BELAJAR LARAVEL

EKO SISWANTO, M.KOM

BELAJAR LARAVEL

EKO SISWANTO, M.KOM

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
JL. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Judul : Belajar Laravel

Penulis:

Eko Siswanto, S.Kom., M.Kom

ISBN : 978-623-8120-43-7 (PDF)

Editor:

Indra Ava Dianta, S.Kom., M.T

Haris Ihsanil Huda, S.Kom, M.M, M.Kom

Zaenal Mustofa, M.Kom

Penyunting :

Agustinus Budi Santoso, S.ST, M.Cs

Bagus Sudirman, S.Kom, M.Kom

Desain Sampul dan Tata Letak :

Muhammad Sholikhan, S.Kom., M.Kom

Penerbit :

Yayasan Prima Agus Teknik
Redaksi: Jln Majapahit No 605 Semarang
Tlpn. (024) 6723456

Fax . 024-6710144

Email: penerbit_ypat@stekom.ac.id

Distributor Tunggal:

UNIVERSITAS STEKOM
Jln Majapahit No 605 Semarang
Tlpn. (024) 6723456
Fax . 024-6710144

Email: info@stekom.ac.id

KATA PENGANTAR

Salam sejahtera untuk para pembaca yang terhormat,

Saya dengan sukacita mempersembahkan buku ini kepada Anda tentang topik yang menarik dan penting dalam dunia pengembangan web, yaitu "Belajar Laravel". Dalam tulisan ini, saya akan memperkenalkan konsep dan manfaat dari framework web Laravel, serta tujuan yang ingin dicapai melalui pemahaman yang mendalam tentang framework ini.

Tujuan dari artikel ini adalah memberikan pemahaman yang komprehensif tentang Laravel kepada para pembaca. Saya berharap dapat membantu Anda memahami secara mendalam tentang framework ini, dari konsep dasar hingga fitur-fitur yang canggih. Dengan mempelajari Laravel, Anda akan memiliki keahlian yang kuat dalam mengembangkan aplikasi web modern yang efisien dan aman.

Manfaat dari pembelajaran Laravel sangatlah signifikan. Pertama, Laravel merupakan salah satu framework web yang paling populer dan kuat saat ini. Dengan mempelajarinya, Anda akan meningkatkan peluang karir Anda sebagai pengembang web, karena permintaan akan keahlian Laravel terus meningkat di industri.

Selain itu, Laravel menawarkan kecepatan dan efisiensi dalam pengembangan aplikasi web. Dengan fitur-fitur yang canggih seperti routing yang kuat, ORM (Object-Relational Mapping) yang terintegrasi, serta sistem template yang mudah digunakan, Anda dapat mengembangkan aplikasi dengan lebih cepat dan efektif.

Selanjutnya, menggunakan Laravel juga memberikan keuntungan dalam hal keamanan. Framework ini menyediakan mekanisme perlindungan terhadap ancaman umum seperti serangan XSS (Cross-Site Scripting) dan CSRF (Cross-Site Request Forgery), sehingga Anda dapat membangun aplikasi yang lebih aman dan terpercaya.

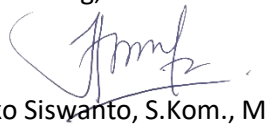
Melalui buku ini, penulis berharap dapat memberikan wawasan yang berharga kepada Anda tentang potensi dan manfaat belajar Laravel. penulis akan menyajikan materi ini dengan bahasa yang formal dan akurat, serta dengan penekanan pada informasi yang relevan dan terkini.

Dengan semangat dan tekad yang kuat, saya yakin Anda akan mendapatkan manfaat besar dari belajar Laravel. Mari kita mulai perjalanan kita dalam dunia pengembangan web yang menarik ini dan membuka peluang baru yang menanti kita.

Penulis menyakini bahwa dalam pembuatan Buku Belajar Laravel ini masih jauh dari sempurna. Oleh karena itu penulis mengharapkan kritik dan saran yang membangun guna penyempurnaan Buku Belajar Laravel ini dimasa yang akan datang.

Akhir kata, penulis mengucapkan terima kasih sebanyak – banyaknya kepada semua pihak yang telah membantu baik secara langsung maupun tidak langsung, Secara Moril Maupun Materiil.

Semarang,21 Juni 2023



Eko Siswanto, S.Kom., M.Kom
Penulis

DAFTAR ISI

BAB 1
MENGENAL LARAVEL

1. Pengertian Framework

Framework merupakan sebuah alat yang berupa pola kerja yang digunakan dalam pengembangan sebuah website. Framework dibuat untuk membantu seorang pembuat website dalam menuliskan sebuah baris kode. Dengan penggunaan framework seorang pembuat program akan menjadi lebih mudah, lebih cepat, dan terstruktur serta rapi dalam menuliskan kode.

Dalam dunia pemrograman, framework adalah pilar yang kokoh yang membantu para pengembang mewujudkan ide-ide mereka dengan lebih efisien dan efektif. Dalam tulisan ini, kita akan menjelajahi gambaran umum tentang keuntungan yang luar biasa dari menggunakan sebuah [Framework] dan memberikan tips yang membantu untuk memulai perjalanan pengembangan dengan semangat dan peralatan yang diperlukan.

1. Efisiensi dalam Pengembangan: Menggunakan [Framework] membawa keuntungan utama berupa efisiensi dalam pengembangan perangkat lunak. Dengan struktur yang telah ditentukan dan modul-modul yang tersedia, pengembang dapat menghemat waktu berharga dalam membangun fitur-fitur yang umum digunakan, sehingga dapat lebih fokus pada pengembangan fitur khusus dan inovasi yang lebih mendalam. Ini menginspirasi semangat untuk menciptakan dengan kecepatan yang lebih tinggi dan memberikan solusi yang lebih baik.
2. Konsistensi dan Standar: Framework menyediakan kerangka kerja yang konsisten dan standar dalam pengembangan perangkat lunak. Ini membantu pengembang dan tim untuk bekerja secara lebih terstruktur dan saling bekerja sama dengan lebih baik. Dengan menggunakan konvensi dan aturan yang telah ditetapkan oleh [Framework], pengembang dapat menghasilkan kode yang mudah dibaca dan dipelihara. Hal ini mendorong semangat kolaborasi dan memungkinkan pemahaman yang lebih baik antara pengembang-pengembang yang bekerja dalam proyek yang sama.
3. Skalabilitas dan Fleksibilitas: [Framework] menawarkan skalabilitas yang kuat, sehingga memungkinkan aplikasi untuk berkembang seiring waktu. Dengan struktur yang sudah ada, pengembang dapat dengan mudah menambahkan fitur baru, memperbaiki bug, atau mengatasi tantangan skalabilitas tanpa perlu mengubah seluruh struktur aplikasi. Fleksibilitas ini membangkitkan semangat inovasi dan memberikan kesempatan bagi pengembang untuk mengeksplorasi potensi tanpa batas.
4. Komunitas yang Solid: Satu keuntungan besar dalam menggunakan [Framework] adalah adanya komunitas yang solid di sekitarnya. Komunitas ini terdiri dari pengembang-pengembang berbakat yang berbagi pengetahuan, pengalaman, dan sumber daya yang sangat berharga. Dalam komunitas ini, pengembang dapat saling mendukung, bertanya, dan berbagi ide-ide. Ini menciptakan semangat kebersamaan dan mempercepat pertumbuhan dan pemahaman di dalam dunia pengembangan.

Tips Memulai:

- Mulailah dengan mempelajari dasar-dasar [Framework]. Lakukan riset, ikuti tutorial, dan baca dokumentasi yang disediakan oleh pengembang [Framework].
- Bergabunglah dengan komunitas pengembang [Framework] yang ada, baik melalui forum online, grup diskusi, atau acara konferensi. Ini akan membantu Anda menjalin hubungan dan belajar dari pengalaman orang lain.
- Praktikkan apa yang telah Anda pelajari dengan membuat proyek sederhana menggunakan [Framework]. Praktik ini akan memperkuat pemahaman Anda dan memberikan Anda kepercayaan diri untuk mengembangkan proyek yang lebih besar.
- Jangan takut untuk mencoba dan bereksperimen. Kesalahan adalah bagian dari proses pembelajaran, jadi berani untuk mencoba hal-hal baru dan berinovasi.

Dalam perjalanan pengembangan perangkat lunak dengan [Framework], semangat dan peralatan yang diperlukan adalah kunci utama untuk meraih kesuksesan. Dengan penerapan tips yang membantu ini dan semangat yang menggebu-gebu, Anda akan menemukan diri Anda menguasai [Framework] dengan mahir dan mencapai potensi yang luar biasa dalam pengembangan perangkat lunak.

Fungsi Framework

Kerangka kerja / Framework itu sendiri memiliki sebuah fungsi bawaan untuk membantu pengembang web dalam membuat sebuah situs website. Ada fungsi tambahan dari kerangka kerja. Lihat di bawah untuk ini :

1. Membuat skrip / kode program menjadi lebih terstruktur dan baik
   Framework sering kali memiliki tata letak arsitektur saat menulis skrip atau potongan kode. Hasilnya, kode yang ditulis lebih mudah dipahami dan lebih terstruktur. Akibatnya, Anda dapat dengan cepat mengidentifikasi bug atau masalah apa pun dan kemudian memperbaikinya dengan cepat.
   Selain membuat kode lebih baik rapi, framework juga dapat digunakan untuk meningkatkan keamanan website yang kita buat. Seperti pada contohnya framework Laravel yang sudah mengadopsi berbagai sistem keamanan seperti autentikasi, enkripsi, dan hashing.
2. Mempercepat pembuatan website
   Framework yang dimaksud dapat mempercepat pengembangan website. Ini karena pengembang dapat menggunakan komponen yang sudah ada sebelumnya tanpa harus menulis kode baru dari awal, memungkinkan mereka mempercepat pembuatan situs web.
3. Pemeliharaan dan perawatan website lebih mudah
   Kerangka kerja yang dimaksud dapat memudahkan Anda untuk membangun dan memelihara situs web Anda. Karena sebagian besar kerangka kerja sudah menggunakan berbagai gaya arsitektur, perbaikan bug, peningkatan fitur, dan pemeliharaan situs web akan lebih mudah.

Jenis-Jenis Framework

Ada berbagai jenis kerangka kerja yang biasa digunakan untuk membangun situs web. Setiap framework memiliki fiturnya masing-masing dan juga menggunakan bahasa pemrograman yang berbeda- beda. Di bawah ini adalah jenis kerangka kerja yang biasa digunakan untuk membangun situs web.

1. Framework CSS

   Dalam dunia desain web modern, [Framework CSS] telah menjadi pilihan yang sangat populer untuk membangun tampilan yang menakjubkan dan responsif. Dalam artikel ini, kita akan menjelajahi gambaran umum tentang keuntungan yang luar biasa dari menggunakan [Framework CSS] dan memberikan tips yang membantu untuk memulai perjalanan desain dengan semangat dan peralatan yang diperlukan.

   1. Efisiensi dalam Pengembangan: Salah satu keuntungan utama menggunakan [Framework CSS] adalah efisiensi yang ditawarkannya dalam pengembangan desain web. Dengan menggunakan komponen-komponen yang telah siap pakai dan gaya yang sudah ditentukan, Anda dapat menghemat waktu berharga dalam membangun tampilan dari awal. Ini memberikan semangat untuk menciptakan dengan kecepatan yang lebih tinggi dan memberikan hasil desain yang lebih baik.
   2. Konsistensi dan Standar: [Framework CSS] menyediakan kerangka kerja yang konsisten dan standar dalam desain web. Dengan gaya yang telah ditetapkan, Anda dapat mencapai konsistensi yang tinggi dalam tampilan dan pengalaman pengguna di seluruh situs web Anda. Ini membantu membangun merek yang kuat dan memberikan kesan profesional yang konsisten kepada pengunjung. Dengan menggunakan standar industri yang telah ditetapkan, Anda juga dapat berkolaborasi dengan pengembang lain dengan lebih mudah.
   3. Responsif dan Fleksibel: [Framework CSS] dirancang dengan perhatian khusus terhadap responsivitas, sehingga memungkinkan desain Anda untuk terlihat indah di berbagai perangkat dan ukuran layar. Ini memberikan fleksibilitas yang luar biasa dalam menghadapi tantangan desain responsif yang kompleks. Anda dapat dengan mudah mengatur tata letak, grid, dan komponen lainnya untuk menyesuaikan dengan kebutuhan perangkat yang berbeda. Fleksibilitas ini memberikan semangat eksplorasi dan menginspirasi Anda untuk menciptakan tampilan yang luar biasa di setiap perangkat.
   4. Komunitas yang Solid: Salah satu keuntungan terbesar menggunakan [Framework CSS] adalah adanya komunitas yang solid di sekitarnya. Komunitas ini terdiri dari desainer dan pengembang yang berbagi pengetahuan, pengalaman, dan sumber daya yang sangat berharga. Anda dapat bergabung dengan forum online, grup diskusi, atau acara konferensi untuk terhubung dengan komunitas ini. Dalam komunitas ini, Anda dapat mendapatkan dukungan, bertanya pertanyaan, dan berbagi inspirasi. Ini menciptakan semangat kolaborasi dan mempercepat pertumbuhan dan pemahaman dalam dunia desain web.

   Tips Memulai :

- Mulailah dengan mempelajari dasar-dasar [Framework CSS] yang Anda pilih. Luangkan waktu untuk membaca dokumentasi resmi dan mengikuti tutorial online yang tersedia.
- Cobalah untuk membuat proyek sederhana menggunakan [Framework CSS]. Praktik ini akan memperkuat pemahaman Anda dan memberikan Anda kepercayaan diri untuk mengembangkan proyek yang lebih besar.
- Jangan takut untuk mencoba hal baru. Eksperimen dengan fitur-fitur yang disediakan oleh [Framework CSS] untuk menciptakan tampilan yang unik dan inovatif.
- Manfaatkan sumber daya online dan komunitas untuk belajar dan tumbuh bersama. Berbagi pengetahuan dengan orang lain dan terus mencari inspirasi dari desainer yang berbakat.

Dalam perjalanan desain web dengan [Framework CSS], semangat dan peralatan yang diperlukan adalah kunci utama untuk meraih kesuksesan. Dengan menerapkan tips yang membantu ini dan semangat yang menggebu-gebu, Anda akan menemukan diri Anda menguasai [Framework CSS] dengan mahir dan mencapai potensi desain yang luar biasa.

CSS merupakan sebuah bahasa pemograman yang digunakan untuk mengatur tampilan atau layout pada sebuah website. Ada beberapa jenis CSS yang sering digunakan oleh paara pembuat webiste adalah sebaagai berikut :

- Bootstrap

  Boostrap merrupakan sebuah framework css yang dinamis dan modern. Fitur yang menjadi andalan boostrap adalah tampilanya yang resposive, sehingga memudahkan user dalam melihat sebuah website.

  CMS Bootstrap memungkinkan pengguna untuk membuat dan mengelola situs web dengan cepat dan mudah, dengan menggunakan Bootstrap sebagai dasar tata letak dan komponen antarmuka. Ini menyediakan serangkaian templat siap pakai, gaya desain, dan komponen yang dapat disesuaikan untuk mempercepat pengembangan situs web.

  Keuntungan utama CMS Bootstrap adalah sebagai berikut:

  1. Responsif: Bootstrap dirancang untuk memberikan responsifitas yang baik pada berbagai perangkat dan ukuran layar. Dengan CMS Bootstrap, pengguna dapat dengan mudah membuat situs web yang menyesuaikan diri dengan perangkat pengguna, seperti desktop, tablet, atau ponsel.
  2. Desain yang Menarik: Bootstrap menyediakan sejumlah gaya desain dan komponen antarmuka yang siap pakai, seperti tombol, formulir, jumbotron, navigasi, dll. Dengan CMS Bootstrap, pengguna dapat memanfaatkan komponen ini dan mengkustomisasinya sesuai kebutuhan mereka, sehingga menciptakan situs web yang menarik secara visual.
  3. Pengembangan Cepat: Dengan menggunakan CMS Bootstrap, pengguna dapat memanfaatkan templat dan komponen siap pakai yang telah disediakan. Ini memungkinkan pengembangan situs web yang cepat dan efisien, menghemat waktu dan upaya dalam proses pengembangan.

4. Konsistensi: Bootstrap mengikuti prinsip desain yang konsisten. Dengan menggunakan CMS Bootstrap, pengguna dapat memastikan bahwa situs web mereka memiliki tampilan dan nuansa yang seragam di seluruh halaman, menciptakan pengalaman pengguna yang mulus.
5. Komunitas dan Dukungan: Bootstrap memiliki komunitas yang besar dan aktif, dengan dokumentasi yang baik dan dukungan yang luas. CMS Bootstrap memanfaatkan keuntungan ini dengan menyediakan sumber daya, tutorial, dan forum diskusi yang dapat membantu pengguna dalam mempelajari dan mengatasi masalah terkait penggunaan Bootstrap.

Dalam kesimpulannya, CMS Bootstrap adalah integrasi antara Bootstrap, framework front-end yang populer, dengan sistem manajemen konten. Ini memungkinkan pengguna untuk dengan mudah membuat dan mengelola situs web yang responsif dan menarik secara visual, dengan memanfaatkan templat, gaya desain, dan komponen siap pakai yang disediakan oleh Bootstrap.

- Semantic UI

Semantic UI memiliki kelebihan yaitu dalam penulisan class yang lebih mudah dibanding dengan css yang lain. Semantic UI juga menyediakan banyak sekali komponen yang dapat dimanfaatkan untuk tampilan HTML agar terlihat lebing menarik.

CMS Semantic UI adalah suatu sistem manajemen konten (CMS) yang didasarkan pada kerangka kerja pengembangan antarmuka pengguna (UI) bernama Semantic UI. CMS ini dirancang untuk memudahkan pembuatan dan pengelolaan situs web dengan antarmuka yang responsif dan menarik.
Semantic UI sendiri adalah kerangka kerja CSS dan JavaScript yang dibangun dengan menggunakan prinsip desain semantik. Prinsip ini berfokus pada penggunaan kode yang jelas dan bermakna, sehingga memungkinkan pengembang dan desainer web untuk dengan mudah memahami dan mengelola komponen-komponen dalam tata letak situs web.

Berikut adalah beberapa fitur dan konsep kunci dari CMS Semantic UI:

1. Desain Semantik: Semantic UI menggunakan kelas-kelas CSS yang bermakna dan mudah dimengerti, yang memungkinkan pengembang untuk dengan mudah menggambarkan struktur dan makna elemen-elemen dalam kode mereka. Ini membuat pengembangan dan pemeliharaan situs web menjadi lebih intuitif dan efisien.
2. Responsif: CMS Semantic UI memastikan bahwa situs web yang dibuat menggunakan platform ini secara otomatis responsif, artinya mereka akan beradaptasi dan terlihat baik di berbagai perangkat dan ukuran layar. Ini sangat penting dalam era mobile-friendly, di mana pengguna mengakses konten web melalui berbagai perangkat seperti smartphone dan tablet.
3. Koleksi Komponen: Semantic UI menyediakan koleksi komponen yang lengkap dan konsisten, seperti tombol, formulir, menu, grid, dan banyak lagi. Komponen

ini telah dirancang dengan baik dan dapat dengan mudah digunakan dalam membangun tata letak dan interaksi situs web.

4. Responsif Grid System: Semantic UI memiliki sistem grid responsif yang fleksibel, yang memungkinkan pengembang untuk dengan mudah mengatur tata letak situs web secara responsif. Dengan menggunakan grid system ini, pengembang dapat membuat tata letak yang responsif dan terstruktur dengan mudah.
5. Customizable: CMS Semantic UI memungkinkan pengguna untuk menyesuaikan tampilan dan gaya situs web mereka sesuai dengan kebutuhan dan preferensi mereka. Ini dapat dilakukan dengan mudah melalui konfigurasi dan pengaturan yang disediakan oleh CMS tersebut.
6. Integrasi: CMS Semantic UI dapat diintegrasikan dengan berbagai sistem dan teknologi lainnya, seperti sistem manajemen konten lain, basis data, atau bahkan alat pengembangan yang disesuaikan. Ini memberikan fleksibilitas dalam penggunaan CMS ini dalam berbagai proyek.
7. Dokumentasi Lengkap: CMS Semantic UI menyediakan dokumentasi yang kaya dan komprehensif, yang menjelaskan penggunaan dan implementasi setiap komponen dan fitur yang tersedia. Hal ini memudahkan pengguna baru untuk mempelajari dan memahami cara menggunakan CMS ini.

Dalam keseluruhan, CMS Semantic UI adalah platform yang kuat untuk membangun dan mengelola situs web yang menarik, responsif, dan sesuai dengan prinsip desain semantik. Dengan koleksi komponen yang lengkap, desain yang responsif, dan fleksobel.

- Materialize

Materialize adalah sebuah framework front-end CSS yang dikembangkan dengan desain responsif dan modern. Framework ini didesain untuk memudahkan pengembangan situs web yang responsif dan menarik dengan menggunakan prinsip-prinsip Material Design, yang merupakan gaya desain yang dikembangkan oleh Google.

Berikut adalah beberapa poin penting yang perlu Anda ketahui tentang Materialize CMS:

1. Responsif dan Mobile-friendly: Materialize didesain untuk memberikan tampilan yang baik dan responsif pada berbagai perangkat, mulai dari desktop hingga perangkat mobile. Dengan menggunakan Materialize, Anda dapat membuat situs web yang terlihat bagus dan berfungsi dengan baik di berbagai ukuran layar.
2. Komponen Siap Pakai: Materialize menyediakan berbagai komponen dan elemen UI yang dapat digunakan untuk membangun tampilan situs web Anda. Mulai dari tombol, kotak teks, kartu, navigasi, ikon, form, slider, dan masih banyak lagi. Komponen-komponen ini telah dirancang dengan gaya Material Design, sehingga memberikan tampilan modern dan menarik pada situs web Anda.
3. Fitur Animasi dan Efek Visual: Materialize menyertakan animasi dan efek visual yang dapat meningkatkan pengalaman pengguna pada situs web Anda.

Misalnya, Anda dapat dengan mudah menambahkan transisi animasi saat elemen muncul atau bergerak pada halaman, memberikan kesan yang halus dan menarik bagi pengunjung.

4. Grid System: Materialize memiliki sistem grid yang fleksibel, yang memungkinkan Anda untuk dengan mudah mengatur tata letak halaman situs web Anda. Dengan menggunakan grid system ini, Anda dapat dengan cepat menentukan ukuran dan posisi elemen-elemen pada halaman Anda.
5. Pustaka JavaScript: Materialize juga menyertakan pustaka JavaScript yang kuat untuk meningkatkan interaktivitas situs web Anda. Pustaka ini mencakup fungsi-fungsi seperti modals, dropdowns, parallax scrolling, form validation, dan banyak lagi. Anda dapat menggunakan pustaka ini untuk menambahkan fitur-fitur dinamis pada situs web Anda tanpa harus menulis kode JavaScript dari awal.
6. Integrasi dengan Sistem Backend: Materialize merupakan sebuah framework front-end, yang berarti fokus utamanya adalah pada tampilan dan interaksi di sisi klien. Namun, Anda dapat dengan mudah mengintegrasikan Materialize dengan sistem backend atau CMS lainnya, seperti WordPress atau Drupal, untuk mengelola konten situs web Anda.

Materialize menyediakan dokumentasi yang lengkap dan contoh penggunaan untuk memudahkan pengembang dalam mempelajari dan menggunakan framework ini. Selain itu, tersedia juga tema dan plugin pihak ketiga yang dibangun di atas Materialize, yang dapat membantu dalam memperluas fungsionalitas dan tampilan situs web Anda.

Dalam kesimpulannya, Materialize adalah sebuah framework front-end CSS yang memungkinkan Anda untuk dengan cepat membangun situs web yang responsif dan menarik dengan menggunakan prinsip-prinsip Material Design. Dengan komponen siap pakai, sistem grid yang fleksibel, efek visual, dan pustaka JavaScript yang kuat, Materialize menjadi pilihan yang baik untuk pengembangan situs web modern.

2. Framework Javascript

Dalam dunia pengembangan web modern, [Framework JavaScript] telah menjadi pilihan yang sangat populer untuk membangun aplikasi web yang interaktif dan canggih. Dalam artikel ini, kita akan menjelajahi gambaran umum tentang keuntungan yang luar biasa dari menggunakan [Framework JavaScript] dan memberikan tips yang membantu untuk memulai perjalanan pengembangan dengan semangat dan peralatan yang diperlukan.

1. Produktivitas yang Meningkat: Salah satu keuntungan utama menggunakan [Framework JavaScript] adalah peningkatan produktivitas yang ditawarkannya. Dengan fitur-fitur yang telah siap pakai dan konvensi yang terstandarisasi, Anda dapat menghemat waktu berharga dalam mengembangkan aplikasi web. [Framework JavaScript] memberikan alat yang diperlukan untuk membangun tampilan yang kompleks dengan lebih mudah dan cepat. Ini memberikan semangat untuk menciptakan dengan kecepatan yang lebih tinggi dan memberikan solusi yang lebih baik.

2. Kode yang Terstruktur dan Mudah Dikelola: [Framework JavaScript] memberikan struktur yang jelas dan konsisten dalam pengembangan aplikasi web. Dengan menggunakan [Framework JavaScript], Anda dapat mengatur kode dengan cara yang terstruktur, memisahkan tugas-tugas menjadi komponen yang terpisah, dan memanfaatkan prinsip-prinsip desain yang baik. Hal ini membuat kode Anda lebih mudah dibaca, dipelihara, dan ditingkatkan. Keuntungan ini memberikan semangat untuk mengembangkan aplikasi yang terorganisir dan mempermudah kolaborasi dengan tim pengembangan.
3. Komunitas yang Aktif: Salah satu keuntungan besar menggunakan [Framework JavaScript] adalah adanya komunitas yang aktif di sekitarnya. Komunitas ini terdiri dari pengembang-pengembang berbakat yang berbagi pengetahuan, pengalaman, dan sumber daya yang sangat berharga. Dalam komunitas ini, Anda dapat mendapatkan dukungan, bertanya pertanyaan, dan berbagi ide-ide. Hal ini menciptakan semangat kebersamaan dan mempercepat pertumbuhan dan pemahaman dalam dunia pengembangan.
4. Skalabilitas dan Fleksibilitas: [Framework JavaScript] dirancang untuk menghadapi tantangan skalabilitas dan fleksibilitas dalam pengembangan aplikasi web. Dengan arsitektur yang tangguh, Anda dapat dengan mudah menambahkan fitur baru, mengintegrasikan dengan API pihak ketiga, atau mengoptimalkan performa aplikasi. Fleksibilitas ini memberikan semangat eksplorasi dan menginspirasi Anda untuk menciptakan aplikasi web yang inovatif dan dapat berkembang seiring waktu.

Tips Memulai:

- Mulailah dengan mempelajari dasar-dasar [Framework JavaScript] yang Anda pilih. Bacalah dokumentasi resmi dan ikuti tutorial yang tersedia untuk memahami konsep dan fitur utama.
- Praktikkan apa yang telah Anda pelajari dengan membuat proyek sederhana menggunakan [Framework JavaScript]. Praktik ini akan memperkuat pemahaman Anda dan memberikan Anda kepercayaan diri untuk mengembangkan proyek yang lebih kompleks.
- Terlibatlah dalam komunitas pengembang [Framework JavaScript]. Bergabunglah dengan forum online, grup diskusi, dan acara konferensi untuk belajar dari pengalaman orang lain, bertanya pertanyaan, dan berbagi pengetahuan.

Dalam perjalanan pengembangan aplikasi web dengan [Framework JavaScript], semangat dan peralatan yang diperlukan adalah kunci utama untuk meraih kesuksesan. Dengan menerapkan tips yang membantu ini dan semangat yang menggebu-gebu, Anda akan menemukan diri Anda menguasai [Framework JavaScript] dengan mahir dan mencapai potensi pengembangan yang luar biasa.

JavaScript merupakan bahasa pemrograman yang sering digunakan oleh para *front end developer* dalam pembuatan tampilan website menjadi lebih interaktif. JavaScript merupakan bahasa pemrograman yang berjalan pada sisi *front end* dan masuk ke dalam golongan bahasa pemrograman tingkat tinggi. Berikut ini merupakan beberapa framework JavaScript yang sering digunakan.

- AngularJS
  Framework AngularJS ini bersifat *open source* dan ia berjalan pada sisi *client* dengan menggunakan konsep MVC untuk membuat tampilan dari website menjadi lebih dinamis.

- ReactJS
  ReactJS adalah framework yang dikembangkan oleh Facebook. Ia sering digunakan untuk membuat UI untuk suatu website atau aplikasi mobile karena ReactJS ini dapat digunakan secara *multi platform*. ReactJS ini merupakan inti dari React Native yang dapat memungkinkan kamu untuk mengembangkan aplikasi mobile di dua sistem operasi secara bersamaan, yaitu Android dan iOS.

3. Framework PHP
   Hypertext Preprocessor atau yang dikenal dengan istilah PHP merupakan bahasa pemrogramann yang dipergunakan untuk berkomunikasi dari sisi server (server side) . Berikut ini merupakan beberapa framework PHP yang populer digunakan dalam pembuatan sebuah website.

   Ada banyak framework PHP yang tersedia untuk pengembangan aplikasi web. Berikut ini beberapa contoh framework PHP populer:

   - CodeIgniter
     Yang pertama ada CodeIgnitera atau yang biasa dikenal dengan Singkatan CI ini merupakan sebuah framework dengan arsitektur yang memiliki sebuah cirikhas, yaitu dengan amengusung arsitektur MVC (*Model, View, controller*). Dengan begitu kamu dapat menulis kode dengan lebih terstruktur dan spesifik sehingga website yang kita buat akan terlihat jauh lebih rapih.

   - Laravel
     Yang kedua adalah Laravel, Framework laravel ini biasanya dipakai oleh para para pengembang website untuk membuat sebuah website dengan kompleksitass yang sangat tinggi. Ia memiliki banyak sekali pustaka (library) yang lengkap serta memilih sintaksis atau kode yang elegan, ringkas, dan rapih.
     Laravel adalah sebuah framework aplikasi web berbasis PHP yang dirancang untuk memudahkan pengembangan aplikasi web dengan menyediakan berbagai fitur dan alat yang kuat. Framework ini dikembangkan dengan tujuan utama untuk menyederhanakan tugas-tugas umum dalam pengembangan web seperti routing, caching, dan autentikasi, sehingga para pengembang dapat fokus pada logika bisnis inti aplikasi mereka.

Salah satu keunggulan utama Laravel adalah kemampuannya dalam mempermudah pengelolaan basis data. Laravel menyediakan ORM (Object-Relational Mapping) yang disebut Eloquent, yang memungkinkan pengembang untuk berinteraksi dengan basis data menggunakan sintaks yang sederhana dan intuitif, tanpa perlu menulis kueri SQL secara langsung. Selain itu, Laravel juga menyediakan sistem migrasi yang memudahkan dalam mengelola skema basis data.

- Symfony
  Yang ketiga ada Symfony, Framework ini sangat baik digunakan untuk mengembangkan sebuah website dengan skala yang sangat besar dan komplekss. Framework ini juga memakan menggunakan sedikit memori jika dibandingkan dengan framework lainnya dan menghasilkan performa yang jauh lebih baik dibandingkan yang lain.
- Zend Framework: Zend Framework adalah framework PHP yang kuat dan modular. Framework ini menyediakan berbagai komponen yang dapat digunakan secara independen atau dalam kombinasi untuk membangun aplikasi web. Zend Framework menekankan fleksibilitas dan skalabilitas, serta mendukung berbagai standar industri.
- Yii: Yii adalah framework PHP yang cepat dan efisien. Framework ini dirancang untuk mempercepat proses pengembangan dengan menyediakan fitur-fitur seperti pembangkitan kode otomatis, caching, pengelolaan akses, dan pengujian unit yang mudah. Yii juga memiliki performa yang baik dan mendukung pembuatan aplikasi berbasis RESTful API.
- CakePHP: CakePHP adalah framework PHP yang mudah dipelajari dan menggunakan pendekatan konvensi daripada konfigurasi. Framework ini menawarkan fitur-fitur seperti routing, validasi form, ORM, dan caching. CakePHP juga mengikuti prinsip MVC untuk memisahkan logika aplikasi dan tampilan.
- Slim: Slim adalah framework PHP yang ringan dan dioptimalkan untuk pembuatan RESTful API dan aplikasi web kecil. Slim menyediakan fitur-fitur dasar seperti routing, penanganan request dan response, dan container dependensi. Framework ini sangat cocok untuk membuat aplikasi web dengan fokus pada performa dan kecilnya ukuran.

  Pilihan framework PHP tergantung pada kebutuhan dan preferensi Anda dalam pengembangan aplikasi web. Setiap framework memiliki kelebihan dan kelemahan sendiri, jadi penting untuk mengevaluasi fitur-fitur yang ditawarkan dan melihat apakah sesuai dengan kebutuhan proyek Anda.

Buku ini akan membahas lebih dalam tentang framework Laravel , khusunya Laravel versi terbaru yaitu versi 9. Pada buki ini kita akan mempelajari dasar - dasar Laravel dengan secara detail dan belajar membuat sebuah project atau aplikasi sederhana menggunakan Framework Laravel.

2. Sejarah Laravel

Sejarah Awalmula Laravel dibuat oleh **Taylor Otwell**. Laravel diciptakan oleh Taylor Otwell untuk memberikan sebuah alternatif yang lebih baik dari Framework PHP yang yang lain yang sudah pernah dikembangkan sebelumnya seperti Codeigniter, YII dan lainya.

Framework Laravel ini di perkenalkana pertama kali pada tanggal 09 juni tahun 2011 dengana versi bheta. Masih berada dibulan yang sama Laraveel merilis versi pertamanya yaitu Laravel versi 1, pada Laravel 1 ini sudah dibekali dengan berbagai fitur diantaranya

- ✓ authentication,
- ✓ localisation,
- ✓ models,
- ✓ views,
- ✓ sessions,
- ✓ routing dan fitur-fitur lainya.

tetapi pada frameworl Laravel 1 ini masih kurang mendukung untuk controller. Pada versi ini Laravel belum menggunakan konsep MVC.

Berikut data sejarah release framework Laravel dalam bentuk tabel :

Versi	Release	Perbaikan Bug	Perbaikan Keamanan
V1	Juni 2011	–	–
V2	September 2011	–	–
v3	Februari 2012	–	–
v4	Mai 2013	–	–
5.0	04 Februari 2015	04 Augustus 2015	04 Februari 2016
5.1 (LTS)	09 Juni 2015	09 Juni 2017	09 Juni 2018
5.2	21 Desember 2015	21 Juni 2016	21 Desember 2016
5.3	23 Augustus 2016	23 Februari 2017	23 Augustus 2017
5.4	24 Januari 2017	24 Juli 2017	24 Januari 2018
5.5 (LTS)	30 Augustus 2017	30 Augustus 2019	30 Augustus 2020
5.6	7 Februari 2018	7 Augustus 2018	7 Februari 2019
5.7	4 Sepember 2018	4 Februari 2019	4 Sepember 2019
5.8	26 Februari 2019	26 Augustus 2019	26 Februari 2020
6.0 (LTS)	3 Sepember 2019	3 Sepember 2021	3 Sepember 2022
7	3 Maret 2020	3 September 2020	3 Maret 2021
8	8 September 2020	26 Juli 2022	24 Januari 2023
9	8 Februari 2022	8 Agustus 2023	8 Februari 2023

Penegertian Framework Laravel

Laravel merupakan framework PHP yang sangat populer saat ini. sama seperti halnya codeigniter jika teman-teman sudah pernah mendengar tentang codeigniter, laravel dan codeigniter sama-sama merupakan framework atau kerangka kerja PHP yang dibuat untuk mempermudah para developer atau programmer dalam membangun sistem/aplikasi yang berukuran kecil, bahkan sampai sekala yang besar.

Bagi para programmer website yang ingin berfokus ke back-end PHP (atau memilih jalur full stack), Laravel bisa menjadi salah satu materi yang wajib dikuasai. Mayoritas lowongan kerja webprogrammer back-end di Indonesia yang mewajibkan paham sampai ke framework, yang biasanya salah satu dari **Code Igniter** atau **Laravel**.

Laravel memili beberapa kelebihan dibandingkan dengan framework lain, diantaranya :

1. Dokumentasi Sangat Lengkap

   Sebuah Platform bisa dikatakan baik adalah ketika sebuah platform itu mudah digunakan dan memiliki sebuah dokumentasi yang lengkap serta jelas. Karena jika kita belum begitu hebat dalam bidangs web development, akan tetapii memiliki pemahaman PHP yang sangtat baik. Maka kita akan dengan mudah mempelajari Laravel hanya dengan membaca dokumentasinya saja, sebab dokumentasi yang dibuat oleh Laravel tergolong kedalam dokumentasi yang sangat sangat baik, rapih, mudah dan jelas. Kita bisa melihat dan memulai belajar dihalaman resmi laravel https://laravel.com/
2. Tersedianya Sebuah Forum dan Komunitas

   Sebelumkita memilih sebuah framework apa yang akan kita gunakan untuk membuat sebuah project web , hal yang cukup penting untuk dipertimbangkan adalah adanya sebuah forums dan komunitas. Sebab dengan adanya forum dan komunitas kita dapat mudah belajar dan mencari solusi atas setiap permasalahan yang mungkin kita temukan dalam menggunakan frame work tersebut.
3. Fitur Lengkap

   Fitur yang ada dalam framework laravel sangat lengkap, sehingga memudahkan kita sebagai seorang develppoer web dalam membuat sebuah project.
4. Laravel Open Source

   Laravel adalah sebuah framework aplikasi web open source yang ditulis dalam bahasa pemrograman PHP. Sebagai proyek open source, artinya Laravel memiliki kode sumber yang dapat diakses, dimodifikasi, dan digunakan secara gratis oleh siapa pun.

   Open source memberikan banyak manfaat dan keunggulan dalam konteks pengembangan perangkat lunak. Berikut adalah beberapa penjelasan mengenai keistimewaan Laravel sebagai proyek open source:

   Fleksibilitas: Dengan menjadi proyek open source, Laravel memberikan fleksibilitas kepada pengembang untuk mengadaptasi dan mengubah framework sesuai dengan kebutuhan mereka. Pengguna dapat menambahkan atau menghapus fitur, memperbaiki bug, atau berkontribusi pada pengembangan framework tersebut.

   Transparansi: Karena Laravel bersifat open source, setiap orang memiliki akses terbuka ke kode sumbernya. Hal ini memungkinkan para pengguna untuk melihat, mempelajari, dan memahami bagaimana framework ini bekerja di balik layar. Transparansi ini memberikan kepercayaan kepada pengguna dan memungkinkan mereka untuk memverifikasi keamanan, stabilitas, dan kualitas framework.

Kolaborasi: Sebagai proyek open source, Laravel mendorong kolaborasi dan partisipasi komunitas. Pengguna dapat berkontribusi dalam pengembangan Laravel dengan menyumbangkan perbaikan bug, mengajukan fitur baru, atau memberikan masukan. Kolaborasi ini menghasilkan kemajuan dan inovasi yang lebih cepat, serta memungkinkan adanya diskusi dan pertukaran pengetahuan di antara anggota komunitas.

Dukungan komunitas: Sebagai framework open source yang populer, Laravel memiliki komunitas yang besar dan aktif. Komunitas ini menyediakan forum diskusi, grup pengguna, dan sumber daya online lainnya yang membantu pengguna dalam mempelajari, memecahkan masalah, dan berbagi pengalaman. Dukungan komunitas ini sangat berharga bagi pengembang yang baru memulai dengan Laravel atau yang ingin meningkatkan keterampilan mereka.

Pengembangan yang berkelanjutan: Dengan menjadi proyek open source, Laravel memiliki potensi untuk pengembangan yang berkelanjutan dan pembaruan yang konsisten. Tim pengembang Laravel secara teratur merilis versi baru dengan perbaikan bug, peningkatan kinerja, dan fitur-fitur baru yang mengikuti perkembangan teknologi web. Pengguna Laravel dapat dengan mudah memperbarui framework mereka ke versi terbaru dan mendapatkan manfaat dari perbaikan dan perbaikan yang dilakukan oleh komunitas.

Dalam kesimpulannya, sebagai proyek open source, Laravel menawarkan fleksibilitas, transparansi, kolaborasi, dukungan komunitas yang kuat, serta pengembangan yang berkelanjutan. Semua ini menjadikan Laravel sebagai pilihan yang populer bagi pengembang web untuk membangun aplikasi web yang kuat, efisien, dan dapat diandalkan.

5. Arsitektur MVC

   Selain dapat memudahkan pekerjaan kita dalam membuat website , arsitektur MVC juga dapat digunakan untuk meninsgkatkan performace website yang akan kita buat. Dengan menggunakan Arsitektur MVC dalam development kita dapat membuat struktur kode menjadi lebih rapih, dimana pola tersebut memisahkan antara logika dengan tampilan.

   Selain fitur-fitur yang kuat dalam pengelolaan basis data, Laravel juga menawarkan berbagai fitur lainnya seperti sistem routing yang fleksibel, manajemen sesi dan otentikasi, pembuatan formulir, sistem caching, dan lain-lain. Laravel juga mendukung pengembangan aplikasi dengan pola desain MVC (Model-View-Controller), yang memisahkan logika bisnis dari tampilan sehingga memudahkan dalam pemeliharaan dan pengembangan kode.

   Selain kelebihan-kelebihan teknisnya, Laravel juga memiliki komunitas yang besar dan aktif, dengan banyak sumber daya, dokumentasi, dan paket-paket ekstensi yang tersedia. Komunitas yang solid ini memberikan dukungan dan pengetahuan yang berharga bagi para pengembang Laravel, serta menjadikan Laravel sebagai salah satu framework PHP yang populer dan banyak digunakan.

   Dengan menggabungkan kekuatan fitur-fitur yang kuat, kemudahan penggunaan, dan dukungan komunitas yang luas, Laravel menjadi pilihan framework yang populer dan efisien untuk mengembangkan aplikasi web yang kuat, skalabel, dan mudah dipelihara.

BAB 2
MENGENAL MVC (MODEL, VIEW, CONTROL)

1. Pengertian MVC

MVC sendiri adsalah Sebuah Singkatan dari Model, View, Controller yang merupakan teknik pemograman yang memisahkan bisnis logic (alur pikir) , data logic (penyimpanan data) dan presentation logic Antarmuka atau secara sederhama memisahkan antara desain, data dan proses

MVC (Model-View-Controller) adalah sebuah desain arsitektur perangkat lunak yang digunakan untuk memisahkan komponen-komponen dalam sebuah aplikasi. Setiap komponen memiliki tanggung jawab dan peran yang berbeda dalam aplikasi.

Pada dasarnya fungsi utama MVC adalah mendukung proses pengembangan aplikasi *web* atau *mobile* menjadi lebih cepat. Kenapa bisa lebih cepat? Karena MVC memisahkan aplikasi menjadi tiga bagian, sehingga orang yang mengerjakannya pun lebih banyak. Pada bagian model dan *controller* biasanya akan ditangani oleh Back End Developer. Sementara itu, untuk bagian *view* akan dikerjakan oleh Front End Developer bersama tim dari UI/UX.

2. Konsep dasar MVC

Seperti yang sudah dijelaskan pada , MVC merupakan konsep yang dipisah menjadi tiga bagian, yaitu model, view, dan controller.

1. Model

Komponen Model mewakili struktur data atau informasi yang digunakan dalam aplikasi. Model berfungsi untuk mengelola logika bisnis, pengolahan data, dan interaksi dengan basis data. Model juga dapat mengirimkan notifikasi kepada komponen View dan Controller tentang perubahan pada data.

Dalam konteks pengembangan web dengan Laravel, istilah "model" mengacu pada komponen dari pola desain Model-View-Controller (MVC). Model mewakili lapisan aplikasi yang bertanggung jawab untuk mengelola logika bisnis dan akses ke data. Ini adalah bagian yang berhubungan langsung dengan basis data atau sumber daya eksternal lainnya.

Dalam Laravel, setiap model direpresentasikan oleh sebuah kelas PHP yang menggambarkan tabel atau koleksi data dalam basis data. Model-model ini biasanya ditempatkan di direktori **app/Models** dalam proyek Laravel. Setiap kelas model mewarisi fungsionalitas dari kelas **Illuminate\Database\Eloquent\Model**, yang disediakan oleh framework Laravel.

Berikut adalah contoh sederhana penjabaran model pada Laravel:

```php
namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class User extends Model
{
    protected $table = 'users';
    protected $primaryKey = 'id';
    public $timestamps = true;

    // Definisi relasi dengan model lain
    public function posts()
    {
        return $this->hasMany(Post::class);
    }

    // Definisi atribut yang dapat diisi massal
    protected $fillable = ['name', 'email', 'password'];

    // Definisi atribut yang harus tersembunyi
    protected $hidden = ['password'];

    // Metode lain, seperti metode aksesors dan mutators, dapat didefinisika
}
```

Dalam contoh di atas, terdapat kelas model **User** yang merepresentasikan tabel "users" dalam basis data. Properti **$table** digunakan untuk menentukan nama tabel yang terkait dengan model ini. Properti **$primaryKey** menentukan kolom kunci utama dari tabel.

Selain itu, model ini memiliki relasi dengan model lain, yaitu **Post**, yang didefinisikan dalam metode **posts()**. Metode ini mengindikasikan bahwa satu pengguna dapat memiliki banyak pos melalui relasi "hasMany".

Model ini juga menggunakan **$fillable** untuk menentukan atribut mana yang dapat diisi massal, dan **$hidden** untuk menentukan atribut mana yang harus disembunyikan ketika objek model diubah menjadi array atau JSON.

Selain properti dan metode yang ditunjukkan dalam contoh di atas, Anda juga dapat mendefinisikan metode aksesors dan mutators, metode query khusus, dan lainnya di dalam kelas model untuk mengelola logika bisnis atau interaksi dengan data.

2. View

Komponen View bertanggung jawab untuk menampilkan informasi kepada pengguna. View mengambil data dari Model dan menggambarkannya dalam bentuk tampilan yang sesuai. Tampilan dapat berupa halaman web, antarmuka pengguna, atau tampilan lainnya. View tidak melakukan pemrosesan data atau logika bisnis, tetapi hanya bertugas menampilkan informasi yang diberikan oleh Model.

Dalam pengembangan web dengan Laravel, "view" merujuk pada bagian dari pola desain Model-View-Controller (MVC) yang bertanggung jawab untuk menampilkan data kepada pengguna. View mengatur tampilan dan struktur halaman yang akan dikirimkan ke peramban (browser) pengguna. Dalam Laravel, view biasanya ditulis menggunakan sintaks template engine Blade yang kuat.

Berikut adalah penjabaran tentang view dalam Laravel:

1. Struktur Direktori:
    - File view Laravel umumnya ditempatkan di direktori **resources/views**.
    - Anda dapat membuat sub-direktori di dalam **views** untuk mengorganisir tampilan Anda sesuai dengan struktur proyek.
2. Membuat View:
    - Untuk membuat view baru, Anda dapat membuat file PHP baru dengan ekstensi **.blade.php** di direktori **views**.
    - Misalnya, jika Anda ingin membuat view untuk halaman beranda, Anda dapat membuat file **home.blade.php** di dalam direktori **views**.
3. Blade Templating:
    - Laravel menggunakan Blade sebagai mesin template default.
    - Blade menyediakan sintaks yang mudah digunakan untuk menggabungkan logika PHP dengan tampilan.
    - Anda dapat menggunakan fitur-fitur Blade seperti looping, kondisional, ekstensi layout, dan komponen untuk membuat tampilan yang fleksibel.
4. Memuat View:
    - Anda dapat menggunakan fungsi **view()** untuk memuat view dalam kontroler atau rute Laravel.
    - Misalnya, jika Anda ingin memuat view **home.blade.php**, Anda dapat menggunakan kode berikut:

```php
return view('home');
```

5. Variabel dalam View:
    - Anda dapat meneruskan variabel ke view untuk digunakan dalam tampilan.

- Misalnya, jika Anda ingin meneruskan variabel **$users** ke view, Anda dapat menggunakan kode berikut:

```php
$users = User::all();
return view('home', ['users' => $users]);
```

Dalam view, Anda dapat mengakses variabel **$users** seperti **$users**.

6. Blade Directives:
    - Blade menyediakan direktif-direktif yang kuat untuk memudahkan penggunaan logika PHP dalam tampilan.
    - Beberapa direktif yang umum digunakan antara lain: **@if**, **@foreach**, **@for**, **@while**, **@extends**, **@section**, **@yield**, **@include**, dan banyak lagi.
    - Direktif ini memungkinkan Anda untuk mengatur kondisi, perulangan, layout, dan penggunaan komponen secara mudah dalam tampilan.
7. Ekstensi Layout:
    - Dalam Blade, Anda dapat menggunakan ekstensi layout untuk mengatur struktur tampilan secara hierarkis.
    - Anda dapat mendefinisikan layout utama yang berisi area yang dapat diisi oleh konten dinamis dari view lain.
    - Dengan cara ini, Anda dapat membuat tampilan yang konsisten dan mengurangi duplikasi kode.

Itulah beberapa penjabaran tentang view dalam Laravel. Dengan Blade dan fitur-fiturnya yang kuat, Anda dapat membuat tampilan yang dinamis, fleksibel, dan mudah diatur dalam pengembangan web dengan Laravel.

3. Controller

Konsep terakhir dari bagian MVC adalah controller. Controller bertindak sebagi sebuah jembatan / penghubung data dan view kedalam sebuah struktur data didalam sebuah model. Tugasnya adalah menampung beberapa variabel yang akan ditampilkan pada View, Memanggil model untuk melakukan akses ke database, menyediakan penanganan bug/kesalahan, mengerjakan proses logika dari suatu aplikasi serta melakukan validasi atau mengecek sebuah input.

Komponen Controller bertindak sebagai perantara antara Model dan View. Controller menerima masukan dari pengguna melalui View dan mengubahnya menjadi aksi yang sesuai. Controller kemudian berinteraksi dengan Model untuk memperbarui data atau melakukan operasi bisnis yang diperlukan. Setelah itu, Controller menginstruksikan View untuk memperbarui tampilan berdasarkan perubahan data yang terjadi.

Dalam arsitektur MVC, Model, View, dan Controller bekerja secara terpisah dan saling bergantung satu sama lain. Model menyimpan data, View menampilkan data, dan Controller mengatur aliran data dan tindakan pengguna. Pemisahan ini memungkinkan pengembangan yang lebih terorganisir, perawatan yang lebih mudah, dan memungkinkan pengembang untuk membuat perubahan pada satu komponen tanpa harus mempengaruhi yang lainnya.

Dalam konteks pengembangan web dengan Laravel, "controller" merujuk pada komponen dari pola desain Model-View-Controller (MVC) yang bertanggung jawab untuk mengelola logika aplikasi dan menangani permintaan dari pengguna. Controller berfungsi sebagai perantara antara model dan view, mengatur bagaimana data diambil dari model dan ditampilkan di view.

Berikut ini adalah penjabaran tentang controller dalam Laravel:

1. Struktur Direktori:
   - File controller Laravel umumnya ditempatkan di direktori **`app/Http/Controllers`.**
   - Anda dapat membuat sub-direktori di dalam **`Controllers`** untuk mengorganisir kontroler sesuai dengan logika bisnis atau struktur proyek.

2. Membuat Controller:
- Untuk membuat controller baru, Anda dapat menggunakan perintah Artisan **`make:controller`.**
- Misalnya, jika Anda ingin membuat controller bernama **`UserController`,** Anda dapat menggunakan perintah berikut di terminal:
  php artisan make:controller UserController
  - Ini akan membuat file `**UserController.php**` di direktori `**app/Http/Controllers**`.

3. Metode dalam Controller:
   - Controller Laravel berisi metode-metode yang mengelola logika aplikasi.
   - Metode-metode ini merespons permintaan HTTP dan mengembalikan respons yang sesuai, seperti menampilkan view, mengirimkan data, atau mengarahkan pengguna ke rute lain.
   - Misalnya, dalam `UserController`, Anda dapat memiliki metode seperti `index()`, `create()`, `store()`, `show()`, `edit()`, `update()`, `destroy()`, dan lainnya, sesuai dengan kebutuhan aplikasi Anda.

4. Mengarahkan Respons:
   - Dalam metode controller, Anda dapat menggunakan metode `view()` untuk mengarahkan ke view yang sesuai.

- Misalnya, jika Anda ingin menampilkan view `home.blade.php`, Anda dapat menggunakan kode berikut:
  return view('home');
- Anda juga dapat menggunakan metode `redirect()` untuk mengarahkan pengguna ke rute atau URL tertentu.
- Misalnya, jika Anda ingin mengarahkan pengguna ke rute `profile`, Anda dapat menggunakan kode berikut:
  return redirect()->route('profile');

5. Mengelola Data:
   - Controller digunakan untuk mengambil data dari model dan meneruskannya ke view atau melakukan pemrosesan data lainnya.
   - Anda dapat menggunakan model dalam controller untuk melakukan operasi basis data, seperti pencarian, pembuatan, pembaruan, atau penghapusan data.
   - Misalnya, Anda dapat menggunakan model `User` untuk mengambil data pengguna dari basis data dan meneruskannya ke view.

6. Dependensi dan Injeksi:
   - Controller Laravel mendukung injeksi dependensi untuk memudahkan penggunaan komponen lain, seperti model, layanan, atau repositori.
   - Anda dapat mengonfigurasi dependensi di konstruktor kontroler dan Laravel secara otomatis akan menyediakan instance yang sesuai saat kontroler dibuat.

7. Middleware:
   - Middleware digunakan untuk menengahi permintaan sebelum atau setelah melewati controller.
   - Dalam kontroler, Anda dapat mendaftarkan middleware untuk melakukan tugas tertentu sebelum atau setelah eksekusi metode kontroler.
   - Misal nya, Anda dapat menggunakan middleware untuk melakukan otentikasi pengguna sebelum menjalankan metode tertentu dalam kontroler.

Dengan controller dalam Laravel, Anda dapat mengatur logika bisnis dan mengelola permintaan dari pengguna dengan jelas. Ini memisahkan tanggung jawab antara model, view, dan controller, sehingga membuat aplikasi lebih terstruktur dan mudah dikelola.

3. Cara Kerja MVC

Secara gampangnya sebenarnya controller adalah “otak” sedang view adalah “muka” dan model adalah “data”. Ketika browser memberi request maka akan dipilih untuk controller mana yang tepat untuk menangani request tersebut dan meminta data

dari model dan meminta view yang dibutuhkan. Kemudian di keluarkan hasilnya lagi lewat controler ke browsernya.

1. View akan menampilkan user interface aplikasi

Pada aplikasi uji coba konsep MVC, view pertama kali akan menampilkan user interface dan informasi tambahan lain yang terdapat di aplikasi pada pengguna. Tahap ini harus dilakukan semenarik mungkin, karena tampilan awal sebuah aplikasi menjadi penentu pengguna menyukainya atau tidak.

Ketika pengguna melakukan request di aplikasi, view akan menampung hal tersebut dan melanjutkannya ke bagiancontroller. Apabila sudah dilanjutkan, controller akan menerima request aplikasi dari bagian view.

Dalam pola desain Model-View-Controller (MVC), view bertanggung jawab untuk menampilkan antarmuka pengguna (user interface) aplikasi. View digunakan untuk mengatur tampilan dan struktur halaman yang akan ditampilkan kepada pengguna.

Berikut adalah cara kerja view dalam MVC untuk menampilkan antarmuka pengguna aplikasi:

1. Pengguna mengirimkan permintaan HTTP (misalnya, mengakses URL) ke aplikasi Anda.
2. Rute (route) di aplikasi Laravel menangkap permintaan dan menentukan kontroler mana yang harus menanganinya.
3. Kontroler akan memproses permintaan, melakukan pemrosesan logika bisnis yang diperlukan, dan mengambil data dari model jika diperlukan.
4. Setelah pemrosesan logika bisnis, kontroler akan mempersiapkan data yang diperlukan untuk ditampilkan di antarmuka pengguna.
5. Kontroler akan memanggil view yang sesuai dan meneruskan data yang telah dipersiapkan sebagai argumen.
6. View akan mengambil data yang diterima dan menggunakan sintaks template engine (seperti Blade dalam Laravel) untuk menghasilkan HTML yang akhirnya akan ditampilkan kepada pengguna.
7. HTML yang dihasilkan oleh view akan dikirimkan kembali sebagai respons ke peramban pengguna.
8. Peramban akan menerima HTML dan akan mengeksekusinya untuk menampilkan antarmuka pengguna aplikasi kepada pengguna.

Dalam proses ini, view bertindak sebagai lapisan presentasi yang memisahkan antarmuka pengguna dari logika bisnis dan akses ke data. View menggunakan template engine untuk memudahkan penggunaan logika PHP dalam menampilkan data dan mengatur tampilan. Dengan menggunakan sintaks template engine, Anda

dapat melakukan perulangan, kondisional, dan penggunaan variabel untuk menghasilkan tampilan yang dinamis.

Perlu diperhatikan bahwa dalam Laravel, view juga dapat menerima data dari kontroler dan melakukan aksi seperti pengaturan nilai variabel, pemanggilan komponen atau partial view, dan penggunaan direktif Blade lainnya. Semua ini membantu dalam menghasilkan antarmuka pengguna yang fleksibel dan dinamis sesuai dengan kebutuhan aplikasi.

Dengan menggunakan pola desain MVC, Anda dapat memisahkan logika bisnis, tampilan, dan interaksi dengan data menjadi bagian-bagian yang terpisah, sehingga membuat kode lebih terorganisir, mudah dimengerti, dan memungkinkan pengembangan aplikasi yang berskala besar dan terkelola dengan baik.

3. Controller memberikan instruksi pada model

Saat mengelola informasi di database, model tak melakukannya sendirian saja, ada bantuan logika pemrograman yang membuat proses pengelolaan lebih cepat. Setelah selesai, model harus menyerahkan hasil pengolahan informasi database ke controller, bukan bagian view. Kemudian, view menggunakan data yang telah siap diterima controller dari model untuk ditampilkan pada pengguna.

Dalam pola desain Model-View-Controller (MVC) dalam Laravel, controller berperan sebagai perantara antara model dan view. Controller memberikan instruksi pada model untuk melakukan operasi basis data atau logika bisnis tertentu. Controller juga mengambil data dari model dan meneruskannya ke view untuk ditampilkan kepada pengguna.

Berikut adalah cara kerja controller dalam Laravel untuk memberikan instruksi pada model:

1. Permintaan HTTP diterima oleh aplikasi Laravel dan ditangani oleh rute yang sesuai.
2. Rute menentukan kontroler mana yang harus menangani permintaan.
3. Kontroler yang sesuai dipanggil dan metode yang sesuai dalam kontroler dijalankan.
4. Dalam metode kontroler, Anda dapat melakukan beberapa instruksi, termasuk instruksi pada model.
5. Untuk berinteraksi dengan model, Anda dapat membuat instance model yang relevan dalam kontroler.

6. Anda dapat menggunakan metode model seperti `find()`, `create()`, `update()`, `delete()`, dan lainnya untuk melakukan operasi basis data pada model.
7. Jika perlu, Anda dapat mengubah atau memanipulasi data sebelum atau setelah memanggil metode model.
8. Setelah mendapatkan data yang diinginkan dari model, Anda dapat meneruskannya ke view menggunakan metode `view()` atau `redirect()` dalam kontroler.
9. Data yang diteruskan ke view dapat diakses dan ditampilkan menggunakan sintaks template engine (misalnya, Blade dalam Laravel) di dalam view.

Melalui langkah-langkah di atas, controller memberikan instruksi pada model untuk melakukan operasi basis data atau logika bisnis tertentu yang diperlukan dalam pemrosesan permintaan pengguna. Setelah mendapatkan data yang diperlukan, controller meneruskannya ke view agar dapat ditampilkan kepada pengguna.

Penting untuk diingat bahwa controller bertanggung jawab untuk mengatur alur aplikasi, memproses permintaan, dan mempersiapkan data sebelum diteruskan ke view. Dengan memisahkan logika bisnis dan interaksi dengan model ke dalam controller, kode Anda akan lebih terstruktur dan lebih mudah dipelihara.

4. Model menyerahkan hasil pengolahan informasi database

Dalam Laravel, model berperan sebagai representasi dari tabel atau entitas dalam basis data. Model bertanggung jawab untuk melakukan operasi pada basis data, seperti mengambil, menyimpan, memperbarui, atau menghapus data. Setelah melakukan pengolahan informasi database, model kemudian menyerahkan hasilnya kepada controller atau bagian lain dari aplikasi.

Berikut adalah cara kerja model dalam Laravel untuk menyerahkan hasil pengolahan informasi database:

a. Model dalam Laravel biasanya ditempatkan di direktori app/Models.
b. Anda dapat membuat model baru dengan menggunakan perintah Artisan **make:model.**
c. Misalnya, jika Anda ingin membuat model untuk entitas "User", Anda dapat menggunakan perintah berikut di terminal:
```
php artisan make:model User
```
d. Ini akan membuat file User.php di direktori app/Models.
e. Model dapat menghubungkan aplikasi Laravel dengan tabel atau entitas tertentu dalam basis data. Anda dapat mendefinisikan penghubung basis data, seperti pengaturan koneksi dan nama tabel, dalam model.
f. Dalam model, Anda dapat mendefinisikan atribut-atribut yang mewakili kolom-kolom dalam tabel basis data. Misalnya, jika tabel "users" memiliki

kolom "name", "email", dan "password", Anda dapat mendefinisikan atribut-atribut tersebut dalam model "User".

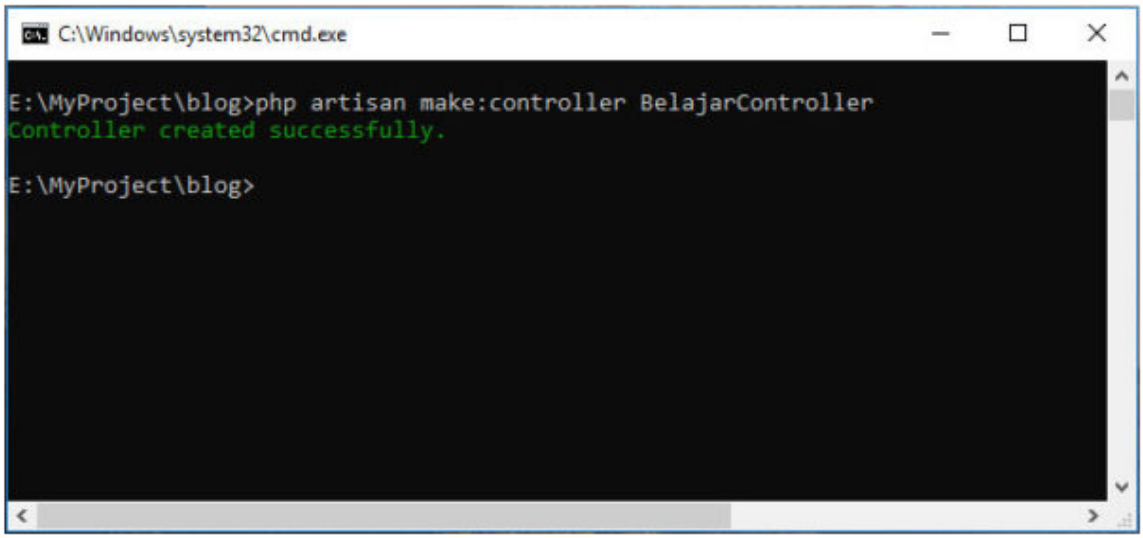
g. Model juga dapat memiliki metode-metode untuk melakukan operasi pada data. Misalnya, Anda dapat memiliki metode "createUser()" untuk membuat pengguna baru, metode "updateUser()" untuk memperbarui data pengguna, atau metode "deleteUser()" untuk menghapus pengguna.
h. Model dapat menggunakan fitur ORM (Object-Relational Mapping) dalam Laravel, yang memungkinkan Anda untuk melakukan operasi basis data menggunakan metode-metode model, seperti find(), create(), update(), delete(), dan lainnya.
i. Setelah melakukan operasi pada data, model akan menghasilkan hasil pengolahan informasi database, seperti hasil pencarian, data yang telah disimpan atau diperbarui, atau status operasi.
j. Model kemudian akan menyerahkan hasil pengolahan informasi tersebut kepada controller atau bagian lain dari aplikasi yang membutuhkannya.
k. Controller atau bagian lain dari aplikasi dapat menerima hasil pengolahan informasi dari model dan menggunakannya sesuai dengan kebutuhan aplikasi, seperti menampilkan data kepada pengguna, melakukan aksi lebih lanjut, atau mengirimkan respons ke peramban (browser) pengguna.
l. Dengan cara kerja di atas, model dalam Laravel bertanggung jawab untuk berinteraksi dengan basis data dan melakukan operasi pada data. Model menyerahkan hasil pengolahan informasi database kepada bagian-bagian lain dari aplikasi yang membutuhkannya, seperti controller, view, atau layanan lainnya. Ini memisahkan tanggung jawab antara model dan bagian-bagian lain dari aplikasi, sehingga membuat kode lebih terorganisir dan mudah dikelola.

5. Contoh MVC

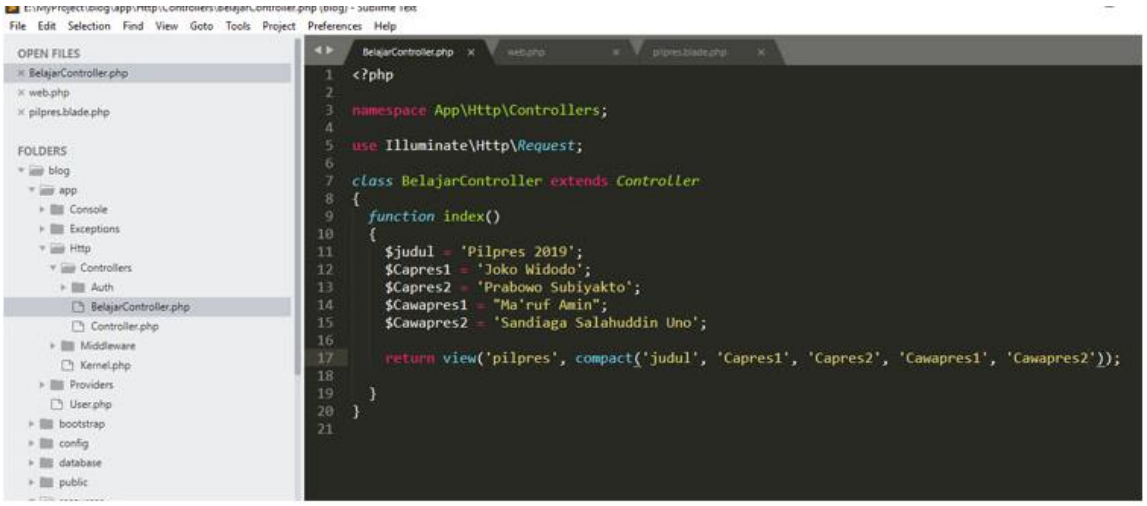
Mari kita buat contoh sederhana dari implementasi MVC di Laravel. Namun untuk sementara Model tidak diambil dari database, melainkan kita defenisikan sendiri dalam bentuk variable data.

Pertama, kita buat controller dengan perintah artisan dengan nama controller BelajarController.

```
php artisan make:controller BelajarController
```

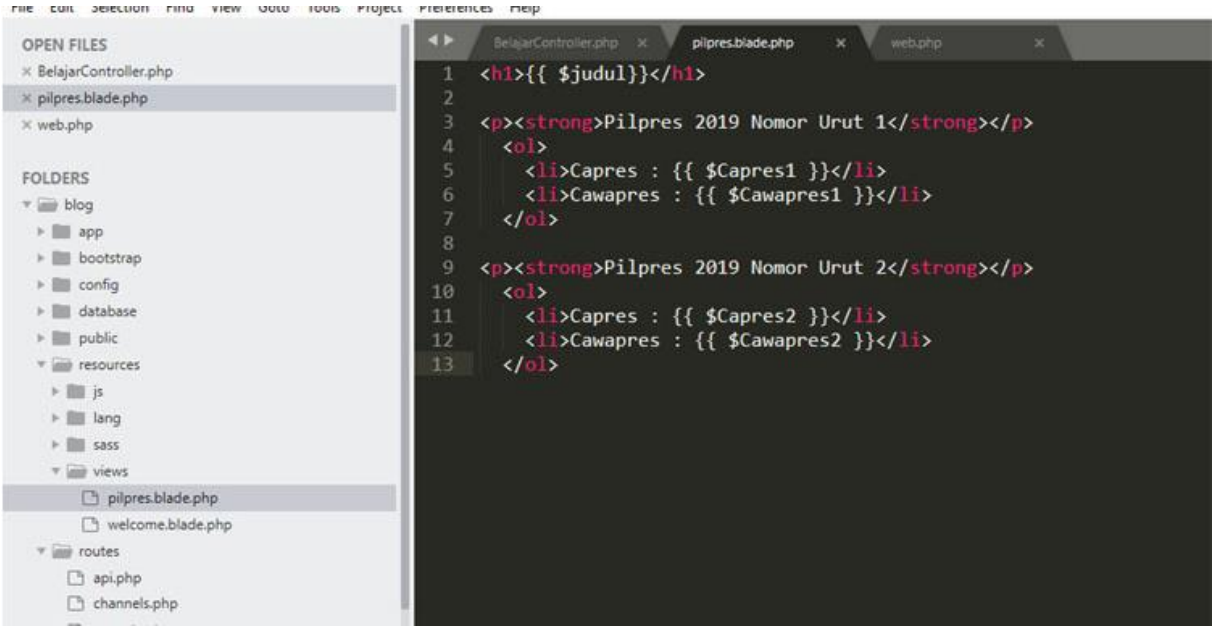


File BelajarController.php tersimpan pada folder app/Http/Controller/BelajarController.php. Silakan buat kode seperti gambar di bawah ini ;



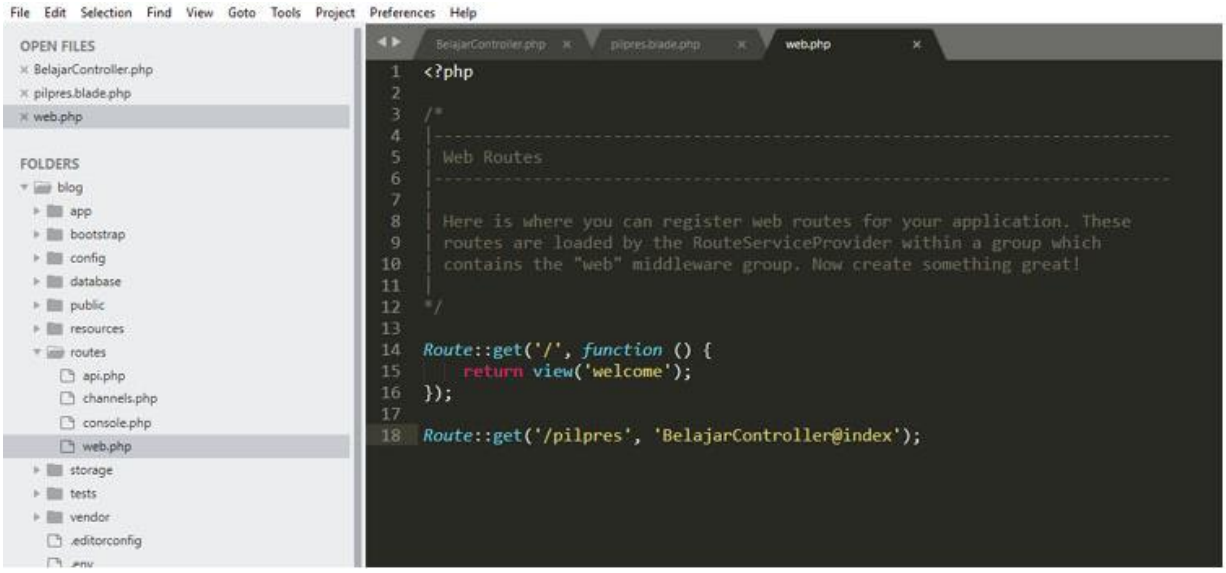
Pada controller BelajarController.php terdapat satu fungsi yaitu function index ();. Pada fungsi tersebut didefenisikan beberapa variable untuk selanjutnya akan ditampilkan pada blade view. Pada baris ke-17 variable tersebut di-return ke View dengan nama file pilpres.blade.php.Pada controller format file view tidak ditulis lengkap cukup dengan nama pilpres.

Kedua,buat file view dengan nama pilpres.blade.php yang disimpan pada folder resources/view/pilpres.blade.php. Silakan buat kode html seperti gambar di bawah ini:

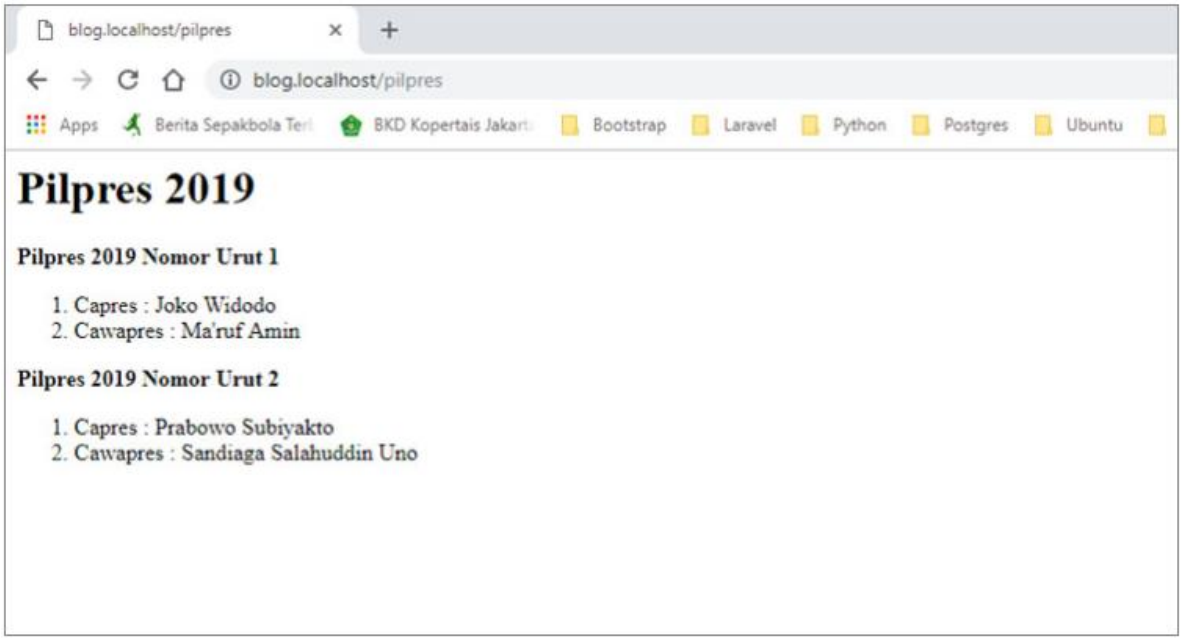


Untuk menampilkan variable yang didefinisikan pada controller, harus di diapit dengan tanda kurawal double {{..}} dan diawali dengan tanda dolar $, seperti pada kode baris ke-1 {{$judul}} di atas.

Ketiga, untuk dapat ditampilkan lewat browser, sebelumnya kita update terlebih dahulu file routes/web.php dengan menambahkan routing sebagai berikut:

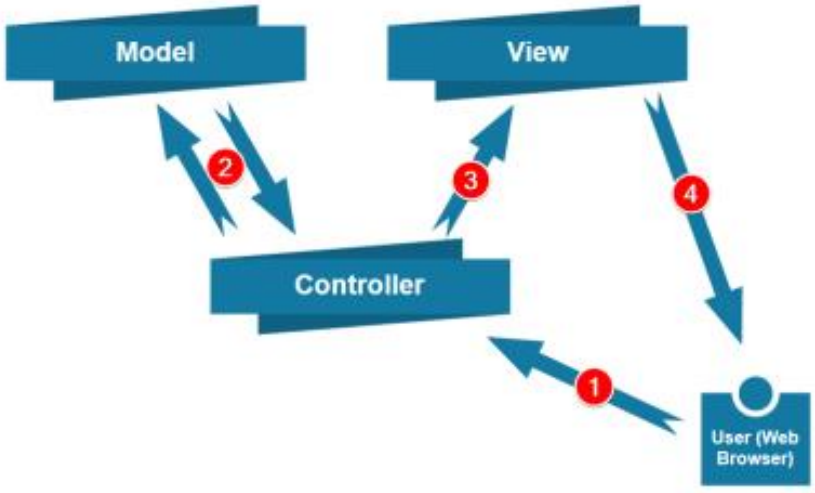


Terkahir, silakan akses dibrowser dengan url http://blog.localhost/pilpres



4. Alur

Kita telah melihat contoh praktek sederhana dari Model, View, dan Controller. Dimana model berisi kode untuk mengakses database, view untuk design tampilan, serta controller sebagai logika dan penghubung antara model dan view. Jika dibuat dalam bentuk diagram, berikut alur proses dari sebuah arsitektur MVC:



Pada gambar diatas dapat kita jelaskan bahwa :

Pertama , pada setiap interaksi yang dilakukan pengguna atau user akan ditangani langsung oleh controller. Contohnya ketika user mengetik alamat situs www.stekom.ac.id, maka controller di server universitas stekom akan menangkap "request" tersebut. Atau ketika user selesai mengisi form register dan men-klik tombol submit, file controller akan menerima proses tersebut.

Pada contoh kita sebelum ini, tahap 1 adalah ketika user mengetik nama file controller_mahasiswa.php di web browser, kemudian menekan tombol Enter. Controller pada dasarnya berisi logika program. Seandainya perlu mengambil data dari

database, maka controller akan memanggil Model (panah nomor 2). Model inilah yang bertanggung jawab mengakses database lalu mengembalikan hasilnya kembali ke controller.

Dalam contoh kita, model ada di file model_mahasiswa.php yang berisi fungsi getTableMahasiswa(). Poin penting di sini adalah, **model hanya berkomunikasi dengan controller**. Setelah data model diterima oleh pengontrol, pengontrol kemudian mengirimkan data ini ke tampilan (panah 3). Data ini kemudian diproses sebagai kode HTML dan CSS di dalam tampilan. Inilah yang dilihat pengguna di browser web (panah 4). Jika pengguna mengklik halaman lain, itu akan diproses lagi oleh pengontrol, yaitu pada langkah 1, dan seterusnya ke alur kerja arsitektur MVC.

Berikut adalah alur kerja umum dalam arsitektur MVC:

1. Pengguna berinteraksi dengan tampilan (View) yang menampilkan antarmuka pengguna, misalnya sebuah halaman web.
2. Ketika pengguna melakukan aksi, seperti mengklik tombol atau mengisi formulir, tampilan (View) menangkap aksi tersebut dan mengirimkannya ke kontroler (Controller).
3. Kontroler (Controller) menerima aksi dari tampilan (View) dan menginterpretasinya. Kontroler dapat memvalidasi input, memanggil model yang sesuai, atau memanggil layanan lain yang diperlukan.
4. Kontroler (Controller) berinteraksi dengan model untuk mendapatkan atau memperbarui data. Model berisi logika bisnis dan akses ke sumber daya data, seperti basis data atau layanan web.
5. Setelah memperoleh data atau melakukan perubahan pada model, kontroler (Controller) mengirimkan data yang relevan kembali ke tampilan (View).
6. Tampilan (View) menerima data dari kontroler (Controller) dan menggunakan data tersebut untuk menghasilkan tampilan yang sesuai, yang kemudian ditampilkan kepada pengguna.
7. Pengguna melihat tampilan yang diperbarui dan mungkin melakukan aksi lebih lanjut, memulai siklus kembali dengan mengirimkan aksi tersebut ke kontroler (Controller).

Alur kerja ini menjaga pemisahan antara tampilan (View), kontroler (Controller), dan model. Tampilan bertanggung jawab untuk menampilkan informasi kepada pengguna, kontroler mengatur aliran data dan tindakan pengguna, dan model menangani logika bisnis dan pengolahan data. Pemisahan ini memungkinkan pengembangan yang lebih terorganisir, memudahkan pemeliharaan, dan memungkinkan perubahan pada satu komponen tanpa harus mempengaruhi yang lainnya.

5. Routing

Laravel atau routeadalah sebuah alamat pada page tertentu agar supaya dapat diakses melalui web browser, yang mempermudah kita dalam memindahkan halaman dari halaman A ke halaman B atau sebaliknya, dll.Route pada laravel sendiri dibagi menjadi beberapa bagian dan pada setiap bagian tersebut memiliki peranan Masing - masing.

Pada framework Laravel, routing digunakan untuk menentukan bagaimana aplikasi web merespons permintaan HTTP dari pengguna. Routing menghubungkan URL yang diminta oleh pengguna dengan kode yang akan dijalankan dalam aplikasi.

Berikut adalah beberapa konsep dan contoh routing pada Laravel:

1. Definisi Routing:

- Routing pada Laravel didefinisikan dalam file `routes/web.php` atau `routes/api.php`.

- Anda dapat mendefinisikan routing menggunakan metode HTTP seperti GET, POST, PUT, DELETE, dll.

- Contoh: `Route::get('/halaman', 'HomeController@index');` akan menetapkan URL '/halaman' ke metode `index` dalam `HomeController`.

2. Parameter Routing:

- Anda dapat menentukan parameter dinamis dalam URL dengan menggunakan tanda kurung kurawal `{}`.

- Contoh: `Route::get('/produk/{id}', 'ProdukController@show');` akan menangani URL seperti '/produk/1' dan meneruskannya ke metode `show` dalam `ProdukController`.

3. Route Name:

- Anda dapat memberikan nama kepada route untuk mempermudah penggunaan dan referensi.

- Contoh: `Route::get('/profil', 'UserController@profile')->name('profil');` memberikan nama 'profil' pada route.

4. Route Grouping:

- Anda dapat mengelompokkan beberapa route bersama-sama menggunakan `Route::group` untuk memberlakukan middleware, prefix, atau namespace yang sama pada sekelompok route.

- Contoh:

```php
Route::prefix('admin')->middleware('auth')->group(function () {
    Route::get('/dashboard', 'AdminController@dashboard');
    Route::get('/pengguna', 'AdminController@users');
});
```

5. Middleware:

- Middleware memungkinkan Anda untuk memodifikasi permintaan HTTP sebelum atau setelah dijalankan oleh route.

- Middleware dapat digunakan untuk otorisasi, verifikasi CSRF, penanganan sesi, dll.

- Contoh: `Route::get('/admin', 'AdminController@index')->middleware('auth');` akan menggunakan middleware 'auth' untuk memeriksa apakah pengguna telah autentikasi sebelum menampilkan halaman admin.

Itu hanya beberapa konsep dasar routing pada Laravel. Laravel memiliki lebih banyak fitur routing yang kuat, seperti route cache, route model binding, prefixing, versioning, dll. Dengan menggunakan routing yang baik, Anda dapat dengan mudah mengorganisir dan mengelola alur navigasi dan permintaan HTTP dalam aplikasi Laravel Anda.

Letak file route sendiri ada poda folder **routes/** dan didalamnya sendiriterdapat 4buah file route yaitu, **api.php**, **channels.php**, **console.php**, dan **web.php**.

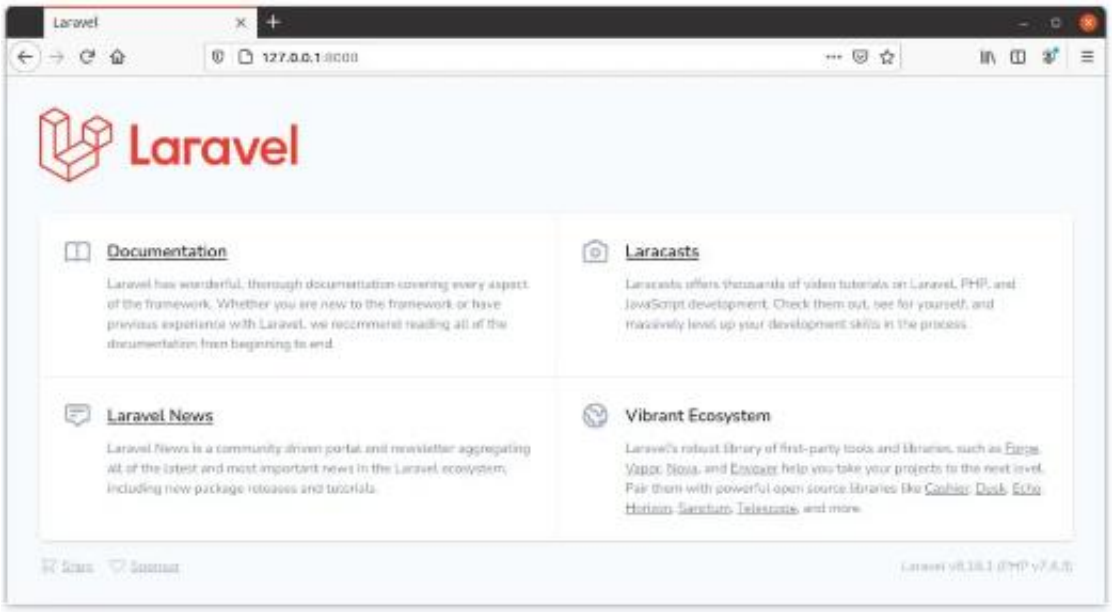
Untuk mengakses halaman website di browser tersebut mari kita fokus ke sebuah file yang bernama **web.php**, dari sini kita akan mulai mempelajari hal yang paling mendasar tentang bagaiman sebuah impelementasi rodari rouute pada sebuah framework laravel.

Mari Langsung saja kita mulai mencoba mempraktekan bagaimana cara penulisan route untuk membuat sebuah halaman dapat akses di web browser kita.

Pada saat pertama kali kita membuka sebuah file **routest/web.php** maka yang akan nampak di sebuah text editor kita adalah kode dibawah ini:

```php
<?php
use Illuminate\Support\Facades\Route;
...
Route::get('/', function () {
    return view('welcome');
});
```

Pada Kode Route::get('/') diatas adalah sebuah halaman pembuka Laravel saat pertama kali kita menginstall laravel dalam komputer kita. Seperti dibawah ini tampilannya.



Halaman utama diatas direnderolehLaravel ke sebuah browser, dimana file pembuka tersebut terletak di sebuah resources/views/welcome.blade.php, blade adalah template engine Laravel.

Menampilkan kata disebuah browser

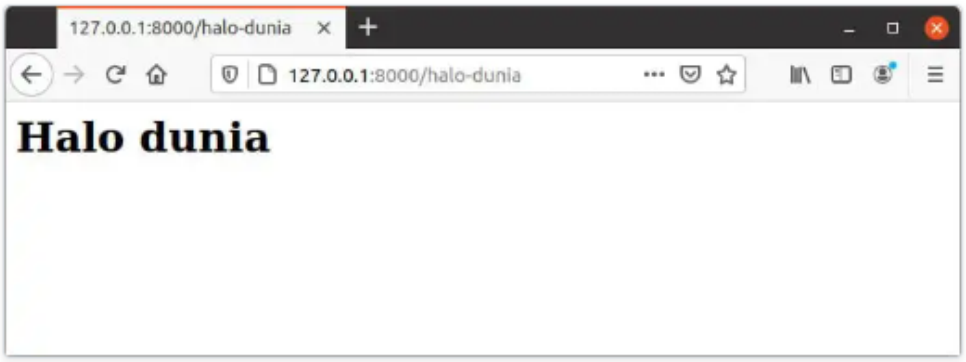
Selanjutnyakita akan mencoba membuat sebuah kata yang nantinya akan ditampikan pada sebuah browser. Mari kita buka file **routes/web.php** lalu tambahkan skrip/kode dibawah ini:

```
Route::get('/halo-dunia', function() {
    return '<h1>Halo dunia</h1>';
});
```

Karena pada saat ini penulis menggunakan php artisan server maka link atau url diatas bisa diakses seperti dibawah ini:

```
http://127.0.0.1:8000/halo-dunia
```

Dan selanjutnya akan menampilkan sebuah hasil seperti gambar dibawah ini:



Ada beberapa routes yangs terssedia di Laravel, berikut ini route tersebut:

```
Route::get($uri, $callback);
Route::post($uri, $callback);
Route::put($uri, $callback);
Route::patch($uri, $callback);
Route::delete($uri, $callback);
Route::options($uri, $callback);
Route::resource($uri, $callback);
```

Seperrti yang kita ketahui bahwa GET merupakan sebuah rute yang hanya bisa diakses langsung di browser, sedangkan Post mengirimkan data formulir ke sebuah server. Mari kita fokus pada GET sebagai bagian dari tutorial dasar ini.

Menambahkan parameter di URL

Apakah route hanya dapat menampilkan **URL statis**? ternyata jawabnya tidak, kita dapat mengirimkan sebuah parameter di URL dan akan dieksekusi di browser, berikut ini contohnya.

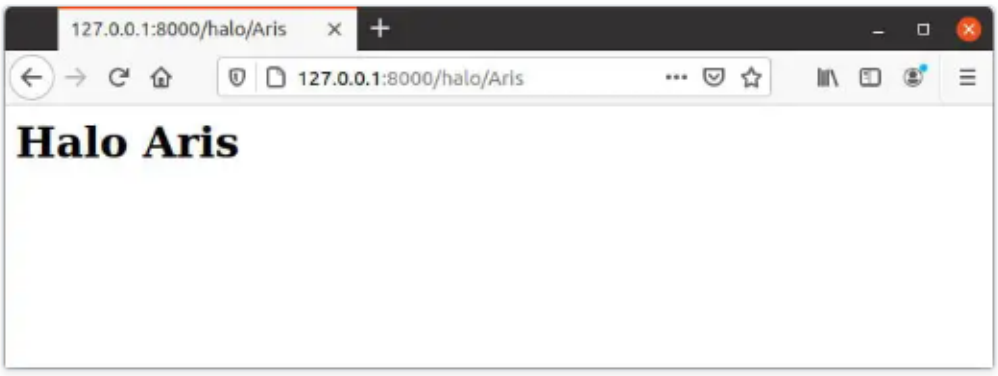
Tambahkanlah kode seperti dibawah ini:

```
Route::get('/halo/{nama}', function($nama) {
    return '<h1>Halo ' . $nama . '</h1>';
});
```

Lalu ketikanlah sebuah URL di browser seperti dibawah ini:

```
http://127.0.0.1:8000/halo/Aris
```

Maka akan menampilkan data seperti gambar dibawah ini:



parameter {nama} merupkan sebuah parameter yang akan dikirim ke variable $nama.

Lalu bagaimana jika ternyata tidak ada parameter **Aris** di sebuah URL? coba saja kita hilangkan parameter **Aris** di sebuahs browser, maka akan muncul sebuah **error 404**,hal ini karena parameter {nama} tidak memberikan sebuah nilai ke variable $nama, sehingga tidak ada satupun halaman yang dituju.

Lantas bagaimana agar tidak terjadi 404? tambahkan saja nilai default pada variable nama dan berikan tanpa tanya "?" diparameter nama apabila tidak ada parameter di URL, seperti kode dibawah ini:

```
Route::get('/halo/{nama?}', function($nama = 'Tanpa Nama') {
    return '<h1>Halo ' . $nama . '</h1>';
});
```

{nama?} akan memberi tahu kepada sebuah browser meskipun tanpa parameter tetap akan di eksekusi di browser karena di variable $nama sudah kita isi nilai default yaitu **Tanpa Nama**. Maka apabila di browser kita buka seperti URL dibawah ini:

```
http://127.0.0.1:8000/halo
```

Sekian merupakan pembahasan tentang sebuahrouting pada laravel, semoga bermanfaat, dan akan kita bahas lebih dalam lagi tentang laravel, Semoga dapat menambah pemahaman kita semua menegenai framework php yang sangat populer pada saat ini.

BAB 3

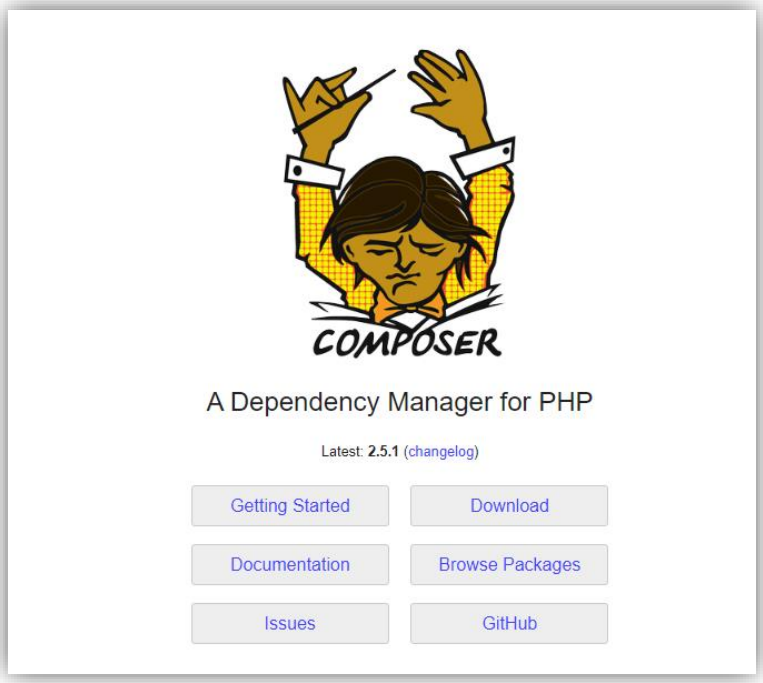
MENGINSTALL LARAVEL

Pada ini kita akan menginstall Laravel beserta aplikasi laian yang dibutuhkan untuk menjalankanya, seperti composer dan yang lain.

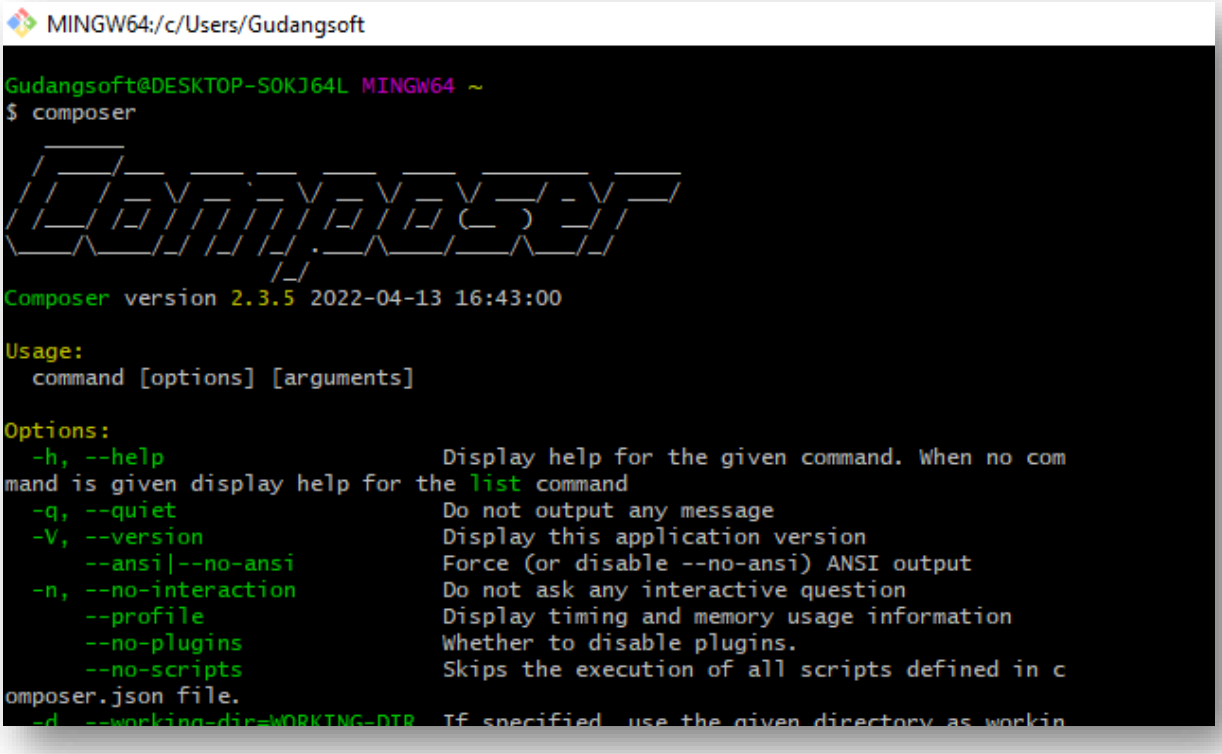
1. MENGINSTALL KOMPOSER

Komposer/Composer adalah aplikasi yang diinstal ke perangkat untuk memfasilitasi *developer* menggunakan pustaka/*library open source* milik orang lain ke dalam sebuah *project* yang sedang dibangun. Dalam sebuah *project* PHP, penggunaan pustaka/library sangat memudahkan kita dalam memproses dalam penulisan sebuah kode. Namun terkadang, library satu dengan yang lainnya saling membutuhkan agar bisa digunakan oleh seorang develpper. Hal ini disebut dengan dependensi atau ketergantungan. Di sinilah peran Composer untuk menghubungkan project PHP dengan library eksternal yang dibutuhkan.

Untuk mengistall composer silahkan masuk kehalaman website https://getcomposer.org/ lalu kitab isa memilih tombol download untuk mengunduh aplikasi composer, silahkan pilih seseaui Operasi Sistem yang kita gunakan.

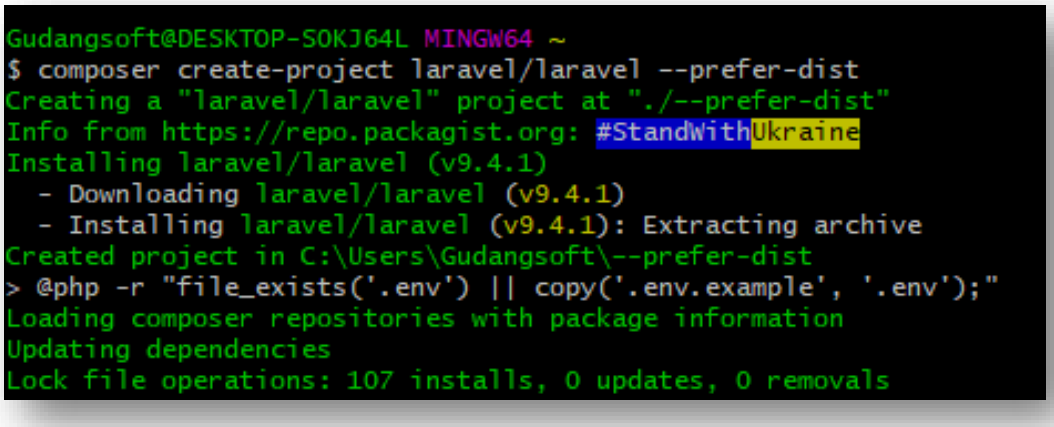


Selanjutnya Setelah kita memasang Composer,silahkan periksa instalasidengan mengetik sebuah perintah Composer di command prompt seperti gambar berikut ini.

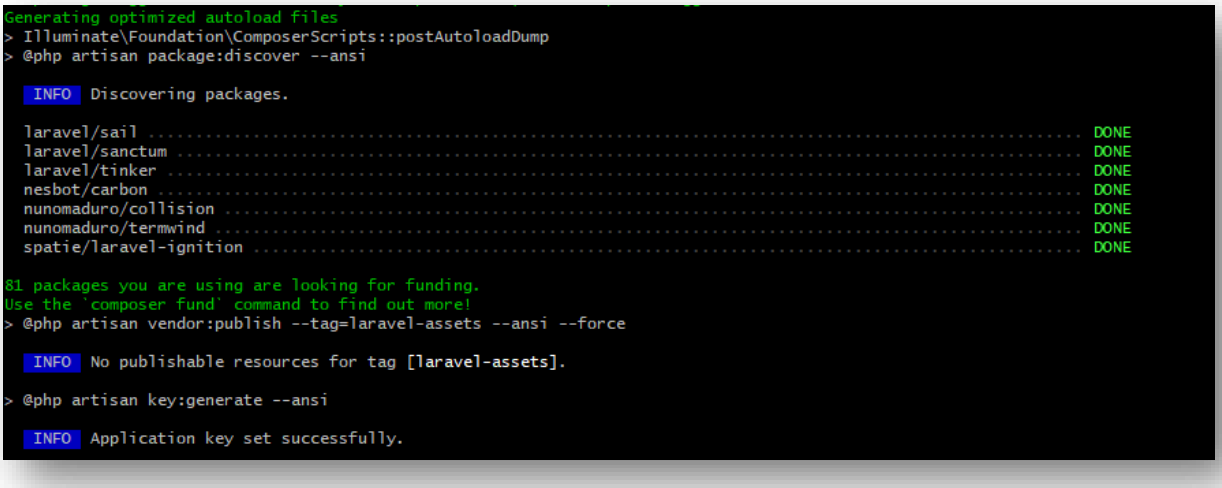


Selanjutnya silahkan sialahkan anda membuat sebuah direktori baru di mana saja di sistem Anda untuk yang baru Proyek Laravel. Setelah itu, pindah ke jalur tempat Anda membuat yang baru direktori dan ketik perintah berikut di sana untuk menginstal Laravel.

composer create-project laravel/laravel –-prefer-dist

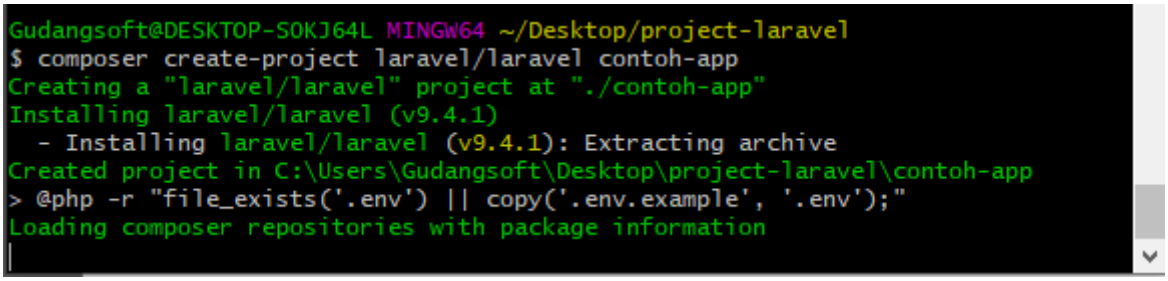


Tunggu hingga proses download selesai

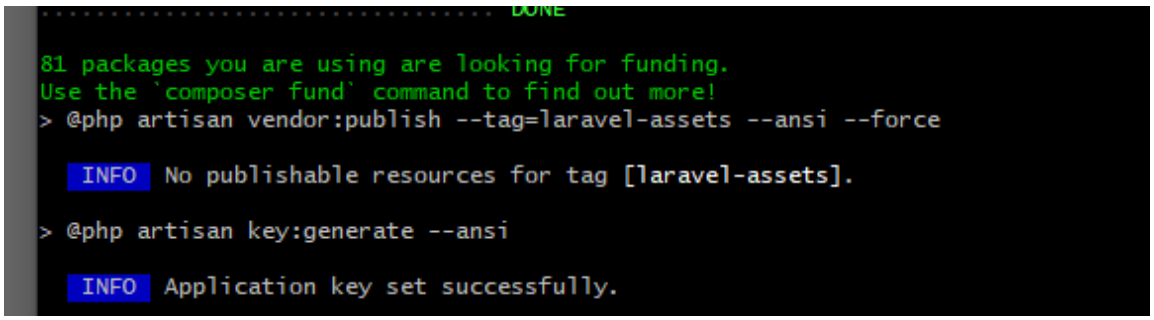


2. MENGINSTALL DAN MENJALANKAN LARAVEL

Proses selanjutnya kita akan mengintal Laravel terbaru saat buku ini dibuat adalah versi 9, silahkan ketikkan kode composer create-project laravel/laravel contoh-app (sesuaikan dengan nama project yang akan kita buat) dan tunggu hingga proses install selesai.



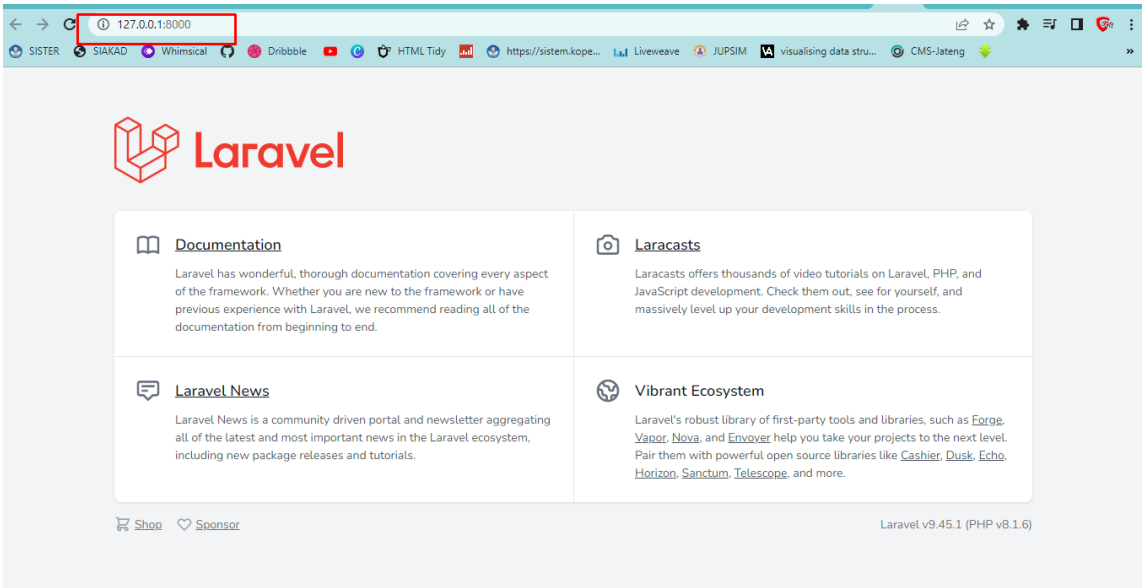
Langkah selanjutnya Kembali ke Command Promp kita, jika sudah selesai maka akan tampil sebagai gambar berikut ini



Untuk Langkah berikutnya silahkan ketikan perintah php artisan serve pada command promp kita

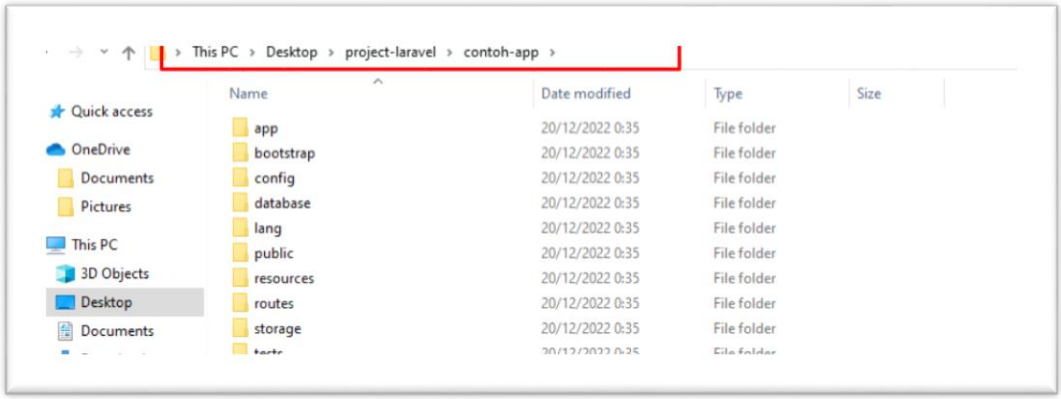
```
Gudangsoft@DESKTOP-SOKJ64L MINGW64 ~/Desktop/project-laravel/contoh-app
$ php artisan serve

  INFO  Server running on [http://127.0.0.1:8000].

  Press Ctrl+C to stop the server
```

Selanjutnya kita bisa membuka browser dengan cara klik tombol Ctrl dan arahkan kursor pada IP yang tampil diatas, sehingga akan membuka project Laravel yang sudah kita install tadi, seperti yangditunjukkan pada gambar dibawah ini, tekan tombol Ctrl+C untuk menghentikan Project.



3. STRUKTUR APLIKASI LARAVEL

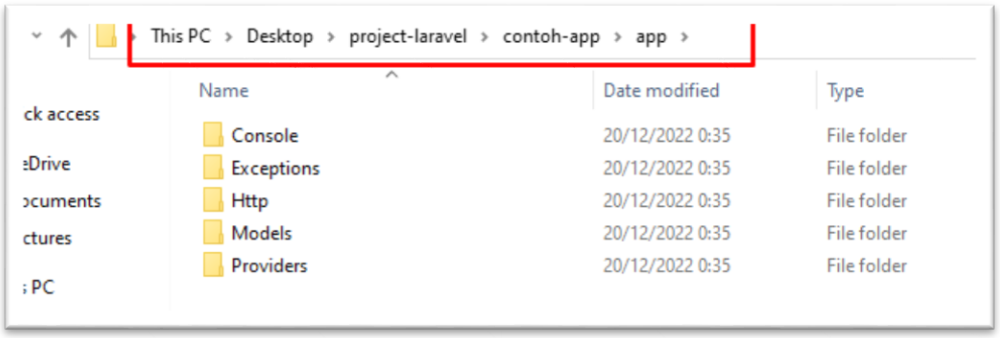
Struktur aplikasi Laravel dirancang dengan baik untuk memisahkan logika bisnis dari tampilan dan memudahkan pengembangan aplikasi web. Laravel mengikuti pola desain Model-View-Controller (MVC) yang memisahkan tiga komponen utama dalam pengembangan aplikasi web.



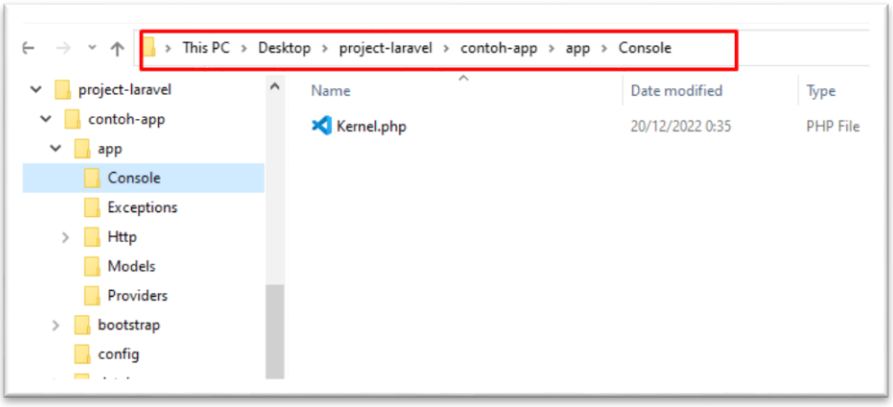
Berikut merupakan detail directory yang ada pada struktur aplikasi Laravel setelah kita selesai menginstall.

App

Ini merupakan folder dari sebuah aplikasi dan menyertakan seluruh sorcecode sumber dari sebuah proyek. Pada direktori/folder Ini berisi sebuah peristiwa, pengecualian, dan deklarasi middleware. Folder aplikasiterdiri dari berbagai sub folder seperti yang dijelaskan di bawah ini



Console



Console pada Laravel adalah sebuah fitur yang memungkinkan Anda untuk menjalankan perintah-perintah secara terprogram melalui Command Line Interface (CLI) atau Terminal. Console ini sangat berguna untuk menjalankan tugas-tugas terkait pengembangan dan pemeliharaan aplikasi Laravel Anda.

Berikut adalah beberapa poin penting yang perlu diketahui tentang console pada Laravel:

1. Artisan: Artisan adalah sistem perintah yang terintegrasi secara langsung dengan aplikasi Laravel. Artisan menyediakan berbagai perintah bawaan yang dapat Anda jalankan untuk melakukan tugas-tugas umum seperti membuat migration, menjalankan seeder, menghasilkan kode scaffold, dan masih banyak lagi. Anda juga dapat membuat perintah khusus Anda sendiri dengan menggunakan Artisan.

2. Menjalankan Perintah: Untuk menjalankan perintah pada console Laravel, Anda perlu menggunakan perintah **php artisan** diikuti dengan nama perintah yang ingin Anda jalankan. Misalnya, untuk menjalankan perintah migration, Anda dapat mengetik **php artisan migrate** di terminal. Anda juga dapat melampirkan argumen dan opsi tambahan ke perintah-perintah ini sesuai kebutuhan.

3. Membuat Perintah Khusus: Anda dapat membuat perintah khusus Anda sendiri dengan Artisan. Dengan perintah khusus, Anda dapat menjalankan tugas-tugas yang

spesifik untuk aplikasi Anda. Untuk membuat perintah khusus, Anda dapat menggunakan perintah **php artisan make:command NamaPerintah**. Setelah perintah khusus dibuat, Anda dapat menentukan logika di dalamnya dan menjalankannya melalui console.

4. Argumen dan Opsi: Perintah pada console Laravel juga dapat menerima argumen dan opsi tambahan. Argumen adalah nilai yang diperlukan oleh perintah, misalnya nama file atau ID, sementara opsi adalah pengaturan tambahan yang dapat diaktifkan atau dinonaktifkan. Anda dapat menentukan argumen dan opsi dalam perintah khusus yang Anda buat, serta saat menjalankan perintah bawaan Laravel.

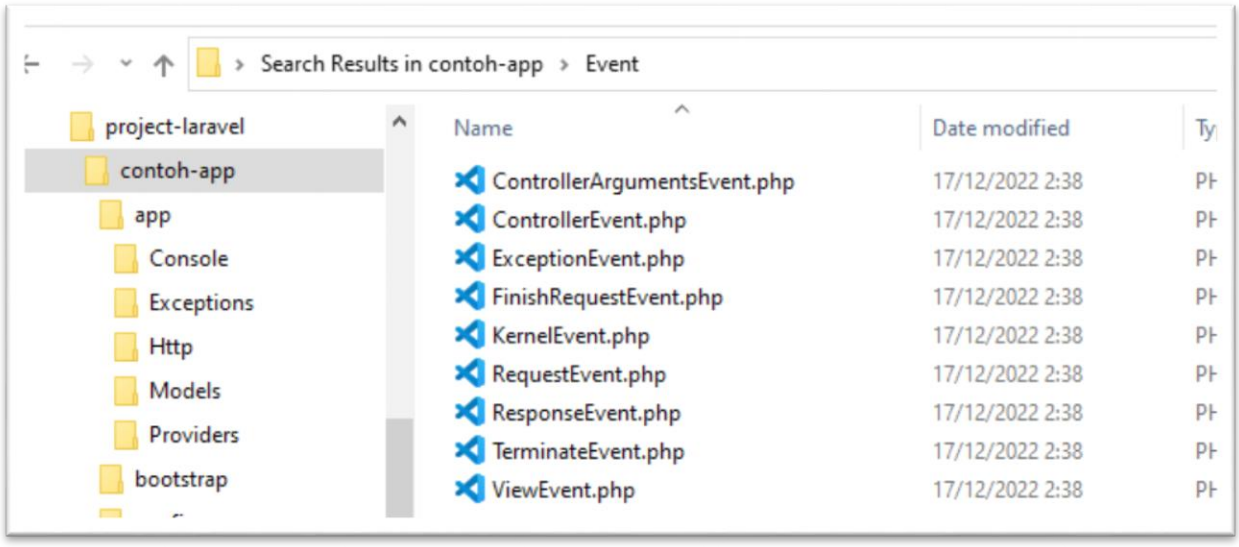
5. Output dan Interaksi: Console Laravel menyediakan kemampuan untuk menghasilkan output dan berinteraksi dengan pengguna. Anda dapat mencetak informasi, kesalahan, atau pesan ke konsol menggunakan metode **info()**, **error()**, atau **line()**. Selain itu, Anda juga dapat menggunakan input dari pengguna melalui metode **ask()** atau **confirm()** untuk mendapatkan respons yang diperlukan saat menjalankan perintah.

Console pada Laravel merupakan alat yang kuat untuk menjalankan dan mengotomatiskan tugas-tugas pengembangan aplikasi Anda. Dengan menggunakan perintah-perintah yang disediakan oleh Artisan, Anda dapat dengan mudah melakukan tugas-tugas seperti migrasi database, pengolahan antrian, menjalankan tes, dan banyak lagi. Selain itu, Anda juga dapat membuat perintah khusus sesuai kebutuhan aplikasi Anda, memperluas kemampuan console Laravel, dan meningkatkan efisiensi pengembangan Anda.

Console menyertakan perintah artisan yang sabgat diperlukan oleh Laravel. Itu termasuk a direktori dinamakan Commands , di mana semua perintah dideklarasikan dengan tanda tangan yang sesuai.

File Kernal.php memanggil perintah yang dideklarasikan Inspire.php.

Event



Event sendiri digunakan untuk memacu sebuah aktivitas, meningkatkan sebuah kesalahan, atau memvalidasi sebuah kode yang diperlukan dan memberikan fleksibilitas yang lebih besar. Laravel menyimpan semua acara di bawah satudirektori. File default yang harus disertakan adalah event.php di mana semua event dasar dinyatakan.

Event pada Laravel adalah fitur yang memungkinkan Anda untuk mengimplementasikan pola desain Observer dalam aplikasi Anda. Dengan menggunakan event, Anda dapat membuat suatu peristiwa atau kejadian di dalam aplikasi Anda, dan kemudian Anda dapat mendaftarkan dan menangani respons terhadap peristiwa tersebut.

Berikut adalah beberapa poin penting tentang event pada Laravel:

- Pembuatan Event:

  Anda dapat membuat event baru dengan menggunakan perintah Artisan php artisan make:event NamaEvent. Event ini akan dibuat di dalam direktori app/Events pada struktur aplikasi Laravel Anda. Setelah event dibuat, Anda dapat menentukan informasi apa pun yang relevan dengan peristiwa tersebut di dalamnya.

- Mendaftarkan Listener:

  Listener adalah kelas yang bertugas menangani respons terhadap suatu event. Anda dapat mendaftarkan listener ke dalam event yang sesuai. Anda dapat menggunakan perintah Artisan php artisan make:listener NamaListener --event=NamaEvent untuk membuat listener baru. Setelah listener dibuat, Anda dapat menentukan logika yang ingin dijalankan ketika event dipicu.

- Pemicu Event:

  Untuk memicu atau melepaskan event, Anda dapat menggunakan metode event() yang disediakan oleh Laravel. Misalnya, jika Anda memiliki event bernama OrderPlaced, Anda dapat memicunya dengan menggunakan event(new OrderPlaced($order)). Pemanggilan metode event() ini akan memicu event dan mengaktifkan semua listener yang terdaftar untuk event tersebut.

- Penanganan Event:

  Ketika event dipicu, semua listener yang terdaftar akan diaktifkan. Listener dapat melakukan berbagai tugas tergantung pada kebutuhan Anda, seperti mengirim email, memperbarui basis data, mengirim notifikasi, atau melakukan tindakan lainnya. Anda dapat menentukan logika penanganan event di dalam metode handle() pada listener yang terkait.

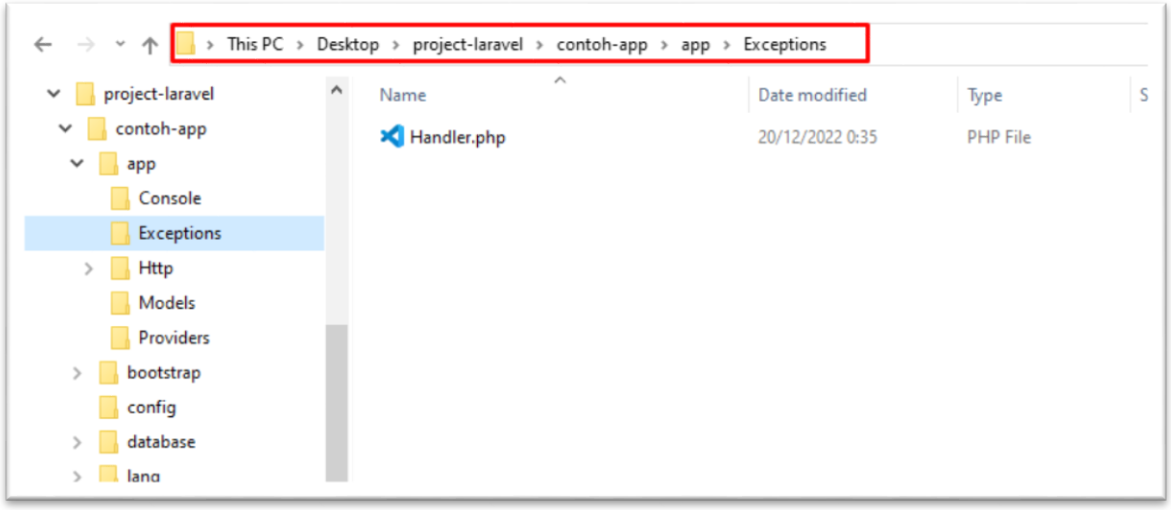
- Pendaftaran Event dan Listener:

  Agar event dan listener dapat berfungsi, Anda perlu mendaftarkannya di dalam aplikasi Anda. Pendaftaran ini biasanya dilakukan di dalam file EventServiceProvider yang terletak di

dalam direktori app/Providers. Anda perlu menentukan daftar event dan listener yang akan didaftarkan di dalam metode listen() pada class tersebut.

Dengan menggunakan event pada Laravel, Anda dapat memisahkan logika bisnis dan tindakan respons dari bagian lain aplikasi Anda. Hal ini memungkinkan Anda untuk membuat aplikasi yang lebih modular, fleksibel, dan dapat di-maintain dengan baik. Anda dapat menangani banyak peristiwa dan memberikan respons yang tepat sesuai kebutuhan aplikasi Anda dengan menggunakan event dan listener.

Exceptions



Exceptions (pengecualian) pada Laravel adalah mekanisme yang digunakan untuk menangani situasi atau kondisi yang tidak terduga atau tidak diinginkan dalam aplikasi. Ketika suatu pengecualian terjadi, Laravel secara otomatis akan menangkapnya dan memberikan respons yang sesuai, seperti menampilkan halaman kesalahan atau memberikan tanggapan JSON.

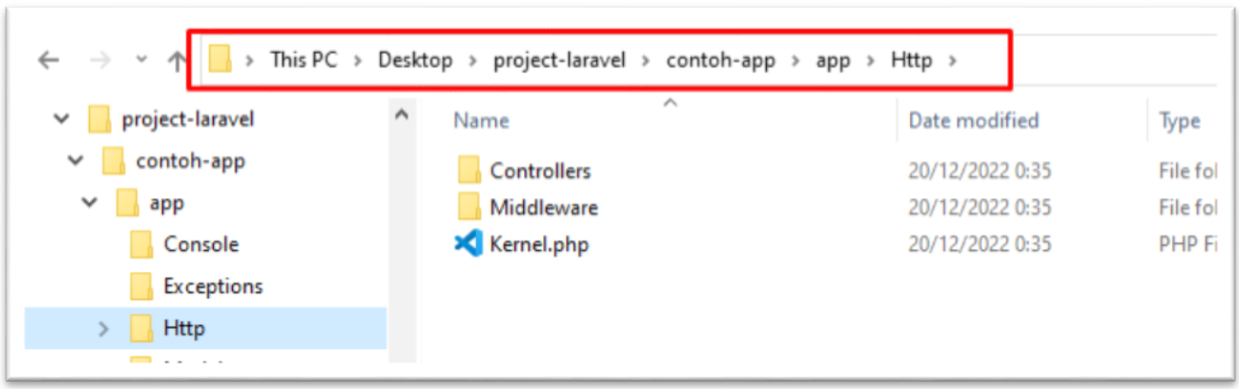
Berikut adalah beberapa poin penting tentang exceptions pada Laravel:

1. Mekanisme Pengecualian:
   Laravel menyediakan berbagai macam pengecualian yang telah ditentukan sebelumnya untuk menangani situasi yang umum terjadi, seperti ModelNotFoundException, FileNotFoundException, dan InvalidRequestException. Anda juga dapat membuat pengecualian khusus yang sesuai dengan kebutuhan aplikasi Anda.
2. Menangkap Pengecualian:
   Laravel secara otomatis menangkap pengecualian yang terjadi di dalam aplikasi. Anda dapat menentukan cara penanganan pengecualian dengan menambahkan kode ke dalam metode `report()` pada file `app/Exceptions/Handler.php`. Anda dapat menentukan berbagai respons, seperti menampilkan halaman kesalahan kustom, mengirim notifikasi, atau melakukan tindakan lainnya.

3. Halaman Kesalahan Kustom:
   Anda dapat membuat halaman kesalahan kustom yang sesuai dengan tampilan dan gaya aplikasi Anda. Halaman ini akan ditampilkan ketika pengecualian terjadi. Anda dapat membuat tampilan kesalahan kustom di dalam direktori `resources/views/errors` dan menentukan tampilan yang ingin digunakan untuk setiap jenis pengecualian.
4. Exception Handling untuk API:
   Jika Anda mengembangkan API dengan Laravel, Anda dapat menangani pengecualian dengan respons JSON yang sesuai. Laravel akan secara otomatis mengonversi pengecualian menjadi respons JSON yang dapat dipahami oleh klien. Anda dapat menentukan respons JSON yang spesifik untuk setiap jenis pengecualian di dalam metode `render()` pada file `app/Exceptions/Handler.php`.
5. Pengecualian Global:
   Anda juga dapat menentukan pengecualian global yang akan menangani semua pengecualian yang tidak tertangkap. Pengecualian global ini dapat digunakan untuk menangani pengecualian yang tidak spesifik atau untuk mengambil tindakan default. Pengecualian global dapat didaftarkan di dalam metode `register()` pada file `app/Exceptions/Handler.php`.

Melalui mekanisme exceptions, Laravel memungkinkan Anda untuk menangani dan memberikan respons yang sesuai terhadap situasi yang tidak terduga atau pengecualian yang terjadi dalam aplikasi. Dengan penanganan pengecualian yang baik, Anda dapat memberikan pesan yang informatif kepada pengguna dan melakukan tindakan yang diperlukan untuk memperbaiki masalah dalam aplikasi Anda.

Http



Folder/direktori Httpini memiliki sebuah sub-folder sebagai pengontrol, middleware, dan aplikasi permintaan. Karena Laravel mengikuti pola desain MVC, folder indi disertakan model, pengontrol, dan tampilan yang ditentukan untuk direktori tertentu. Sub-folder Mdiddleware menyertakan mekanisme middleware, yang terdiri dari mekanisme filter dan komunikasi antara respons dan permintaan. Sub-folder Permintaan mencakup semua permintaan aplikasi.

HTTP pada Laravel merujuk pada protokol komunikasi yang digunakan untuk pertukaran data antara klien (browser) dan server dalam konteks pengembangan aplikasi web. Laravel menyediakan

berbagai fitur dan komponen yang mempermudah penanganan permintaan HTTP dan pembuatan respons.

Berikut adalah beberapa poin penting tentang HTTP pada Laravel:

1. Routing: Laravel menggunakan komponen routing yang kuat untuk mengatur rute (routes) HTTP dalam aplikasi. Dengan menggunakan file **routes/web.php**, Anda dapat mendefinisikan rute-rute untuk berbagai URL yang ingin ditangani oleh aplikasi Anda. Anda dapat menentukan aksi (action) yang terkait dengan setiap rute, yang bisa menjadi fungsi atau metode di dalam controller.

2. Middleware: Middleware pada Laravel digunakan untuk memodifikasi atau memeriksa permintaan HTTP sebelum mencapai tujuan akhirnya. Middleware memungkinkan Anda untuk menambahkan logika tambahan seperti autentikasi, otorisasi, atau manipulasi data sebelum permintaan mencapai aksi yang dituju. Middleware dapat diterapkan pada level rute atau grup rute tertentu.

3. Controller: Controller dalam Laravel berperan penting dalam menangani permintaan HTTP. Controller adalah kelas yang mengelola logika bisnis aplikasi Anda. Ketika rute terpanggil, controller yang sesuai akan diaktifkan dan metode tertentu di dalamnya akan dieksekusi. Controller biasanya digunakan untuk memproses input, berinteraksi dengan model, dan menghasilkan respons.

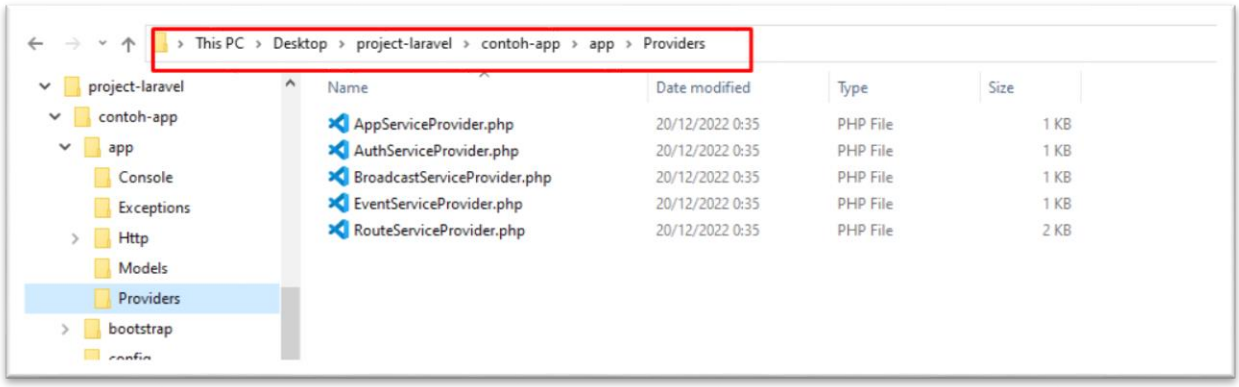
4. Request dan Response: Laravel menyediakan kelas-kelas Request dan Response yang mempermudah manipulasi data permintaan dan pembuatan respons HTTP. Dengan kelas Request, Anda dapat mengakses data pengguna seperti input form, file upload, dan header permintaan. Dengan kelas Response, Anda dapat membuat respons HTTP dengan mudah, mengatur status kode, header, dan isi respons.

5. Session dan Cookie: Laravel menyediakan dukungan yang kuat untuk pengelolaan session dan cookie dalam aplikasi web. Anda dapat menyimpan data dalam session untuk menjaga keadaan antar permintaan, serta menggunakan cookie untuk menyimpan informasi yang berkaitan dengan pengguna.

6. Validasi Input: Laravel menyediakan fitur validasi yang kuat untuk memeriksa data input dari permintaan HTTP. Dengan menggunakan aturan validasi, Anda dapat memverifikasi bahwa data yang diterima sesuai dengan persyaratan yang diinginkan sebelum memprosesnya. Jika validasi gagal, Laravel akan menghasilkan pesan kesalahan yang dapat ditampilkan kepada pengguna.

HTTP adalah bagian integral dalam pengembangan aplikasi web, dan Laravel menyediakan berbagai fitur yang memudahkan penanganan permintaan HTTP dan pembuatan respons. Dengan menggunakan komponen-komponen tersebut, Anda dapat dengan mudah membuat rute, mengatur middleware, mengelola permintaan dan respons, serta melakukan validasi dan manipulasi data.

Provider



Provider pada Laravel merujuk pada kelas-kelas yang digunakan untuk mengatur berbagai komponen dan layanan dalam aplikasi. Provider bertindak sebagai jembatan antara framework Laravel dan komponen eksternal, serta menyediakan konfigurasi dan logika untuk menginisialisasi komponen dan layanan yang dibutuhkan.

Berikut adalah beberapa poin penting tentang provider pada Laravel:

1. Service Provider: Service Provider adalah jenis provider yang paling umum digunakan dalam Laravel. Service Provider bertanggung jawab untuk mengonfigurasi dan mendaftarkan berbagai layanan (services) dalam aplikasi. Layanan ini bisa berupa kelas-kelas yang Anda buat, seperti model, repository, atau utilitas, atau bisa juga berupa komponen eksternal yang diperlukan oleh aplikasi, seperti library atau package.

2. Pembuatan Service Provider: Anda dapat membuat Service Provider baru dengan menggunakan perintah Artisan php artisan make:provider NamaProvider. Service Provider ini akan dibuat di dalam direktori app/Providers. Setelah itu, Anda dapat menentukan berbagai layanan yang ingin didaftarkan dalam metode register() pada Service Provider tersebut.

3. Pendaftaran Service Provider: Anda perlu mendaftarkan Service Provider dalam aplikasi agar dapat digunakan. Pendaftaran dilakukan di dalam file config/app.php. Anda dapat menambahkan Service Provider baru ke dalam array providers pada file tersebut. Atau, jika Anda ingin melakukan pendaftaran secara otomatis, Anda dapat menggunakan metode register() pada class App\Providers\AppServiceProvider.

4. Fasade: Fasade adalah fitur Laravel yang memungkinkan Anda mengakses layanan-layanan yang didaftarkan dalam Service Provider dengan cara yang lebih mudah dan ekspresif. Fasade menyediakan antarmuka yang sederhana dan intuitif untuk berinteraksi dengan layanan tersebut. Anda dapat menggunakan Fasade untuk mengakses layanan menggunakan sintaks statis, tanpa harus menginisialisasi objek secara manual.

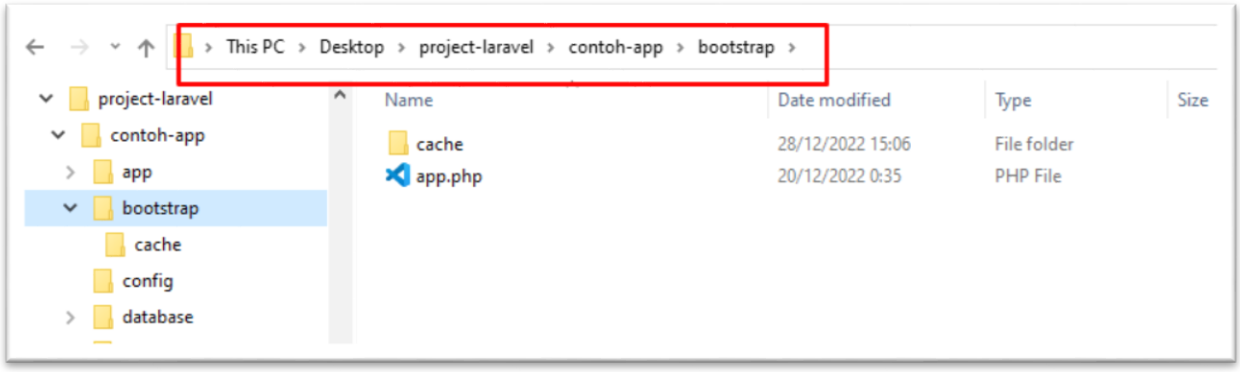
5. Booting Service Provider: Selain metode register(), Service Provider juga dapat memiliki metode boot() yang digunakan untuk melakukan tindakan atau pengaturan yang perlu dilakukan setelah Service Provider terdaftar. Metode boot() akan dipanggil setelah semua Service Provider terdaftar dalam aplikasi.

6. Provider Khusus: Selain Service Provider, Laravel juga memiliki jenis provider lain yang dapat digunakan, seperti Event Provider, View Provider, dan Route Provider. Provider-provider ini memiliki tanggung jawab spesifik terkait dengan pengaturan event, tampilan, atau rute dalam aplikasi.

Provider pada Laravel memainkan peran penting dalam mengatur komponen-komponen dan layanan-layanan dalam aplikasi Anda. Dengan menggunakan Service Provider, Anda dapat dengan mudah mendaftarkan dan mengonfigurasi berbagai layanan, memanfaatkan Fasade untuk akses yang lebih sederhana, dan memisahkan logika konfigurasi dari logika bisnis inti aplikasi Anda. Provider-provider khusus juga memungkinkan Anda untuk mengatur aspek-aspek spesifik aplikasi seperti event, tampilan, dan rute dengan cara yang terstruktur.

Folder ini mencakup semua penyedia layanan yang diperlukan untuk mendaftarkan acara server inti dan untuk mengonfigurasi aplikasi Laravel.

Bootstrap



Folder/direkroti ini menyertakan semua skrip bootstrap aplikasi. Ini berisi subfolder yaitu cache, yang mencakup semua file yang terkait untuk caching a xaplikasi web. Anda juga dapat menemukan file app.php, yang menginisialisasi skrip yang diperlukan untuk bootstrap.

Bootstrap pada Laravel merujuk pada proses inisialisasi awal yang dilakukan oleh framework Laravel untuk mempersiapkan lingkungan kerja aplikasi. Proses bootstrap ini melibatkan pengaturan konfigurasi, pendaftaran layanan, autoload, dan beberapa tindakan lainnya yang diperlukan sebelum aplikasi dapat berjalan dengan baik.

Berikut adalah beberapa poin penting tentang bootstrap pada Laravel:

1. File index.php: Proses bootstrap dimulai dengan file index.php, yang merupakan titik masuk utama aplikasi Laravel. File ini berfungsi untuk memuat autoload composer, mendaftarkan error handler, dan memulai aplikasi dengan memanggil metode run() pada class Illuminate\Foundation\Application.

2. Environment Configuration: Pada tahap bootstrap, Laravel akan membaca file .env yang berisi konfigurasi lingkungan (environment) aplikasi. File ini berisi variabel-variabel yang menentukan pengaturan seperti koneksi database, pengaturan cache, dan variabel-variabel lainnya. Laravel menggunakan pustaka Dotenv untuk membaca dan mengakses variabel-variabel dalam file .env.

3. Application Instance: Setelah membaca konfigurasi lingkungan, Laravel membuat instansi Application yang mewakili aplikasi Anda. Instansi ini merupakan pintu gerbang untuk mengakses fitur-fitur Laravel dan menyediakan metode untuk mengonfigurasi, mendaftarkan layanan, dan menjalankan aplikasi.

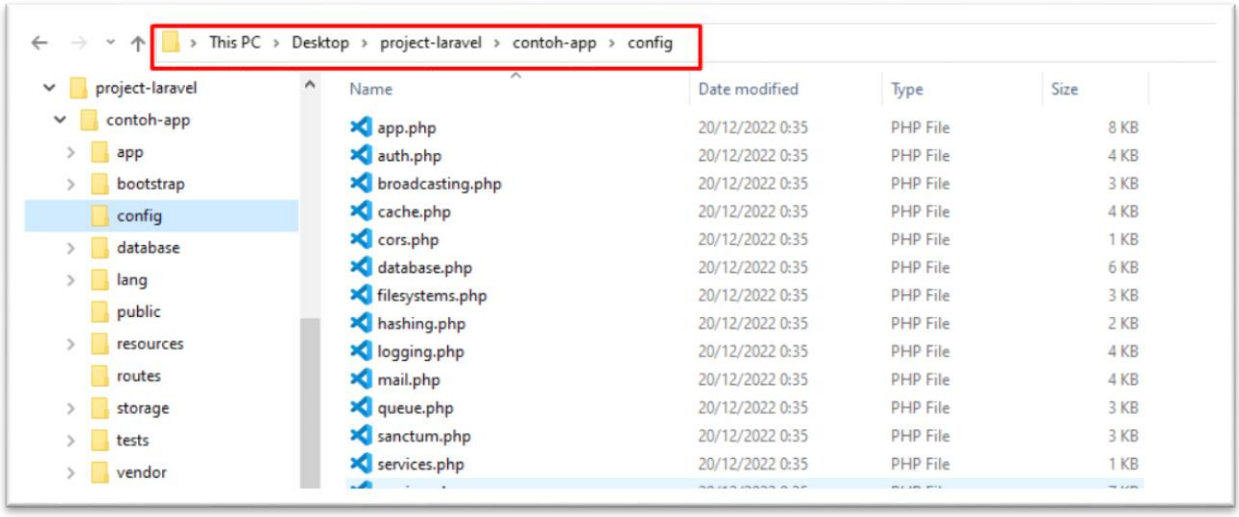
4. Service Container: Service container pada Laravel adalah komponen yang bertanggung jawab untuk mengelola dan mempertukarkan objek-objek dalam aplikasi. Pada tahap bootstrap, Laravel akan mendaftarkan berbagai layanan yang diperlukan oleh aplikasi, seperti database, cache, session, dan lainnya, ke dalam service container. Layanan-layanan ini kemudian dapat diakses secara mudah di berbagai bagian aplikasi.

5. Autoload dan Namespace: Laravel menggunakan mekanisme autoload untuk memuat kelas-kelas aplikasi secara otomatis. Ini memungkinkan Anda untuk menggunakan kelas-kelas tanpa harus secara manual memuatnya dengan perintah require. Autoload dilakukan berdasarkan namespace yang didefinisikan dalam aplikasi, dan file composer.json digunakan untuk mengatur autoload secara konfigurasi.

6. Application Middleware: Middleware pada Laravel adalah komponen yang digunakan untuk memodifikasi atau memproses permintaan HTTP sebelum mencapai tujuan akhirnya. Pada tahap bootstrap, Laravel akan mendaftarkan middleware bawaan yang termasuk dalam aplikasi, seperti middleware sesi, middleware autentikasi, dan lainnya. Middleware ini akan diterapkan pada permintaan sepanjang siklus aplikasi.

Proses bootstrap pada Laravel mempersiapkan aplikasi Anda untuk berjalan dengan baik. Dengan mengatur konfigurasi lingkungan, mendaftarkan layanan, menginisialisasi service container, dan memuat kelas-kelas secara otomatis, Laravel memberikan fondasi yang kokoh untuk pengembangan aplikasi web yang efisien dan fleksibel.

Config



Folder/direktori config mencakup berbagai konfigurasi dan parameter terkait diperlukan untuk kelancaran fungsi aplikasi Laravel. Berbagai files termasuk dalam folder config seperti yang ditunjukkan pada gambar di sini. Itu nama file berfungsi sesuai fungsi yang terkait dengannya.

Konfigurasi (config) pada Laravel mengacu pada pengaturan atau konfigurasi yang digunakan untuk mengonfigurasi berbagai aspek dalam aplikasi. Laravel menyediakan berkas konfigurasi yang dapat Anda gunakan untuk mengatur berbagai opsi, seperti pengaturan database, konfigurasi cache, pengaturan mail, dan banyak lagi.

Berikut adalah beberapa poin penting tentang konfigurasi pada Laravel:

1. Direktori Config: Berkas konfigurasi pada Laravel terletak di direktori **config** di dalam struktur proyek Laravel. Di dalam direktori ini, Anda akan menemukan berbagai berkas konfigurasi, seperti **app.php**, **database.php**, **mail.php**, dan lainnya. Setiap berkas konfigurasi berisi pengaturan khusus yang berkaitan dengan aspek tertentu dalam aplikasi.

2. Penyesuaian Konfigurasi: Anda dapat menyesuaikan berkas konfigurasi sesuai dengan kebutuhan aplikasi Anda. Untuk melakukan penyesuaian, Anda dapat membuka berkas konfigurasi yang sesuai dan mengubah nilai-nilai pengaturan yang ada di dalamnya. Misalnya, Anda dapat mengubah pengaturan koneksi database, mengatur driver cache yang digunakan, atau mengonfigurasi pengiriman email.

3. Konfigurasi Environment: Laravel juga mendukung konfigurasi berbasis lingkungan (environment-based configuration). Ini berarti Anda dapat menentukan pengaturan yang berbeda untuk setiap lingkungan, seperti pengaturan development, production, atau staging. Ini dilakukan melalui file **.env** yang berisi variabel-variabel konfigurasi yang dapat diatur sesuai dengan lingkungan yang berbeda.

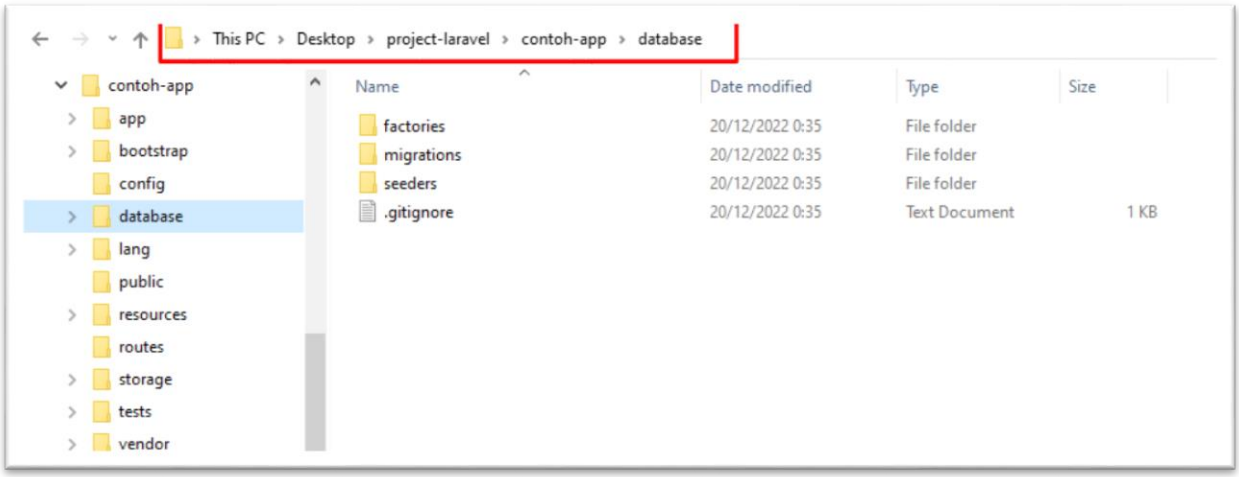
4. Variabel Env: File **.env** menyediakan cara untuk mengatur variabel-variabel konfigurasi menggunakan sintaks **KEY=VALUE**. Variabel-variabel ini dapat digunakan dalam berkas konfigurasi dengan menggunakan fungsi **env('KEY', 'default')**. Ini memungkinkan Anda untuk dengan mudah mengubah konfigurasi berdasarkan lingkungan tanpa harus mengubah berkas konfigurasi secara langsung.

5. Penggunaan Konfigurasi: Konfigurasi dalam Laravel digunakan oleh berbagai komponen dan layanan di dalam aplikasi. Misalnya, konfigurasi database digunakan oleh komponen Eloquent ORM untuk mengatur koneksi ke database. Konfigurasi cache digunakan oleh komponen Cache untuk mengatur penyimpanan cache yang digunakan. Demikian pula, konfigurasi mail digunakan oleh komponen Mail untuk mengatur pengiriman email.

6. Penyesuaian Lanjutan: Selain mengubah nilai pengaturan yang ada dalam berkas konfigurasi, Anda juga dapat menyesuaikan konfigurasi lebih lanjut dengan menambahkan pengaturan kustom di dalam berkas **config** yang baru. Anda dapat membuat berkas konfigurasi baru dan mengakses nilai pengaturan tersebut di dalam aplikasi menggunakan fungsi **config('KEY')**.

Konfigurasi pada Laravel memungkinkan Anda untuk mengatur berbagai aspek dalam aplikasi dengan mudah. Dengan mengubah berkas konfigurasi, menyesuaikan variabel **.env**, dan menggunakan nilai pengaturan dalam komponen dan layanan, Anda dapat mengkon

Database



Folder **database** pada Laravel berperan penting dalam pengaturan dan pengelolaan struktur basis data aplikasi. Di dalam folder ini, Anda akan menemukan beberapa subfolder dan berkas yang memiliki fungsi dan tujuan spesifik. Berikut adalah penjelasan lebih detail tentang folder **database** pada Laravel:

1. **migrations**: Folder **migrations** digunakan untuk menyimpan berkas migrasi. Migrasi adalah mekanisme yang disediakan oleh Laravel untuk mengelola struktur basis data secara terstruktur dan terdokumentasi. Setiap berkas migrasi mewakili satu set perubahan dalam struktur basis data, seperti pembuatan tabel baru, penambahan kolom, atau indeks baru. Migrasi memungkinkan Anda dan tim Anda untuk bekerja sama dalam membangun dan mengubah struktur basis data dengan mudah.

2. **seeds**: Folder **seeds** berisi berkas-berkas seeder. Seeder adalah mekanisme yang digunakan untuk mengisi data awal atau dummy ke dalam basis data. Seeder sangat berguna dalam pengembangan dan pengujian aplikasi, di mana Anda dapat dengan mudah membuat data awal yang diperlukan untuk menjalankan dan menguji fitur aplikasi. Anda dapat membuat berkas seeder baru di dalam folder ini dan menggunakan metode **seeder** pada class **DatabaseSeeder** untuk memanggil dan menjalankannya.

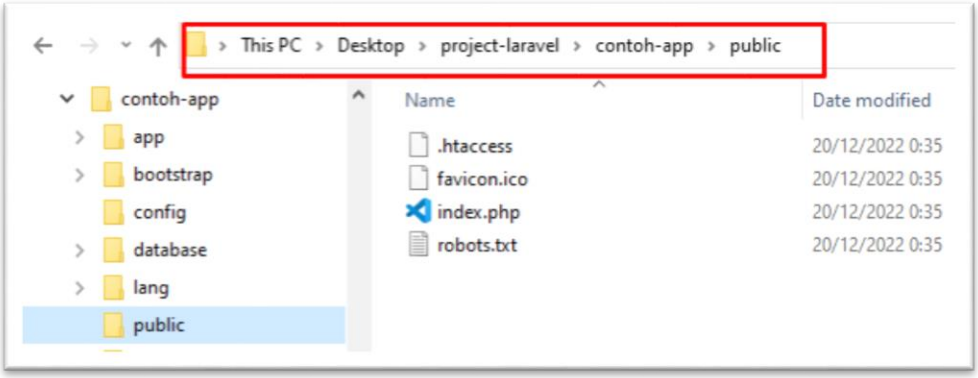
3. **factories**: Folder **factories** berisi berkas-berkas factory. Factory digunakan untuk menghasilkan data dummy atau acak untuk pengujian atau pengisian data. Dengan menggunakan factory, Anda dapat dengan mudah membuat data yang konsisten dan bervariasi untuk menguji berbagai skenario dalam aplikasi. Anda dapat membuat berkas factory baru di dalam folder ini dan menggunakan metode **factory** untuk menghasilkan data menggunakan model yang berkaitan.

4. **seeds**: Folder **seeds** berisi berkas-berkas seeder. Seeder adalah mekanisme yang digunakan untuk mengisi data awal atau dummy ke dalam basis data. Seeder sangat berguna dalam pengembangan dan pengujian aplikasi, di mana Anda dapat dengan mudah membuat data awal yang diperlukan untuk menjalankan dan menguji fitur aplikasi. Anda dapat membuat berkas seeder baru di dalam folder ini dan menggunakan metode **seeder** pada class **DatabaseSeeder** untuk memanggil dan menjalankannya.

5. **factories**: Folder **factories** berisi berkas-berkas factory. Factory digunakan untuk menghasilkan data dummy atau acak untuk pengujian atau pengisian data. Dengan menggunakan factory, Anda dapat dengan mudah membuat data yang konsisten dan bervariasi untuk menguji berbagai skenario dalam aplikasi. Anda dapat membuat berkas factory baru di dalam folder ini dan menggunakan metode **factory** untuk menghasilkan data menggunakan model yang berkaitan.
6. **sqlite** (opsional): Jika Anda menggunakan basis data SQLite, Laravel akan secara otomatis membuat folder **sqlite** di dalam folder **database**. Folder ini digunakan untuk menyimpan basis data SQLite yang digunakan dalam pengembangan lokal. File basis data SQLite akan disimpan di dalam folder ini.

Folder **database** pada Laravel menyediakan struktur dan mekanisme yang kuat untuk mengelola struktur dan data dalam basis data aplikasi. Dengan menggunakan migrasi, seeder, dan factory, Anda dapat dengan mudah

Public



Dalam konteks pengembangan web dengan framework Laravel, folder "public" adalah salah satu komponen penting dari struktur direktori proyek. Fungsi utama folder "public" adalah menyimpan semua file publik yang dapat diakses langsung oleh browser.

Secara umum, saat Anda menjalankan aplikasi Laravel, file yang berada di dalam folder "public" dapat diakses secara langsung melalui URL. Ini berarti Anda dapat menampilkan file tersebut, seperti gambar, CSS, JavaScript, atau file statis lainnya, di browser tanpa memerlukan pemrosesan melalui PHP atau Laravel itu sendiri.

Folder "public" adalah titik awal atau root dari aplikasi Laravel. Ketika permintaan dikirim ke aplikasi, server web umumnya mengarahkan permintaan tersebut ke file "index.php" yang terletak di dalam folder "public". File ini bertanggung jawab untuk memuat dan menjalankan aplikasi Laravel.

Struktur folder "public" biasanya terdiri dari file seperti "index.php", yang merupakan file pintu masuk untuk aplikasi Laravel, dan file ".htaccess" (jika Anda menggunakan server Apache) atau "web.config" (jika Anda menggunakan server IIS) yang mengatur konfigurasi server untuk mengarahkan semua permintaan ke file "index.php".

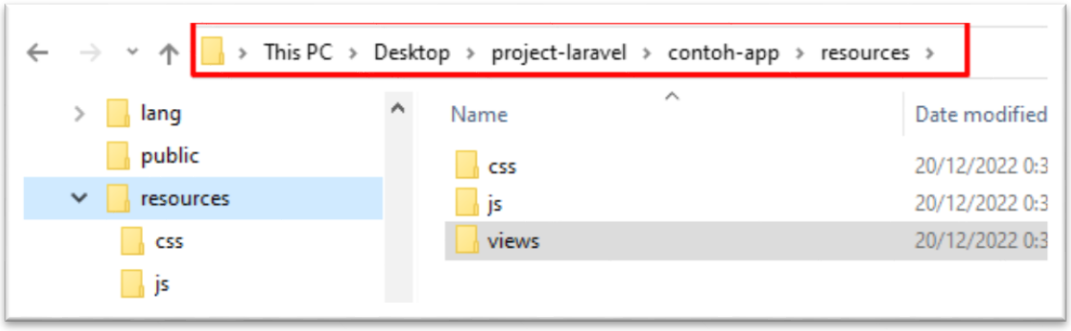
Selain itu, folder "public" juga sering digunakan untuk menyimpan aset publik seperti gambar, file CSS, file JavaScript, dan file media lainnya yang akan digunakan oleh aplikasi Anda. Misalnya, jika

Anda memiliki gambar dengan path "public/images/logo.png", Anda dapat mengaksesnya di browser dengan menggunakan URL "http://domainanda.com/images/logo.png".

Penting untuk diingat bahwa file-file yang berada di dalam folder "public" dapat diakses oleh publik secara langsung melalui URL. Jadi, jika Anda memiliki file sensitif atau data yang harus dijaga keamanannya, sebaiknya jangan letakkan file-file tersebut di dalam folder "public". Sebagai gantinya, letakkan file-file tersebut di folder "storage" atau gunakan konfigurasi khusus untuk mengamankan akses ke file tersebut.

Dengan menggunakan folder "public" dalam Laravel, Anda dapat dengan mudah mengelola aset publik Anda dan memastikan bahwa mereka dapat diakses dengan benar oleh browser.

Resources



Folder "resources" dalam Laravel merupakan direktori yang berisi berbagai jenis sumber daya yang digunakan dalam pengembangan aplikasi. Folder ini memiliki peran penting dalam menyimpan template, aset statis, bahasa lokal, dan file-file lain yang diperlukan oleh aplikasi.

Berikut adalah beberapa subdirektori yang umumnya ada di dalam folder "resources":

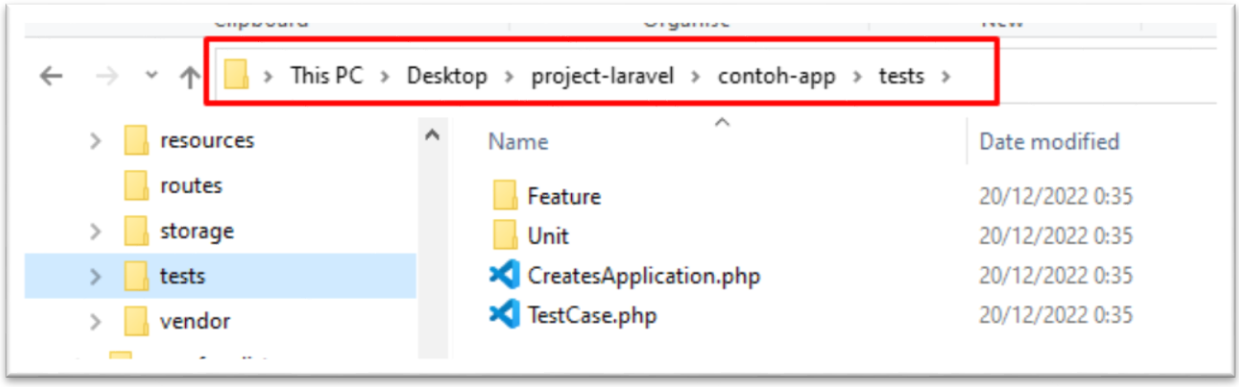
1. views: Folder ini digunakan untuk menyimpan file-template berbasis Blade. Template Blade digunakan untuk membangun tampilan aplikasi dengan sintaks yang mudah dipahami dan ekspresif.
2. lang: Folder ini berisi file-file lokal aplikasi yang digunakan untuk lokalizing aplikasi dalam berbagai bahasa. Anda dapat menyimpan file-file bahasa yang berbeda di dalam subdirektori yang sesuai (misalnya, "en" untuk bahasa Inggris, "id" untuk bahasa Indonesia).
3. assets: Folder ini digunakan untuk menyimpan aset statis seperti file CSS, JavaScript, dan gambar. Anda dapat mengatur struktur direktori sesuai kebutuhan Anda, misalnya membagi file CSS dan JavaScript ke dalam folder yang terpisah.
4. js: Subdirektori ini secara khusus digunakan untuk menyimpan file JavaScript yang digunakan dalam aplikasi Anda.
5. sass atau css: Subdirektori ini digunakan untuk menyimpan file-file yang berisi kode SASS atau CSS. Anda dapat memilih untuk menggunakan SASS (Syntactically Awesome Stylesheets) untuk mengorganisir dan memperluas kemampuan CSS.
6. database: Folder ini berisi file-file migrasi database dan file factory untuk seeding. File migrasi digunakan untuk membuat dan mengubah struktur database aplikasi Anda,

sedangkan file factory digunakan untuk mengisi database dengan data dummy saat pengembangan.

7. tests: Folder ini digunakan untuk menyimpan file-file pengujian (testing) aplikasi. Anda dapat menggunakan Laravel Dusk atau PHPUnit untuk menguji berbagai komponen aplikasi, seperti rute (routes), model, dan pengontrol (controller).

Folder "resources" memainkan peran penting dalam pengembangan aplikasi Laravel karena ini adalah tempat sentral untuk menyimpan berbagai sumber daya yang dibutuhkan oleh aplikasi. Dengan struktur direktori yang terorganisir dengan baik, Anda dapat dengan mudah mengelola dan mengakses sumber daya tersebut saat membangun aplikasi Laravel yang kuat dan bersih.

Tests



Folder "tests" dalam Laravel adalah direktori yang digunakan untuk menyimpan file-file pengujian (testing) aplikasi. Pengujian adalah bagian penting dari pengembangan perangkat lunak yang memungkinkan Anda untuk memverifikasi bahwa kode Anda berfungsi seperti yang diharapkan dan menjaga kualitas aplikasi Anda.

Berikut adalah beberapa poin penting tentang folder "tests" dalam Laravel:

1. Struktur Direktori: Folder "tests" terletak di direktori utama proyek Laravel dan biasanya memiliki struktur direktori bawaan. Beberapa subdirektori yang umumnya ada di dalamnya adalah sebagai berikut:
    - Feature: Folder ini berisi pengujian yang lebih tingkat tinggi yang mencakup beberapa bagian sistem atau fitur utama dalam aplikasi.
    - Unit: Folder ini berisi pengujian unit yang lebih kecil dan fokus pada pengujian komponen individu, seperti model, pengontrol (controller), atau helper.
    - Browser: Folder ini digunakan khusus untuk pengujian yang melibatkan interaksi dengan peramban web menggunakan Laravel Dusk.
    - Console: Folder ini digunakan untuk pengujian pada perintah (command) dan tugas-tugas konsol di Laravel.
2. File Pengujian: File-file pengujian umumnya menggunakan ekstensi ".php" dan biasanya diberi nama dengan format yang deskriptif untuk memudahkan pemahaman tujuan

pengujian. Misalnya, "ExampleTest.php" atau "UserRegistrationTest.php". Anda dapat membuat file pengujian baru di direktori yang sesuai dengan tingkat pengujian yang Anda lakukan.

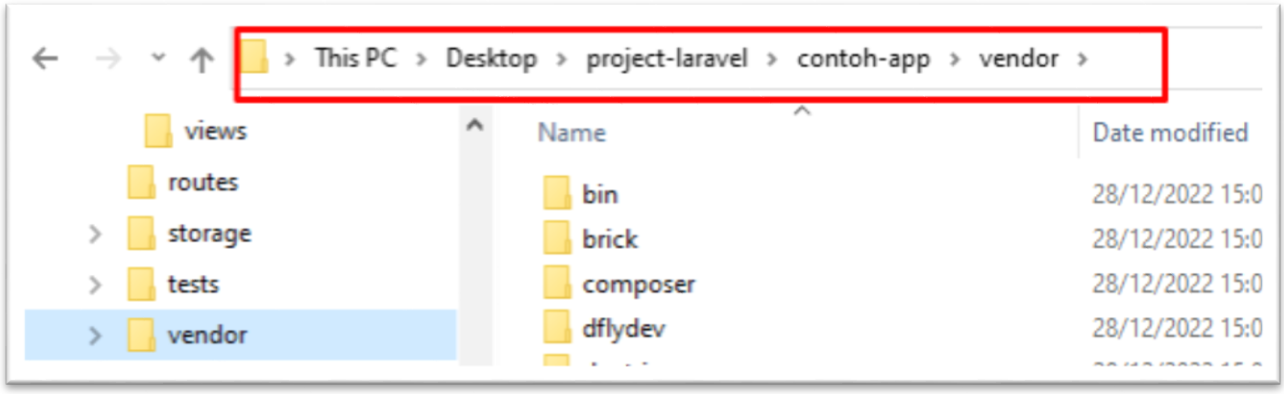
3. Framework Pengujian: Laravel menyediakan framework pengujian yang kuat yang memudahkan Anda untuk menulis dan menjalankan pengujian. Dalam pengujian unit, Anda dapat menggunakan metode dan fungsi bawaan seperti "assertEquals()", "assertTrue()", dan lainnya untuk memverifikasi hasil pengujian. Dalam pengujian fitur, Anda dapat menggunakan fasilitas Laravel seperti "actingAs()" untuk melakukan autentikasi pengguna atau "get()" dan "post()" untuk menguji rute (routes).

4. Menjalankan Pengujian: Anda dapat menjalankan semua pengujian dengan menjalankan perintah "php artisan test" di terminal Anda. Perintah ini akan menjalankan semua file pengujian yang ada di dalam folder "tests" dan memberikan laporan hasil pengujian.

Folder "tests" sangat berguna untuk memastikan bahwa aplikasi Laravel Anda berfungsi dengan baik dan sesuai dengan harapan. Dengan menulis pengujian yang baik, Anda dapat meningkatkan kualitas kode Anda, mengidentifikasi dan memperbaiki bug, serta menjaga stabilitas dan keandalan aplikasi Anda seiring waktu.

Semua kasus uji unit disertakan dalam direktori ini. Konvensi penamaan untuk penamaan kelas kasus uji adalah camel_case dan msengikuti konvensi sesuai fungsionalitas kelas.

Vendor



Folder "vendor" dalam Laravel adalah direktori yang berisi semua dependensi pihak ketiga yang diperlukan oleh proyek Laravel Anda. Saat Anda menginstal paket-paket melalui Composer, file-file dan kode dari paket-paket tersebut akan didownload dan disimpan di folder "vendor".

Berikut adalah beberapa poin penting tentang folder "vendor" dalam Laravel:

1. Struktur Direktori: Folder "vendor" biasanya terletak di direktori utama proyek Laravel, tepat di samping folder "app", "config", dan lainnya. Saat Anda menginstal paket melalui Composer, Composer akan membuat struktur direktori di dalam folder "vendor" sesuai dengan namespace atau penamaan paket yang diberikan oleh masing-masing paket.

2. Dependensi Pihak Ketiga: Folder "vendor" menyimpan semua dependensi atau paket pihak ketiga yang digunakan oleh proyek Laravel Anda. Ini mencakup pustaka dan komponen yang

diperlukan oleh Laravel itu sendiri, serta paket-paket tambahan yang Anda tambahkan melalui Composer, seperti library PHP atau paket JavaScript.

3. Autoloading: Laravel menggunakan autoloading yang disediakan oleh Composer untuk memuat kelas-kelas yang ada dalam folder "vendor". Ini berarti Anda tidak perlu secara manual memuat atau menyertakan file kelas dari direktori "vendor" saat menggunakan paket-paket yang ada di dalamnya. Autoloading akan memuat kelas secara otomatis saat diperlukan.

4. Penanganan Versi: Composer mengelola versi paket-paket yang ada dalam folder "vendor" melalui file "composer.lock". File ini berisi daftar lengkap dependensi dan versi yang spesifik yang digunakan dalam proyek Anda. Ini memastikan konsistensi dalam pengembangan dan penerapan proyek Laravel Anda, serta memudahkan tim Anda untuk bekerja dengan versi yang sama.

5. Tidak Dilakukan Perubahan Langsung: Penting untuk dicatat bahwa folder "vendor" seharusnya tidak dimodifikasi secara langsung. Composer akan secara otomatis mengelola folder ini dan memperbarui paket-paket sesuai dengan konfigurasi yang ada dalam file "composer.json". Jika Anda perlu menyesuaikan atau memodifikasi paket-paket yang ada, sebaiknya lakukan melalui file "composer.json" dan jalankan perintah Composer yang sesuai.

Folder "vendor" adalah komponen kunci dalam ekosistem Laravel yang memungkinkan Anda untuk mengelola dependensi pihak ketiga dengan mudah dan mengintegrasikannya ke dalam proyek Anda. Dengan mengandalkan Composer untuk mengelola paket-paket tersebut, Anda dapat menghindari kerumitan pengelolaan dependensi secara manual dan mengoptimalkan efisiensi pengembangan aplikasi Laravel.

4. KONFIGURASI LARAVEL

sebelumnya, kita telah melihat bahwa file konfigurasi dasar dari Laravel termasuk dalam direktori config. Pada bab ini, mari kita bahas tentang kategori termasuk dalam konfigurasi.

Konfigurasi Laravel melibatkan serangkaian pengaturan yang memungkinkan Anda menyesuaikan perilaku dan fitur framework sesuai dengan kebutuhan proyek Anda. Beberapa konfigurasi utama yang dapat Anda lakukan dalam Laravel meliputi:

1. File .env: File ".env" adalah file konfigurasi utama dalam Laravel yang berisi variabel lingkungan (environment variables) yang digunakan untuk mengonfigurasi aplikasi. Anda dapat mengatur parameter seperti koneksi database, pengaturan cache, URL, dan kunci enkripsi di file ini. Pastikan untuk tidak menyimpan informasi sensitif seperti password langsung di file .env dan gunakan .env.example sebagai panduan untuk mengonfigurasi file .env.

2. File config/app.php: File "config/app.php" berisi konfigurasi umum untuk aplikasi Laravel. Di sini, Anda dapat mengatur parameter seperti timezone, locale, nama aplikasi, dan pengaturan lainnya. Anda juga dapat menambahkan penyedia layanan (service providers) dan alias kelas (class aliases) dalam file ini.

3. File config/database.php: File "config/database.php" digunakan untuk mengonfigurasi koneksi database aplikasi Anda. Anda dapat menentukan pengaturan seperti driver database, koneksi, dan pilihan spesifik untuk masing-masing driver. Pastikan untuk mengatur detail koneksi database Anda dengan benar di file ini.
4. File config/cache.php: File "config/cache.php" memungkinkan Anda mengonfigurasi sistem cache dalam Laravel. Anda dapat memilih driver cache yang akan digunakan, seperti memori, file, database, atau cache eksternal seperti Redis atau Memcached. Anda juga dapat mengatur pengaturan waktu kedaluwarsa cache dan opsi lainnya.
5. File config/session.php: File "config/session.php" digunakan untuk mengonfigurasi pengaturan sesi dalam Laravel. Anda dapat mengatur driver sesi (misalnya, file, database, cookie), waktu kadaluwarsa sesi, penyimpanan sesi, dan pengaturan lainnya.
6. File config/logging.php: File "config/logging.php" memungkinkan Anda mengonfigurasi sistem pencatatan (logging) dalam Laravel. Anda dapat menentukan pengaturan seperti driver logging, level log yang diizinkan, dan tujuan log (file, basis data, Slack, email, dll).
7. File config/mail.php: File "config/mail.php" digunakan untuk mengonfigurasi pengiriman email dalam Laravel. Anda dapat mengatur driver email (SMTP, Mailgun, Sendmail, dll), pengaturan koneksi, dan opsi lainnya untuk mengirim email melalui aplikasi Anda.

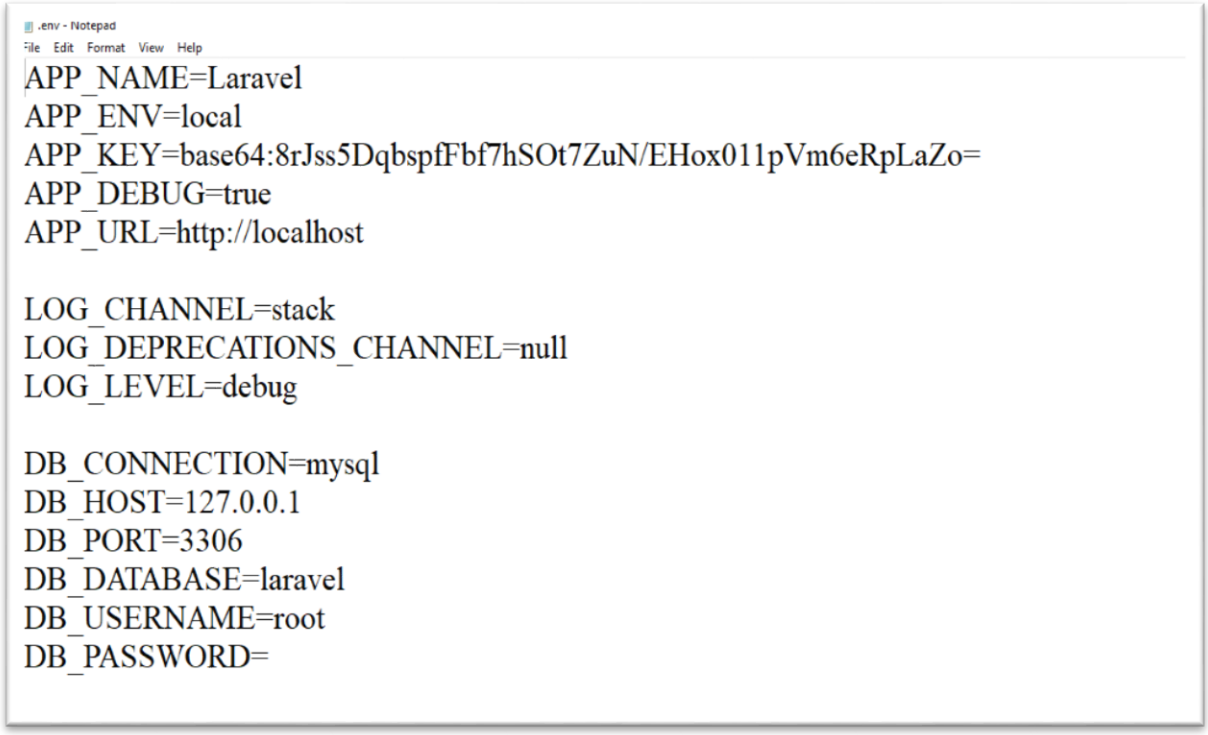
Selain konfigurasi di atas, Laravel juga menyediakan berbagai file konfigurasi lainnya untuk komponen seperti antrian (queues), autentikasi (authentication), notifikasi (notifications), dan banyak lagi. Anda dapat menyesuaikan konfigurasi ini sesuai dengan kebutuhan proyek Anda dengan mengedit file konfigurasi yang sesuai.

Penting untuk memperhatikan bahwa ketika melakukan perubahan pada file konfigurasi, pastikan untuk menjaga keamanan dan integritas aplikasi Anda. Selalu simpan salinan cadangan (backup) file konfigurasi sebelum melakukan perubahan, dan pastikan untuk mengonfigurasi dengan benar agar sesuai dengan lingkungan produksi Anda.

Environment Configuration

adalah variabel yang menyediakan daftar layanan web untuk aplikasi web kita. Semua variabel lingkungan dideklarasikan dalam file .env yang menyertakan parameter yang diperlukan untuk menginisialisasi konfigurasi.

Secara default, file .env menyertakan parameter berikut



Saat bekerja dengan file konfigurasi dasar Laravel, poin-poin berikut perlu diperhatikan

- File .env tidak boleh dikomit ke sumber aplikasi kontrol, karena setiap pengembang atau pengguna memiliki beberapa yang telah ditentukan sebelumnya konfigurasi lingkungan untuk aplikasi web.
- Untuk opsi pencadangan, tim pengembangan harus menyertakan file .env.example file, yang harus berisi default konfigurasi

Semua variabel lingkungan yangg dideklarasikan dalam file .env dapat diakses oleh fungsi env-helper yang akan memanggil parameter masing-masing. Variabel-variabel ini juga dicantumkan ke dalam variabel global $_ENV setiap kali aplikasi menerima permintaan dari ujung pengguna. Anda dapat mengakses lingkungan variabel seperti yang ditunjukkan di bawah ini

```
'env' => env('APP_ENV', 'production'),
```

fungsi env-helper dipanggil dalam file app.php yang disertakan dalam konfigurasi Folder. Contoh yang diberikan di atas memanggil parameter lokal dasar

Mengakses Nilai Konfigurasi

Kita dapat dengan mudah mengakses nilai konfigurasi di mana saja dalam aplikasi menggunakan fungsi pembantu konfigurasi global. Jika nilai konfigurasi adalah tidak diinisialisasi, nilai default dikembalikan.

Misalnya, untuk menyetel zona waktu default, maka kode berikut yang digunakan

```
config(['app.timezone' => 'Asia/Kolkata']);
```

Caching Konfigurasi

Untuk meningkatkan kinerja dan meningkatkan aplikasi web, itu penting untuk menyimpan semua nilai konfigurasi. Perintah untuk menyimpan file nilai konfigurasi adalah

```
config:cache
```

Berikut adalah conoh penggunaan cache

```
Gudangsoft@DESKTOP-SOKJ64L MINGW64 ~/Desktop/project-laravel/contoh-app
$ php artisan config:cache

  INFO  Configuration cached successfully.
```

Maintenance Mode

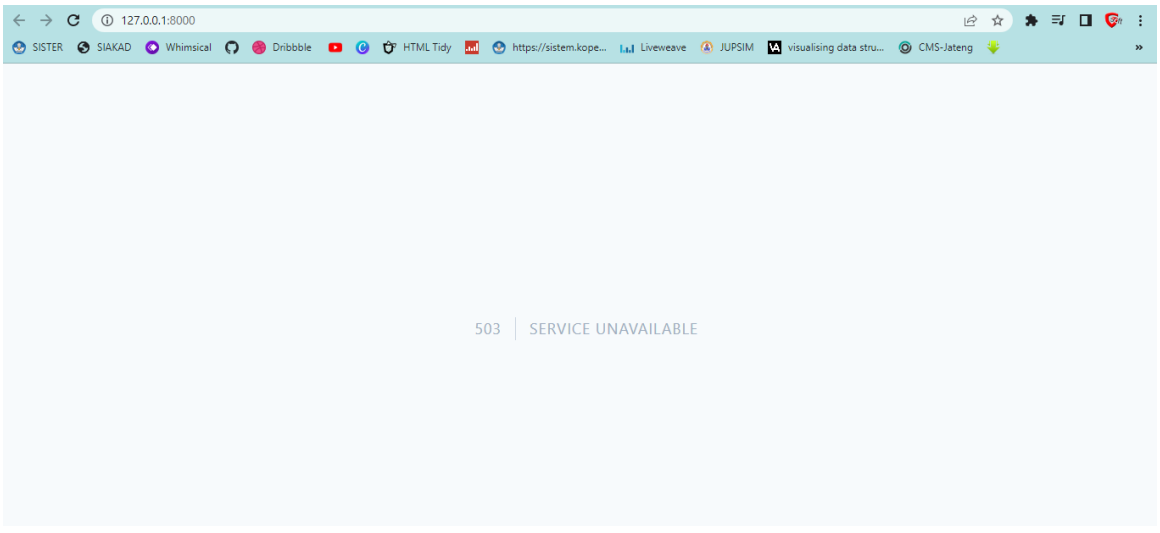
Terkadang kita perlu memperbarui beberapa nilai konfigurasi atau perform pemeliharaan di situs web yang sudah kita buat . Dalam kasus seperti itu, untuk menjaganya dalam mode pemeliharaan, sangat memudahkan kita. Aplikasi web seperti itu yang disimpan di mode pemeliharaan, berikan pengecualian yaitu MaintenanceModeException dengan kode status 503.

Kita dapat mengaktifkan mode pemeliharaan pada aplikasi web Laravel Anda menggunakan

perintah berikut php artisan down

```
Gudangsoft@DESKTOP-SOKJ64L MINGW64 ~/Desktop/project-laravel/contoh-app
$ php artisan down

  INFO  Application is now in maintenance mode.
```

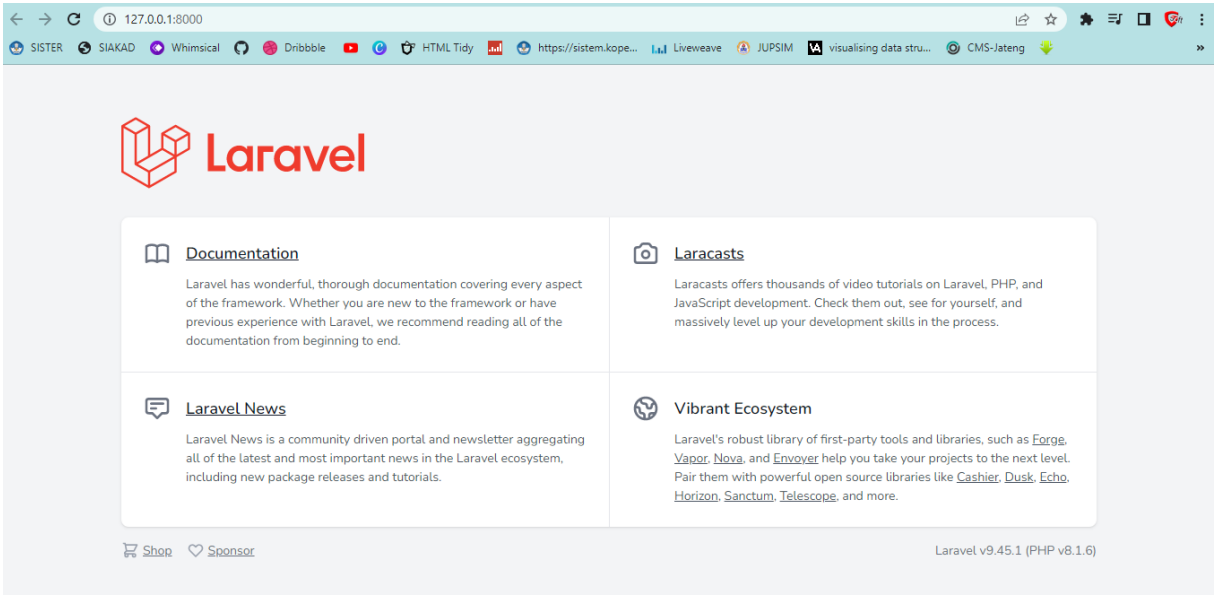
Maka Ketika kita membuka website dibrowser akan tampil sebagai berikut



Untuk mengembalikan, bisa menggunakan perintah php artisan up

```
Gudangsoft@DESKTOP-SOKJ64L MINGW64 ~/Desktop/project-laravel/contoh-app
$ php artisan up

  INFO  Application is now live.
```

Selanjutnya tampilan website kita akan normal Kembali.



5. CRUD

CRUD (Create, Read, Update, Delete) pada Laravel merupakan sebuah operasi manipulasi dan pengolahan data di dalam sebuah basis data yang bisa dilangsungkan dengan Laravel. CRUD Laravel merupakan singkatan (akronim) dari Create (buat), Read (baca/menampilkan), Update (memperbaharui), dan Delete (menghapus) data dari database.

Seluruh aktivitas ini merupakan kegiatan-kegiatan dasar dari pengolahan data di database, jadi, Anda pun wajib untuk bisa menguasainya. Dengan mempelajari tutorial CRUD Laravel, Anda bisa mengolah aneka data di database dan menampilkannya dengan lebih rapi. Data yang rapi pun akan lebih mudah untuk dicerna dan dipahami oleh user dan berikut kita akan membahas langkah demi langkah membuat CRUD.

- **Installasi & Perisiapan Laravel 9**

  Untuk membuat CRUD Pada bagian awal yang kita harus kita lakukan adalah install Laravel 9 dan composer seperti yang sudah kita bahas pada bab sebelumnya. Namun karena kita akan menambahkan gambar pada contoh CRUD yang aka kita buat, kita harus menjalankan perintab **artisan storage:link ,** Perintah ini berfungsi membuat sebuah link dari forlder **storage** kedalam folder Public di Laravel kita.

- **Membuat dan Menjalankan Migration**

Selesai melakukan installasi Langkah selanjutnya adalah melakukan **migration,** migrasion sendiri merupakan sebuah versi control database yang berfungsi untuk membatu team jika ingin merubah dan membagikan skema database dari aplikasi yang akan dibangun.
Mungkin sebelumnya kita sudah pernah membuat table pada projek kita dengan cara manual, maka dengan Migration kita tidak perlu susah sudah membuat seperti itu lagi.
Langkah untuk melakukan migarasi :

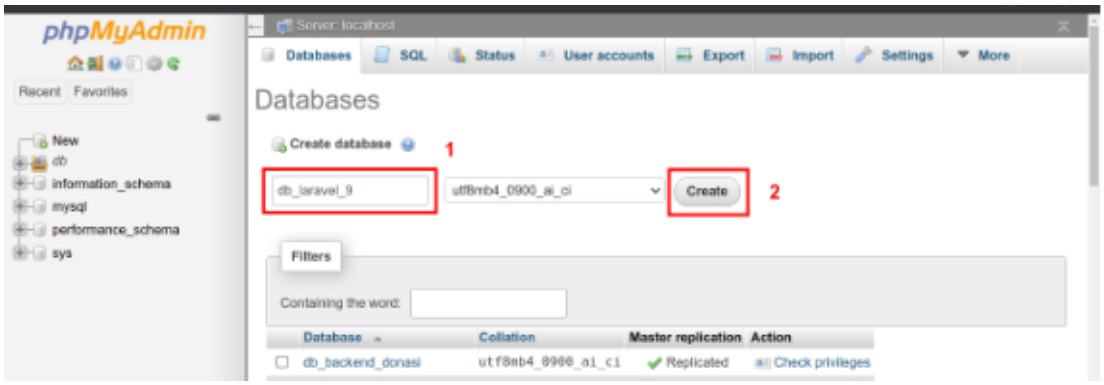
- **Konfigurasi Koneksi Database**
  Untuk melakukan konfigurasi silahkan anda buka file **.inv** menggunakan teks editor favorit anda, silahkan cari kode seperti dibawah ini

```
DB_DATABASE=laravel
DB_USERNAME=root
DB_PASSWORD=
```

  Kemudian silahkan rubah menjadi

```
DB_DATABASE=db_laravel_9
DB_USERNAME=root
DB_PASSWORD=
```

  Penjelasan kode diatas adalah kita mengatur untuk **DB_DATABASE** dan mengisi dengan database yang kita buat yaitu **db_laravel_9**. Selanjutnya untuk bairkan default **root** dan untuk **DB_PASSWORD** silahkan sesuaikan dengan keinginan anda, jika menggunakan XAMPP maka sebaiknya dikosongkan saja.
  Jika sudah selesai melakukan konfigurasi, silahkan melakukan resrtart server.
- **Membuat Database**
  Setelah selesai melakukan konfigurasi, Langkah selanjutnya adalah membuat database. Untuk membuat database silahkan buka link http://localhost/phpmyadmin pada browser anda, kemudian bautlah database baru dengan nama **db_laravel_9**



- **Membuat Model dan Migration**
  Selesai membuat database di phpMyAdmin, Langkah selanjutnya adalah membuat model dan Migration. Silahkan anda jalankan perintah seperti dibawah ini

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

  Perintah tersebut digunakan untuk membuat sebuah model baru dengan menggunakan nama **Post** dan menambahkan flag **-m** yang memiliki arti file migration juga ikut dibuat.

Setelah berhasil menjalan perintah diatas maka kita akan meiliki 2 file baru, yaitu :

- app/Models/Post.php
- database/migrations/2022_02_09_003543_create_posts_table.php

- **Menambahkan Field dalam Tabel**

  Jika sudah selesai membuat model dan migration selanjutnya kita akan menambahkan filed atau kolom dalcam file migration dan filed tersebut akan kita generate dalam table posts. Silahkan buka file migrations dan rubah isi pada bagian **function up** menjadi

```
public function up()
{
  Schema::create('posts', function (Blueprint $table) {
    $table->id();
    $table->string('image');
    $table->string('title');
    $table->text('content');
    $table->timestamps();
  });
}
```

Maka kita sudah menambahkan field baru yaitu

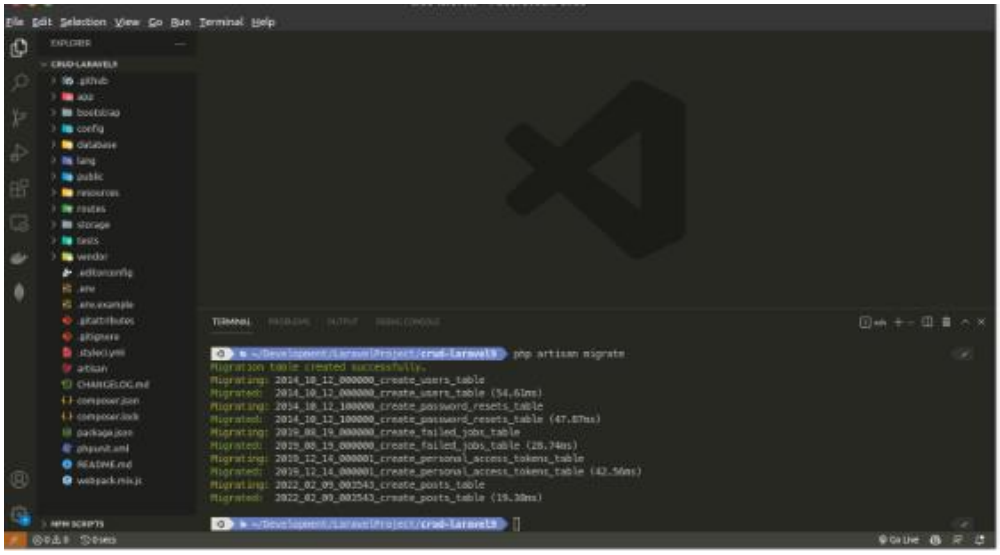
FIELD	TYPE DATA
image	string
title	string
content	text

- **Menjalankan Migration**

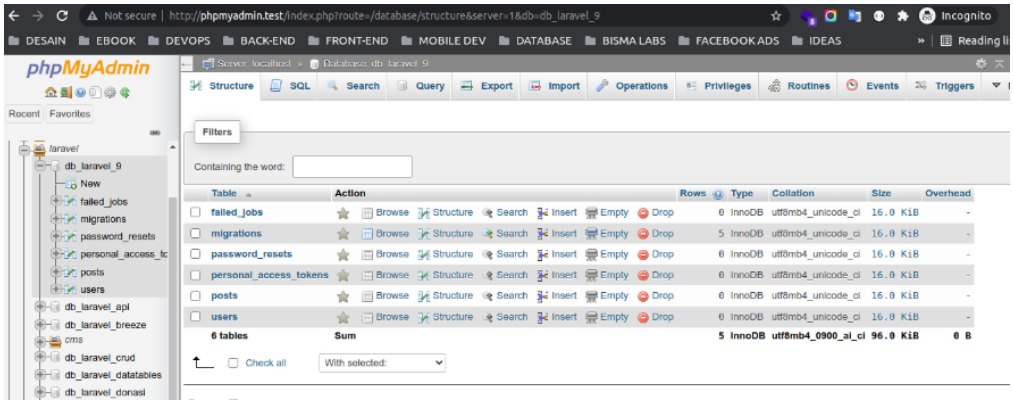
  Tahap selanjutnya adalah menjalatkan perintah migrate yang digunakan untuk melakukan proses pemzbentukan tebel posts beserta isi fieldnya kedalam sebuah database yang sudah kita buat sebelumnya.

```
php artisan migrate
```

Perintah diatas akan menghasilkan



Silahkan cek adatabase kita, makan akan terlihat seperti gambar dibawah



- **Konfigurasi Mass Assigment**

  Fungsi Mass Assigment adalah untuk mengizinkan sebuah field agar dapat meyimpan data.

  Silahkan kita buka file pada **app/Models/Post.php** selanjutnya kita rubah kodenya menjadi berikut ini :

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Post extends Model
{
    use HasFactory;

    /**
     * fillable
     *
     * @var array
     */
    protected $fillable = [
        'image',
        'title',
        'content',
    ];
}
```

Dari kode diatas kita telah menambahkan sebuah property yang Bernama **$fillable** yang berfungsi untuk menabahkan field-field yang telah di izinkan untuk melakukan sebuah manipulasi data.

- **Menampilkan Data dari Database**

Selanjunya, untuk menampilkan data dari sebuah database ada beberapa Langkah yang harus kita lakukan, diantaranya adalah:

- **Membuat Sebuah Controller Post**

  Untuk membuat sebuah controller silahkan ketikkan perintah dalam CMD

  ```
  php artisan make:controller PostController
  ```

  Selanjutnya kita akan mendapatkan file controller dengan nama **PostController.php** dalam sebuah Folder **app/Http/Controllers**. Selanjutnya silahkan buka file tersebut dan rubah kodenya seperti dibawah ini:

```php
<?php

namespace App\Http\Controllers;

use App\Models\Post;
use Illuminate\Http\Request;

class PostController extends Controller
{
    /**
     * index
     *
     * @return void
     */
    public function index()
    {
        //get posts
        $posts = Post::latest()->paginate(5);

        //render view with posts
        return view('posts.index', compact('posts'));
    }
}
```

Selanjutnya silahkan import model Post terlebih dahulu

```php
use App\Models\Post;
```

Jika sudah, silahkan membuat method baru dengan nama **index**

```php
public function index()
{


//...


}
```

Di dalam method index kita get data post dari database melalui model Post.

```php
//get posts
$posts = Post::latest()->paginate(5);
```

- **Menambahkan Route**

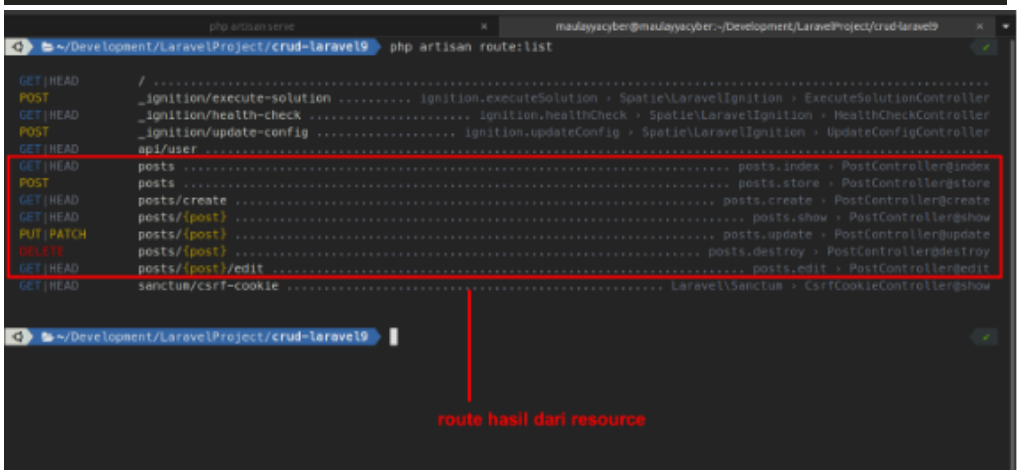
  Disini kita akan menggunakan route dengan type **resource** yang dapat diartikan route tersebut akan berisi beberapa route – route daclam kebutuhan CRUD, Seperti hanya Index, Create,storage,show,edit,update serta destroy yang tentunya akan membuat waktu lebih hemat daripada kita harus membuat secara manula satu persatu.

Silahkan buka file **routes/web.php** dan ubah semua kodenya seperti dibawah ini :

```php
use Illuminate\Support\Facades\Route;

Route::get('/', function () {
    return view('welcome');
});

//route resource
Route::resource('/posts', \App\Http\Controllers\PostController::class);
```

Selanjutnya silahkan jalankan

```
php artisan route:list
```



- **Membuat View dan Menampilkan Data**

  Untuk membuats sebuah views yang akan menampilkan sebuah data pada website , silahkan membuat sebuah folder dengavn nama **posts** dalam folder **resource/views** dan juga dalam folder **posts** lalu buat file dengan nama **index.blade.php** dan tambahkan kode seperti dibawah ini :

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Data Posts - SantriKoding.com</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/b
    <link rel="stylesheet" href="//cdnjs.cloudflare.com/ajax/libs/toastr.js/latest/toastr
</head>
<body style="background: lightgray">
```

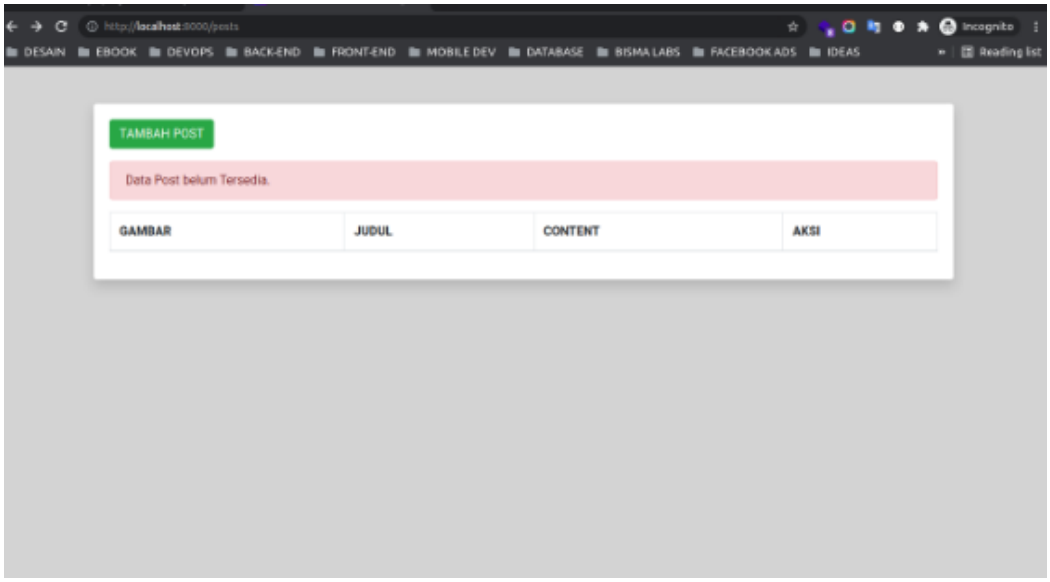
```
                            @forelse ($posts as $post)
                              <tr>
                                  <td class="text-center">
                                      <img src="{{ Storage::url('public/posts/').$post
                                  </td>
                                  <td>{{ $post->title }}</td>
                                  <td>{!! $post->content !!}</td>
                                  <td class="text-center">
                                      <form onsubmit="return confirm('Apakah Anda Yaki
                                          <a href="{{ route('posts.edit', $post->id) }}
                                          @csrf
                                          @method('DELETE')
                                          <button type="submit" class="btn btn-sm btn-
                                      </form>
                                  </td>
                              </tr>
                            @empty
                                <div class="alert alert-danger">
                                    Data Post belum Tersedia.
                                </div>
                            @endforelse
                          </tbody>
                        </table>
                        {{ $posts->links() }}
                    </div>
                </div>
            </div>
        </div>
    </div>
```

```
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/jquery.min.js"></scr
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js">
    <script src="//cdnjs.cloudflare.com/ajax/libs/toastr.js/latest/toastr.min.js"></scrip

    <script>
        //message with toastr
        @if(session()->has('success'))

            toastr.success('{{ session('success') }}', 'BERHASIL!');

        @elseif(session()->has('error'))

            toastr.error('{{ session('error') }}', 'GAGAL!');

        @endif
    </script>

</body>
</html>
```

Dengan membuat kode diatas kita telah melakukan perulangan untuk menapilkan sebuah data yang diksirim melalui controller.

- **Menjalakan Menapilkan Data**

  Untuk menapilkan hasilnya silahkan anda ketikan **php artisan serve** maka broser akan terbuka http://localhost:8080 lalu tambahkan post sehingga menjadi http://localhost:8080/post maka akan tampil seperti dibawah ini :



- **Inputkan Data kedalam sebuah Database**

  Pada tahapan ini kita akan membuat sebuah form untuk menambahkan data dalam sebauh database.

  Ada beberapa Langkah yang harus kita lakukan, diatantara adalah :

  - **Menambahkan Methode Create Storage**

    Bukalah file yang terletak **dalam app/Http/Controller/PostController.php** selanjutnya kitamerubah kodenya menjadi seperti dibawah ini:

```php
<?php

namespace App\Http\Controllers;

use App\Models\Post;
use Illuminate\Http\Request;

class PostController extends Controller
{
    /**
     * index
     *
     * @return void
     */
    public function index()
    {
        //get posts
        $posts = Post::latest()->paginate(5);

        //render view with posts
        return view('posts.index', compact('posts'));
    }
```

```php
/**
 * create
 *
 * @return void
 */
public function create()
{
    return view('posts.create');
}

/**
 * store
 *
 * @param Request $request
 * @return void
 */
public function store(Request $request)
{
    //validate form
    $this->validate($request, [
        'image'     => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048',
        'title'     => 'required|min:5',
        'content'   => 'required|min:10'
    ]);
```

```php
        //upload image
        $image = $request->file('image');
        $image->storeAs('public/posts', $image->hashName());

        //create post
        Post::create([
            'image'     => $image->hashName(),
            'title'     => $request->title,
            'content'   => $request->content
        ]);

        //redirect to index
        return redirect()->route('posts.index')->with(['success' => 'Data Berhasil Disimpan!']);
    }
}
```

Dari kode yang sudah kita buat diatas kita telah menambahkan 2 method baru **create** yang digunakan untuk menampilkan halaman form yang berfungsi untuk menginput sebuah data dan **store** yang berfungsi untuk memproses data yang akan disimpan kedalam database.

Selanjutnya kita akan membuat sebuah validasi yang dapat kita gunakan untuk mengecek apakah data yang kita simpan sudah sesuai.

```php
$this->validate($request, [
    'image'     => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048',
    'title'     => 'required|min:5',
    'content'   => 'required|min:10'
]);
```

Keterangan dari kode diatas adalah :

KEY	VALIDATION	KETERANGAN
`image`	required	field wajib diisi.
	image	field harus berupa gambar
	mimes:jpeg,png,jpg,gif,svg	filed harus memiliki extensi `jpeg`, `png`, `jpg`, `gif` dan `svg`.
	max:2048	field maksimal berukuran 2048 Mb / 2Mb.
`title`	required	field wajib diisi.
	min:5	field minimal memiliki 5 karakter/huruf.
`content`	required	field wajib diisi.
	min:10	field minimal memiliki 10 karakter/huruf.

Jika data yang dikirm sudah sesuai dengan validasi maka kita lanjutkan dengan proses upload gambar

```
$image = $request->file('image');
$image->storeAs('public/posts', $image->hashName());
```

Jika sudah berhasil diupload selanjutnya kita tambahkan data baru dalam database

```
//create post
Post::create([
    'image'     => $image->hashName(),
    'title'     => $request->title,
    'content'   => $request->content
]);
```

Selanjutnya kita redirect kedalan route post.index dengan memberikan sebuah pesan **return redirect()->route('posts.index')->with(['success' => 'Data Berhasil Disimpan!']);**

- **Membuat Halaman View**

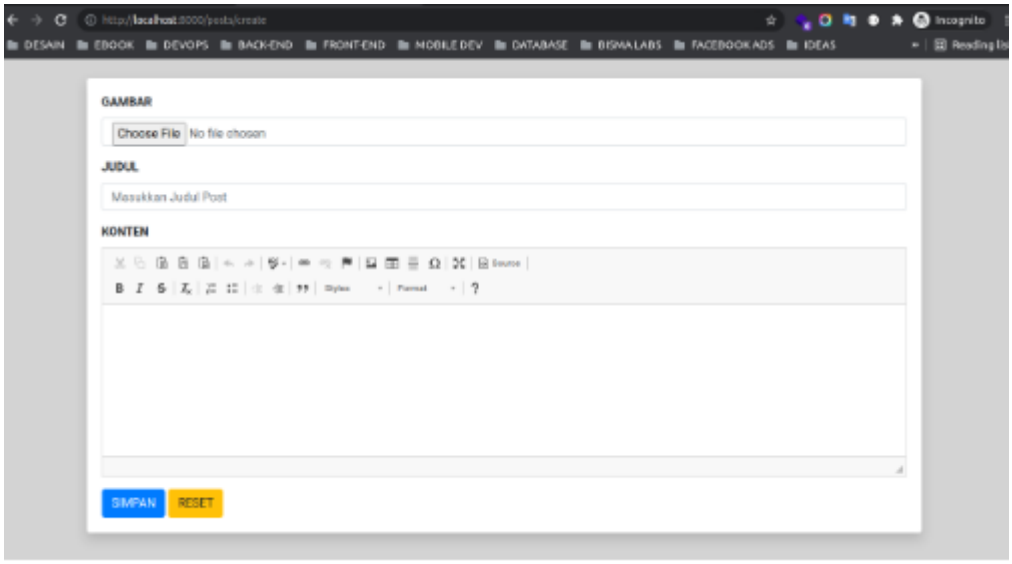
  Untuk membuat halaman view silahkan membuat sebuah file dengan nama **create.blade.php** dalam folder **resources/views/posts**.

```blade
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Tambah Data Post - SantriKoding.com</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body style="background: lightgray">

    <div class="container mt-5 mb-5">
        <div class="row">
            <div class="col-md-12">
                <div class="card border-0 shadow rounded">
                    <div class="card-body">
                        <form action="{{ route('posts.store') }}" method="POST" enctype="multipart/form-data">

                            @csrf

                            <div class="form-group">
                                <label class="font-weight-bold">GAMBAR</label>
                                <input type="file" class="form-control @error('image') is-invalid @enderror" nam
```

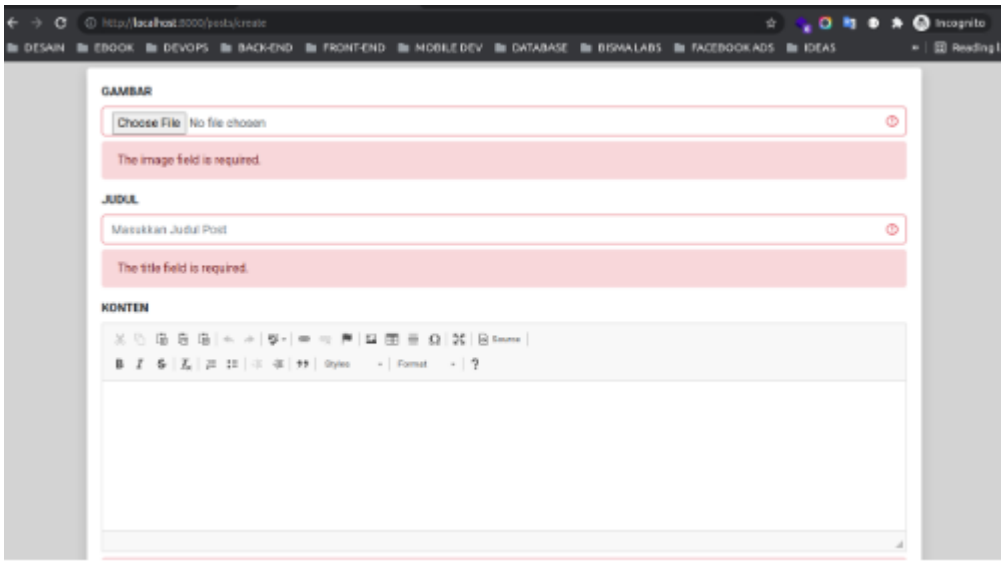
```blade
                            <!-- error message untuk title -->
                            @error('image')
                                <div class="alert alert-danger mt-2">
                                    {{ $message }}
                                </div>
                            @enderror
                        </div>

                        <div class="form-group">
                            <label class="font-weight-bold">JUDUL</label>
                            <input type="text" class="form-control @error('title') is-invalid @enderror" nam
                            <label class="font-weight-bold">KONTEN</label>
                            <textarea class="form-control @error('content') is-invalid @enderror" name="cont

                            <!-- error message untuk content -->
                            @error('content')
                                <div class="alert alert-danger mt-2">
                                    {{ $message }}
                                </div>
                            @enderror
                        </div>

                        <button type="submit" class="btn btn-md btn-primary">SIMPAN</button>
                        <button type="reset" class="btn btn-md btn-warning">RESET</button>

                    </form>
                </div>
            </div>
        </div>
    </div>

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
<script src="https://cdn.ckeditor.com/4.13.1/standard/ckeditor.js"></script>
<script>
    CKEDITOR.replace( 'content' );
</script>
```

- **Test Insert Data**

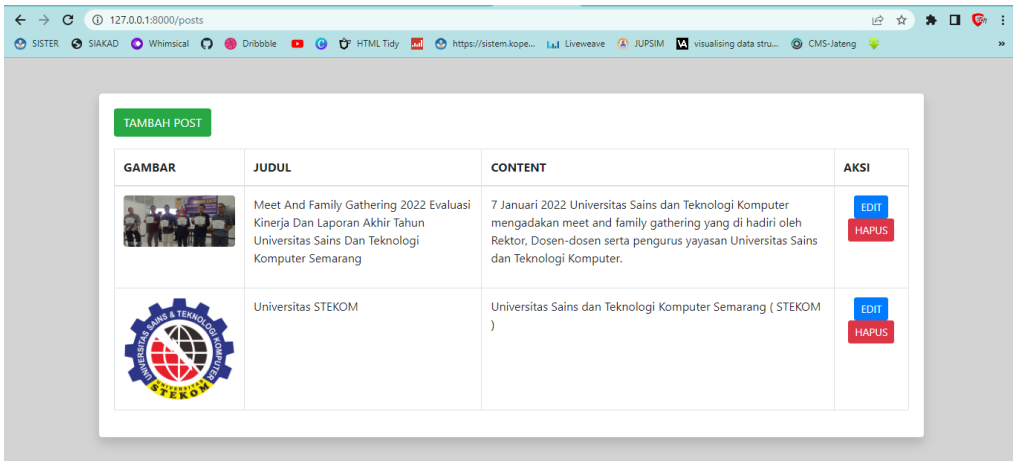
  Silahkan dicoba http://localhost/post/ceate



Dan kli tombol simpan



Hasilnya adalah



- **Edit dan Update Data ke Database**

  Untuk melakukan edit dan update kedatabase kita perlu membuat sebuah method edit dan update terlebih dahulu. Method edit akan kita gunakan untuk menapilkan halama edit dan untuk method update kita gunakan untuk proses data yang sudah kita edit.

Silahkan buka file dalam app/Http/Controllers/Controller.php da ubah kodenya seperti berikut :

```php
<?php

namespace App\Http\Controllers;

use App\Models\Post;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;

class PostController extends Controller
{
    /**
     * index
     *
     * @return void
     */
    public function index()
    {
        //get posts
        $posts = Post::latest()->paginate(5);

        //render view with posts
        return view('posts.index', compact('posts'));
    }

    /**
     * create
     *
     * @return void
     */
    public function create()
    {
        return view('posts.create');
    }

    /**
     * store
     *
     * @param Request $request
     * @return void
     */
    public function store(Request $request)
    {
        //validate form
        $this->validate($request, [
            'image'     => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048',
            'title'     => 'required|min:5',
            'content'   => 'required|min:10'
        ]);

        //upload image
        $image = $request->file('image');
        $image->storeAs('public/posts', $image->hashName());

        //create post
        Post::create([
            'image'     => $image->hashName(),
            'title'     => $request->title,
            'content'   => $request->content
        ]);

        //redirect to index
        return redirect()->route('posts.index')->with(['success' => 'Data Berhasil Disimpan!']);
    }
```

```php
    /**
     * edit
     *
     * @param  mixed $post
     * @return void
     */
    public function edit(Post $post)
    {
        return view('posts.edit', compact('post'));
    }

    /**
     * update
     *
     * @param  mixed $request
     * @param  mixed $post
     * @return void
     */
```

```php
        //validate form
        $this->validate($request, [
            'image'     => 'image|mimes:jpeg,png,jpg,gif,svg|max:2048',
            'title'     => 'required|min:5',
            'content'   => 'required|min:10'
        ]);

        //check if image is uploaded
        if ($request->hasFile('image')) {

            //upload new image
            $image = $request->file('image');
            $image->storeAs('public/posts', $image->hashName());

            //delete old image
            Storage::delete('public/posts/'.$post->image);

            //update post with new image
            $post->update([
                'image'     => $image->hashName(),
                'title'     => $request->title,
                'content'   => $request->content
            ]);

        } else {
```

```php
            //update post without image
            $post->update([
                'title'     => $request->title,
                'content'   => $request->content
            ]);
        }

        //redirect to index
        return redirect()->route('posts.index')->with(['success' => 'Data Berhasil Diubah!']);
    }
}
```

Selanjutnya kita menambah 2 method baru, yaitu edit dan update.

Berikutnya kita akan membuat Form Edit

Silahkan buat file baru dengan nama **edit.balde.php** dalam folder resource/views/posts dan masukan kode sebagai berikut :

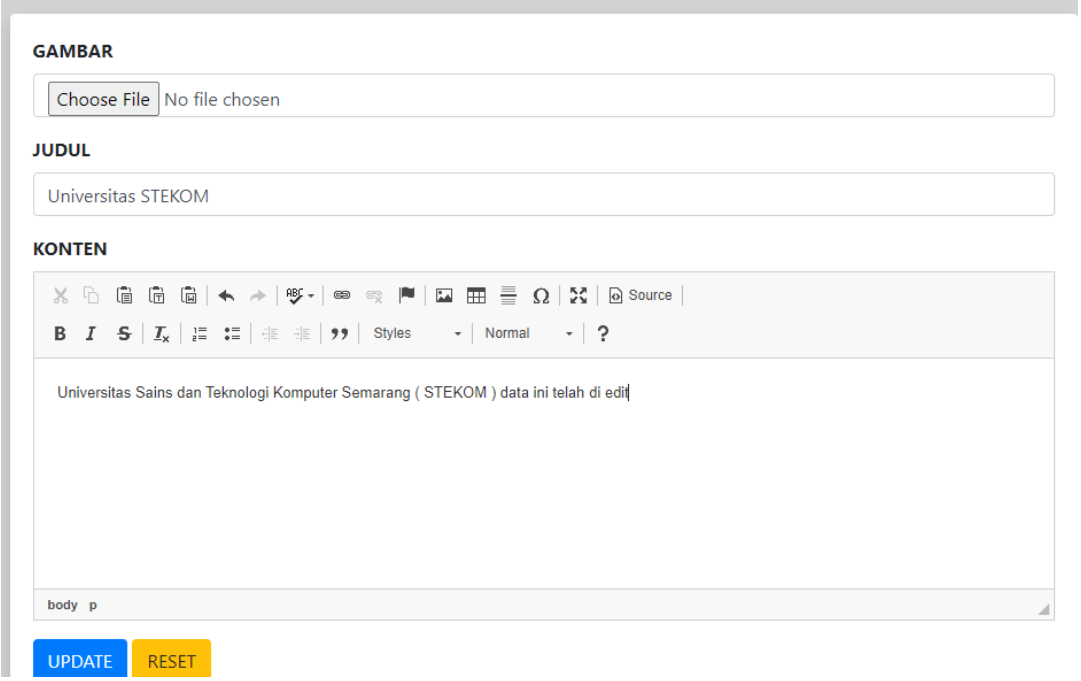
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Edit Data Post - SantriKoding.com</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body style="background: lightgray">

    <div class="container mt-5 mb-5">
        <div class="row">
            <div class="col-md-12">
                <div class="card border-0 shadow rounded">
                    <div class="card-body">
                        <form action="{{ route('posts.update', $post->id) }}" method="POST" enctype="multipart/
                            @csrf
                            @method('PUT')

                            <div class="form-group">
                                <label class="font-weight-bold">GAMBAR</label>
                                <input type="file" class="form-control" name="image">
                            </div>
```

```
                            <div class="form-group">
                                <label class="font-weight-bold">JUDUL</label>
                                <input type="text" class="form-control @error('title') is-invalid @enderror" na

                                <!-- error message untuk title -->
                                @error('title')
                                    <div class="alert alert-danger mt-2">
                                        {{ $message }}
                                    </div>
                                @enderror
                            </div>

                            <div class="form-group">
                                <label class="font-weight-bold">KONTEN</label>
                                <textarea class="form-control @error('content') is-invalid @enderror" name="con

                                <!-- error message untuk content -->
                                @error('content')
                                    <div class="alert alert-danger mt-2">
                                        {{ $message }}
                                    </div>
                                @enderror
                            </div>
```

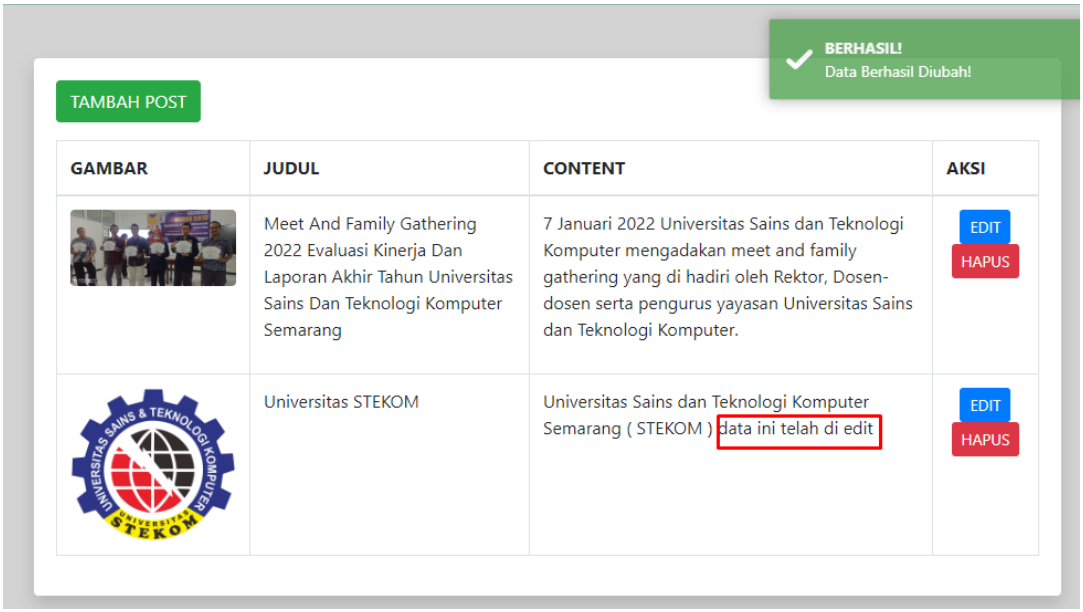
```
                            <button type="submit" class="btn btn-md btn-primary">UPDATE</button>
                            <button type="reset" class="btn btn-md btn-warning">RESET</button>

                        </form>
                    </div>
                </div>
            </div>
        </div>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
<script src="https://cdn.ckeditor.com/4.13.1/standard/ckeditor.js"></script>
<script>
    CKEDITOR.replace( 'content' );
</script>
</body>
</html>
```

Jika sudah silahkan jalankan formnya dan klik tombol EDIT, isikan datanya dan klik tombol Update

Data sebelum di edit, silahkan klik tombol Edit pada form dan ubah isi konten



Jika sudah dirubah, silahkan klik tombol UPDATE untuk menyimpan perubahan



Maka sudah ada perubahan data pada form diatas.

- **Hapus Data dari Database**

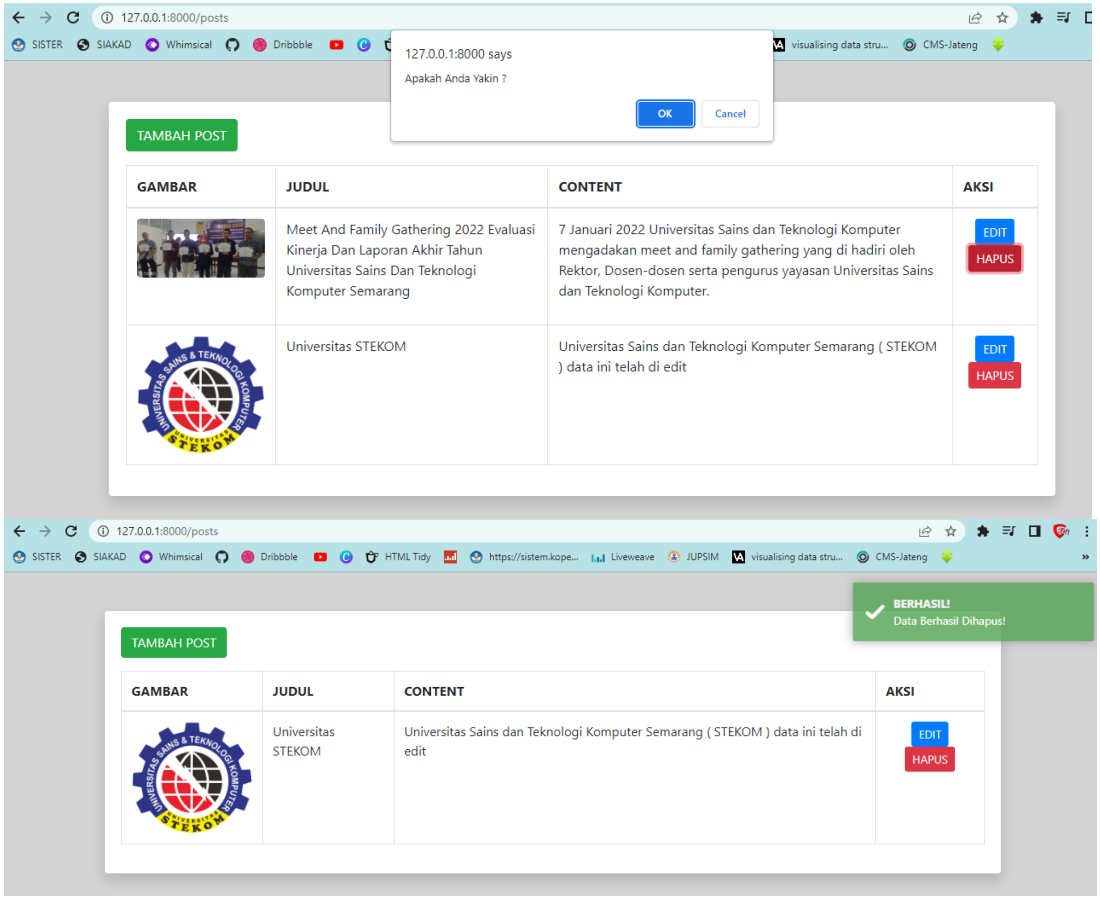
Untuk menghapus data Langkah awal yang harus kita siapakan adalah membuat Method **Destroy** dalam controller post yang berada dalam **app/Http/Controllers/PostController.php**

Untuk kodenya adalah sebagai berikut :

```php
<?php

namespace App\Http\Controllers;

use App\Models\Post;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Storage;

class PostController extends Controller
{
    /**
     * index
     *
     * @return void
     */
    public function index()
    {
        //get posts
        $posts = Post::latest()->paginate(5);

        //render view with posts
        return view('posts.index', compact('posts'));
    }
```

```
    public function update(Request $request, Post $post)
    {
        //validate form
        $this->validate($request, [
            'image'     => 'image|mimes:jpeg,png,jpg,gif,svg|max:2048',
            'title'     => 'required|min:5',
            'content'   => 'required|min:10'
        ]);

        //check if image is uploaded
        if ($request->hasFile('image')) {

            //upload new image
            $image = $request->file('image');
            $image->storeAs('public/posts', $image->hashName());

            //delete old image
            Storage::delete('public/posts/'.$post->image);

            //update post with new image
            $post->update([
                'image'     => $image->hashName(),
                'title'     => $request->title,
                'content'   => $request->content
            ]);

        //delete post
        $post->delete();

        //redirect to index
        return redirect()->route('posts.index')->with(['success' => 'Data Berhasil Dihapus!']);
    }
}
```

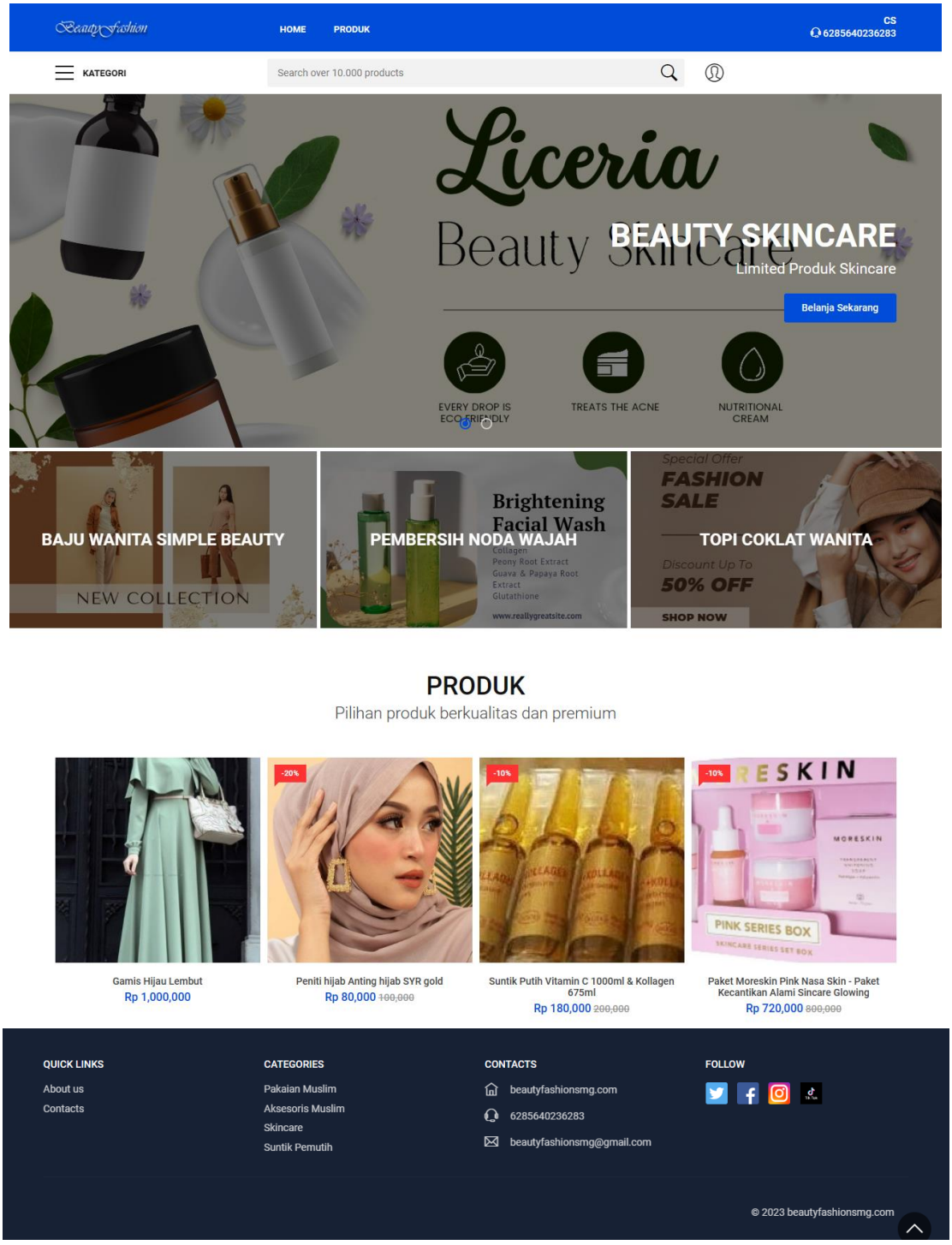
Penambahan methode destroy digunakan untuk melakukan penghapusan gambar dari post terkait, selanjutnya menghapus post dari database dan yang terakhir kita arahkan ke post.index untuk notifikasi



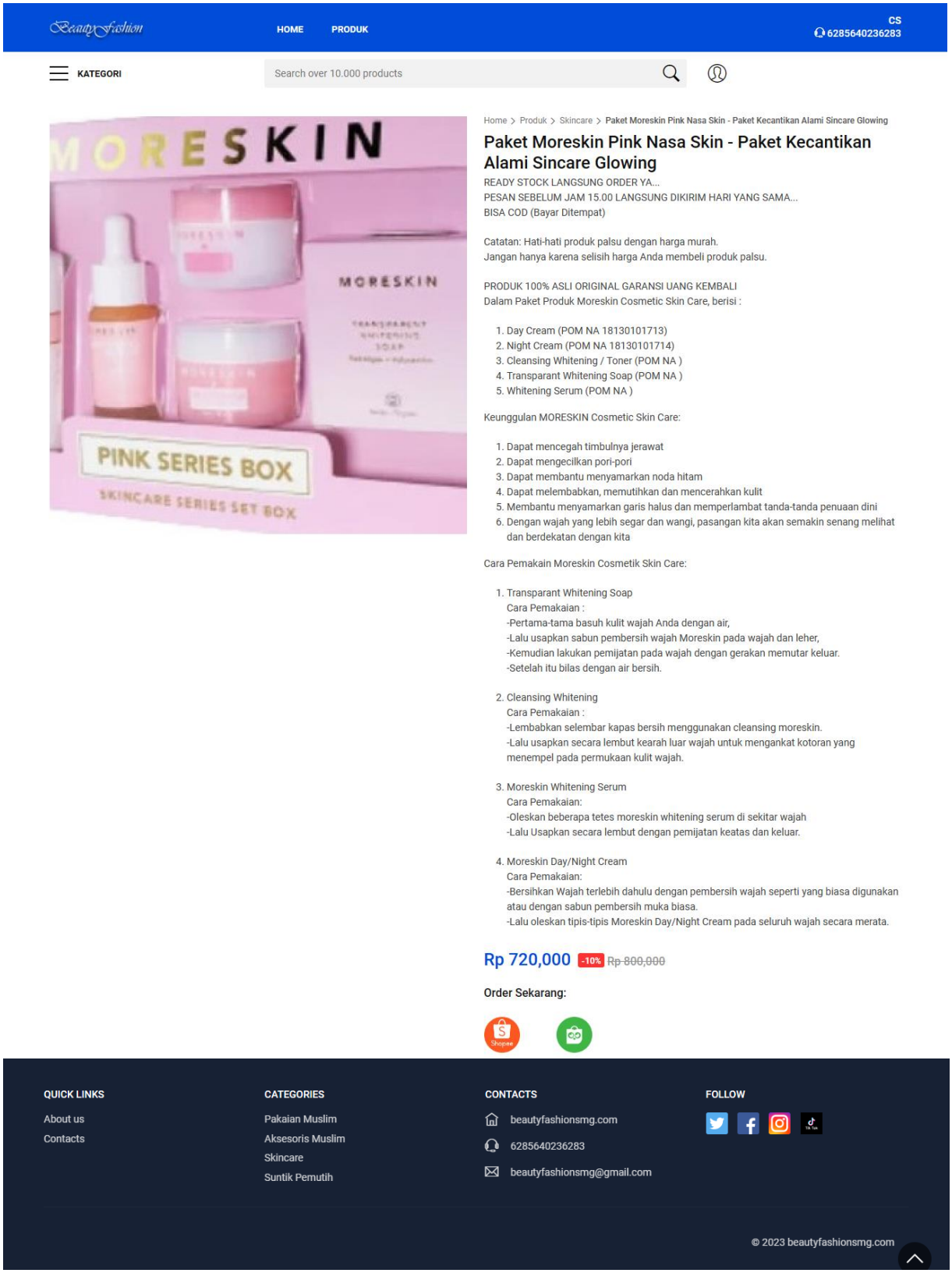
BAB 4

CONTOH KASUS PROJECT MENGGUNAKAN LARAVEL

Pada bab ini kita akan memebahas contoh kasus penggunaan Laravel dalam membangun sebuah website toko online.

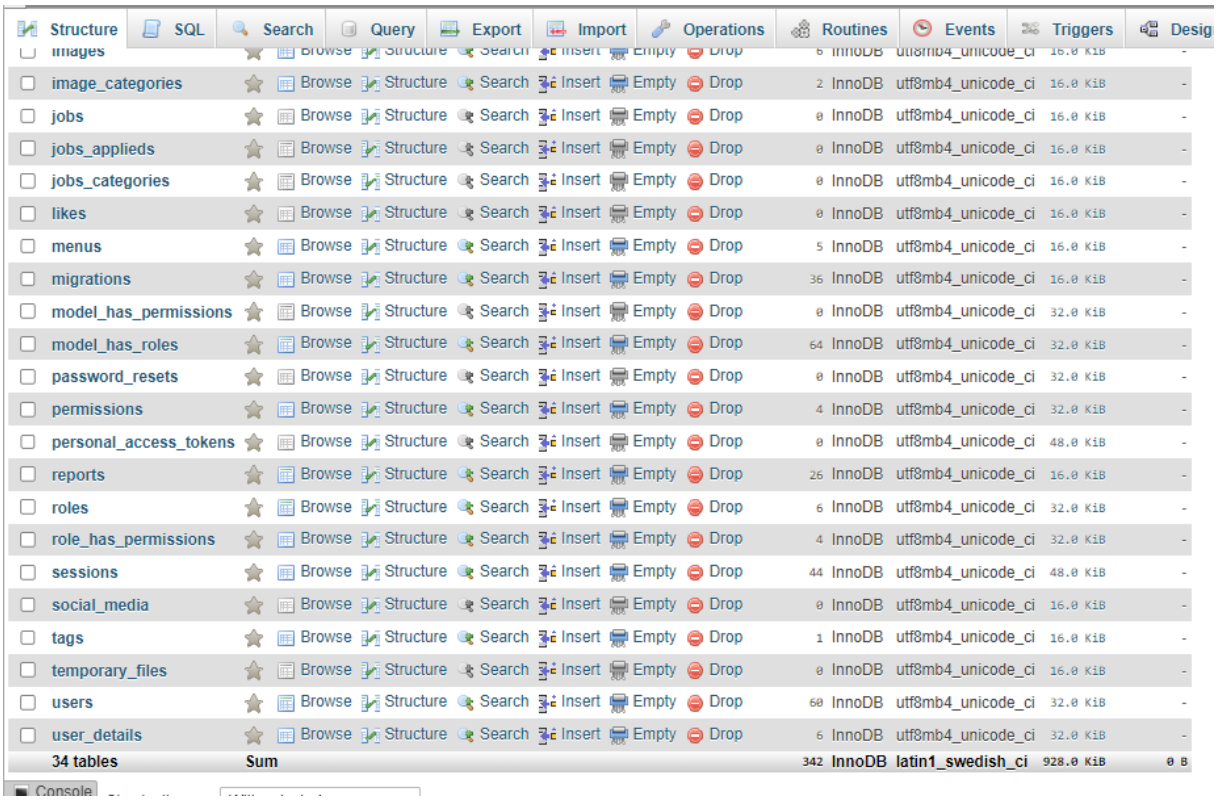


Gambra : Halaman Utama Toko Online



Gambar : Halaman Detail Produk

Langkah awal buatlah sebuah database pada PhpMyadmin dengan nama **beauty_fashion**Selanjutnya buatlah tabel dengan struktur sebagai berikut :



Structure	SQL	Search	Query	Export	Import	Operations	Routines	Events	Triggers	Desig

image_categories	2	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
jobs	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
jobs_applieds	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
jobs_categories	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
likes	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
menus	5	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
migrations	36	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
model_has_permissions	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
model_has_roles	64	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
password_resets	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
permissions	4	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
personal_access_tokens	0	InnoDB	utf8mb4_unicode_ci	48.0 KiB	-
reports	26	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
roles	6	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
role_has_permissions	4	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
sessions	44	InnoDB	utf8mb4_unicode_ci	48.0 KiB	-
social_media	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
tags	1	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
temporary_files	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
users	60	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
user_details	6	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
34 tables	Sum	342	InnoDB	latin1_swedish_ci	928.0 KiB	0 B

Selanjutnya silahkan Menuju teks editor untuk meniliskan kode kode yang dibutuhkan masuk directory beauty-fashion-shop/app/Http/**Controllers**/ untuk mengatur controller.

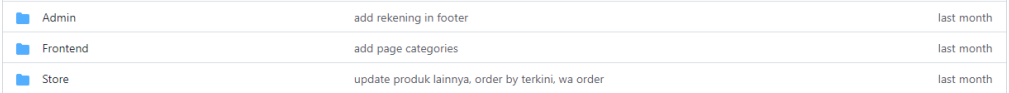
Kode controller.php

13 lines (10 sloc) | 361 Bytes

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Foundation\Auth\Access\AuthorizesRequests;
use Illuminate\Foundation\Bus\DispatchesJobs;
use Illuminate\Foundation\Validation\ValidatesRequests;
use Illuminate\Routing\Controller as BaseController;

class Controller extends BaseController
{
    use AuthorizesRequests, DispatchesJobs, ValidatesRequests;
}
```

Give feedback

Dalam directory controller terdapat 3 Folder penting,yaitu **Admin, Frontend dan Store** yang berfungsi untuk mengatur jalanya aplikasi mulai dari fronend hingga backend.



Admin	add rekening in footer	last month
Frontend	add page categories	last month
Store	update produk lainnya, order by terkini, wa order	last month

Berikut adalah Struktur file pada beauty-fashion-shop/app/Http/Controllers/**Admin**/

Jobs	initial commit	last month
Post	initial commit	last month
Settings	add rekening in footer	last month
Store	update home	last month
User	initial commit	last month
ChatsController.php	initial commit	last month
DashboardController.php	initial commit	last month
ImageCategoryController.php	initial commit	last month
ImageController.php	initial commit	last month
MenuController.php	initial commit	last month
PartnerController.php	initial commit	last month
QrCodeController.php	initial commit	last month
UserController.php	initial commit	last month
VideoController.php	initial commit	last month

Pada pembahasan ini kita akan ambil sample pada directory Store

..		
BookController.php	update home	last month
DataOrderController.php	initial commit	last month
ReportController.php	initial commit	last month

BookController.php

```php
<?php

namespace App\Http\Controllers\Admin\Store;

use App\Http\Controllers\Controller;
use App\ImageProses;
use App\Models\Author;
use App\Models\Editor;
use App\Models\Post\PostCategories;
use App\Models\Store\Book;
use App\Models\Tag;
use App\Models\User;
use Exception;
use Illuminate\Support\Str;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Cache;
use Illuminate\Support\Facades\Validator;
use RealRashid\SweetAlert\Facades\Alert;
use RobertSeghedi\News\Models\Category;

class BookController extends Controller
{

    public function index()
    {
        return view('admin.store.book.index');
    }
```

```php
class BookController extends Controller
{

    public function index()
    {
        return view('admin.store.book.index');
    }

    public function create()
    {
        // dd(User::role('author')->get());
        // $user = User::find(5);
        // dd($user->biodata);
        return view('admin.store.book.create', [
            'categories' => PostCategories::all(),
            'tags'       => Tag::where('status', true)->get(),
            'authors'    => User::all(),
            'editors'    => User::all(),
        ]);
    }
```

```
43      public function store(Request $request)
44      {
45          $dataImageSetting = [
46              'ori_width'=>config('app.img_size.ori_width'),
47              'ori_height'=>config('app.img_size.ori_height'),
48              'mid_width'=>config('app.img_size.mid_width'),
49              'mid_height'=>config('app.img_size.mid_height'),
50              'thumb_width'=>config('app.img_size.thumb_width'),
51              'thumb_height'=>config('app.img_size.thumb_height')
52          ];
53
54          $validator = Validator::make($request->all(), [
55              'title' => 'required',
56              'category' => 'required',
57              'price' => 'required',
58              'stock' => 'required',
59              'description' => 'required',
60              'image'=>'required|image|mimes:jpeg,png,jpg',
61          ]);
62
63          if($validator->fails()) {
64              Alert::toast($validator->errors()->first(), 'error');
65              return redirect()->back();
66          }
70          $dataImg = array(
71              'skala169' => array(
72                  'width'=>$request->input('16_9_width'),
73                  'height'=>$request->input('16_9_height'),
74                  'x'=>$request->input('16_9_x'),
75                  'y'=>$request->input('16_9_y')
76              ),
77              'skala43' => array(
78                  'width'=>$request->input('4_3_width'),
79                  'height'=>$request->input('4_3_height'),
80                  'x'=>$request->input('4_3_x'),
81                  'y'=>$request->input('4_3_y')
82              ),
83              'skala11' => array(
84                  'width'=>$request->input('1_1_width'),
85                  'height'=>$request->input('1_1_height'),
86                  'x'=>$request->input('1_1_x'),
87                  'y'=>$request->input('1_1_y')
88              )
89          );
90
91          $dataImage = [
92              'file'=>$request->file('image'),
93              'setting'=>$dataImageSetting,
94              'path'=>public_path('storage/pictures/product/'),
95              'modul'=>'product',
96              'dataImg'=>$dataImg
97          ];
98
99          $uploadImg = ImageProses::imageCropDimensi($dataImage);
100         if($uploadImg['status'] == true){
```

```
            //          'book_id' => ($data['book_author_id'])
            //     ]);
            // }

            // foreach ($request->editors as $k => $editor) {
            //     $data['editor']           = $editor;
            //     $data['book_author_id']   = $book->id;

            //     Editor::where('book_id', $id)->update([
            //         'user_id' => ($data['editor']),
            //         'book_id' => ($data['book_author_id'])
            //     ]);
            // }

            Cache::flush();

            if($book){
                Alert::success('Updated', 'Produk berhasil diupdate.');
                return redirect()->route('products.index');
            }
        } catch (Exception $error) {
            Alert::error('Terjadi masalah', $error->getMessage());
            return back();
        }
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
```

```
<?php

namespace App\Http\Controllers\Admin\Store;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class ReportController extends Controller
{

    public function index()
    {
        return view('admin.store.report.index');
    }

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

Pada script diatas kita telah membuat sebuah controller untuk penjualan produk, Se;anjutnya kita akan membuat scrip **DataOrderController.php**

```php
<?php

namespace App\Http\Controllers\Admin\Store;

use App\Http\Controllers\Controller;
use App\Models\Admin\Configuration;
use App\Models\Store\Cart;
use Illuminate\Http\Request;
use PDF;

class DataOrderController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        return view('admin.store.data-order.index');
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        //
    }

    public function downloadInvoice($id){
        $data = Cart::findOrFail($id);
        $pdf = PDF::loadView('admin.store.data-order.invoice', [
            'data' => $data,
            'site' => Configuration::latest()->first(),
        ])->setOptions(['defaultFont' => 'sans-serif']);

        // dd($pdf);
        return $pdf->download('invoice'.time().'.pdf');
    }
}
```

ReportController.php digunakan untuk membuat kontroller dari laporan penjualan

```php
<?php

namespace App\Http\Controllers\Admin\Store;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class ReportController extends Controller
{

    public function index()
    {
        return view('admin.store.report.index');
    }

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

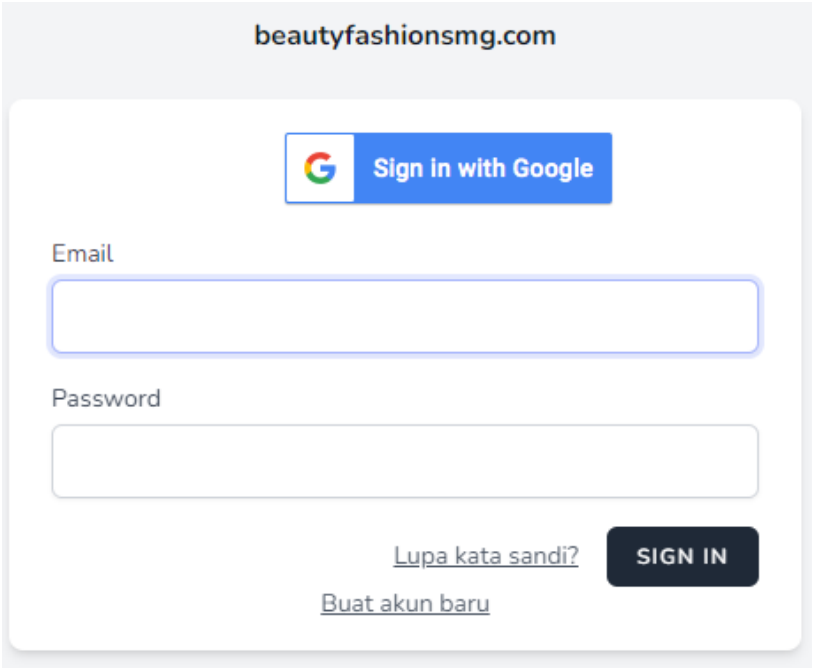
Selanjutnya silahkan buat direktori beauty-fashion-shop/**public**/ yang berisikan data file

assets	update header	last month
backend	update form cpanel	last month
css	initial commit	last month
frontend	update form cpanel	last month
images	initial commit	last month
js	initial commit	last month
vendor	initial commit	last month
.htaccess	update form cpanel	last month
favicon.ico	initial commit	last month
index.php	update header	last month
mix-manifest.json	initial commit	last month
robots.txt	initial commit	last month
sitemap.xml	initial commit	last month
storage	update form cpanel	last month

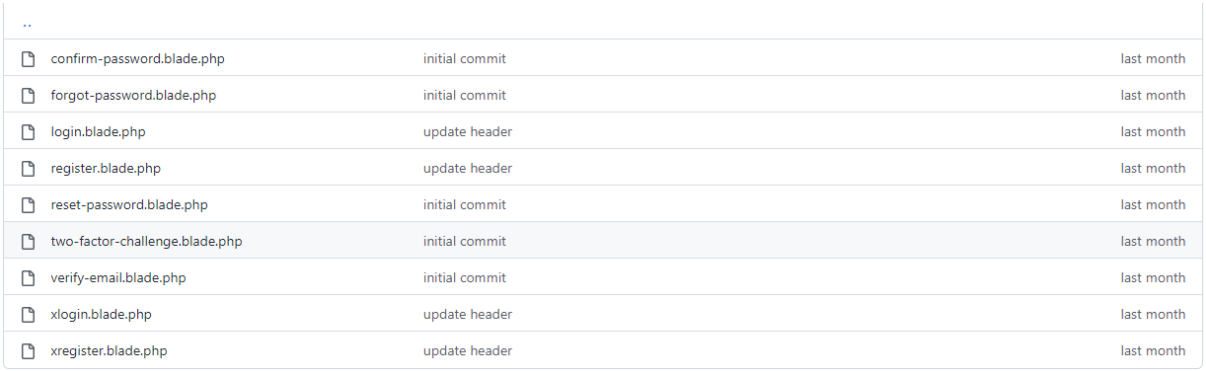
Sementara untuk viewnya terletak pada direktori beauty-fashion-shop/resources/**views**/

..		
admin	add rekening in footer	last month
api	initial commit	last month
auth	update header	last month
errors	update login	last month
frontend	detail produk slider	last month
layouts	update home	last month
livewire	update home	last month
profile	initial commit	last month
vendor	update header	last month
dashboard.blade.php	initial commit	last month
home.blade.php	initial commit	last month
navigation-menu.blade.php	initial commit	last month
policy.blade.php	initial commit	last month
terms.blade.php	initial commit	last month
welcome.blade.php	initial commit	last month

Selanjutnya kita akan bahas kode mulai dari halaman login



Untuk kode halaman login terletak pada sebuah direktori beauty-fashion-shop/resources/views/**auth**/ yang terdiri dari struktur file sebagai berikut ini :



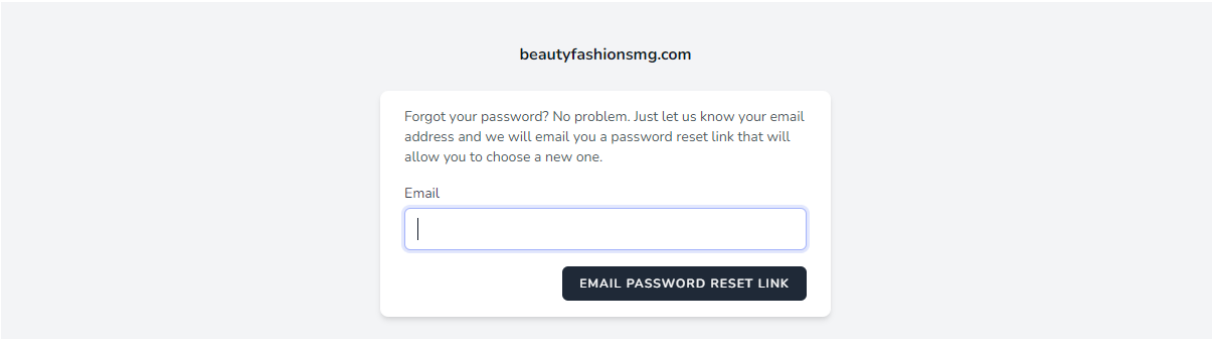
login.blade.php

```
<x-guest-layout>
    @include('sweetalert::alert')
    <x-jet-authentication-card>
        <x-slot name="logo">
            <x-jet-authentication-card-logo />
        </x-slot>

        <x-jet-validation-errors class="mb-4" />

        @if (session('status'))
            <div class="mb-4 font-medium text-sm text-green-600">
                {{ session('status') }}
            </div>
        @endif
        <div class="flex items-center justify-center mb-4">
            <a href="{{ url('auth/google') }}">
                <img src="https://developers.google.com/identity/images/btn_google_signin_dark_normal_web.png" style="margin-left: 3em;">
            </a>
        </div>
        <form method="POST" action="{{ route('login') }}">
            @csrf

            <div>
                <x-jet-label for="email" value="{{ __('Email') }}" />
                <x-jet-input id="email" class="block mt-1 w-full" type="email" name="email" :value="old('email')" required autofocus />
            </div>

            <div class="mt-4">
                <x-jet-label for="password" value="{{ __('Password') }}" />
                <x-jet-input id="password" class="block mt-1 w-full" type="password" name="password" required autocomplete="current-password" />
            </div>

            <div class="flex items-center justify-end mt-4">
                @if (Route::has('password.request'))
                    <a class="underline text-sm text-gray-600 hover:text-gray-900" href="{{ route('password.request') }}">
                        {{ __('Lupa kata sandi?') }}
                    </a>
                @endif

                <x-jet-button class="ml-4">
                    {{ __('Sign in') }}
                </x-jet-button>
            </div>
            <div class="flex items-center justify-center">
                <a class="underline text-sm text-gray-600 hover:text-gray-900" href="{{ route('register') }}">
                    {{ __('Buat akun baru') }}
                </a>
            </div>

        </form>
    </x-jet-authentication-card>
</x-guest-layout>
```

Dengan kode tersebut kita telah membuat perintah pada form login, selanjutnya jika user memasuka nama user dan pasword maka akan di konfirmasi apakah data user sudah sesaui dengan yang ada pada database, unutk membuat kode tersebut silahkan siapkan file **confirm-password.blade.php**

```
<x-guest-layout>
    <x-jet-authentication-card>
        <x-slot name="logo">
            <x-jet-authentication-card-logo />
        </x-slot>

        <div class="mb-4 text-sm text-gray-600">
            {{ __('This is a secure area of the application. Please confirm your password before continuing.') }}
        </div>

        <x-jet-validation-errors class="mb-4" />

        <form method="POST" action="{{ route('password.confirm') }}">
            @csrf

            <div>
                <x-jet-label for="password" value="{{ __('Password') }}" />
                <x-jet-input id="password" class="block mt-1 w-full" type="password" name="password" required autocomplete="current-password" autofocus />
            </div>

            <div class="flex justify-end mt-4">
                <x-jet-button class="ml-4">
                    {{ __('Confirm') }}
                </x-jet-button>
            </div>
        </form>
    </x-jet-authentication-card>
</x-guest-layout>
```

forgot-password.blade.php digunakan ketika user lupa password

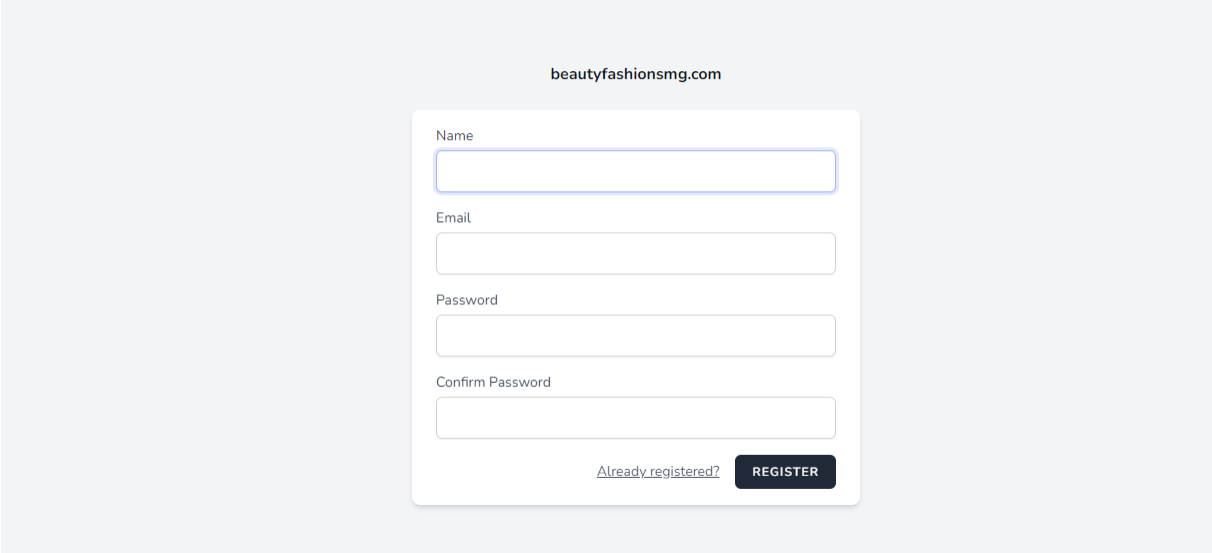



```blade
<x-guest-layout>
    <x-jet-authentication-card>
        <x-slot name="logo">
            <x-jet-authentication-card-logo />
        </x-slot>

        <div class="mb-4 text-sm text-gray-600">
            {{ __('Forgot your password? No problem. Just let us know your email address and we will email you a password reset link that will allow you to choose a new one.') }}
        </div>

        @if (session('status'))
            <div class="mb-4 font-medium text-sm text-green-600">
                {{ session('status') }}
            </div>
        @endif

        <x-jet-validation-errors class="mb-4" />

        <form method="POST" action="{{ route('password.email') }}">
            @csrf

            <div class="block">
                <x-jet-label for="email" value="{{ __('Email') }}" />
                <x-jet-input id="email" class="block mt-1 w-full" type="email" name="email" :value="old('email')" required autofocus />
            </div>

            <div class="flex items-center justify-end mt-4">
                <x-jet-button>
                    {{ __('Email Password Reset Link') }}
                </x-jet-button>
            </div>
        </form>
    </x-jet-authentication-card>
</x-guest-layout>
```

Ketika user hendak mereset password maka kita tambah kan kode **reset-password.blade.php**

```blade
<x-guest-layout>
    <x-jet-authentication-card>
        <x-slot name="logo">
            <x-jet-authentication-card-logo />
        </x-slot>

        <x-jet-validation-errors class="mb-4" />

        <form method="POST" action="{{ route('password.update') }}">
            @csrf

            <input type="hidden" name="token" value="{{ $request->route('token') }}">

            <div class="block">
                <x-jet-label for="email" value="{{ __('Email') }}" />
                <x-jet-input id="email" class="block mt-1 w-full" type="email" name="email" :value="old('email', $request->email)" required autofocus />
            </div>

            <div class="mt-4">
                <x-jet-label for="password" value="{{ __('Password') }}" />
                <x-jet-input id="password" class="block mt-1 w-full" type="password" name="password" required autocomplete="new-password" />
            </div>

            <div class="mt-4">
                <x-jet-label for="password_confirmation" value="{{ __('Confirm Password') }}" />
                <x-jet-input id="password_confirmation" class="block mt-1 w-full" type="password" name="password_confirmation" required autocomplete="new-password" />
            </div>

            <div class="flex items-center justify-end mt-4">
                <x-jet-button>
                    {{ __('Reset Password') }}
                </x-jet-button>
            </div>
        </form>
    </x-jet-authentication-card>
</x-guest-layout>
```

Selanjutnya kita akan membahas form regitrasi user, digunakan ketika user hendak menjadi member atau administrator pada toko yang sudah kita buat



Gambar diata merupakan tampilan form registrasi / pendaftran user baru, untuk kodenya silahkan siapkan sebuah file dengan nama **beauty-fashion-shop/resources/views/auth/register.blade.php**

```
<x-guest-layout>
    <x-jet-authentication-card>
        <x-slot name="logo">
            <x-jet-authentication-card-logo />
        </x-slot>

        <x-jet-validation-errors class="mb-4" />

        <form method="POST" action="{{ route('register') }}">
            @csrf

            <div>
                <x-jet-label for="name" value="{{ __('Name') }}" />
                <x-jet-input id="name" class="block mt-1 w-full" type="text" name="name" :value="old('name')" required autofocus autocomplete="name" />
            </div>

            <div class="mt-4">
                <x-jet-label for="email" value="{{ __('Email') }}" />
                <x-jet-input id="email" class="block mt-1 w-full" type="email" name="email" :value="old('email')" required />
            </div>

            <div class="mt-4">
                <x-jet-label for="password" value="{{ __('Password') }}" />
                <x-jet-input id="password" class="block mt-1 w-full" type="password" name="password" required autocomplete="new-password" />
            </div>

            <div class="mt-4">
                <x-jet-label for="password_confirmation" value="{{ __('Confirm Password') }}" />
                <x-jet-input id="password_confirmation" class="block mt-1 w-full" type="password" name="password_confirmation" required autocomplete="new-password" />
            </div>

            @if (Laravel\Jetstream\Jetstream::hasTermsAndPrivacyPolicyFeature())
                <div class="mt-4">
                    <x-jet-label for="terms">
                        <div class="flex items-center">
                            <x-jet-checkbox name="terms" id="terms"/>

                            <div class="ml-2">
                                {!! __('I agree to the :terms_of_service and :privacy_policy', [
                                        'terms_of_service' => '<a target="_blank" href="'.route('terms.show').'" class="underline text-sm text-gray-600 hover:text-gray-900">'.
                                        'privacy_policy' => '<a target="_blank" href="'.route('policy.show').'" class="underline text-sm text-gray-600 hover:text-gray-900">'._
                                    ]) !!}
                            </div>
                        </div>
                    </x-jet-label>
                </div>
            @endif

            <div class="flex items-center justify-end mt-4">
                <a class="underline text-sm text-gray-600 hover:text-gray-900" href="{{ route('login') }}">
                    {{ __('Already registered?') }}
                </a>

                <x-jet-button class="ml-4">
                    {{ __('Register') }}
                </x-jet-button>
            </div>
        </form>
    </x-jet-authentication-card>
</x-guest-layout>
```

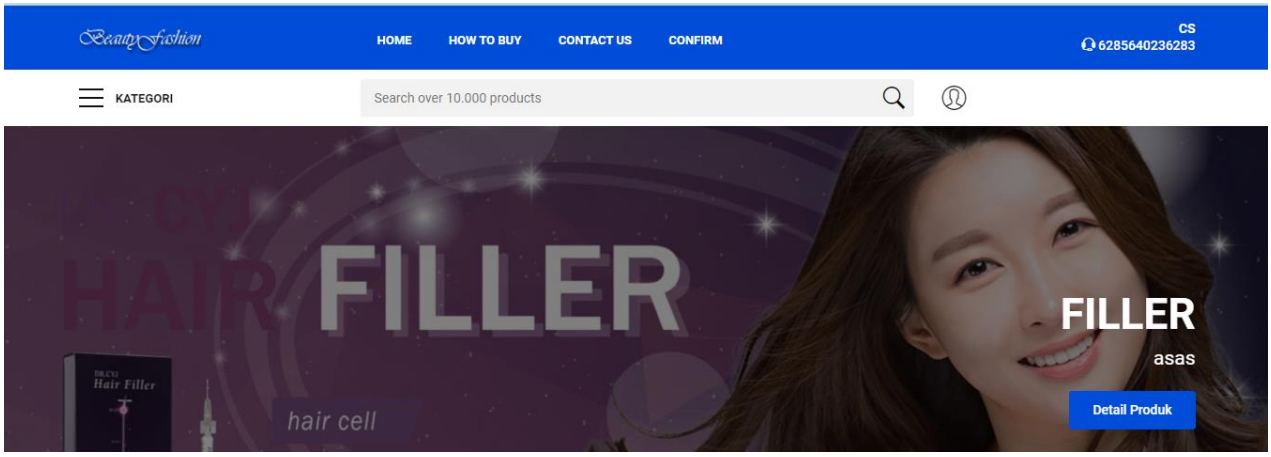
Setelah login dan register kita akan membahas pada halaman frontend, halam ini adalah halaman antarmuka yang terdapat pada pengunjung website kita, silahkan siapkan direktori sebagai berikut beauty-fashion-shop/resources/views/**frontend**/ dan dalam direktori tersebut berikan beberapa file dan folder :

..		
articles	initial commit	last month
books	tombol MP muncul jika link terisi	last month
jobs	initial commit	last month
layouts	update logo mobile	last month
page	initial commit	last month
sections	update produk lainnya, order by terkini, wa order	last month
about.blade.php	initial commit	last month
coming-soon.blade.php	initial commit	last month
home.blade.php	detail produk slider	last month
search.blade.php	initial commit	last month

Sebagai contoh kita akan membahas isi dala folder layout beauty-fashionshop/resources/views/ frontend/**layouts**/ yang berisi kan beberapa file sebagai berikut ini :

..		
banner-top.blade.php	initial commit	last month
footer.blade.php	add rekening in footer	last month
header.blade.php	update logo mobile	last month
master.blade.php	update logo mobile	last month
middle.blade.php	initial commit	last month
navbar.blade.php	initial commit	last month

yang pertama silahkan siapkan file dengan nama **beauty-fashion-shop/resources/views/frontend/ layouts/banner-top.blade.php** yang berfungsi untuk menampung data banner atas yang berfungsi sebagi tempat iklan atau kategori produk.



untuk kode **banner-top.blade.php** adalah sebagai berikut ini

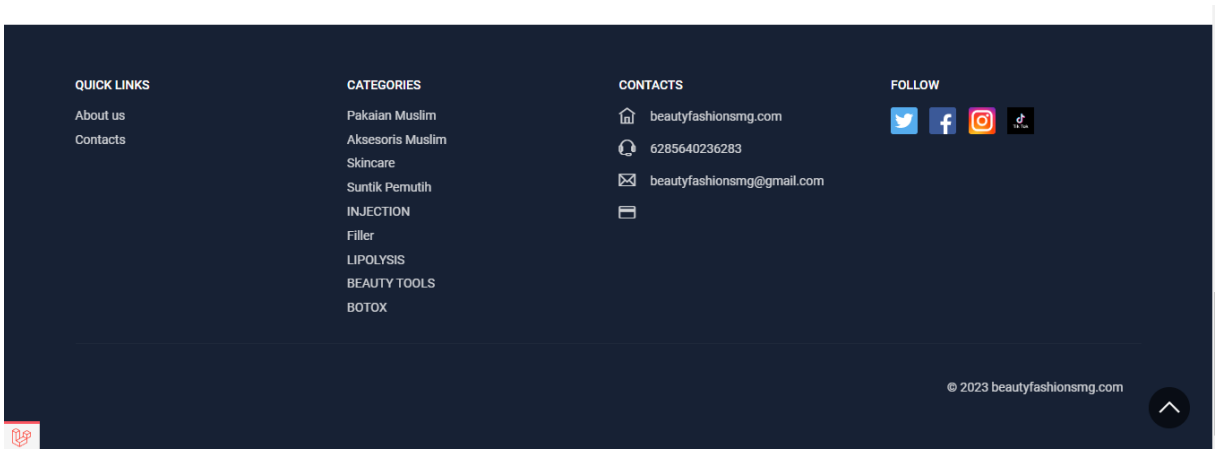
```
<!-- Middle Header -->
<div class="container-fluid no-left-padding no-right-padding middle-header">
    <!-- Container -->
    <div class="container">
        <!-- Row -->
        <div class="row">
            <div class="col-md-4 logo-block">
                <a href="index.html"><img src="{{ asset('frontend') }}/assets/images/logo.png" alt="Logo" /></a>
            </div>
            <div class="col-md-8 add-block-banner">
                <a href="#"><img src="{{ asset('frontend') }}/assets/images/top-banner.jpg" alt="Banner" /></a>
            </div>
        </div><!-- Row /- -->
    </div><!-- Container /- -->
</div>
<!-- Middle Header /- -->
```

header.blade.php

```
@php
    $config = \App\Models\Admin\Configuration::latest()->first();
@endphp
<header class="version_1">
    <div class="layer"></div>
    <div class="main_header">
        <div class="container">
            <div class="row small-gutters">
                <div class="col-xl-3 col-lg-3 d-lg-flex align-items-center">
                    <div id="logo">
                        <a href="/"><img src="{{ asset('frontend/assets/images/logo.png') }}" alt="" width="130" height="35"></a>
                    </div>
                </div>
                <nav class="col-xl-6 col-lg-7">
                    <a class="open_close" href="javascript:void(0);">
                        <div class="hamburger hamburger--spin">
                            <div class="hamburger-box">
                                <div class="hamburger-inner"></div>
                            </div>
                        </div>
                    </a>
                    <div class="main-menu">
                        <div id="header_menu">
                            <a href="index.html"><img src="{{ asset('frontend/assets/images/logo.png') }}" alt="" width="130" height="35"></a>
                            <a href="#" class="open_close" id="close_in"><i class="ti-close"></i></a>
                        </div>
                        <ul>
                            @php
                                $menu = \App\Models\Menu::where('status', 1)->orderBy('order', 'ASC')->get();
                            @endphp
                            @foreach ($menu as $item)
                            @if ($item->parent_id == 0 && $item->category_id == 1)
                                <li class="{{ $item->type == 1 ? 'submenu' : '#' }}">
                                    <a href="{{ $item->slug }}" class="show-submenu">{{ $item->name }}</a>
                                    @if ($item->type == 1)
                                        <ul>
                                            @foreach ($menu as $value)
                                                @if($value->parent_id == $item->id)
                                                    <li><a href="{{ $value->slug }}">{{ $value->name }}</a></li>
                                                @endif
                                            @endforeach
                                        </ul>
                                    @endif
                                </li>
                            @endif
                            @endforeach
                        </ul>
                    </div>
                </nav>
                <div class="col-xl-3 col-lg-2 d-lg-flex align-items-center justify-content-end text-end">
                    <a class="phone_top" href="https://wa.me/{{ $config->whatsapp }}"><strong><span>CS</span><i class="ti-headphone-alt"></i> {{ $config->whatsapp }}</strong
                </div>
            </div>
            <!-- /row -->
        </div>
    </div>
    <!-- /main_header -->
```

```
    <div class="main_nav Sticky">
        <div class="container">
            <div class="row small-gutters">
                <div class="col-xl-3 col-lg-3 col-md-3">
                    <nav class="categories">
                        <ul class="clearfix">
                            <li><span>
                                    <a href="#">
                                        <span class="hamburger hamburger--spin">
                                            <span class="hamburger-box">
                                                <span class="hamburger-inner"></span>
                                            </span>
                                        </span>
                                        Kategori
                                    </a>
                                </span>
                                <div id="menu">
                                    <ul>
                                        @php
                                            $categories = \App\Models\Post\PostCategories::all();
                                        @endphp
                                        @foreach ($categories as $item)
                                            <li><span><a href="/categories/{{ $item->slug }}">{{ $item->name }}</a></span></li>
                                        @endforeach
                                    </ul>
                                </div>
                            </li>
                        </ul>
                    </nav>
                </div>
                <div class="col-xl-6 col-lg-7 col-md-6 d-none d-md-block">
                    <div class="custom-search-input">
                        <input type="text" placeholder="Search over 10.000 products">
                        <button type="submit"><i class="header-icon_search_custom"></i></button>
                    </div>
                </div>
                <div class="col-xl-3 col-lg-2 col-md-3 d-flex justify-content-start">
                    <ul class="top_tools">
                        <li>
                            <div class="dropdown dropdown-access">
                                @if(isset(auth()->user()->id))
                                    @if(auth()->user()->hasRole(['admin', 'super admin']))
                                        <a href="{{ route('dashboard') }}" id="#" class="access_link" target="_blank"><span>Account</span></a>
                                    @else
                                        <a href="{{ route('login') }}" class="access_link"><span>Account</span></a>
                                    @endif
                                @else
                                    <a href="{{ route('login') }}" class="access_link"><span>Account</span></a>
                                @endif
                                @isset(auth()->user()->id)
                                <div class="dropdown-menu">
                                    @if(auth()->user()->hasRole(['admin', 'super admin']))
                                        <a href="{{ route('dashboard') }}" class="btn_1" target="_blank">DASHBOARD</a>
                                    @endif
                                    <ul>
                                        <li>
                                            <a href="#"><i class="ti-user"></i>My Profile</a>
                                        </li>
                                        <li>
                                        </li>
                                    {{-- <li>
                                        <a href="#menu" class="btn_cat_mob">
                                            <div class="hamburger hamburger--spin" id="hamburger">
                                                <div class="hamburger-box">
                                                    <div class="hamburger-inner"></div>
                                                </div>
                                            </div>
                                            Categories
                                        </a>
                                    </li> --}}
                                </ul>
                            </div>
                        </div>
                    </div>
                </div>
                <div class="search_mob_wp">
                    <input type="text" class="form-control" placeholder="Search over 10.000 products">
                    <input type="submit" class="btn_1 full-width" value="Search">
                </div>
            </div>
        </header>
```

beauty-fashion-shop/resources/views/frontend/layouts/footer.blade.php digunakan untuk mebuat kode pada kaki / footer sebuah website



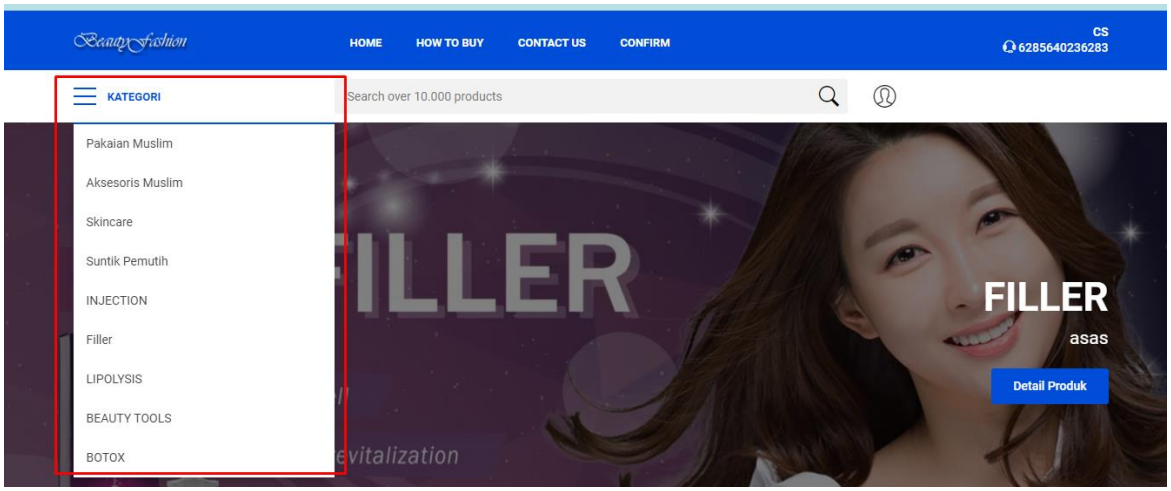
Untuk kodenya silahkan ketikan seperti berikut

```
<footer class="revealed">
    <div class="container">
        <div class="row">
            <div class="col-lg-3 col-md-6">
                <h3 data-bs-target="#collapse_1">Quick Links</h3>
                <div class="collapse dont-collapse-sm links" id="collapse_1">
                    <ul>
                        <li><a href="/about">About us</a></li>
                        <li><a href="/contacts">Contacts</a></li>
                    </ul>
                </div>
            </div>
            <div class="col-lg-3 col-md-6">
                <h3 data-bs-target="#collapse_2">Categories</h3>
                <div class="collapse dont-collapse-sm links" id="collapse_2">
                    <ul>
                        @php
                            $categories = \App\Models\Post\PostCategories::all();
                        @endphp
                        @foreach ($categories as $item)
                            <li><a href="/categories/{{ $item->slug }}">{{ $item->name }}</a></li>
                        @endforeach
                    </ul>
                </div>
            </div>
            <div class="col-lg-3 col-md-6">
                <h3 data-bs-target="#collapse_3">Contacts</h3>
                <div class="collapse dont-collapse-sm contacts" id="collapse_3">
                    <ul>
                        <li><i class="ti-home"></i>{{ website()->address }}</li>
                        <li><i class="ti-headphone-alt"></i>{{ website()->whatsapp }}</li>
                        <li><i class="ti-email"></i><a href="#0">{{ website()->email }}</a></li>
                        <li><i class="ti-credit-card"></i><a href="#0">{{ website()->data_rekening }}</a></li>
                    </ul>
                </div>
            </div>
            <div class="col-lg-3 col-md-6">
                <h3 data-bs-target="#collapse_4">Follow</h3>
                <div class="collapse dont-collapse-sm" id="collapse_4">
                    <div class="follow_us">
                        <ul>
                            <li><a href="{{ website()->twitter }}"><img src="" data-src="{{ a
                            <li><a href="{{ website()->facebook }}"><img src="" data-src="{{
                            <li><a href="{{ website()->instagram }}"><img src="" data-src="{{
                            <li><a href="{{ website()->tiktok }}"><img src="https://seeklogo.com/images/T/tiktok-logo-F8178CE216-seeklogo.com.png" data-src="https://seeklogo.c
                        </ul>
                    </div>
                </div>
            </div>
        </div>
        <!-- /row-->
        <hr>
```

Sementara untuk tampilan konten terletak pada direktori **beautyfashionshop/resources/views/frontend/layouts/master.blade.php**

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    {!! Meta::toHtml() !!}

    <!-- GOOGLE WEB FONT -->
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500;700&display=swap" rel="stylesheet">

    <!-- BASE CSS -->
    <link href="{{ asset('frontend/css') }}/bootstrap.min.css" rel="stylesheet">
    <link href="{{ asset('frontend/css') }}/style.css" rel="stylesheet">

    <!-- SPECIFIC CSS -->
    <link href="{{ asset('frontend/css') }}/home_1.css" rel="stylesheet">

    <!-- YOUR CUSTOM CSS -->
    <link href="{{ asset('frontend/css') }}/custom.css" rel="stylesheet">
    @livewireStyles
    @stack('styles')

</head>

<body>

    <div id="page">

    @include('frontend.layouts.header')

    {{ $slot }}

    @include('frontend.layouts.footer')

    </div>

    <div id="toTop"></div>

    <script src="{{ asset('frontend/js') }}/common_scripts.min.js"></script>
    <script src="{{ asset('frontend/js') }}/main.js"></script>

    <script src="{{ asset('frontend/js') }}/carousel-home.js"></script>
    <script src="{{ asset('frontend/js') }}/jquery.cookiebar.js"></script>
    <script>
        $(document).ready(function() {
            'use strict';
            $.cookieBar({
                fixed: true
            });
        });
    </script>
    @livewireScripts
    @stack('scripts')


</body>
</html>
```

Selanjutnya kita akan membahas bagian Navigasi / Navbar / Menu yang berfungsi untuk menghubungkan antara page / halamn satu ke halaman yang lain.



Untuk membuat kode tersebut silahkan siapkan sebuah direktori dan file dengan nama seperti berikut **beauty-fashion-shop/resources/views/frontend/layouts/navbar.blade.php**

```
@php
    $config = \App\Models\Admin\Configuration::latest()->first();
@endphp
<nav class="navbar navbar-expand-lg fixed-top sticky" id="navbar">
    <div class="container-fluid custom-container">
        <a class="navbar-brand text-dark fw-bold me-auto" href="/">
            <img src="{{ asset('frontend/assets/images').'/'.$config->logo }}" height="70" alt="" class="logo-dark" />
            <img src="{{ asset('frontend/assets/images').'/'.$config->logo }}" height="70" alt="" class="logo-light" />
            {{-- <span class="text-secondary">{{ strtoupper($config->name) }}</span> --}}
        </a>
        <div>
            <button class="navbar-toggler me-3" type="button" data-bs-toggle="collapse"
                data-bs-target="#navbarCollapse" aria-controls="navbarCollapse" aria-label="Toggle navigation">
                <i class="mdi mdi-menu"></i>
            </button>
        </div>

        <div class="collapse navbar-collapse" id="navbarCollapse">
            @php
                $menu = \App\Models\Menu::where('status', 1)->orderBy('order', 'ASC')->get();
            @endphp
            <ul class="navbar-nav mx-auto navbar-center">
                @foreach ($menu as $item)
                    @if ($item->parent_id == 0 && $item->category_id == 1)
                        <li class="nav-item {{ $item->type == 1 ? 'dropdown dropdown-hover' : '' }}">
                            <a class="nav-link" href="{{ $item->slug }}"
                                @if ($item->type == 1)
                                    id="jobsdropdown" role="button" data-bs-toggle="dropdown"
                                @endif
                                {{ $item->name }}

                                @if ($item->type == 1)
                                    <div class="arrow-down"></div>
                                @endif
                            </a>
                        @if ($item->type == 1)
                            <ul class="dropdown-menu dropdown-menu-center" aria-labelledby="jobsdropdown">
                                @foreach ($menu as $value)
                                    @if($value->parent_id == $item->id)
                                        <li><a class="dropdown-item" href="{{ $value->slug }}">{{ $value->name }}</a></li>
                                    @endif
                                @endforeach
                            </ul>
                        </li>
                        @endif
                    @endif
                @endforeach
                @if (Auth::check())
                    <li class="ms-4">
                        <ul class="header-menu list-inline d-flex align-items-center mb-0">
                            <li class="list-inline-item dropdown me-4">
                                @php
                                    $carts = App\Models\Store\Cart::where('user_id', auth()->user()->id)
                                            ->where('status_cart', 'cart')
                                            ->first();
                                @endphp
```

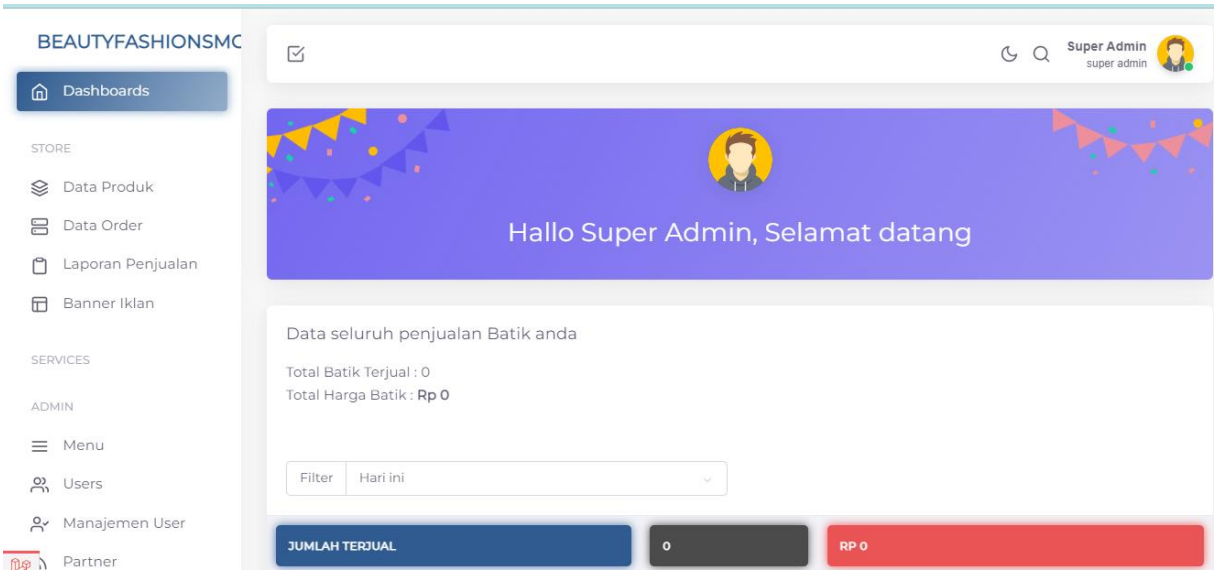
```
                        @endphp
                        <a href="javascript:void(0)" class="header-item noti-icon position-relative" id="notification" data-bs-toggle="dropdown"
                            aria-expanded="false">
                            <i class="mdi mdi-cart fs-22"></i>
                            <div class="count position-absolute">{{ isset($carts->detail) ? $carts->detail->count() :'0' }}</div>
                        </a>
                        <div class="dropdown-menu dropdown-menu-sm dropdown-menu-end p-0" aria-labelledby="notification">
                            <div class="notification-header border-bottom bg-light">
                                <h6 class="mb-1"> Keranjang Belanja </h6>
                                <p class="text-muted fs-13 mb-0">Kamu memiliki {{ isset($carts->detail) ? $carts->detail->count() :'0' }} produk dalam keranjang</p>
                            </div>
                            <div class="notification-wrapper dropdown-scroll">
                                @if (isset($carts->detail))
                                    @foreach ($carts->detail as $item)
                                        <a href="javascript:void(0)" class="text-dark notification-item d-block">
                                            <div class="d-flex align-items-center">
                                                <div class="flex-shrink-0 me-3">
                                                    <img src="{{ $item->produk->thumbnail }}" class="rounded-circle avatar-xs" alt="user-pic">
                                                </div>
                                                <div class="flex-grow-1">
                                                    <h6 class="mt-0 mb-1 fs-14">{{ $item->produk->title }}</h6>
                                                    <p class="text-muted fs-12 mb-0"><i class="mdi mdi-clock-outline"></i> <span> {{ $item->qty }} x {{ $item->harga }}
                                                </div>
                                            </div>
                                        </a><!--end notification-item-->
                                    @endforeach
                                @endif
```

```
                        $carts = App\Models\Store\Cart::where('user_id', auth()->user()->id)
                                ->where('status_cart', 'cart')
                                ->first();
                        @endphp
                        <a href="javascript:void(0)" class="header-item noti-icon position-relative" id="notification" data-bs-toggle="dropdown"
                            aria-expanded="false">
                            <i class="mdi mdi-cart fs-22"></i>
                            <div class="count position-absolute">{{ isset($carts->detail) ? $carts->detail->count() :'0' }}</div>
                        </a>
                        <div class="dropdown-menu dropdown-menu-sm dropdown-menu-end p-0" aria-labelledby="notification">
                            <div class="notification-header border-bottom bg-light">
                                <h6 class="mb-1"> Keranjang Belanja </h6>
                                <p class="text-muted fs-13 mb-0">Kamu memiliki {{ isset($carts->detail) ? $carts->detail->count() :'0' }} produk dalam keranjang</p>
                            </div>
                            <div class="notification-wrapper dropdown-scroll">
                                @if (isset($carts->detail))
                                    @foreach ($carts->detail as $item)
                                        <a href="javascript:void(0)" class="text-dark notification-item d-block">
                                            <div class="d-flex align-items-center">
                                                <div class="flex-shrink-0 me-3">
                                                    <img src="{{ $item->produk->thumbnail }}" class="rounded-circle avatar-xs" alt="user-pic">
                                                </div>
                                                <div class="flex-grow-1">
                                                    <h6 class="mt-0 mb-1 fs-14">{{ $item->produk->title }}</h6>
                                                    <p class="text-muted fs-12 mb-0"><i class="mdi mdi-clock-outline"></i> <span> {{ $item->qty }} x {{ $item->harga
                                                </div>
                                            </div>
```

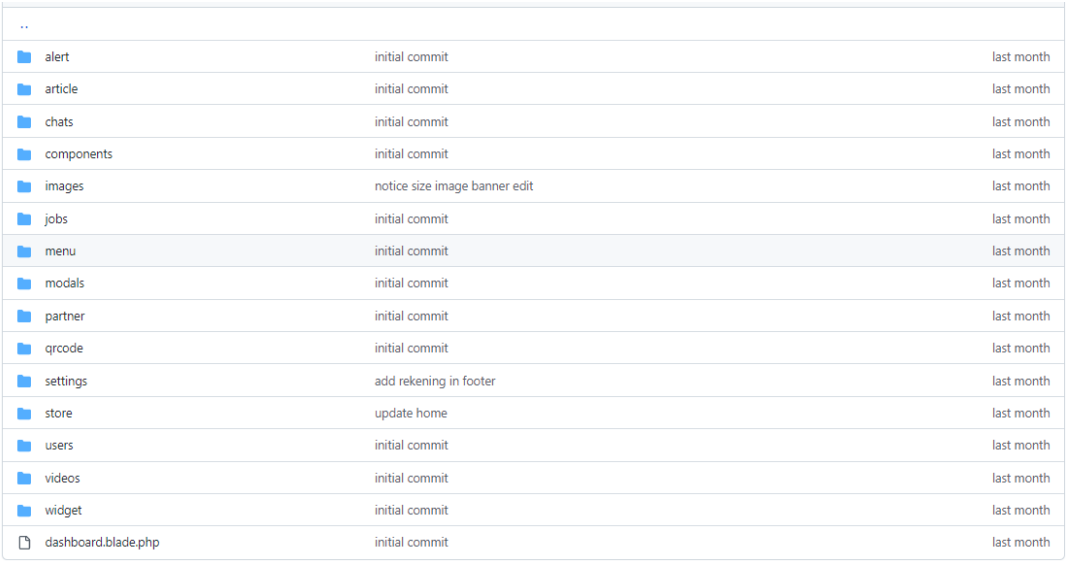
```
                                <a class="primary-link fs-13" href="/cart">
                                    <i class="mdi mdi-arrow-right-circle me-1"></i> <span>Lebih lengkap..</span>
                                </a>
                            </div>
                        </div>
                    </li>
                </ul>
            </li>
            <li class="list-inline-item dropdown">
                <a href="javascript:void(0)" class="header-item" id="userdropdown" data-bs-toggle="dropdown"
                    aria-expanded="false">
                    <img src="{{ auth()->user()->profile_photo_path == null ? auth()->user()->profile_photo_url : auth()->user()->avatar }}" alt="mdo" width="35" heigh
                </a>
                <ul class="dropdown-menu dropdown-menu-end" aria-labelledby="userdropdown">
                    @role('super admin|admin|author|editor')
                    <li><a class="dropdown-item" href="/admin">Admin</a></li>
                    @endrole
                    <li><a class="dropdown-item" href="/cart">Kerajang Belanja</a></li>
                    <li><a class="dropdown-item" href="{{ route('profile', ['id' => auth()->user()->id]) }}">My Profile</a></li>
                    <form method="POST" action="{{ route('logout') }}">
                        @csrf
                        <li>
                            <a class="dropdown-item" href="{{ route('logout') }}" onclick="event.preventDefault();
                                        this.closest('form').submit(); " role="button">
                                <i class="mr-50" data-feather="power"></i>

                                {{ __('Logout') }}
                            </a>
                        </li>
                    </form>
                </ul>
```

Dengan kode diatas kita telah membuat sebuah menu / navigasi.

Selanjutnya kita akan membahas halama admin



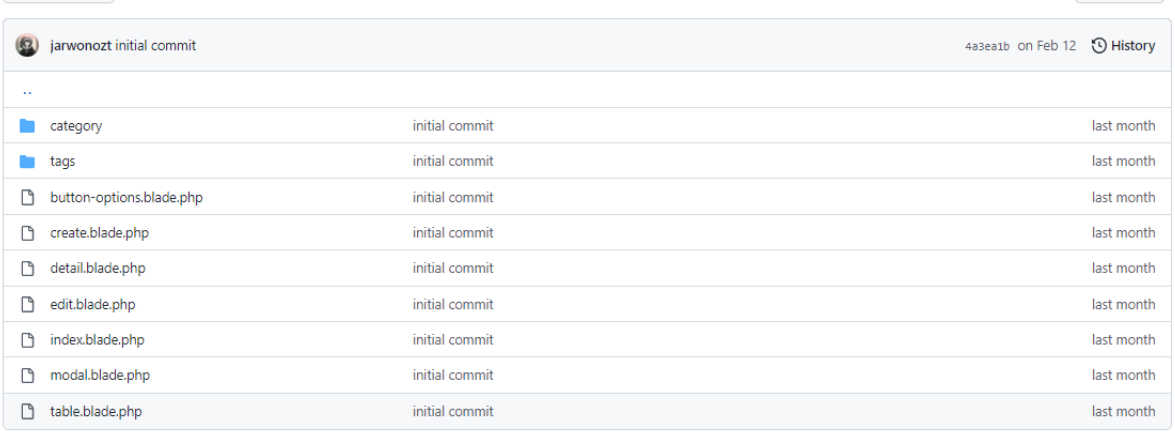
siapkan direktori dan file beauty-fashion-shop/resources/views/**admin**/ , untuk struktur file dan folder adalah sebagai berikut



Mari kita bedah penulisan kode pada struktur data diatas, yang pertama adalah file alert **beauty-fashion-shop/resources/views/admin/alert/alert.blade.php** file ini berfungsi sebagai sebuah alarm/ notifikasi.

```
@if (session()->has('success'))
    <div class="alert alert-primary" role="alert">
        <div class="alert-body">
            <i data-feather="check"></i>
            <span> {{ session('success') }}</span>
        </div>
    </div>
@endif
@if (session()->has('error'))
    <div class="alert alert-danger" role="alert">
        <div class="alert-body">
            <i data-feather="alert-circle"></i>
            <span> {{ session('error') }}</span>
        </div>
    </div>
@endif
```

beauty-fashion-shop/resources/views/admin/**article**/ digunakan untuk membuat post atau artikel yang terkait dengan produk pada sebuah toko online, terdapat bebrapa file didalamnya.



jarwonozt initial commit — 4a3ea1b on Feb 12 — History

..		
category	initial commit	last month
tags	initial commit	last month
button-options.blade.php	initial commit	last month
create.blade.php	initial commit	last month
detail.blade.php	initial commit	last month
edit.blade.php	initial commit	last month
index.blade.php	initial commit	last month
modal.blade.php	initial commit	last month
table.blade.php	initial commit	last month

Kita bahas mula file **index.blade.php**

```
@section('title')
    Posts -
@endsection

<x-master-layouts>

@include('sweetalert::alert')
<div class="app-content content ">
    <div class="content-overlay"></div>
    <div class="header-navbar-shadow"></div>
    <div class="content-wrapper">
        <div class="content-header row">
            <div class="content-header-left col-md-7 col-12 mb-2">
                <div class="row breadcrumbs-top">
                    <div class="col-12">
                        <h2 class="content-header-title float-left mb-0">Articles</h2>
                        <div class="breadcrumb-wrapper">
                            <ol class="breadcrumb">
                                <li class="breadcrumb-item"><a href="/dashboard">Home</a>
                                </li>
                                <li class="breadcrumb-item active">Article List
                                </li>
                            </ol>
                        </div>
                    </div>
                </div>
            </div>
            <div class="content-header-right text-md-right col-md-5 col-12">
                <div class="form-group">
                    <a href="{{ route('articles.create') }}" class="btn-icon btn btn-primary btn-round"><i data-feather="edit"></i> Create New Article</a>
                </div>
            </div>
        </div>
        <div class="content-body">
            <div class="row" id="basic-table">
                <div class="col-12">
                    <div class="card p-1">
                        <div class="row" id="table-hover-row">
                            <div class="col-12">
                                <div class="card">
                                    <div class="table-responsive">
                                        <livewire:articles-table></livewire:articles-table>
                                    </div>
                                </div>
                            </div>
                        </div>
                        {{-- <livewire:user-table></livewire:user-table> --}}
                    </div>
                </div>
            </div>
            @include('admin.article.modal')
            @include('admin.modals.alert')
        </div>
    </div>
</div>
```

```
@push('custom-scripts')
<script>
    window.addEventListener('success', event => {
        $("#successAlert").modal('show');
        $("#message").html(event.detail.message);

    });
</script>
<script>
    document.addEventListener("DOMContentLoaded", () => {
        $(function () {
            $('[data-toggle="tooltip"]').tooltip()
        })
        Livewire.hook('message.processed', (message, component) => {
            $(function () {
                $('[data-toggle="tooltip"]').tooltip()
            })
        })
    });

    window.addEventListener('openModalDelete', event => {
        $("#delete-modal").modal('show');
    });

    window.addEventListener('closeModalDelete', event => {
        $("#delete-modal").modal('hide');
    });
</script>
@endpush

</x-master-layouts>
```

detail.blade.php

```
@section('title')
{!! $data->title !!} -
@endsection

<x-master-layouts>
    @include('sweetalert::alert')

    <div class="app-content content ">
        <div class="content-overlay"></div>
        <div class="header-navbar-shadow"></div>
        <div class="content-wrapper">
            <div class="content-header row">
                <div class="content-header-left col-md-9 col-12 mb-2">
                    <div class="row breadcrumbs-top">
                        <div class="col-12">
                            <h2 class="content-header-title float-left mb-0">Article Detail</h2>
                            <div class="breadcrumb-wrapper">
                                <ol class="breadcrumb">
                                    <li class="breadcrumb-item"><a href="index.html">Home</a>
                                    </li>
                                    <li class="breadcrumb-item"><a href="{{ route('articles.index') }}">Articles</a>
                                    </li>
                                    <li class="breadcrumb-item active">Detail
                                    </li>
                                </ol>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
```

```
<div class="content-detached">
    <div class="content-body">
        <div class="blog-detail-wrapper">
            <div class="row">
                <div class="col-12">
                    <div class="card">
                        {{-- <img src="{{ $data->image ? asset(config('app.POST_BIG')).'/'.$data->image : asset('assets').'/images/banner/banner-12.jpg'}}" class="

                        <div class="card-body">
                            <h4 class="card-title">{!! $data->title !!} </h4>

                            <div class="media">
                                <div class="avatar mr-50">
                                    <img src="{{ $data->getUser->profile_photo_path ? asset('storage/images/users').'/'.$data->getUser->profile_photo_path : asset(
                                </div>
                                <div class="media-body">
                                    <small class="text-muted mr-25">by</small>
                                    <small><a href="javascript:void(0);" class="text-body">{{ $data->getUser->name }}</a></small>
                                    <span class="text-muted ml-50 mr-25">|</span>
                                    <small class="text-muted">{{ \Carbon\Carbon::parse($data->updated_at)->Format('d M Y - H:i') }} | </small>
                                    <a href="javascript:void(0);">
                                        <div class="badge badge-light-primary">{{ strtoupper($data->getCategory->name) }}</div>
                                    </a>
                                </div>
                            </div>
                            <div class="my-1 py-25">
                                tags :
                                @php
                                    $explode = explode(',', $data->tags);
                                    $tags    = \App\Models\Tag::whereIn('id', $explode)->get();
                                @endphp
                                  @foreach ( $tags as $tag)
                                    <a href="/tags/{{ $tag->slug }}">
                                        <div class="badge badge-pill badge-light-danger mr-50">
                                            #{{ $tag->title }}
                                        </div>
                                    </a>
                                  @endforeach

                            </div>
                            <div class="media">
                                <div class="media-left">
                                    <img src="{{ $data->image ? asset(config('app.POST_MID')).'/'.$data->image : asset('assets').'/images/portrait/small/avatar-s-6
                                </div>
                                <div class="media-body ml-2 text-justify">
                                    {!! $data->content !!}
                                </div>
                            </div>
                            <hr class="my-2" />
                            <div class="d-flex align-items-center justify-content-between">
                                <div class="d-flex align-items-center">
                                    <div class="d-flex align-items-center mr-1">
                                        <a href="javascript:void(0);" class="mr-50">
                                            <i data-feather="message-square" class="font-medium-5 text-body align-middle"></i>
                                        </a>
                                        <a href="javascript:void(0);">
                                            <div class="text-body align-middle">{{ $data->comment_count }}</div>
                                        </a>
                                    </div>
                                         <div class="d-flex align-items-center">
                                            <a href="javascript:void(0);" class="mr-50">
                                                <i data-feather="thumbs-up" class="font-medium-5 text-body align-middle"></i>
                                            </a>
                                            <a href="javascript:void(0);">
                                                <div class="text-body align-middle">{{ $data->like_count }}</div>
                                            </a>
                                        </div>
                                    </div>
                                </div>
                            </div>
                        </div>
                    </div>
                    <!--/ Blog -->

                    <!-- Blog Comment -->
                    <div class="col-12 mt-1" id="blogComment">
                        <h6 class="section-label mt-25">Comment</h6>
                        @livewire('article.replay-comment-article', [
                            'articleId' => $data->id,
                        ])

```

```
                        </div>
                        <!--/ Blog Comment -->
                    </div>
                </div>
                <!--/ Blog Detail -->

            </div>
        </div>
    </div>
</div>


@push('vendor-css')
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/vendors/css/vendors.min.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/vendors/css/forms/select/select2.min.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/vendors/css/editors/quill/katex.min.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/vendors/css/editors/quill/monokai-sublime.min.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/vendors/css/editors/quill/quill.snow.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/vendors/css/pickers/pickadate/pickadate.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/vendors/css/pickers/flatpickr/flatpickr.min.css">
@endpush
@push('page-css')
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/css/core/menu/menu-types/vertical-menu.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/css/plugins/forms/form-quill-editor.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/css/plugins/forms/pickers/form-flat-pickr.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/css/plugins/forms/pickers/form-pickadate.css">
@endpush

@push('custom-scripts')
<script src="{{ asset('assets') }}/vendors/js/pickers/pickadate/picker.js"></script>
<script src="{{ asset('assets') }}/vendors/js/pickers/pickadate/picker.date.js"></script>
<script src="{{ asset('assets') }}/vendors/js/pickers/pickadate/picker.time.js"></script>
<script src="{{ asset('assets') }}/vendors/js/pickers/pickadate/legacy.js"></script>
<script src="{{ asset('assets') }}/vendors/js/pickers/flatpickr/flatpickr.min.js"></script>
<script src="{{ asset('assets') }}/vendors/js/forms/select/select2.full.min.js"></script>
<script src="{{ asset('assets') }}/vendors/js/editors/quill/katex.min.js"></script>
<script src="{{ asset('assets') }}/vendors/js/editors/quill/highlight.min.js"></script>
<script src="{{ asset('assets') }}/vendors/js/editors/quill/quill.min.js"></script>
@endpush
@push('page-js')
<script src="{{ asset('assets') }}/js/scripts/pages/page-blog-edit.js"></script>
<script src="{{ asset('assets') }}/ckeditorx/ckeditor.js"></script>
<script src="{{ asset('assets') }}/js/scripts/forms/pickers/form-pickers.js"></script>
<script type="text/javascript">
    $(document).ready(function () {
        $('.ckeditor').ckeditor();
    });
</script>
<script>
    function edValueKeyPress() {
        var edValue = document.getElementById("title");
        var s = edValue.value;

        var lblValue = document.getElementById("slug");
        lblValue.value = s.toLowerCase().replace(/[^\w-]+/g, '-');
    }

</script>

@endpush
</x-master-layouts>
```

create.blade.php untuk membuat sebuah file artikel / berita

```
@section('title')
    Create Post -
@endsection

<x-master-layouts>
@include('sweetalert::alert')

<div class="app-content content ">
    <div class="content-overlay"></div>
    <div class="header-navbar-shadow"></div>
    <div class="content-wrapper">
        <div class="content-header row">
            <div class="content-header-left col-md-9 col-12 mb-2">
                <div class="row breadcrumbs-top">
                    <div class="col-12">
                        <h2 class="content-header-title float-left mb-0">Create New Article</h2>
                        <div class="breadcrumb-wrapper">
                            <ol class="breadcrumb">
                                <li class="breadcrumb-item"><a href="index.html">Home</a>
                                </li>
                                <li class="breadcrumb-item"><a href="{{ route('articles.index') }}">Articles</a>
                                </li>
                                <li class="breadcrumb-item active">New
                                </li>
                            </ol>
                        </div>
                    </div>
                </div>
            </div>
            <div class="content-header-right text-md-right col-md-3 col-12 d-md-block d-none">
                <div class="form-group breadcrumb-right">
                    <div class="dropdown">
                        <button class="btn-icon btn btn-primary btn-round btn-sm dropdown-toggle" type="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="fals
                        <div class="dropdown-menu dropdown-menu-right"><a class="dropdown-item" href="app-todo.html"><i class="mr-1" data-feather="check-square"></i><span clas
                    </div>
                </div>
            </div>
        </div>
    </div>
    <div class="content-body">
        <!-- Blog Edit -->
        <div class="blog-edit-wrapper">
            <div class="row">
                <div class="col-12">
                    <div class="card">
                        <div class="card-body">
                            <div class="media">
                                <div class="avatar mr-75">
                                    <img src="{{ asset('assets') }}/images/portrait/small/avatar-s-9.jpg" width="38" height="38" alt="Avatar" />
                                </div>
                                <div class="media-body">
                                    <h6 class="mb-25">{{ auth()->user()->name }}</h6>
                                    <p class="card-text">{{ \Carbon\Carbon::now()->format('D, d M Y') }}</p>
                                </div>
                            </div>
                            <!-- Form -->
                            <form action="{{ route('articles.store') }}" method="POST" enctype="multipart/form-data" class="mt-2">
                                @csrf
                                <div class="row">
                                    <div class="col-md-6 col-12">
                                        <div class="form-group mb-2">
                                            <label for="blog-edit-title">Title</label>
                                            <input type="text" name="title" id="title" value="{{ old('title') }}" class="form-control" onInput="edValueKeyPress()">
                                        </div>
```

```
                </div>
                <div class="col-md-6 col-12">
                    <div class="form-group mb-2">
                        <label for="fp-date-time">Date & Time</label>
                        <input type="text" id="fp-date-time" name="date" class="form-control flatpickr-date-time" placeholder="YYYY-MM-DD HH:MM" />
                    </div>
                </div>

                <div class="col-md-6 col-12">
                    <div class="form-group mb-2">
                        <label for="blog-edit-slug">Slug</label>
                        <input type="text" name="slug" id="slug" class="form-control">
                    </div>
                </div>
                <div class="col-md-6 col-12 mb-1">
                    <label for="blog-edit-category">Category</label>
                    <select class="select2 form-control" name="category">

                        @foreach ($categories as $category)
                            <option value="{{ $category->id }}">{{ strtoupper($category->name) }}</option>
                        @endforeach
                    </select>
                </div>
                <div class="col-12 mb-2">
                    <label for="blog-edit-title">Post Type</label>
                    <div class="demo-inline-spacing">
                        <div class="custom-control custom-radio">
                            <input type="radio" id="customRadio1" name="type" value="0" class="custom-control-input" checked />
                            <label class="custom-control-label" for="customRadio1">Standard</label>
                        </div>
                        <div class="custom-control custom-control-warning custom-radio">
                            <input type="radio" id="customRadio2" name="type" value="1" class="custom-control-input" />
                            <label class="custom-control-label" for="customRadio2">Headline</label>
                        </div>
                        <div class="custom-control custom-control-success custom-radio">
                            <input type="radio" id="customRadio3" name="type" value="2" class="custom-control-input" />
                            <label class="custom-control-label" for="customRadio3">Latest Slider</label>
                        </div>
                    </div>
                </div>
                <div class="col-12 mb-2">
                    <label for="blog-edit-title">Content</label>
                    <textarea name="content" class="ckeditor" id="" cols="30" rows="10">{{ old('content') }}</textarea>
                </div>
                <div class="col-md-6 col-12">
                    <div class="form-group mb-2">
                        <label for="blog-edit-status">Status</label>
                        <select class="form-control" id="blog-edit-status" name="status">
                            <option value="1">Published</option>
                            <option value="2">Pending</option>
                            <option value="3">Draft</option>
                        </select>
                    </div>
                    <div class="form-group mb-2">
                        {{-- <label for="blog-edit-title">Tags</label> --}}
                        <label>Tags</label>
                        <select class="select2 form-control" name="tags[]" multiple>
                            <optgroup label="Tags">
                                @foreach ($tags as $tag)
                                    <option value="{{ $tag->id }}">{{ $tag->title }}</option>
                                @endforeach
                            </optgroup>
                        </select>
                    </div>
                </div>

                <div class="col-md-6 col-12">
                    <div class="border rounded p-2">
                        <h4 class="mb-1">Image</h4>
                        <div class="media flex-column flex-md-row">
                            <img src="{{ asset('assets') }}/images/slider/03.jpg" id="blog-feature-image" class="rounded mr-2 mb-1 mb-md-0" width="170"
                            <div class="media-body">
                                <div class="d-inline-block">
                                    <div class="form-group mb-0">
                                        <input class="w-50" type="file" id="pic-form" name="image" accept="image/*">
                                        <input type="hidden" name="16_9_width" id="16_9_width"/>
                                        <input type="hidden" name="16_9_height" id="16_9_height"/>
                                        <input type="hidden" name="16_9_x" id="16_9_x"/>
                                        <input type="hidden" name="16_9_y" id="16_9_y"/>

                                        <input type="hidden" name="4_3_width" id="4_3_width"/>
                                        <input type="hidden" name="4_3_height" id="4_3_height"/>
                                        <input type="hidden" name="4_3_x" id="4_3_x"/>
                                        <input type="hidden" name="4_3_y" id="4_3_y"/>

                                        <input type="hidden" name="1_1_width" id="1_1_width"/>
                                        <input type="hidden" name="1_1_height" id="1_1_height"/>
                                        <input type="hidden" name="1_1_x" id="1_1_x"/>
                                        <input type="hidden" name="1_1_y" id="1_1_y"/>
                                    </div>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
```

```
                                    <div class="col-12 mt-5 d-flex justify-content-end">
                                        <button type="submit" class="btn btn-primary mr-1">Save Changes</button>
                                        <a href="{{ route('articles.index') }}" class="btn btn-outline-secondary">Cancel</a>
                                    </div>
                                </div>
                            </form>
                        </div>
                    </div>
                </div>
            </div>

        </div>
    </div>
</div>

<div class="modal fade" id="modal-default" tabindex="-1" role="dialog" aria-labelledby="modalLabel" aria-hidden="true">
    <div class="modal-dialog modal-lg" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title">Crop Photo</h5>
                <button type="button" class="close" id="closeAtas" aria-label="Close">
                    <span aria-hidden="true">&times;</span>
                </button>
            </div>
            <div class="modal-body">
                <div class="row d-flex justify-content-center">
                    <div class="col-12 d-flex justify-content-center">
                        <div class="img-container p-2 text-center">
                            <div id="preview-16-9" class="d-none"></div>
                            <h4 class="text-primary">Aspect Ratio 16:9</h4>
                            <img id="16-9-show">
                        </div>
                    </div>
                    <div class="col-12 d-flex justify-content-center">
                        <div class="img-container p-2 text-center">
                            <div id="preview-4-3" class="d-none"></div>
                            <h4 class="text-primary">Aspect Ratio 4:3</h4>
                            <img id="4-3-show">
                        </div>
                    </div>
                    <div class="col-12 d-flex justify-content-center">
                        <div class="img-container p-2 text-center">
                            <div id="preview-1-1" class="d-none"></div>
                            <h4 class="text-primary">Aspect Ratio 1:1</h4>
                            <img id="1-1-show">
                        </div>
                    </div>
                </div>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-primary" data-dismiss="modal">Crop</button>
                <button type="button" class="btn btn-secondary" aria-label="Close" id="onClose">Close</button>
            </div>
        </div>
    </div>
</div>
@push('vendor-css')
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/vendors/css/vendors.min.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/vendors/css/forms/select/select2.min.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/vendors/css/editors/quill/katex.min.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/vendors/css/editors/quill/monokai-sublime.min.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/vendors/css/editors/quill/quill.snow.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/vendors/css/pickers/pickadate/pickadate.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/vendors/css/pickers/flatpickr/flatpickr.min.css">
@endpush
```

```
@push('page-css')
<link rel="stylesheet" href="{{asset('backend/plugins/bootstrap-fileinput/css/fileinput.css')}}">
<link rel="stylesheet" href="{{ asset('assets') }}/vendors/cropperjs/cropper.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/css/core/menu/menu-types/vertical-menu.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/css/plugins/forms/form-quill-editor.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/css/plugins/forms/pickers/form-flat-pickr.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/css/plugins/forms/pickers/form-pickadate.css">
@endpush
@push('custom-scripts')
<script src="{{ asset('assets') }}/vendors/js/pickers/pickadate/picker.js"></script>
<script src="{{ asset('assets') }}/vendors/js/pickers/pickadate/picker.date.js"></script>
<script src="{{ asset('assets') }}/vendors/js/pickers/pickadate/picker.time.js"></script>
<script src="{{ asset('assets') }}/vendors/js/pickers/pickadate/legacy.js"></script>
<script src="{{ asset('assets') }}/vendors/js/pickers/flatpickr/flatpickr.min.js"></script>
<script src="{{ asset('assets') }}/vendors/js/forms/select/select2.full.min.js"></script>
<script src="{{ asset('assets') }}/vendors/js/editors/quill/katex.min.js"></script>
<script src="{{ asset('assets') }}/vendors/js/editors/quill/highlight.min.js"></script>
<script src="{{ asset('assets') }}/vendors/js/editors/quill/quill.min.js"></script>
@endpush
@push('page-js')
<script src="{{asset('backend/plugins/bootstrap-fileinput/js/fileinput.js')}}"></script>
<script src="{{asset('backend/plugins/bootstrap-fileinput/themes/fa/theme.js')}}"></script>
<script src="{{ asset('assets') }}/vendors/cropperjs/cropper.js"></script>
<script src="{{ asset('assets') }}/js/scripts/pages/page-blog-edit.js"></script>
<script src="{{ asset('assets') }}/ckeditorx/ckeditor.js"></script>
<script src="{{ asset('assets') }}/js/scripts/forms/pickers/form-pickers.js"></script>
<script src="{{ asset('assets') }}/js/scripts/forms/form-select2.js"></script>
<script type="text/javascript">
    $(document).ready(function () {
        $('.ckeditor').ckeditor();
    });
</script>
<script>
    function edValueKeyPress() {
        var edValue = document.getElementById("title");
        var s = edValue.value;

        var lblValue = document.getElementById("slug");
        lblValue.value = s.toLowerCase().replace(/[^\w-]+/g, '-');
    }

</script>

<script src="{{ asset('assets') }}/js/scripts/forms/form-crop-image.js"></script>

@endpush
</x-master-layouts>
```

button-options.blade.php

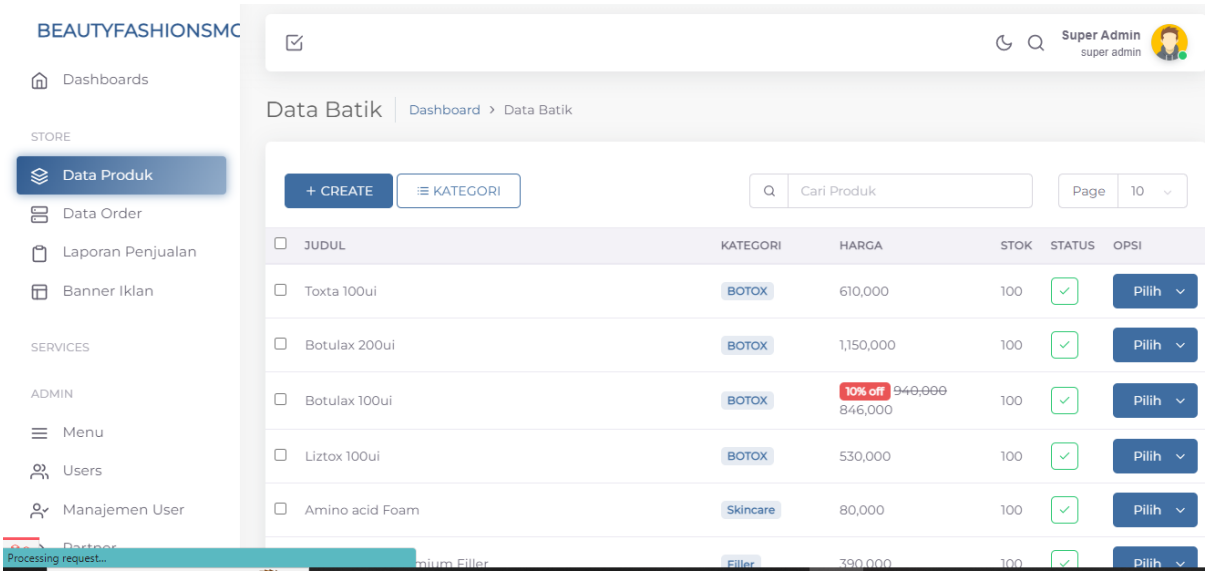
```
{{-- <button class="btn btn-success">test</button> --}}
@if ($status)
    <svg xmlns="http://www.w3.org/2000/svg" class="inline-block h-5 w-5 @if ($successValue === true) text-green-500 @else text-red-500 @endif" fill="none" viewBox="0 0 24 24"
        <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M9 12l2 2 4-4m6 2a9 9 0 11-18 0 9 9 0 0118 0z" />
    </svg>
@else
    <svg xmlns="http://www.w3.org/2000/svg" class="inline-block h-5 w-5 @if ($successValue === false) text-green-500 @else text-red-500 @endif" fill="none" viewBox="0 0 24 24'
        <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M10 14l2-2m0 0l2-2m-2 2l-2-2m2 2l2 2m7-2a9 9 0 11-18 0 9 9 0 0118 0z" />
    </svg>
@endif
```

chats/index.blade.php digunakan ketika pemebeli hendak berdiskusi langsung tentang barang / produk kepada penjual.

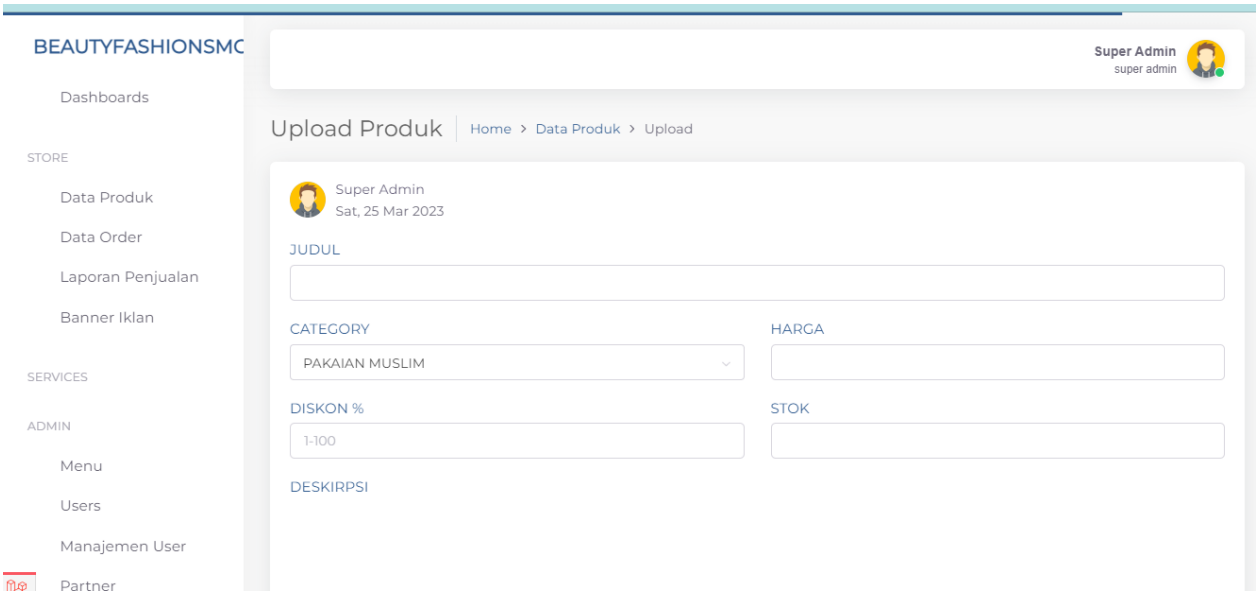
```
@extends('layouts.master')
@section('content')
@include('sweetalert::alert')
<div class="app-content content chat-application">
    <div class="content-overlay"></div>
    <div class="header-navbar-shadow"></div>
    <div class="content-area-wrapper">
        <div class="sidebar-left">
            <div class="sidebar">
                <!-- Admin user profile area -->
                <div class="chat-profile-sidebar">
                    <header class="chat-profile-header">
                        <span class="close-icon">
                            <i data-feather="x"></i>
                        </span>
                        <!-- User Information -->
                        <div class="header-profile-sidebar">
                            <div class="avatar box-shadow-1 avatar-xl avatar-border">
                                <img src="{{asset('assets')}}/images/portrait/small/avatar-s-11.jpg" alt="user_avatar" />
                                <span class="avatar-status-online avatar-status-xl"></span>
                            </div>
                            <h4 class="chat-user-name">John Doe</h4>
                            <span class="user-post">Admin</span>
                        </div>
                        <!--/ User Information -->
                    </header>
                    <!-- User Details start -->
                    <div class="profile-sidebar-area">
                        <h6 class="section-label mb-1">About</h6>
                        <div class="about-user">
                            <textarea data-length="120" class="form-control char-textarea" id="textarea-counter" rows="5" placeholder="About User">
 Dessert chocolate cake lemon drops jujubes. Biscuit cupcake ice cream bear claw brownie brownie marshmallow.</textarea>
                            <small class="counter-value float-right"><span class="char-count">108</span> / 120 </small>
                        </div>
                        <!-- To set user status -->
                        <h6 class="section-label mb-1 mt-3">Status</h6>
                        <ul class="list-unstyled user-status">
                            <li class="pb-1">
                                <div class="custom-control custom-control-success custom-radio">
                                    <input type="radio" id="activeStatusRadio" name="userStatus" class="custom-control-input" value="online" checked />
                                    <label class="custom-control-label ml-25" for="activeStatusRadio">Active</label>
                                </div>
                            </li>
                            <li class="pb-1">
                                <div class="custom-control custom-control-danger custom-radio">
                                    <input type="radio" id="dndStatusRadio" name="userStatus" class="custom-control-input" value="busy" />
                                    <label class="custom-control-label ml-25" for="dndStatusRadio">Do Not Disturb</label>
                                </div>
                            </li>
                            <li class="pb-1">
                                <div class="custom-control custom-control-warning custom-radio">
                                    <input type="radio" id="awayStatusRadio" name="userStatus" class="custom-control-input" value="away" />
                                    <label class="custom-control-label ml-25" for="awayStatusRadio">Away</label>
                                </div>
                            </li>
```

```
                        <li class="pb-1">
                            <div class="custom-control custom-control-secondary custom-radio">
                                <input type="radio" id="offlineStatusRadio" name="userStatus" class="custom-control-input" value="offline" />
                                <label class="custom-control-label ml-25" for="offlineStatusRadio">Offline</label>
                            </div>
                        </li>
                    </ul>
                    <!--/ To set user status -->

                    <!-- User settings -->
                    <h6 class="section-label mb-1 mt-2">Settings</h6>
                    <ul class="list-unstyled">
                        <li class="d-flex justify-content-between align-items-center mb-1">
                            <div class="d-flex align-items-center">
                                <i data-feather="check-square" class="mr-75 font-medium-3"></i>
                                <span class="align-middle">Two-step Verification</span>
                            </div>
                            <div class="custom-control custom-switch mr-0">
                                <input type="checkbox" class="custom-control-input" id="customSwitch1" checked />
                                <label class="custom-control-label" for="customSwitch1"></label>
                            </div>
                        </li>
                        <li class="d-flex justify-content-between align-items-center mb-1">
                            <div class="d-flex align-items-center">
                                <i data-feather="bell" class="mr-75 font-medium-3"></i>
                                <span class="align-middle">Notification</span>
                            </div>
                            <div class="custom-control custom-switch mr-0">
                                <input type="checkbox" class="custom-control-input" id="customSwitch2" />
                                <label class="custom-control-label" for="customSwitch2"></label>
                            </div>
                        </li>
                        <li class="mb-1 d-flex align-items-center cursor-pointer">
                            <i data-feather="user" class="mr-75 font-medium-3"></i>
                            <span class="align-middle">Invite Friends</span>
                        </li>
                        <li class="d-flex align-items-center cursor-pointer">
                            <i data-feather="trash" class="mr-75 font-medium-3"></i>
                            <span class="align-middle">Delete Account</span>
                        </li>
                    </ul>
                    <!--/ User settings -->

                    <!-- Logout Button -->
                    <div class="mt-3">
                        <button class="btn btn-primary">
                            <span>Logout</span>
                        </button>
                    </div>
                    <!--/ Logout Button -->
                </div>
                <!-- User Details end -->
            </div>
            <!--/ Admin user profile area -->

---
        <!-- Chat Sidebar area -->
        <div class="sidebar-content">
            <span class="sidebar-close-icon">
                <i data-feather="x"></i>
            </span>
            <!-- Sidebar header start -->
            <div class="chat-fixed-search">
                <div class="d-flex align-items-center w-100">
                    <div class="sidebar-profile-toggle">
                        <div class="avatar avatar-border">
                            <img src="{{asset('assets')}}/images/portrait/small/avatar-s-11.jpg" alt="user_avatar" height="42" width="42" />
                            <span class="avatar-status-online"></span>
                        </div>
                    </div>
                    <div class="input-group input-group-merge ml-1 w-100">
                        <div class="input-group-prepend">
                            <span class="input-group-text round"><i data-feather="search" class="text-muted"></i></span>
                        </div>
                        <input type="text" class="form-control round" id="chat-search" placeholder="Search or start a new chat" aria-label="Search..." aria-describedby
                    </div>
                </div>
            </div>
            <!-- Sidebar header end -->

            <!-- Sidebar Users start -->
            <div id="users-list" class="chat-user-list-wrapper list-group">
                <h4 class="chat-list-title">Chats</h4>
                <ul class="chat-users-list chat-list media-list">
                    <li>
                        <span class="avatar"><img src="{{asset('assets')}}/images/portrait/small/avatar-s-3.jpg" height="42" width="42" alt="Generic placeholder image"
```

Selanjutnya mari kita bahas form dalam penggunaan transaksi pada sebuah toko online



Data produk digunakan untuk menambahkan data – data produk akan dijual pada sebuah toko online



Untuk menambah kita bisa klik tombol cretate dan akan mucul sebuah form untuk input data produk, silahkan isikan data dengan lengkap selanjutnya bisa di klik tombol save. Maka data produk sudah siap untuk dijual dalam toko online, untuk file nya adalah sebagai berikut :

index.blade.php

```
@section('title')
    Data Batik -
@endsection

<x-master-layouts>
    @include('sweetalert::alert')
    <div class="app-content content ">
        <div class="content-overlay"></div>
        <div class="header-navbar-shadow"></div>
        <div class="content-wrapper">
            <div class="content-header row">
                <div class="content-header-left col-md-12 col-12 mb-2">
                    <div class="row breadcrumbs-top">
                        <div class="col-12">
                            <h2 class="content-header-title float-left mb-0">Data Batik</h2>
                            <div class="breadcrumb-wrapper">
                                <ol class="breadcrumb">
                                    <li class="breadcrumb-item"><a href="/dashboard">Dashboard</a>
                                    </li>
                                    <li class="breadcrumb-item active">Data Batik
                                    </li>
                                </ol>
                            </div>
                        </div>
                    </div>
                </div>
            </div>

            @livewire('store.book-table')
        </div>
    </div>
</x-master-layouts>
```

detail.blade.php

```
@section('title')
{!! $data->title !!} -
@endsection

<x-master-layouts>
    @include('sweetalert::alert')

    <div class="app-content content ">
        <div class="content-overlay"></div>
        <div class="header-navbar-shadow"></div>
        <div class="content-wrapper">
            <div class="content-header row">
                <div class="content-header-left col-md-9 col-12 mb-2">
                    <div class="row breadcrumbs-top">
                        <div class="col-12">
                            <h2 class="content-header-title float-left mb-0">Produk Detail</h2>
                            <div class="breadcrumb-wrapper">
                                <ol class="breadcrumb">
                                    <li class="breadcrumb-item"><a href="/dashboard">Home</a>
                                    </li>
                                    <li class="breadcrumb-item"><a href="{{ route('products.index') }}">Data Produk</a>
                                    </li>
                                    <li class="breadcrumb-item active">Detail
                                    </li>
                                </ol>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
        <div class="content-detached">
            <div class="content-body">
                <!-- app e-commerce details start -->
                <section class="app-ecommerce-details">
                    <div class="card">
                        <!-- Product Details starts -->
                        <div class="card-body">
                            <div class="row my-2">
                                <div class="col-12 col-md-5 d-flex align-items-center justify-content-center mb-2 mb-md-0">
                                    <div class="d-flex align-items-center justify-content-center">
                                        <img src="{{ asset('/storage/pictures/product/1_1/mid/') }}/{{ $data->image }}" class="img-fluid product-img" alt="product image" /
                                    </div>
                                </div>
                                <div class="col-12 col-md-7">
                                    <h4>{{ $data->title }}</h4>
                                    <span class="card-text item-company">By <a href="javascript:void(0)" class="company-name">{{ $data->addBy->name }}</a></span>
                                    <div class="ecommerce-details-price d-flex flex-wrap mt-1">
                                        <h4 class="item-price mr-1">Rp {{ $data->price }}</h4>
```

```
                                    </div>
                                    <p class="card-text">Stok - <span class="text-success">{{ $data->stock }}</span></p>
                                    <hr />
                                    {!! $data->description !!}

                                </div>
                            </div>
                        </div>
                    </div>
                </section>
                <!-- app e-commerce details end -->

            </div>
        </div>
    </div>
</div>


@push('vendor-css')
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/vendors/css/vendors.min.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/vendors/css/forms/select/select2.min.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/vendors/css/editors/quill/katex.min.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/vendors/css/editors/quill/monokai-sublime.min.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/vendors/css/editors/quill/quill.snow.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/vendors/css/pickers/pickadate/pickadate.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/vendors/css/pickers/flatpickr/flatpickr.min.css">
@endpush
    @endpush
    @push('page-css')
    <link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/css/core/menu/menu-types/vertical-menu.css">
    <link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/css/plugins/forms/form-quill-editor.css">
    <link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/css/plugins/forms/pickers/form-flat-pickr.css">
    <link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/css/plugins/forms/pickers/form-pickadate.css">
    @endpush
    @push('custom-scripts')
    <script src="{{ asset('assets') }}/vendors/js/pickers/pickadate/picker.js"></script>
    <script src="{{ asset('assets') }}/vendors/js/pickers/pickadate/picker.date.js"></script>
    <script src="{{ asset('assets') }}/vendors/js/pickers/pickadate/picker.time.js"></script>
    <script src="{{ asset('assets') }}/vendors/js/pickers/pickadate/legacy.js"></script>
    <script src="{{ asset('assets') }}/vendors/js/pickers/flatpickr/flatpickr.min.js"></script>
    <script src="{{ asset('assets') }}/vendors/js/forms/select/select2.full.min.js"></script>
    <script src="{{ asset('assets') }}/vendors/js/editors/quill/katex.min.js"></script>
    <script src="{{ asset('assets') }}/vendors/js/editors/quill/highlight.min.js"></script>
    <script src="{{ asset('assets') }}/vendors/js/editors/quill/quill.min.js"></script>
    @endpush
    @push('page-js')
    <script src="{{ asset('assets') }}/js/scripts/pages/page-blog-edit.js"></script>
    <script src="{{ asset('assets') }}/ckeditorx/ckeditor.js"></script>
    <script src="{{ asset('assets') }}/js/scripts/forms/pickers/form-pickers.js"></script>
    <script type="text/javascript">
        $(document).ready(function () {
            $('.ckeditor').ckeditor();
        });
    </script>
    <script>
        function edValueKeyPress() {
            var edValue = document.getElementById("title");
            var s = edValue.value;

            var lblValue = document.getElementById("slug");
            lblValue.value = s.toLowerCase().replace(/[^\w-]+/g, '-');
        }

    </script>

    @endpush
    </x-master-layouts>
```

edit.blade.php

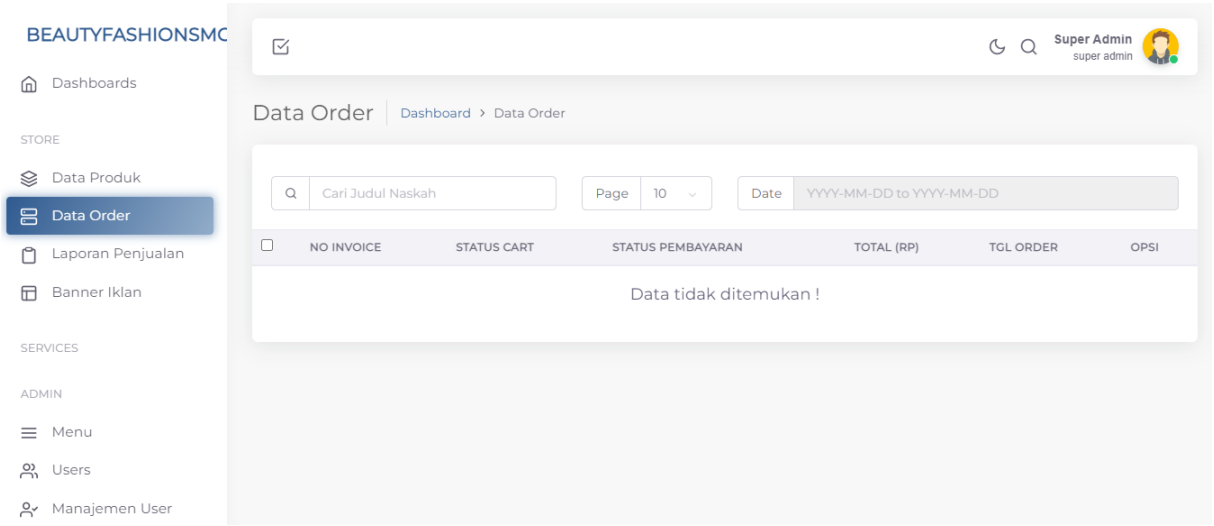
```
@section('title')
    Edit {{ $data->title }} -
@endsection

<x-master-layouts>
@include('sweetalert::alert')

<div class="app-content content ">
    <div class="content-overlay"></div>
    <div class="header-navbar-shadow"></div>
    <div class="content-wrapper">
        <div class="content-header row">
            <div class="content-header-left col-md-9 col-12 mb-2">
                <div class="row breadcrumbs-top">
                    <div class="col-12">
                        <h2 class="content-header-title float-left mb-0">Edit Produk</h2>
                        <div class="breadcrumb-wrapper">
                            <ol class="breadcrumb">
                                <li class="breadcrumb-item"><a href="/dashboard">Home</a>
                                </li>
                                <li class="breadcrumb-item"><a href="{{ route('products.index') }}">Data Produk</a>
                                </li>
                                <li class="breadcrumb-item active">Edit
                                </li>
                            </ol>
                        </div>
                    </div>
                </div>
            </div>
        </div>
        <div class="content-body">
          <!-- Blog Edit -->
          <div class="blog-edit-wrapper">
              <div class="row">
                  <div class="col-12">
                      <div class="card">
                          <div class="card-body">
                              <div class="media">
                                  <div class="avatar mr-75">
                                      <img src="{{ auth()->user()->profile_photo_path ? asset('storage/images/users').'/'.auth()->user()->profile_photo_path : asset('assets/
                                  </div>
                                  <div class="media-body">
                                      <h6 class="mb-25">{{ auth()->user()->name }}</h6>
                                      <p class="card-text">{{ \Carbon\Carbon::now()->format('D, d M Y') }}</p>
                                  </div>
                              </div>

                              <form action="{{ route('products.update', $data->id) }}" method="POST" enctype="multipart/form-data" class="mt-2">
                                  @csrf
                                  @method('PUT')
                                  <div class="row">
                                      <div class="col-md-12 col-12">
                                          <div class="form-group mb-2">
                                              <h5 class="text-primary text-uppercase">Judul</h5>
                                              <input type="text" name="title" id="title" value="{{ $data->title }}" class="form-control" onInput="edValueKeyPress()">
                                          </div>
                                      </div>


                                      <div class="col-md-6 col-12 mb-1">
                                          <h5 class="text-primary text-uppercase" for="blog-edit-category">Category</h5>
                                          <select class="select2 form-control" name="category">
                                              @if ($currentCategory != null)
                                                  <option value="{{ $currentCategory->id }}">{{ strtoupper($currentCategory->name) }}</option>
                                              @endif
                                              @foreach ($categories as $category)
                                                  <option value="{{ $category->id }}">{{ strtoupper($category->name) }}</option>
                                              @endforeach
```

```
69                          </select>
70                      </div>
71
72                      <div class="col-md-6 col-12">
73                          <div class="form-group mb-2">
74                              <h5 class="text-primary text-uppercase">Harga</h5>
75                              <input type="number" name="price" value="{{ $data->price }}" class="form-control">
76                          </div>
77                      </div>
78
79                      <div class="col-md-6 col-12">
80                          <div class="form-group mb-2">
81                              <h5 class="text-primary text-uppercase">Diskon %</h5>
82                              <input type="number" name="discount" id="discount" value="{{ $data->discount }}" max="100" placeholder="1-100" class="form-cont
83                          </div>
84                      </div>
85
86                      <div class="col-md-6 col-12">
87                          <div class="form-group mb-2">
88                              <h5 class="text-primary text-uppercase">Stok</h5>
89                              <input type="number" name="stock" id="stock" value="{{ $data->stock }}" class="form-control">
90                          </div>
91                      </div>
92
93                      {{-- <div class="col-12 mb-2">
94                          <h5 class="text-primary text-uppercase">Post Type</h5>
95                          <div class="demo-inline-spacing">
96                              <div class="custom-control custom-radio">
97                                  <input type="radio" id="customRadio1" name="type" value="0" class="custom-control-input" checked />
98                                  <label class="custom-control-label" for="customRadio1">Standard</label>
99                              </div>
100                             <div class="custom-control custom-control-warning custom-radio">
101                                 <input type="radio" id="customRadio2" name="type" value="1" class="custom-control-input" />
102                                 <label class="custom-control-label" for="customRadio2">Headline</label>
103                             </div>
104                             <div class="custom-control custom-control-success custom-radio">
105                                 <input type="radio" id="customRadio3" name="type" value="2" class="custom-control-input" />
106                                 <label class="custom-control-label" for="customRadio3">Latest Slider</label>
107                             </div>
108                         </div>
109                     </div> --}}
110                     <div class="col-12 mb-2">
111                         <h5 class="text-primary text-uppercase">Deskripsi</h5>
112                         <textarea name="description" class="ckeditor" id="" cols="30" rows="5">{{ $data->description }}</textarea>
113                     </div>
114                     <div class="col-md-6 col-12">
115                         <div class="form-group mb-2">
116                             <h5 class="text-primary text-uppercase">Shopee</h5>
117                             <input type="url" name="shopee_url" value="{{ $data->shopee_url }}" class="form-control" placeholder="https://">
118                         </div>
119                     </div>
120                     <div class="col-md-6 col-12">
121                         <div class="form-group mb-2">
122                             <h5 class="text-primary text-uppercase">Toko Pedia</h5>
123                             <input type="url" name="tokopedia_url" value="{{ $data->tokopedia_url }}" class="form-control" placeholder="https://">
124                         </div>
125                     </div>
126                     <div class="col-md-6 col-12">
127                         <div class="form-group mb-2">
128                             <h5 class="text-primary text-uppercase">Lazada</h5>
129                             <input type="url" name="lazada_url" value="{{ $data->lazada_url }}" class="form-control" placeholder="https://">
130                         </div>
131                     </div>
132                     <div class="col-md-6 col-12">
133                         <div class="form-group mb-2">
134                             <h5 class="text-primary text-uppercase">Buka Lapak</h5>
135                             <input type="url" name="lazada_url" value="{{ $data->lazada_url }}" class="form-control" placeholder="https://">
136                         </div>
137                     </div>
138
139                     <div class="col-md-12 col-12">
140                         <div class="border rounded p-2">
141                             <h5 class="text-primary text-uppercase" class="mb-1">Image</h5>
142                             <div class="media flex-column flex-md-row">
143                                 @if ($data->image != null)
```

```
                                                <img src="{{ asset('/storage/pictures/product/4_3/mid/') }}/{{ $data->image }}" id="blog-feature-image" class="rounded
                                            @else
                                                <img src="{{ asset('assets') }}/images/slider/10.jpg" id="blog-feature-image" class="rounded mr-2 mb-1 mb-md-0" width="
                                            @endif
                                            <div class="media-body">
                                                <div class="d-inline-block">
                                                    <div class="form-group mb-0">
                                                        <input class="w-50" type="file" id="pic-form" name="image" accept="image/*">
                                                        <input type="hidden" name="16_9_width" id="16_9_width"/>
                                                        <input type="hidden" name="16_9_height" id="16_9_height"/>
                                                        <input type="hidden" name="16_9_x" id="16_9_x"/>
                                                        <input type="hidden" name="16_9_y" id="16_9_y"/>

                                                        <input type="hidden" name="4_3_width" id="4_3_width"/>
                                                        <input type="hidden" name="4_3_height" id="4_3_height"/>
                                                        <input type="hidden" name="4_3_x" id="4_3_x"/>
                                                        <input type="hidden" name="4_3_y" id="4_3_y"/>

                                                        <input type="hidden" name="1_1_width" id="1_1_width"/>
                                                        <input type="hidden" name="1_1_height" id="1_1_height"/>
                                                        <input type="hidden" name="1_1_x" id="1_1_x"/>
                                                        <input type="hidden" name="1_1_y" id="1_1_y"/>
                                                    </div>
                                                </div>
                                            </div>
                                        </div>
                                    </div>

                                    <div class="col-md-12 col-12 mt-2">
                                        <div class="form-group">
                                            <h5 class="text-primary text-uppercase" for="blog-edit-status">Status</h5>
                                            <div class="demo-inline-spacing">
                                                <div class="custom-control custom-control-primary custom-radio">
                                                    <input type="radio" id="customRadio1" name="status" value="1" {{ $data->status == 1 ? 'checked' : '' }} class="custom-c
                                                    <label class="custom-control-label" for="customRadio1">Active</label>
                                                </div>
                                                <div class="custom-control custom-control-danger custom-radio">
                                                    <input type="radio" id="customRadio2" name="status" value="0" {{ $data->status == 0 ? 'checked' : '' }} class="custom-c
                                                    <label class="custom-control-label" for="customRadio2">Hidden</label>
                                                </div>
                                            </div>
                                        </div>
                                    </div>

                                    <div class="col-12 mt-5 d-flex justify-content-start">
                                        <button type="submit" class="btn btn-primary mr-1">Save Changes</button>
                                        <a href="{{ route('products.index') }}" class="btn btn-outline-secondary">Cancel</a>
                                    </div>
                                </div>
                            </form>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>

<div class="modal fade" id="modal-default" tabindex="-1" role="dialog" aria-labelledby="modalLabel" aria-hidden="true">
    <div class="modal-dialog modal-lg" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="text-primary text-uppercase" class="modal-title">Crop Photo</h5>
                <button type="button" class="close" id="closeAtas" aria-label="Close">
                    <span aria-hidden="true">&times;</span>
                </button>
            </div>
            <div class="modal-body">
                <div class="row d-flex justify-content-center">
                    <div class="col-12 d-flex justify-content-center">
                        <div class="img-container p-2 text-center">
                            <div id="preview-16-9" class="d-none"></div>
```

```
                            <h4 class="text-primary">Aspect Ratio 16:9</h4>
                            <img id="16-9-show">
                        </div>
                    </div>
                    <div class="col-12 d-flex justify-content-center">
                        <div class="img-container p-2 text-center">
                            <div id="preview-4-3" class="d-none"></div>
                            <h4 class="text-primary">Aspect Ratio 4:3</h4>
                            <img id="4-3-show">
                        </div>
                    </div>
                    <div class="col-12 d-flex justify-content-center">
                        <div class="img-container p-2 text-center">
                            <div id="preview-1-1" class="d-none"></div>
                            <h4 class="text-primary">Aspect Ratio 1:1</h4>
                            <img id="1-1-show">
                        </div>
                    </div>
                </div>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-primary" data-dismiss="modal">Crop</button>
                <button type="button" class="btn btn-secondary" aria-label="Close" id="onClose">Close</button>
            </div>
        </div>
    </div>
</div>

@push('vendor-css')
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/vendors/css/vendors.min.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/vendors/css/forms/select/select2.min.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/vendors/css/pickers/pickadate/pickadate.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/vendors/css/pickers/flatpickr/flatpickr.min.css">
@endpush
<link rel="stylesheet" href="{{ asset('assets') }}/vendors/cropperjs/cropper.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/css/core/menu/menu-types/vertical-menu.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/css/plugins/forms/form-quill-editor.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/css/plugins/forms/pickers/form-flat-pickr.css">
<link rel="stylesheet" type="text/css" href="{{ asset('assets') }}/css/plugins/forms/pickers/form-pickadate.css">
@endpush
@push('custom-scripts')
<script src="{{ asset('assets') }}/vendors/js/pickers/pickadate/picker.js"></script>
<script src="{{ asset('assets') }}/vendors/js/pickers/pickadate/picker.date.js"></script>
<script src="{{ asset('assets') }}/vendors/js/pickers/pickadate/picker.time.js"></script>
<script src="{{ asset('assets') }}/vendors/js/pickers/pickadate/legacy.js"></script>
<script src="{{ asset('assets') }}/vendors/js/pickers/flatpickr/flatpickr.min.js"></script>
<script src="{{ asset('assets') }}/vendors/js/forms/cleave/cleave.min.js"></script>
<script src="{{ asset('assets') }}/vendors/js/forms/cleave/addons/cleave-phone.us.js"></script>
@endpush

@push('page-vendor')
<script src="{{ asset('assets') }}/vendors/js/forms/select/select2.full.min.js"></script>
@endpush

@push('page-js')
<script src="{{asset('backend/plugins/bootstrap-fileinput/js/fileinput.js')}}"></script>
<script src="{{asset('backend/plugins/bootstrap-fileinput/themes/fa/theme.js')}}"></script>
<script src="{{ asset('assets') }}/vendors/cropperjs/cropper.js"></script>
<script src="{{ asset('assets') }}/ckeditorx/ckeditor.js"></script>
<script src="https://cdn.ckeditor.com/4.20.0/basic/ckeditor.js"></script>
<script src="{{ asset('assets') }}/js/scripts/forms/pickers/form-pickers.js"></script>
<script src="{{ asset('assets') }}/js/scripts/forms/form-select2.js"></script>
<script src="{{ asset('assets') }}/js/scripts/forms/form-input-mask.js"></script>

<script type="text/javascript">
    $(document).ready(function () {
        $('.ckeditor').ckeditor();
    });
</script>

<script src="{{ asset('assets') }}/js/scripts/forms/form-crop-image.js"></script>

@endpush
```

Selanjutnya kita akan membahas form data order, pada form ini digunakan untuk menapilkan produk yang telah dibeli oleh pelanggan



Untuk filenya adalah sebagai berikut detail.blade.php, index.blade.php dan invoice.blade.php mari kita bahas kodenya satu persatu.

index.blade.php

```
@section('title')
    Data Order -
@endsection

<x-master-layouts>
    @include('sweetalert::alert')
    <div class="app-content content ">
        <div class="content-overlay"></div>
        <div class="header-navbar-shadow"></div>
        <div class="content-wrapper">
            <div class="content-header row">
                <div class="content-header-left col-md-12 col-12 mb-2">
                    <div class="row breadcrumbs-top">
                        <div class="col-12">
                            <h2 class="content-header-title float-left mb-0">Data Order</h2>
                            <div class="breadcrumb-wrapper">
                                <ol class="breadcrumb">
                                    <li class="breadcrumb-item"><a href="/dashboard">Dashboard</a>
                                    </li>
                                    <li class="breadcrumb-item active">Data Order
                                    </li>
                                </ol>
                            </div>
                        </div>
                    </div>
                </div>
            </div>

            @livewire('store.data-order-table')
        </div>
    </div>
</x-master-layouts>
```

detail.blade.php

```
@section('title')
    Detail Order -
@endsection

<x-master-layouts>
    @include('sweetalert::alert')
    <div class="app-content content ">
        <div class="content-overlay"></div>
        <div class="header-navbar-shadow"></div>
        <div class="content-wrapper">
            <div class="content-header row">
                <div class="content-header-left col-md-12 col-12 mb-2">
                    <div class="row breadcrumbs-top">
                        <div class="col-12">
                            <h2 class="content-header-title float-left mb-0">Data Order</h2>
                            <div class="breadcrumb-wrapper">
                                <ol class="breadcrumb">
                                    <li class="breadcrumb-item"><a href="/dashboard">Dashboard</a>
                                    </li>
                                    <li class="breadcrumb-item">Data Order
                                    </li>
                                    <li class="breadcrumb-item active">Detail
                                    </li>
                                </ol>
                            </div>
                        </div>
                    </div>
                </div>
            </div>

            <div class="content-body">
                <section class="invoice-preview-wrapper">
                    <div class="row invoice-preview">
                        <!-- Invoice -->
                        <div class="col-xl-9 col-md-8 col-12">
                            <div class="card invoice-preview-card">
                                <div class="card invoice-preview-card">
                                    <div class="card-body invoice-padding pb-0">
                                        <!-- Header starts -->
                                        <div class="d-flex justify-content-between flex-md-row flex-column invoice-spacing mt-0">
                                            <div>
                                                <div class="logo-wrapper">
                                                    <img src="{{ asset('frontend') }}/assets/images/logo.jpeg" height="70" alt="" class="logo-light" />
                                                    <h6 class="text-primary invoice-logo">Aksaraglobal</h6>
                                                </div>
                                                {!! $site->address !!}
                                            </div>
                                            <div class="mt-md-0 mt-2">
                                                <h4 class="invoice-title">
                                                    <span class="invoice-number">#{{ $data->no_invoice }}</span>
                                                </h4>
                                                <div class="invoice-date-wrapper">
                                                    <p class="invoice-date-title">Tanggal:</p>
                                                    <p class="invoice-date">{{ \Carbon\Carbon::parse($data->created_at)->format('d/M/Y') }}</p>
                                                </div>
                                            </div>
                                        </div>
                                        <!-- Header ends -->
                                    </div>

                                    <hr class="invoice-spacing" />

                                    <!-- Address and Contact starts -->
                                    <div class="card-body invoice-padding pt-0">
                                        <div class="row invoice-spacing">
                                            <div class="col-xl-8 p-0">
                                                <h6 class="mb-2">Invoice untuk:</h6>
                                                <h6 class="mb-25">{{ $data->user->name }}</h6>
                                                {!! $site->address !!}
                                                <p class="card-text mb-0">{!! $data->user->email !!}</p>
                                            </div>
                                            <div class="col-xl-4 p-0 mt-xl-0 mt-2">
                                                {{-- <h6 class="mb-2">Detail Pembayaran:</h6> --}}
                                                <table>
```

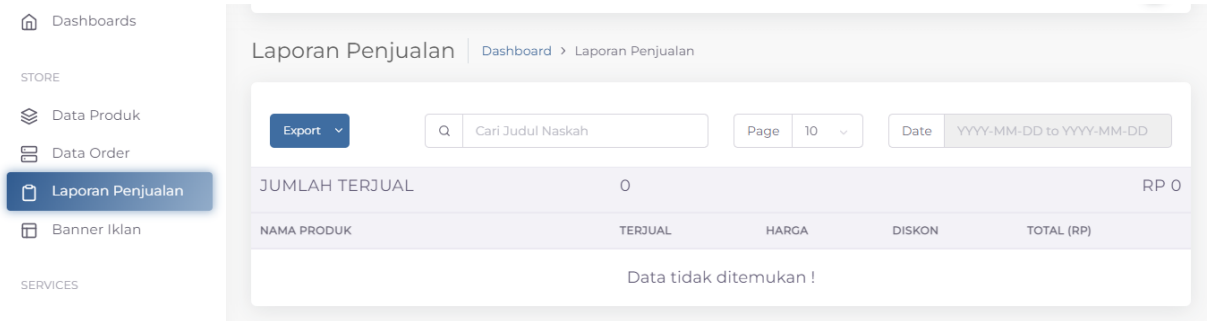
invoice.blade.php

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Invoice {{ $data->no_invoice }}</title>

        <style>
            .invoice-box {
                max-width: 800px;
                margin: auto;
                padding: 30px;
                border: 1px solid #eee;
                box-shadow: 0 0 10px rgba(0, 0, 0, 0.15);
                font-size: 16px;
                line-height: 24px;
                font-family: 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif;
                color: #555;
            }

            .invoice-box table {
                width: 100%;
                line-height: inherit;
                text-align: left;
            }

            .invoice-box table td {
                padding: 5px;
                vertical-align: top;
            }

            .invoice-box table tr td:nth-child(2) {
                text-align: right;
            }

            .invoice-box table tr.top table td {
                padding-bottom: 20px;
            }

            .invoice-box table tr.top table td.title {
                font-size: 45px;
                line-height: 45px;
                color: #333;
            }

            .invoice-box table tr.information table td {
                padding-bottom: 40px;
            }

            .invoice-box table tr.heading td {
                background: #eee;
                border-bottom: 1px solid #ddd;
                font-weight: bold;
            }

            .invoice-box table tr.details td {
                padding-bottom: 20px;
            }

            .invoice-box table tr.item td {
                border-bottom: 1px solid #eee;
            }

            .invoice-box table tr.item.last td {
                border-bottom: none;
            }

            .invoice-box table tr.total td:nth-child(2) {
                border-top: 2px solid #eee;
                font-weight: bold;
            }
```

```
71
72              @media only screen and (max-width: 600px) {
73                  .invoice-box table tr.top table td {
74                      width: 100%;
75                      display: block;
76                      text-align: center;
77                  }
78
79                  .invoice-box table tr.information table td {
80                      width: 100%;
81                      display: block;
82                      text-align: center;
83                  }
84              }
85
86              /** RTL **/
87              .invoice-box.rtl {
88                  direction: rtl;
89                  font-family: Tahoma, 'Helvetica Neue', 'Helvetica', Helvetica, Arial, sans-serif;
90              }
91
92              .invoice-box.rtl table {
93                  text-align: right;
94              }
95
96              .invoice-box.rtl table tr td:nth-child(2) {
97                  text-align: left;
98              }
99          </style>
100     </head>
102     <body>
103         <div class="invoice-box">
104             <table cellpadding="0" cellspacing="0">
105                 <tr class="top">
106                     <td colspan="2">
107                         <table>
108                             <tr>
109                                 <td class="title">
110                                     <img src="{{ asset('frontend') }}/assets/images/logo.jpeg" height="70" alt="" class="logo-light" />
111                                 </td>
112
113                                 <td>
114                                     Invoice {{ $data->no_invoice }}<br />
115                                     Created: {{ \Carbon\Carbon::parse($data->created_at)->format('d/M/Y') }}<br />
116                                 </td>
117                             </tr>
118                         </table>
119                     </td>
120                 </tr>
121
122                 <tr class="information">
123                     <td colspan="2">
124                         <table>
125                             <tr>
126                                 <td style="padding-right:100px;">
127                                     <small>{!! $site->address !!}</small>
128                                 </td>
129
130                                 <td>
131                                     <small>
132                                         {!! $data->user->name !!}John Doe<br />
133                                         {!! $data->user->email !!}
134                                     </small>
135                                 </td>
136                             </tr>
137                         </table>
```

```
                    </td>
                </tr>

                <tr class="heading">
                    <td>Payment Method</td>

                    <td>Transfer #</td>
                </tr>

                <tr class="details">
                    <td>Total</td>

                    <td>Rp {{ number_format($data->total) }}</td>
                </tr>
            </table>
            <table>
                <tr class="heading">
                    <td>Item</td>
                    <td>Harga</td>
                    <td>Qty</td>
                    <td>Subtotal</td>
                </tr>
                @foreach ($data->detail as $item)
                    <tr class="item">
                        <td>{{ $item->produk->title }}</td>
                        <td>Rp {{ number_format($item->harga) }}</td>
                        <td> x {{ $item->qty }}</td>
                        <td>Rp {{ number_format($item->subtotal) }}</td>
                    </tr>
                @endforeach


                <tr class="total">
                    <td colspan="2" style="text-align: right"><b>Total: </b></td>

                    <td colspan="2"> Rp {{ number_format($data->total) }}</td>
                </tr>
            </table>
        </div>
    </body>
</html>
```

Selanjutnya kita akan membahas Form laporan penjualan, form ini digunakan untuk mengetahui berapa produk yang telah terjual



struktur file **beauty-fashion-shop/resources/views/admin/store/report/index.blade.php**

index.blade.php

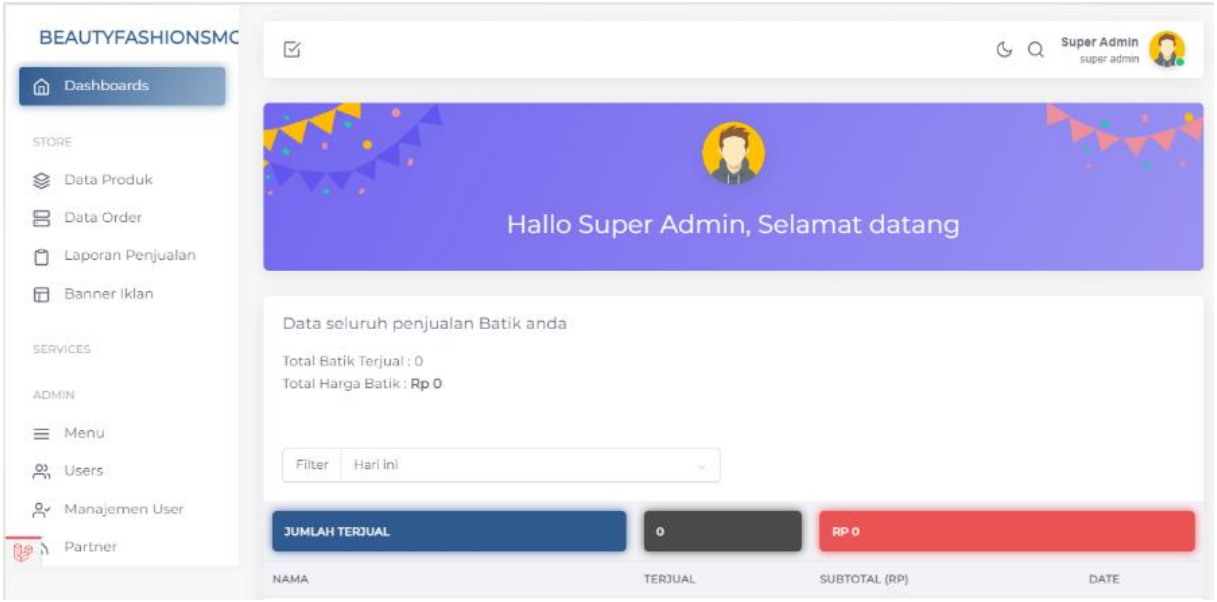
```
@section('title')
    Laporan Penjualan -
@endsection

<x-master-layouts>
    @include('sweetalert::alert')
    <div class="app-content content ">
        <div class="content-overlay"></div>
        <div class="header-navbar-shadow"></div>
        <div class="content-wrapper">
            <div class="content-header row">
                <div class="content-header-left col-md-12 col-12 mb-2">
                    <div class="row breadcrumbs-top">
                        <div class="col-12">
                            <h2 class="content-header-title float-left mb-0">Laporan Penjualan</h2>
                            <div class="breadcrumb-wrapper">
                                <ol class="breadcrumb">
                                    <li class="breadcrumb-item"><a href="/dashboard">Dashboard</a>
                                    </li>
                                    <li class="breadcrumb-item active">Laporan Penjualan
                                    </li>
                                </ol>
                            </div>
                        </div>
                    </div>
                </div>
            </div>

            @livewire('store.report')
        </div>
    </div>
</x-master-layouts>
```

export-report.blade.php

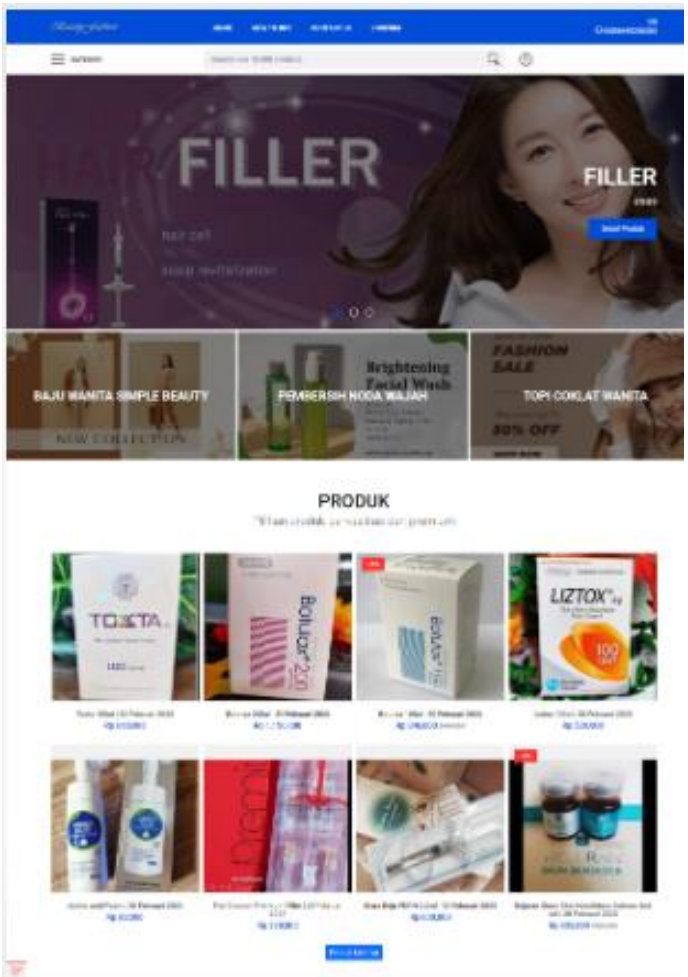
```
<div class="table-responsive">
    <table class="table">
        <thead>
            <tr>
                <th><h5>Jumlah Terjual</h5></th>
                <th><h5>{{ $data->sum('qty') }}</h5></th>
                <th colspan="3" class="text-right"><h5>Rp {{ number_format($data->sum('total_sale')) }}</h5></th>
            </tr>
            <tr>
                <th>Nama Produk</th>
                <th>Terjual</th>
                <th>Harga</th>
                <th>Diskon</th>
                <th>Total (RP)</th>
            </tr>
        </thead>
        <tbody>
            @forelse ($data as $row)
                <tr>
                    <td>{{ $row->produk->title }}</td>
                    <td>{{ $row->qty }}</td>
                    <td>
                        {{ number_format($row->produk->price) }}
                    </td>
                    <td>
                        @if ($row->produk->discount > 0)
                            {{ $row->produk->discount }}%
                            <s>{{ number_format($row->produk->price * $row->qty) }}</s>
                        @else
                            -
                        @endif
                    </td>
                    <td>{{ number_format($row->total_sale) }}</td>
                </tr>
            @empty
            <tr>
                <td colspan="9" class="pt-2 pb-1 text-center"><h5>Data tidak ditemukan !</h5></td>
            </tr>
            @endforelse

        </tbody>
    </table>
</div>
```

Hasil Pembuatan toko onine

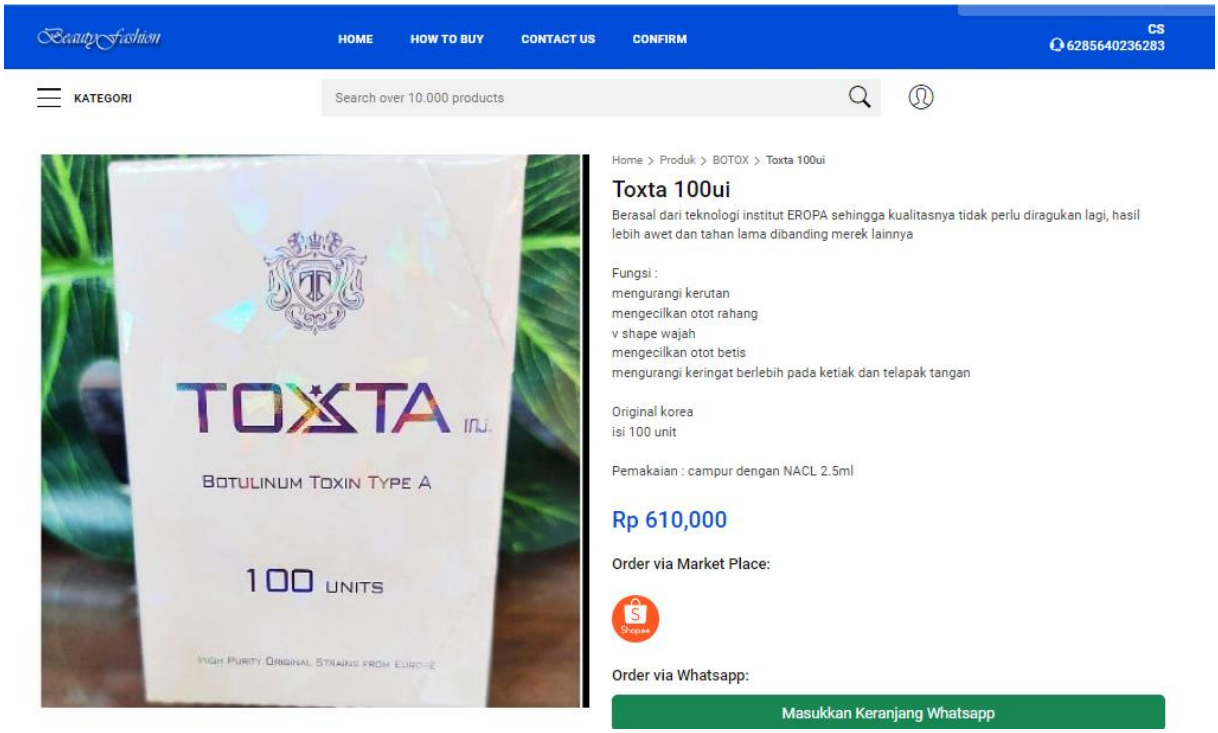
Halaman admin :



Halaman Pengunjung :



Halaman detail Produk



Untuk hasil sialahkan diakses pada link : https://beautyfashionsmg.com/

Daftar Pustaka

Laravel Manual: https://laravel.com/docs/9.x

Laravel-best-practices: https://github.com/alexeymezenin/laravel-best-practices

Traversy Media: https://www.youtube.com/user/TechGuyWeb

Bitfumes: https://www.youtube.com/bitfumes

Coder's Tape: https://coderstape.com

Laracast Forum: https://laracasts.com/discuss

Stackoverflow Laravel: https://stackoverflow.com/questions/tagged/laravel

YAYASAN PRIMA AGUS TEKNIK

ISBN 978-623-8120-43-7 (PDF)