

# **RAHASIA EFEKTIFITAS MENGOPTIMALKAN KOMUNIKASI DENGAN AUTOSENDER *WHATSAPP***



**DASLAN JOSUA VALENTINO SIAHAAN  
MARJUNIATI PUTRI  
RONI ANDARSYAH**

# **RAHASIA EFEKTIFITAS MENGOPTIMALKAN KOMUNIKASI DENGAN AUTOSENDER *WHATSAPP***

**DASLAN JOSUA VALENTINO SIAHAAN  
MARJUNIATI PUTRI  
RONI ANDARSYAH**



**PT. Penerbit Buku Pedia  
2023**

# **RAHASIA EFEKTIFITAS MENGOPTIMALKAN KOMUNIKASI DENGAN AUTOSENDER *WHATSAPP***

***Penulis :***

Daslan Josua Valentino Siahaan  
Marjuniati Putri  
Roni Andarsyah

***ISBN :***

***Editor :***

Rolly Maulana Awangga

***Penyunting :***

Rolly Maulana Awangga

***Desain sampul dan Tata letak :***

Daslan Josua Valentino Siahaan  
Marjuniati Putri  
Roni Andarsyah

***Font:***

Calibri

***Penerbit :***

Penerbit Buku Pedia

***Redaksi:***

Athena Residence Blok E No. 1, Desa Ciwaruga,  
Kec. Paronpong, Kab. Bandung Barat 40559  
Tel. 628-775-2000-300  
Email: [penerbit@bukupedia.co.id](mailto:penerbit@bukupedia.co.id)

***Distributor:***

Informatics Research Center  
Jl.Sariasih No.54  
Bandung 40151  
Email: [irc@ulbi.ac.id](mailto:irc@ulbi.ac.id)

Cetakan pertama, 2023

Hak cipta dilindungi undang-undang

Dilarang memperbanyak karya tulis ini dalam bentuk dan  
dengan cara apa pun tanpa izin tertulis dari penerbit

# PRAKATA

Pengembangan *whatsapp* adalah sebuah program dalam kecerdasan buatan yang dirancang untuk dapat berkomunikasi langsung dengan manusia. Dengan adanya intergrasi *API* dan *micro-Frontends* pada pengembangan *whatsapp* dapat meningkatkan kemampuan pemasaran interaktif, memberikan pengalaman pengguna yang memungkinkan pemasar untuk berinteraksi secara lebih dinamis dengan pelanggan mereka. Yang membedakan pengembangan *whatsapp* dengan menggunakan *API* & dan *micro frontends* dengan yang lain adalah memberikan fleksibilitas dan responsibilitas yang lebih besar dalam pengembalian aplikasi *whatsapp* yang lebih canggih dan adaptif. Sistem yang akan dibangun menggunakan *Micro frontends* yaitu sistem yang menggunakan pendekatan antar muka (*UI*) secara terpisah dan independen. Penelitian ini berfokus pada *UI* dan *micro frontend*. Penelitian ini dilakukan sebagai bagian dari pengabdian masyarakat untuk memberikan kontribusi dalam memahami teknologi terkini

Aplikasi dapat diakses melalui link:

<https://GitHub.com/WASENDERAUTO/MATERI-BUKPED.git>

# DAFTAR ISI

PRAKATA .....	i
<b>DAFTAR ISI .....</b>	<b>ii</b>
<b>BAB I MICRO frontend.....</b>	<b>1</b>
1. <i>Micro frontend</i> .....	1
A. Beberapa fitur penting dari <i>Micro frontend</i> : .....	2
B. Cara Kerja <i>Micro frontend</i> : .....	5
C. Fungsi <i>Micro frontend</i> : .....	6
D. Cara Membangun <i>Micro frontend</i> :.....	8
<b>BAB II PHP, CSS &amp; MySQL .....</b>	<b>11</b>
1. <i>PHP (Hypertext Preprocessor)</i> .....	11
A. Karakteristik dan penggunaan utama PHP:.....	11
B. Kelebihan PHP:.....	13
C. Kekurangan PHP:.....	14
D. Cara penggunaan <i>PHP</i> :.....	15
2. <i>CSS (Cascading Style Sheets)</i> .....	15
A. Pengembangan <i>CSS</i> .....	15
3. <i>MYSQL</i> .....	16
A. Apa perbedaan menggunakan <i>MySQL &amp; SQL</i> ? .....	17
B. Mengapa menggunakan <i>MySQL</i> ? .....	18
C. Contoh kode program dalam menggunakan <i>MySQL</i> : .....	19
.....	19
<b>BAB III.....</b>	<b>20</b>

<i>COMPOSER</i> .....	20
1. <i>Composer</i> .....	20
A. Konsep dasar <i>Composer</i> .....	20
B. Kelebihan & kekurangan <i>Composer</i> .....	21
C. Cara instalasi <i>Composer PHP</i> pada <i>windows</i> .....	23
BAB VI .....	25
<i>JAVASCRIPT</i> .....	25
A. <i>JavaScript</i> .....	25
B. Apa itu <i>JavaScript</i> ? .....	27
C. Mengapa Anda Harus Belajar <i>JavaScript</i> ? .....	28
1. <i>Open source</i> dan Gratis .....	28
2. Tidak Memerlukan Instalasi .....	29
3. Komunitas yang Luas .....	29
4. Ekosistem yang Besar .....	29
1. <i>Web &amp; Browser</i> .....	30
2. <i>Code Editor</i> .....	30
D. Hal-hal yang Perlu Dipahami Sebelum Menggunakan <i>JavaScript</i> .....	31
E. Aturan Penulisan Kode <i>JavaScript</i> .....	31
1. <i>Whitespace</i> .....	31
2. <i>Line Length</i> .....	31
3. <i>Comments</i> .....	32
4. <i>Case Sensitive</i> .....	32
5. <i>Statement Rules</i> .....	32
6. <i>Reserved Words</i> .....	32
F. Variabel <i>JavaScript</i> : .....	33

G. Tipe Data <i>JavaScript</i> .....	35
1. <i>Number</i> .....	35
2. <i>String</i> .....	35
4. <i>Function</i> .....	36
5. <i>Array</i> .....	36
6. <i>Object</i> .....	36
H. Operator <i>JavaScript</i> .....	37
1. Aritmatika .....	37
2. Penugasan .....	37
3. Perbandingan .....	37
4. Logika .....	37
5. <i>String</i> .....	37
6. <i>Typeof</i> .....	38
7. Kondisional .....	38
I. Cara Menulis Kode <i>JavaScript</i> .....	38
2. Terpisah dari File <i>HTML</i> ( <i>External JavaScript</i> ) .....	38
Cara Membuat Program Sederhana dengan <i>JavaScript</i> .....	39
BAB V .....	40
<i>JSON</i> FILE .....	40
A. Apa itu <i>JSON</i> ? .....	40
B. Mengapa menggunakan <i>JSON</i> ? .....	40
C. Sejarah <i>JSON</i> .....	41
D. Fitur <i>JSON</i> .....	42
E. Aturan untuk Sintaks <i>JSON</i> .....	42
F. Tipe Data di <i>JSON</i> .....	42

G. Rangkaian:.....	43
H.Himpunan .....	44
I.Objek <i>JSON</i> .....	44
J.Aplikasi <i>JSON</i> .....	44
K. <i>JSON</i> vs XML .....	45
L.Kekurangan <i>JSON</i> .....	45
A. Alat <i>JSON</i> Populer (Pengaya) .....	46
1. <i>Lint JSON</i> :.....	46
2. Editor <i>JSON Daring</i> :.....	46
Alat Pengurang <i>JSON</i> : .....	46
Konverter <i>JSON</i> ke XML: .....	46
Pemformat <i>JSON</i> :.....	47
B. Ringkasan:.....	47
BAB VI .....	48
<i>GCP &amp; GCF</i> .....	48
1. Apa itu <i>GCP</i> ? .....	48
B. Kekurangan & Kelebihan <i>GCP</i> .....	50
C.Hal apa saja yang dibutuhkan dalam <i>GCP</i> .....	52
2.Apa itu <i>GCF</i> ? .....	54
A. Alasan menggunakan google <i>cloud function</i> .....	54
B.Hubungan <i>Google cloud Platform</i> dan <i>googe cloud function</i> .....	56
BAB VII .....	57
POMOKIT / POMODORO .....	57
1. Apa itu Pomodoro ? .....	57
A. Cara menggunakan Pomodoro:.....	58



INFO PENTING: .....	59
BAB VIII.....	60
TUTORIAL MEMBUAT <i>WHATSAPP</i> AUTOSENDER.....	60
<i>A.Frontend (Client-Side):</i> .....	60
<i>B.Backend (Server-Side):</i> .....	61
<i>C.Integrasi Whatsapp API:</i> .....	61
<i>D.Deployment:</i> .....	62
DAFTAR PUSTAKA .....	90
KREDIT GAMBAR.....	92
Tentang Penulis .....	93

# BAB I

## MICRO *frontend*

### 1. *Micro frontend*

*Micro frontend* adalah sebuah konsep yang terbilang cukup baru tetapi telah mendapatkan popularitas cukup signifikan di kalangan praktisi pada bidang ini. *Micro frontend* menerapkan gaya arsitektur tunggal dari *microservice*, oleh karena itu *Micro frontend* menerapkan prinsip-prinsip dari *microservice* yang diterapkan oleh, yaitu *modeled around business domains*, *culture of automation*, *hide implementation details*, *decision decentralization*, *independent deployment*, serta *failure isolation*. Menurut, seiring dengan berjalannya waktu, pengembangan aplikasi dari sisi *frontend* semakin meningkat kompleksitasnya, salah satu penyebabnya adalah dari sisi *client-side*, aplikasi terus berkembang sehingga pengembangannya menjadi lebih sulit untuk diukur terutama jika melibatkan juga tim yang berbeda-beda dalam satu *front-end* yang sama secara bersamaan. Dengan adanya masalah tersebut, *micro frontend* hadir untuk memperluas konsep dari *microservice* ke dalam sisi *frontend* aplikasi. Ide dibalik konsep *Micro frontend* adalah untuk berpikir bahwa suatu *web,site* merupakan gabungan dari beberapa fitur yang masing-masing fitur dimiliki oleh tim yang independen. Setiap tim memiliki tujuan bisnis yang harus dicapai masing-masing, dan melakukan *development* dengan konsep *end-to-end*, yaitu dari *database* hingga ke *user interface (UI)*.

**A. Beberapa fitur penting dari *Micro frontend*:**

- 1. Pemisahan Antarmuka Pengguna (UI):** *Microfrontend* memungkinkan pemisahan antarmuka pengguna menjadi modul-modul yang independen. Setiap modul dapat dikembangkan oleh tim yang berbeda, memungkinkan pemeliharaan dan pengembangan yang lebih efisien.
- 2. Skalabilitas Tim Pengembangan:** Dengan membagi aplikasi menjadi *mikrofrontend*, tim pengembangan dapat diorganisir dan berfokus pada bagian-bagian spesifik. Ini mendukung pengembangan berskala dan memungkinkan tim bekerja secara independen pada modul yang mereka tangani.
- 3. Penggunaan Teknologi *frontend* yang Beragam:** *Microfrontend* memungkinkan penggunaan teknologi *frontend* yang berbeda-beda di berbagai bagian aplikasi. Sebagai contoh, satu modul dapat menggunakan *React*, sementara yang lain menggunakan *Angular* atau *Vue.js*, sesuai dengan kebutuhan dan preferensi tim pengembangan.
- 4. Pengembangan Aplikasi *Real-Time*:** *Microfrontend* cocok untuk pengembangan aplikasi *real-time*, seperti aplikasi obrolan atau kolaborasi, karena memungkinkan pengembangan dan pemeliharaan yang cepat pada bagian-bagian spesifik yang terlibat dalam interaksi *real-time*.
- 5. Perbaikan Kinerja dan Responsivitas:** Dengan memungkinkan pengembangan independen pada modul-modul tertentu, *Microfrontend* dapat mendukung perbaikan kinerja dan responsivitas aplikasi. Pembaharuan pada satu modul tidak memengaruhi seluruh aplikasi, memungkinkan untuk pengoptimalan lebih cepat.
- 6. Pengembangan Aplikasi *Cross-Functional*:** *Microfrontend* memfasilitasi pengembangan aplikasi oleh tim yang memiliki keahlian fungsional dan bisnis tertentu. Misalnya, tim pengembangan dapat fokus pada pengembangan *frontend* untuk fitur tertentu tanpa harus bergantung pada tim lain.

7. **Integrasi dengan Teknologi *Server-Side Rendering (SSR)*:** *Microfrontend* dapat digunakan dalam pengembangan aplikasi dengan *SSR*, memungkinkan pengoptimalan kinerja dan pengalaman pengguna pada sisi *server*.
8. **Penyusunan dan Penyelarasan Proyek yang Lebih Baik:** *Microfrontend* membantu dalam penyusunan dan penyelarasan proyek yang lebih baik, karena memungkinkan fokus pada spesifik bagian aplikasi tanpa harus mengkoordinasikan seluruh tim untuk setiap perubahan.
9. **Penggunaan Alat dan Kerangka Kerja Terkini:** *Microfrontend* mendukung penggunaan alat dan kerangka kerja terkini dalam pengembangan *frontend*. Ini memungkinkan pengembang untuk memilih teknologi yang paling sesuai untuk tugas tertentu.
10. **Pengembangan dan Penyelarasan Independen:** *Microfrontend* memungkinkan pengembangan dan penyelarasan independen pada modul-modul tertentu tanpa mempengaruhi bagian lain dari aplikasi.

Dalam pengembangan *Micro frontend* beberapa tools dan teknologi yang sering digunakan antara lain:

1. ***Webpack Module Federation*:** *Webpack Module Federation* adalah fitur *Web,pack* yang memungkinkan modul *JavaScript* untuk berbagi kode secara dinamis di antara aplikasi atau *mikrofrontend* yang terpisah. Ini sangat berguna untuk mengintegrasikan modul-modul *frontend* secara efisien.
2. ***Single-SPA*:** *Single-SPA* adalah kerangka kerja *JavaScript* yang dirancang khusus untuk pengembangan *microfrontend*. Ini memungkinkan integrasi modul-modul *frontend* yang dikembangkan secara independen dan berbasis teknologi yang berbeda-beda.
3. ***Module Federation Plugin (for Other Frameworks)*:** Beberapa kerangka kerja *frontend* seperti *React*, *Angular*, dan *Vue.js* memiliki dukungan atau *plugin* sendiri untuk implementasi *Module Federation*. Misalnya, *React* dapat menggunakan "*React-module-federation*" untuk mendukung integrasi modul-modul.

4. **SystemJS:** *SystemJS* adalah loader modul *JavaScript* yang mendukung *dynamic loading* dan bisa digunakan untuk mengimplementasikan *microfrontend*. Ini membantu dalam memuat dan mengeksekusi modul-modul secara dinamis saat diperlukan.
5. **Microfrontend CLI Tools:** Beberapa alat pengembangan berbasis baris perintah (*CLI*) dikembangkan khusus untuk membantu pembuatan, pengujian, dan penerapan *microfrontend*. Contohnya termasuk "*create-Single-SPA*" dan "*mf-Scripts*."
6. **Lerna:** *Lerna* adalah alat manajemen paket yang memungkinkan untuk mengelola beberapa paket atau modul dalam satu repositori. Ini dapat digunakan untuk memudahkan pengembangan dan penerapan *Microfrontend* dengan mengelola dependensi dan versi paket.
7. **Yarn Workspaces:** *Yarn Workspaces* adalah fitur dalam manajer paket *Yarn* yang memungkinkan untuk mengelola beberapa paket dalam satu repositori. Ini membantu dalam manajemen dependensi di lingkungan pengembangan yang terdiri dari beberapa *mikrofrontend*.
8. **Vue Micro frontends:** *Vue Micro frontends* adalah ekstensi *Vue.js* yang menyediakan alat dan pedoman untuk pengembangan *Microfrontend* menggunakan *Vue.js*. Ini mempermudah integrasi modul-modul *Vue.js* dalam arsitektur *microfrontend*.
9. **React Micro frontends:** *React Micro frontends* adalah pendekatan dan alat untuk mengembangkan *mikrofrontend* menggunakan *React*. Ini memberikan panduan dan alat untuk membangun aplikasi skala besar dengan memisahkan dan mengelola modul-modul *React* secara independen.
10. **Redux atau Context API (untuk Manajemen Keadaan):** Untuk manajemen keadaan global, terutama dalam kasus aplikasi skala besar, alat seperti *Redux* atau *Context API* dalam *React* dapat digunakan. Ini membantu dalam berbagi keadaan aplikasi di antara modul-modul *frontend*.

## B. Cara Kerja *Micro frontend*:

1. **Pembagian Antarmuka Pengguna:** Antarmuka pengguna (*UI*) aplikasi *web* dibagi menjadi modul-modul kecil yang dapat berdiri sendiri. Modul-modul ini dapat mencakup bagian-bagian seperti *header*, *sidebar*, konten utama, dan lainnya. Setiap modul ini berisi kode, gaya, dan sumber daya yang diperlukan untuk menangani fungsionalitas spesifik.
2. **Pengembangan Independen:** Setiap modul dikembangkan secara independen oleh tim pengembangnya. Tim dapat menggunakan teknologi *frontend* yang berbeda-beda, dan mereka dapat mengambil keputusan teknis yang paling sesuai untuk modul mereka. Hal ini memungkinkan pembaruan dan perbaikan dilakukan tanpa harus memengaruhi bagian-bagian lain dari aplikasi.
3. **Manajemen Keadaan Global (Opsional):** Beberapa proyek *Microfrontend* menggunakan manajemen keadaan global, seperti *Redux* atau *Context API* dalam *React*, untuk berbagi keadaan dan data di antara modul-modul *frontend*. Ini membantu menjaga konsistensi keadaan aplikasi di seluruh *mikrofrontend*.
4. **Komunikasi Antar Modul:** Modul-modul *frontend* dapat berkomunikasi satu sama lain melalui antarmuka publik yang ditetapkan. Beberapa teknik yang digunakan untuk komunikasi antarmodul melibatkan penggunaan *URL*, peristiwa (*events*), atau *API* khusus yang dibuat untuk tujuan tersebut. Komunikasi ini memungkinkan modul satu untuk menyampaikan informasi atau permintaan ke modul lainnya.
5. **Integrasi pada Level *UI*:** Modul-modul *frontend* diintegrasikan pada tingkat *UI*. Beberapa kerangka kerja dan alat, seperti *Webpack Module Federation* atau *Single-SPA*, dapat digunakan untuk mengelola integrasi ini secara dinamis dan efisien. Modul-modul ini dapat di-load secara asinkron sesuai dengan kebutuhan aplikasi.

6. **Pengujian Independen:** Setiap modul dapat diuji secara independen untuk memastikan bahwa mereka berfungsi sebagaimana mestinya. Pengujian ini dapat mencakup pengujian unit, integrasi, dan pengujian untuk memastikan bahwa setiap modul berinteraksi dengan benar dengan yang lainnya.
7. **Penerapan Independen:** Modul-modul dapat diimplementasikan secara independen, memungkinkan untuk pembaruan atau perubahan pada satu modul tanpa harus merilis ulang seluruh aplikasi. Hal ini mendukung praktik *continuous deployment (CI/CD)* untuk pengembangan yang cepat.
8. **Manajemen Versi:** Manajemen versi dapat dilakukan secara independen pada setiap modul. Ini memungkinkan penggunaan versi yang berbeda-beda untuk modul-modul tertentu, membantu mengelola perbedaan versi dan dependensi.
9. **Koordinasi Integrasi dan Pengembangan:** Meskipun modul-modul *frontend* dikembangkan secara independen, masih dibutuhkan koordinasi untuk integrasi dan pengembangan aplikasi secara keseluruhan. Tim koordinasi atau manajemen proyek memastikan bahwa integrasi berjalan lancar dan bahwa modul-modul bekerja bersama-sama secara harmonis.

### C. Fungsi Micro *frontend*:

*Microfrontend* memiliki beberapa fungsi yang memberikan keuntungan dalam pengembangan aplikasi *web* skala besar.

Berikut adalah beberapa fungsi utama dari arsitektur *microfrontend*:

1. **Pembagian Tugas dan Tim Pengembangan:** *Microfrontend* memungkinkan pembagian tugas yang jelas di antara tim pengembangan. Setiap tim dapat bertanggung jawab atas pengembangan, pemeliharaan, dan evolusi modul *frontend* yang spesifik tanpa harus berkoordinasi secara ketat dengan tim lain.

2. **Independensi Pengembangan dan Pemeliharaan:** Modul-modul *frontend* dapat dikembangkan secara independen. Perubahan pada satu modul tidak memerlukan perubahan pada modul lainnya, memungkinkan tim untuk merilis pembaruan atau perbaikan tanpa mempengaruhi bagian lain dari aplikasi.
3. **Skalabilitas Pengembangan:** Dengan membagi aplikasi menjadi modul-modul, *Microfrontend* mendukung pengembangan secara berskala. Tim dapat menambahkan atau menghapus modul sesuai dengan kebutuhan tanpa mengganggu modul-modul lainnya.
4. **Pemilihan Teknologi yang Fleksibel:** *Microfrontend* memungkinkan penggunaan teknologi *frontend* yang berbeda-beda untuk setiap modul. Ini memungkinkan tim untuk memilih teknologi yang paling sesuai dengan kebutuhan dan keahlian mereka.
5. **Manajemen Kode yang Lebih Baik:** Setiap modul memiliki tanggung jawab yang jelas dan terbatas, membuat manajemen kode menjadi lebih terkelola. Tim dapat fokus pada pengembangan dan pemeliharaan modul tertentu tanpa harus terjebak dalam kompleksitas keseluruhan aplikasi.
6. **Integrasi yang Dinamis:** *Microfrontend* memungkinkan integrasi dinamis modul-modul *frontend* pada tingkat *UI*. Modul-modul dapat dimuat dan diintegrasikan secara dinamis sesuai dengan kebutuhan aplikasi, memberikan fleksibilitas dalam mengelola tampilan dan fungsionalitas.
7. **Perbaikan dan Pembaruan yang Cepat:** Karena modul-modul dapat dirilis secara independen, perbaikan dan pembaruan dapat dilakukan dengan cepat tanpa harus menunggu atau mengkoordinasikan dengan tim pengembangan lainnya. Hal ini mendukung praktik *continuous deployment (CI/CD)*.
8. **Keterlibatan Tim yang Lebih Tinggi:** Tim pengembangan lebih terlibat dan bertanggung jawab atas bagian-bagian spesifik dari aplikasi. Hal ini dapat meningkatkan keterlibatan dan kepuasan tim karena mereka memiliki kendali lebih besar atas modul yang mereka tangani.



9. **Manajemen Keadaan Global:** Beberapa implementasi *Microfrontend* menggunakan manajemen keadaan global untuk berbagi keadaan dan data di antara modul-modul *frontend*. Ini membantu menjaga konsistensi antara modul-modul.
10. **Evolutivitas yang Lebih Baik:** *Microfrontend* mendukung evolusi aplikasi secara lebih fleksibel. Modul-modul dapat diperbarui atau diganti tanpa harus mempengaruhi aplikasi secara keseluruhan, memungkinkan pengembangan berkelanjutan.

Penting untuk diingat bahwa sementara *Microfrontend* memberikan keuntungan dalam pengembangan aplikasi *web*, penerapannya juga memerlukan pemahaman yang baik, manajemen keadaan yang efektif, dan koordinasi antar tim yang baik untuk memaksimalkan potensinya.

#### D. Cara Membangun *Micro frontend*:

Membangun *Microfrontend* melibatkan beberapa langkah kunci untuk memastikan pemisahan dan integrasi yang efisien dari modul-modul *frontend* yang berdiri sendiri. Berikut adalah langkah-langkah umum untuk membangun *microfrontend*:

1. **Desain Arsitektur *Microfrontend*:** Tentukan bagaimana aplikasi *web* akan dibagi menjadi modul-modul *frontend*. Identifikasi fungsionalitas dan tampilan yang dapat dikelola secara independen oleh tim pengembangan. Pilih teknologi *frontend* yang sesuai dengan kebutuhan.
2. **Pilih dan Konfigurasi Alat Pengembangan:** Pilih alat dan kerangka kerja yang mendukung pengembangan *microfrontend*. Beberapa opsi termasuk *Web*, pack dengan *Module Federation*, *Single-SPA*, atau kerangka kerja khusus seperti *Vue Micro frontends* atau *React Micro frontends*.
3. **Buat Struktur Repositori:** Buat struktur repositori yang memungkinkan untuk menyimpan modul-modul *frontend* secara terpisah. Gunakan alat manajemen paket seperti *Lerna* atau *Yarn*

*Workspaces* jika diperlukan.

4. **Pengembangan Modul-Modul *frontend*:** Setiap tim pengembangan bertanggung jawab atas pengembangan satu atau beberapa modul *frontend*. Pastikan modul-modul tersebut memiliki tanggung jawab yang jelas dan dapat berdiri sendiri.
5. **Integrasi dengan Alat *Webpack Module Federation* (Opsional):**  
Jika menggunakan *Webpack* dengan *Module Federation*, konfigurasi modul-modul untuk diintegrasikan dinamis pada tingkat *UI*. Tentukan bagaimana modul-modul akan berkomunikasi satu sama lain.
6. **Manajemen Keadaan Global (Opsional):**  
Jika diperlukan, pilih dan konfigurasi alat manajemen keadaan global seperti *Redux* atau *Context API* dalam *React*. Setiap modul dapat berbagi dan mengakses keadaan aplikasi secara konsisten.
7. **Penanganan *Routing* (Opsional):**  
Atur penanganan *routing* antar modul-modul *frontend*. Pastikan navigasi antar modul dapat diatur dengan baik dan sesuai dengan kebutuhan aplikasi.
8. **Pengujian Independen:**  
Lakukan pengujian pada setiap modul secara independen untuk memastikan fungsionalitas dan integrasi yang benar. Lakukan pengujian unit, integrasi, dan pengujian *end-to-end* sesuai kebutuhan.
9. **Penerapan Independen:**  
Setiap modul dapat diimplementasikan secara independen. Pastikan bahwa pembaruan atau perubahan pada satu modul tidak mempengaruhi modul-modul lainnya.
10. **Manajemen Versi:**  
Kelola manajemen versi setiap modul secara terpisah. Pastikan bahwa setiap modul dapat menggunakan versi yang sesuai dengan kebutuhan dan dependensi yang kompatibel.

### **11. Integrasi *Continuous Deployment (CI/CD)*:**

Pastikan bahwa *integrasi CI/CD* telah diatur untuk memfasilitasi penerapan dan pengujian berkelanjutan dari setiap modul *frontend*.

### **12. Dokumentasi:**

Buat dokumentasi yang jelas untuk setiap modul dan integrasi antar modul. Dokumentasikan antarmuka publik, dependensi, dan cara mengembangkan serta menguji setiap modul.

### **13. *Monitoring* dan *Debugging*:**

Pastikan bahwa alat *monitoring* dan *debugging* telah diatur untuk memudahkan pemantauan kinerja dan identifikasi masalah di seluruh aplikasi.

### **14. Pelatihan Tim:**

Berikan pelatihan kepada tim pengembangan tentang cara menggunakan dan mengembangkan modul-modul *frontend*. Pastikan pemahaman yang baik tentang arsitektur dan konvensi pengembangan yang diikuti.

### **15. Evaluasi dan Optimasi:**

Secara berkala, lakukan evaluasi kinerja dan optimasi pada aplikasi. Pertimbangkan apakah ada modul yang perlu diperbarui atau digantikan, dan pastikan bahwa integrasi masih berfungsi dengan baik.

Langkah-langkah ini dapat disesuaikan sesuai dengan alat dan kerangka kerja yang digunakan, serta kebutuhan spesifik proyek. Penting untuk terus memonitor dan memperbaiki arsitektur *Microfrontend* seiring berjalannya waktu sesuai dengan evolusi kebutuhan proyek dan teknologi.

# BAB II

## PHP, CSS & MySQL

### 1. PHP (Hypertext Preprocessor)

*PHP (Hypertext Preprocessor)* adalah bahasa pemrograman skrip sisi *server* yang digunakan secara luas untuk pengembangan *web*. *PHP* terintegrasi dengan *HTML* dan digunakan untuk membuat halaman *web*, dinamis, berinteraksi dengan basis data, dan menjalankan tugas di sisi *server*. Karakteristiknya mencakup kemampuan pemrograman *web* interaksi dengan basis data, fungsi *server-side*, kemampuan menangani formulir, dan fleksibilitas.

*PHP* telah menjadi salah satu bahasa pemrograman *web* yang paling populer dan banyak digunakan. Meskipun ada banyak bahasa pemrograman *web* lainnya, *PHP* tetap menjadi pilihan yang kuat untuk proyek-proyek *web*, terutama karena dukungan yang luas, dokumentasi yang baik, dan komunitas pengembang yang besar.

#### A. Karakteristik dan penggunaan utama PHP:

1. **Bahasa Sisi Server:** *PHP* dijalankan di sisi *server*, artinya proses pemrosesan dan eksekusi kode *PHP* terjadi di *server web* sebelum hasilnya dikirim ke perangkat pengguna.
2. **Pemrograman Web:** *PHP* dirancang khusus untuk pengembangan *web*. Dengan sintaksis yang mirip dengan *HTML*, *PHP* memungkinkan integrasi yang mudah dengan markup bahasa *HTML*.
3. **Sintaksis Dinamis:** *PHP* memiliki sintaksis yang dinamis dan mudah dipahami, menjadikannya ramah bagi pengembang pemula dan juga memberikan fleksibilitas bagi pengembang berpengalaman.
4. **Interaksi dengan Basis Data:** *PHP* dapat berinteraksi dengan berbagai jenis basis data, seperti *MySQL*, *PostgreSQL*, *Oracle*, dan lainnya. Ini

memungkinkan aplikasi *web*, menggunakan data dinamis dari basis data.

5. **Fungsi *Server-Side*:** *PHP* digunakan untuk melakukan berbagai tugas di sisi *server*, termasuk pengolahan formulir, mengelola *cookie* dan sesi, pembuatan, pembacaan, dan penulisan file, dan banyak lagi.
6. **Kemampuan untuk Menangani Formulir:** *PHP* sangat efektif dalam menangani data formulir dari halaman *web* yang merupakan fungsi penting dalam pengembangan aplikasi *web*.
7. **Kemampuan untuk Menghasilkan Konten Dinamis:** *PHP* memungkinkan pengembang untuk menghasilkan konten dinamis yang dapat berubah berdasarkan waktu, *input* pengguna, atau kondisi lainnya.
8. **Pengembangan Aplikasi Berbasis *Web*:** *PHP* banyak digunakan untuk membangun berbagai jenis aplikasi *web*,, termasuk , situs *e-commerce*, *forum*, manajemen konten, dan aplikasi *web*, lainnya.
9. ***Open source*:** *PHP* adalah perangkat lunak sumber terbuka, yang berarti dapat digunakan dan dimodifikasi secara gratis oleh siapa saja. Komunitas *PHP* yang besar juga menyediakan dukungan dan sumber daya yang berlimpah.
10. **Fleksibilitas:** *PHP* dapat diintegrasikan dengan berbagai teknologi dan *server web*,, membuatnya sangat fleksibel dan dapat digunakan di banyak lingkungan pengembangan.

## **B. Kelebihan PHP:**

### **1. Mudah Dipahami dan Dipelajari:**

- *PHP* memiliki sintaksis yang mirip dengan *C* dan *Java*, membuatnya relatif mudah dipahami dan dipelajari, terutama bagi pemula.

### **2. Integrasi dengan *HTML*:**

- *PHP* dapat diintegrasikan dengan mudah dalam kode *HTML*, memungkinkan pengembangan aplikasi *web*, yang dinamis dengan lebih efisien.

### **3. *Open source*:**

- *PHP* adalah perangkat lunak sumber terbuka, artinya dapat digunakan dan dimodifikasi secara gratis. Komunitas *PHP* yang besar memberikan dukungan dan kontribusi berlimpah.

### **4. Fleksibilitas:**

- *PHP* dapat dijalankan di berbagai *server web*, dan *platform*, memberikan fleksibilitas dalam memilih lingkungan pengembangan.

### **5. Manajemen *Basis Data*:**

- *PHP* memiliki dukungan kuat untuk koneksi dan interaksi dengan berbagai *jenis basis data*, termasuk *MySQL*, *PostgreSQL*, dan *Oracle*.

### **6. Kemampuan Menangani Formulir:**

- *PHP* memiliki fitur yang memudahkan pengolahan data formulir dari halaman *web*, yang sangat penting dalam pengembangan aplikasi *web*.

### **7. Pustaka dan Kerangka Kerja:**

- *PHP* memiliki banyak pustaka dan kerangka kerja (seperti *Laravel*, *Symfony*, dan *CodeIgniter*) yang memudahkan pengembangan aplikasi dengan struktur yang terorganisir.

### **8. Ketersediaan Hosting:**

- Banyak penyedia *hosting web*, mendukung *PHP*, membuatnya mudah untuk

menyebarkan aplikasi *PHP* di berbagai lingkungan *hosting*.

### C. Kekurangan PHP:

**1. Kinerja:** Meskipun *PHP* telah mengalami banyak perbaikan kinerja, bahasa ini mungkin tidak secepat bahasa pemrograman tingkat rendah atau yang dikompilasi seperti *C++* atau *Java*.

**2. Ketidakamanan Default:** Beberapa kebijakan keamanan *PHP* yang *default* dapat menjadi risiko keamanan jika tidak dikonfigurasi dengan benar. Perlu perhatian ekstra untuk memitigasi potensi risiko keamanan.

**3. Manajemen Memori:** *PHP* tidak memiliki kontrol memori yang sebaik bahasa pemrograman lainnya. Hal ini dapat mengakibatkan masalah kinerja jika tidak dikelola dengan baik.

**4. Ekosistem yang Besar dan Beragam:** Meskipun keberagaman dapat dianggap sebagai kelebihan, tetapi juga dapat menjadi kekurangan karena adanya banyak pustaka dan kerangka kerja yang berbeda membuat pemilihan menjadi sulit.

**5. Ketergantungan pada Apache:** *PHP* biasanya dijalankan sebagai modul *Apache*, dan itu dapat membatasi fleksibilitas penggunaan di luar *server web*, *Apache*.

**6. Skalabilitas *horizontal* yang Terbatas:** Dalam beberapa kasus, *PHP* dapat menghadapi batasan dalam skalabilitas *horizontal* di mana aplikasi perlu disesuaikan untuk meningkatkan kinerja secara *linier* dengan penambahan sumber daya.

**7. Penanganan Kesalahan:** Penanganan kesalahan dalam *PHP* dapat kurang deskriptif, yang bisa menyulitkan proses *debugging*.

**8. Kurangnya Dukungan untuk Program Berorientasi Objek pada Versi Awal:** Versi awal *PHP* kurang mendukung paradigma pemrograman berorientasi objek, meskipun hal ini telah berubah dengan peningkatan versi.

Penting untuk dicatat bahwa beberapa kekurangan dapat diatasi atau dikelola dengan baik melalui praktik pengembangan terbaik dan konfigurasi yang cermat. Keputusan untuk menggunakan *PHP* harus dipertimbangkan dengan memperhatikan kebutuhan spesifik proyek dan preferensi pengembang.

#### **D. Cara penggunaan *PHP* :**

Penggunaan *PHP* melibatkan penulisan skrip *PHP* yang terintegrasi dengan *HTML* atau digunakan secara terpisah untuk mengolah data dan menghasilkan konten dinamis di sisi *server*.

Untuk contohnya bisa diakses pada link *GitHub* dibawah ini:  
<https://GitHub.com/WASENDERAUTO/MATERI-BUKPED.git>

## **2. *CSS (Cascading Style Sheets)***

*CSS (Cascading Style Sheets)* adalah bahasa *stylesheet* untuk mengontrol presentasi dan tata letak elemen *HTML* pada halaman *web*. *CSS* memungkinkan pengembang untuk memberikan gaya visual, mengatur tata letak, membuat desain responsif, menerapkan animasi, menggunakan *pseudo-classes* dan *pseudo-elements*, dan mengontrol keutamaan gaya dengan prinsip *cascading*. *CSS* membantu memisahkan struktur *HTML* dari tampilan dan gaya, meningkatkan efisiensi pengelolaan desain halaman *web*,.

### **A. Pengembangan *CSS***

*CSS* memungkinkan pengembang untuk:

**1. Memberikan Gaya Visual:** Menentukan warna, jenis *font*, ukuran *font*, dan properti-properti tampilan lainnya untuk elemen-elemen *HTML*.

**2. Mengatur Tata Letak:** Mengendalikan penempatan, ukuran, dan jarak antara elemen-elemen *HTML* menggunakan properti seperti *margin*, *padding*, dan *float*.

**3. Responsif:** Menggunakan media *queries* untuk membuat desain responsif yang dapat menyesuaikan tampilan halaman *web*, berdasarkan ukuran layar perangkat pengguna.



**4. Animasi dan Transisi:** Menerapkan efek animasi dan transisi untuk memberikan pengalaman pengguna yang lebih dinamis.

**5. Pseudo-classes dan Pseudo-elements:** Menggunakan *pseudo-classes* (seperti *hover* untuk keadaan *hover*) dan *pseudo-elements* (seperti: *before* untuk menambahkan elemen sebelum konten) untuk memberikan gaya khusus pada elemen dalam keadaan tertentu.

**6. Menerapkan Gaya Berbasis Keadaan:** Memberikan gaya berdasarkan keadaan elemen, seperti tombol yang berubah warna saat dihover.

**7. Kontrol Terhadap Kasus Spesifik (*Specificity*):** Menggunakan aturan CSS untuk menentukan keutamaan gaya saat ada beberapa aturan yang berlaku pada elemen yang sama.

CSS bekerja dengan prinsip *cascading*, yang berarti aturan-aturan gaya dapat bersarang dan saling menimpa berdasarkan tingkat keutamaan. Pengembang dapat menyusun aturan-aturan CSS dalam file terpisah atau menuliskannya langsung dalam elemen *HTML* menggunakan atribut *style*.

Untuk contohnya bisa diakses pada link *GitHub* dibawah ini:  
<https://GitHub.com/WASENDERAUTO/MATERI-BUKPED.git>

### 3. MYSQL

*MySQL* adalah sebuah sistem manajemen basis data (*DBMS*) yang bersifat relasional. *DBMS* relasional adalah sebuah program perangkat lunak yang digunakan untuk membuat dan mengelola basis data relasional. Basis data relasional adalah kumpulan data yang terorganisir dalam tabel yang saling terhubung satu sama lain melalui kunci atau hubungan tertentu. Pengguna *MySQL* dapat berinteraksi dengan basis data menggunakan perintah *SQL* untuk melakukan operasi seperti pengambilan data (*query*), penambahan data (*insert*), pembaruan data (*update*), dan penghapusan data (*delete*). *MySQL* juga mendukung penyimpanan prosedur dan fungsi yang memungkinkan pengguna untuk membuat logika pemrograman di dalam basis data. Dengan kepopulerannya dan sifat *open-source*-nya, *MySQL* sering menjadi pilihan utama untuk pengembangan aplikasi *web*, dan proyek-proyek perangkat lunak yang membutuhkan sistem manajemen basis data relasional.

### A. Apa perbedaan menggunakan *MySQL* & *SQL* ?

Penting untuk memahami bahwa perbandingan antara *MySQL* dan *SQL* bukanlah perbandingan yang sepenuhnya tepat. *SQL* (*Structured Query Language*) adalah sebuah bahasa pemrograman khusus yang digunakan untuk mengelola dan berinteraksi dengan basis data relasional. *MySQL*, di sisi lain, adalah salah satu implementasi dari *DBMS* (*Database Management System*) yang mendukung *SQL*. Jadi, sebenarnya, *MySQL* menggunakan *SQL* sebagai bahasa utama untuk berinteraksi dengan basis datanya.

Berikut adalah beberapa konsep yang dapat membantu Anda memahami perbedaan antara *MySQL* dan *SQL*:

**1. *SQL* sebagai Bahasa:** *SQL* adalah bahasa standar yang digunakan untuk mengelola basis data relasional. Ini mencakup operasi seperti pengambilan data (*SELECT*), penambahan data (*INSERT*), pembaruan data (*UPDATE*), penghapusan data (*DELETE*), dan operasi lainnya yang terkait dengan manipulasi dan definisi data.

**2. *MySQL* sebagai *DBMS*:** *MySQL*, pada dasarnya, adalah sebuah sistem manajemen basis data (*DBMS*) yang mendukung bahasa *SQL*. Ini menyediakan lingkungan di mana Anda dapat membuat, mengelola, dan mengakses basis data relasional. *MySQL* tidak identik dengan *SQL* sebaliknya, *MySQL* adalah salah satu implementasi dari *DBMS* yang mendukung penggunaan *SQL*.

**3. Implementasi *DBMS* Lain:** Selain *MySQL*, ada banyak *DBMS* lain yang juga mendukung *SQL*, seperti *PostgreSQL*, *SQLite*, *Microsoft SQL Server*, *Oracle Database*, dan banyak lagi. Masing-masing dari *DBMS* ini adalah implementasi dari *SQL*, tetapi memiliki fitur, kinerja, dan karakteristik unik mereka sendiri.

Jadi, singkatnya, *SQL* adalah bahasa yang digunakan untuk mengelola basis data relasional, sementara *MySQL* adalah salah satu dari banyak *DBMS* yang menggunakan *SQL* sebagai bahasa utama untuk berinteraksi dengan basis data. Ketika seseorang berbicara tentang menggunakan *MySQL*, sebenarnya mereka berbicara tentang menggunakan *MySQL* sebagai platform *DBMS* yang mendukung *SQL* untuk mengelola data

## B. Mengapa menggunakan MySQL ?

Beberapa alasan mengapa seseorang atau organisasi memilih untuk menggunakan *MySQL* sebagai sistem manajemen basis data (*DBMS*) mereka.

1. **Open source:** *MySQL* bersifat *open-source*, dapat diunduh, digunakan, dan dimodifikasi secara gratis, menjadi pilihan ekonomis untuk pengembang dan organisasi dengan anggaran terbatas.
2. **Komunitas Besar:** Komunitas pengguna *MySQL* yang besar mendukung pertukaran informasi, dukungan teknis, dan pengembangan ekstensi atau plugin tambahan.
3. **Kinerja Tinggi:** *MySQL* dirancang untuk kinerja tinggi, mampu menangani beban kerja besar dan memberikan indeks efisien untuk pencarian cepat. Dapat dioptimalkan sesuai kebutuhan aplikasi.
4. **Dukungan Standar SQL:** Mendukung *SQL*, memudahkan penggunaan dan pindah antar *DBMS* yang mendukung *SQL* tanpa banyak perubahan pada kueri dan perintah dasar.
5. **Skalabilitas:** Dapat diukur secara *horizontal* dan *vertikal*, mendukung penambahan *server* atau peningkatan daya pemrosesan dan memori pada satu *server*.
6. **Dukungan dari Industri:** Diterima secara luas di industri dan digunakan oleh banyak perusahaan besar, memberikan kepercayaan kepada pengguna terkait keandalan produk.
7. **Ketersediaan Cross-Platform:** Dapat dijalankan di berbagai sistem operasi, memberikan fleksibilitas dalam pemilihan platform untuk aplikasi yang dikembangkan.
8. **Keamanan:** Menyediakan fitur keamanan seperti otentikasi pengguna, enkripsi data, dan kontrol akses yang dapat dikonfigurasi, memberikan tingkat keamanan yang baik.
9. **Replication dan Ketersediaan Tinggi:** Mendukung replikasi data

untuk tujuan pencadangan, distribusi beban kerja, dan mencapai ketersediaan tinggi melalui *failover*.

10. **Dukungan dari Oracle Corporation:** Oracle Corporation memberikan dukungan dan sumber daya tambahan untuk pengembangan dan pemeliharaan *MySQL*, meningkatkan kepercayaan pengguna terhadap kelangsungan dan dukungan.

*MySQL* adalah pilihan populer dalam lingkungan *web*, dan bisnis, namun, penting untuk mempertimbangkan kebutuhan spesifik proyek dan membandingkannya dengan *DBMS* lain sebelum membuat keputusan.

### C. Contoh kode program dalam menggunakan *MySQL*:

Berikut adalah beberapa contoh kode program menggunakan *MySQL* untuk operasi-operasi dasar seperti membuat tabel, menyisipkan data, mengambil data, memperbarui data, dan menghapus data.

Untuk contohnya bisa diakses pada link *GitHub* dibawah ini:  
<https://GitHub.com/WASENDERAUTO/MATERI-BUKPED.git>

### Kesimpulan:

PHP, *MySQL*, dan *CSS* merupakan elemen kunci dalam pengembangan *web*., *PHP* digunakan untuk logika sisi *server*, *MySQL* untuk manajemen basis data, dan *CSS* untuk mengatur tampilan dan gaya. Mereka bekerja bersama-sama untuk menciptakan aplikasi *web*, dinamis dengan desain yang menarik dan responsif. Dengan sifat *open-source* dan dukungan yang luas, PHP, *MySQL*, dan *CSS* menjadi pilihan yang kuat bagi pengembang *web*, dalam membangun berbagai jenis proyek *Online*.

# BAB III

## COMPOSER

### 1. *Composer*

*Composer* adalah manajer dependensi untuk *PHP* yang digunakan untuk mengelola paket-paket (*libraries* atau *frameworks*) yang digunakan dalam pengembangan proyek *PHP*. Dengan menggunakan *Composer*, Anda dapat dengan mudah menambahkan, menghapus, atau memperbarui paket-paket *PHP* dalam proyek Anda.

#### A. Konsep dasar *Composer*

Berikut beberapa konsep dasar seputar *Composer*:

1. **Paket (*Package*):** Paket dalam konteks *Composer* adalah kumpulan file *PHP* yang disertakan bersama dengan informasi mengenai dependensinya. Paket ini bisa berupa *libraries*, *frameworks*, atau alat-alat lainnya.
2. ***Composer.json*:** File *Composer.json* adalah file konfigurasi untuk proyek *PHP* yang mengandung informasi tentang proyek dan dependensinya. File ini berisi daftar paket yang dibutuhkan oleh ***Composer.lock*:** File *Composer.lock* berisi informasi tentang versi yang tepat dari setiap paket dan dependensinya. File ini dibuat saat instalasi paket pertama kali dan harus disertakan dalam pengelolaan versi.
3. ***Autoloading (autoload)*:** *Composer* menyediakan *autoloading* otomatis untuk kelas-kelas dalam proyek. Ini memudahkan penggunaan kelas-kelas tanpa perlu menyertakan banyak pernyataan *require* atau *include*.
4. ***Command-line Interface (CLI)*:** *Composer* dijalankan melalui *command-line (terminal)* dengan berbagai perintah seperti *install*, *update*, *require*, dan lainnya.

*Composer* sangat membantu dalam manajemen dependensi dalam pengembangan proyek *PHP*. Ini memastikan bahwa semua paket yang dibutuhkan proyek dapat diinstal dengan mudah dan versi yang tepat.

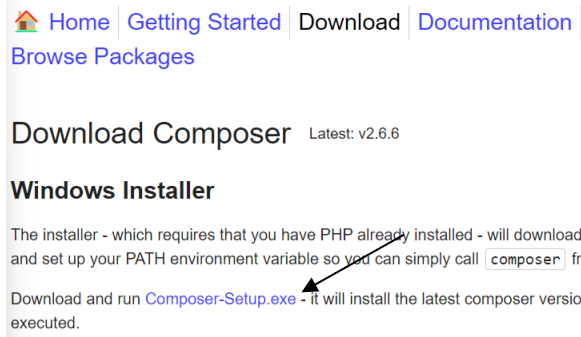
## B. Kelebihan & kekurangan *Composer*

- **Kelebihan:**
  1. **Manajemen Paket yang Efisien:** *Composer* memudahkan pengembang dalam mengelola dependensi dan paket *PHP*. Hal ini membuat instalasi dan pemeliharaan paket menjadi lebih mudah.
  2. **Otomatisasi Pengaturan Dependensi:** *Composer* secara otomatis menangani resolusi dependensi, memastikan bahwa versi paket yang diinstal kompatibel satu sama lain. Ini mencegah konflik dan masalah versi.
  3. **Reproducibility:** *Composer* memungkinkan pengembang untuk mendefinisikan semua dependensi dan versi yang tepat dalam file *Composer.json*, sehingga proyek dapat dengan mudah direproduksi pada lingkungan pengembangan lainnya.
  4. **Pengelolaan Autoloading:** *Composer* menyediakan *autoloaders* untuk mengatasi masalah autoloading kelas dalam proyek *PHP*. Ini memungkinkan penggunaan *namespace* dan *autoload* kelas tanpa perlu penyesuaian manual.
  5. **Ekosistem yang Kaya:** Terdapat banyak paket *PHP* yang tersedia melalui *Composer*, dan ekosistemnya terus berkembang. Pengembang dapat dengan mudah menemukan dan menggunakan paket-paket yang dibutuhkan untuk proyek mereka.
  6. **Dukungan untuk Proyek Besar:** *Composer* dapat dengan efisien menangani proyek-proyek *PHP* yang besar dengan banyak dependensi. Ini membuatnya cocok untuk pengembangan aplikasi kompleks.

- **Kekurangan :**
  1. **Waktu Instalasi Awal yang Lama:** Saat pertama kali menginstal *Composer* atau menginstal paket baru dalam proyek, waktu instalasi dapat terasa cukup lama, terutama jika proyek memiliki banyak dependensi.
  2. **Membutuhkan Koneksi Internet:** *Composer* memerlukan koneksi internet saat menginstal atau memperbarui paket. Hal ini dapat menjadi kendala jika pengembang bekerja di lingkungan tanpa koneksi internet yang stabil.
  3. **Keterbatasan dalam Proyek *Non-PHP*:** *Composer* dirancang khusus untuk manajemen paket *PHP*, sehingga tidak dapat digunakan dengan mudah untuk manajemen paket dalam proyek *non-PHP*.
  4. **Memerlukan Pemahaman yang Baik:** Untuk memanfaatkan sepenuhnya kelebihan *Composer*, pengembang perlu memahami dengan baik bagaimana mendefinisikan dan mengelola file *Composer.json*, serta cara menggunakan fitur-fitur *Composer* secara optimal.
  5. **Ketergantungan pada Ekosistem *Packagist*:** *Packagist* adalah repositori paket resmi untuk *Composer*. Ketergantungan yang kuat pada ekosistem *Packagist* bisa menjadi kelemahan jika repositori tersebut mengalami masalah atau tidak dapat diakses.

### C. Cara instalasi *Composer PHP* pada *windows*

1. Unduh *Composer* pada halaman <https://getcomposer.org/download/>



2. Pilih *Windows Installer*, dan unduh file 'exe' installer.
3. jalankan *File installer* yang sudah diunduh, dan instalasi.
4. Kemudian jalankan scrip berikut, pada terminal. Gunakan *Script* berikut:

```
PHP -r "copy('https://getcomposer.org/installer', 'Composer-setup.php');"  
PHP -r "if (hash_file('sha384', 'Composer-setup.php') ===  
'e21205b207c3ff031906575712edab6f13eb0b361f2085f1f1237b7126d785e826a  
450292b6cfd1d64d92e6563bbde02') { echo 'Installer verified'; } else { echo  
'Installer corrupt'; unlink('Composer-setup.php'); } echo PHP_EOL;"  
PHP Composer-setup.php  
PHP -r "unlink('Composer-setup.php');"
```

5. kemudian pada *directory PATH*, untuk memanggil *Composer* dari direktori manapun, gunakan kode dibawah ini:

```
sudo mv Composer.phar /usr/local/bin/Composer
```



- Menginstal *Composer* ke direktori tertentu dengan menggunakan opsi **-install-dir—** gunakan kode dibawah ini:

```
PHP Composer-setup.PHP --install-dir=bin
```

8. Tentukan nama file (**default: Composer.phar**) dengan menggunakan opsi **-filename** gunakan kode dibawah ini:

*PHP Composer-setup.PHP --filename=Composer*

9. Kemudian bisa menginstal *Composer* dengan versi tertentu menggunakan opsi **-version** gunakan kode dibawah ini:


*PHP Composer-setup.PHP --version=1.0.0-alpha8*

10. Setelah memilih versi, kemudian pada **'Download Channels'** untuk melihat detail yang lebih lanjut. gunakan kode dibawah ini:

*PHP Composer-setup.PHP --2.2*

**Contoh composer yang sudah diinstal :**

```
C:\Users\arffd>cd c:\xampp  
  
c:\xampp>composer
```



```
Composer version 2.5.4 2023-02-15 13:10:06  
  
Usage:  
command [options] [arguments]
```

# BAB VI

## JAVASCRIPT

### A. JavaScript

*JavaScript* adalah salah satu bahasa pemrograman populer saat ini. *JavaScript* ini punya banyak keunggulan yang membuatnya cocok untuk pemula. Untuk itu, belajar *JavaScript* sangat disarankan jika Anda ingin membuat *web,site*. Nah, Anda ingin belajar *JavaScript* tapi bingung harus mulai dari mana? Anda berada di tempat yang tepat! Karena di artikel ini, kami telah menyiapkan panduan *JavaScript* terlengkap. Anda akan mengetahui apa saja yang dibutuhkan untuk belajar. Mulai dari hal-hal yang perlu dipahami, seperti aturan coding *JavaScript*, penggunaan variabel dan tipe data, sampai mempraktikkan sendiri tutorial *JavaScript* ini.



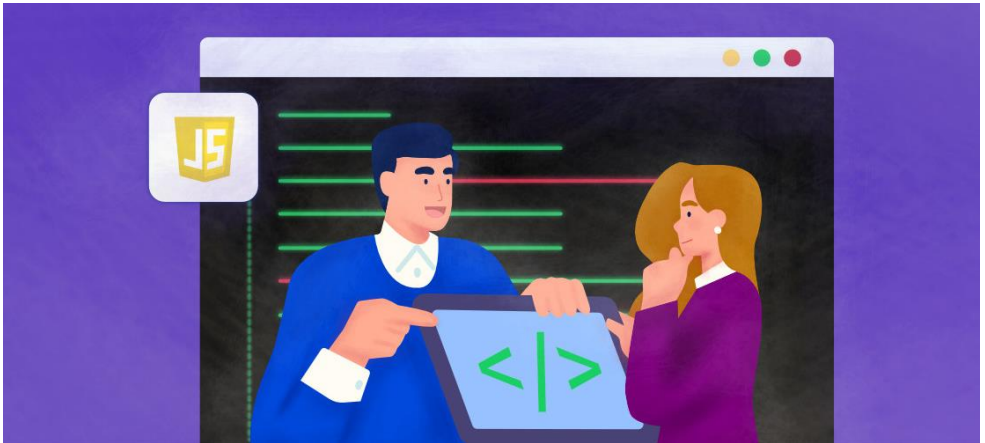
Gambar 2.1 *JavaScript*

#### Kekurangan:

1. Tidak membutuhkan resource memory besar (ringan dan kecil)
2. Mudah untuk dipelajari
3. Dinamis
4. *Multi platform*, bisa dijalankan dibanyak sistem operasi

### Kelebihan:

1. *Script* yang digunakan pada *JavaScript* tidak terenkripsi
2. Bukan untuk pengembangan aplikasi *stand alone*
3. Memiliki keterbatasan objek



Gambar 2.2 *JavaScript*

*JavaScript* adalah salah satu bahasa pemrograman populer saat ini. *JavaScript* ini punya banyak keunggulan yang membuatnya cocok untuk pemula. Untuk itu, belajar *JavaScript* sangat disarankan jika Anda ingin membuat *web,site*. Nah, Anda ingin belajar *JavaScript* tapi bingung harus mulai dari mana? Anda berada di tempat yang tepat! Karena di artikel ini, kami telah menyiapkan panduan *JavaScript* terlengkap. Anda akan mengetahui apa saja yang dibutuhkan untuk belajar. Mulai dari hal-hal yang perlu dipahami, seperti aturan coding *JavaScript*, penggunaan variabel dan tipe data, sampai mempraktikkan sendiri tutorial *JavaScript* ini.

## B. Apa itu *JavaScript*?



Gambar 2.3 *JavaScript*

*JavaScript* adalah bahasa pemrograman yang digunakan untuk *JavaScript* adalah bahasa pemrograman yang digunakan untuk membangun *website* dari sisi *client* (*client side*). *JavaScript* membuat *website* Anda menjadi lebih dinamis dan interaktif. Sesuatu yang tidak bisa dilakukan *HTML* dan *CSS*. Bahasa pemrograman *JavaScript* adalah bahasa tingkat tinggi, berjenis *Scripting* (*Scripting language*), ditulis secara dinamis (*dynamically typed*), dan dijalankan oleh *interpreter*. Apa maksudnya? *JavaScript* tergolong bahasa tingkat tinggi. Artinya ia punya aturan penulisan yang menyerupai bahasa manusia. Dengan begitu, belajar *JavaScript* jadi lebih mudah. Bahkan untuk pemula sekalipun. Sedangkan *Scripting language* maksudnya *JavaScript* adalah bahasa pemrograman yang tertanam di dalam program yang lebih besar, seperti *web*, *browser*. Jadi, yang bisa menggunakan *JavaScript* adalah program tersebut. Sementara *dynamically typed* artinya, satu *variabel* pada program dengan *coding JavaScript* dapat berisi tipe data yang berbeda. Anda bisa mengisi variabel dengan angka, kemudian menggantinya ke huruf tanpa mengalami *error*. *JavaScript* harus dijalankan oleh *interpreter*. Maksudnya, ia harus diterjemahkan ke dalam bahasa yang dimengerti komputer agar bisa dijalankan.

Nah, proses penerjemahan ini dilakukan secara otomatis oleh *web, browser*.



Gambar 2.4 Interpreter

*JavaScript* adalah salah satu bahasa tertua yang ada di dunia. Ia sudah ada sejak 1995 dengan nama awal *Mocca*. Lalu berganti menjadi *LiveScript*, sebelum akhirnya bernama *JavaScript* dengan standarisasi dari *EcmaScript*. Nah, *EcmaScript* sendiri adalah standarisasi bahasa *Scripting* yang dibuat oleh *European Computer Manufacturers Association (ECMA)*. Dengan adanya *EcmaScript*, *JavaScript* bisa berjalan dengan lancar di berbagai *browser*. Karena sudah ada cukup lama, tak heran kalau *JavaScript* adalah bahasa pemrograman terpopuler berdasarkan *survey Stack Overflow*. Sebanyak 97,7% *website* di *internet* menggunakan bahasa pemrograman *JavaScript*. Nah, setelah mengetahui apa itu *JavaScript*, berikutnya mari ketahui alasan Anda harus belajar *JavaScript*.

### C. Mengapa Anda Harus Belajar *JavaScript*?

Berikut adalah beberapa alasan mengapa Anda harus mulai belajar *JavaScript* sekarang juga:

#### 1. *Open source* dan Gratis

*JavaScript* adalah bahasa pemrograman *open source*. Artinya, Anda tidak perlu meminta izin lisensi untuk dapat menggunakan bahasa ini. Baik itu untuk belajar maupun keperluan lain yang lebih komersial. Selain itu, *JavaScript* juga gratis.

Dengan begitu, semua fiturnya dapat Anda gunakan secara penuh tanpa perlu membeli serial *number* atau bahkan *software* tertentu.

## 2. Tidak Memerlukan Instalasi

Seperti yang tadi dijelaskan, *JavaScript* berjalan di atas program lain seperti *web*, *browser*. Nah, semua *browser* yang ada di pasaran sudah memiliki semacam engine untuk dapat menjalankan *JavaScript*. Selama Anda punya *web*, *browser*, Anda bisa langsung melihat hasil coding *JavaScript*. Dengan begitu, Anda tidak perlu menginstall *web*, *server*, *compiler*, atau bahkan melakukan perubahan konfigurasi pada sistem.

## 3. Komunitas yang Luas

Sebagai bahasa pemrograman terpopuler, *JavaScript* punya dukungan komunitas yang sangat luas dan suportif. Dengan begitu, Anda tidak akan kesulitan dalam belajar *JavaScript*. Jika sewaktu-waktu Anda menghadapi kendala atau error, Anda bisa langsung bertanya ke komunitas. Mereka dengan senang hati akan menjadi solusi bagi Anda agar error tersebut dapat diperbaiki dengan segera. Nah, berikut ini beberapa komunitas terkenal untuk belajar *JavaScript*:

## 4. Ekosistem yang Besar

*JavaScript* adalah bahasa pemrograman dengan ekosistem yang besar. Mengingat perkembangan teknologinya yang pesat, ia memiliki banyak framework *JavaScript*, *library*, dan *plugin* yang bisa Anda gunakan sesuai kebutuhan. Dengan ekosistem besar tersebut, *JavaScript* kini tidak hanya digunakan untuk membangun *website* dari *client side*. *JavaScript* juga bisa dimanfaatkan untuk pengembangan *back end*, *mobile app*, dan sebagainya.

- **Client-side Web Development**

- Native Javascript / *Vanilla Javascript*
- JQuery
- AngularJS, React, Ember, Backbone
- dll,

- **Server-side Development**

- NodeJS
- ExpressJS

- **Browser Extension / Add-on**

- **Desktop Applications**

- Electron, AppJS

- **Mobile App Development**

- JQuery Mobile, Cordova / PhoneGap

- **IoT & Robotics**

- CyclonJS, Johnny-Five

- **JSON**

Gambar 2.5 Belajar *JavaScript*

Wah, ternyata banyak juga keuntungan belajar *JavaScript*. Berikutnya mari mempersiapkan hal-hal yang dibutuhkan untuk coding *JavaScript*.

### A. Apa Saja yang Dibutuhkan untuk Belajar *JavaScript*?

Untuk bisa belajar *JavaScript*, Anda cuma membutuhkan dua tools, yaitu:

#### 1. *Web & Browser*

*Web & browser* berfungsi untuk menerjemahkan (*interpreter*) bahasa *JavaScript* ke bahasa komputer. Beberapa *web, browser* yang bisa Anda gunakan antara lain *Google Chrome, Mozilla Firefox, Microsoft Edge*, dan lain-lain.

#### 2. *Code Editor*

**Code editor** berfungsi untuk menuliskan sintaks atau kode program agar menjadi sebuah program *JavaScript* utuh. Beberapa code editor yang bisa Anda coba diantaranya *Visual Studio Code, Sublime Text*, dan *Notepad++*. Setelah mempersiapkan keduanya, pastikan Anda memahami dulu beberapa panduan dalam bahasa pemrograman *JavaScript*.

## D. Hal-hal yang Perlu Dipahami Sebelum Menggunakan *JavaScript*

Sebelum Anda lanjut belajar coding *JavaScript*, ada beberapa hal yang wajib Anda pahami. Mulai dari aturan penulisannya, *variabel*, tipe data, sampai operator *JavaScript*. Mari simak penjelasan selengkapnya!

## E. Aturan Penulisan Kode *JavaScript*

Bahasa pemrograman *JavaScript* punya beberapa aturan penulisan kode yang harus Anda patuhi, diantaranya:

### 1. *Whitespace*

***Whitespace*** adalah area kosong berisi karakter tidak terlihat pada *JavaScript*, seperti spasi, *tab*, atau *enter*. *Whitespace* berguna untuk menata susunan kode agar terlihat lebih rapi.

Penggunaan *Whitespace* tidak akan menambah ukuran *file JavaScript* secara signifikan, sehingga kinerja *website* tetap optimal. Ditambah lagi, *Whitespace* akan diabaikan ketika *browser* menjalankan *JavaScript*.

Berikut adalah contoh penggunaan *Whitespace* saat coding *JavaScript*:

### 2. *Line Length*

*Line Length* atau panjang baris adalah jumlah karakter *JavaScript* yang ada dalam satu baris kode. Idealnya, satu baris kode terdiri kurang dari 80 karakter. Hal ini bertujuan untuk meningkatkan kenyamanan saat coding *JavaScript*. Jika sebuah perintah *JavaScript* dalam satu baris terlalu panjang, potong perintah tersebut menjadi dua baris. Usahakan untuk memotong perintah setelah operator (+, -, =) atau koma (,) seperti contoh berikut:



### 3. Comments

Pada *JavaScript*, komentar berguna untuk mencatat maksud dari sebuah atau beberapa baris kode. Untuk itu, pastikan menulis komentar dengan bahasa yang ringkas dan mudah dimengerti. Ada dua cara menandai komentar di *JavaScript*. Yang pertama yaitu komentar satu baris yang diawali dengan tanda //

Cara yang kedua adalah komentar beberapa baris dengan tanda /\* pada awal dan akhir komentar seperti ini:

### 4. Case Sensitive

Bahasa pemrograman *JavaScript* bersifat *Case Sensitive*, artinya ia membedakan penulisan huruf besar dan kecil. Contohnya variabel bernama **'helloworld'** akan dianggap berbeda dengan perintah **'HelloWorld'**. Untuk itu, usahakan selalu konsisten dalam menulis kode. Sebab, kesalahan penulisan sekecil apapun dapat menyebabkan error pada website Anda. Jika perlu, hanya gunakan huruf kecil saat coding *JavaScript*.

### 5. Statement Rules

*Statement* atau pernyataan adalah instruksi atau alur kerja sebuah program. Pada *JavaScript*, terdapat dua aturan dalam penulisan statement. Yang pertama adalah simple statement yang diakhiri dengan titik koma (;) seperti berikut: Sementara yang kedua adalah complex statement yang bisa diakhiri tanpa titik koma.

### 6. Reserved Words

*Reserved Words* adalah kata-kata khusus yang tidak boleh digunakan sebagai nama variabel atau fungsi. Pasalnya, kata-kata tersebut sudah dipakai untuk menjalankan perintah bawaan pada bahasa pemrograman *JavaScript*.

Berikut adalah daftar *Reserved Words JavaScript* yang wajib Anda ketahui:

Reserved Words in JavaScript			
abstract	else	instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

Gambar 2.6 *Reserved Words in JavaScript*

Berikut aturan penulisan untuk beberapa *Reserved Words* di atas:

- **While:**
- **For:**
- **Switch Case:**

Selain memahami aturan penulisan kode, Anda juga harus paham tentang variabel. Simak penjelasannya pada poin berikut ini.

#### **F. Variabel JavaScript:**

Variabel adalah sebuah wadah yang digunakan untuk menampung nilai atau data pada bahasa pemrograman. Variabel yang bisa diberi nama ini bisa menampung angka, tulisan, *boolean*, dan lain-lain.

Ada tiga tahap dalam menggunakan variabel ketika belajar *JavaScript*, yaitu:

1. **Declaration**- Tahap mendaftarkan variabel ke dalam program yang ditulis.
2. **Initialization**- Tahap menyiapkan memori yang nantinya digunakan oleh variabel.
3. **Assignment** – Tahap menetapkan nilai spesifik untuk disimpan ke dalam variabel.

Nah, tiga *Reserved Words* yang dapat Anda gunakan untuk mendeklarasikan variabel yaitu **var**, **let**, dan **const**. TAPI untuk pemula, direkomendasikan untuk menggunakan **var** saja karena punya aturan yang lebih simpel. Berikut beberapa aturan pembuatan variabel yang harus Anda perhatikan:

1. **Hindari Penggunaan Spasi** – Jika nama variabel terdiri dari dua kata, jangan gunakan spasi sebagai pemisah. Sebaiknya gunakan *underscore*.
2. **Hindari Awalan Angka** – Angka tidak boleh digunakan sebagai awalan nama variabel, tapi bisa digunakan untuk mengakhiri nama variabel.
3. **Gunakan camelCase** – *camelCase* adalah standar penulisan variabel yang direkomendasikan. Format dasarnya adalah seluruh kata pertama memakai huruf kecil dan huruf pertama kata kedua memakai kapital.
4. **Gunakan Shorthand** – *Shorthand* adalah metode yang digunakan untuk menyingkat penulisan variabel. Gunanya agar kode program yang dihasilkan Dengan adanya *Shorthand*, Anda tidak perlu menuliskan **var** pada **'age'** dan **'pass'**, meskipun mereka berdua sebenarnya adalah variabel.

Setelah belajar apa itu Variabel, berikutnya mari mempelajari apa itu tipe data *JavaScript*.

## G. Tipe Data *JavaScript*

Seperti yang sudah sedikit dijelaskan di atas, nilai atau value disimpan pada variabel. Nah, nilai tersebut bisa berisi beberapa tipe data berikut:

### 1. *Number*

Tidak seperti bahasa pemrograman lain, *JavaScript* tidak memakai tipe data angka yang berbeda-beda. Misalnya, *integer*, *float*, dan *long*. Ia hanya punya satu tipe angka, yaitu *number* yang sanggup menyimpan data hingga 64 *bit*. Demikian, *JavaScript* tetap bisa menggunakan berbagai jenis bilangan, seperti:

- **Bilangan Bulat** – Misalnya 10, 1500, dan 12345.
- **Bilangan Pecahan** – Contohnya 3.14, 0.5, dan 100.00.
- **Bilangan Eksponensial** – Misalnya 123e5 atau 123e-5.
- **Bilangan Negatif** – Contohnya -0.25 atau -123.
- **Bilangan Spesial** – Misalnya *Infinity* (10/0), *-Infinity* (-10/0), dan *NaN* (0/0).

### 2. *String*

Pada *JavaScript*, *string* adalah tipe data yang digunakan untuk menuliskan data tekstual (*plain text*). Untuk membuat *string*, Anda perlu menambahkan tanda petik pada teks tersebut. Bisa petik dua ("" ) atau petik satu ("). Sebenarnya tidak ada aturan baku yang menjelaskan kapan harus menggunakan petik dua atau petik satu. Meski begitu, disarankan untuk memakai petik dua untuk penulisan *string*. Lalu, gunakan petik satu untuk penekanan di teks tertentu.

### 3. *Boolean*

*Boolean* adalah sebuah tipe data *JavaScript* yang digunakan untuk memberikan nilai logika **true** atau **false**. Nah, *Boolean* berguna untuk menentukan aksi berbeda dan mengatur alur kerja program.

#### 4. Function

*Function* adalah tipe data dalam bentuk perintah yang dapat digunakan di bagian program yang memerlukan. Untuk menggunakannya, Anda perlu membuat *Function* terlebih dahulu, baru kemudian memanggilnya. Ketika belajar *JavaScript*, ada dua jenis *Function* yang perlu Anda ketahui, yaitu:

- **Built-in Function** – *Function* bawaan milik *JavaScript* yang siap digunakan. Contohnya ***alert()***, ***confirm()***, dan ***prompt()***.
- **User-defined Function** – *Function* yang Anda buat sendiri. Formatnya adalah ***function namaFunction()***.

#### 5. Array

*Array* adalah tipe data *JavaScript* yang digunakan untuk menampung lebih dari satu nilai dan memiliki *index*. Dengan *Array*, Anda bisa memasukkan tipe data yang berbeda dalam satu *Array*. Selain itu, ia berguna untuk menghemat penggunaan variabel ketika coding *JavaScript*. Anda bisa memakai *Array* dengan membuat kurung siku. Lalu, menulis nilai *Array* di dalamnya dan dipisahkan dengan koma.

#### 6. Object

Mirip dengan *Array*, *Object* adalah tipe data yang memiliki lebih dari satu nilai. Bedanya, *Object* tidak punya *index* tapi nama. Kenapa Anda harus menggunakan *Object* saat belajar *JavaScript*? Ternyata, *Object* dapat mengorganisir data lebih baik dari *Array*. Sebab, Anda bisa memasukkan *Function* dan *Array* pada *Object*. Hasilnya, Anda bisa menghemat baris kode yang dihasilkan saat coding *JavaScript*.

Selain menguasai tipe data *JavaScript*, pastikan Anda juga belajar tentang operator *JavaScript*. Karena, kedua hal ini saling berkaitan.

## H. Operator *JavaScript*

Pada *JavaScript*, operator adalah sebuah simbol yang digunakan untuk melakukan operasi. Baik itu perhitungan, perbandingan, atau logika. Nah, jenis-jenis operator *JavaScript* adalah sebagai berikut:

### 1. Aritmatika

Operator aritmatika adalah operator yang digunakan untuk melakukan perhitungan matematika. Misalnya, penambahan, pengurangan, dan perkalian.

### 2. Penugasan

Ketika belajar *JavaScript*, operator penugasan (*assignment*) adalah operator yang digunakan untuk mendeklarasikan nilai dari suatu variabel

### 3. Perbandingan

Operator perbandingan (*comparison*) adalah operator pada *JavaScript* yang dipakai untuk membandingkan nilai dari dua variabel. Nah, operator perbandingan ini nantinya akan menghasilkan Boolean (**true** atau **false**).

### 4. Logika

Operator logika adalah operator yang digunakan untuk menentukan logika dari beberapa value yang digabungkan. Mirip dengan operator perbandingan, operator yang satu ini juga menghasilkan *Boolean*.

### 5. String

Sesuai namanya, operator *string* adalah operator yang digunakan untuk menggabungkan nilai dari tipe data *string*. Caranya dengan menggunakan tanda + seperti pada operator aritmatika.

## 6.Typeof

Operator *typeof* adalah operator yang dipakai untuk mengetahui tipe data dari suatu nilai.

## 7.Kondisional

Operator kondisional adalah operator yang digunakan untuk melakukan pengecekan pada suatu kondisi. Ia juga dapat menentukan nilai ketika kondisi tersebut bernilai **true** dan **false**.

Nah, itu tadi beberapa operator yang akan Anda temukan ketika belajar *JavaScript*. Berikutnya, mari memahami beberapa cara menulis kode *JavaScript*.

### I. Cara Menulis Kode *JavaScript*

Ada dua cara yang bisa Anda coba untuk membuat kode *JavaScript*, yaitu:

#### 1.Langsung di File *HTML* (*Internal JavaScript*)

Sesuai namanya, Anda bisa langsung memasukkan kode *JavaScript* pada file berformat *HTML*. Caranya dengan menuliskan kode di antara tag **<Script>** **</Script>**, lalu tempatkan skrip tersebut pada tag **<body>**.

#### 2.Terpisah dari File *HTML* (*External JavaScript*)

Cara kedua adalah dengan membuat file *JavaScript* terpisah, lalu menghubungkannya dengan file *HTML*. *External JavaScript* lebih direkomendasikan, karena:

- **Reusability** – Memungkinkan *JavaScript* digunakan di banyak file *HTML*.
- **Readability** – Membuat kode program jadi lebih mudah dibaca.
- **Conciseness** – Meringkas jumlah baris kode yang dihasilkan.

## Cara Membuat Program Sederhana dengan *JavaScript*

Seperti yang sudah dijelaskan, Anda membutuhkan *web*, *browser* dan code editor untuk mempraktikkan tutorial *JavaScript*. Nah, di sini kami menggunakan *browser* Google Chrome dan editor Visual Studio Code. Selain itu, Anda perlu menyiapkan sebuah folder pada komputer sebagai tempat menyimpan file *HTML* dan *JavaScript*. Di sini kami membuat folder dengan nama **helloworld**. Setelah menyiapkan beberapa hal di atas, mari ikuti tutorial *JavaScript* membuat program sederhana yang menampilkan pop up ketika menekan sebuah tombol.

Untuk contohnya bisa diakses pada link *GitHub* dibawah ini:  
<https://GitHub.com/WASENDERAUTO/MATERI-BUKPED.git>



# BAB V

## JSON FILE

### A. Apa itu JSON?

**JSON** adalah format file yang digunakan untuk menyimpan informasi secara terorganisir dan mudah diakses. Bentuk lengkapnya adalah Notasi Objek *JavaScript*. Ini menawarkan kumpulan data yang dapat dibaca manusia yang dapat diakses secara logis. Ekstensi nama file untuk kode pemrograman tertulis adalah *json*. Jenis media Internet untuk *JSON* adalah *application/json*, dan *Uniform Type Identifier*-nya adalah *public.json*.

Dalam tutorial *JSON* untuk pemula ini, Anda akan mempelajari dasar-dasar *JSON* seperti:

- ***Apa itu JSON?***
- ***Mengapa menggunakan JSON?***
- ***Sejarah JSON***
- ***Fitur JSON***
- ***Tipe Data di JSON***
- ***Contoh JSON***
- ***Aplikasi JSON***
- ***JSON vs XML***
- ***Apa itu JSON bukan?***
- ***Kekurangan JSON***
- ***Alat JSON Populer (Pengaya)***

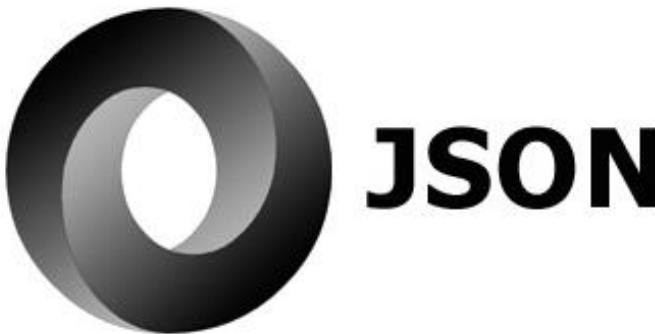
### B. Mengapa menggunakan JSON?

Berikut adalah manfaat/kelebihan penting menggunakan *JSON*:

- Berikan dukungan untuk semua *browser*
- Mudah dibaca dan ditulis
- Sintaks langsung

- Anda dapat mengurai secara native dalam *JavaScript* menggunakan fungsi `eval()`
- Mudah dibuat dan dimanipulasi
- Didukung oleh semua framework *JavaScript* utama
- Didukung oleh sebagian besar teknologi *backend*
- *JSON* dikenali secara native oleh *JavaScript*
- Ini memungkinkan Anda untuk mengirim dan membuat serialisasi data terstruktur menggunakan koneksi jaringan.
- Anda dapat menggunakannya dengan bahasa pemrograman modern.
- *JSON* adalah teks yang dapat dikonversi ke objek apa pun dari *JavaScript* menjadi *JSON* dan mengirimkan *JSON* ini ke *server*.

### C. Sejarah *JSON*



Gambar 4.1 Hasil *Script*

Berikut adalah landmark penting yang membentuk sejarah *JSON*:

- *Douglas Crockford* menentukan format *JSON* di awal tahun 2000-an.
- Situs *web*, resmi diluncurkan pada tahun 2002.
- Pada Desember 2005, Yahoo! mulai menawarkan beberapa layanan *web*,nya di *JSON*.
- *JSON* menjadi standar internasional *ECMA* pada tahun 2013.
- Standar format *JSON* terbaru diterbitkan pada tahun 2017.

#### D. Fitur *JSON*

- **Mudah digunakan** – *JSON API* menawarkan fasad tingkat tinggi, yang membantu Anda menyederhanakan kasus penggunaan yang umum digunakan.
- **Performa** – *JSON* cukup cepat karena menghabiskan ruang memori yang sangat sedikit, yang sangat cocok untuk grafik atau sistem objek besar.
- **Alat gratis** – Pustaka *JSON* adalah sumber terbuka dan gratis untuk digunakan.
- **Tidak perlu membuat pemetaan** – *Jackson API* menyediakan pemetaan default untuk banyak objek yang akan diserialisasi.
- **Clean *JSON*** – Membuat hasil *JSON* yang bersih dan kompatibel yang mudah dibaca.
- **Ketergantungan** – Pustaka *JSON* tidak memerlukan pustaka lain untuk diproses.

#### E. Aturan untuk Sintaks *JSON*

Aturan untuk sintaks *JSON* adalah:

- Data harus dalam pasangan nama/nilai
- Data harus dipisahkan dengan koma
- Kurung kurawal harus menahan objek
- Kurung perseg menahan *Array*

#### F. Tipe Data di *JSON*

Tipe data penting yang digunakan dalam *JSON* adalah:

Tipe data	Keterangan
Nomor	Ini termasuk bilangan real, bilangan bulat atau bilangan mengambang
Rangkaian	Ini terdiri dari teks apa pun atau Unicode yang dikutip ganda dengan escapement backslash
Boolean	Tipe data Boolean mewakili nilai Benar atau Salah
Batal	Nilai Null menunjukkan bahwa variabel terkait tidak memiliki nilai apapun

Obyek	Ini adalah kumpulan pasangan nilai kunci dan selalu dipisahkan oleh koma dan diapit oleh tanda kurung kurawal.
Himpunan	Ini adalah urutan urutan nilai yang dipisahkan.

### Nomor:

- Angka tersebut adalah format floating-point presisi ganda yang bergantung pada metode penerapannya.
- Di JSON Anda tidak dapat menggunakan format Heksadesimal

### G. Rangkaian:

Ini adalah serangkaian karakter Unicode yang dikutip ganda dan memiliki garis miring terbalik.

Tabel berikut menunjukkan berbagai jenis *string*:

Jenis	Keterangan
*	Gunakan untuk mengetik kutipan ganda
/	Gunakan untuk solidus
\	Gunakan untuk solidus terbalik
B	Gunakan untuk menambahkan backspace
F	Dari pakan
N	Untuk membuat baris baru
R	Gunakan untuk carriage return
T	Untuk menampilkan tab <i>horizontal</i>
AS	Digit heksadesimal

## H.Himpunan

- Ini adalah kumpulan nilai yang dipesan.
- Anda harus menggunakan larik jika nama kuncinya adalah bilangan bulat berurutan.
- Itu harus diapit di dalam tanda kurung siku yang harus dipisahkan dengan ',' (koma)

## I.Objek *JSON*

Objek ***JSON*** adalah entitas dalam *JSON* yang dilampirkan dalam tanda kurung kurawal. Itu ditulis dalam kumpulan pasangan nama dan nilai yang tidak diurutkan di mana nama harus diikuti dengan ":" (titik dua), dan pasangan nama/nilai harus dipisahkan menggunakan "," (koma). Ini dapat digunakan ketika nama kunci adalah *string* arbitrer.

## J.Aplikasi *JSON*

Berikut adalah beberapa aplikasi *JSON* yang umum:

- Membantu Anda mentransfer data dari *server*
- Contoh format file *JSON* membantu dalam mengirimkan dan membuat cerita bersambung semua jenis data terstruktur.
- Memungkinkan Anda melakukan panggilan data asinkron tanpa perlu melakukan penyegaran halaman
- Membantu Anda mengirimkan data antara *server* dan aplikasi *web*,.
- Ini banyak digunakan untuk aplikasi berbasis *JavaScript*, yang mencakup ekstensi *browser* dan situs *web*,.
- Anda dapat mengirimkan data antara *server* dan aplikasi *web*, menggunakan *JSON*.
- Kita dapat menggunakan *JSON* dengan bahasa pemrograman modern.
- Ini digunakan untuk menulis aplikasi berbasis *JavaScript* yang menyertakan add-on *browser*.
- Layanan *web*, dan Restful *API* menggunakan format *JSON* untuk mendapatkan data publik.

## K.JSON vs XML

Tabel 1.1

<i>JSON</i>	<i>XML</i>
Objek <i>JSON</i> memiliki tipe	Data XML tidak bertipe
Jenis <i>JSON</i> : <i>string</i> , angka, larik, Boolean	Semua data XML harus berupa <i>string</i>
Data mudah diakses sebagai objek <i>JSON</i>	Data XML perlu diuraikan.
File <i>JSON</i> lebih mudah dibaca manusia.	File XML kurang dapat dibaca oleh manusia.
<i>JSON</i> didukung oleh sebagian besar <i>browser</i> .	Parsing XML lintas- <i>browser</i> bisa rumit
<i>JSON</i> tidak memiliki kemampuan tampilan.	XML menyediakan kemampuan untuk menampilkan data karena merupakan bahasa markup.
Mengambil nilai itu mudah	Mengambil nilai itu sulit
Didukung oleh banyak toolkit Ajax	Tidak sepenuhnya didukung oleh toolkit Ajax
Cara deserializing/serialisasi <i>JavaScript</i> yang sepenuhnya otomatis.	Pengembang harus menulis kode <i>JavaScript</i> untuk membuat serial/menghapus serial dari XML
Dukungan asli untuk objek.	Objek harus diekspresikan oleh konvensi – sebagian besar penggunaan atribut dan elemen yang terlewatkan.

Ini adalah perbedaan utama antara *JSON* vs. XML

### L.Kekurangan *JSON*

Berikut adalah beberapa keunggulan *JSON*:

- Tidak ada dukungan namespace, karenanya ekstensibilitas buruk
- Dukungan [alat pengembangan](#) terbatas
- Tidak ada dukungan untuk definisi tata bahasa formal

## 10. Kelebihan *JSON*:

Berikut adalah beberapa keunggulan *JSON*:

- Mudah Dibaca dan Ringan
- Independen Bahasa
- Mudah Diparsing dan Dihasilkan

### A. Alat *JSON* Populer (Pengaya)

Berikut adalah alat *JSON* yang penting:

#### 1. *Lint JSON*:

*JSONLint* adalah proyek sumber terbuka yang digunakan sebagai validator dan pemformat ulang untuk *JSON*. Ini adalah format pertukaran data yang ringan. Salin dan tempel, ketik langsung, atau masukkan *URL* di alat validator *JSON* untuk memvalidasi kode *JSON* Anda. **Tautan:** <https://jsonlint.com>

#### 2. Editor *JSON Daring*:

*JSON Editor Online* adalah alat berbasis *web*, yang berguna. Ini memungkinkan Anda untuk mengedit, melihat, dan memformat *JSON*. Ini menampilkan data Anda berdampingan dalam [perangkat lunak editor kode](#) yang jelas dan dapat diedit. **Tautan:** <http://jsoneditoronline.org/>

#### Alat Pengurang *JSON*:

Ini adalah alat yang membantu Anda menghapus spasi putih dan memberikan kode *JSON* yang membutuhkan ruang paling sedikit. **Tautan:** <https://www.browserling.com/tools/json-minify>

#### Konverter *JSON* ke XML:

Konverter *JSON* ke XML adalah alat sederhana dan efektif yang membantu Anda mengonversi kode *JSON*. **Tautan:** <https://codebeautify.org/jsontoxml>

### **Pemformat JSON:**

Pemformat *JSON* membantu Anda memecahkan masalah dengan memformat data *JSON* sehingga mudah dibaca dan di-debug oleh manusia. **Tautan:** <https://jsonformatter.curiousconcept.com/>

### **B. Ringkasan:**

- Metode *JSON* digunakan untuk menyimpan informasi secara terorganisir, dan mudah diakses.
- *JSON* adalah singkatan dari Notasi Objek *JavaScript*.
- *JSON* Memberikan dukungan untuk semua *browser* yang ditawarkan oleh banyak bahasa.
- Douglas Crockford menentukan format *JSON* di awal tahun 2000-an
- *JSON API* menawarkan fasad tingkat tinggi, yang membantu Anda menyederhanakan kasus penggunaan yang umum digunakan
- Aturan penting untuk menulis sistem *JSON* adalah bahwa data harus ditulis dalam pasangan nama/nilai.
- Number, *string*, Boolean, Null, Object, dan *Array* adalah tipe Data penting yang digunakan dalam *JSON*.
- Ini membantu Anda mentransfer data dari *server*.
- Objek *JSON* memiliki tipe sedangkan data XML tidak bertipe
- *JSON* bukan format dokumen
- Tidak ada dukungan namespace, karenanya ekstensibilitas buruk
- *JSONLint* adalah proyek sumber terbuka yang digunakan sebagai validator dan pemformat ulang untuk *JSON*.



# BAB VI

## ***GCP & GCF***

### **1. Apa itu GCP?**

*GCP biasanya merujuk ke Google cloud Platform, yang merupakan salah satu penyedia layanan cloud computing. Google cloud Platform menyediakan berbagai layanan termasuk komputasi awan, penyimpanan data, pembelajaran mesin, analisis data, dan banyak lagi. Pengguna dapat menggunakan GCP untuk meng-host aplikasi, menyimpan dan mengelola data, serta menggunakan berbagai layanan cerdas untuk pengembangan dan operasional aplikasi. Google cloud Platform bersaing dengan penyedia layanan cloud lainnya seperti Amazon WebServices (AWS) dan Microsoft Azure.*



Gambar 5.1 *Google cloud Function*

#### **A. Mengapa menggunakan GCP?**

Ada beberapa alasan mengapa seseorang atau organisasi mungkin memilih untuk menggunakan *Google cloud Platform (GCP)*:

- 1. Infrastruktur Global:** *GCP* memiliki pusat data dan infrastruktur *cloud* yang tersebar di seluruh dunia. Ini memungkinkan pengguna untuk

menempatkan sumber daya mereka di lokasi yang paling dekat dengan pengguna akhir mereka, meningkatkan kinerja dan responsivitas aplikasi.

2. **Skalabilitas dan Elastisitas:** *GCP* memungkinkan pengguna untuk dengan mudah menyesuaikan kapasitas sumber daya sesuai kebutuhan proyek. Ini memungkinkan skalabilitas *vertikal* dan *horizontal* untuk menangani lonjakan beban atau pertumbuhan aplikasi.
3. **Berbagai Layanan:** *GCP* menyediakan berbagai layanan cloud, termasuk komputasi, penyimpanan data, basis data, kecerdasan buatan (*AI*), *machine learning (ML)*, analisis data, dan banyak lagi. Ini memberikan fleksibilitas dan pilihan untuk membangun dan mengelola berbagai jenis aplikasi.
4. **Kecerdasan Buatan dan *Machine learning*:** *GCP* menawarkan berbagai layanan *AI* dan *ML* yang memungkinkan pengguna untuk mengembangkan dan menerapkan model *machine learning* dengan mudah. Ini mencakup *Vision AI*, *Speech-to-Text*, *Natural Language Processing*, dan layanan lainnya.
5. **Keamanan dan Kepatuhan:** *Google cloud Platform* menempatkan penekanan besar pada keamanan dan kepatuhan. Ini mencakup kontrol akses yang ketat, enkripsi data, pemantauan keamanan, dan kepatuhan dengan standar industri tertentu.
6. **Kolaborasi dan Integrasi:** *GCP* terintegrasi dengan berbagai alat pengembangan dan kolaborasi, seperti *Kubernetes*, *Jenkins*, dan *GitHub*. Ini memudahkan pengguna untuk mengelola siklus hidup aplikasi dan berkolaborasi dalam pengembangan perangkat lunak.
7. **Biaya yang Efisien:** Model penetapan harga *GCP* yang fleksibel memungkinkan pengguna membayar hanya untuk sumber daya yang mereka gunakan. Ini dapat memberikan biaya operasional yang lebih efisien dan dapat diukur.

8. **Dukungan dan Komunitas:** *GCP* menyediakan layanan dukungan teknis dan memiliki komunitas pengguna yang besar. Ini memudahkan pengguna untuk mendapatkan bantuan dan sumber daya saat mengelola dan mengoptimalkan infrastruktur *cloud* mereka.

Penting untuk dicatat bahwa keputusan untuk menggunakan *GCP* atau penyedia layanan *cloud* lainnya akan bergantung pada kebutuhan dan preferensi spesifik proyek atau organisasi. Setiap penyedia layanan *cloud* memiliki kelebihan dan kekurangan masing-masing.

## B. Kekurangan & Kelebihan *GCP*

- **Kelebihan *Google cloud Platform (GCP)***

1. **Infrastruktur Global:** *GCP* memiliki infrastruktur global yang tersebar di berbagai lokasi di seluruh dunia. Ini memungkinkan pengguna untuk menempatkan sumber daya mereka di lokasi yang paling dekat dengan pengguna akhir, meningkatkan kinerja aplikasi.

2. **Skalabilitas dan Elastisitas:** *GCP* memungkinkan pengguna untuk dengan mudah menyesuaikan kapasitas sumber daya sesuai kebutuhan proyek. Ini memberikan skalabilitas *vertikal* dan *horizontal* untuk menangani perubahan beban kerja.

3. **Beragam Layanan:** *GCP* menyediakan berbagai layanan *cloud*, mencakup komputasi, penyimpanan data, basis data, kecerdasan buatan, *machine learning*, analisis data, dan lainnya. Ini memberikan fleksibilitas untuk membangun dan mengelola berbagai jenis aplikasi.

4. **Kecerdasan Buatan dan *Machine learning*:** *GCP* memiliki layanan kecerdasan buatan dan *machine learning* yang kuat, seperti *Vision AI*, *Speech-to-Text*, *Natural Language Processing*, dan *BigQuery ML*, yang memudahkan pengguna untuk menerapkan solusi *AI* dan *ML*.

**5. Kepatuhan dan Keamanan:** *GCP* menempatkan penekanan besar pada kepatuhan dan keamanan. Ini mencakup kontrol akses yang ketat, enkripsi data, pemantauan keamanan, dan kepatuhan dengan berbagai standar industri.

**6. Harga yang Efisien:** Model penetapan harga *GCP* yang fleksibel memungkinkan pengguna membayar hanya untuk sumber daya yang mereka gunakan. Ini dapat memberikan biaya operasional yang lebih efisien dan dapat diukur.

**7. Peralatan Pengembangan dan Integrasi:** *GCP* terintegrasi dengan berbagai alat pengembangan dan kolaborasi, seperti Kubernetes, Jenkins, dan *GitHub*. Ini memudahkan pengguna untuk mengelola siklus hidup aplikasi dan berkolaborasi dalam pengembangan perangkat lunak.

**Kekurangan Google cloud Platform (GCP):**

**1. Kurangnya Keterlibatan di Pasar:** Meskipun *GCP* memiliki pangsa pasar yang berkembang, *AWS* dan *Azure* masih mendominasi pasar *cloud* computing. Keterlibatan *GCP* di pasar mungkin belum sebesar pesaing utamanya.

**2. Keterbatasan Pilihan Produk:** Beberapa fitur dan produk mungkin tidak sebanyak yang ditawarkan oleh *AWS* atau *Azure*. Pengguna perlu memastikan bahwa *GCP* menyediakan layanan yang mereka butuhkan untuk proyek mereka.

**3. Kemungkinan Pembelajaran Kurva Tinggi:** Bagi pengguna yang baru mengenal *cloud* computing atau *GCP*, mungkin ada kurva pembelajaran yang tinggi. Ini dapat memerlukan waktu bagi tim untuk menguasai semua aspek platform.

**4. Ketergantungan pada Koneksi Internet:** Beberapa layanan *GCP* memerlukan koneksi internet yang stabil. Ketergantungan ini dapat menjadi kendala di lingkungan dengan konektivitas internet yang tidak stabil.

**5. Keterbatasan di Beberapa Wilayah:** Meskipun *GCP* memiliki infrastruktur global, beberapa layanan mungkin memiliki keterbatasan di wilayah tertentu. Pengguna perlu memastikan bahwa layanan yang mereka butuhkan tersedia di lokasi yang diinginkan.

Penting untuk mengevaluasi kelebihan dan kekurangan *GCP* dengan mempertimbangkan kebutuhan dan tujuan spesifik proyek atau organisasi. Setiap penyedia *cloud* memiliki karakteristik uniknya sendiri.

### **C. Hal apa saja yang dibutuhkan dalam *GCP***

1. **Akun google Cloud:** Untuk mengakses layanan *GCP*, Anda perlu membuat akun Google Cloud. Dengan akun ini, Anda dapat mengelola proyek, menggunakan layanan, dan mengelola sumber daya *cloud* Anda.
2. **Proyek *GCP*:** Setelah membuat akun, Anda perlu membuat proyek *GCP*. Proyek ini berfungsi sebagai wadah untuk sumber daya, data, dan aplikasi Anda di Google Cloud.
3. **Billing dan Pembayaran:** Untuk menggunakan sumber daya *GCP*, Anda perlu mengonfigurasi informasi penagihan dan pembayaran. Ini melibatkan menambahkan metode pembayaran dan memahami struktur harga layanan *GCP*.
4. **Google *cloud SDK*:** Google *cloud SDK* adalah seperangkat perintah baris dan alat manajemen untuk berinteraksi dengan *GCP*. Dengan SDK ini, Anda dapat mengakses dan mengelola sumber daya *cloud* dari terminal atau skrip.
5. **Akses dan Identitas:** Anda perlu mengelola akses dan identitas pengguna atau entitas layanan yang akan berinteraksi dengan sumber daya *GCP*. Ini melibatkan konfigurasi akun layanan, manajemen peran, dan penanganan otentikasi.

6. **Layanan *cloud* yang Dibutuhkan:** Tentukan layanan *cloud GCP* yang sesuai dengan kebutuhan proyek Anda. *GCP* menyediakan berbagai layanan, termasuk komputasi, penyimpanan, *database*, kecerdasan buatan, *machine learning*, dan banyak lagi.
7. **Penyimpanan dan Pengelolaan Data:** Identifikasi bagaimana data Anda akan disimpan dan dikelola. *GCP* menyediakan berbagai layanan penyimpanan, termasuk *cloud Storage* untuk penyimpanan objek, *cloud SQL* untuk basis data relasional, dan *BigQuery* untuk analisis data.
8. **Kunci *API* dan Otentikasi:** Jika Anda menggunakan *API GCP* atau mengakses sumber daya melalui *API*, Anda perlu mengelola kunci *API* dan otentikasi. Ini melibatkan pembuatan dan manajemen kunci *API* serta konfigurasi otentikasi.
9. **Keamanan dan Kepatuhan:** Terapkan praktik keamanan dan kepatuhan yang sesuai dengan kebutuhan proyek Anda. Konfigurasikan pengaturan keamanan seperti *firewall*, enkripsi data, dan manajemen kunci.
10. **Monitoring dan Pemantauan:** Konfigurasikan alat pemantauan dan *logging GCP* untuk memantau kinerja, mendeteksi masalah, dan memperoleh wawasan tentang penggunaan sumber daya.
11. **Dokumentasi dan Pelatihan:** Pastikan tim Anda memahami dokumentasi *GCP* dan mendapatkan pelatihan yang diperlukan. *GCP* menyediakan sumber daya belajar *Online* dan dokumentasi yang kaya.

## 2. Apa itu GCF?

*GCF (Google cloud Functions)* adalah layanan di *Google cloud Platform (GCP)* yang memungkinkan Anda menulis, *mendeploy*, dan menjalankan fungsi *server* tanpa harus mengelola infrastruktur *server*. Ini adalah bagian dari komputasi tanpa *server* di mana Anda hanya membayar untuk waktu eksekusi fungsi yang sebenarnya. Fungsi ini dapat diaktifkan oleh berbagai peristiwa, seperti pemanggilan *HTTP*, perubahan pada penyimpanan *cloud Storage*, atau peristiwa lain di ekosistem *GCP*.

### A. Alasan menggunakan google *cloud function*

Berikut adalah beberapa keuntungan utama menggunakan *Google cloud Functions*:

1. **Tanpa Server (Serverless):** *GCF* memungkinkan pengembang untuk menulis dan menjalankan fungsi tanpa harus mengelola infrastruktur *server*. Ini menghilangkan kebutuhan untuk merencanakan, mengelola, atau memantau *server*, sehingga fokus dapat lebih ditekankan pada pengembangan kode.
2. **Aktivasi Berbasis Peristiwa:** Fungsi-fungsi di *GCF* diaktifkan oleh peristiwa tertentu, seperti pemanggilan *HTTP*, perubahan pada penyimpanan *cloud Storage*, atau peristiwa lainnya di ekosistem *Google cloud Platform*. Ini memungkinkan fungsi-fungsi merespons secara otomatis terhadap perubahan di lingkungan *cloud*.
3. **Skalabilitas Otomatis:** *GCF* menyediakan skalabilitas otomatis untuk fungsi-fungsi. Jumlah instance yang dijalankan dapat disesuaikan secara dinamis berdasarkan beban kerja yang diterima, sehingga memastikan kinerja optimal.
4. **Biaya Berbasis Penggunaan Aktual:** Dengan model tanpa *server*, Anda hanya membayar untuk sumber daya yang digunakan selama eksekusi fungsi. Ini dapat mengurangi biaya operasional karena tidak ada biaya tetap untuk infrastruktur yang tidak digunakan.

5. **Integrasi dengan Layanan GCP Lainnya:** GCF dapat berintegrasi dengan berbagai layanan *Google cloud Platform (GCP)*, seperti *cloud Storage*, *cloud Pub/Sub*, *cloud Firestore*, dan lainnya. Hal ini memungkinkan pembangunan aplikasi yang lebih terdistribusi dan terkoneksi.
6. **Fleksibilitas dan Keciklan Waktu Pembaruan:** Pengembang dapat menulis fungsi kecil dan fokus pada tugas spesifik tanpa perlu mengelola seluruh aplikasi atau *server*. Ini dapat mempercepat siklus pengembangan dan pembaruan.
7. **Pemantauan dan Logging Terintegrasi:** GCF menyediakan pemantauan dan *logging* terintegrasi, memudahkan dalam melacak dan menganalisis perilaku fungsi. Ini dapat membantu dalam pemecahan masalah dan peningkatan kinerja.
8. **Dukungan untuk Berbagai Bahasa Pemrograman:** GCF mendukung beberapa bahasa pemrograman, termasuk *Node.js*, *Python*, *Go*, *Java*, dan lainnya. Ini memberikan fleksibilitas kepada pengembang untuk menggunakan bahasa yang paling sesuai dengan kebutuhan proyek.



## **B.Hubungan *Google cloud Platform* dan *googe cloud function***

Hubungan antara *Google cloud Platform* dan *Google cloud Functions* adalah sebagai berikut:

**1.GCP sebagai Platform Utama:** *GCP* adalah platform yang menyediakan berbagai layanan cloud. *GCF* adalah salah satu dari banyak layanan yang ditawarkan di bawah payung *GCP*.

**2.Fungsi Tanpa Server di GCP:** *GCF* memanfaatkan infrastruktur tanpa server, memungkinkan pengembang untuk fokus pada penulisan kode fungsi tanpa harus khawatir tentang manajemen server atau infrastruktur di belakangnya.

**3.Aktivasi Berdasarkan Peristiwa:** *Google cloud Functions* diaktifkan oleh peristiwa tertentu, seperti pemanggilan *HTTP*, perubahan pada penyimpanan *cloud Storage*, atau peristiwa lainnya di ekosistem *GCP*. Ini memungkinkan fungsi-fungsi tersebut merespons secara otomatis terhadap perubahan di lingkungan *cloud*.

**4.Integrasi dengan Layanan Lain di GCP:** *GCF* dapat berintegrasi dengan layanan-layanan lain di *GCP*, memungkinkan pengembang untuk membuat alur kerja yang kompleks dengan memanfaatkan berbagai layanan *cloud*.

**5.Fleksibilitas dan Skalabilitas:** *GCF* memberikan fleksibilitas dan skalabilitas yang baik, karena fungsi-fungsi yang ditulis dapat secara otomatis disesuaikan dengan jumlah beban kerja yang diterima.

Dengan demikian, *Google cloud Functions* adalah salah satu layanan di bawah *Google cloud Platform* yang menawarkan model komputasi tanpa server untuk pengembangan dan eksekusi fungsi berbasis peristiwa.

## BAB VII

# POMOKIT / POMODORO

### 1. Apa itu Pomodoro ?

Pomodoro adalah sebuah sistem atau metode manajemen waktu yang dikembangkan oleh Francesco Cirillo pada akhir 1980-an. Metode ini didasarkan pada penggunaan timer atau penghitung waktu untuk membagi pekerjaan menjadi periode waktu fokus yang disebut "pomodoro" (tomat dalam bahasa Italia). Nama "pomodoro" berasal dari timer dapur bentuk tomat yang digunakan oleh Cirillo ketika ia masih mahasiswa.

Berikut adalah prinsip dasar dari metode Pomodoro:

1. **Pomodoro (Tomat):** Satu siklus Pomodoro adalah periode waktu 25 menit. Selama periode ini, Anda fokus sepenuhnya pada tugas yang dihadapi.
2. **Istirahat Singkat:** Setelah satu siklus Pomodoro selesai, Anda memberikan diri Anda istirahat singkat selama 5 menit. Ini adalah waktu untuk menghilangkan mata dari layar, berdiri, dan melakukan peregangan atau kegiatan ringan lainnya.
3. **Pomodoro Berikutnya:** Setelah istirahat singkat, Anda mulai siklus Pomodoro berikutnya. Setiap empat siklus Pomodoro, disarankan untuk mengambil istirahat lebih panjang, misalnya 15-30 menit.

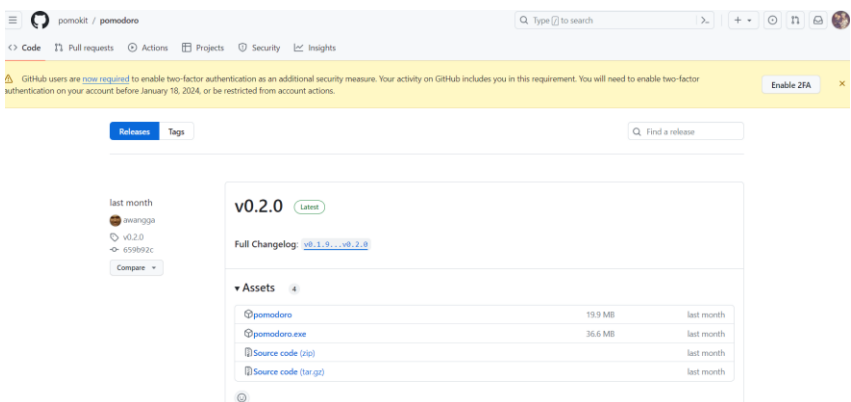
4. **Menghormati Pomodoro:** Penting untuk tidak terganggu selama periode fokus. Jangan mengabaikan atau menyela Pomodoro. Jika ada gangguan atau pemikiran yang muncul selama Pomodoro, catat dan tindakani setelahnya.

5. **Monitoring dan Evaluasi:** Anda dapat melacak kemajuan dan efektivitas Anda dengan mencatat berapa banyak Pomodoro yang telah Anda selesaikan dan jenis tugas apa yang telah diselesaikan selama periode waktu tertentu. Ini membantu Anda memahami bagaimana Anda menghabiskan waktu dan di mana perbaikan dapat dilakukan. Metode Pomodoro dirancang untuk meningkatkan fokus, produktivitas, dan manajemen waktu dengan memecah pekerjaan menjadi unit-unit waktu terfokus. Sementara metode ini mungkin tidak cocok untuk semua orang atau setiap jenis tugas, banyak orang menemukan bahwa pendekatan ini membantu mereka mengatasi prokrastinasi dan meningkatkan efisiensi kerja.

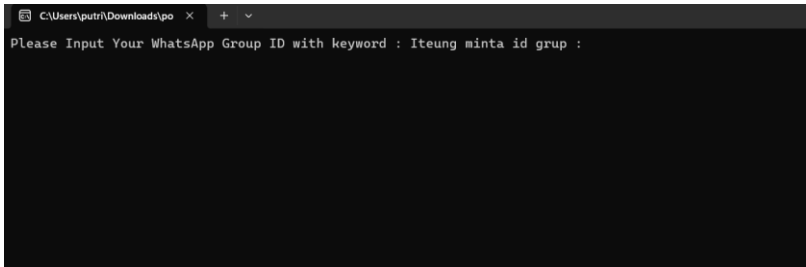
## A. Cara menggunakan Pomodoro:

1. Kunjungi dan Klik link *GitHub* dibawah ini :

**<https://GitHub.com/pomokit/pomodoro/releases>**



2. Pilih *pomodoro.exe* dan download
3. Buka file pomodoro, dan akan keluar seperti dibawah ini, dan *input* sesuai yang ada pada *powersell*



4. Jika sudah, terinput semua maka akan keluar kode *Scan QR* dan, tautkan perangkat pada *Whatsapp*.
5. Kemudian, pomodoro sudah tersambung di hp

#### **INFO PENTING:**

Pada saat melaksanakan Pomodoro kita akan diperintahkan task 25 menit untuk bekerja dan 5 menit untuk break, dan task yg terakhir adalah break 25 menit terakhir. Pada saat menggunakan pomodoro kita harus mencommit minimal 3x, agar pengerjaan kita dapat diklik oleh Dosen / penguji kita. Dan pomodoro ini akan merekam *Screenshoot* secara otomatis pada saat kita melakukan pomodoro.

# BAB VIII

## TUTORIAL MEMBUAT *WHATSAPP* AUTOSENDER

Jika Anda ingin membuat antarmuka pengguna (*UI*) front-end untuk ledakan *Whatsapp* Autosender, Anda dapat menggunakan *HTML*, *CSS*, dan *JavaScript* untuk pengembangan sisi klien. Di bawah ini adalah contoh sederhana untuk membantu Anda memulai. Contoh ini mengasumsikan bahwa Anda memiliki aplikasi sisi *server* (menggunakan bahasa seperti *Node.js*) untuk menangani fungsionalitas peledakan *Whatsapp* yang sebenarnya. Membuat *Whatsapp* Autosender *Web*,site melibatkan pengembangan di bagian *frontend* dan *backend*. Berikut langkah-langkah umumnya:

### **A.Frontend (Client-Side):**

#### **1. *HTML* dan *CSS* :**

- Buat struktur dasar *HTML* untuk halaman *web*,.
- Desain tata letak dan tampilan dengan menggunakan *CSS* .
- Pertimbangkan penggunaan framework *CSS* seperti *Bootstrap* atau *Tailwind CSS* untuk mempercepat pengembangan.

#### **2. *JavaScript* (Optional):**

- Implementasikan logika interaktif dengan *JavaScript*.
- Pastikan untuk menyertakan validasi *input* dan elemen interaktif pada halaman.

### **B.Backend (Server-Side):**

1. **Pilih Stack Teknologi:** Pilih teknologi *backend* yang sesuai. *Node.js*, *Python (Flask atau Django)*, atau *PHP* bisa menjadi pilihan tergantung pada keahlian Anda.
2. **Buat Server:** Inisialisasi *server backend* menggunakan framework atau *library* yang Anda pilih. Contoh: Jika Anda menggunakan *Node.js*, Anda bisa menggunakan *Express.js*.
3. **Endpoints API:** Tentukan *endpoints API* untuk berkomunikasi antara *frontend* dan *backend*. Misalnya, *endpoint* untuk menerima pesan dari *frontend* dan mengirimnya ke layanan *Whatsapp*.
4. **Integrasi dengan Whatsapp API:** Dapatkan akses ke *API Whatsapp Business* atau layanan *Whatsapp Gateway* tertentu. Gunakan *API* tersebut untuk mengirim dan menerima pesan *Whatsapp*.
5. **Logika Bisnis:** Implementasikan logika bisnis yang diperlukan, seperti pengelolaan antrian pesan, penyimpanan data pengguna, dan sebagainya.
6. **Keamanan:** Pastikan keamanan dengan mengimplementasikan otentikasi dan otorisasi yang aman. Enkripsi data yang dikirimkan antara *frontend* dan *backend*.

### **C.Integrasi Whatsapp API:**

1. **Daftar dan Dapatkan API Key:**
  - Daftar ke layanan *Whatsapp API* yang Anda pilih.
  - Dapatkan kunci *API* yang diperlukan untuk mengakses layanan tersebut.
2. **Konfigurasi API Key:** Konfigurasi *API Key* pada *backend* Anda agar dapat berkomunikasi dengan layanan *Whatsapp*.

3. **Uji Coba:** Lakukan uji coba untuk memastikan integrasi berjalan dengan baik.

**D.Deployment:**

1. **Hosting:** Pilih platform hosting untuk menyiapkan *frontend* dan *backend* Anda. Contoh: *Heroku, AWS, atau Google Cloud*.
2. **Deploy:** Deploy aplikasi *frontend* dan *backend* ke *platform* hosting yang Anda pilih.
3. **Monitor dan Maintenance:** Monitor performa aplikasi secara berkala. Lakukan pemeliharaan dan pembaruan sesuai kebutuhan.

**Catatan Tambahan:**

- Pastikan untuk mematuhi ketentuan layanan dan kebijakan privasi dari layanan *Whatsapp*.
- Berikan dokumentasi yang baik untuk penggunaan *frontend* dan *backend*.
- Pertimbangkan untuk menyertakan fitur-fitur seperti otentikasi dua faktor dan notifikasi.

Membangun *Whatsapp* Autosender dengan pendekatan *Micro frontends* membutuhkan beberapa langkah dasar. *Micro frontends* adalah pendekatan arsitektur di mana aplikasi *web*, dibangun sebagai sekumpulan modul independen, yang dapat dikelola dan diperbarui secara terpisah. Berikut adalah langkah-langkah umum yang dapat diambil untuk membuat *Whatsapp* Bot dengan menggunakan *Micro frontends*:

**1. Persiapkan Proyek Utama:**

- Inisiasi proyek utama sebagai aplikasi induk yang akan menjadi host untuk modul *Micro frontends*.
- Gunakan alat manajemen paket seperti npm atau yarn untuk mengelola dependensi.

**2. Pilih Framework *Micro frontends*:**

- Pilih framework atau *library Micro frontends* yang sesuai dengan kebutuhan proyek Anda, seperti *Single-SPA* atau *Module Federation* di *Webpack*.

**3. Desain *Micro frontends*:**

- Pisahkan fungsionalitas bot ke dalam modul-modul independen, misalnya, satu modul untuk antarmuka pengguna, satu untuk logika bisnis, dan satu lagi untuk integrasi dengan *API Whatsapp*.

**4. Pengaturan *API Whatsapp*:**

- Dapatkan akses ke *API Whatsapp* melalui *Whatsapp Business API*.
- Peroleh token dan konfigurasi yang diperlukan untuk berinteraksi dengan layanan *Whatsapp*.



5. **Pengembangan Modul *frontend Whatsapp*:**

- Implementasikan modul *Micro frontend* yang bertanggung jawab untuk antarmuka pengguna bot *Whatsapp*.
- Gunakan teknologi *web*, seperti *React*, *Vue*, atau *Angular* sesuai dengan preferensi Anda.

6. **Pengembangan Modul Logika Bisnis:**

- Buat modul terpisah untuk logika bisnis bot, termasuk manajemen pesan, logika pemrosesan, dan interaksi dengan *API Whatsapp*.

7. **Integrasi dengan *Whatsapp API*:**

- Terapkan logika integrasi dengan *API Whatsapp* pada modul bisnis.
- Pastikan untuk menangani otorisasi dan autentikasi sesuai dengan kebijakan *Whatsapp*.

8. **Pengaturan *Micro frontends*:**

- Konfigurasi aplikasi utama untuk memuat dan mengeksekusi modul *Micro frontends* saat aplikasi dijalankan.

9. **Uji dan Pemecahan Masalah:**

- Uji integrasi antara modul *Micro frontends* dan pastikan bahwa bot dapat berfungsi dengan baik.
- Perbaiki masalah atau bug yang mungkin muncul selama proses pengembangan.

## 10. Penyebaran:

- Terapkan proses penyebaran modul *Micro frontends* secara terpisah jika memungkinkan.
- Pastikan bahwa perubahan di modul tertentu tidak memengaruhi keseluruhan aplikasi.

## 11. Pemeliharaan dan Pembaruan:

- Kelola dan perbarUI modul *Micro frontends* secara terpisah sesuai kebutuhan.
- Terus pantau perubahan di *API Whatsapp* dan sesuaikan integrasi jika diperlukan.

## TUTORIAL MEMBUAT AUTOSENDER PADA WHATSAPP

Tentukan bahasa yang digunakan untuk membuat *backend* dan *frontend*, disini kami menggunakan *HTML* dan sebagai *frontend* dan untuk controller menggunakan *PHP*, serta *JavaScript* sebagai *backend*, juga kami menggunakan *MySQL* untuk database.

### A. membuat *Whatsapp* sender

melibatkan pengembangan antarmuka pengguna (*UI*) yang memungkinkan pengguna untuk berinteraksi dengan aplikasi dan mengirim pesan melalUI layanan *Whatsapp*. Berikut adalah beberapa tugas kunci yang perlu diperhatikan dalam pembuatan *frontend* untuk *Whatsapp* sender:

#### 1. Desain Antarmuka Pengguna (*UI/UX*):

- **Desain Responsif:** Pastikan antarmuka dapat diakses dan berfungsi dengan baik di berbagai perangkat dan ukuran layar.

- **Navigasi yang Intuitif:** Rancang navigasi yang mudah dipahami dan intuitif sehingga pengguna dapat dengan cepat mengakses fitur yang mereka butuhkan.

- **Elemen UI yang Jelas:** Gunakan elemen UI yang jelas dan bersih, seperti tombol, formulir, dan ikon, untuk meningkatkan pengalaman pengguna.

Contoh: <https://GitHub.com/WASENDERAUTO/WACYBEROREN/tree/main/core/resources/views>

## 2. Autentikasi Pengguna:

- **Formulir Login/Registrasi:** Sediakan formulir login dan registrasi untuk otentikasi pengguna. Validasi *input* di sisi klien untuk mencegah kesalahan.

Contoh:

<https://GitHub.com/WASENDERAUTO/Warenweb,.GitHub.io/blob/main/register.HTML>

Contoh:

<https://GitHub.com/WASENDERAUTO/Warenweb.GitHub.io/blob/main/register.HTML>

- **Manajemen Sesi:** Implementasikan manajemen sesi untuk menjaga agar pengguna tetap diotentikasi selama sesi mereka.

## 3. Pengelolaan Pengguna:

- **Daftar Kontak:** Tampilkan daftar kontak pengguna dan kemungkinan pengelolaan kontak.

- **Manajemen Grup:** Jika mendukung pengiriman pesan grup, sediakan antarmuka untuk membuat, mengelola, dan mengirim pesan ke grup.

#### 4. Pengiriman Pesan:

- **Formulir Pengiriman Pesan:** Sediakan formulir pengiriman pesan yang memungkinkan pengguna memilih kontak atau grup tujuan dan menginput pesan.
- **Pemilihan Media:** Jika mendukung pengiriman media seperti gambar atau video, sediakan opsi untuk memilih dan mengunggah file-media.

#### 5. Riwayat Pesan:

- **Tampilan Riwayat Pesan:** Tampilkan riwayat pesan dengan kontak atau grup tertentu.
- **Filter dan Pencarian Pesan:** Sediakan opsi untuk memfilter atau mencari pesan tertentu dalam riwayat.

#### 6. Pengaturan Akun:

- **Pengaturan Profil:** Izinkan pengguna untuk mengelola pengaturan profil mereka, seperti foto profil, status, atau pengaturan lainnya.
- **Pengaturan Aplikasi:** Sediakan opsi pengaturan aplikasi, seperti notifikasi atau preferensi pengguna lainnya.

#### 7. Keamanan:

- **Enkripsi:** Pastikan komunikasi antara *frontend* dan *backend* dienkripsi untuk menjaga keamanan data.
- **Validasi Input:** Lakukan validasi *input* di sisi klien dan *server* untuk mencegah serangan XSS atau injeksi data berbahaya.

#### 8. Integrasi API:

- **Integrasi Backend:** Hubungkan *frontend* dengan *backend* yang memiliki logika pengiriman pesan dan interaksi dengan layanan *Whatsapp*.

## 9. Notifikasi:

- **Notifikasi *Real-Time*:** Jika memungkinkan, implementasikan notifikasi *real-time* untuk memberi tahu pengguna tentang pesan masuk atau pembaruan lainnya.

## 10. Pengujian dan Pemeliharaan:

- **Pengujian Fungsionalitas:** Lakukan pengujian fungsionalitas secara menyeluruh untuk memastikan setiap fitur berfungsi seperti yang diharapkan.

- **Pemeliharaan dan Pembaruan:** Pastikan untuk memelihara aplikasi dengan melakukan pembaruan, perbaikan *bug*, dan peningkatan fungsionalitas.

- **Optimasi Kinerja:** Pastikan antarmuka berjalan dengan lancar dan memiliki kinerja yang baik, terutama jika aplikasi memiliki banyak pengguna.

Selain itu, pastikan untuk mematuhi pedoman desain dan keamanan, dan pertimbangkan untuk menggunakan kerangka kerja *JavaScript* seperti *React*, *Vue*, atau *Angular* untuk memudahkan pengembangan dan memelihara aplikasi *frontend* Anda.

## B. Tugas *backend*

Menggunakan *JavaScript* dapat melibatkan sejumlah hal tergantung pada kebutuhan proyek dan tujuannya. Berikut beberapa contoh tugas yang mungkin Anda hadAPI dalam pengembangan *backend* menggunakan *JavaScript*, khususnya dengan *Node.js*:

### 1. **Pengelolaan Data:**

Membuat, membaca, memperbarUI, dan menghapus data dari basis data. Anda dapat menggunakan database *SQL* seperti *MySQL* atau *PostgreSQL*, atau database *NoSQL* seperti *MongoDB*, Contoh:

membuat database *MySQL* otomatis melibatkan beberapa langkah, termasuk membuat koneksi ke *server MySQL*, mengeksekusi pernyataan *SQL* untuk membuat database, dan menutup koneksi setelah selesai. Berikut adalah contoh menggunakan *Node.js* dan *library MySQL* untuk membuat database *MySQL* otomatis:

**1. Instalasi Paket:** Pastikan Anda telah menginstal paket *MySQL* melalUI *npm*. Anda dapat melakukannya dengan menjalankan perintah berikut di terminal atau command prompt:

```
npm install MySQL
```

### 2. **Script Pembuatan Database:**

Buat *Script Node.js* yang akan membuat database. Contoh *Script* berikut menggunakan modul *MySQL*:

```
const MySQL = require('mysql');  
// Konfigurasi koneksi ke MySQL  
const connection = MySQL.createConnection({  
  host: 'localhost',  
  user: 'root',  
  password: 'password'  
});
```

```
// Nama database yang akan dibuat
const dbName = 'nama_database_baru';

// Membuat koneksi ke server MySQL
connection.connect((err) => {
  if (err) throw err;

  console.log('Connected to MySQL server');

  // Membuat database
  connection.query(`CREATE DATABASE ${dbName}`, (err,
result) => {
    if (err) throw err;
    console.log(`Database ${dbName} created successfully`);
  });

  // Menutup koneksi setelah membuat database
  connection.end((err) => {
    if (err) throw err;
    console.log('Connection closed');
  });
});
```

**3.Menjalankan *Script*:** Simpan *Script* di atas dalam file misalnya *createDatabase.js* dan jalankan di terminal atau *command prompt*:

```
node createDatabase.js
```

Pastikan untuk mengganti nilai *host*, *user*, *password*, dan *dbName* sesuai dengan konfigurasi *MySQL* Anda.

**4.Verifikasi:** Setelah menjalankan *Script*, Anda dapat memverifikasi bahwa database baru telah dibuat dengan menggunakan alat manajemen database atau perintah *SQL* seperti:

```
SHOW DATABASES
```

Pastikan database baru (nama\_database\_baru dalam contoh ini) terdaftar.

**Catatan:** Pastikan bahwa akses ke *server MySQL* sudah diizinkan dan bahwa Anda memiliki hak akses yang cukup untuk membuat database. Anda juga dapat menyimpan kredensial koneksi di tempat yang lebih aman, misalnya dalam variabel lingkungan atau berkas konfigurasi. Selain itu, penting untuk mencatat bahwa membuat database secara otomatis seperti ini umumnya hanya dilakukan dalam skenario pengembangan atau setup awal. Dalam lingkungan produksi, biasanya tugas ini sudah diatur dengan lebih cermat untuk memastikan keamanan dan keandalan.

## **2.Pengelolaan Pengguna:**

Pembuatan dan otentikasi pengguna adalah komponen penting dalam pengembangan aplikasi atau bot, termasuk bot *Whatsapp*. Dalam konteks bot *Whatsapp* menggunakan *JavaScript*, Anda perlu membuat mekanisme untuk mendaftarkan pengguna dan mengotentikasi identitas mereka. Dalam contoh ini, kami akan menggunakan *Node.js* dan *Express* untuk membuat *server backend* sederhana



## Pembuatan dan Otentikasi Pengguna dengan *Node.js* dan *Express*:

1. **Instalasi Dependensi:** Pastikan Anda telah menginstal *Node.js* di komputer Anda. Buat folder proyek dan jalankan perintah berikut untuk menginisialisasi proyek *Node.js* dan menginstal dependensi *Express*:

```
npm init -y  
npm install express body-parser
```

### 2. File *index.js*:

Buat file *index.js* untuk *server Express*:

```
const express = require('express');  
const bodyParser = require('body-parser');  
const app = express();  
const port = 3000;  
  
// Middleware untuk parsing body dari permintaan HTTP  
app.use(bodyParser.json());  
app.use(bodyParser.urlencoded({ extended: true }));  
  
// Daftar pengguna (simpannya di memori, seharusnya  
disimpan di basis data)  
const users = [];  
  
// Rute untuk mendaftar pengguna  
app.post('/register', (req, res) => {  
  const { username, password } = req.body;  
  
  // Simpan pengguna ke dalam daftar (seharusnya disimpan  
di basis data)  
  users.push({ username, password });
```

```
res.status(201).json({ message: 'Pengguna terdaftar dengan
sukses!' });
});

// Rute untuk mengotentikasi pengguna
app.post('/login', (req, res) => {
  const { username, password } = req.body;

  // Cari pengguna di daftar (seharusnya dicari di basis data)
  const user = users.find(u => u.username === username &&
u.password === password);

  if (user) {
    res.json({ message: 'Otentikasi berhasil!' });
  } else {
    res.status(401).json({ message: 'Otentikasi gagal. Periksa
kembali username dan password.' });
  }
});

// Jalankan server
app.listen(port, () => {
  console.log(`Server berjalan di HTTP://localhost:${port}`);
});
```

### 3. Menjalankan Server:

Jalankan *server* dengan menjalankan perintah:

```
node index.js
```

#### 4.Registrasi Pengguna:

Gunakan aplikasi pihak ketiga (misalnya, Postman) untuk membuat permintaan POST ke `HTTP://localhost:3000/register` dengan payload *JSON*:

```
{
  "username": "user123",
  "password": "pass123"
}
```

#### 5.Otentikasi Pengguna:

Lakukan permintaan POST ke `HTTP://localhost:3000/login` dengan payload *JSON*:

```
{
  "username": "user123",
  "password": "pass123"
}
```

Jika informasi otentikasi benar, *server* akan mengembalikan pesan "Otentikasi berhasil!" Ini adalah contoh sederhana dan seharusnya tidak digunakan di lingkungan produksi. Untuk keamanan yang lebih baik, Anda seharusnya menggunakan otentikasi yang kuat dan menyimpan informasi pengguna di basis data yang aman. Selain itu, pertimbangkan penggunaan teknologi otentikasi yang mapan seperti *JSON Web*, Tokens (JWT) untuk mengelola sesi dan otentikasi pengguna.

### 3..API RESTful:

Membuat dan menjaga API RESTful untuk berkomunikasi dengan *frontend* atau aplikasi lainnya. MenanggAPI permintaan HTTP seperti GET, POST, PUT, dan DELETE.

Untuk membuat dan menjaga API RESTful untuk berkomunikasi dengan *frontend* atau aplikasi lainnya dalam konteks bot *Whatsapp*, Anda dapat menggunakan *Node.js* dan Express. Berikut adalah langkah-langkah umum untuk membuat API RESTful sederhana yang menanggAPI permintaan HTTP seperti GET, POST, PUT, dan DELETE:

#### 1.Instalasi Dependensi:

Pastikan Anda telah menginstal *Node.js* dan npm. Buat folder proyek dan jalankan perintah berikut untuk menginisialisasi proyek dan menginstal dependensi Express:

```
npm init -y  
npm install express body-parser
```

#### 2.File index.js:

Buat file *index.js* untuk *server* Express dan API RESTful:

```
const express = require('express');  
const bodyParser = require('body-parser');  
const app = express();  
const port = 3000;  
// Middleware untuk parsing body dari permintaan HTTP  
app.use(bodyParser.json());  
app.use(bodyParser.urlencoded({ extended: true }));  
  
// Contoh data (seharusnya disimpan di basis data)  
let messages = [
```

```

    { id: 1, text: 'Hello, how can I help you?' },
    { id: 2, text: 'This is another message.' }
  ];

  // Rute GET untuk mendapatkan semua pesan
  app.get('/messages', (req, res) => {
    res.json(messages);
  });

  // Rute GET untuk mendapatkan pesan berdasarkan ID
  app.get('/messages/:id', (req, res) => {
    const messageId = parseInt(req.params.id);
    const message = messages.find(msg => msg.id ===
    messageId);
    if (message) {
      res.json(message);
    } else {
      res.status(404).json({ message: 'Message not found' });
    }
  });

  // Rute POST untuk menambahkan pesan baru
  app.post('/messages', (req, res) => {
    const newMessage = req.body;
    messages.push(newMessage);
    res.status(201).json(newMessage);
  });

  // Rute PUT untuk memperbaiki pesan berdasarkan ID
  app.put('/messages/:id', (req, res) => {
    const messageId = parseInt(req.params.id);
    const updatedMessage = req.body;

    messages = messages.map(msg => (msg.id === messageId ?
    updatedMessage : msg));
  });

```

```
res.json(updatedMessage);
});

// Rute DELETE untuk menghapus pesan berdasarkan ID
app.delete('/messages/:id', (req, res) => {
  const messageId = parseInt(req.params.id);
  messages = messages.filter(msg => msg.id !== messageId);

  res.json({ message: 'Message deleted successfully' });
});

// Jalankan server
app.listen(port, () => {
  console.log(`Server berjalan di HTTP://localhost:${port}`);
});
```

### 3. Menjalankan Server:

Jalankan *server* dengan perintah:

```
node index.js
```

### 4. Uji API:

Gunakan aplikasi pihak ketiga seperti Postman atau curl untuk menguji *API*. Contoh permintaan dan respons:

#### A. GET Semua Pesan:

```
curl HTTP://localhost:3000/messages
```

#### B.GET Pesan Berdasarkan ID:

```
curl HTTP://localhost:3000/messages/1
```

#### C.POST Pesan Baru:

```
curl -X POST -H "Content-Type: application/json" -d '{"text": "New message"}' HTTP://localhost:3000/messages
```

#### D.PUT Pembaruan Pesan Berdasarkan ID:

```
curl -X PUT -H "Content-Type: application/json" -d '{"text": "Updated message"}' HTTP://localhost:3000/messages/1
```

#### E.DELETE Pesan Berdasarkan ID:

```
curl -X DELETE HTTP://localhost:3000/messages/1
```

Pastikan untuk menggantinya dengan logika sesuai kebutuhan bot *Whatsapp* Anda. Ini hanya contoh sederhana, dan dalam penggunaan sebenarnya, Anda kemungkinan besar perlu menyimpan dan mengelola data di basis data dan menerapkan keamanan yang sesuai.

#### 4.Integrasi dengan Layanan Pihak Ketiga:

Menyambungkan *backend* dengan layanan eksternal seperti *API* pihak ketiga (mungkin integrasi media sosial, pembayaran, dll.). contoh:

```
const axios = require('axios');  
  
// URL API pihak ketiga (gantilah dengan URL yang sesuai)  
const externalAPIUrl = 'https://API.example.com';
```

```
// Contoh fungsi untuk mengambil data dari API pihak ketiga
async function fetchDataFromExternalAPI() {
  try {
    const response = await axios.get(externalAPIUrl +
'/endpoint');
    const data = response.data;
    console.log('Data from external API:', data);
  } catch (error) {
    console.error('Error fetching data:', error.message);
  }
}

// Contoh fungsi untuk mengirim data ke API pihak ketiga
async function sendDataToExternalAPI(dataToSend) {
  try {
    const response = await axios.post(externalAPIUrl +
'/endpoint', dataToSend);
    const responseData = response.data;
    console.log('Response from external API:', responseData);
  } catch (error) {
    console.error('Error sending data:', error.message);
  }
}

// Panggil fungsi-fungsi di atas
fetchDataFromExternalAPI();

// Contoh data yang akan dikirim ke API pihak ketiga (gantilah
dengan data yang sesuai)
const sampleData = {
  key1: 'value1',
  key2: 'value2'
};

sendDataToExternalAPI(sampleData);
```



## 5. Middleware dan Penggunaan Express.js:

Menerapkan middleware untuk otentikasi, otorisasi, logging, dan fungsionalitas lainnya. Menggunakan Express.js untuk mengelola rute dan permintaan *HTTP*. Contoh:

```
const express = require('express');
const app = express();
const port = 3000;

// Middleware otentikasi sederhana
const authenticate = (req, res, next) => {
  const authToken = req.headers.authorization;

  if (authToken === 'secret_token') {
    // Pengguna dianggap terotentikasi
    next();
  } else {
    res.status(401).json({ message: 'Unauthorized' });
  }
};

// Middleware logging
const logRequest = (req, res, next) => {
  console.log(`[${new Date().toLocaleString()}]
${req.method} ${req.url}`);
  next();
};

// Menggunakan middleware untuk seluruh aplikasi
app.use(authenticate);
app.use(logRequest);

// Rute yang memanfaatkan middleware
```

```

app.get('/', (req, res) => {
  res.send('Hello, this is the home route!');
});

// Rute lain yang memanfaatkan middleware
app.get('/secure', (req, res) => {
  res.send('This route is secure and reqUires authentication.');
```

```

});

// Jalankan server
app.listen(port, () => {
  console.log(`Server is running on HTTP://localhost:${port}`);
});
```

## 6.Manajemen Kode:

Strukturisasi proyek *backend* Anda agar mudah dipahami dan dikembangkan. Memisahkan logika bisnis dari logika rute dan pengelolaan database.

## 7.Keamanan:

Menangani keamanan aplikasi, seperti pengamanan *input* pengguna, pencegahan serangan injeksi SQL, dan perlindungan terhadap serangan CSRF.

### Contoh Pengamanan *input* Pengguna

```

const { body, validationResult } = reqUire('express-validator');

app.post('/route', [
  body('username').isAlphanumeric(),
  body('email').isEmail(),
  // ... tambahkan aturan validasi lainnya
], (req, res) => {
```

```
const errors = validationResult(req);
if (!errors.isEmpty()) {
  return res.status(400).json({ errors: errors.Array() });
}

// Proses data jika valid
});
```

### Contoh Pencegahan Serangan Injeksi Sql

```
const MySQL = reqUrlre('mysql');
const connection = MySQL.createConnection({ /* konfigurasi koneksi */ });

const username = req.body.username;
const password = req.body.password;

// Parameterized query untuk mencegah injeksi SQL
const query = 'SELECT * FROM users WHERE username = ? AND password = ?';
connection.query(query, [username, password], (error, results) => {
  // Handle hasil query
});
```

### Contoh Perlindungan Terhadap Serangan CSRF

```
const csrf = reqU/re('csrf');
const cookieParser = reqU/re('cookie-parser');

app.use(cookieParser());
app.use(csrf({ cookie: true }));

// Sertakan token CSRF dalam respons untuk setiap
// permintaan
app.use((req, res, next) => {
  res.cookie('XSRF-TOKEN', req.csrfToken());
  next();
});

// Gunakan token CSRF pada permintaan POST
app.post('/route', (req, res) => {
  const token = req.body._csrf;
  // Validasi token sebelum memproses permintaan
});
```

### 8.Cron Jobs dan Penjadwalan Tugas:

Menjadwalkan dan menjalankan tugas otomatis pada waktu tertentu menggunakan cron jobs atau alat penjadwalan lainnya.

Contoh:

```
const cron = reqU/re('node-cron');

// Tugas yang akan dijadwalkan (ganti dengan logika bisnis
// Anda)
const scheduledTask = () => {
  console.log('Tugas dijalankan pada: ', new Date());
};
```

```
// Tambahkan logika bisnis atau panggil fungsi lain di sini
};

// Menjadwalkan tugas menggunakan cron syntax (setiap
menit pada menit ke-30)
cron.schedule('30 * * * *', scheduledTask);

console.log('Tugas telah dijadwalkan.');
```

```
// Biarkan proses berjalan sehingga tugas dapat dijalankan
setInterval(() => {}, 1000);
```

### **9.Manajemen File:**

Menangani unggahan dan unduhan file.

Menyimpan file di *server* atau penyimpanan awan seperti *AWS S3* atau *Google cloud Storage*.

### **10.Logging dan Monitoring:**

Menerapkan sistem logging untuk melacak aktivitas dan kesalahan.

Menyiapkan sistem pemantauan untuk memantau kinerja dan kesehatan aplikasi.

### **11.Pengujian dan Penjaminan Kualitas:**

Menulis unit test dan tes integrasi untuk memastikan kestabilan dan keandalan *backend*.

Melibatkan praktik pengujian seperti Test-Driven Development (TDD).

## 12.Optimasi dan Skalabilitas:

Mengoptimalkan kinerja *backend*.

Merancang *backend* agar dapat diskalakan secara *horizontal*.

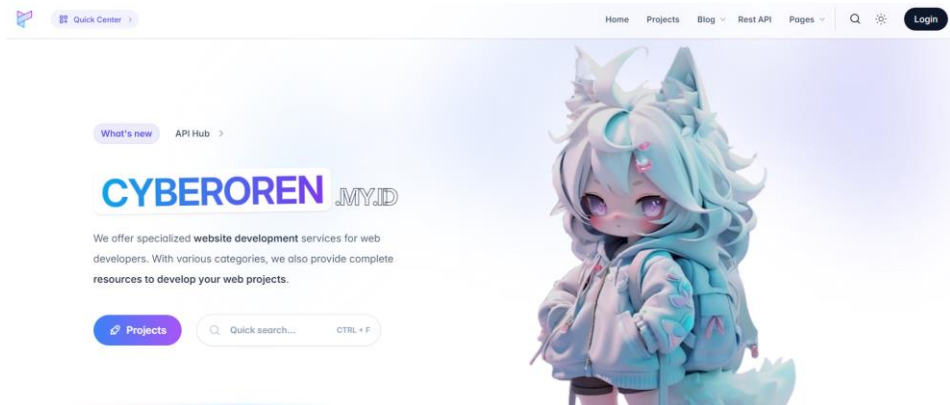
## 13.Dokumentasi:

Menyusun dokumentasi lengkap untuk *backend*, mencakup cara mengonfigurasi, menjalankan, dan berkontribusi pada proyek.

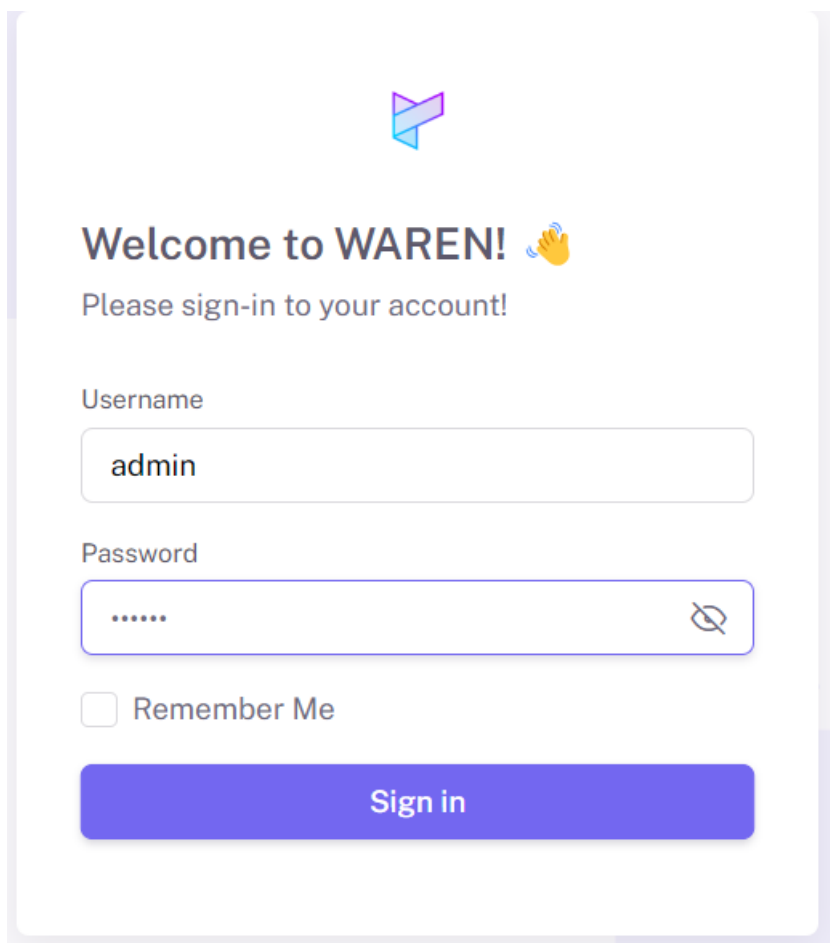
Setiap proyek *backend* memiliki kebutuhan yang unik, jadi pastikan untuk merinci tugas yang spesifik sesuai dengan persyaratan proyek yang Anda kerjakan. Selain itu, penting untuk memahami kebijakan keamanan dan praktik terbaik dalam pengembangan *backend* untuk menghasilkan solusi yang aman, efisien, dan dapat diskalakan.

Pastikan untuk merinci langkah-langkah ini sesuai dengan teknologi dan alat yang Anda pilih. Selain itu, pertimbangkan juga untuk melibatkan tim pengembang *frontend* dan *backend* untuk mengoptimalkan pengembangan dan pemeliharaan bot *Whatsapp* dengan pendekatan *Micro frontends*.

Berikut tampilan dari aplikasi yang telah dibuat:

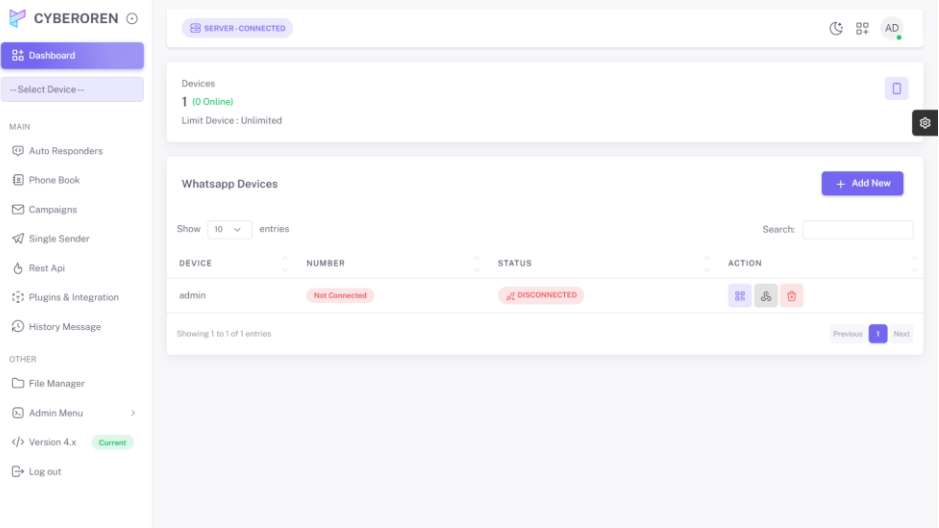


Gambar tampilan awal aplikasi

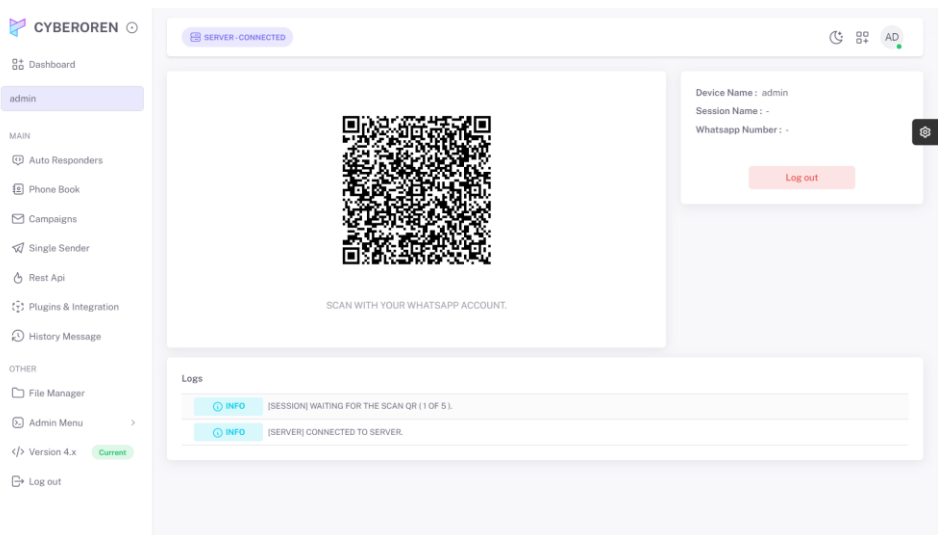


The image shows a login screen for an application named 'WAREN'. At the top center is a logo consisting of two overlapping chevron-like shapes, one purple and one blue. Below the logo, the text 'Welcome to WAREN!' is displayed in a bold, dark blue font, followed by a yellow hand icon with blue motion lines. Underneath this, a message 'Please sign-in to your account!' is written in a smaller, grey font. The login form includes a 'Username' label above a text input field containing the text 'admin'. Below that is a 'Password' label above a password input field with masked characters '.....' and a toggle icon on the right. A checkbox labeled 'Remember Me' is positioned below the password field. At the bottom of the form is a large, rounded blue button with the text 'Sign in' in white.

Gambar tampilan login

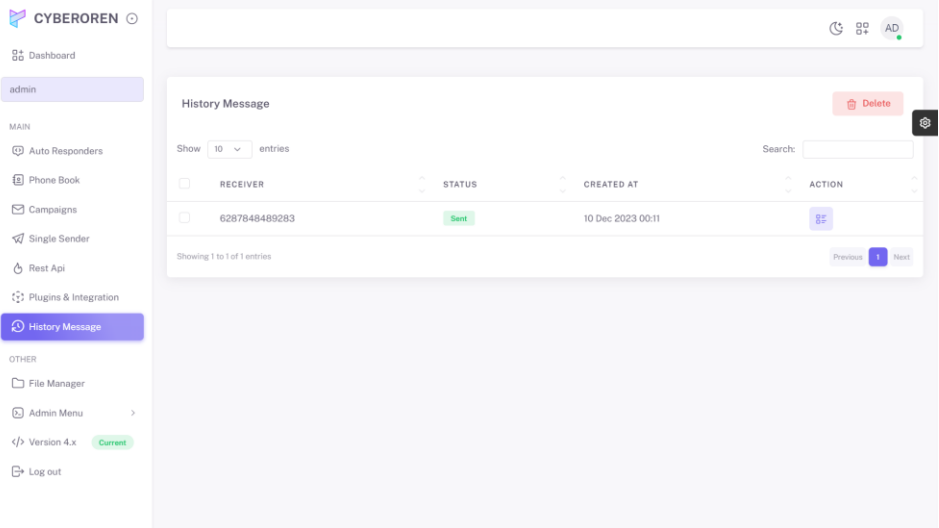


Gambar tampilan aplikasi

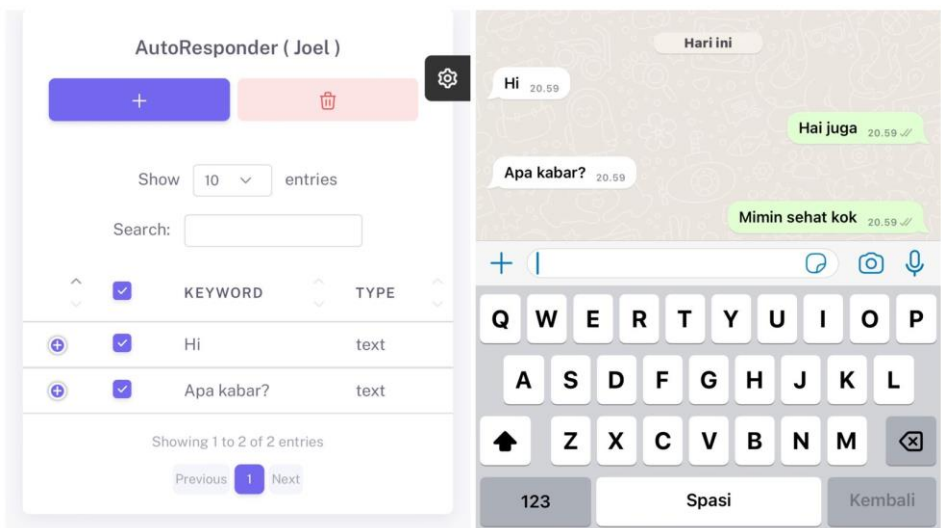


Gambar scan kode QR





Gambar hasil history pesan



Gambar keyword dan Hasil Autosender

Tutorial ini memberikan gambaran umum. Detail implementasi akan tergantung pada teknologi yang Anda pilih dan kebutuhan spesifik proyek Anda, Contoh Source Code ada dilink *GitHub* ini:

<https://GitHub.com/WASENDERAUTO/MATERIBUKPED/TutorialWaAutosender>

Demikian tutorial buku yang kami buat mengenai **RAHASIA EFEKTIFITAS MENGOPTIMALKAN KOMUNIKASI DENGAN AUTOSENDER WHATSAPP** semoga dapat memperluas wawasan pembaca. Terimakasih

## DAFTAR PUSTAKA

- R, A. D., Imamah, F., Andre S, Y. M., & Ardiansyah. (2018). APLIKASI *CHATBOT* (MILKI BOT)YANG TERINTEGRASI DENGAN *WEB*, CMS. Jurnal Cendikia Vol. XVI Cendikia , 0216-9436.
- Putra Prima, P. ( 2018). DESAIN *FRAMEWORK* LINE *CHATBOT* MENGGUNAKAN NODEJS DAN MONGODB. Jurnal prosiding SENTIA Vol. 11, ISSN: 2085-2347
- Heriyanto. (2015), SISTEM *CHATBOT* UNTUK MEMBANTU DIAGNOSA PADA KERUSAKAN KOMPUTER. Jurnal Sains & Teknologi, Vol. V, No. 2, September 2015, 2-7.
- Eka Yuniar., & Heri Purnomo. (2019). IMPLEMENTASI *CHATBOT* "ALITTA" ASISTEN VIRTUAL DARI BALITTAS SEBAGAI PUSAT INFORMASI DI BALITTAS. Jurnal Ilmiah TeknikInformatika (p – ISSN: 1978 – 5232; e – ISSN: 2527 – 337X) Vol. 12 No. 1 Mei 2019, pp. 24
- Rahartri., (2019). “*WHATSAPP*” MEDIA KOMUNIKASI EFEKTIF MASA KINI (STUDI KASUS PADA LAYANAN JASA INFORMASI ILMIAH DI KAWASAN PUSPIPTEK). Jurnal VISIPUSTAKA Vol. 21, No. 2, Agustus 2019.
- Nelfira., (2018), Rancang Bangun Aplikasi Pembelajaran Sistem Operasi Windows Pada Matakuliah Sistem Operasi Di STMIK Indonesia Padang Berbasis Multimedia Interaktif. Jurnal Edik Informatika, V2.i2(182-189)
- Nadira., (2016). APLIKASI KOLABORATIF PEMBELAJARAN *Online*. Jurnal fasilkom Vol. IV/No.1/Juni/2016.
- A. Yudi Permana., (2019). PERANCANGAN SISTEM INFORMASI PENJUALAN PERUMAHAN MENGGUNAKAN METODE SDLC PADA PT. MANDIRI LAND PROSPEROUS BERBASIS MOBILE. Jurnal Teknologi pelita Bangsa, Vol. 10, No. 2, Desember 2019, ISSN : 2407- 3903.
- Tri susyanto., & Deny Lestiono. (2019). OPTIMALISASI PENGGUNAAN CMD DAN SYSINTERNALSU/TS SEBAGAI MALWARE DETECTION. Jurnal TRANSFORMASI, Vol. 15, No. 1, 2019 : 65 – 74

- Nisaul Fadila., & Rinabi Tanama. (2021). Penerapan Rule-Based Expert System (RBES) Dalam Perancangan Aplikasi Sistem Pakar Untuk Mendiagnosa Penyakit Infeksi Saluran Pernapasan Akut (ISPA) Berbasis Android.
- Jurnal Ilmiah Teknologi Informasi Asia Vol.15, No.2, Tahun 2021 ISSN: 2580-8397 (O); 0852-730X
- Hormansyah, D. S. and Utama, Y. P. (2018) 'Aplikasi *Chatbot* Berbasis *Web*, Pada Sistem Informasi Layanan Publik Kesehatan Di Malang Dengan Menggunakan Metode Tf-Idf', Jurnal Informatika Polinema, 4(3), p. 224

## KREDIT GAMBAR

Gambar 2.1 *JavaScript* (<https://www.niagahoster.co.id/blog/belajar-JavaScript/>)

Gambar 2.2 *JavaScript* (<https://www.niagahoster.co.id/blog/belajar-JavaScript/>)

Gambar 2.3 *JavaScript* (<https://www.niagahoster.co.id/blog/belajar-JavaScript/>)

Gambar 2.4 Interpreter (<https://www.niagahoster.co.id/blog/belajar-JavaScript/>)

Gambar 2.5 Belajar *JavaScript* (<https://www.niagahoster.co.id/blog/belajar-JavaScript/>)

Gambar 2.6 *Reserved Words in JavaScript*  
(<https://www.niagahoster.co.id/blog/belajar-JavaScript/>)

Gambar 2.7 Hasil *Script* (<https://www.niagahoster.co.id/blog/belajar-JavaScript/>)

Gambar 2.8 Hasil *Script* (<https://www.niagahoster.co.id/blog/belajar-JavaScript/>)

Gambar 4.1 Hasil *Script* (<https://www.niagahoster.co.id/blog/json-adalah/>)

Gambar 8.1 Pengetesan seluruh folder (-)

## Tentang Penulis



Daslan Josua Valentino Siahaan, lahir di kota medan pada tanggal 12 desember 2002. Pendidikan tingkat dasar hingga menengah dan atas di tempuh di pekanbaru. Melanjutkan Pendidikan D4 D4 Teknik Informatika di Universitas Logistik Dan Bisnis Internasional (ULBI), Bandung.



Marjuniati Putri lahir di kota bima, pada tanggal 30 Juni 2003. Pendidikan tingkat dasar hingga menengah dan atas di tempuh di Bima. Melanjutkan Pendidikan D4 Teknik Informatika di Universitas Logistik Dan Bisnis Internasional (ULBI), Bandung.



Roni Andarsyah, lahir di Padaherang pada tanggal 20 Mei 1988. Menempuh Pendidikan D3 jurusan Teknik Informatika di Polekpos Bandung, kemudian melanjutkan S1 di ST Inten Bandung dengan jurusan yang sama. Dan melanjutkan Pendidikan S2 di STMIK LIKMI jurusan Sistem Informasi.

# JS

Buku ini membawa para pembaca dalam perjalanan mendalam ke dunia pengembangan *Whatsapp* dengan fokus pada pemanfaatan API dan pendekatan micro frontends. Dalam era teknologi yang terus berkembang, penulis mengungkapkan metode terbaik untuk memahami dan mengimplementasikan fitur-fitur canggih pada *Whatsapp* melalui penggunaan API yang tersedia.

Pembaca akan dibimbing melalui konsep dasar pengembangan *Whatsapp*, termasuk integrasi API untuk memperluas fungsionalitas. Buku ini tidak hanya memberikan panduan langkah demi langkah, tetapi juga memberikan wawasan mendalam tentang arsitektur micro frontends dan bagaimana hal itu dapat diterapkan secara efektif dalam pengembangan *Whatsapp*.

Buku ini dirancang untuk para pengembang perangkat lunak yang ingin mendalami pengembangan *Whatsapp* dengan memanfaatkan API dan menerapkan konsep micro frontends. Dengan membaca buku ini, pembaca diharapkan dapat menguasai keterampilan yang diperlukan untuk menciptakan pengalaman pengguna yang responsif, efisien, dan terdepan dalam industri.