemen Form

Elemen-elemen form pada html adalah sebagai berikut:

Tabel 2.1 Elemen form

Tag	Type	Fungsionalitas
<input/>	type = "text"	Menampilkan elemen untuk input data teks
	type = "password"	Menampilkan elemen untuk input data password
	type = "email"	Menampilkan elemen untuk input data email
	type = "radio"	Menampilkan elemen untuk pemilihan data berbentuk radio
	type = "checkbox"	Menampilkan elemen untuk pemilihan data berbentuk checkbox
	type = "submit"	Menampilkan elemen tombol untuk pengolahan data <i>form</i>
<select> <option> ... </option> </select>	-	Menampilkan elemen untuk pemilihan data berbentuk <i>dropdown list</i>
<textarea> ... </textarea>	-	Menampilkan elemen untuk input data dalam bentuk paragraf panjang.
<button> ... </button>	type = "button"	

2.8.2. Atribut Elemen Form

Atribut-atribut elemen *form* yang sering digunakan antara lain sebagai berikut :

Tabel 2.2 Atribut elemen form

Atribut	Value	Fungsionalitas
id	Bebas	Menambahkan informasi ID dari elemen untuk kebutuhan CSS, JavaScript atau Bahasa Pemrograman Web yang lain.
name	Bebas	Menambahkan informasi Name dari elemen untuk kebutuhan JavaScript atau Bahasa Pemrograman Web yang lain.
class	Bebas	Menambahkan informasi Class dari elemen untuk kebutuhan CSS, JavaScript atau Bahasa Pemrograman Web yang lain.
placeholder	Bebas	Menampilkan informasi sementara tentang elemen sebelum memberikan inputan pada elemen.

value	Bebas	Memberikan nilai <i>default</i> pada elemen khususnya elemen type dan option
disabled	-	Membuat elemen menjadi tidak dapat dioperasikan
readonly	-	Membuat elemen type tidak dapat dirubah atau diberi inputan
checked	-	Membuat elemen <i>checkbox</i> atau <i>radio button</i> menjadi terpilih secara <i>default</i> .
selected	-	Membuat elemen option pada tag <i>select</i> menjadi terpilih secara <i>default</i> .

```

<!DOCTYPE html>
<html>
    <head>
        <title>Image HTML</title>
        <link rel="stylesheet" href="">
    </head>
    <body>
        <h2> Formulir Pendaftaran Praktikan </h2>
        <form action="" method="POST">
            <table border="0" width="120">
                <tr>
                    <td>Nama</td>
                    <td>:</td>
                    <td width="900"><input type="text" name="nama_input" class="text-field" id="nama_id" placeholder="Input Nama" value="Praktikan" readonly /></td>
                </tr>
                <tr>
                    <td>Username</td>
                    <td>:</td>
                    <td><input type="text" name="uname_input" class="text-field" id="uname_id" placeholder="Input Username" value="Praktikum" disabled/></td>
                </tr>
                <tr>
                    <td>Password</td>
                    <td>:</td>
                    <td>
                        <input type="password" name="password_input" class="text-field" id="password_id" placeholder="Input Password"/>
                    </td>
                </tr>
                <tr>
                    <td>Email</td>
                    <td>:</td>
                    <td><input type="email" name="email_input" class="text-field" id="email_id" placeholder="Input Email"/></td>
                </tr>
                <tr>
                    <td>Jenis Kelamin</td>
                    <td>:</td>
                    <td>
                        <input type="radio" name="jk_input" class="radioB" id="radioB" value="Pria">Pria</input>
                        <input type="radio" name="jk_input" class="radioB" id="radioB" value="Wanita">Wanita</input>
                    </td>
                </tr>
            </table>
        </form>
    </body>
</html>

```

```

        <tr>
            <td>Hobi</td>
            <td>:</td>
            <td>
                <input type="checkbox" name="hobi_input"
class="checkBox" id="checkB_id" value="Renang">Renang</input><br/>
                <input type="checkbox" name="hobi_input"
class="checkBox" id="checkB_id" value="Bersepeda">Bersepeda</input><br/>
                <input type="checkbox" name="hobi_input"
class="checkBox" id="checkB_id" value="Memancing">Memancing</input>
            </td>
        </tr>
        <tr>
            <td>Jenjang Pendidikan</td>
            <td>:</td>
            <td>
                <select name="jp_input" class="dropdown"
id="jp_id">
                    <option value="" selected>-----Pilih-----</option>
                    <option value="D3">Tamat D3</option>
                    <option value="S1">Tamat S1</option>
                    <option value="S2">Tamat S2</option>
                    <option value="S3">Tamat S3</option>
                </select>
            </td>
        </tr>
        <tr>
            <td>Kritik & Saran</td>
            <td>:</td>
            <td>
                <textarea rows="5" cols="40%"></textarea>
            </td>
        </tr>
        <tr>
            <td colspan="3" align="right">
                <button type="Button" name="btnCnc"
class="btnCancel" id="id_btncnc">Cancel</button>
                <input type="Submit" name="btnSub"
class="btnSubmit" id="id_btnsub" value="Submit"/>
            </td>
        </tr>
    </table>
</form>
</body>
</html>

```

Formulir Pendaftaran Praktikan

The form consists of several input fields:

- Nama : Praktikan
- Username : Praktikum
- Password : (redacted)
- Email : Input Email
- Jenis Kelamin : Pria Wanita
- Hobi : Renang Bersepeda Memancing
- Jenjang Pendidikan :
- Kritik & Saran :

Below the dropdown menu, there is a list of options:
Tamat D3
Tamat S1
Tamat S2
Tamat S3

At the bottom right of the form area are two buttons: Cancel and Submit.

Gambar 2.7 Contoh form pendaftaran

2.9. Penggunaan Git

2.9.1. Membuat repositori baru

Untuk membuat repositaru baru, gunakan perintah dibawah ini:

```
git init modul-1
```

Jika berhasil maka tampilannya akan seperti ini pada *command prompt* Anda.

```
D:\WEBPRO>git init modul-1
Initialized empty Git repository in D:/WEBPRO/modul-1/.git/
```

Gambar 2.8 Inisialisasi repositori git

Perintah **git init** akan membuat sebuah direktori bernama **.git** di dalam proyek yang akan dikerjakan. Direktori ini digunakan Git sebagai *database* untuk menyimpan perubahan yang kita lakukan.

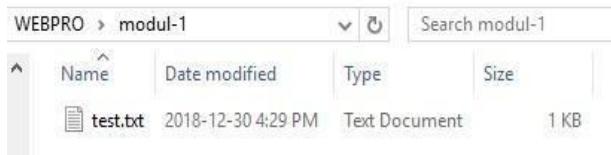
2.9.2. Menambahkan isi repositori

Untuk menambahkan suatu *file* ke dalam repositori, kita langsung dapat menambahkan *file* yang kita inginkan ke dalam *folder* projek yang telah kita buat, contohnya dapat dilihat dalam dengan perintah yang ada pada Gambar 2-9.

```
D:\WEBPRO\modul-1>touch test.txt
D:\WEBPRO\modul-1>echo "Halo Git" >> test.txt
D:\WEBPRO\modul-1>cat test.txt
"Halo Git"
D:\WEBPRO\modul-1>ls
test.txt
```

Gambar 2.9 Langkah membuat file

touch test.txt adalah perintah untuk membuat satu *file* baru yaitu *test.txt*, **echo "Halo Git" >> test.txt** adalah perintah untuk mengisi *file* *test.txt* dengan "Halo Git", lalu gunakan perintah **cat test.txt** untuk melihat apa isi yang ada pada *file* *test.txt*



Gambar 2.10 Hasil file yang telah dibuat

Dapat kita lihat bahwa dengan mengeksekusi perintah **git status** kita dapat melihat *file* yang berubah yang ada pada project kita.

```
D:\WEBPRO\modul-1>git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    test.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Gambar 2.11 Hasil perintah dari git status

Namun harus diingat bahwa *untracked files* menandakan bahwa *file* tersebut belum disimpan dalam catatan repository kita, untuk menyimpannya, kita bisa menggunakan perintah **git add test.txt**

```
D:\WEBPRO\modul-1>git add test.txt
D:\WEBPRO\modul-1>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   test.txt
```

Gambar 2.12 File telah ditambahkan

Dapat kita lihat bahwa sekarang file test.txt siap untuk disimpan, namun data belum benar-benar tersimpan sampai kita melakukan perintah **git commit -m "pesan commit"**. Setelah perintah **git commit**, maka **git** akan menyimpan semua perubahan yang ada, dan dapat dilihat dengan menggunakan perintah **git status**, maka akan jadi seperti ini.

```
D:\WEBPRO\modul-1>git commit -m "menambahkan test file"
[master (root-commit) 1570abc] menambahkan test file
  1 file changed, 1 insertion(+)
  create mode 100644 test.txt

D:\WEBPRO\modul-1>git status
On branch master
nothing to commit, working tree clean
```

Gambar 2.13 Setelah commit dan melihat status

2.9.3. Membuat repositori online

Pada tahap ini kita akan menggunakan tempat penyimpanan repositori *online* yang cukup popular, yaitu Github untuk menyimpan hasil pekerjaan kita. Pastikan kita sudah memiliki akun Github, dan sudah masuk kedalam akun kalian. Langkah yang harus kalian lakukan adalah sebagai berikut:

1. Buka link berikut ini. <https://github.com/new>

2. Isi detail dari *repository* yang akan kalian buat, lihatlah contoh acuan pada gambar 2-14.

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: kamalhm / Repository name: Webpro

Great repository names are short and memorable. Need inspiration? How about `literate-octo-spork`.

Description (optional): Belajar membuat repository

Public: Anyone can see this repository. You choose who can commit.

Private: You choose who can see and commit to this repository.

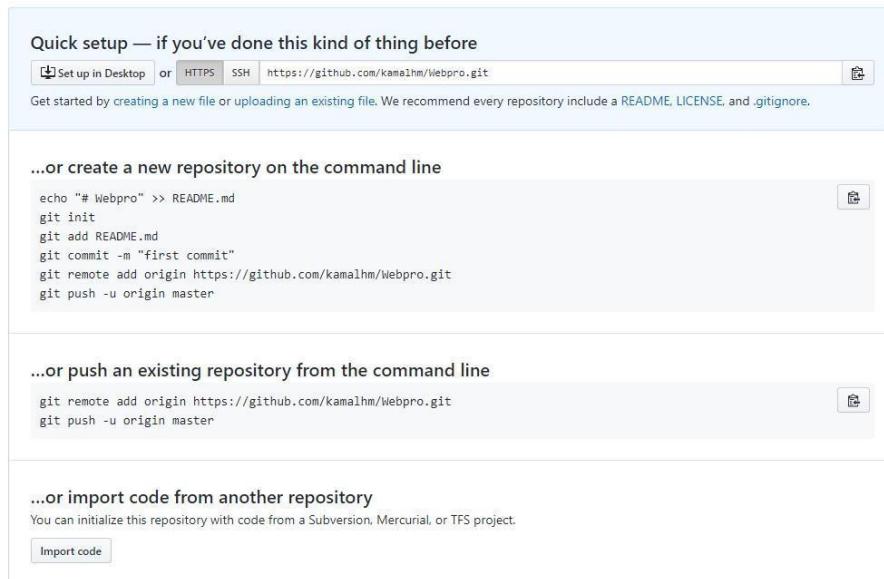
Initialize this repository with a README: This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None

Create repository

Gambar 2.14 Tampilan membuat repositori pada Github

3. Lalu klik tombol “*Create a repository*”.
4. Jika sudah maka akan muncul tampilan seperti gambar 2.15.

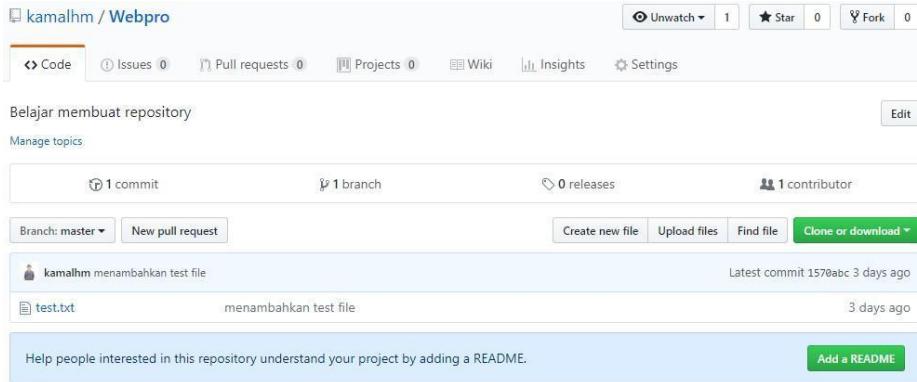


Gambar 2.15 Tampilan setelah repositori dibuat

2.9.4. Menyimpan hasil pekerjaan di repositori online.

Setelah membuat repositori pada Github, sekarang kita dapat menyimpan hasil pekerjaan yang telah kita buat kedalam Github. Untuk melakukan itu, lakukan langkah-langkah berikut:

- Ketikan perintah ini, sesuaikan dengan *username* dan repository Anda:
`git remote add origin https://github.com/usernameanda/namarepo.git`
 Perintah ini akan menambahkan repositori *online* yang ada pada Github kedalam daftar repositori jarak jauh yang ada.
- Untuk mengirimkan data yang ada di komputer kalian ke repositori jarak jauh, gunakan perintah ini:
`git push -u origin master`
- Setelah selesai maka tampilan repositori Anda pada github akan menjadi seperti gambar 2-16

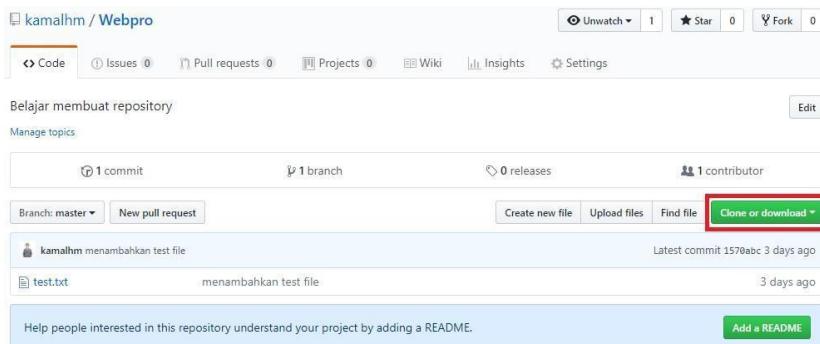


Gambar 2.16 Tampilan repositori setelah di-push pada Github

2.9.5. Clone repositori milik orang lain.

Untuk dapat bekerja sama dengan orang lain, kita dapat melakukan *cloning* repositori orang lain, berikut ini caranya:

1. Buka repositori yang akan di-clone pada Github, lalu klik tombol *clone*.



Gambar 2.17 Tombol clone terlihat pada kotak merah

2. Copy text yang muncul seperti dibawah ini, ini merupakan url dari repositori tujuan yang akan di-clone.



Gambar 2.18 URL repositori target yang akan di-clone

3. Buka *command prompt* dan ketikan perintah ini,
`git clone [url repositori tujuan]`

MODUL 3. CSS – Cascading Style Sheet

Tujuan Praktikum

1. Mahasiswa mampu memahami konsep dan implementasi CSS pada web.
2. Mahasiswa mampu mengenali jenis-jenis dan penggunaan CSS.
3. Mahasiswa mampu memahami konsep dan implementasi Bootstrap pada web.
4. Mahasiswa mengetahui jenis-jenis Bootstrap serta cara penggunaannya.

3.1. Pengenalan CSS

Cascading Style Sheets (CSS) merupakan bahasa yang membantu memperindah tampilan dari laman web yang telah dibangun dengan HTML. CSS mendeskripsikan bagaimana bentuk tampilan elemen HTML seharusnya saat ditampilkan pada laman *browser*. Format penulisan CSS secara umum ditunjukkan pada gambar berikut.



Gambar 3.1. Penulisan CSS

Selector merupakan elemen HTML yang akan ditambahkan CSS kemudian diikuti dengan *declaration block* yang terdiri dari *property* elemen yang akan dirubah beserta *value* dari *property*-nya. Setiap deklarasi *selector* dapat merubah banyak nilai *property* sekaligus dengan dipisahkan dengan titik koma dan untuk semua *declaration block* dari satu *selector* berada di antara kurung kurawal.

3.1.1. Cara Menyisipkan CSS

Terdapat tiga cara untuk menyisipkan atau mendefinisikan CSS ke dalam HTML, antara lain:

1. External Style Sheet

Eksternal Style Sheet merupakan cara menyisipkan atau mendefinisikan CSS ke dalam HTML dengan memanggil file dengan ekstensi **.css** ke dalam *file* HTML. Pemanggilannya diletakkan di antara elemen `<head></head>` dengan menggunakan tag `<link/>`.

```
<head>
    <link rel="stylesheet" type="text/css" href="myStyleSheet.css"> </head>
```

2. Internal Style Sheet

Internal Style Sheet merupakan cara menyisipkan atau mendefinisikan CSS ke dalam HTML dengan menggunakan tag `<style> </style>` pada elemen `<head></head>`. Biasanya digunakan ketika satu laman membutuhkan style CSS yang berbeda dari yang telah dipanggil pada *Eksternal Style Sheet*.

```
<head>
    <style>
        body {
            background-color: blue;
        }
        h1 {
            color: maroon;
            margin-left: 40px;
        }
    </style>
</head>
```

3. Inline Style

Inline Style menyisipkan atau mendefinisikan CSS ke dalam HTML dengan menambahkan atribut **style** pada elemen yang ingin ditambahkan CSS. Biasanya digunakan hanya untuk satu elemen

yang membutuhkan *style* CSS yang berbeda dari yang telah didefinisikan pada *Internal Style* atau *Eksternal Style*.

```
<h1 style="color:lightblue; font-size:30px;">Praktikum Web Programming</h1>
```

3.1.2. Selector

Selector pada CSS digunakan untuk menemukan elemen HTML untuk diberi CSS berdasarkan *selector* yang didefinisikan. Bentuk *selector* ada beberapa antara lain nama elemen HTML, atribut ID dan atribut Class.

```
/*Selector dengan Elemen HTML*/ p {
    text-align: center;      color: red;
}
/*Selector dengan Id Elemen HTML*/
#para1 {
    text-align: center;      color: red;
}
/*Selector dengan Class Elemen HTML*/ p.center {
    text-align: center;      color: red;
}
```

3.2. Font Properties

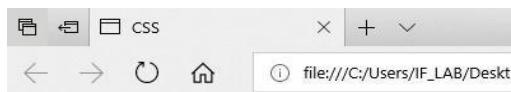
Sebuah laman *web* tentunya tidak lepas oleh penggunaan teks, oleh karena itu memiliki tampilan teks yang tepat sangat diperlukan agar sebuah *web* memiliki tampilan yang baik dan menarik. CSS dapat menangani kebutuhan tampilan teks dengan *font properties*.

Tabel 3.1 Tabel font properties

Font Properties	Keterangan
Font-family	Menentukan jenis font yang digunakan
Font-size	Mengatur ukuran font
Font-style	Mengatur style font (normal, italic, oblique)
Font-weight	Mengatur style font (normal atau bold)

Contoh penerapannya sebagai berikut :

```
p.example {
    font-family : Arial;          font-size : 20px;          color : ligh;
    font-style : italic;
    font-weight : bold;
}
```



Contoh penerapan font properties.

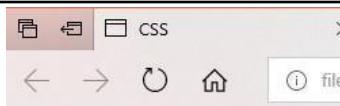
Gambar 3.2 Penerapan font properties

3.3. List Properties

Dalam HTML terdapat elemen yang berguna membuat sebuah *list* menggunakan simbol dan karakter. Tag yang digunakan adalah tag `` atau ``. Tag `` digunakan ketika akan menggunakan *list* dengan penanda berupa simbol atau bisa dikatakan *unordered list*, sedangkan tag `` digunakan ketika akan menggunakan *list* dengan penanda karakter yang memiliki urutan atau bisa dikatakan *ordered list*. Namun di dalam tag tersebut juga harus didefinisikan tag pendukung yaitu `` untuk mendefinisikan elemen-elemen *list* yang akan ditampilkan. Untuk setiap tag *ordered list* atau *unordered list* memiliki satu atribut untuk mendefinisikan tipe simbol atau karakter

yang akan digunakan yaitu atribut ***type***. Contoh penerapan dan tipe masing-masing *tag* sebagai berikut :

```
<h3>List of Property</h3>
<ol type="1">
<li>Indoor
    <ul type="circle">
        <li>Sofa</li>
    </ul>
    <ul type="disc">
        <li>Tanaman Hias</li>
    </ul>
    <ul type="square">
        <li>Lampu Baca</li>
    </ul>
    <ul type="none">
        <li>Rak Buku</li>
    </ul>
</li>
<li>Outdoor
    <ol type="A">
        <li>Payung Pantai</li>
    </ol>
    <ol type="a">
        <li>Ayunan</li>
    </ol>
    <ol type="I">
        <li>Kursi Taman</li>
    </ol>
    <ol type="i">
        <li>Lampu Taman</li>
    </ol>
</li>
</ol>
```



List of Property

1. Indoor
 - Sofa
 - Tanaman Hias
 - Lampu Baca
 - Rak Buku
2. Outdoor
 - A. Payung Pantai
 - a. Ayunan
 - I. Kursi Taman
 - i. Lampu Taman

Gambar 3.3 Tampilan penggunaan list properties

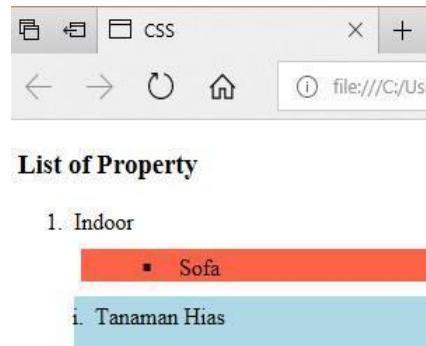
Dengan ditambahkan CSS pada elemen *list*, maka *list* yang ditampilkan dapat lebih menarik, berikut CSS properties untuk elemen *list*.

Tabel 3.2 CSS properties untuk elemen list

Lists Specified Properties	Keterangan
List-style-image	Membuat sebuah gambar menjadi penanda <i>list</i>
List-style-position	Mengatur posisi penanda list di dalam konten atau di luar konten
List-style-type	Mengatur jenis penanda <i>list</i>

Lists General Properties	Keterangan
Background-color	Mengatur warna latar belakang elemen <i>list</i>
Padding	Mengatur ruang jarak elemen konten dengan pembatas pada bagian dalam
Margin	Mengatur ruang jarak elemen konten dengan pembatas pada bagian luar

Contoh penerapannya sebagai berikut:



Gambar 3.4 List properties dengan tambahan CSS

```
ul.listsatu{
    background-color:tomato;    margin: 10px 5px 10px 5px;
list-style-type:lower-alpha;
    list-style-position:inside;
}
ol.listdua{
    background-color:lightblue;      list-style-type:lower-roman;      padding:
5px 5px 15px 15px;
    list-style-position:inside;
}
```

3.4. Alignment of Text

Pengaturan *alignment* pada sebuah teks juga dapat ditangani oleh CSS dengan *properties* pada tabel 3-3.

Tabel 3.3 Alignment of text

Properties	Value	Keterangan
Text-align	Center	Membuat teks menjadi rata tengah
	Left	Membuat teks menjadi rata kiri
	Right	Membuat teks menjadi rata kanan
	Justify	Membuat paragraf menjadi rata kanan dan kiri

Contoh penerapannya pada gambar 3.5 :



Gambar 3.5 Penerapan alignment of text

```

h1 {
    text-align: center;
}
h2 {
    text-align: left;
}
h3 {
    text-align: right;
}

```

3.5. Colors

Jika berbicara desain antar muka *web*, permasalahan tentang warna merupakan salah satu hal yang penting. Pada dasarnya Tag HTML dapat menangani pengaturan warna latar belakang atau teks menggunakan atribut dari HTML sendiri, namun CSS dapat menangani lebih baik dengan menawarkan pengaturan yang lebih lengkap.

Tabel 3.4 CSS Colors

Properties	Keterangan	Value	Color Names (Red, Green, Orange dll.)
Background-color	Mengatur warna latar belakang elemen HTML		RGB Value (R, G, B)
			Hex Value (#FFFF00)
			HSL Value (Hue, Saturation, Light)
Color	Mengatur warna teks elemen HTML		RGBA (dengan <i>Opacity</i>)
			HSLA (dengan <i>Opacity</i>)

Contoh penerapannya sebagai berikut :

```

body{
    background-color : HSL(20%, 40%, 70%);
    color : orange;
}

#teks{
    color : #2F3CDE;
}

/*dengan opacity sebesar 0.5*/
input.text-field{
    background-color : RGBA(32, 55, 122, 0.5);
}

```

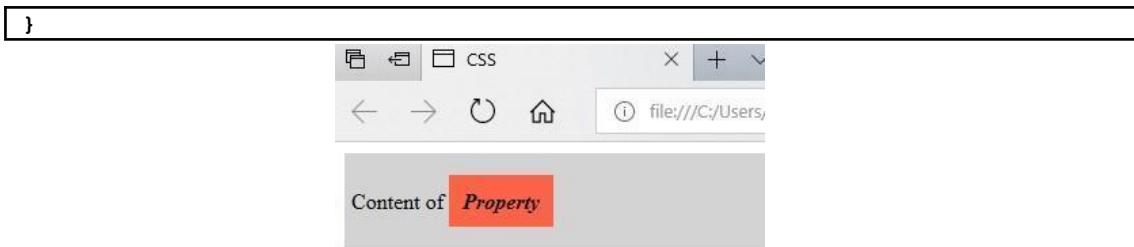
3.6. Span & Div

Span merupakan elemen HTML yang dapat menangani perubahan konten elemen pada satu baris. Tag yang digunakan adalah ``. Sedangkan *Div* merupakan elemen HTML yang digunakan untuk membuat *section* untuk beberapa elemen HTML di dalamnya. Tag yang digunakan yaitu `<div></div>`.

```

<div class="section1">
    <p> Content of <span class="mark"> Property </span> </p> </div>
/*CSS Properties*/
.section1{
    background-color:lightgrey;
    padding: 10px 5px 10px 5px;
}
.mark{
    background-color:tomato;
    font-style:italic;          font-weight:bold;
    padding: 10px 10px 10px 10px;
}

```



Gambar 3.6 Penerapan Span & Div

3.7. Bootstrap

Bootstrap merupakan sebuah *front-end framework* gratis untuk pengembangan antar muka *web* yang lebih cepat dan lebih mudah. Dikembangkan oleh Mark Otto dan Jacom Thornton di Twitter dan dirilis sebagai produk *open source* pada Agustus 2011 di GitHub. Bootstrap mencakup *template* desain berbasis HTML dan CSS untuk tipografi, *form*, *button*, navigasi, *modal*, *image carousells* dan masih banyak lagi, serta terdapat opsional *plugin* JavaScript. Selain itu, Bootstrap memiliki kemampuan untuk membuat desain responsif yang secara otomatis menyesuaikan diri agar terlihat baik di segala perangkat, mulai dari perangkat ponsel hingga *desktop pc*.

3.7.1. Pemasangan Bootstrap

Bootstrap merupakan produk yang mengusung konsep *open source* sehingga untuk pemasangannya dapat dilakukan dengan beberapa cara sebagai berikut:

1. Unduh di <http://getbootstrap.com>, selanjutnya pasang pada *project web* kalian seperti memanggil *External Style Sheet* pada CSS.
2. Memanggil Bootstrap CDN (*Content Delivery Network*), sehingga kita tidak perlu mengunduh dan memasangnya pada laman *website*, hanya memanggil *source* dari Bootstrap. Cara ini membutuhkan koneksi internet untuk menghasilkan perubahan tampilan CSS.

```
<!-- Pemanggilan Bootstrap dengan CDN -->
<!-- CSS -->
<link rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

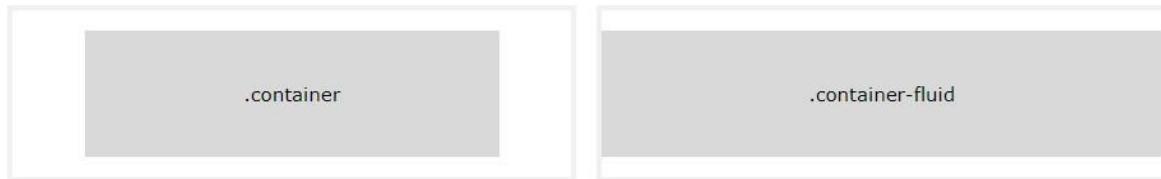
<!-- jQuery library -->
<script
      src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

<!-- JavaScript -->
<script
      src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
```

3.7.2. Bootstrap Container

Bootstrap *container* adalah elemen paling dasar yang dibutuhkan dalam *layouting* menggunakan Bootstrap *Grid*. *Container* berbentuk class CSS yang sisipkan pada elemen HTML *<div>*. Pada gambar 3-7 terdapat dua *class container* pada Bootstrap yang dapat dipilih yaitu:

1. Class **.container** menyediakan *container* yang *responsive* dengan lebar yang tetap.
2. Class **.container-fluid** menyediakan *container* dengan lebar yang penuh mencakup seluruh area pandang.



Gambar 3.7 Class container

3.7.3. Bootstrap Grid

Sistem *grid* pada Bootstrap menggunakan rangkaian *container*, *rows* dan *column* untuk tata letak dan keselarasan elemen atau konten. Dibangun dengan *flexbox* dan sangat responsif terhadap perangkat yang digunakan untuk menampilkan laman *web*. Struktur dasar *grid* pada Bootstrap sebagai berikut.

```
<div class="row">
  <div class="col-*-*"></div>
  <div class="col-*-*"></div>
</div>
<div class="row">
  <div class="col-*-*"></div>
  <div class="col-*-*"></div>
  <div class="col-*-*"></div>
</div>
```

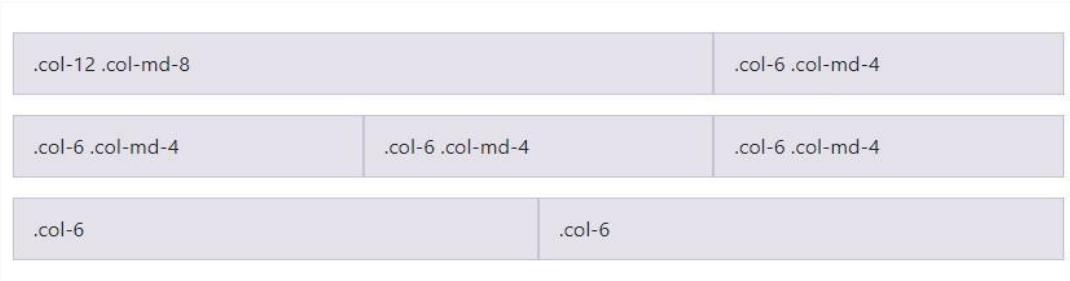
Pertama diawali dengan `<div class="container">`. Kemudian buat sebuah baris sebelum mendeklarasikan sebuah kolom dengan menggunakan `<div class="row">`. Terakhir buat elemen div dengan mendefinisikan class “`col-*-*`”. Tanda * dan # mewakili jenis dan ukuran *column* yang akan digunakan, *value* yang dapat didefinisikan dapat dilihat pada tabel berikut:

Tabel 3.5 Sistem grid Bootstrap

	Extra Small	Small	Medium	Large	Extra Large
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-
# per columns	12				
Nestable	Yes				
Column Ordering	Yes				

Contoh penerapannya sebagai berikut:

```
<div class="container">
<div class="row">
  <div class="col-12 col-md-8">.col-12 .col-md-8</div>
  <div class="col-6 col-md-4">.col-6 .col-md-4</div>
</div>
<div class="row">
  <div class="col-6 col-md-4">.col-6 .col-md-4</div>
  <div class="col-6 col-md-4">.col-6 .col-md-4</div>
  <div class="col-6 col-md-4">.col-6 .col-md-4</div>
</div>
<div class="row">
  <div class="col-6">.col-6</div>
  <div class="col-6">.col-6</div>
</div>
</div>
```



Gambar 3.8 Hasil penerapan Bootstrap grid

3.7.4. Text Style

Bootstrap menyediakan banyak *class* untuk mengatur *style* sebuah teks elemen HTML. Beberapa contohnya antara lain :

Tabel 3.6 Text style

Class	Keterangan
.text-left	Mengatur teks menjadi rata kiri dalam sebuah elemen.
.text-center	Mengatur teks menjadi rata tengah dalam sebuah elemen.
.text-right	Mengatur teks menjadi rata kanan dalam sebuah elemen.
.text-lowercase	Mengatur seluruh teks pada elemen menjadi huruf kecil
.text-uppercase	Mengatur seluruh teks pada elemen menjadi huruf besar
.text-capitalize	Menjadikan huruf pertama besar untuk setiap kata pada sebuah elemen.
.font-weight-bold	Mengatur ketebalan huruf menjadi <i>bold</i>
.font-weight-light	Mengatur ketebalan huruf menjadi <i>light</i>
.font-weight-normal	Mengatur ketebalan huruf menjadi normal
.font-italic	Mengatur gaya teks menjadi miring
.h1 s.d .h6	Mengatur seluruh teks pada sebuah elemen sehingga memiliki tampilan selayaknya tag elemen H1 s.d H6 pada HTML.

3.7.5. Bootstrap Table, Image, & Button

Bootstrap menyediakan *class* untuk pengaturan *style* elemen tabel, gambar dan tombol menjadi lebih menarik.

1. Bootstrap Table

Tabel pada Bootstrap dipanggil dengan *class* **.table** secara *default*, namun ada beberapa *class* tambahan yang dapat didefinisikan pada elemen tabel yang lain berikut dapat dilihat pada Tabel 3.7:

Tabel 3.7 Elemen tabel

Class	Keterangan
.table-dark	Membuat tampilan tabel memiliki latar belakang gelap
.thead-light	Membuat elemen <thead> pada tabel memiliki latar belakang cerah

.thead-dark	Membuat elemen <thead> pada tabel memiliki latar belakang gelap
.table-striped	Membuat tampilan tabel memiliki latar belakang setiap row yang berbeda
.table-bordered	Membuat tampilan tabel sederhana dengan border tipis
.table-hover	Membuat tampilan tabel yang akan berubah warna latar belakang row saat didekati kursor.
.table-sm	Membuat tampilan tabel sederhana yang dengan ukuran <i>padding</i> yang minim

Contoh penerapannya sebagai berikut:

```
<!--Tabel Hover Style -->
<table class="table table-hover">
  <thead>
    <tr>
      <th scope="col">#</th>
      <th scope="col">Nama Lengkap</th>
      <th scope="col">Asal Kota</th>
      <th scope="col">Umur</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th scope="row">1</th>
      <td>Budi Rojadi</td>
      <td>Semarang</td>
      <td>35 th</td>
    </tr>
    <tr>
      <th scope="row">2</th>
      <td>Yulia Santi</td>
      <td>Bekasi</td>
      <td>32</td>
    </tr>
    <tr>
      <th scope="row">3</th>
      <td>Fahri Abdilah</td>
      <td>Medan</td>
      <td>38 th</td>
    </tr>
  </tbody>
</table>
```

#	Nama Lengkap	Asal Kota	Umur
1	Budi Rojadi	Semarang	35 th
2	Yulia Santi	Bekasi	32
3	Fahri Abdilah	Medan	38 th

Gambar 3.9 Hasil penerapan Class.table-hover

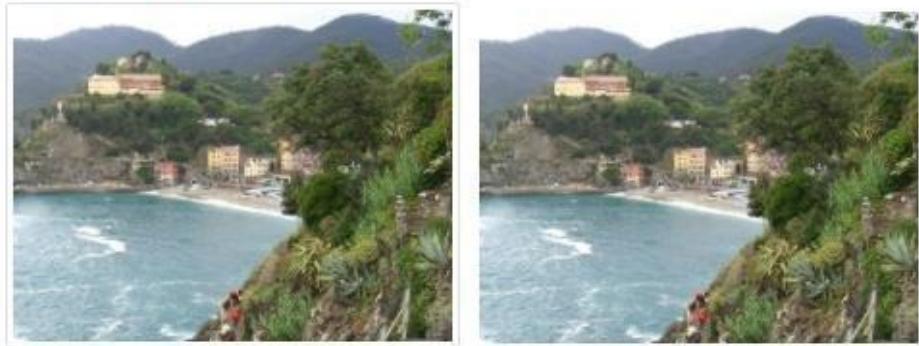
#	Nama Lengkap	Asal Kota	Umur
1	Budi Rojadi	Semarang	35 th
2	Yulia Santi	Bekasi	32

Gambar 3.10 Hasil penerapan Class.thead-dark

2. Bootstrap Image

Bootstrap dapat menangani desain gambar agar responsif pada setiap perangkat yang menampilkan laman *web*. Dengan menambahkan *class* **.img-fluid** pada elemen *tag* pada HTML maka gambar yang didefinisikan pada laman *web* akan memiliki ukuran yang responsif menyesuaikan ukuran layar perangkat. *Class* tersebut mengatur ukuran gambar dengan menyesuaikan ukuran dari *parent element* sebagai wadah atau *container* elemen gambar. Terdapat *class* **.thumbnail** yang berguna menjadikan gambar menjadi berukuran kecil dan sedikit memiliki border disekitarnya dapat dilihat pada gambar 5.

Thumbnail & Fluid



Gambar 3.11 Image class.thumbnail dan class.fluid

```
<div class="container">
  <h2>Thumbnail & Fluid</h2>
  
  
</div>
```

3. Bootstrap Button

Tampilan *button* pada elemen HTML dapat dirubah dengan menambahkan beberapa *class* untuk *button* oleh Bootstrap. Bootstrap membuat tampilan *button* menjadi lebih menarik dan memberikan *user experience* yang baik. *Class* yang digunakan secara *default* adalah **.btn** namun dengan disertai *class* lain seperti berikut untuk memberikan perubahan warna dan ukuran *button*:

Tabel 3.8 Class button

Class	Keterangan
.btn-primary	Membuat tampilan button dengan desain utama
.btn-danger	Membuat tampilan button dengan desain berwarna merah
.btn-success	Membuat tampilan button dengan desain berwarna hijau
.btn-warning	Membuat tampilan button dengan desain berwarna kuning
.btn-info	Membuat tampilan button dengan desain sebuah informasi
.btn-link	Membuat tampilan button dengan desain sebuah <i>hyperlink</i>
.btn-xs	Membuat tampilan button berukuran <i>extra small</i>
.btn-sm	Membuat tampilan button berukuran <i>small</i>

.btn-md	Membuat tampilan button berukuran medium
.btn-lg	Membuat tampilan button berukuran besar
.btn-block	Membuat tampilan button menjadi sebuah blok besar yang mengisi seluruh ruang pada parent

Contoh penerapan *class button*:

```
<div class="container">
  <h4>Button Styles</h4>
  <button type="button" class="btn btn-default btn-md">Default</button>
  <button type="button" class="btn btn-primary btn-lg">Primary</button>
  <button type="button" class="btn btn-success btn-block">Success</button>
  <button type="button" class="btn btn-info btn-xs">Info</button>
  <button type="button" class="btn btn-warning">Warning</button>
  <button type="button" class="btn btn-danger">Danger</button>
  <button type="button" class="btn btn-link">Link</button>
</div>
```

Button Styles



Gambar 3.12 Hasil penerapan *class button*

3.7.6. Bootstrap Form

Bootstrap menyediakan perubahan elemen *form* pada HTML baik pada segi tata letak tampilan atau tampilan antarmuka elemen-elemen dalam *form*. Class **.form-group** didefinisikan untuk setiap *parent* atau *container* untuk elemen *form*, sedangkan class **.form-control** didefinisikan untuk setiap elemen dalam tag *<form>*. Ada tiga *class* yang dapat dipilih untuk mengatur tata letak tampilan *form* yaitu :

1. **Vertical Form**, ini merupakan tampilan *default* saat tag *form* tidak didefinisikan *class*.
2. **Inline Form**, ini merupakan tampilan *form* dengan class **.form-inline** yang membuat *form* memiliki tampilan seluruh elemen dalam satu garis.
3. **Horizontal Form**, ini merupakan tampilan *form* dengan class **.form-inline** yang membuat *form* memiliki tampilan elemen label dengan *input type* dalam satu baris. Pada tampilan ini butuh pendefinisian class **.control-label** untuk setiap elemen *<label>* Contoh :

```
<div class="container">
  <h3>Horizontal form</h3>
  <form class="form-horizontal" action="/action_page.php">
    <div class="form-group">
      <label class="control-label col-sm-2" for="email">Username:</label>
      <div class="col-sm-10">
        <input type="text" class="form-control" id="uname" placeholder="Enter
username" name="uname">
      </div>
    </div>
    <div class="form-group">
      <label class="control-label col-sm-2" for="pwd">Password:</label>
      <div class="col-sm-10">
        <input type="password" class="form-control" id="pwd" placeholder="Enter
password" name="pwd">
      </div>
    </div>
  </form>
</div>
```

```
<div class="form-group">
  <div class="col-sm-offset-2 col-sm-10">
    <button type="submit" class="btn btn-success">Submit</button>
  </div>
</div>
</form>
</div>
```

Contoh Horizontal Form

The image shows a user interface for a login or registration form. It consists of several form groups arranged horizontally. Each group includes a label (e.g., "Username:" or "Password:") followed by an input field. There is also a "Remember me" checkbox and a prominent green "Submit" button at the bottom.

Username:

Password:

Remember me

Submit

Gambar 3.13 Contoh Bootstrap form

MODUL 4. JAVASCRIPT

Tujuan Praktikum
1. Mahasiswa mampu memahami konsep dan implementasi Javascript pada web.
2. Mahasiswa mampu memahami sintaks, elemen, dan fungsi pada Javascript.
3. Mahasiswa mampu memahami konsep dan implementasi jQuery pada web.
4. Mahasiswa mampu memahami sintaks jQuery.

4.1. Pengenalan Javascript

4.1.1. Sejarah Singkat Javascript

Javascript, seperti namanya, merupakan bahasa pemrograman *scripting*. Dan seperti bahasa *scripting* lainnya, Javascript umumnya digunakan hanya untuk program yang tidak terlalu besar, biasanya hanya beberapa ratus baris. Javascript pada umumnya mengontrol program yang berbasis Java. Jadi memang pada dasarnya Javascript tidak dirancang untuk digunakan dalam aplikasi skala besar.

Meskipun dibuat dengan tujuan awal untuk mengendalikan program Java, komunitas Javascript menggunakan bahasa ini untuk tujuan lain, memanipulasi gambar dan isi dari dokumen HTML. Singkatnya, pada akhirnya Javascript digunakan untuk satu tujuan utama, “menghidupkan” dokumen HTML dengan mengubah konten statis menjadi dinamis dan interaktif. Bersamaan dengan perkembangan Internet dan dunia *web* yang pesat, Javascript akhirnya menjadi bahasa utama dan satu-satunya untuk membuat HTML menjadi interaktif di dalam browser.

4.1.2. Prinsip Dasar Javascript

Prinsip dasar yang terdapat pada bahasa pemrograman javascript adalah sebagai berikut.

1. Javascript mendukung paradigma pemrograman imparatif (Javascript dapat menjalankan perintah program baris demi baris, dengan masing-masing baris berisi satu atau lebih perintah), fungsional (struktur dan elemen-elemen dalam program sebagai fungsi matematis yang tidak memiliki keadaan (*state*) dan data yang dapat berubah (*mutable data*)), dan orientasi objek (segala sesuatu yang terlibat dalam program dapat disebut sebagai “objek”).
2. Javascript memiliki model pemrograman fungsional yang sangat ekspresif.
3. Pemrograman berorientasi objek (PBO) pada Javascript memiliki perbedaan dari PBO pada umumnya.
4. Program kompleks pada Javascript umumnya dipandang sebagai program-program kecil yang saling berinteraksi.

4.2. Sintaks Umum pada Javascript

4.2.1. Tipe data dasar

Seperti kebanyakan bahasa pemrograman lainnya, Javascript memiliki beberapa tipe data untuk dimanipulasi. Seluruh nilai yang ada dalam Javascript selalu memiliki tipe data. Tipe data yang dimiliki oleh Javascript adalah sebagai berikut:

- *Number* (bilangan)
- *String* (serangkaian karakter)
- *Boolean* (benar / salah)
- *Object*
- *Function* (fungsi) ○ *Array* ○ *Date*
 - *RegExp* (*regular expression*)
- *Null* (tidak berlaku / kosong)

- Undefined (tidak didefinisikan)

Kebanyakan dari tipe data yang disebutkan di atas sama seperti tipe data sejenis pada bahasa pemrograman lainnya. Misalnya, sebuah boolean terdiri dari dua nilai saja, yaitu true dan false.

4.2.2. Variabel

Seperti pada bahasa pemrograman lainnya, variabel dalam Javascript merupakan sebuah tempat untuk menyimpan data sementara. Variabel dibuat dengan kata kunci var pada Javascript.

```
var a;           // a berisi undefined
var nama = "Budi"; // nama berisi "Budi"
```

Nilai yang ada di dalam variabel dapat diganti dengan mengisikan nilai baru, dan bahkan dapat diganti tipe datanya juga.

```
nama = "Anton"; // nama sekarang berisi string "Anton"
nama = 1;        // nama sekarang berisi integer 1
```

Walaupun kemampuan untuk menggantikan tipe data ini sangat memudahkan kita dalam mengembangkan aplikasi, fitur ini harus digunakan dengan sangat hati-hati. Perubahan tipe data yang tidak diperkirakan dengan baik dapat menyebabkan berbagai kesalahan (*error*) pada program, misalnya jika kita mencoba mengakses method `charAt()` (fungsi yang mengembalikan nilai *char* pada indeks sebuah *string*) setelah mengubah tipe pada contoh di atas.

4.2.3. Array

Array merupakan sebuah tipe data yang digunakan untuk menampung banyak tipe data lainnya. Berbeda dengan tipe data *object*, *array* pada Javascript merupakan sebuah tipe khusus. Walaupun memiliki *method* dan properti, *array* bukanlah objek, melainkan sebuah tipe yang “mirip objek”. Pembuatan *array* dalam Javascript dilakukan dengan menggunakan kurung siku ([]):

```
var data = ["satu", 2, true];
```

Elemen *array* pada Javascript tidak harus memiliki tipe data yang sama seperti contoh diatas. Selain itu Javascript juga mendukung untuk membuat *array* di dalam array yang biasa dikenal dengan *array* dua dimensi seperti contoh dibawah ini.

```
var arr2 = [[ "satu", "dua"], [ "tiga", "empat"]];
```

Pengaksesan elemen dalam *array* dilakukan dengan menggunakan kurung siku. Nilai yang kita berikan dalam kurung siku adalah urutan elemen penulisan *array* (indeks), yang dimulai dari nilai 0. Jika indeks yang diakses tidak ada, maka kita akan mendapatkan nilai *undefined*.

```
data[2];      // mengembalikan true
arr2[0][1];  // mengembalikan "dua"
data[10];     // mengembalikan undefined
```

Sebagai sebuah objek khusus, *array* juga memiliki *method* dan properti. Beberapa *method* dan properti yang populer misalnya *length*, *pop()*, dan *push()*.

```
var data = [ "a", "b", "c"]; data.length //mengembalikan 3
data.push("d"); // mengembalikan 4, data menjadi [ "a", "b", "c", "d"]
data.pop(); // mengembalikan "d", data menjadi [ "a", "b", "c"]
```

4.2.4. Pengendalian Struktur

Javascript memiliki perintah-perintah pengendalian struktur (*control structure*) yang sama dengan bahasa dalam keluarga C. Perintah `if` dan `else` digunakan untuk percabangan, sementara perintah `for`, `for-in`, `while`, dan `do-while` digunakan untuk perulangan.

Percabangan pada Javascript bisa dikatakan sama persis dengan C atau Java:

```

var gelar;
var pendidikan = "S2"; if (pendidikan === "S1") {     gelar = "Sarjana";
} else if (pendidikan === "S2") {     gelar = "Master";
} else if (pendidikan === "S3") {     gelar = "Doktor";
} else {     gelar = "Tidak Diketahui";
}
gelar; // gelar berisi "Master"

```

Satu hal yang perlu diperhatikan, tiga buah sama dengan (==) digunakan pada operasi perbandingan di Javascript. Javascript mendukung dua operator perbandingan sama dengan, yaitu == dan ===. Perbedaan utamanya adalah == mengubah tipe data yang dicek menjadi nilai terdekat, sementara === memastikan tipe data dari dua nilai yang dibandingkan sama. Untuk mendapatkan nilai perbandingan paling akurat, selalu gunakan === ketika mengecek nilai.

Sama seperti `if`, perulangan `do`, `do-while`, dan `for` memiliki cara pemakaian yang dapat dikatakan sama persis dengan C atau Java:

```

while (true) {
    // tak pernah berhenti
} var input; do {     input = get_input();
} while (inputIsNotValid(input)) for (var i = 0; i < 5; i++) {
    // berulang sebanyak 5 kali }

```

4.3. Object Orientation pada Javascript

Javascript memiliki dua jenis tipe data utama, yaitu tipe data dasar dan objek. Tipe data dasar pada Javascript adalah angka (*numbers*), rentetan karakter (*strings*), boolean (*true* dan *false*), *null*, dan *undefined*. Nilai-nilai selain tipe data dasar secara otomatis dianggap sebagai objek. Objek dalam Javascript didefinisikan sebagai *mutable properties collection*, yang artinya adalah sekumpulan properti (ciri khas) yang dapat berubah nilainya. Karena nilai-nilai selain tipe data dasar merupakan objek, maka pada Javascript sebuah *Array* adalah objek. Fungsi adalah objek dan *Regular expression* juga merupakan objek.

4.3.1 Pembuatan Object pada Javascript

Notasi pembuatan objek pada Javascript sangat sederhana, yaitu sepasang kurung kurawal yang membungkus properti. Notasi pembuatan objek ini dikenal dengan nama *object literal*. *Object literal* dapat digunakan kapanpun pada ekspresi Javascript yang valid:

```

var objek_kosong = {}; var mobil = {
    "warna-badan": "merah",
    "nomor-polisi": "BK1234AB" };

```

Nama properti dari sebuah objek harus berupa *string*, dan boleh berisi *string* kosong (""). Jika merupakan nama Javascript yang legal, kita tidak memerlukan petik ganda pada nama properti. Petik ganda seperti pada contoh ("warna-badan") hanya diperlukan untuk nama Javascript ilegal atau kata kunci seperti "if" atau "var". Misalnya, "nomor-polisi" memerlukan tanda petik, sementara nomor_polisi tidak. Contoh lain, variasi tidak memerlukan tanda petik, sementara "var" perlu.

Sebuah objek dapat menyimpan banyak properti, dan setiap properti dipisahkan dengan tanda koma (,). Jika ada banyak properti, nilai dari properti pada setiap objek boleh berbeda-beda:

```

var jadwal = {     platform: 34,     telah_berangkat: false,     tujuan: "Medan",
asal: "Jakarta" };

```

Karena dapat diisi dengan nilai apapun (termasuk objek), maka kita dapat membuat objek yang mengandung objek lain (*nested object*; objek bersarang) seperti berikut:

```

var jadwal = {     platform: 34,     telah_berangkat: false,     asal: {
kode_kota: "MDN",             nama_kota: "Medan",             waktu: "2013-12-29 14:00"
},     tujuan: {             kode_kota: "JKT",             nama_kota: "Jakarta",
waktu: "2013-12-29 17.30" } }

```

```
};
```

4.3.2. Akses Nilai Property

Akses nilai properti dapat dilakukan dengan dua cara, yaitu.

1. Penggunaan kurung siku ([]) setelah nama objek. Kurung siku kemudian diisi dengan nama properti, yang harus berupa *string*. Cara ini biasanya digunakan untuk nama properti yang adalah nama ilegal atau kata kunci Javascript.
2. Penggunaan tanda titik (.) setelah nama objek diikuti dengan nama properti. Notasi ini merupakan notasi yang umum digunakan pada bahasa pemrograman lainnya. Notasi ini tidak dapat digunakan untuk nama ilegal atau kata kunci Javascript.

Contoh penggunaan kedua cara pemanggilan di atas adalah sebagai berikut:

```
mobil["warna-badan"] // Hasil: "merah"  
jadwal.platform // Hasil: 34
```

Sebagai bahasa dinamis, Javascript tidak akan melemparkan pesan kesalahan jika kita mengakses properti yang tidak ada dalam objek. Kita akan menerima nilai *undefined* jika mengakses properti yang tidak ada:

```
jadwal.nomor_kursi // Hasil: undefined mobil  
["jumlah-roda"] // Hasil: undefined
```

Pengaksesan properti pada Javascript juga dapat digunakan secara dinamis untuk mengubah nilai dari properti tersebut. Perubahan nilai properti juga dapat dilakukan untuk properti yang bahkan tidak ada pada objek tersebut:

```
mobil["jumlah-roda"] = 4;  
mobil.bahan_bakar = "Bensin";
```

4.3.3. Prototype pada Javascript

Pada Javascript yang mengimplementasikan PBO kita tidak lagi perlu menuliskan kelas, dan langsung melakukan penurunan terhadap objek. Misalkan kita memiliki objek mobil yang sederhana seperti berikut:

```
var mobil = { nama: "Mobil", jumlahBan: 4 };
```

Kita dapat langsung menurunkan objek tersebut dengan menggunakan fungsi `Object.create` seperti berikut:

```
var truk = Object.create(mobil);  
// truk.nama === "Mobil"  
// truk.jumlahBan === 4
```

4.4. Function pada Javascript

Sebuah fungsi membungkus satu atau banyak perintah. Setiap kali fungsi dipanggil, maka perintah-perintah yang ada di dalam fungsi tersebut dijalankan. Secara umum fungsi digunakan untuk penggunaan kembali kode (*code reuse*) dan penyimpanan informasi (*information hiding*). Implementasi fungsi kelas pertama juga memungkinkan penggunaan fungsi sebagai unit-unit yang dapat dikombinasikan, seperti layaknya sebuah lego. Dukungan terhadap pemrograman berorientasi objek juga berarti fungsi dapat digunakan untuk memberikan perilaku tertentu dari sebuah objek.

4.4.1 Pembuatan Fungsi pada Javascript

Sebuah fungsi pada Javascript dibuat dengan cara seperti berikut:

```
function tambah(a, b) { hasil = a + b; return hasil; }
```

Cara penulisan fungsi seperti diatas dikenal dengan nama *function declaration*, atau deklarasi fungsi. Terdapat empat komponen yang membangun fungsi di atas, yaitu:

1. Kata kunci *function*, yang memberitahu Javascript bahwa akan dibuat sebuah fungsi.
2. Nama fungsi, dalam contoh di atas adalah tambah. Dengan memberikan sebuah fungsi nama maka pemanggilan fungsi dapat dirujuk dengan nama tersebut. Harus diingat bahwa nama fungsi bersifat opsional, yang berarti **fungsi pada Javascript tidak harus diberi nama**.
3. Daftar parameter fungsi, yaitu a, b pada contoh di atas. Daftar parameter ini selalu dikelilingi oleh tanda kurung (()). Parameter boleh kosong, tetapi tanda kurung wajib tetap dituliskan. Parameter fungsi akan secara otomatis didefinisikan menjadi variabel yang hanya bisa dipakai di dalam fungsi. Variabel pada parameter ini diisi dengan nilai yang dikirimkan kepada fungsi secara otomatis.
4. Sekumpulan perintah yang ada di dalam kurung kurawal ({}). Perintah-perintah ini dikenal dengan nama badan fungsi. Badan fungsi dieksekusi secara berurut ketika fungsi dijalankan.

Penulisan deklarasi fungsi (*function declaration*) seperti di atas merupakan cara penulisan fungsi yang umumnya kita gunakan pada bahasa pemrograman imperatif dan berorientasi objek. Tetapi selain deklarasi fungsi Javascript juga mendukung cara penulisan fungsi lain, yaitu dengan memanfaatkan ekspresi fungsi (*function expression*). Ekspresi fungsi merupakan cara pembuatan fungsi yang memperbolehkan menuliskan fungsi tanpa nama. Fungsi yang dibuat tanpa nama dikenal dengan sebutan fungsi anonim atau fungsi lambda. Berikut adalah cara membuat fungsi dengan ekspresi fungsi:

```
var tambah = function (a, b) { hasil = a + b;
return hasil;
};
```

Terdapat hanya sedikit perbedaan antara ekspresi fungsi dan deklarasi fungsi:

1. Penamaan fungsi. Pada deklarasi fungsi, nama fungsi langsung diberikan sesuai dengan sintaks yang disediakan Javascript. Penggunaan ekspresi fungsi pada dasarnya menyimpan sebuah fungsi anonim ke dalam variabel dan nama fungsi adalah nama variabel yang dibuat. Perlu diingat juga bahwa pada dasarnya ekspresi fungsi adalah fungsi anonim. Penyimpanan ke dalam variabel hanya diperlukan karena kita akan memanggil fungsi nantinya.
2. Ekspresi fungsi dapat dipandang sebagai sebuah ekspresi atau perintah standar bagi Javascript, sama seperti penulisan kode var i = 0. Deklarasi fungsi merupakan konstruksi khusus untuk membuat fungsi. Hal ini berarti pada akhir dari ekspresi fungsi kita harus menambahkan, sementara pada deklarasi fungsi hal tersebut tidak penting.

4.4.2. Pemanggilan Fungsi

Sebuah fungsi dapat dipanggil untuk menjalankan seluruh kode yang ada di dalam fungsi tersebut, sesuai dengan parameter yang kita berikan. Pemanggilan fungsi dilakukan dengan cara menuliskan nama fungsi tersebut, kemudian mengisikan argumen yang ada di dalam tanda kurung.

Misalkan fungsi tambah yang kita buat pada bagian sebelumnya:

```
var tambah = function (a, b) { hasil = a + b;
return hasil;
};
```

dapat dipanggil seperti berikut:

```
tambah(3, 5);
```

Yang terjadi pada kode di atas adalah nilai a dan b masing-masing digantikan dengan 3 dan 5. Seperti yang dapat dilihat, hal ini berarti pengisian argumen pada saat pemanggilan fungsi harus berurut sesuai dengan deklarasi fungsi.

Sama seperti sebuah variabel, fungsi juga mengembalikan nilai ketika dipanggil. Dalam kasus di atas, tambah(3, 5) akan mengembalikan nilai 8. Nilai ini tentunya dapat disimpan ke dalam variabel baru, atau bahkan dikirimkan sebagai sebuah argumen ke fungsi lain lagi:

```

var simpan = tambah(3, 5); // simpan === 8 tambah(simpan, 2);           //
mengembalikan 10 tambah(tambah(3, 5), 2)      // juga mengembalikan 10
tambah(tambah(2, 3), 4) // mengembalikan 9

```

Fungsi akan mengembalikan nilai ketika kata kunci *return* ditemukan. Pengembalian nilai fungsi dapat dilakukan kapanpun, dan fungsi akan segera berhenti ketika kata kunci *return* ditemukan. Berikut adalah contoh kode yang memberikan gambaran tentang pengembalian nilai fungsi:

```

var naikkan = function (n) { var hasil = n + 10; return hasil;
// kode di bawah tidak dijalankan
lagi hasil = hasil * 100;
} naikkan(10); // mengembalikan 20 naikkan(25); // mengembalikan 35

```

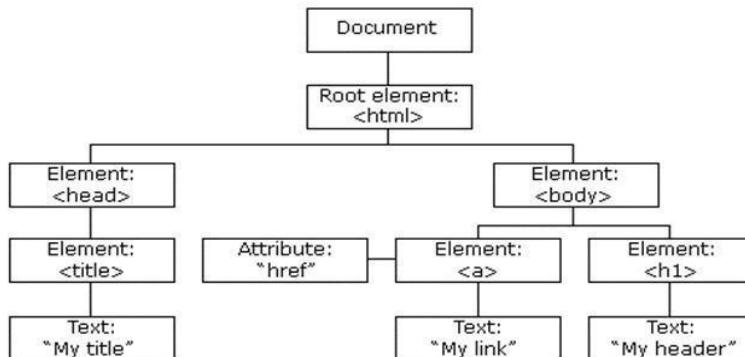
Sebuah ekspresi dapat juga diberikan langsung kepada keyword *return*, dan ekspresi tersebut akan dijalankan sebelum nilai dikembalikan. Hal ini berarti fungsi tambah maupun naikkan yang sebelumnya bisa disederhanakan dengan tidak lagi menyimpan nilai di variabel hasil terlebih dahulu:

```

var naikkan = function (n) {
return n + 10;
} var tambah = function (a, b)
{
    return a + b;
} tambah(4, 4); // mengembalikan 8
naikkan(10); // mengembalikan 20
tambah(naikkan(5), 7); // mengembalikan 22

```

4.5. DOM Manipulation



Gambar 4.1 Struktur DOM

Salah satu aspek terpenting pada JavaScript maupun jQuery adalah manipulasi DOM. DOM atau yang disebut *Document Object Model* adalah mekanisme untuk merepresentasikan dan berinteraksi dengan dokumen HTML, XHTML atau XML. DOM dapat memanipulasi dokumen melalui bahasa pemrograman, dimana biasanya pada *browser* selalu melalui JavaScript. Navigasi dan manipulasi DOM menggunakan JavaScript standar sebenarnya kurang efektif namun kenyataannya jQuery bersama DOM memiliki banyak *method* yang saling berhubungan sehingga membuatnya lebih mudah. Sintaks DOM bekerja diantara tag *<script></script>* pada HTML, beberapa sintaks DOM antara lain :

Tabel 4.1 Daftar sintaks DOM

Sintaks	Keterangan
<code>document.getElementById(id)</code>	Mencari elemen HTML berdasarkan atribut ID
<code>document.getElementsByTagName(tag_name)</code>	Mencari elemen HTML berdasarkan nama tag elemen HTML
<code>document.getElementsByClassName(name)</code>	Mencari elemen HTML berdasarkan atribut class
<code>element.style.property = new style</code>	Merubah nilai atribut style dan property CSS suatu elemen HTML
<code>element.setAttribute(attribute, value)</code>	Merubah nilai atribut elemen HTML

4.6. jQuery

jQuery adalah sebuah library Javascript yang dibuat oleh John Resig pada tahun 2006. jQuery memungkinkan manipulasi dokumen HTML dilakukan hanya dalam beberapa baris *code*. Beberapa fitur utama yang terdapat pada jQuery adalah:

- **DOM manipulation** – jQuery memungkinkan untuk memodifikasi DOM (*Document Object Model*) menggunakan *source selector* yang disebut dengan **Sizzle**.
- **Event Handling** – jQuery dapat menangani sebuah aksi pada dokumen HTML seperti saat pengguna melakukan *click* pada sebuah objek.
- **Ajax Support** – jQuery dapat memfasilitasi pembuatan *website* menggunakan teknologi AJAX.
- **Animations** – pada jQuery terdapat *build-in* animasi yang dapat digunakan pada halaman *web*.
- **Lightweight** – ukuran *file* jQuery sangat ringan yaitu sekitar 19KB.

jQuery dapat dengan mudah digunakan pada sebuah situs *web* dengan berbagai cara.

1. Instalasi Lokal

- Kunjungi link <https://jquery.com/download/> untuk mengunduh *library* jQuery.
- Letakkan *library* yang sudah diunduh pada satu folder yang sama dengan *file* HTML dengan kode berikut.

```
<html>
  <head>
    <title>The jQuery Example</title>
    <script type = "text/javascript" src = "jquery-3.2.1.min.js">
    </script>

    <script type = "text/javascript">
$(document).ready(function() {                               document.write("Hello,
  World!");
  });
    </script>
  </head>
  <body>
    <h1>Hello</h1>
  </body>
</html>
```

Note: pastikan atribut `src` memiliki nilai yang sama dengan nama *file* *library* jQuery.

- Buka *file* HTML tersebut menggunakan *web browser* seperti Mozilla atau Chrome. Dan hasil yang didapatkan adalah sebuah teks “Hello World” seperti yang ditulis pada bagian `document.write()`.

2. Menggunakan CDN (*Content Delivery Network*)

- Buka file HTML tersebut menggunakan *web browser* seperti Mozilla atau Chrome. Dan hasil yang didapatkan adalah sebuah teks “Hello World” seperti yang ditulis pada bagian `document.write()`.

<https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js>

- Buka *file* HTML menggunakan *web browser* dan hasil yang ditampilkan akan sama dengan cara instalasi lokal.

jQuery memiliki beberapa fungsionalitas yang langsung dapat digunakan seperti:

4.6.1. Efek hide/show

Contoh code:

```
<!DOCTYPE html>
<html>
<head> <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
> <script>
$(document).ready(function() {
    $("#hide").click(function() {
        $("p").hide();
    });
    $("#show").click(function() {
        $("p").show();
    });
});
</script>
</head>
<body>

<p>If you click on the "Hide" button, I will disappear.</p>
<button id="hide">Hide</button>
<button id="show">Show</button>

</body>
</html>
```

Penjelasan code:

- Pada baris 4-5, dilakukan pemanggilan *library* jQuery menggunakan CDN.
- Pada baris 6-15, fungsi jQuery *hide/show* diaplikasikan pada tag html <p>. Apabila *button* dengan id 'hide' diklik, maka semua konten pada tag <p> akan disembunyikan. Selain itu, apabila *button* dengan id 'show' diklik, maka semua konten pada tag <p> akan dimunculkan.

4.6.2. Efek animasi

Contoh code:

```
<!DOCTYPE html>
<html>
<head> <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
> <script>
$(document).ready(function() {
    $("button").click(function() {           $("div").animate({
height: 'toggle'
    });
});
});
</script>
```

```

</head>
<body>

<p>Click the button multiple times to toggle the animation.</p>
<button>Start Animation</button>
<p>By default, all HTML elements have a static position, and cannot be moved.
To manipulate the position, remember to first set the CSS position property of
the element to relative, fixed, or absolute!</p>
<div
style="background:#98bf21;height:100px;width:100px;position:absolute;"></div>

</body>
</html>

```

Penjelasan code:

- Pada baris 6-15, fungsi jQuery menganimasikan “toggle” pada tag `<div>` yang terdapat pada `body` HTML berdasarkan tinggi elemen.

4.6.3. DOM Manipulation

Jenis Jenis DOM Manipulation pada JQuery diantaranya :

Tabel 4.2 Daftar Fungsi DOM Manipulation pada JQuery

Fungsi	Keterangan
<code>append('content')</code>	menambahkan konten pada element yang dipilih oleh selector
<code>before('content')</code>	menambahkan konten baru sebelum element yang dipilih selector
<code>after('content')</code>	menambahkan konten baru setelah element yang dipilih oleh selector
<code>prepend('content')</code>	menambahkan konten di awal element yang telah dipilih oleh selector
<code>remove()</code>	menghapus elemen yang dipilih oleh selector
<code>replaceAll('expression')</code>	mengganti elemen yang dipilih dengan elemen baru
<code>wrap('tag')</code>	membungkus struktur HTML sekitar dengan elemen yang dipilih oleh selector

contoh penggunaan DOM Manipulation:

Tabel 4.3 Contoh DOM Manipulation

Sebelum DOM Manipulation	JQuery Syntax	Setelah DOM Manipulation
<code><div id="div1">div 1</div></code>	<code>\$('#div1').after('<div>New </div>');</code>	<code><div id="div1">div 1</div> <div> New </div></code>
<code><div id="div1">div 1</div></code>	<code>\$('#div1').before('<div>New </div>');</code>	<code><div> New </div> <div id="div1">div 1</div></code>

<p>Hello </p>	<code>\$('p').append('World!');</code>	<p>Hello World! </p>
<div> <label>label</label> <td><code>\$('div').prepend('<p>paragraph </p>');</code></td> <td><div> <p> paragraph </p> <label>label</label>
<="" div><="" td=""/></td>	<code>\$('div').prepend('<p>paragraph </p>');</code>	<div> <p> paragraph </p> <label>label</label>
<div> <label>label</label> <td><code>\$('label').remove();</code></td> <td><div> label
<="" div><="" td=""/></td>	<code>\$('label').remove();</code>	<div> label
<div> <p>paragraph 1</p> <p>paragraph 2</p>	<code>\$('<h1> ini head </h1>').replaceAll('p');</code>	<div> <h1> ini head </h1>
 ini span 	<code>\$('span').wrap('<p></p>') ;</code>	<p> ini span </p>

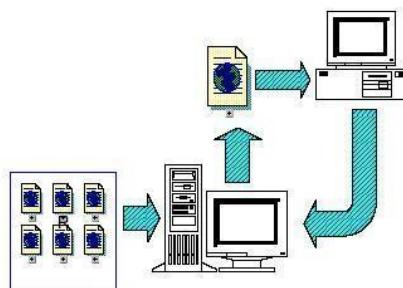
MODUL 5. PHP

Tujuan Praktikum

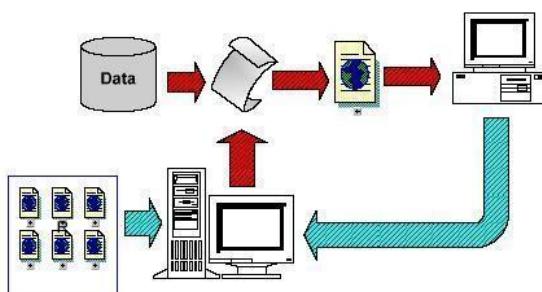
1. Mahasiswa mampu memahami konsep dan implementasi PHP pada web.
2. Mahasiswa mampu memahami sintaks, elemen, dan fungsi pada PHP.

5.1. Web Server dan Server Side Scripting

Web Server merupakan sebuah perangkat lunak dalam *server* yang berfungsi menerima permintaan (*request*) berupa halaman *web* melalui HTTP atau HTTPS dari *client* yang dikenal dengan *web browser* dan mengirimkan kembali (*response*) hasilnya dalam bentuk halaman-halaman *web* yang umumnya berbentuk dokumen HTML.



Gambar 5.1 Arsitektur web standar



Gambar 5.2. Arsitektur web dinamis

Beberapa *web server* yang banyak digunakan antara lain seperti berikut:

1. Apache Web Server (<https://httpd.apache.org/>)
2. Internet Information Service, IIS (<https://www.iis.net/>)
3. Xitami Web Server
4. Sun Java System Web Server

Server Side Scripting merupakan sebuah teknologi *scripting* atau pemrograman *web* dimana *script* (program) dikompilasi atau diterjemahkan di *server*. Dengan *server side scripting*, memungkinkan untuk menghasilkan halaman *web* yang dinamis.

Beberapa contoh *Server Side Scripting (Programming)* :

1. ASP (Active Server Page) dan ASP.NET
2. ColdFusion (<http://www.adobe.com/products/coldfusion-family.html>)
3. Java Server Pages (<http://www.oracle.com/technetwork/java/javaee/jsp/index.html>)
4. Perl (<https://www.perl.org/>)
5. Python (<https://www.python.org/>)
6. PHP (<http://www.php.net/>)

Keistimewaan PHP sebagai bahasa pemrograman berbasis web adalah :

1. Cepat
2. Free
3. Mudah dipelajari
4. Multi-platform
5. Dukungan technical support
6. Banyaknya komunitas PHP
7. Aman

5.2. Pengenalan PHP

Merupakan singkatan rekursif dari **PHP** : Hypertext Preprocessor. Pertama kali diciptakan oleh **Rasmus Lerdorf** pada tahun 1994. PHP sendiri harus ditulis diantara tag :

- <? dan ?>
- <?php dan ?>
- <script language ="php"> dan </script>
- <% dan %>

Setiap satu *statement* (perintah) biasanya diakhiri dengan titik-koma (;). PHP juga **case sensitive** untuk nama *identifier* yang dibuat oleh *user* sedangkan *identifier* bawaan dari PHP **tidak case sensitive**. Contoh program yang ditulis dengan bahasa PHP

```
<?php echo "Hello World!"; ?>
```

Simpan file tersebut dengan nama **hello.php** pada direktori **htdocs** yang ada di folder XAMPP. Kemudian, jalankan pada *browser* dengan mengetikkan alamat <http://localhost/hello.php> . Hasilnya akan muncul di *web browser* seperti berikut:



Gambar 5.3 Tampilan ketika menjalankan file PHP pada browser

5.3. Variabel

Variabel digunakan untuk menyimpan sebuah *value* (nilai), data atau informasi. Nama variabel pada PHP diawali dengan tanda \$. Panjang dari suatu variabel tidak terbatas dan variabel tidak perlu dideklarasi terlebih dahulu sebelumnya. Setelah tanda \$, dapat diawali dengan huruf atau *under-score* (_). Karakter berikutnya bisa terdiri dari huruf, angka dan atau karakter tertentu yang diperbolehkan (karakter ASCII dari 127 – 255).

Variabel pada PHP bersifat *case sensitive* artinya besar kecilnya suatu karakter berpengaruh pada variabel tersebut. Suatu karakter pada PHP tidak boleh mengandung spasi.

Berikut adalah contoh penggunaan variabel pada PHP :

```
<?php  
    $nim = "1301165454";  
    $nama = "Baharudin";  
  
    echo "NIM : " . $nim;  
    echo "Nama : " . $nama;  
?>
```

Pada PHP, tipe data dari suatu variabel tidak didefinisikan langsung oleh *programmer*, akan tetapi secara otomatis akan ditentukan oleh interpreter PHP. Namun demikian, PHP mendukung 8 (delapan) buah tipe data primitif, yaitu :

1. Boolean
2. Integer
3. Float
4. String
5. Array
6. Object
7. Resource
8. NULL

5.4. Konstanta

Konstanta merupakan variabel konstan yang nilainya tidak berubah-ubah. Untuk mendefinisikan konstanta pada PHP, dapat menggunakan fungsi `define()` yang telah tersedia pada PHP. Berikut adalah contohnya :

```
<?php
define("NAMA" , "Baharuddin"); define("NIM" "1301165454");
echo "Nama : " . NAMA; echo "NIM : " . NIM;
?>
```

5.5. Operator dalam PHP

Ada beberapa jenis operator pada PHP, yaitu :

Tabel 5.1 Operator pada PHP

Jenis Operator	Operator	Contoh	Keterangan
Arimatika	+	\$a + \$b	Pertambahan
	-	\$a - \$b	Pengurangan
	*	\$a * \$b	Perkalian
	/	\$a / \$b	Pembagian
	%	\$a % \$b	Modulus, sisa hasil bagi
Penugasan	=	\$a = 4	Variabel \$a akan diisi oleh 4
Bitwise	&	\$a & \$b	Bitwise AND
		\$a \$b	Bitwise OR
	^	\$a ^ \$b	Bitwise XOR
	~	~\$a	Bitwise NOT
	<<	\$a << \$b	Shift Left
	>>	\$a >> \$b	Shift Right
Perbandingan	==	\$a == \$b	Sama dengan
	====	\$a === \$b	Identik
	!=	\$a != \$b	Tidak sama dengan
	<>	\$a <> \$b	Tidak sama dengan
	!==	\$a !== \$b	Tidak identik
	<	\$a < \$b	Kurang dari

	>	$\$a > \b	Lebih dari
	\leq	$\$a \leq \b	Kurang dari sama dengan
	\geq	$\$a \geq \b	Lebih dari sama dengan
Logika	and	$\$a \text{ and } \b	TRUE jika $\$a$ dan $\$b$ TRUE
	$\&\&$	$\$a \&\& \b	TRUE jika $\$a$ dan $\$b$ TRUE
	or	$\$a \text{ or } \b	TRUE jika $\$a$ atau $\$b$ TRUE
	$\ $	$\$a \ \b	TRUE jika $\$a$ atau $\$b$ TRUE
	xor	$\$a \text{ xor } \b	TRUE jika $\$a$ atau $\$b$ TRUE, tapi tidak keduanya
	!	$!\$b$	TRUE jika $\$b$ FALSE
<i>String</i>	.	$\$a . \b	Penggabungan string $\$a$ dan $\$b$

5.6. Struktur Kondisi

Struktur kondisi pada PHP sama halnya dengan bahasa pemrograman lainnya seperti Java. Berikut adalah contoh penulisan struktur kondisi **if-then** pada PHP:

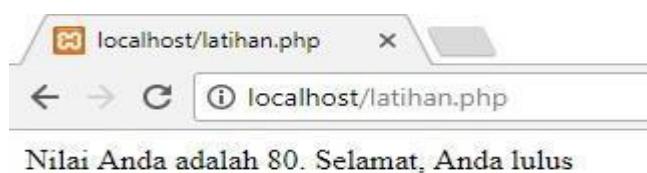
```
if (kondisi) {                      statement-jika-kondisi-TRUE;
} else {                            Statement-jika-kondisi-FALSE; }
```

Selain struktur kondisi **if-then**, terdapat pula struktur kondisi **switch-case** seperti berikut:

```
switch ($var) {
    case '1' : statement-1; break;           case '2' : statement-2;
break;                                .
. .
}
```

Berikut adalah contoh ketika *statement* kondisi **if-then** dijalankan :

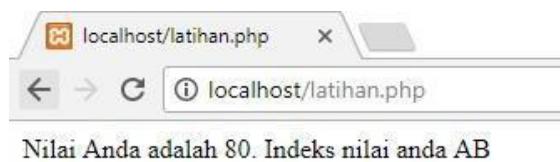
```
$nilai = 80; if ($nilai > 50) {           echo "Nilai Anda adalah " . $nilai . ".
Selamat, Anda lulus";
} else {
    echo "Nilai Anda adalah " . $nilai . ". Maaf, Anda tidak lulus"; }
```



Gambar 5.4 Hasil penggunaan IF-THEN

Dan berikut ini adalah contoh ketika *statement switch-case* dijalankan :

```
$nilai = 80; switch ($nilai) {    case ($nilai > 50 && $nilai <= 60) :  
echo "Nilai Anda adalah " . $nilai . ". Indeks nilai anda  
C"; break;  
case ($nilai > 60 && $nilai <= 70) : echo "Nilai Anda adalah " . $nilai .  
". Indeks nilai anda  
BC"; break;  
case ($nilai > 70 && $nilai <= 75) : echo "Nilai Anda adalah " . $nilai .  
. Indeks nilai anda  
B"; break;  
case ($nilai > 75 && $nilai <= 80) : echo "Nilai Anda adalah " . $nilai .  
. Indeks nilai anda  
AB"; break;  
case ($nilai > 80 && $nilai <= 100) : echo "Nilai Anda adalah " . $nilai .  
. Indeks nilai anda  
A"; break;  
    default :  
        echo "Nilai Anda adalah " . $nilai . ". Maaf, Anda tidak  
lulus"; break; }
```



Gambar 5.5 Hasil penggunaan switch-case

5.7. Perulangan (Looping)

Banyak jenis perulangan yang terdapat pada PHP. Adapun beberapa diantaranya adalah :

1. Perulangan for

```
for (init_awal, kondisi, counter) {  
    statement;  
}
```

2. Perulangan while

```
init_awal; while (kondisi) {  
    statement;  
    counter;  
}
```

3. Perulangan do-while

```
init_awal;  
do {  
    statement;    counter;  
} while (kondisi);
```

4. Perulangan foreach

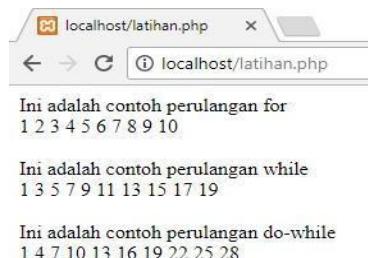
```
foreach (array_expression as $value) {  
    statement;  
}
```

Berikut adalah contoh penggunaan perulangan :

```
<?php
    echo "Ini adalah contoh perulangan for";
    echo "<br>";
    for ($i=1 ; $i <= 10 ; $i++) {
        echo $i . " ";
    }

    echo "<br>";      echo "<br>";
    echo "Ini adalah contoh perulangan while";
    echo "<br>";      $i = 1;
    while ($i <= 20) {
        echo $i . " ";
        $i+=2;
    }

    echo "<br>";      echo "<br>";
    echo "Ini adalah contoh perulangan do-while";      echo "<br>";
    $i = 1;          do {
        echo $i . " ";
        $i+=3;
    } while ($i <30);
?>
```



Gambar 5.6 Hasil penggunaan perulangan

5.8. Function

Dalam merancang kode program, kadang kita sering membuat kode yang melakukan tugas yang sama secara berulang-ulang, seperti membaca tabel dari *database*, menampilkan penjumlahan, dan lain-lain. Tugas yang sama ini akan lebih efektif jika dipisahkan dari program utama, dan dirancang menjadi sebuah **fungsi**.

Fungsi dipanggil dengan menulis nama dari fungsi tersebut, dan diikuti dengan argumen (jika ada). Argumen ditulis di dalam tanda kurung, dan jika jumlah argumen lebih dari satu, maka diantaranya dipisahkan oleh karakter koma.

Bentuk umum pendefinisian fungsi pada PHP adalah sebagai berikut:

```
function nama_fungsi(parameter1, parameter2, ... , n) {
    statement;
}
```

Contoh fungsi pada PHP tanpa menggunakan parameter dan *return value*:

```
<?php
    function cetakGenap() {
        if ($i%2 == 0) {
            for ($i = 1 ; $i <= 100 ; $i++) {
```

```

        echo "$i ";
    }
}
//pemanggilan fungsi
cetakGenap();
?>

```

Contoh fungsi pada PHP menggunakan parameter dan tanpa *return value*:

```

<?php
    function cetakGenap($awal , $akhir) { for ($i = $awal ; $i <= $akhir ;
$i++) {
        if ($i%2 == 0) {
            echo "$i ";
        }
    }
}

//pemanggilan fungsi
$a = 10;
$b = 50;
echo "Bilangan ganjil dari $a sampai $b adalah : <br>" . cetakGenap($a
, $b);
?>

```

Contoh fungsi pada PHP dengan *return value*:

```

<?php
    function luasSegitiga($alas , $tinggi) {
        return 0.5 * $alas * $tinggi;
    }

//pemanggilan fungsi
$a= 10;      $t = 50;
echo "Luas Segitiga dengan alas $a dan tinggi $t adalah : " .
luasSegitiga($a , $t);
?>

```

5.9. Array

Array merupakan tipe data terstruktur yang berguna untuk menyimpan sejumlah data yang bertipe sama. Bagian yang menyusun *array* disebut elemen *array*, yang masing-masing elemen dapat diakses tersendiri melalui *index array*. *Index array* dapat berupa bilangan *integer* atau *string*.

Untuk mendeklarasikan atau mendefinisikan sebuah *array* di PHP bisa menggunakan *keyword array()*. Jumlah elemen *array* tidak perlu disebutkan saat deklarasi. Sedangkan untuk menampilkan isi *array* pada elemen tertentu, cukup dengan menyebutkan nama *array* beserta *index array*-nya.

Berikut adalah cara mendeklarasikan suatu *array* di PHP :

```

<?php
    $arrKendaraan = array("Mobil","Pesawat","Kereta Api","Kapal Laut");
echo $arrKendaraan[0] . "<br>"; //Mobil
echo $arrKendaraan[2] . "<br>"; //Kereta Api

    $arrKota = array();
    $arrKota[] = "Jakarta";
    $arrKota[] = "Medan";
    $arrKota[] = "Bandung";
    $arrKota[] = "Malang";    $arrKota[] = "Sulawesi"; echo $arrKota[1] .
"<br>"; //Medan    echo $arrKota[2] . "<br>"; //Bandung    echo $arrKota[4] .
"<br>"; //Sulawesi ?>

```

Cara mendeklarasikan suatu *array* pada PHP bisa dengan *index string* atau yang dinamakan dengan *array assosiatif*. Berikut adalah contoh pendeklarasian *array assosiatif* :

```
<?php
$arrAlamat = array ("Rona" => "Banjarmasin" , "Dhiva" => "Bandung" ,
                     "Ilham" => "Medan" , "Oku" => "Hongkong");      echo $arrAlamat['Dhiva'] .
"<br>"; //Bandung   echo $arrAlamat['Oku'] . "<br>"; //Hongkong
$arrNim = array();
$arrNim['Rona'] = '11011112';
$arrNim['Dhiva'] = '11011101';
$arrNim['Ilham'] = '11011309';
$arrNim['Oku'] = '11014765';      $arrNim['Fadhlal'] = '11011113';
echo $arrNim['Ilham'] . "<br>"; //11011309      echo $arrNim['Fadhlal'] . "<br>";
//11011113 ?>
```

5.10. GET dan POST

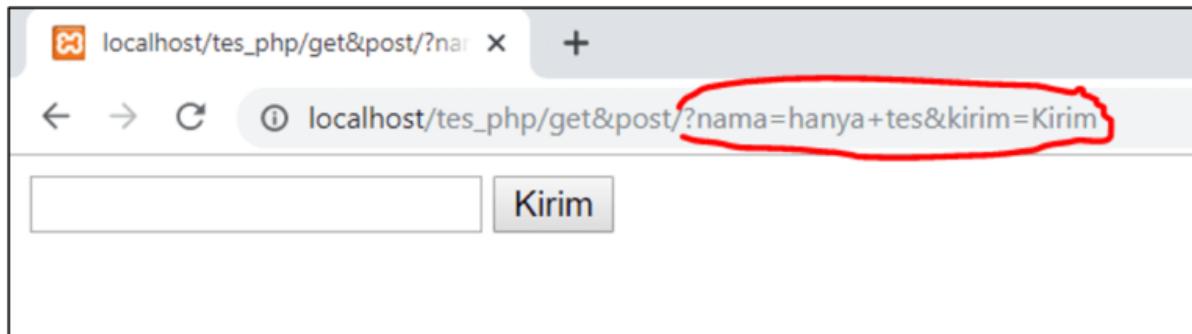
Pada dasarnya GET dan POST adalah metode pengiriman yang digunakan dalam proses pengiriman data pada Form.

5.10.1. Metode GET

Pada metode GET nilai dari form yang dikirim dapat dilihat langsung pada URL. Untuk membuat metode GET, silahkan ketik kode di bawah ini :

```
<form method="GET">
  <input type="text" name="nama">
  <input type="submit" name="kirim">
</form>
```

Lalu jalankan, maka outputnya akan seperti dibawah ini :



Gambar 5.7 Output URL dari Metode GET

Untuk dapat melihat data yang dikirimkan menggunakan metode GET, dapat menggunakan perintah `$_GET`. Silahkan tambahkan barisan kode dibawah ini :

```
<pre><?php print_r($_GET) ?></pre>
```

maka hasilnya akan seperti ini :



Gambar 5.8 Output Data dari Metode GET

Metode GET sangat tidak disarankan untuk memproses data yang bersifat sensitive seperti password. Akan tetapi, metode GET disarankan untuk mendapatkan data seperti id user, sesuai dengan namanya, yaitu GET yang berarti mendapatkan.

5.10.2. Metode POST

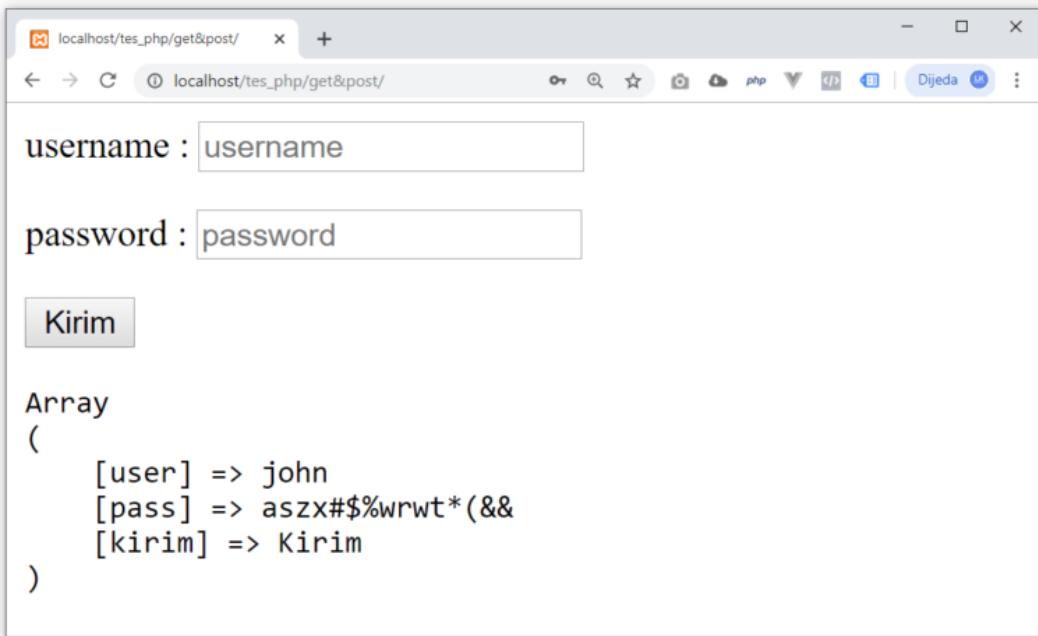
Berbanding terbalik dengan GET, pada metode POST data yang dikirim tidak akan ditampilkan pada url browser, sehingga metode pengiriman ini lebih aman untuk data yang sifatnya sensitif seperti password atau nomor rekening. Metode POST lebih sering digunakan dalam hal registrasi user atau login.

Untuk dapat menggunakan metode POST, silahkan isi dengan barisan kode dibawah ini :

```
<form method="POST">
<p>username : <input type="text" name="user" placeholder="username"></p>
<p>password : <input type="password" name="pass" placeholder="password"></p>
<p><input type="submit" name="kirim"></p>
</form>

<?php if ( isset($_POST) ) : ?>
<pre><?php print_r($_POST) ?></pre>
<?php endif ?>
```

Maka hasilnya akan seperti dibawah ini :



Gambar 5.9 Output dengan Metode POST

5.11. XML

5.11.1. Pengertian XML

XML (*Extensible Markup Language*) adalah bahasa *markup* (*markup language*) yang berfungsi untuk mengidentifikasi data, menyimpan data, serta mengorganisir suatu data. Beberapa kegunaan yang bisa didapat dari XML adalah sebagai berikut:

- XML bisa digunakan untuk menyederhanakan dokumen HTML pada informasi yang besar.
- XML bisa digunakan untuk pertukaran informasi antar sistem.
- XML bisa digunakan sebagai sebuah *database*.
- Secara virtual, semua data dapat diekspresikan sebagai sebuah dokumen XML

5.11.2. Sintaks XML

Seperti yang sudah dijelaskan sebelumnya, pada umumnya XML digunakan untuk menyimpan suatu informasi (data). Berikut adalah contoh dokumen XML sederhana.

```
<?xml version="1.0" encoding="UTF-8"?>
<Resep nama="roti" waktu_persiapan="5 menit" waktu_masak="3 jam">
    <judul>Roti tawar</judul>
    <bahan jumlah="3" satuan="cangkir">tepung</bahan>
    <bahan jumlah="0,25" satuan="ons">ragi</bahan>
    <bahan jumlah="1,5" satuan="cangkir">air hangat</bahan>
    <bahan jumlah="1" satuan="sendok teh">garam</bahan>
    <Cara_membuat>
        <langkah>Campur semua bahan dan uleni adonan sampai merata.</langkah>
        <langkah>Tutup dengan kain lembap dan biarkan selama satu jam di
ruangan yang hangat.</langkah>
        <langkah>Ulangi lagi, letakkan di loyang dan panggang di
oven.</langkah>
        <langkah>Keluarkan, hidangkan</langkah>
    </Cara_membuat>
</Resep>
```

Dari contoh diatas, terdapat beberapa komponen yang ada pada sebuah dokumen XML sederhana :

1. Deklarasi XML

Example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
```

Gambar 5.10 Deklarasi XML

Sebuah dokumen XML tidak harus memiliki sebuah deklarasi, namun jika diperlukan, **sebuah deklarasi XML harus berada pada baris pertama** dari sebuah dokumen XML. Syarat untuk sebuah deklarasi XML adalah:

- Deklarasi XML bersifat *case sensitive*, sebuah deklarasi XML harus dimulai dengan "<?xml>" dan "xml" ditulis dengan huruf kecil.
- Harus memiliki atribut *version* didalamnya.

2. Tags dan Elements

Sebuah dokumen XML disusun dari beberapa Elemen XML yang biasa disebut dengan *XMLnodes* atau *XML-tags*. Nama dari sebuah elemen XML dimulai dengan "<" dan ditutup dengan ">".

```
<element>....</element>
```

Selain itu, sebuah elemen XML dapat memiliki beberapa elemen XML yang disebut dengan *child*. Setiap *child* harus ditutup dengan *tag* dari *child* tersebut sebelum memulai *tag* dari *child* baru.

Berikut adalah contoh penggunaan *child* yang **salah**:

```
<?xml version = "1.0"?>
<contact-info>
<company>Telkom University
<contact-info>
</company>
```

Sedangkan berikut adalah contoh penggunaan *child* yang **benar**:

```
<?xml version = "1.0"?>
<contact-info>
    <company>Telkom University</company>
<contact-info>
```

3. Atribut XML

Atribut XML adalah properti yang terdapat pada sebuah elemen XML. Sebuah elemen XML dapat memiliki lebih dari satu atribut.

```
<a href = "http://www.telkomuniversity.ac.id/">Telkom University</a>
```

Contoh diatas, "href" merupakan nama atribut dan "http://www.telkomuniversity.ac.id/" adalah nilai dari atribut tersebut.

Penulisan atribut juga memiliki aturan sebagai berikut:

- Nama atribut bersifat *case sensitive*, karena itu *HREF* dan *href* merupakan nama atribut yang berbeda.
- Nama atribut tidak boleh sama pada sebuah *XML tags*
- Nama atribut harus dituliskan tanpa tanda kutip (""), namun nilai atribut harus dituliskan menggunakan tanda kutip.

5.12. JSON

5.12.1. Pengertian JSON

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari bahasa pemrograman JavaScript, Standar ECMA-262 Edisi ke-3 - Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemrograman

apapun karena menggunakan gaya bahasa yang umum digunakan oleh *programmer* keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran data.

JSON dapat disusun oleh dua struktur:

- Kumpulan pasangan nama-nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel *hash* (*hash table*), daftar berkunci (*keyed list*), atau *associative array*.
- Daftar nilai terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).

Beberapa kelebihan JSON yang menjadikan JSON dapat menggantikan XML sebagai format pertukaran data adalah:

- Ukuran lebih kecil dibanding XML, efeknya transfer data lebih cepat dan lebih hemat *resource*, terutama *bandwidth*
- JSON adalah format data bawaan di Javascript, artinya jika data dari *server* dikirim ke *client*, dan *client* menggunakan Javascript, maka tidak perlu *library* tambahan untuk memprosesnya.
- Dibanding XML, format JSON lebih sederhana.
- Library JSON ada di setiap bahasa pemrograman sehingga memudahkan *programmer* yang berbeda bahasa pemrograman.

5.12.2. Perbedaan JSON dan XML

Berikut contoh dokumen JSON yang menyimpan informasi terkait buku.

```
{  
  "book": [  
    {  
      "id": "01",  
      "language": "Java",  
      "edition": "third",  
      "author": "Herbert Schildt"  
    },  
    {  
      "id": "07",  
      "language": "C++",  
      "edition": "second",  
      "author": "E.Balagurusamy"  
    }  
  ]  
}
```

Suatu objek pada JSON dinotasikan dengan simbol kurung kurawal ({}), sedangkan kumpulan dari beberapa objek dapat dihimpun dalam notasi kurung siku ([]). Data yang disimpan dalam JSON juga dapat ditranslasi kedalam format XML seperti contoh berikut.

```
<?xml version="1.0" encoding="UTF-8" ?>  
<root>  
  <book>  
    <id>01</id>  
    <language>Java</language>  
    <edition>third</edition>  
    <author>Herbert Schildt</author>  
  </book>  <book>  
    <id>07</id>  
    <language>C++</language>  
    <edition>second</edition>  
    <author>E.Balagurusamy</author>  
  </book>  
</root>
```

5.13. AJAX

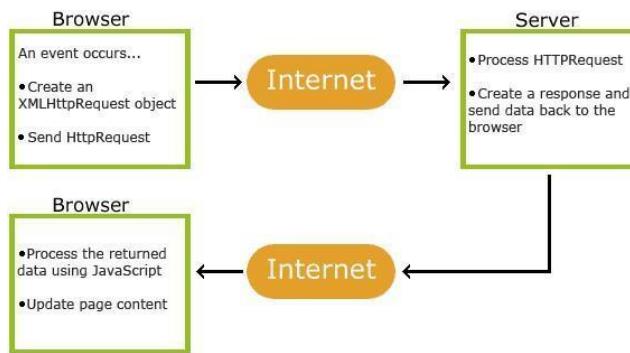
5.13.1. Apa Itu AJAX

AJAX (*Asynchronous JavaScript and XML*) suatu teknik pemrograman berbasis *web* untuk menciptakan aplikasi *web* interaktif. Tujuannya adalah untuk memindahkan sebagian besar interaksi pada komputer *user*, melakukan pertukaran data dengan *server* di belakang layar, sehingga halaman *web* tidak harus dibaca ulang secara keseluruhan setiap kali seorang pengguna melakukan perubahan. Hal ini akan meningkatkan interaktivitas, kecepatan, dan *usability*.

Secara umum, AJAX melibatkan dua hal yakni:

1. Objek XMLHttpRequest bawaan *browser* (untuk meminta data dari sebuah *web server*).
2. Javascript dan HTML DOM (untuk menampilkan data pada *web browser*).

5.13.2. Cara Kerja AJAX



Gambar 5.11 Cara kerja AJAX

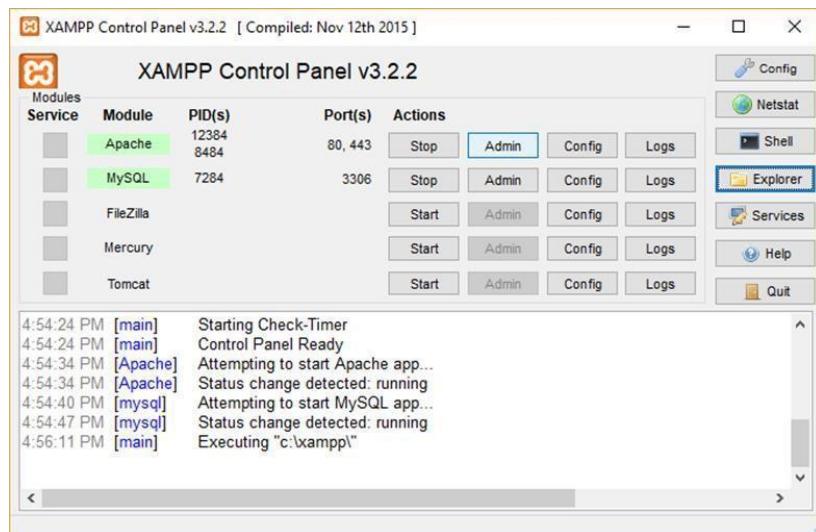
Dalam aplikasinya, AJAX melakukan hal-hal berikut:

1. Suatu *event* terjadi pada halaman *web* (seperti *page loaded* atau *button clicked*).
2. Sebuah objek XMLHttpRequest dibuat oleh Javascript
3. Objek XMLHttpRequest mengirimkan *request* kepada *web server*.
4. *Web server* mengelola *request*.
5. *Web server* mengirimkan *response* kepada *client*.
6. *Response* dibaca oleh Javascript.
7. Javascript melakukan perubahan pada halaman *web* menggunakan DOM.

5.13.3. Event Handling

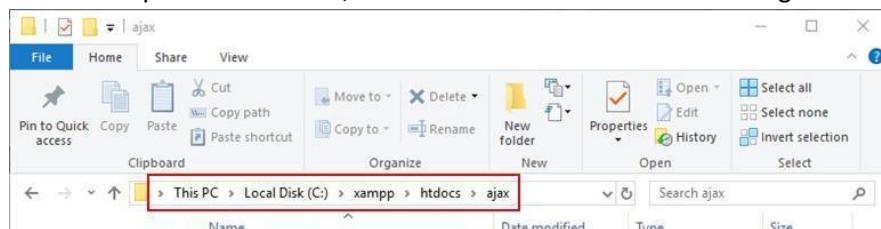
Pada contoh berikut, akan dilakukan perubahan halaman *web* menggunakan teknik AJAX. Berikut langkah-langkah yang perlu dilakukan :

1. Pastikan PHP *web server* sudah berjalan dengan baik. Pada modul ini digunakan Apache *web server* yang terdapat pada XAMPP v3.2.2.



Gambar 5.12 Tampilan control panel XAMPP

2. Akses folder htdocs pada *local server*, dan kemudian buat folder baru dengan nama seperti: ajax



Gambar 5.13 Folder penyimpanan file AJAX

3. Buat file .txt berikut yang berfungsi sebagai pengganti konten halaman web.

```
<h1>AJAX</h1>
<p>AJAX is not a programming language.</p>
<p>AJAX is a technique for accessing web servers from a web page.</p>
<p>AJAX stands for Asynchronous JavaScript And XML.</p>
```

Simpan file sebagai ajax_info.txt.

4. Buat juga file HTML utama yang berisikan code sebagai berikut. Dan simpan sebagai index.html

```
<!DOCTYPE html>
<html>
<body>

<h2>The XMLHttpRequest Object</h2>

<p id="demo">Let AJAX change this text.</p>

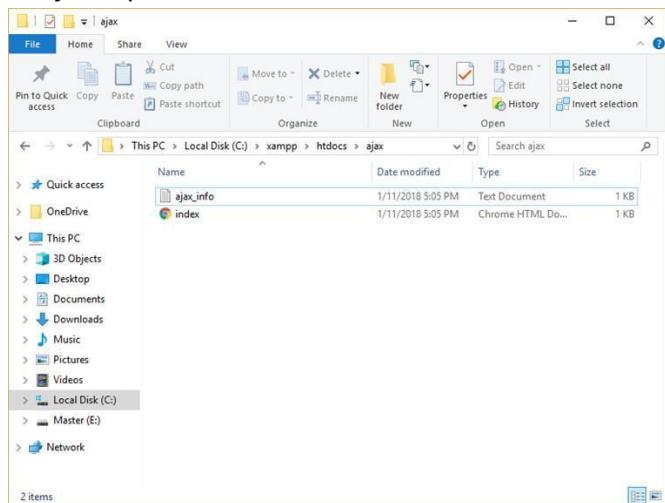
<button type="button" onclick="loadDoc()">Change Content</button>

<script> function loadDoc() { var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() { if (this.readyState == 4 &&
this.status == 200) {
document.getElementById("demo").innerHTML = this.responseText;
} }; xhttp.open("GET", "ajax_info.txt", true); xhttp.send();
}

</script>

</body>
</html>
```

5. Tempatkan kedua *file* tersebut kedalam folder ajax yang telah dibuat pada langkah kedua sehingga posisi kedua *file* seperti berikut.



Gambar 5.14 File ajax_info pada folder penyimpanan file AJAX

6. Ketika anda mengakses halaman *web* tersebut pada alamat <http://localhost/ajax/>, tampilan yang akan muncul adalah seperti gambar 5-12.

The XMLHttpRequest Object

Let AJAX change this text.

Gambar 5.15 Tampilan file AJAX ketika dieksekusi

7. Namun jika anda melakukan *action* yaitu menekan *button Change Content*, maka konten pada halaman *web* akan menjadi seperti ini.

The XMLHttpRequest Object

AJAX

AJAX is not a programming language.

AJAX is a technique for accessing web servers from a web page.

AJAX stands for Asynchronous JavaScript And XML.

Gambar 5.16 Tampilan file AJAX-2

Berikut adalah penjelasannya :

1. Peran terbesar AJAX pada contoh diatas adalah pada *code* Javascript yang terdapat di *file* index.html.

```
<script> function loadDoc() { var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() { if (this.readyState == 4 && this.status == 200) {
document.getElementById("demo").innerHTML = this.responseText;
} }; xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
}
</script>
```

Code tersebut akan dieksekusi apabila *button Change Content* ditekan karena atribut `onClick()` yang terdapat pada *button* tersebut.

2. Pembuatan objek `XMLHttpRequest()` dilakukan pada *variable* `xhttp`.
3. Selanjutnya pada fungsi `onreadystatechange()` yang terdapat pada objek `XMLHttpRequest` dilakukan pengecekan apakah *request* dapat dilakukan.
4. Jika pengecekan pada langkah ke-3 berhasil, maka dilakukan perubahan isi konten dengan atribut `id="demo"` pada bagian *body* HTML menjadi *response* yang didapat pada baris selanjutnya.
5. Objek `XMLHttpRequest` mengeksekusi *method* `open()` untuk mengirim *request* kepada *web server*. Parameter yang terdapat pada *method* `open()` adalah sebagai berikut:

<code>open(method, url, async, user, psw)</code>	Specifies the request
<i>method</i> : the request type GET or POST <i>url</i> : the file location <i>async</i> : true (asynchronous) or false (synchronous) <i>user</i> : optional user name <i>psw</i> : optional password	

Gambar STYLEREF 1 \s 5. SEQ Gambar * ARABIC \s 1 17
Parameter pada salah satu fungsi AJAX

Gambar 5.17 Parameter pada salah satu fungsi AJAX

6. Sehingga yang dilakukan pada *code* secara *asynchronous* kemudian mengirimkannya ke *server*.
`xhttp.open("GET", "ajax_info.txt", true); xhttp.send();`
7. Jika *file* yang diminta terdapat pada *server*, maka konten akan berubah.

MODUL 6. LARAVEL

Tujuan Praktikum
1. Mahasiswa mampu memahami konsep dan implementasi MVC menggunakan <i>web framework</i> Laravel.

6.1. Framework & MVC

Framework atau dalam Bahasa Indonesia dapat diartikan sebagai “kerangka kerja” merupakan kumpulan dari fungsi-fungsi/prosedur-prosedur dan *class-class* untuk tujuan tertentu yang sudah siap digunakan sehingga bisa lebih mempermudah dan mempercepat pekerjaan seorang *programmer*, tanpa harus membuat fungsi atau *class* dari awal.

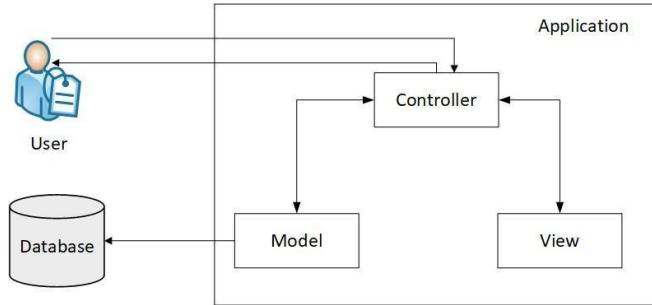
Alasan mengapa menggunakan *Framework*

- Mempercepat dan mempermudah pembangunan sebuah aplikasi *web*.
- Relatif memudahkan dalam proses *maintenance* karena sudah ada pola tertentu dalam sebuah *framework* (dengan syarat *programmer* mengikuti pola standar yang ada).
- Umumnya *framework* menyediakan fasilitas-fasilitas yang umum dipakai sehingga kita tidak perlu membangun dari awal (misalnya validasi, ORM, *pagination*, *multiple database*, *scaffolding*, pengaturan *session*, *error handling*, dll).
- Lebih bebas dalam pengembangan jika dibandingkan CMS.

Model-View-Controller (MVC) merupakan suatu konsep yang cukup popular dalam pembangunan aplikasi *web*. MVC memisahkan pengembangan aplikasi berdasarkan komponen utama yang membangun sebuah aplikasi seperti manipulasi data, *user interface*, dan bagian yang menjadi kontrol aplikasi. Terdapat 3 jenis komponen yang membangun suatu MVC *pattern* dalam suatu aplikasi yaitu:

- *View*, merupakan bagian yang menangani *presentation logic*. Pada suatu aplikasi *web* bagian ini biasanya berupa *file template* HTML, yang diatur oleh *controller*. *View* berfungsi untuk menerima dan merepresentasikan data kepada *user*. Bagian ini tidak memiliki akses langsung terhadap bagian *model*.
- *Model*, biasanya berhubungan langsung dengan *database* untuk memanipulasi data (*insert*, *update*, *delete*, *search*), menangani validasi dari bagian *controller*, namun tidak dapat berhubungan langsung dengan bagian *view*.
- *Controller*, merupakan bagian yang mengatur hubungan antara bagian *model* dan bagian *view*, *controller* berfungsi untuk menerima *request* dan data dari *user* kemudian menentukan apa yang akan diproses oleh aplikasi.

Singkat kata **Model** untuk mengatur alur *database*, **View** untuk menampilkan *web*, sedangkan **Controller** untuk mengatur alur kerja antara *Model* dan *View*. Jadi misalnya Anda akan membuat akun *e-mail*. Pertama anda akan melihat tampilan *sign-up* / *register*, itulah yang disebut dengan *View*. Kemudian Anda mengisi *form username*, *password*, dan lain-lain dan Anda klik tombol *Register*, maka disinilah *View* akan memanggil *Controller* dan *Controller* memanggil *Model*. Adapun tugas *Model* disini untuk mengecek apakah Anda sudah mengisi sesuai dengan kriteria dan akan dihubungkan dengan *database*. Kemudian *Model* akan mengembalikan ke *Controller* dan *Controller* akan mengembalikan ke *View*. Berikut adalah gambaran konsep MVC yang diterapkan pada framework.



Gambar 6.1 Cara kerja MVC

6.2. Pengenalan Laravel

Laravel adalah sebuah web application framework yang bersifat open-source yang digunakan untuk membangun aplikasi php dinamis. Laravel menjadi sebuah framework PHP dengan model MVC (Model, View, Controller) yang dapat mempercepat pengembang untuk membuat sebuah aplikasi web. Selain ringan dan cepat, Laravel juga memiliki dokumentasi yang lengkap disertai dengan contoh implementasi kodenya.

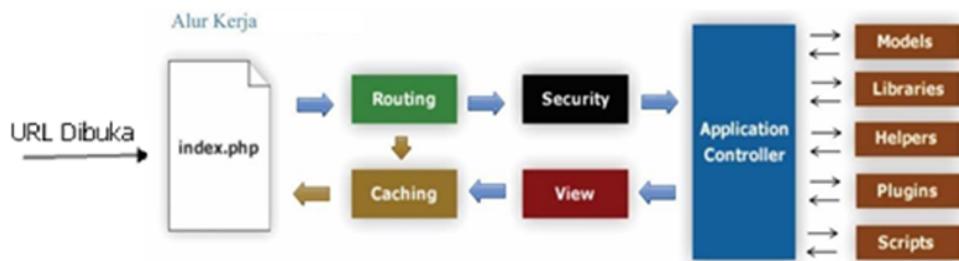
Laravel pertama kali dikembangkan pada tahun 2011 oleh Taylor Otwell. Saat ini Laravel sudah mencapai versi 9 yang dirilis pada tanggal 8 Februari 2022. Sebagai web framework populer yang menggunakan bahasa pemrograman PHP, Laravel mempunyai beberapa keunggulan yaitu:

1. *Free*, karena berada di bawah lisensi open source, kita dapat melakukan apa pun.
2. Menggunakan kaidah MVC, dengan menggunakan *Model-View-Controller*, kita dapat memisahkan bagian *logic* dan *presentation* dari aplikasi yang kita bangun.
3. Menghasilkan URL yang bersih. URL yang dihasilkan oleh Codeigniter bersih dan ramah terhadap *search engine*. Codeigniter menggunakan pendekatan *segment-based* dibandingkan dengan *query string* yang biasa digunakan oleh programmer yang tidak menggunakan *web framework*.
4. *Packs a Punch*, Laravel hadir dengan berbagai *library* yang akan membantu tugas-tugas di pengembangan web yang sudah umum dan sering dilakukan, seperti mengakses *database*, mengirim email, validasi data dari form, mengelola *session*, memanipulasi file, dan masih banyak lagi.
5. *Extensible*, kita dapat menambahkan *library* atau *helper* yang kita ciptakan sendiri ke dalam Laravel. Selain itu, kita dapat juga menambahkan fitur lewat *class extension*.
6. *Throughly Documented*, hampir semua fitur, *library*, dan *helper* yang ada di Laravel telah terdokumentasi dengan lengkap dan tersusun dengan baik. Dokumentasi cara penggunaannya dapat dilihat di <https://laravel.com/docs/9.x>.

Berikut adalah istilah yang sering ditemui di Laravel.

1. *Model*, class PHP yang dirancang untuk bekerja dengan informasi dari *database*.
2. *Controller*, inti aplikasi yang menentukan penanganan logic dari aplikasi web
3. *Route*, bagian yang menangani HTTP *request*.
4. *View*, halaman web seperti *header*, *footer*, *sidebar*, dan sebagainya yang ditanamkan di halaman web. *View* tidak pernah dipanggil secara langsung, tetapi harus dipanggil dari *controller*.
5. *Library*, class yang berisi fungsi-fungsi untuk penyelesaian kasus tertentu.

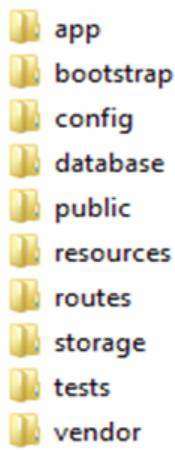
6.3. Cara Kerja Laravel



Gambar 6.2 Alur kerja Laravel

Ketika suatu URL dibuka, maka file index.php akan dibaca. Di dalam file ini, Laravel memeriksa apakah URL tersebut terdaftar pada Routing yang sudah dibuat. Jika ada pemetaannya dan memenuhi Security (jika diharuskan, autentikasi misalnya) maka Laravel akan memanggil Controller sesuai yang dipetakan oleh Routing. Di dalam Controller semua logika back-end dieksekusi, lalu dapat mengembalikan View (tampilan) ke browser. Tampilan ini disimpan dalam Caching agar pemrosesan setelahnya lebih cepat.

Sedangkan struktur folder Laravel dapat dilihat pada gambar di bawah ini:



Gambar 6.3 Struktur Folder Laravel

- app
Folder ini berisi Controller, Model, Middleware, dan Provider
- bootstrap
Folder ini berisi Cache untuk mempercepat pemrosesan
- config
Folder ini berisi semua file konfigurasi untuk aplikasi
- database
Folder ini berisi file-file migrasi dari/ke database
- public

Folder ini berisi file yang dapat diakses langsung, biasanya file asset (gambar, CSS, dan JS) atau file konten (file untuk di-download)

- resources
Folder ini berisi View (tampilan)
- routes
Folder ini berisi Routing untuk pemetaan URL ke aplikasi
- storage
Folder ini berisi hasil kompilasi View dan log dari aplikasi
- tests
Folder ini berisi file-file untuk unit testing
- vendor
Folder ini berisi file-file library yang dibutuhkan aplikasi

6.3.1. Routing

Pengaturan Routing berada pada file routes/web.php. Disini kita dapat mengatur pemetaan URL dengan aksi yang ingin kita lakukan. Method nya pun dapat diatur jika hanya untuk menerima method tertentu saja (post, get, atau yang lainnya). Contoh:

```
// menampilkan halaman welcome.blade.php dalam folder resources/views
Route::get('/', function () {
    return view('welcome');
});

// jika hanya menampilkan view, bisa juga menggunakan ini
Route::view('/', 'welcome');

// memanggil fungsi dari suatu Controller
Route::post('/auth', [SiteController::class, 'auth']);

// atau bisa juga seperti ini
Route::get('/product', 'App\Http\Controllers\ProductController@index');
```

Routing pun bisa diatur dengan parameter, middleware, group, maupun dengan prefix.

6.3.2. View

Secara default file tampilan pada Laravel menggunakan template engine, yaitu Blade. Sehingga, setiap file tampilan yang kita buat harus mengikuti penamaan **namaview.blade.php** dan diletakkan pada folder **resources/views** agar dapat di-load oleh fungsi **view()**. Penggunaan Blade juga menyediakan banyak directive yang sangat berguna terutama untuk looping data dan templating halaman agar kita tidak menulis kode yang sama berulang kali (biasanya header, menu samping, dan footer memiliki tampilan yang sama untuk tiap halaman). Secara isi, tampilan pada Laravel tetap menggunakan HTML, CSS, dan Javascript. Hal yang harus diperhatikan yaitu jika ingin menggunakan file eksternal CSS / Javascript, maka harus menggunakan fungsi **{{ asset('path file di folder public') }}**, dan khusus untuk halaman yang berisi form dengan method selain GET maka harus menambahkan directive **@csrf** untuk keamanan pengiriman form. Berikut adalah contoh tampilan halaman login yang menggunakan file eksternal jquery.js yang diletakkan di dalam folder **assets**:

```

<!DOCTYPE html>
<html>
  <head>
    <title>Login</title>
    <link rel="stylesheet"
    href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css"
    integrity="sha384-zCbKRCUGaJDkqS1kPbPd7TveP5iyJE0EjAuZQTgFLD2ylzuqKfdKlfG/eSrtxU
    kn" crossorigin="anonymous">
    <script src="{{ asset('assets/jquery.js') }}></script>
    <script
    src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/js/bootstrap.bundle.min.j
    s"
    integrity="sha384-fQybJgWLrvvRgtW6bF1B7jaZrFsaBXjsOMm/tB9LTS58ONXgqbR9W8oWht/amn
    pF" crossorigin="anonymous"></script>
  </head>
  <body style="width:95%">
    <div class="row justify-content-center" style="margin-top:13%">
      <div class="col-3 border">
        <form style="margin:20px" method="POST" action="/auth">
          @csrf
          <div class="form-group">
            <label for="email">Email</label>
            <input type="email" name="em" class="form-control" />
          </div>
          <div class="form-group">
            <label>Password</label>
            <input type="password" name="pwd" class="form-control" />
          </div>
          <div style="text-align:center">
            <button class="btn btn-success">Login</button>
          </div>
        </form>
      </div>
    </div>
  </body>
</html>

```

6.3.3. Controller

File Controller diletakkan pada folder **app\Http\Controllers**. Cara membuatnya bisa dengan cara pembuatan file manual dengan memperhatikan hal-hal berikut: meng-extend base Controller dari Laravel atau turunannya dan dituliskan namespace “App\Http\Controllers” agar dapat digunakan oleh file lain dan dipetakan dari Routing. Atau cara lain dengan perintah di command prompt / console / terminal. Perintahnya:

```
php artisan make:controller SiteController
```

Jika Controller akan dipakai untuk manajemen data dengan Database, maka perintah di atas dapat ditambahkan --resource sehingga fungsi-fungsi default yang akan digunakan untuk manajemen data di-generate secara otomatis oleh Laravel.

```
php artisan make:controller ProductController --resource
```

Kelebihan lain dari menggunakan resource ini adalah kita dapat menambah satu Routing saja untuk menangani semua aksi dari satu Controller. Di web/routes.php kita cukup menambahkan:

```
Route::resource('product', ProductController::class);
```

Routing akan secara otomatis menangani URL berikut ini:

Tabel 6.1 URL yang ditangani otomatis

Method	URL	Fungsi yang dieksekusi
GET	/product	index
GET	/product/create	create
POST	/product	store
GET	/product/{id}	show
GET	/product/{id}/edit	edit
PUT/PATCH	/product/{id}	update
DELETE	/product/{id}	destroy

Berikut adalah hasil generate dari ProductController:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        //
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        //
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
```

```
//  
}  
  
/**  
 * Display the specified resource.  
 *  
 * @param int $id  
 * @return \Illuminate\Http\Response  
 */  
public function show($id)  
{  
    //  
}  
  
/**  
 * Show the form for editing the specified resource.  
 *  
 * @param int $id  
 * @return \Illuminate\Http\Response  
 */  
public function edit($id)  
{  
    //  
}  
  
/**  
 * Update the specified resource in storage.  
 *  
 * @param \Illuminate\Http\Request $request  
 * @param int $id  
 * @return \Illuminate\Http\Response  
 */  
public function update(Request $request, $id)  
{  
    //  
}  
  
/**  
 * Remove the specified resource from storage.  
 *  
 * @param int $id  
 * @return \Illuminate\Http\Response  
 */  
public function destroy($id)  
{  
    //  
}  
}
```

MODUL 7. LARAVEL LANJUT

Tujuan Praktikum
1. Mahasiswa mampu memahami konsep dan implementasi CodeIgniter pada pembuatan aplikasi berbasis web. 2. Mahasiswa mampu menerapkan CRUD menggunakan Laravel dan konsep MVC. 3. Mahasiswa mampu mengimplementasikan fitur-fitur yang sering digunakan pada Laravel.

7.1. CRUD

Pada modul ini kita akan membuat aplikasi yang dapat melakukan create, read, update, dan delete terhadap suatu data/tabel. Aplikasi yang akan kita bangun yaitu aplikasi e-commerce dimana kita akan fokus melakukan CRUD terhadap tabel produknya saja.

7.1.1. Konfigurasi dan Skema

Hal pertama yang harus dilakukan agar aplikasi dapat terhubung dengan database adalah mengatur konfigurasi koneksi. Secara detil, konfigurasi database berada pada file **config/database.php**. Tetapi untuk konfigurasi standar, kita dapat mengubahnya di file **.env** pada baris yang berisi DB_CONNECTION hingga DB_PASSWORD. Ubah nilainya dengan pengaturan yang kita inginkan.

```
...  
DB_CONNECTION=mysql  
DB_HOST=127.0.0.1  
DB_PORT=3306  
DB_DATABASE=ecommerce  
DB_USERNAME=root  
DB_PASSWORD=  
...
```

Buat database bernama **ecommerce** di DBMS. Untuk membuat tabelnya, dapat dilakukan secara otomatis (di-generate) oleh Laravel dari command prompt / console / terminal dengan perintah. Perintah yang pertama yaitu membuat file migrasi sebagai skema dari tabel:

```
php artisan make:migration create_products_table
```

File migrasi **xxx_create_products_table.php** akan terbuat pada folder **database/migrations**. Edit file ini pada fungsi **Schema::create()** untuk mendefinisikan kolom yang kita inginkan.

```
Schema::create('products', function (Blueprint $table) {  
    $table->id();  
    $table->string('name');  
    $table->integer('price');  
    $table->timestamps();  
});
```

Penjabaran dari kode ini:

- Fungsi **id()** yaitu kita mendefinisikan kolom **id** sebagai PK dengan tipe int dan auto-increment.
- Fungsi **string('name')** yaitu kita mendefinisikan kolom **name** dengan tipe varchar
- Fungsi **integer('price')** yaitu kita mendefinisikan kolom **price** dengan tipe int
- Fungsi **timestamps()** yaitu kita mendefinisikan kolom **created_at** yang akan terisi jika data dibuat melalui ORM dan **updated_at** yang akan terisi jika data diubah melalui ORM

Untuk membuat tabel ke database dari skema ini, maka ketikkan perintah berikut pada command prompt / console / terminal:

```
php artisan migrate
```

7.1.2. Model

Laravel memberikan tiga cara untuk mengakses ataupun manipulasi data ke database: query langsung, query builder, dan ORM. Query langsung dan query builder menggunakan library Illuminate (**Illuminate\Support\Facades\DB**) sedangkan ORM menggunakan Eloquent, yang merupakan kelas extend dari Illuminate. File Model diletakkan pada folder **app/Models**. Untuk membuat Model dalam Laravel, dapat dilakukan dengan manual dengan menambahkan namespace "App\Models" dan meng-extend kelas base Model dari Eloquent (**Illuminate\Database\Eloquent\Model**) ataupun di-generate oleh Laravel dari command prompt / console / terminal dengan perintah:

```
php artisan make:model Product
```

Jika sebelumnya belum membuat Controller atau file migration-nya, perintah generate Model ini dapat ditambahkan parameter sehingga bisa sekaligus me-generate file Controller-nya (-c), file migration-nya (-m), ataupun keduanya langsung (-cm).

Eloquent memiliki beberapa konvensi / aturan yang harus diperhatikan yaitu:

- Nama sebuah Model “X” secara otomatis merepresentasikan tabel database bernama “xs”. Contoh, Model Product akan merepresentasikan tabel **products**. Jika tabel yang direpresentasikan berbeda, maka harus ditambahkan atribut **protected \$table = 'nama_tabel'**; pada kelas Model.
- Kolom PK pada tabel bernama “id”. Jika kolom PK bukan “id”, maka harus ditambahkan atribut **protected \$primaryKey = 'nama_kolom_PK'**; pada kelas Model.
- Kolom PK pada tabel di-set auto-increment. Jika kolom PK tidak auto-increment, maka harus ditambahkan atribut **public \$incrementing = false**; pada kelas Model.
- Kolom PK pada tabel bertipe integer. Jika kolom PK bukan integer, maka harus ditambahkan atribut **protected \$keyType = 'string'**; pada kelas Model.
- Ada kolom “created_at” dan “updated_at” pada tabel. Jika tidak ada, maka harus ditambahkan atribut **public \$timestamps = false**; pada kelas Model.

Beberapa aturan lain juga dapat diatur seperti date format, koneksi DB yang berbeda, dan nilai default untuk atribut tertentu.

7.1.3. Controller

Setelah Model siap, kita tinggal memanggilnya pada Controller. Berikut kode lengkapnya dalam ProductController:

```
...
use App\Models\Product;

class ProductController extends Controller {
    public function index()
    {
        $prods = Product::get();
        return view('product.index', ['list' => $prods]);
    }
}
```

```

public function create()
{
    return view('product.form', [
        'title' => 'Tambah',
        'method' => 'POST',
        'action' => 'product'
    ]);
}

public function store(Request $request)
{
    $prod = new Product;
    $prod->name = $request->name;
    $prod->price = $request->price;
    $prod->save();
    return redirect('/product')->with('msg', 'Tambah berhasil');
}

public function show($id)
{
    return Product::find($id);
}

public function edit($id)
{
    return view('product.form', [
        'title' => 'Edit',
        'method' => 'PUT',
        'action' => "product/$id",
        'data' => Product::find($id)
    ]);
}

public function update(Request $request, $id)
{
    $prod = Product::find($id);
    $prod->name = $request->name;
    $prod->price = $request->price;
    $prod->save();
    return redirect('/product')->with('msg', 'Edit berhasil');
}

public function destroy($id)
{
    Product::destroy($id);
    // atau
    /* $prod = Product::find($id);
    $prod->delete(); */
    return redirect('/product')->with('msg', 'Hapus berhasil');
}
}

```

7.1.4. View

Untuk tampilannya, buat file dengan nama **index.blade.php** yang diletakkan di folder **resources/views/product** (folder product dibuat dahulu agar rapi).

```
<!DOCTYPE html>
<html>
  <head>
    <title>Daftar Produk</title>
    <link rel="stylesheet"
    href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css"
    integrity="sha384-zCbKRCUGaJDkqS1kPbPd7TveP5iyJE0EjAuZQTgFLD2ylzuqKfdKlfG/eSrtx
    Ukn" crossorigin="anonymous">
      <script src="{{ asset('assets/jquery.js') }}></script>
      <script
    src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/js/bootstrap.bundle.min.
    js"
    integrity="sha384-fQybjgWLrvvRgtW6bF1B7jaZrFsaBXjsOMm/tB9LTS58ONXgqbR9W8oWht/am
    npF" crossorigin="anonymous"></script>
  </head>
  <body style="width:95%">
    <div class="row justify-content-center" style="margin-top:13%">
      <div class="col-4">
        <span class="float-left">{{ session('msg') }}</span>
        <a href="/product/create" class="btn btn-secondary
        float-right">Tambah</a><br /><br />
        <table class="table table-bordered table-striped">
          <tr>
            <th>Nama</th>
            <th>Harga</th>
            <th>Aksi</th>
          </tr>
          @foreach($list as $d)
          <tr>
            <td>{{ $d->name }}</td>
            <td>{{ $d->price }}</td>
            <td>
              <a href="/product/{{ $d->id }}/edit" class="btn
              btn-primary">Edit</a>
              <form method="post" action="/product/{{ $d->id }}"
              style="display:inline" onsubmit="return confirm('Yakin hapus?')">
                @csrf
                @method('DELETE')
                <button class="btn btn-danger">Hapus</button>
              </form>
            </td>
          </tr>
        @endforeach
      </table>
    </div>
  </div>
</body>
</html>
```

Dan untuk tampilan form tambah dan edit, dapat dibuat satu halaman saja karena hampir sama. Hanya perlu tambahan pengkondisian di beberapa tempat.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Form {{ $title }} Produk</title>
    <link rel="stylesheet"
    href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css"
    integrity="sha384-zCbKRCUGaJDkqS1kPbPd7TveP5iyJE0EjAuZQTgFLD2ylzuqKfdKlfG/eSrtx
    Ukn" crossorigin="anonymous">
```

```

<script src="{{ asset('assets/jquery.js') }}"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/js/bootstrap.bundle.min.js"
integrity="sha384-fQybjgWLrvvRgtW6bFlB7jaZrFsaBXjsOMm/tB9LTS58ONXgqbR9W8oWh/amnpF" crossorigin="anonymous"></script>
</head>
<body style="width:95%">
<div class="row justify-content-center" style="margin-top:13%">
<div class="col-3">
    <h4>Form {{ $title }} Produk</h4>
    <form class="border" style="padding:20px" method="POST" action="/{{ $action }}">
        @csrf
        <input type="hidden" name="_method" value="{{ $method }}" />
        <div class="form-group">
            <label>Nama</label>
            <input type="text" name="name" class="form-control" value="{{ isset($data) ? $data->name : '' }}"/>
        </div>
        <div class="form-group">
            <label>Harga</label>
            <input type="number" name="price" class="form-control" value="{{ isset($data) ? $data->price : '' }}"/>
        </div>
        <div style="text-align:center">
            <button class="btn btn-success">Simpan</button>
        </div>
    </form>
</div>
</div>
</body>
</html>

```

7.1.5. Tampilan Halaman

Jalankan aplikasi. Jika *kode* yang dibuat sesuai dengan semua gambar di atas pada modul ini, maka tampilan aplikasi *webnya* akan seperti gambar dibawah ini.

1. <http://localhost:8000/product>



The screenshot shows a simple web interface for managing products. At the top right is a dark grey button labeled "Tambah". Below it is a table with three columns: "Nama", "Harga", and "Aksi". The table currently contains one row of data. The "Aksi" column for the first row contains a small icon, likely for deletion or edit.

Nama	Harga	Aksi
Produk A	Rp. 100.000	

Gambar 7.1 Tampilan halaman view

2. <http://localhost:8000/product/create>

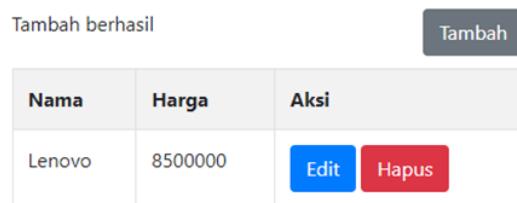
Form Tambah Produk



A screenshot of a web form titled "Form Tambah Produk". It contains two input fields: "Nama" and "Harga", each with a corresponding text input box below it. Below the input boxes is a green "Simpan" button.

Gambar 7.2 Tampilan halaman form tambah produk

3. <http://localhost:8000/product>



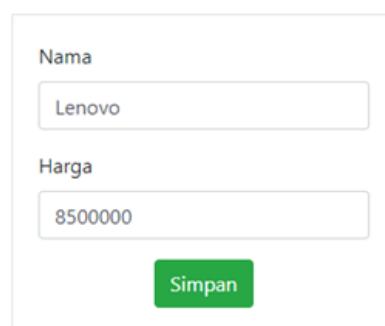
A screenshot of a product list page. At the top left is a message "Tambah berhasil". At the top right is a "Tambah" button. Below is a table with three columns: "Nama", "Harga", and "Aksi". A single row is shown for "Lenovo" with a price of "8500000". To the right of the row are "Edit" and "Hapus" buttons.

Nama	Harga	Aksi
Lenovo	8500000	<button>Edit</button> <button>Hapus</button>

Gambar 7.3 Tampilan halaman view setelah tambah data

4. [http://localhost:8000/product/\[id\]/edit](http://localhost:8000/product/[id]/edit)

Form Edit Produk



A screenshot of an edit form titled "Form Edit Produk". It contains two input fields: "Nama" and "Harga", each with a corresponding text input box below it. The "Nama" field contains "Lenovo" and the "Harga" field contains "8500000". Below the input boxes is a green "Simpan" button.

Gambar 7.4 Tampilan halaman form ubah data

7.2. Templating Halaman

Sebelumnya kita sudah menggunakan beberapa directive Blade. Directive ini juga dapat digunakan untuk templating halaman, yaitu bagian-bagian dari halaman yang sama / berulang untuk semua halaman dapat dibuat menjadi satu file Blade, dan semua halaman itu dapat me-load-nya tanpa harus ditulis ulang.

Langkah pertama dalam templating adalah membuat file template-nya, yang dimiliki oleh tiap halaman. Untuk contoh kasus pada modul ini, kita membuat file template dengan nama **template.blade.php** dengan isi:

```
<!DOCTYPE html>
<html>
  <head>
    <title>@yield('title')</title>
    <link rel="stylesheet"
      href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css"
      integrity="sha384-zCbKRCUGaJDkqS1kPbPd7TveP5iyJE0EjAuZQTgFLD2ylzuqKfdKlfG/eSrtx
      Ukn" crossorigin="anonymous">
      <script src="{{ asset('assets/jquery.js') }}></script>
      <script
        src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/js/bootstrap.bundle.min.
        js"
        integrity="sha384-fQybjgWLrvvRgtW6bF1B7jaZrFsaBXjsOMm/tB9LTS58ONXgqbR9W8oWht/am
        npF" crossorigin="anonymous"></script>
  </head>
  <body style="width:95%">
    @if(session('email'))
      <div class="row justify-content-end" style="margin-top:2%">
        <div class="col-3">
          {{ session('name') }}
          <a href="/logout" class="btn btn-warning">Logout</a>
        </div>
      </div>
    @endif
    <div class="row justify-content-center" style="margin-top:13%">
      @yield('content')
    </div>
  </body>
</html>
```

Directive **@yield** digunakan untuk menandakan bahwa di bagian itulah yang akan berbeda untuk tiap halamannya. Langkah selanjutnya adalah mengubah tampilan di tiap halaman seperti contoh **index.blade.php** di bawah ini:

```
@extends('template')

@section('title', 'Daftar Produk')

@section('content')
  <div class="col-4">
    <span class="float-left">{{ session('msg') }}</span>
    <a href="/product/create" class="btn btn-secondary
    float-right">Tambah</a><br /><br />
    <table class="table table-striped table-bordered">
      <tr>
        <th>Nama</th>
        <th>Harga</th>
        <th>Aksi</th>
      </tr>
      @foreach($list as $d)
```

```

<tr>
    <td>{{ $d->name }}</td>
    <td>{{ $d->price }}</td>
    <td>
        <a href="/product/{{ $d->id }}/edit" class="btn btn-primary">Edit</a>
        <form method="post" action="/product/{{ $d->id }}" onsubmit="return confirm('Yakin hapus?')" style="display:inline">
            @csrf
            @method('DELETE')
            <button class="btn btn-danger">Hapus</button>
        </form>
    </td>
</tr>
@endforeach
</table>
</div>
@endsection

```

Directive **@extend** digunakan untuk menentukan file template mana yang digunakan oleh halaman ini. Sedangkan directive **@section** digunakan untuk mengisi halaman sesuai dengan nama **yield** yang di-define dalam file template. Directive **@section** dapat diisi dengan string ataupun tag HTML.

7.3. Form Validation

Laravel dapat ditambahkan form validation dari sisi server. Di controller, sebelum kita memanggil fungsi untuk menambahkan (fungsi **store**) / mengubah (fungsi **update**), kita dapat menambahkan kode:

```

...
public function store(Request $request)
{
    $this->validate($request, [
        'name' => 'required|min:4',
        'price' => 'required|integer|min:1000000'
    ]);
}

public function update(Request $request)
{
    $this->validate($request, [
        'name' => 'required|min:4',
        'price' => 'required|integer|min:1000000'
    ]);
}

```

Di kode validasi ini, kita mengatur nama tidak boleh kosong dan minimal 4 karakter, sedangkan harga tidak boleh kosong, harus integer, dan nilai minimal 1.000.000. Kemudian di halaman **form.blade.php** kita harus menambahkan directive **@error** seperti ini:

```

@extends('template')

@section('title')
    Form {{ $title }} Produk
@endsection

@section('content')
    <div class="col-3">
        <h4>Form {{ $title }} Produk</h4>
        <form class="border" style="padding:20px" method="POST" action="/{{ $action }}">
            @csrf
            <input type="hidden" name="_method" value="{{ $method }}" />
            <div class="form-group">
                <label>Nama</label>

```

```

        <input type="text" name="name" class="form-control @error('name')
is-invalid @enderror" value="{{ isset($data)?$data->name:old('name') }}" />
        @error('name')
        <div class="invalid-feedback">{{ $message }}</div>
        @enderror
    </div>
    <div class="form-group">
        <label>Harga</label>
        <input type="number" name="price" class="form-control @error('price')
is-invalid @enderror" value="{{ isset($data)?$data->price:old('price') }}" />
        @error('price')
        <div class="invalid-feedback">{{ $message }}</div>
        @enderror
    </div>
    <div style="text-align:center">
        <button class="btn btn-success">Simpan</button>
    </div>
</form>
</div>
@endsection

```

Directive `@error` diletakkan di atribut class pada tag `<input>` untuk validasinya dan diletakkan di bawah tag `<input>` untuk menampilkan pesan validasinya. Kemudian agar data pada form di-repopulate, maka isikan atribut value pada tag `<input>` dengan `old('value-di-atribut-name')`.

7.4. Session

Session memiliki fungsi menyimpan suatu variabel pada server dan variabel ini dapat diakses dan diubah dimanapun: Routing, Controller, ataupun halaman manapun. Session default pada Laravel dikelola sendiri oleh Laravel dan disimpan dalam bentuk file. Selain file, Session pada Laravel juga dapat diubah menjadi disimpan ke database atau menggunakan mekanisme lain.

Laravel memiliki dua jenis Session, yaitu Session biasa dan Session flash. Session biasa akan selalu ada selama belum dihapus, time-out, atau browser ditutup. Sedangkan Session flash adalah session yang hanya berlaku untuk satu request saja, setelah itu Laravel akan menghapusnya secara otomatis. Contoh Session flash sudah pernah kita gunakan melalui fungsi `with('x', y)` yang digunakan bersamaan dengan fungsi `redirect`. Untuk Session biasa, berikut contoh penggunaannya di Routing:

```

...
Route::get('/login', function () {
    if (session()->has('email')) return redirect('/product');
    return view('login');
});

Route::get('/logout', function () {
    session()->flush();
    return redirect('/login');
});
...

```

Fungsi `session()->has('x')` digunakan untuk memeriksa apakah ada Session bernama "x". Sedangkan `session()->flush()` digunakan untuk menghapus semua Session. Contoh lain, berikut penggunaan Session di SiteController:

```

...
public function auth(Request $req) {
    $u = User::where([
        ['email', $req->em],
        ['password', $req->pwd],
    ])->first();
    if (isset($u)) {
        session()->put('email', $u->email);
    }
}

```

```

session()->put('name', $u->name);
return "<script>
alert('Welcome, " . session('name') . "');
location.href='/product';
</script>";
}
return redirect('/login')->with('msg', 'Email / password salah');
}
...

```

Fungsi `session()->put('x', y)` digunakan untuk membuat Session bernama “x” dengan nilai y. Sedangkan `session('x')` digunakan untuk mengambil nilai dari Session “x”.

7.5. Middleware

Middleware pada Laravel adalah suatu mekanisme untuk menyaring request yang masuk ke suatu Routing. Sebagai contoh, kita bisa membuat Middleware untuk memverifikasi apakah seorang user sudah terautentikasi atau memiliki hak akses untuk mengakses URL. Jika user belum terautentikasi atau tidak memiliki akses, maka Middleware dapat me-redirect user tersebut ke URL lain. Sebaliknya jika user terautentikasi atau memiliki hak akses, maka Middleware akan meneruskan request ke aksi yang dipetakan di Routing.

Middleware dapat melakukan hal lain selain untuk autentikasi, misalnya untuk menulis log atau error yang terjadi. Dan Middleware dapat kita atur mekanisme yang dilakukannya, yaitu bisa sebelum request diteruskan atau sesudah request diteruskan. Middleware yang kita buat harus berada di folder `app/Http/Middleware`.

Sekarang kita akan membuat autentikasi dengan menggunakan library **Auth**. Library ini sudah termasuk di dalam paket instalasi Laravel, sehingga kita tidak perlu menambahkannya lagi melalui composer. Dalam library ini sudah mencakup semua fitur untuk autentikasi, registrasi, dan middleware-nya. Dengan menggunakan **Auth**, autentikasi menjadi lebih simpel, aman, dan bisa menjadi alternatif pengganti yang lebih baik dari Session.

Langkah pertama adalah pengaturan pada Routing. Ganti Routing untuk login, logout, dan product:

```

...
Route::get('/login', function () {
    if (Auth::check()) return redirect('/product');
    return view('login');
})->name('login');

Route::get('/logout', function () {
    Auth::logout();
    return redirect('/login');
});

Route::resource('product', ProductController::class)->middleware('auth');

```

Fungsi `Auth::check()` digunakan untuk memeriksa apakah user telah terautentikasi. Routing login kita berikan nama alias karena kebutuhan dari Middleware **auth**. Sedangkan `Auth::logout()` digunakan untuk menghapus autentikasi. Untuk product, Middleware kita ganti dengan Middleware **auth** yang berkolaborasi dengan library Auth.

Lalu langkah kedua ubah isi dari fungsi **auth** di **SiteController**:

```

...
use Illuminate\Support\Facades\Auth;

```

```

class SiteController extends Controller
{
    public function auth(Request $req) {
        if (Auth::attempt(['email'=>$req->em, 'password'=>$req->pwd])) {
            //jika ada nilai lain selain data user yang ingin disimpan di session, baru
            //gunakan session disini
            return redirect('/product');
        }
        return redirect('/login')->with('msg', 'Email / password salah');
    }
}
...

```

Fungsi **Auth::attempt()** digunakan untuk proses autentikasi, yaitu apakah email dan password yang di-submit ada dalam database pada tabel **users**. Jika email dan password cocok maka akan menghasilkan **true**. Hal yang harus diperhatikan adalah ketika menambahkan data User ke database, pastikan isi dari kolom password di-hash dengan metode **Bcrypt** agar dapat menggunakan library Auth ini. Laravel sudah menyediakan fungsi **bcrypt('x')** untuk mempermudahnya.

Langkah ketiga, dalam View yang sebelumnya menggunakan **@if(session('x'))** dapat kita ganti menjadi directive **@auth** untuk pengecekan apakah user telah terautentikasi, dan dapat menggunakan **Auth::user()** untuk mengakses data user yang login. Berikut contohnya dalam **template.blade.php**:

```

...
<body style="width:95%">
@auth
<div class="row justify-content-end" style="margin-top:2%">
    <div class="col-3">
        {{ Auth::user() ->name }}
        <a href="/logout" class="btn btn-warning">Logout</a>
    </div>
</div>
@endauth
<div class="row justify-content-center" style="margin-top:10%">
    @yield('content')
...

```

7.6 Model Relasi

Model Eloquent menyediakan fasilitas agar dua Model yang berelasi dapat langsung memanggil satu sama lain. Kita akan mencoba salah satunya yaitu relasi one to many, dengan menggunakan contoh kasus relasi **Product** dengan **Variant**. Tiap **Product** dapat memiliki banyak **Variant**, tetapi tiap **Variant** hanya dimiliki oleh satu **Product**. Langkah awal yaitu membuat Model dan file migration-nya, maka perintahnya:

```
php artisan make:model Variant -m
```

Setelah itu edit file **xxx_create_variants_table.php** pada folder **database/migrations** untuk mendefinisikan kolomnya:

```

...
Schema::create('variants', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->text('description');
    $table->string('processor');
    $table->string('memory');
    $table->string('storage');
    $table->foreignId('product_id')->constrained();
    $table->timestamps();
});

```

```
});
```

```
...
```

Untuk kebutuhan foreign key, kita menggunakan **foreignId** dan **constrained** jika tabel entitas yang diacu mengikuti konvensi Model Eloquent (PK bernama "id", tabel database ditambah akhiran "s", dsb). Jika berbeda, maka pendefinisian foreign key harus menggunakan **foreign**, **references**, dan **on**. Contoh:

```
...





```

Kemudian generate tabel **variants** dengan perintah seperti sebelumnya (php artisan migrate). Selanjutnya edit file Model **Variants**, tambahkan fungsi berikut:

```
...
public function product() {
    return $this->belongsTo(Product::class);
}
...
```

Fungsi **belongsTo()** secara otomatis akan memanggil object **Product** yang berelasi dengan dirinya berdasarkan **product_id**. Kemudian tambahkan juga fungsi ke dalam Model **Product**:

```
...
public function variants() {
    return $this->hasMany(Variant::class);
}
...
```

Fungsi **hasMany()** secara otomatis akan memanggil semua object **Variant** yang berelasi dengan dirinya berdasarkan **product_id**. Untuk mencobanya kita akan menambahkan satu kolom pada tampilan tabel di **index.blade.php** untuk menampilkan data variant dari tiap product:

```
...
<th>Harga</th>
<th>Variant</th>
<th>Aksi</th>
</tr>
@foreach($list as $d)
<tr>
    <td>{{ $d->name }}</td>
    <td>{{ $d->price }}</td>
    <td>
        <ul>
            @foreach($d->variants()->get() as $var)
                <li>{{ $var->name }}</li>
                Desc: {{ $var->description }} <br />
                Proc: {{ $var->processor }} <br />
                RAM: {{ $var->memory }} <br />
                Strg: {{ $var->storage }} <br />
                Product: {{ $var->product->name }}
            @endforeach
        </ul>
    </td>
    <td align="center">
```

Tiap data product dapat langsung mengakses semua data variant-nya dengan fungsi **variants()** dan tiap variant dapat mengakses product yang berelasi dengannya dengan menggunakan atribut **product**. Uji dengan memasukkan data **variants** di database dan membuat form untuk menambahkan data **variants** ke database.

MODUL 8. PENGENALAN DART

Tujuan Praktikum

1. Mahasiswa mampu memahami konsep layout pada Flutter
2. Mahasiswa dapat mengimplementasikan desain user interface pada Flutter

8.1. Pengenalan Dart

Untuk belajar flutter, tidak perlu terlalu fasih untuk mempelajari bahasa dart. Terdapat fundamental yang perlu dipelajari seperti variable, statement control, looping, array, fungsi, dsb. Karakteristik bahasa dart mirip dengan bahasa C ataupun Java. Wajib menggunakan titik koma diakhir codingan.

8.1.1. Variable

Untuk penggunaan variable di dart, terdapat beberapa cara, yaitu dengan var, type annotation dan multiple variable.

```
// var  
var <variable_name>;  
var <name> = <expression>;  
  
// type annotation  
<type> <variable_name>;  
<type> <name> = <expression>;  
  
// multiple variable  
<type> <var1,var2...varN>;
```

Variable primitif yang tersedia di dart :

1. Integer
2. Double
3. String
4. Boolean

8.1.2. Statement Control

Terdapat beberapa cara untuk mendeklarasikan statement control, yaitu if, if else, if else if, switch case.

IF Statement

```
// IF STATEMENT  
if(condition){  
    // statements
```

```
}
```

IF ELSE Statement

```
// IF ELSE STATEMENT
if(condition){
    // statements
} else {
    // statements
}
```

IF ELSE IF Statement

```
// IF ELSE IF STATEMENT
if(condition1) {
    // statement(s)
}
else if(condition2){
    // statement(s)
}
.
.
else if(conditionN){
    // statement(s)
}
else {
    // statement(s)
}
```

SWITCH CASE Statement

```
// SWITCH CASE
switch(expression){
    case value1: {
        // statements
    }
    break;
    case value2: {
        // statements
    }
    break;
    default: {
        // statements
    }
    break;
}
```

8.1.3. Looping

Secara umum, terdapat dua cara untuk melakukan looping di dart, yaitu menggunakan for loop dan while loop.

For Loops

Gunakan for loop saat kondisinya tau persis seberapa banyak looping akan dilakukan, contohnya melakukan perulangan sebanyak 10 kali dengan iterasi sebanyak 1 tingkat atau 1 kali.

```
for (initial_count_value; termination-condition; step) {  
    //statements  
}
```

While Loops

Gunakan while loop saat kondisinya tidak tahu kapan perulangan akan berhenti, contohnya sediakan input angka hingga user menginput tanda "-".

```
while (expression) {  
    // Statement(s) to be executed if expression is true  
}
```

8.1.4. List

Secara umum, kumpulan banyak data dalam satu variable disebut array. Tetapi beberapa bahasa pemrograman menyebutnya dengan list, termasuk bahasa dart ini. List memiliki 2 tipe, yaitu Fixed Length List dan Growable List.

Fixed Length List

Dari namanya bisa diketahui bahwa tipe list ini memiliki panjang index yang tetap dan tidak dapat bertambah banyak.

```
// Mendeklarasikan list  
var list_name = new List(initial_size);  
  
// Menginisialisasikan list  
list_name[index] = value;  
  
// Contohnya  
var newList = new List(3);  
newList[0] = 12;  
newList[1] = 13;  
newList[2] = 11;
```

Growable List

Gunakan growable list apabila memiliki banyak object yang tidak menentu atau banyaknya object yang terus bertambah.

```
// Mendeklarasikan list
var list_name = new List();

// Menginisialisasikan list
list_name[index] = value;

// Contohnya
var newList = new List(3);
newList[0] = 12;
newList[1] = 13;
newList[2] = 11;
```

8.1.5. Fungsi

Pada bahasa pemrograman yang mendukung Object Oriented Programming, fungsi atau prosedur memiliki peranan yang sangat penting. Untuk menghasilkan kualitas kode yang sangat baik, programmer bisa menggunakan beberapa prinsip pemrograman yang umum digunakan seperti SOLID, KISS, YAGNI, dsb. Semua prinsip tersebut menjunjung tinggi separation of concern yang artinya setiap kodingan memiliki tanggung jawabnya sendiri dan mengurangi sebanyak mungkin boilerplate code.

Mendefinisikan Fungsi

```
void function_name() {
    //statements
}
```

Memanggil Fungsi

```
void main() {
    print(factorial(6));
}
```

Mengembalikan Nilai

Tambahkan return apabila anda mendefinisikan sebuah fungsi, contohnya ada pada codingan dibawah yang bisa mengembalikan nilai faktorial dari angka yang sudah ditentukan.

```
factorial(number) {
    if (number <= 0) {
        // termination case
```

```
    return 1;
} else {
    return (number * factorial(number - 1));
    // function invokes itself
}
}
```

Menambahkan Parameter

Fungsi memiliki scope yang terbatas, tentunya fungsi butuh input dari luar agar program didalamnya bisa memproses tugasnya.

```
factorial(number) {
    if (number <= 0) {
        // termination case
        return 1;
    } else {
        return (number * factorial(number - 1));
        // function invokes itself
    }
}
```

Pada fungsi diatas, number merupakan parameter. Variable diluar fungsi yang dibuat agar dapat digunakan didalam fungsi.

MODUL 9. FLUTTER LAYOUT DASAR

Tujuan Praktikum

3. Mahasiswa mampu memahami konsep layout pada Flutter
4. Mahasiswa dapat mengimplementasikan desain user interface pada Flutter

9.1. Pengenalan Widget

Flutter dibuat oleh google yang terinspirasi oleh Reactjs. Pada dasarnya semua tampilan akan dipecah menjadi komponen-komponen yang kecil dan memiliki environment sendiri untuk mengelola dirinya. Komponen tersebut dinamai Widget pada Flutter. Masing-masing widget memiliki state dan konfigurasinya sendiri, sehingga ketika state pada widget berubah, widget akan membuat ulang dirinya agar selalu update dengan perubahan yang terjadi.

Berikut contoh penerapan HelloWorld pada Flutter:

```
import 'package:flutter/material.dart';

void main() {
  runApp(
    const Center(
      child: Text(
        'Hello, world!',
        textDirection: TextDirection.ltr,
      ),
    ),
  );
}
```

9.2. Container

Layout pertama yang harus dipahami adalah Container. Container merupakan *widget* untuk membuat elemen visual seperti kotak. Container dapat didekorasi menggunakan BoxDecoration seperti *background*, *border*, atau *shadow*. Container memiliki *margin* dan *padding* untuk memberikan jarak antara komponen satu dengan komponen yang lainnya. Berikut cara penerapan Container pada Flutter:

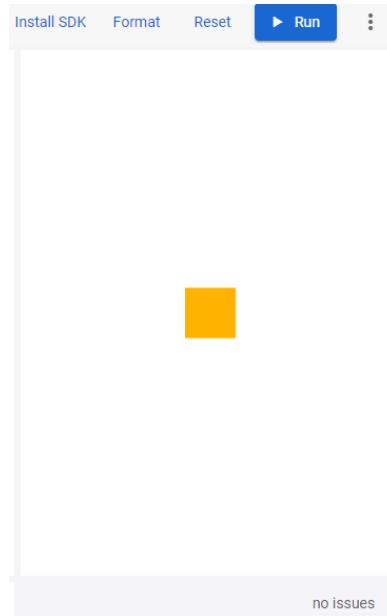
Tambahkan kode berikut pada:

```
import 'package:flutter/material.dart';

void main() {
  runApp(
    Center(
      child: Container(
        margin: const EdgeInsets.all(10.0),
        color: Colors.amber[600],
        width: 48.0,
        height: 48.0,
      ),
    ),
  );
}
```

```
    ),  
    );  
}
```

Nanti hasilnya akan seperti ini:



Gambar 9.1 Tampilan Container

9.3. GridView

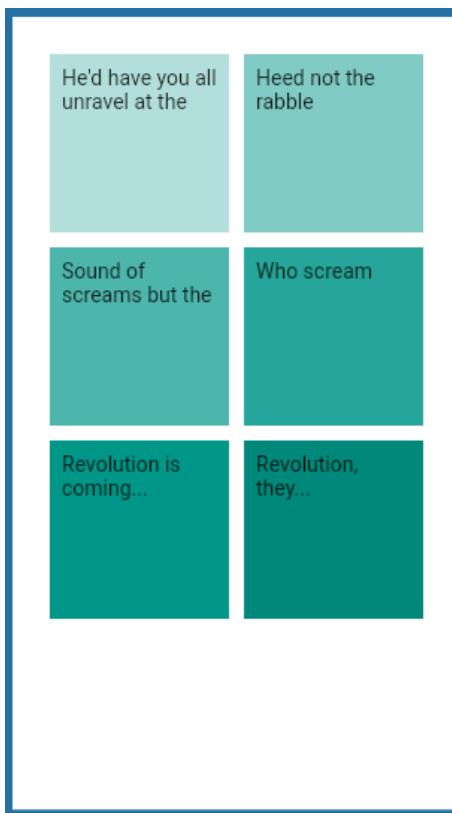
GridView merupakan widget yang serupa dengan Array 2D dalam bahasa pemrograman apapun. Widget tersebut digunakan ketika harus menampilkan sesuatu pada Grid tersebut, seperti menampilkan *images*, *text*, *icons*, dll. Berikut contoh penerapan GridView:

Tambahkan kode berikut di dalam kurung runApp()

```
GridView.count(  
    primary: false,  
    padding: const EdgeInsets.all(20),  
    crossAxisSpacing: 10,  
    mainAxisSpacing: 10,  
    crossAxisCount: 2,  
    children: <Widget>[  
        Container(  
            padding: const EdgeInsets.all(8),  
            child: const Text("He'd have you all unravel at the"),  
            color: Colors.teal[100],  
        ),  
        Container(  
            padding: const EdgeInsets.all(8),  
            child: const Text('Heed not the rabble'),  
            color: Colors.teal[200],  
        ),
```

```
Container(  
  padding: const EdgeInsets.all(8),  
  child: const Text('Sound of screams but the'),  
  color: Colors.teal[300],  
,  
Container(  
  padding: const EdgeInsets.all(8),  
  child: const Text('Who scream'),  
  color: Colors.teal[400],  
,  
Container(  
  padding: const EdgeInsets.all(8),  
  child: const Text('Revolution is coming...'),  
  color: Colors.teal[500],  
,  
Container(  
  padding: const EdgeInsets.all(8),  
  child: const Text('Revolution, they...'),  
  color: Colors.teal[600],  
,  
],  
)
```

Hasilnya akan seperti berikut:



Gambar 9.2 Tampilan Penerapan GridView

9.4. ListView

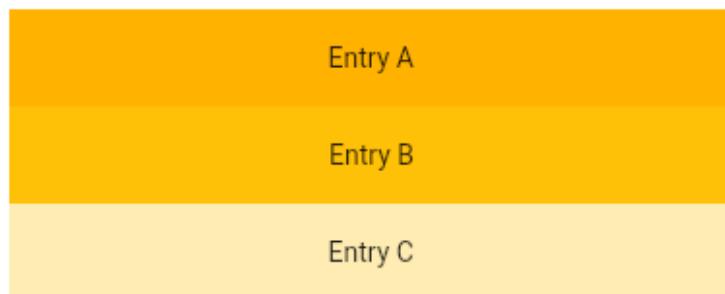
ListView merupakan *widget scroll* yang paling umum digunakan. *Widget* ini dapat menampilkan lebih dari satu komponen atau *widget* melalui variabel *children*.

Pada pembahasan kali ini akan menggunakan ListView default dengan variabel *children* pada widget tersebut List<Widget>. Cara penggunaan ListView ini dengan memasukkan widget yang ingin disusun sebagai children dari ListView.

Berikut contoh penerapan ListView:

```
ListView(  
  padding: const EdgeInsets.all(8),  
  children: <Widget>[  
    Container(  
      height: 50,  
      color: Colors.amber[600],  
      child: const Center(child: Text('Entry A')),  
    ),  
    Container(  
      height: 50,  
      color: Colors.amber[500],  
      child: const Center(child: Text('Entry B')),  
    ),  
    Container(  
      height: 50,  
      color: Colors.amber[100],  
      child: const Center(child: Text('Entry C')),  
    ),  
  ],  
)
```

Ketika dijalankan, maka akan tampil seperti ini



Gambar 9.3 Tampilan Penerapan ListView

9.4.1. ListView.builder

Widget ini cocok digunakan ketika memiliki data list yang lebih besar. ListView.builder membutuhkan itemBuilder dan itemCount. Parameter itemBuilder merupakan fungsi yang mengembalikan widget untuk ditampilkan. Sedangkan itemCount kita isi dengan jumlah seluruh item yang ingin ditampilkan.

Berikut ini adalah contoh penerapan ListView.builder

```
final List<String> entries = <String>['A', 'B', 'C'];
final List<int> colorCodes = <int>[600, 500, 100];

ListView.builder(
  padding: const EdgeInsets.all(8),
  itemCount: entries.length,
  itemBuilder: (BuildContext context, int index) {
    return Container(
      height: 50,
      color: Colors.amber[colorCodes[index]],
      child: Center(child: Text('Entry ${entries[index]}')),
    );
  }
);
```

Hasil dan tampilan nya akan sama dengan sebelumnya. Karena ListView.builder adalah cara ketika memiliki data yang besar dan tidak ingin menggunakan kode yang sama secara berulang.

9.4.2. ListView.separated

ListView jenis ini akan menampilkan daftar item yang dipisahkan dengan separator. Penggunaan ListView.separated mirip dengan builder, yang membedakan adalah terdapat satu parameter tambahan wajib yaitu separatorBuilder yang mengembalikan Widget yang akan berperan sebagai separator.

Berikut contoh penerapan ListView.separated:

```
final List<String> entries = <String>['A', 'B', 'C'];
final List<int> colorCodes = <int>[600, 500, 100];

ListView.separated(
  padding: const EdgeInsets.all(8),
  itemCount: entries.length,
  itemBuilder: (BuildContext context, int index) {
    return Container(
      height: 50,
      color: Colors.amber[colorCodes[index]],
      child: Center(child: Text('Entry ${entries[index]}')),
    );
  },
  separatorBuilder: (BuildContext context, int index) => const Divider(),
);
```

Maka hasilnya akan seperti ini:



Gambar 9.4 Tampilan Penerapan ListView.Separated

9.5. Stack

Widget ini merupakan widget yang saling tumpang tindih terhadap widget lain. Seperti *image* dan *text* yang saling bertumpuk, atau overlay yang terdapat button dan widget lainnya.

Dengan menggunakan Stack dapat memposisikan widget satu sama lain dan bertumpukan antar widget.

Berikut contoh penerapan Stack:

```
Stack(  
  children: <Widget>[  
    Container(  
      width: 100,  
      height: 100,  
      color: Colors.red,  
    ),  
    Container(  
      width: 90,  
      height: 90,  
      color: Colors.green,  
    ),  
    Container(  
      width: 80,  
      height: 80,  
      color: Colors.blue,  
    ),  
  ],  
)
```

Maka hasilnya akan seperti ini:

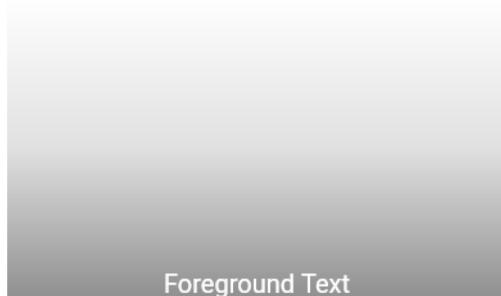


Gambar 9.5 Tampilan Penerapan Stack

Selanjutnya adalah penerapan Stack dengan text dan ditambahkan dengan background gradient di belakangnya. Berikut contoh penerapannya:

```
SizedBox(  
    width: 250,  
    height: 250,  
    child: Stack(  
        children: <Widget>[  
            Container(  
                width: 250,  
                height: 250,  
                color: Colors.white,  
            ),  
            Container(  
                padding: const EdgeInsets.all(5.0),  
                alignment: Alignment.bottomCenter,  
                decoration: BoxDecoration(  
                    gradient: LinearGradient(  
                        begin: Alignment.topCenter,  
                        end: Alignment.bottomCenter,  
                        colors: <Color>[  
                            Colors.black.withOpacity(0),  
                            Colors.black12,  
                            Colors.black45  
                        ],  
                    ),  
                ),  
                child: const Text(  
                    'Foreground Text',  
                    style: TextStyle(color: Colors.white, fontSize: 20.0),  
                ),  
            ),  
        ],  
    ),  
)
```

Maka hasilnya akan seperti berikut:



Gambar 9.6 Tampilan Background Gradient

MODUL 10. FLUTTER LAYOUT LANJUT

Tujuan Praktikum

1. Mahasiswa mampu memahami konsep layout lanjutan pada Android
2. Mahasiswa dapat mengimplementasikan desain user interface kompleks pada Android

10.1. Row

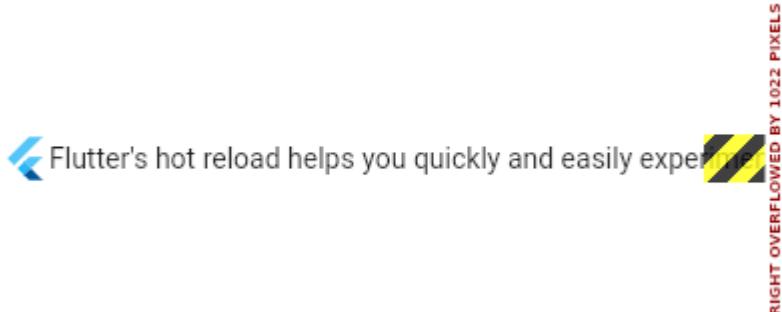
Row merupakan suatu widget yang digunakan untuk membuat widget-widget tersusun berjejer secara horizontal. Row memiliki sintaks seperti berikut:

```
Row(  
  children: <Widget>[  
    //widget code  
  ],  
)
```

Parameter children berisi kumpulan atau list dari widget karena kita dapat menyusun beberapa widget sekaligus di dalamnya. Jika mengacu pada contoh tombol-tombol di atas kodennya seperti berikut:

```
Row(  
  children: <Widget>[  
    const FlutterLogo(),  
    const Expanded(  
      child: Text("Flutter's hot reload helps you quickly and easily experiment, build UIs, add  
      features, and fix bugs faster. Experience sub-second reload times, without losing state, on  
      emulators, simulators, and hardware for iOS and Android."),  
    ),  
    const Icon(Icons.sentiment_very_satisfied),  
  ],  
)
```

Jika kita jalankan, maka hasilnya akan tampil seperti ini:



Gambar 10.1 Tampilan Overflowed di Flutter

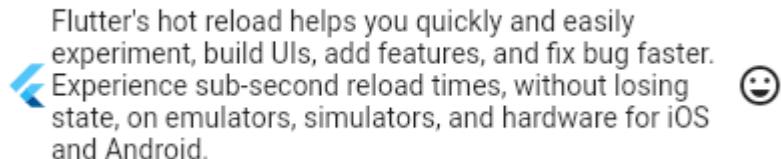
Terdapat permasalahan yang terjadi ketika menggunakan widget Row dengan data yang banyak atau data yang panjang. Dalam kasus diatas, widget text memiliki data text yang panjang dan menampilkan gambar "yellow & black" yang menunjukkan bahwa tidak adanya ruang atau melebihi ruang kosong yang tersedia.

Untuk mengatasi hal tersebut yaitu dengan menggunakan widget Expanded, yang dimana widget tersebut akan memberikan ruang kosong yang tersisa.

Tambahkan Expanded pada kode yang telah dibuat sebelumnya, seperti berikut:

```
Row(  
  children: <Widget>[  
    const FlutterLogo(),  
    const Expanded(  
      child: Text("Flutter's hot reload helps you quickly and easily experiment, build UIs, add  
      features, and fix bug faster. Experience sub-second reload times, without losing state, on  
      emulators, simulators, and hardware for iOS and Android."),  
    ),  
    const Icon(Icons.sentiment_very_satisfied),  
  ],  
)
```

Maka hasilnya akan menjadi seperti ini:



Gambar 10.2 Tampilan Penggunaan Expanded

10.2. Column

Column merupakan suatu widget yang digunakan untuk membuat widget-widget tersusun berjajar secara vertikal. Column memiliki sintaks mirip dengan Row, seperti berikut:

```
Column(  
  children: <Widget>[  
    //widget code  
  ]  
)
```

Dan berikut adalah contoh penerapan Column:

```
Column(
```

```
children: const <Widget>[
    Text('Deliver features faster'),
    Text('Craft beautiful UIs'),
    Expanded(
        child: FittedBox(
            fit: BoxFit.contain, // otherwise the logo will be tiny
            child: FlutterLogo(),
        ),
    ),
],
)
```

Maka tampilannya akan seperti ini:



Gambar 10.3 Tampilan Penerapan Column

Jika dilihat kembali, secara default tampilan tersebut memiliki alignment rata tengah. Untuk membuat tampilan tersebut rata kiri atau kanan, maka kita bisa menambahkan kode crossAxisAlignment dan mainAxisSize dengan penerapan kode seperti di bawah ini:

```
Column(
    mainAxisAlignment: MainAxisAlignment.start,
    mainAxisSize: MainAxisSize.min,
    children: <Widget>[
        const Text('We move under cover and we move as one'),
        const Text('Through the night, we have one shot to live another day'),
        const Text('We cannot let a stray gunshot give us away'),
        const Text('We will fight up close, seize the moment and stay in it'),
        const Text('It's either that or meet the business end of a bayonet'),
        const Text('The code word is 'Rochambeau,' dig me?'),
        Text('Rochambeau!', style: DefaultTextStyle.of(context).style.apply(fontSizeFactor: 2.0)),
    ],
)
```

Maka hasilnya akan seperti ini:

We move under cover and we move as one
Through the night, we have one shot to live another day
We cannot let a stray gunshot give us away
We will fight up close, seize the moment and stay in it
It's either that or meet the business end of a bayonet
The code word is 'Rochambeau,' dig me?

Rochambeau!

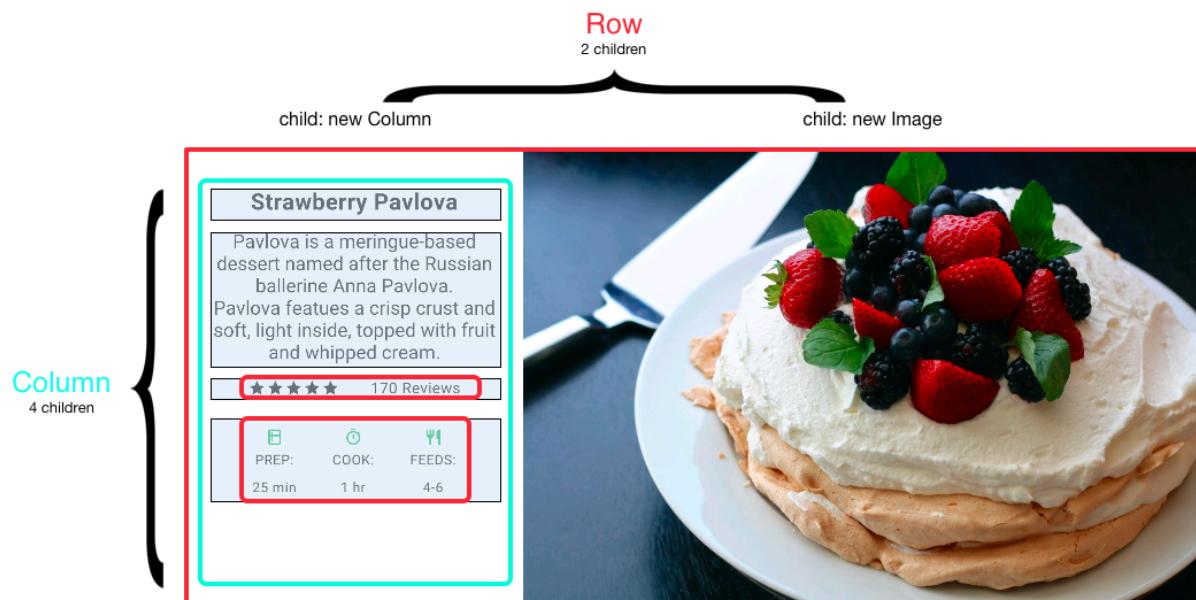
Gambar 10.4 Tampilan Penerapan crossAxisAlignment dan mainAxisSize

10.3. Nested Rows & Columns

Salah satu hal yang paling mendasar ketika membuat layout adalah mengaturnya secara horizontal dan vertikal. Untuk mengatasi hal tersebut bisa menggunakan widget Row dan Column secara bersamaan.

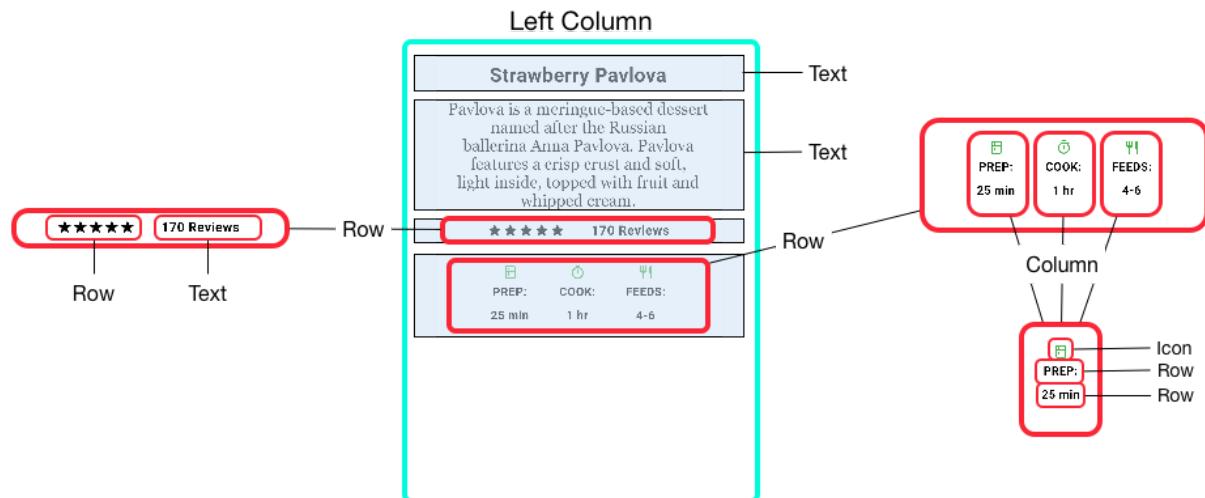
Untuk membuat Row atau Column, bisa ditambahkan pada children di setiap widget Row ataupun Column. Contoh berikut ini akan menunjukkan bahwa memungkinkan untuk membuat Row atau Column di dalam widget Row atau Column.

Layout dibawah ini disusun dengan Row, yang dimana Row memiliki 2 widget Column di sebelah kiri dan Image di sebelah kanan.



Gambar 10.5 Tampilan Penerapan Nested Rows dan Column

Pada widget Column di sebelah kiri memiliki nested Rows dan Columns.



Gambar 10.6 Tampilan Penerapan Nested Rows dan Column 2

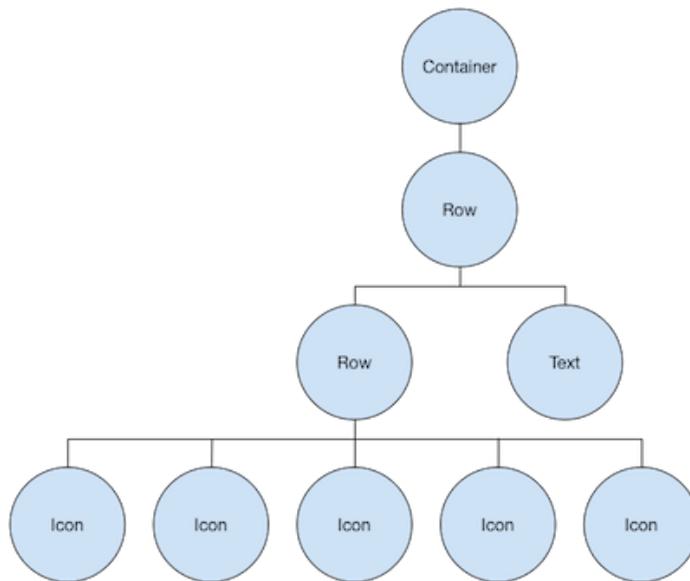
Pada pembahasan kali ini akan membuat layout yang ada pada gambar di bawah ini.



Gambar 10.7 Tampilan Penerapan Nested Rows dan Column 3

Hal pertama pada pembahasan kali ini akan membuat 2 baris yang ada pada container warna merah, meliputi 1 baris yang berisi rating dan jumlah ulasan, lalu 1 baris yang berisi 3 icons dan text.

Widget tree nya adalah sebagai berikut:



Gambar 10.8 Widget Tree

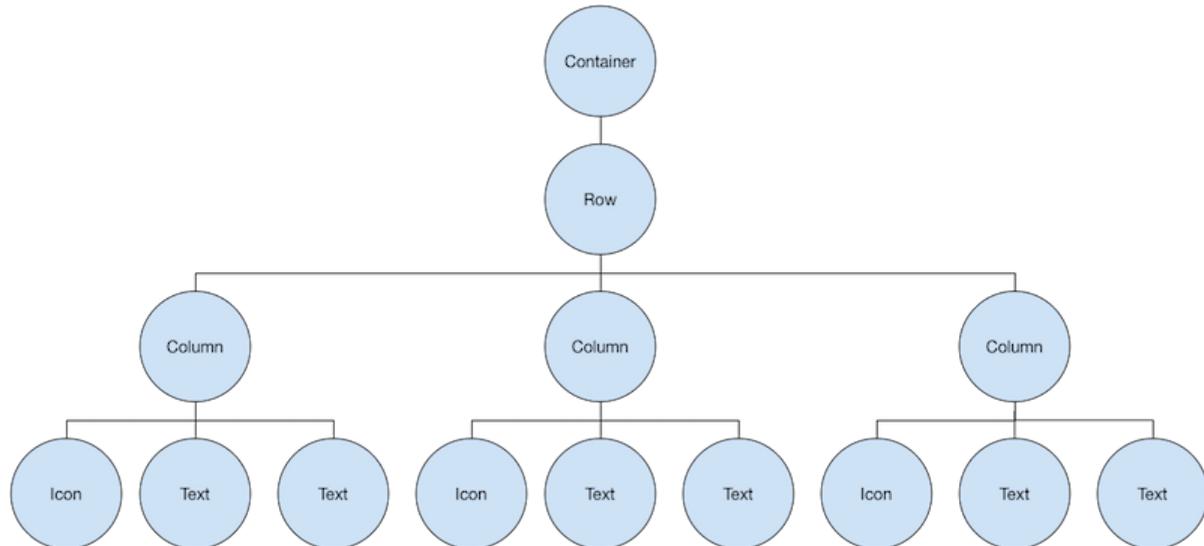
Rating pada layout diatas meliputi Row yang memiliki Row untuk icons bintang dan text:

```
var stars = Row(
  mainAxisAlignment: MainAxisAlignment.min,
  children: [
    Icon(Icons.star, color: Colors.green[500]),
    Icon(Icons.star, color: Colors.green[500]),
    Icon(Icons.star, color: Colors.green[500]),
    const Icon(Icons.star, color: Colors.black),
    const Icon(Icons.star, color: Colors.black),
  ],
);

final ratings = Container(
  padding: const EdgeInsets.all(20),
  child: Row(
    mainAxisAlignment: MainAxisAlignment.spaceEvenly,
    children: [
      stars,
      const Text(
        '170 Reviews',
        style: TextStyle(
          color: Colors.black,
          fontWeight: FontWeight.w800,
          fontFamily: 'Roboto',
          letterSpacing: 0.5,
          fontSize: 20,
        ),
      ),
    ],
),
```

```
 ),  
 );
```

Pada bagian bawah rating, memiliki 3 Columns yang dimana masing-masing Column meliputi icon dan dua baris text, berikut widget tree nya:



Gambar 10.9 Widget Tree dengan 3 Column

Lalu buat iconList untuk mendefinisikan icons tiap row:

```
const descTextStyle = TextStyle(  
  color: Colors.black,  
  fontWeight: FontWeight.w800,  
  fontFamily: 'Roboto',  
  letterSpacing: 0.5,  
  fontSize: 18,  
  height: 2,  
,);  
  
// DefaultTextStyle.merge() allows you to create a default text  
// style that is inherited by its child and all subsequent children.  
final iconList = DefaultTextStyle.merge(  
  style: descTextStyle,  
  child: Container(  
    padding: const EdgeInsets.all(20),  
    child: Row(  
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,  
      children: [  
        Column(  
          children: [  
            Icon(Icons.kitchen, color: Colors.green[500]),  
            const Text('PREP:'),  
            const Text('25 min'),  
          ],  
        ),  
      ],  
    ),  
  ),  
);
```

```

    ],
),
Column(
children: [
Icon(Icons.timer, color: Colors.green[500]),
const Text('COOK:'),
const Text('1 hr'),
],
),
Column(
children: [
Icon(Icons.restaurant, color: Colors.green[500]),
const Text('FEEDS:'),
const Text('4-6'),
],
),
],
),
),
);

```

Lalu buat leftColumn yang meliputi ratings dan iconList yang telah dibuat tadi, untuk titleText dan subTitle bisa diisi dengan teks seperti pada gambar diatas. Berikut baris kodennya:

```

final leftColumn = Container(
padding: const EdgeInsets.fromLTRB(20, 30, 20, 20),
child: Column(
children: [
titleText,
subTitle,
ratings,
iconList,
],
),
);

```

Untuk leftColumn akan berada pada widget SizedBox untuk membatasi lebar dari konten tersebut. Hasilnya, keseluruhan UI akan memiliki Column di sisi kiri dan Image di sisi kanan. Berikut kodennya:

```

body: Center(
child: Container(
margin: const EdgeInsets.fromLTRB(0, 40, 0, 30),
height: 600,
child: Card(
child: Row(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
SizedBox(
width: 440,
child: leftColumn,

```

```
        ),
        mainImage,
    ],
),
),
),
),
),
```

Untuk melihat keseluruhan kodennya, bisa diakses melalui link [ini](#).

10.4. CustomScrollView

Widget ini memungkinkan membuat efek pada list, grid, maupun header yang lebar. Misalnya, ketika ingin membuat scroll view yang berisi app bar yang lebar yang meliputi list dan grid secara bersamaan, maka bisa menggunakan 3 widget sliver, yaitu SliverAppBar, SliverList, dan SliverGrid.

Berikut adalah barisan kode yang menunjukkan scroll view dengan meliputi app bar yang fleksibel, grid dan infinite list.

```
CustomScrollView(
  slivers: <Widget>[
    const SliverAppBar(
      pinned: true,
      expandedHeight: 250.0,
      flexibleSpace: FlexibleSpaceBar(
        title: Text('Demo'),
      ),
    ),
    SliverGrid(
      gridDelegate: const SliverGridDelegateWithMaxCrossAxisExtent(
        maxCrossAxisExtent: 200.0,
        mainAxisSpacing: 10.0,
        crossAxisSpacing: 10.0,
        childAspectRatio: 4.0,
      ),
      delegate: SliverChildBuilderDelegate(
        (BuildContext context, int index) {
          return Container(
            alignment: Alignment.center,
            color: Colors.teal[100 * (index % 9)],
            child: Text('Grid Item $index'),
          );
        },
        childCount: 20,
      ),
    ),
    SliverFixedExtentList(
      itemExtent: 50.0,
```

```
delegate: SliverChildBuilderDelegate(  
   (BuildContext context, int index) {  
    return Container(  
        alignment: Alignment.center,  
        color: Colors.lightBlue[100 * (index % 9)],  
        child: Text('List Item $index'),  
    );  
},  
),  
],  
)
```

MODUL 11. PACKAGES & USER INTERACTION

Tujuan Praktikum

1. Mahasiswa mampu memahami konsep user interaction pada Android
2. Mahasiswa dapat mengimplementasikan berbagai user interaction pada Android

11.1. Packages

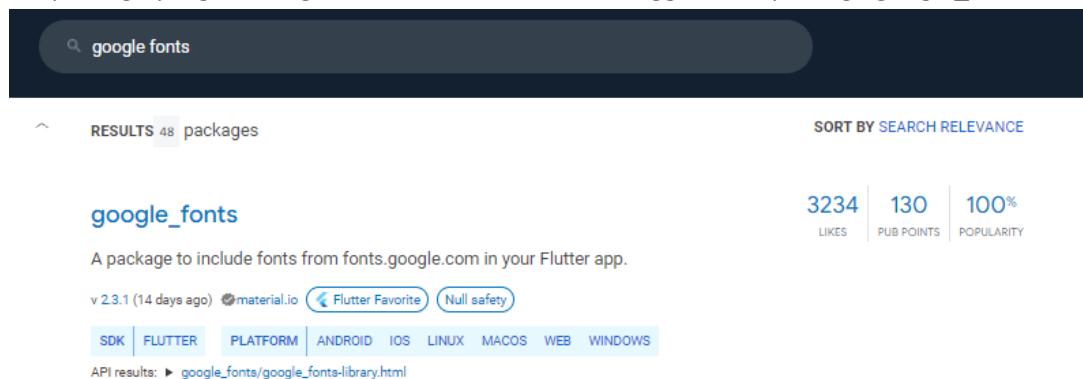
11.1.1. Pengenalan Packages

Secara singkat, dart package terdapat pada direktori yang didalamnya terdapat file `pubspec`. Contoh penggunaan packages adalah membuat request ke server menggunakan protokol [http](#), custom navigation/route handling menggunakan [fluro](#), dsb.

11.1.2. Penggunaan Packages

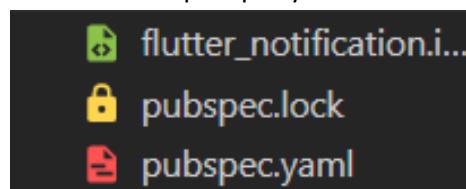
Untuk penggunaan package, silahkan ikuti langkah-langkah berikut ini dengan menggunakan contoh yang akan kita pakai nantinya:

1. Akses website pub.dev melalui browser
2. Cari package yang mau digunakan, disini kita akan menggunakan package `google_fonts`



Gambar 11.1 Google Fonts

3. Buka folder project, lalu cari file bernama `pubspec.yaml`



Gambar 11.2 Yaml file

4. Tambahkan `google_fonts` dibawah dependencies

```
dependencies:  
  flutter:  
    |  sdk: flutter  
    |  google_fonts: ^2.3.1|
```

Gambar 11.3 Depedencies

5. Lalu save dengan cara CTRL + S pada keyboard atau klik tombol run pada pojok kanan atas
6. Tunggu hingga proses pub get selesai
7. Untuk menggunakannya, import package tersebut pada file Dart.

11.2. User Interaction

11.2.1. Stateful & Stateless

Widget stateless tidak pernah berubah. Ikon, IconButton, dan Teks adalah contoh widget stateless. Sub kelas widget stateless StatelessWidget. Widget stateful bersifat dinamis misalnya, ia dapat mengubah tampilannya sebagai respons terhadap peristiwa yang dipicu oleh interaksi pengguna atau saat menerima data. Kotak centang, Radio, Slider, InkWell, Form, dan TextField adalah contoh widget stateful. Subkelas widget stateful StatefulWidget.

11.2.2. Form

```
import 'package:flutter/material.dart';

void main() => runApp(const MyApp());

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    const appTitle = 'Form Styling Demo';
    return MaterialApp(
      title: appTitle,
      home: Scaffold(
        appBar: AppBar(
          title: const Text(appTitle),
        ),
        body: const MyCustomForm(),
      ),
    );
  }
}

class MyCustomForm extends StatelessWidget {
  const MyCustomForm({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.start,
      children: <Widget>[
        const Padding(
          padding: EdgeInsets.symmetric(horizontal: 8, vertical: 16),
          child: TextField(
            decoration: InputDecoration(
              border: OutlineInputBorder(),
              hintText: 'Enter a search term',
            ),
          ),
        ),
      ],
    );
  }
}
```

```
        ),
        ),
        ),
        Padding(
            padding: const EdgeInsets.symmetric(horizontal: 8,
vertical: 16),
            child: TextFormField(
                decoration: const InputDecoration(
                    border: UnderlineInputBorder(),
                    labelText: 'Enter your username',
                ),
                ),
            ],
        );
    }
}
```

11.2.3. Menu

Salah satu hal penting dari pembuatan aplikasi adalah menu. Menu ini berfungsi untuk separasi antar fitur atau page. Sulit rasanya apabila semua fitur ditampilkan dalam satu halaman, selain sulit pengguna akan kesulitan dalam mengoperasikannya. Maka disini menu page sangat bermanfaat.

Secara umum terdapat 2 jenis widget menu yang sering digunakan, yaitu 'bottom navigation bar' dan 'tab bar'. Karena Flutter mendukung penuh guideline yang dibuat oleh Google, yaitu Material Design.

11.2.3.1. Tab Bar

Resep untuk membuat tab bar adalah mengikuti 3 step dibawah :

1. Membuat 'TabController'.
2. Membuat tabs.
3. Membuat konten untuk setiap tab.

Membuat TabController

Agar tab berfungsi, Anda harus tetap menyinkronkan tab yang dipilih dan bagian konten. Menggunakan DefaultTabController adalah opsi paling sederhana, karena ia membuat TabController dan membuatnya tersedia untuk semua widget turunan.

```
DefaultTabController(
    // The number of tabs / content sections to display.
    length: 3,
    child: // Complete this code in the next step.
);
```

Membuat Tabs

Saat tab dipilih, maka harus menampilkan sebuah konten. Anda dapat membuat tabs menggunakan TabBar widget. Contoh dibawah adalah membuat TabBar dengan tiga widget Tab yang disimpan dalam widget AppBar.

```
DefaultTabController(  
  length: 3,  
  child: Scaffold(  
    appBar: AppBar(  
      bottom: TabBar(  
        tabs: [  
          Tab(icon: Icon(Icons.directions_car)),  
          Tab(icon: Icon(Icons.directions_transit)),  
          Tab(icon: Icon(Icons.directions_bike)),  
        ],  
      ),  
    ),  
  ),  
);
```

Membuat konten untuk masing-masing tab

Setelah anda memiliki tabs, tampilkan konten saat tab dipilih. Untuk tujuan ini, gunakan widget TabBarView.

```
TabBarView(  
  children: [  
    Icon(Icons.directions_car),  
    Icon(Icons.directions_transit),  
    Icon(Icons.directions_bike),  
  ],  
);
```

i Perhatikan urutan

Urutan sangat penting dan gunakan urutan sesuai dengan urutan pada TabBar

Contoh implementasi:

```
import 'package:flutter/material.dart';  
  
void main() {  
  runApp(const TabBarDemo());  
}  
  
class TabBarDemo extends StatelessWidget {  
  const TabBarDemo({Key? key}) : super(key: key);  
  
  @override  
  Widget build(BuildContext context) {  
    return MaterialApp(  
      home: DefaultTabController(  
       
```

```
length: 3,
child: Scaffold(
  appBar: AppBar(
    bottom: const TabBar(
      tabs: [
        Tab(icon: Icon(Icons.directions_car)),
        Tab(icon: Icon(Icons.directions_transit)),
        Tab(icon: Icon(Icons.directions_bike)),
      ],
    ),
    title: const Text('Tabs Demo'),
  ),
  body: const TabBarView(
    children: [
      Icon(Icons.directions_car),
      Icon(Icons.directions_transit),
      Icon(Icons.directions_bike),
    ],
  ),
),
),
),
);
}
}
```

11.2.3.2. Bottom Navigation Bar

Mirip dengan membuat TabBar, dibawah ini contoh untuk implementasi Bottom Navigation Bar

```
/// Flutter code sample for BottomNavigationBar

// This example shows a [BottomNavigationBar] as it is used within a [Scaffold]
// widget. The [BottomNavigationBar] has three [BottomNavigationBarItem]
// widgets, which means it defaults to [BottomNavigationBarType.fixed], and
// the [currentIndex] is set to index 0. The selected item is
// amber. The `'_onItemTapped` function changes the selected item's index
// and displays a corresponding message in the center of the [Scaffold].
```

```
import 'package:flutter/material.dart';

void main() => runApp(const MyApp());

/// This is the main application widget.
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  static const String _title = 'Flutter Code Sample';

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
```

```

        title: _title,
        home: My StatefulWidget(),
    );
}
}

/// This is the stateful widget that the main application instantiates.
class My StatefulWidget extends StatefulWidget {
    const My StatefulWidget({Key? key}) : super(key: key);

    @override
    State<My StatefulWidget> createState() => _My StatefulWidget State();
}

/// This is the private State class that goes with My StatefulWidget.
class _My StatefulWidget State extends State<My StatefulWidget> {
    int _selectedIndex = 0;
    static const TextStyle optionStyle =
        TextStyle(fontSize: 30, fontWeight: FontWeight.bold);
    static const List<Widget> _widgetOptions = <Widget>[
        Text(
            'Index 0: Home',
            style: optionStyle,
        ),
        Text(
            'Index 1: Business',
            style: optionStyle,
        ),
        Text(
            'Index 2: School',
            style: optionStyle,
        ),
    ];
}

void _onItemTapped(int index) {
    setState(() {
        _selectedIndex = index;
    });
}

@Override
Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
            title: const Text('BottomNavigationBar Sample'),
        ),
        body: Center(
            child: _widgetOptions.elementAt(_selectedIndex),
        ),
        bottomNavigationBar: BottomNavigationBar(

```

```
items: const <BottomNavigationBarItem>[
    BottomNavigationBarItem(
        icon: Icon(Icons.home),
        label: 'Home',
    ),
    BottomNavigationBarItem(
        icon: Icon(Icons.business),
        label: 'Business',
    ),
    BottomNavigationBarItem(
        icon: Icon(Icons.school),
        label: 'School',
    ),
],
currentIndex: _selectedIndex,
selectedItemColor: Colors.amber[800],
onTap: _onItemTapped,
),
);
}
}
```

11.2.4. Buttons

11.2.4.1. ElevatedButton

ElevatedButton adalah tombol yang biasa kita gunakan saat kita mendaftar,submit,login dst. berikut merupakan sourcecode dari ElevatedButton :

```
ElevatedButton(
    onPressed: () {
        print('ini done');
    },
    child: new Text('submit'),
),
```

11.2.4.2. TextButton

```
TextButton (
    child : text('menu'),
    onpressed : () {
        print ('sukses');
    }
)
```

11.2.4.3. DropdownButton

untuk membuat DropdownButton kita harus memiliki value di dalamnya agar dapat bekerja contoh sourcecode sebagai berikut :

```
DropdownButton(  
    value: selectedValue,  
    onChanged: (String? newValue){  
        setState(() {  
            selectedValue = newValue!;  
        });  
    },  
    items: dropdownItems  
)
```

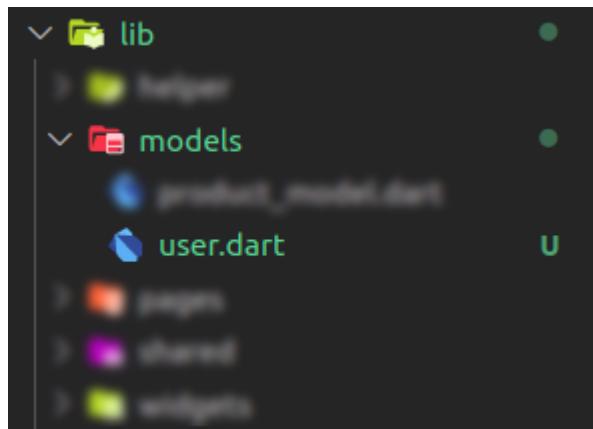
MODUL 12. MODEL, NAVIGATION & NOTIFICATION

Tujuan Praktikum

1. Mahasiswa mampu memahami konsep activity life cycle pada Android
2. Mahasiswa dapat mengimplementasikan activity dan intent pada Android

12.1. Model

12.1.1 Pengenalan Model



Gambar 12.1 Model pada flutter

Pada umumnya, hampir seluruh aplikasi yang dibuat akan bekerja dengan data. Data dalam sebuah aplikasi memiliki sangat banyak bentuk, tergantung dari aplikasi yang dibuat. Setiap data yang diterima atau dikirimkan akan lebih baik apabila memiliki standar yang sama. Hampir mustahil untuk melakukan pemeliharaan *project* yang kompleks tanpa model.

Model sendiri adalah bagian yang bersentuhan langsung dengan *database* dan mengkonversinya menjadi *class* dart yang dapat diakses lebih mudah. Umumnya model akan dibuat dari *response JSON*.

12.1.2 Membuat Model Class

Untuk membuat model, buatlah direktori baru pada folder lib project flutter, kemudian buat sebuah file *class* dart dengan nama filenya adalah nama data yang ingin dijadikan model.

Sebagai contoh, ketika membuat model user dengan response json seperti di bawah ini:

```
{  
  "user_id": 1,  
  "id": 13,  
  "title": "Bedroom Pop"  
}
```

Maka buatlah file user.dart di dalam folder models, dengan *code* seperti di bawah ini:

```
class Album {  
    final int userId;  
    final int id;  
    final String title;  
  
    const Album({  
        required this.userId,  
        required this.id,  
        required this.title,  
    });  
  
    factory Album.fromJson(Map<String, dynamic> json) {  
        return Album(  
            userId: json['user_id'],  
            id: json['id'],  
            title: json['title'],  
        );  
    }  
}
```

12.2. Navigation

12.2.1 Navigation Pindah Halaman

Untuk melakukan navigasi ke halaman lain pada Flutter, dapat gunakan *code* seperti di bawah ini:

```
Navigator.push(  
    context,  
    MaterialPageRoute(builder: (context) => SecondRoute()),  
>);
```

Untuk melakukan navigasi kembali ke halaman sebelumnya, dapat gunakan *code* seperti di bawah ini:

```
Navigator.pop(context);
```

Potongan code di atas harus diletakkan dalam *function* sebuah *widget*, misalnya pada onPressed milik ElevatedButton. SecondRoute pada contoh dapat diubah menjadi halaman baru yang dituju.

12.2.2 Navigation Mengirim Data

Untuk dapat melakukan navigasi dengan mengirimkan data ke halaman lain, perlu disiapkan 2 hal.

1. Halaman baru memiliki parameter data yang diminta.
2. Halaman awal mengirimkan data melalui parameter.

Untuk dapat melakukan hal tersebut, kita dapat membuat sebuah halaman baru seperti di bawah ini:

```
class DetailScreen extends StatelessWidget {  
    const DetailScreen({Key? key, required this.title}) : super(key: key);
```

```
final String title;

@Override
Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
            title: Text(title),
        ),
    );
}
```

Pada *code* di atas, telah dibuat halaman baru yang memiliki parameter data yang diinginkan yaitu atribut *title* dengan tipe data string.

```
Navigator.push(
    context,
    MaterialPageRoute(builder: (context) => DetailScreen(title: "Detail User")),
);
```

Berbeda dengan contoh navigasi sebelumnya, pada navigasi di atas ditambahkan parameter *title* berisi *string* “Detail User” yang akan dikirimkan ke halaman baru.

12.3. Notification

Untuk mengirimkan notifikasi dalam aplikasi flutter, dapat digunakan package bernama *flutter_local_notifications*.

Tambahkan terlebih dahulu package tersebut ke dalam aplikasi flutter dengan menuliskan *code* di bawah pada *pubspec.yaml*.

```
dependencies:
  flutter:
    sdk: flutter
  flutter_local_notifications: ^8.0.0
```

Setelah menambahkan package, ubah file *AndroidManifest* dengan menambahkan barisan kode seperti di bawah ini:

```
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.VIBRATE" />
```

Buatlah sebuah stateful widget yang akan digunakan sebagai halaman aplikasi.

Tambahkan potongan *code* di bawah ini di luar *build* dari stateful widget tersebut:

```
FlutterLocalNotificationsPlugin flutterLocalNotificationsPlugin = FlutterLocalNotificationsPlugin();
```

Dengan membuat sebuah object `FlutterLocalNotificationsPlugin`, operasi yang terdapat dalam package `flutter_notification` dapat digunakan.

Override method `initState` dari widget tersebut dengan menambahkan code seperti di bawah ini:

```
void initState() {
    super.initState();
    var initializationSettingsAndroid =
        AndroidInitializationSettings('flutter_devs');
    var initializationSettingsIOS = IOSInitializationSettings();
    var initSettings = InitializationSettings(
        initializationSettingsAndroid, initializationSettingsIOS);
    flutterLocalNotificationsPlugin.initialize(initSettings,
        onSelectNotification: onSelectNotification);
}
```

Kemudian buat sebuah function yang akan mengendalikan ketika notifikasi dipilih:

```
Future onSelectNotification(String payload) {
    Navigator.of(context).push(MaterialPageRoute(builder: (_) {
        return NewScreen(
            payload: payload,
        );
    }));
}
```

Buatlah sebuah widget baru yang akan menjadi halaman berikutnya setelah notifikasi dipilih:

```
class NewScreen extends StatelessWidget {
    String payload;
    NewScreen({
        @required this.payload,
    });
    @override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: AppBar(
                title: Text(payload),
            );
    }
}
```

Kemudian buat sebuah function yang fungsinya untuk menampilkan notifikasi sederhana untuk android:

```

showNotification() async {
    var android = new AndroidNotificationDetails(
        'id', 'channel ', 'description',
        priority: Priority.High, importance: Importance.Max);
    var iOS = new IOSNotificationDetails();
    var platform = new NotificationDetails(android, iOS);
    await flutterLocalNotificationsPlugin.show(
        0, 'Flutter devs', 'Flutter Local Notification Demo', platform,
        payload: 'Welcome to the Local Notification demo ');
}

```

AndroidNotificationDetails akan berisi mengenai detail notifikasi pada android.

IOSNotificationDetails akan berisi mengenai detail notifikasi pada iOS.

Kunci untuk menampilkan notifikasi terletak pada pemanggilan function flutterLocalNotificationsPlugin yang berfungsi untuk menampilkan notifikasi sesuai dengan platform yang digunakan.

Keseluruhan code di atas akan membentuk sebuah file seperti di bawah ini:

main.dart

```

import 'dart:async';
import 'package:flutter/material.dart';
import 'package:flutter_local_notifications/flutter_local_notifications.dart';

void main() => runApp(new MaterialApp(
    theme: ThemeData(
        appBarTheme: AppBarTheme(
            color: Colors.amber,
        )),
    home: new MyApp(),
    debugShowCheckedModeBanner: false,
));

class MyApp extends StatefulWidget {
    @override
    _MyAppState createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
    FlutterLocalNotificationsPlugin flutterLocalNotificationsPlugin =
        FlutterLocalNotificationsPlugin();

    @override
    void initState() {
        super.initState();
        var initializationSettingsAndroid =
            AndroidInitializationSettings('flutter_devs');

```

```

var initializationSettingsIOS = IOSInitializationSettings();
var initSetttings = InitializationSettings(
    initializationSettingsAndroid, initializationSettingsIOS);

flutterLocalNotificationsPlugin.initialize(initSetttings,
    onSelectNotification: onSelectNotification);
}

Future onSelectNotification(String payload) {
    Navigator.of(context).push(MaterialPageRoute(builder: (_) {
        return NewScreen(
            payload: payload,
        );
    }));
}

@Override
Widget build(BuildContext context) {
    return Scaffold(
        appBar: new AppBar(
            backgroundColor: Colors.amber,
            title: new Text('Flutter notification demo'),
        ),
        body: new Center(
            child: Column(
                children: <Widget>[
                    ButtonTheme(
                        minWidth: 250.0,
                        child: RaisedButton(
                            color: Colors.blueAccent,
                            onPressed: showNotification,
                            child: new Text(
                                'showNotification',
                            ),
                        ),
                    ),
                    ],
                ],
            ),
        );
    );
}

showNotification() async {
    var android = new AndroidNotificationDetails(
        'id', 'channel ', 'description',
        priority: Priority.High, importance: Importance.Max);
    var iOS = new IOSNotificationDetails();
    var platform = new NotificationDetails(android, iOS);
    await flutterLocalNotificationsPlugin.show(
        0, 'Flutter devs', 'Flutter Local Notification Demo', platform,

```

```
        payload: 'Welcome to the Local Notification demo ');
    }

class NewScreen extends StatelessWidget {
    String payload;

    NewScreen({
        @required this.payload,
    });

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: AppBar(
                title: Text(payload),
            ),
        );
    }
}
```

Di atas adalah keseluruhan kode yang digunakan untuk menampilkan sebuah notifikasi sederhana.

MODUL 13. STATE MANAGEMENT & NETWORKING

Tujuan Praktikum

1. Mahasiswa mampu memahami konsep web service pada Android
2. Mahasiswa mampu mengimplementasikan protocol http dan pemrosesan JSON pada Android

13.1. State Management

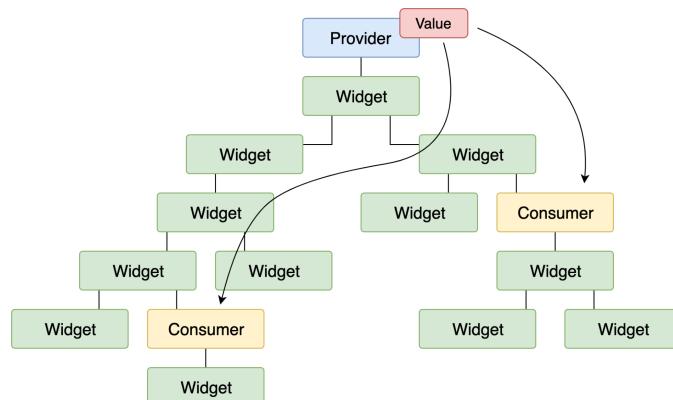
13.1.1. Pengenalan State Management

Ketika aplikasi semakin kompleks dibuat, maka pasti akan ada saatnya dimana harus dibagikan *state* aplikasi ke berbagai halaman yang ada.

Flutter adalah deklaratif, sehingga Flutter membangun *user interface* berdasarkan state saat ini.

Dengan menggunakan *state management*, dapat dilakukan sentralisasi semua *state* dari berbagai macam *UI Control* untuk mengendalikan aliran data lintas aplikasi.

13.1.2. State Management Provider



Gambar 13.1 State Management Provider

Provider adalah salah satu state management yang tersedia di Flutter. Provider memiliki *widget* Consumer, yaitu sebuah *widget* yang *listen* terhadap perubahan *value* dari provider tersebut, yang nantinya akan melakukan *rebuild widget* di bawahnya ketika perubahan tersebut terjadi.

Di bawah ini adalah contoh penggunaan provider pada aplikasi pembelanjaan sederhana.

Untuk dapat menggunakan Provider, tambahkan package Provider pada pubspec.yaml seperti di bawah ini:

```
dependencies:  
  flutter:  
    sdk: flutter  
  
  provider: ^6.0.0
```

Provider menggunakan 3 konsep, yaitu:

- ChangeNotifier

ChangeNotifier adalah sebuah *class* sederhana yang tersedia pada FlutterSDK yang memiliki fitur untuk mengganti notifikasi kepada *listener*nya.

Dalam provider, ChangeNotifier adalah cara untuk melakukan enkapsulasi *state* aplikasi.

Buat sebuah class baru yang melakukan *extend* terhadap ChangeNotifier untuk mengatur *state* dari *cart*.

```
class CartModel extends ChangeNotifier {  
    // state dari items cart yang bersifat private.  
    final List<Item> _items = [];  
  
    // Getter untuk items cart yang tidak dapat dimodifikasi.  
    UnmodifiableListView<Item> get items => UnmodifiableListView(_items);  
  
    // Getter untuk total harga semua item, asumsi setiap item harganya 42.  
    int get totalPrice => _items.length * 42;  
  
    // Menambahkan [item] ke cart. Proses penambahan dan RemoveAll adalah jalan  
    // satu-satunya cara untuk memodifikasi cart dari luar.  
    void add(Item item) {  
        _items.add(item);  
        // notifyListeners() menginformasikan widget yang listening untuk melakukan rebuild.  
        notifyListeners();  
    }  
  
    // Menghapus semua items pada cart.  
    void removeAll() {  
        _items.clear();  
        // notifyListeners() menginformasikan widget yang listening untuk melakukan rebuild.  
        notifyListeners();  
    }  
}
```

- **ChangeNotifierProvider**

ChangeNotifierProvider adalah widget yang menyediakan instance ChangeNotifier ke seluruh *child*nya. Itu berasal dari package Provider. Pada konteks aplikasi pembelanjaan sederhana ini, ChangeNotifierProvider diletakkan di bagian teratas, yaitu MyApp.

ChangeNotifierProvider tidak perlu diletakkan lebih tinggi dari yang diperlukan.

```
void main() {  
    runApp(  
        ChangeNotifierProvider(  
            create: (context) => CartModel(),  
            child: const MyApp(),  
        ),  
    );  
}
```

Apabila dibutuhkan lebih dari satu class ChangeNotifier, gunakan MultiProvider:

```

void main() {
  runApp(
    MultiProvider(
      providers: [
        ChangeNotifierProvider(create: (context) => CartModel()),
        Provider(create: (context) => SomeOtherClass()),
      ],
      child: const MyApp(),
    ),
  );
}

```

- Consumer

Tambahkan consumer di atas widget yang membutuhkan informasi dari state provider yang ada.

Letakkan consumer di bagian terdalam yang mungkin, untuk menghindari *rebuild* yang tidak diperlukan.

```

return Consumer<CartModel>(
  builder: (context, cart, child) => Stack(
    children: [
      // Letakkan widget yang memang membutuhkan state cart di sini.
      if (child != null) child,
      Text("Total price: ${cart.totalPrice}"),
    ],
  ),
  child: const SomeExpensiveWidget(),
);

```

Parameter *child* pada *Consumer* adalah bagian dari optimasi, di mana hanya akan *dibuild* sekali dan tidak akan melakukan *rebuild* ketika ada perubahan *state*.

- Provider.of

Ada kondisi di mana tidak diperlukan data dari provider untuk selalu melakukan *rebuild*, namun tetap butuh untuk diakses. Pada konteks aplikasi ini, misalkan diperlukan untuk memanggil method *RemoveAll()* yang mana tidak perlu melakukan *rebuild* ketika ada perubahan UI.

```
Provider.of<CartModel>(context, listen: false).removeAll();
```

13.2. Networking

Untuk dapat melakukan aktivitas yang membutuhkan *networking*, seperti HTTP Request, kita harus menambahkan package HTTP pada pubspec.yaml.

```
dependencies:  
http: <latest_version>
```

Setelah itu, import package HTTP pada file yang melakukan aktivitas *networking*.

```
import 'package:http/http.dart' as http;
```

Kemudian tambahkan *internet permission* pada *AndroidManifest.xml* apabila aplikasi dijalankan tidak dalam mode debug.

13.2.1. Fetch & Delete Data

Untuk melakukan fetch data, tambahkan kode seperti di bawah ini:

```
Future<Album> fetchAlbum() async {  
    final response = await http  
        .get(Uri.parse('https://jsonplaceholder.typicode.com/albums/1'));  
  
    if (response.statusCode == 200) {  
        return Album.fromJson(jsonDecode(response.body));  
    } else {  
        throw Exception('Failed to load album');  
    }  
}
```

Dengan memanfaatkan package HTTP, kita dapat melakukan request bertipe GET ke RestAPI.

Kemudian pengkondisian dilakukan untuk mengecek apakah response dari request GET memiliki *status code* 200 atau tidak.

Perhatikan bahwa terdapat potongan kode “Album.fromJson(jsonDecode(response.body))”. Yang terjadi pada potongan kode tersebut adalah akan dihasilkan sebuah objek dengan class Album yang parameternya adalah “jsonDecode(response.body)”.

Untuk melakukan delete data, tambahkan kode seperti di bawah ini:

```
Future<http.Response> deleteAlbum(String id) async {  
    final http.Response response = await http.delete(  
        Uri.parse('https://jsonplaceholder.typicode.com/albums/$id'),  
        headers: <String, String>{  
            'Content-Type': 'application/json; charset=UTF-8',  
        },  
    );  
  
    return response;  
}
```

Perhatikan bahwa terdapat parameter yaitu ID dari Album yang akan dihapus yang dikirimkan melalui parameter dari URL API.

13.2.2. Send & Update

Untuk melakukan send data, tambahkan kode seperti di bawah ini:

```
Future<http.Response> createAlbum(String title) {  
    return http.post(  
        Uri.parse('https://jsonplaceholder.typicode.com/albums'),  
        headers: <String, String>{  
            'Content-Type': 'application/json; charset=UTF-8',  
        },  
        body: jsonEncode(<String, String>{  
            'title': title,  
        }),  
    );  
}
```

Ketika melakukan send data, gunakan request bertipe POST. Kemudian parameter body diisi dengan data yang ingin dikirimkan.

13.2.3. Parse JSON

Setiap aktivitas networking yang dilakukan, umumnya akan menghasilkan atau mengirimkan JSON. Untuk dapat mengonversi JSON menjadi map, atau map menjadi JSON, kita dapat melakukan import convert, seperti di bawah ini:

```
import 'dart:convert';
```

Untuk melakukan konversi JSON menjadi Map, kode yang digunakan adalah:

```
dynamic jsonDecode(String source,  
    {Object? reviver(Object? key, Object? value)?}) =>  
    json.decode(source, reviver: reviver);
```

Untuk melakukan konversi Map menjadi JSON, kode yang digunakan adalah:

```
String jsonEncode(Object? object,  
    {Object? toEncodable(Object? nonEncodable)?}) =>  
    json.encode(object, toEncodable: toEncodable);
```

MODUL 14. MAPS, PLACES, CAMERA & MEDIA

Tujuan Praktikum

1. Mahasiswa mampu memahami konsep API pada Android
2. Mahasiswa dapat mengimplementasikan penggunaan kamera dan media pada Android

14.1. Google Maps API

Google Maps API merupakan salah satu layanan dari Google untuk membantu developer menciptakan aplikasi yang menggunakan fitur peta atau maps. Pada Google Maps API kita dapat memasang *marker*, menggunakan fitur *route*, mencari tempat, dan masih banyak lagi.

Cara implementasi Google API pada flutter dapat dilakukan dengan menggunakan packages Google Maps. Tahapan dalam menambahkan *Google Maps API* dapat mengikuti langkah-langkah berikut :

1. Dapatkan API key melalui link berikut <https://cloud.google.com/maps-platform/>
2. Selanjutnya, enable Google Map SDK di tiap platform yang akan menggunakan Google Maps.
 - a. Pergi ke <https://console.cloud.google.com/> (Google Developers Console)
 - b. Pilih project yang ingin menggunakan Google Maps
 - c. Pilih pada navigation menu, lalu pilih "Google Maps"
 - d. Pilih "APIs" di bawah menu Google Maps
 - e. Untuk mengaktifkan Google Maps di Android, pilih "Maps SDK for Android" pada section "Additional APIs", lalu pilih "ENABLE"
 - f. Untuk mengaktifkan Google Maps di iOS, pilih "Maps SDK for iOS" pada section "Additional APIs", lalu pilih "ENABLE"
 - g. Pastikan bahwa APIs telah aktif pada section "Enabled APIs"
 - h. Untuk lebih detail bisa cek di <https://developers.google.com/maps/gmp-get-started>
3. Android
 - a. Set minSdkVersion di android/app/build.gradle:

```
android {  
    defaultConfig {  
        minSdkVersion 20  
    }  
}
```

Ini dimaksudkan agar aplikasinya support atau bersedia pada Android SDK 20 atau lebih tinggi.

- a. Tambahkan API key pada manifest aplikasi

```
android/app/src/main/AndroidManifest.xml
```

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
    package="com.pinkesh.google_maps_flutter">  
  
    <uses-permission  
        android:name="android.permission.ACCESS_FINE_LOCATION"/>  
    <uses-permission  
        android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

```

<application
    android:label="google_maps_flutter"
    android:icon="@mipmap/ic_launcher">

    <!-- TODO: Add your API key here -->
    <meta-data android:name="com.google.android.geo.API_KEY"
        android:value="YOUR KEY HERE"/>

    <activity>...</activity>
</application>
</manifest>

```

4. Hybrid Composition

Untuk menggunakan [Hybrid Composition](#) yang digunakan untuk merender GoogleMap pada widget Android, terapkan `AndroidGoogleMapsFlutter.useAndroidViewSurface` ke true.

```

if (defaultTargetPlatform == TargetPlatform.android) {
  AndroidGoogleMapsFlutter.useAndroidViewSurface = true;
}

```

5. iOS

Plugin ini membutuhkan iOS 9.0 atau lebih tinggi. Untuk menerapkan, tambahkan API key pada application delegate `ios/Runner/AppDelegate.m`:

```

#include "AppDelegate.h"
#include "GeneratedPluginRegistrant.h"
#import "GoogleMaps/GoogleMaps.h"

@implementation AppDelegate

- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
[GMServices provideAPIKey:@"YOUR KEY HERE"];
[GeneratedPluginRegistrant registerWithRegistry:self];
return [super application:application didFinishLaunchingWithOptions:launchOptions];
}
@end

```

Atau dalam penulisan bahasa Swift, tambahkan API key pada application delegate `ios/Runner/AppDelegate.swift`:

```

import UIKit
import Flutter
import GoogleMaps

@UIApplicationMain
@objc class AppDelegate: FlutterAppDelegate {

```

```

override func application(
    _ application: UIApplication,
    didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?
) -> Bool {
    GMSServices.provideAPIKey("YOUR KEY HERE")
    GeneratedPluginRegistrant.register(with: self)
    return super.application(application, didFinishLaunchingWithOptions: launchOptions)
}
}

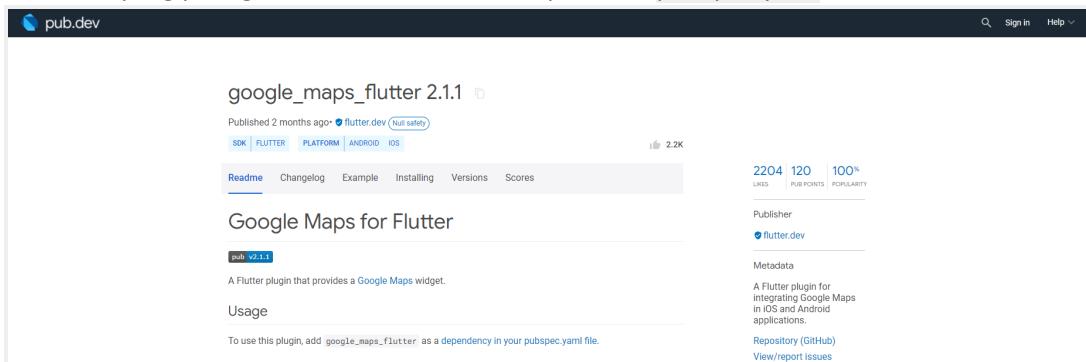
```

Langkah diatas untuk menambahkan Google Maps API ke aplikasi.

14.1.1. Menambahkan Packages Google Maps

Setelah mengikuti langkah diatas, sekarang adalah langkah-langkah menambahkan Google Maps ke layar aplikasi Flutter:

1. Pergi ke <https://www.pub.dev>, lalu cari packages Google Maps. Nama packagesnya adalah `google_maps_flutter`.
2. Cari versi yang paling terbaru lalu tambahkan pada file `pubspec.yaml`



Gambar 14.1 google_maps_flutter

```

dependencies:
  flutter:
    sdk: flutter
  cupertino_icons: 1.0.0
  google_maps_flutter: ^any version

```

3. Selanjutnya, import packages ke dalam file Dart

```
import 'package:google_maps_flutter/google_maps_flutter';
```

4. Lalu, tambahkan widget GoogleMap ke file Dart

```
GoogleMap(
    initialCameraPosition: _kInitialPosition,
),
```

GoogleMap diberi _kInitialPosition, yang dimana untuk menyimpan lokasi default saat aplikasi dijalankan atau dimuat.

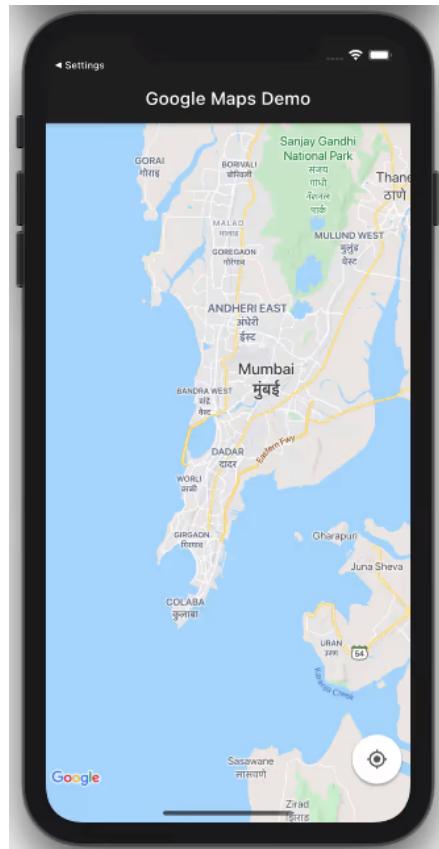
5. Buat fungsi _kMapCenter dan _kInitialPosition dengan kode sebagai berikut:

```
static final LatLng _kMapCenter =  
    LatLng(19.018255973653343, 72.84793849278007);  
  
static final CameraPosition _kInitialPosition =  
    CameraPosition(target: _kMapCenter, zoom: 11.0, tilt: 0, bearing: 0);
```

6. Berikut adalah tampilan kode yang lengkap

```
import 'package:flutter/material.dart';  
import 'package:google_maps_flutter/google_maps_flutter.dart';  
  
class SimpleMap extends StatefulWidget {  
    @override  
    _SimpleMapState createState() => _SimpleMapState();  
}  
  
class _SimpleMapState extends State<SimpleMap> {  
    static final LatLng _kMapCenter =  
        LatLng(19.018255973653343, 72.84793849278007);  
  
    static final CameraPosition _kInitialPosition =  
        CameraPosition(target: _kMapCenter, zoom: 11.0, tilt: 0, bearing: 0);  
  
    @override  
    Widget build(BuildContext context) {  
        return Scaffold(  
            appBar: AppBar(  
                title: Text('Google Maps Demo'),  
            ),  
            body: GoogleMap(  
                initialCameraPosition: _kInitialPosition,  
            ),  
        );  
    }  
}
```

Berikut tampilannya



Gambar 14.2 Demo Gmaps pada Flutter

14.1.2. Menambahkan Akses Lokasi Kita Pada Manifest

Secara default, map akan menampilkan lokasi yang sudah kita definisikan pada initialCameraPosition yang ada pada parameter widget. Jika pengguna ingin menampilkan lokasi mereka, ubah pengaturan myLocationEnable menjad true.

Berikut barisan kode untuk menampilkan lokasi kita saat ini:

```
GoogleMap(  
  initialCameraPosition: _kInitialPosition,  
  onMapCreated: onMapCreated,  
  myLocationEnabled: true,  
)
```

14.1.3. Menambahkan Marker pada Google Maps

Marker merupakan cara untuk menunjukkan lokasi tertentu.

Untuk membuat marker pada map, berikut adalah barisan kodenya:

```
Set<Marker> _createMarker() {  
  return {  
    Marker(
```

```
markerId: MarkerId("marker_1"),
position: _kMapCenter,
infoWindow: InfoWindow(title: 'Marker 1'),
rotation: 90),
Marker(
markerId: MarkerId("marker_2"),
position: LatLng(-6.9733165,107.6281415,17),
),
);
}
```

14.2. Place Picker

Place picker merupakan plugin untuk memberi informasi terkait lokasi yang sedang ditunjuk oleh map. Untuk menggunakan place picker, terlebih dahulu kita tambahkan API Google Maps dan ubah beberapa pengaturan dengan mengikuti langkah-langkah sebelum sub bab ini.

Jika sudah mengubah pengaturan yang diminta, terlebih dahulu kita tambahkan package ke dalam pubspec.yaml seperti ini:

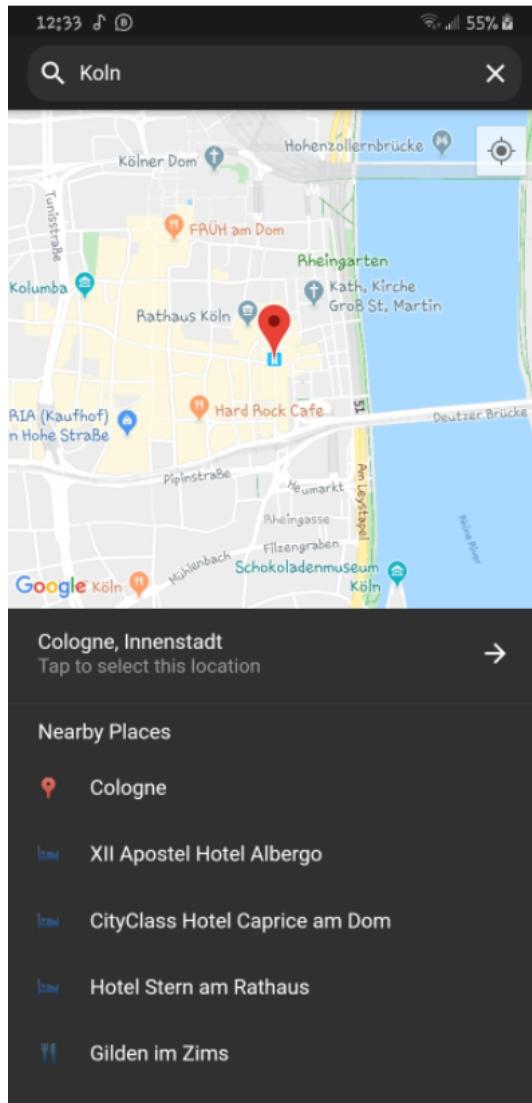
```
import 'package:place_picker/place_picker.dart';
```

Setelah itu, kita buat method seperti yang akan di bahas pada sub bab ini, lalu gunakan onTap di Button atau InkWell untuk memanggil method tersebut. Berikut barisan kodennya:

```
void showPlacePicker() async {
LocationResult result = await Navigator.of(context).push(MaterialPageRoute(
builder: (context) =>
PlacePicker("YOUR API KEY",
displayLocation: customLocation,
)));
}

// Handle the result in your way
print(result);
}
```

Kira-kira seperti inilah tampilannya:



Gambar 14.3 Demo Pencarian pada Gmaps

14.3. Camera API

Pada bahasan kali ini, kita akan menggunakan packages atau plugin Camera supaya aplikasi yang dibuat dapat mengakses kamera yang ada pada device.

1. iOS

Plugin ini dapat berjalan di versi iOS 10.0 atau lebih tinggi. Jika dijalankan di versi di bawah 10.0, pastikan bahwa ada program untuk cek versi dari iOS sebelum menggunakan fitur yang ada pada kamera. Plugin untuk cek versi iOS yaitu device_info_plus.

Tambahkan 2 baris pada ios/Runner/Info.plist:

- Satu baris dengan Privacy - Camera Usage Description
- Dan satu baris dengan Privacy - Microphone Usage Description

Atau dengan format teks penambahan key:

```
<key>NSCameraUsageDescription</key>
<string>Can I use the camera please?</string>
```

```
<key>NSMicrophoneUsageDescription</key>
<string>Can I use the mic please?</string>
```

2. Android

Ubah minimum versi Android sdk ke 21 (atau lebih tinggi) pada file android/app/build.gradle.

```
minSdkVersion 21
```

Hal tersebut penting karena MediaRecorder tidak bekerja dengan baik di emulator sesuai dengan [dokumentasi](#).

Berikut contoh aplikasi flutter dengan menampilkan kamera:

```
import 'dart:async';
import 'package:flutter/material.dart';
import 'package:camera/camera.dart';

List<CameraDescription> cameras;

Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();

  cameras = await availableCameras();
  runApp(CameraApp());
}

class CameraApp extends StatefulWidget {
  @override
  _CameraAppState createState() => _CameraAppState();
}

class _CameraAppState extends State<CameraApp> {
  CameraController controller;

  @override
  void initState() {
    super.initState();
    controller = CameraController(cameras[0], ResolutionPreset.max);
    controller.initialize().then((_) {
      if (!mounted) {
        return;
      }
      setState(() {});
    });
  }
}
```

```

@Override
void dispose() {
    controller?.dispose();
    super.dispose();
}

@Override
Widget build(BuildContext context) {
    if (!controller.value.isInitialized) {
        return Container();
    }
    return MaterialApp(
        home: CameraPreview(controller),
    );
}
}

```

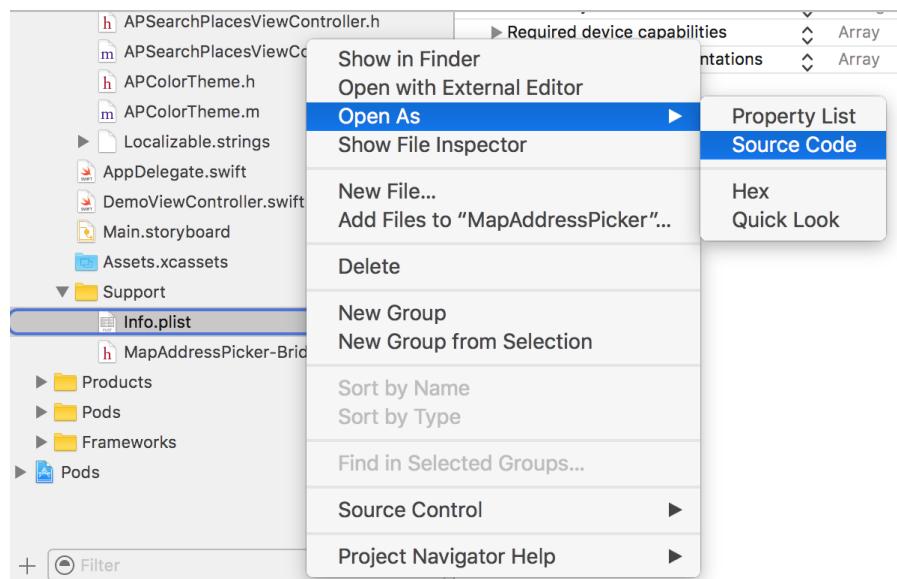
14.4. Media API

Pada bahasan kali ini, kita akan menggunakan packages atau plugin Image Picker supaya aplikasi dapat mengakses media galeri pada device. Pada platform iOS diperlukan konfigurasi tambahan, namun pada android tidak diperlukan konfigurasi tambahan. Berikut untuk konfigurasi tambahan:

1. iOS

Pada iOS diperlukan versi iOS 9.0.0 keatas, pastikan build version di set ke iOS 9.0.0 atau lebih tinggi. tambahkan konfigurasi pada file `Info.plist` di Runner iOS.

- Rubah format tampilan `.plist` dengan tipe source code. Dapat dilihat pada gambar dibawah.



Gambar 14.4 Membuka file `.plist`

- Tambahkan potongan kode berikut pada info.plist

```
<key>NSPhotoLibraryUsageDescription</key>
<string>Can I use the photo library?</string>
<key>NSCameraUsageDescription</key>
<string>Can I use the camera please?</string>
<key>NSMicrophoneUsageDescription</key>
<string>Can I use the mic please?</string>
```

- Konfigurasi tambahan pada iOS sudah selesai.

Untuk penggunaan image picker, dapat dilihat pada potongan kode berikut :

```
class ImageFromGalleryEx extends StatefulWidget {
  final type;
  ImageFromGalleryEx(this.type);

  @override
  ImageFromGalleryExState createState() => ImageFromGalleryExState(this.type);
}

class ImageFromGalleryExState extends State<ImageFromGalleryEx> {
  var _image;
  var imagePicker;
  var type;

  ImageFromGalleryExState(this.type);

  @override
  void initState() {
    super.initState();
    imagePicker = new ImagePicker();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(type == ImageSourceType.camera
          ? "Image from Camera"
          : "Image from Gallery")),
      body: Column(
        children: <Widget>[
          SizedBox(
            height: 52,
          ),
          Center(
```

```
child: GestureDetector(
  onTap: () async {
    var source = type == ImageSourceType.camera
      ? ImageSource.camera
      : ImageSource.gallery;
    XFile image = await imagePicker.pickImage(
      source: source, imageQuality: 50, preferredCameraDevice:
    CameraDevice.front);
    setState(() {
      _image = File(image.path);
    });
  },
  child: Container(
    width: 200,
    height: 200,
    decoration: BoxDecoration(
      color: Colors.red[200]),
    child: _image != null
      ? Image.file(
        _image,
        width: 200.0,
        height: 200.0,
        fit: BoxFit.fitHeight,
      )
      : Container(
        decoration: BoxDecoration(
          color: Colors.red[200]),
        width: 200,
        height: 200,
        child: Icon(
          Icons.camera_alt,
          color: Colors.grey[800],
        ),
      ),
    ),
  ),
),
),
),
),
);
}
}
```

DAFTAR PUSTAKA

- [1] R. W. Sebesta, Programming the World Wide Web, Addison-Wesley, 2014.
- [2] "W3Schools," December 2017. [Online]. Available: <https://www.w3schools.com/>.
- [3] "Bootstrap," January 2018. [Online]. Available: <https://getbootstrap.com/>.
- [4] "CodeIgniter," January 2018. [Online]. Available: <https://codeigniter.com/docs>.
- [5] A. Solichin, Pemrograman Web dengan PHP dan MySQL, Budi Luhur, 2016.
- [6] A. Subagia, Membangun Aplikasi dengan Codeigniter dan Database SQL Server, Jakarta: PT Elex Media Komputindo, 2017.
- [7] "Memulai Git - Dasar Git," Januari 2019. [Online]. Available: <https://gitscm.com/book/id/v1/Memulai-Git-Dasar-Git>.
- [8] "Mengenal Restful API," Januari 2019. [Online]. Available: <https://kudo.co.id/engineering/2016/09/15/mengenal-restful-api/>.
- [9] "Pengertian Node.js," Januari 2019. [Online]. Available: <https://teknojurnal.com/pengertianapa-itu-node-js/>.
- [10] "Cara Menginstall XAMPP di Windows," Januari 2019. [Online]. Available: <https://www.duniailkom.com/tutorial-belajar-wordpress-cara-menginstall-xampp-diwindows/>.
- [11] "Learn Dart Programming," Januari 2022. [Online]. Available: https://www.tutorialspoint.com/dart_programming
- [12] "Work with tabs," Januari 2022. [Online]. Available: <https://docs.flutter.dev/cookbook/design/tabs>
- [13] "Using slivers to achieve fancy scrolling," Januari 2022. [Online]. Available: <https://docs.flutter.dev/development/ui/advanced/slivers>
- [14] "Navigation," Januari 2022. [Online]. Available: <https://docs.flutter.dev/cookbook/navigation>
- [15] "Adding interactivity to your Flutter app." [Online]. Available: <https://docs.flutter.dev/development/ui/interactive>
- [16] "Layouts in Flutter," [Online]. Available: <https://docs.flutter.dev/development/ui/layout>
- [17] "Flutter local notifications," [Online]. Available: https://pub.dev/packages/flutter_local_notifications
- [18] "Adding Google Maps to a Flutter app," [Online]. Available: <https://blog.logrocket.com/adding-google-maps-to-a-flutter-app/>
- [19] "Building an image picker in Flutter," [Online]. Available: <https://blog.logrocket.com/building-an-image-picker-in-flutter/>



Kontak Kami :

-  @fiflab
-  Praktikum IF LAB
-  informaticslab@telkomuniversity.ac.id
-  informatics.labs.telkomuniversity.ac.id