



Hafid Mukhlasin
Muhammad Azamuddin

2.6

FRONTEND WEB SERIES

Vue JS

The progressive Javascript framework

KUNGFU KODING

Lisensi & Informasi Versi

Lisensi

Ebook ini hanya boleh digunakan oleh pemilik email yang tertera di header buku ini. Penggunaan buku oleh selain pemilik email tersebut merupakan tindakan yang tidak diperbolehkan.

Siapapun (termasuk pemilik buku) tidak memiliki hak untuk menyalin dan atau menyebarkan buku ini tanpa seizin penulis.

Versi ebook

versi	tanggal	author	keterangan
1.0.0	10 Oktober 2018	Hafid Mukhlasin	Rilis pertama
1.0.1	10 November 2018	Hafid Mukhlasin	Rilis kedua
1.0.2	18 November 2018	Hafid Mukhlasin	Rilis ketiga
2.0.0	4 Agustus 2019	Hafid Mukhlasin	Rilis keempat

Persembahan

Bismillahirrahmanirrahim. Ucapan tanpa batas untuk Yang Maha Kuasa, Allah SWT atas setiap nafasku dan keberkahanNya. Shalawat serta salam bagi junjunganku, Nabi Muhammad SAW atas teladannya.

Terima kasih kepada Bapak dan Ibu penulis, atas cinta dan doa tulus yang tak pernah putus. Terima kasih juga kepada penulis sampaikan kepada istri tercinta, Hari Dwipanjayani, yang telah sabar menemani penulis dalam menghabiskan sisa umur ini. Tentu saja kepada keempat anak-anak penulis yang telah menjadi penyejuk hati, Ammar, Faqih, Syamil, dan Hilyah.

Buku ini juga saya persembahkan kepada mereka yang telah menginspirasi penulis yaitu Irfan Maulana, Peter Jack Kambey, Mulia Nasution, Yohan Totting, Adib Firman serta seluruh komunitas Vue JS Indonesia, Laravel Indonesia, Yii Framework Indonesia dan WWWID yang tidak bisa penulis sebutkan satu persatu.

Dan juga tentu saja buku ini saya persembahkan tidak lain tidak bukan untuk Anda para pembaca dan komunitas TI Indonesia, semoga ilmu yang sedikit ini dapat bermanfaat bagi kemajuan ilmu pengetahuan dan teknologi di Indonesia. Amiin.

Kata Pengantar

Puji syukur kehadirat Allah Subhanahu Wata'ala atas limpahan nikmatnya sehingga buku panduan belajar Vue JS ini dapat diselesaikan dengan baik. Buku ini merupakan bagian dari paket buku "Be Fullstack Developer" yang ditulis bersama dengan rekan penulis yaitu Muhammad Azamuddin.

Tidak lupa penulis ucapan terima kasih kepada berbagai pihak baik yang telah membantu kami secara langsung atau tidak langsung dalam proses penyusunan buku ini, diantaranya om Peter Jack Kambey (PHP Indonesia), om Irfan Maulana (Vue JS Indonesia) dan om Fachruzi Ramadhan (Laravel Indonesia).

Buku ini membahas Vue JS mulai dari dasar hingga membuat aplikasi berbasis Vue dan interaksinya dengan backend (Laravel web service). Untuk memudahkan dalam memahami materi dalam buku ini maka penulis juga melengkapinya dengan studi kasus yaitu membuat toko online berbasis mobile web.

Akhirnya penulis menyadari bahwa dalam penyusunan buku ini tak lepas dari kekurangan disana sini, oleh karenanya penulis memohon kritik saran dan masukan demi perbaikan pada edisi berikutnya.

Jakarta 3 Oktober 2018

Hafid Mukhlisin

Daftar Isi

Lisensi & Informasi Versi	2
Lisensi	2
Versi ebook	2
Persembahan	3
Kata Pengantar	4
Mengenal Vue	12
Intro	12
Apa itu Vue?	12
Sejarah Vue	12
Mengapa Memilih Vue?	13
Framework Javascript Populer	13
Didukung Banyak Pustaka	14
Bukan One Man Show	14
Digunakan Perusahaan Besar	14
Mudah Dipelajari	14
Mudah Diintegrasikan dengan Pustaka Lain	15
Dukungan Official untuk Pengembangan Aplikasi Enterprise	15
Fitur Utama	15
Virtual DOM	15
Component Base	16
Template	16
Modularity	16
Reactivity	16
Routing	16
State Management	16
Development Tools	16
Instalasi & Konfigurasi	17
Hello World	18
Menguji Reaktifitas	21
Kesimpulan	22
Dasar-Dasar Vue	23
Intro	23
Objek Vue	23
Inisiasi Objek Vue	23
Properti el	24
Properti Data	24
Siklus Objek Vue	25
create	27
mount	30
update	31
destroy	33
Penulisan Template	34
Data Teks	34
Data Raw HTML	34
Data Attribute	35
JS Expression	36
Properti Template	36
Properti Methods, Computed, & Filters	38
Properti Methods	38
Properti Computed	39

Properti Filters	40
Argumen Pada Filters	42
Chaining Filters	43
Deklarasi Filters Secara Terpisah	44
Kesimpulan	44
Directive	45
Intro	45
Mengenal Directive	45
v-html	45
v-once	45
v-text	45
v-show	45
v-if	46
v-on	47
Modifier .exact	53
Modifier Mouse Button	53
v-bind	54
Dynamic Argument	55
Kesimpulan	56
List	57
Intro	57
Menampilkan Data Array	57
v-for Menggunakan Tag Template	58
v-for Menggunakan Index	58
Menampilkan Data Objek	59
Menampilkan Data Collection	60
Atribut Key	63
Membatasi v-for menggunakan v-if	63
Perubahan (mutation) Data Pada Array	64
push() & pop()	65
unshift() & shift()	65
sort() & reverse()	66
splice()	66
fungsi set pada Vue	69
Perubahan Data Pada Objek	70
Kesimpulan	71
Form	73
Intro	73
Input Binding	73
Text	75
Boolean	77
Array	77
Filtering Data List	79
Handling Submit Form & Validation	80
Validasi Data	83
Prepare Data Submit	87
Send Data To Server	88
Handling File Upload	91
Kesimpulan	93
Component	94
Intro	94
Component Dasar	94
Penamaan Component	95
Component Registration	95

Global Component	95
Local Component	96
Deklarasi Properti Data	98
Reusable Component	99
Component Lanjutan	99
Kirim Data ke Component	99
Directive Pada Component	101
Update Data Parent From Component	104
Two Way Data Binding on Component	106
Content Distribution with Slots	107
Fallback Slots	108
Penamaan Slot	109
Scoped Slot	110
Dynamic Slot Names	111
Named Slots Shorthand	111
Single File Component	111
Dynamic Components	114
Transition Effect	116
Mixins	117
Plugins	118
Deklarasi Plugins	119
Menggunakan Plugin	119
Kesimpulan	119
Routing	121
Intro	121
Features	121
Installation	121
Getting Started	121
Dynamic Routing	123
Membuat BooksComponent	124
Membuat BookComponent	126
Programmatic Navigation	128
Penamaan Routing	130
Mengirimkan Props ke Component Routing	130
Transitions Effect	131
Navigation Guards	132
Global	132
Per Route	132
Dalam Component	133
Prevent Leave Accident	133
Authentication Route	134
Kesimpulan	135
Intro	136
Mengenal State Management	136
Kapan Menggunakan State Management?	137
Pustaka State Management	137
Arsitektur Vuex	138
Instalasi Vuex	138
Instalasi Dev Tools	139
Getting Started	141
Mengakses Store Via Component	145
Getters	148
Mutations	149
Actions	150

Asynchronous Actions	152
Menangani Two Way Data Binding	156
Kesimpulan	159
Scaffolding Application	161
Intro	161
Briefing Projek	161
Fitur Utama Aplikasi	161
Unified Model Language	161
Use Case Diagram	162
Activity Diagram	163
Class Diagram	163
Desain Database	164
Preparing Project	165
Command Line Tools	165
Package Manager for JS	165
Instalasi NodeJS & NPM	166
Instalasi Vue melalui NPM	168
Apa itu Bundler	170
Instalasi Vue menggunakan JS Bundler	170
Browserify	170
Webpack	173
Single File Component	177
Web Server Development	180
Hot Reload	181
Vue Command Line Interface (CLI)	183
Create New Project	184
Create New Project on Web Base	190
Menambahkan Plugin Baru	200
Kesimpulan	211
Web Service	212
Intro	212
Mengenal Web Service	212
Definisi Web Service	212
Standard Web Service	212
Method Web Service	212
Cara Kerja Web Service	212
HTTP Response Code	213
Stateles pada Web Service	213
Persiapan Tools Pengembangan	214
Bahasa Pemrograman: PHP	214
Database Server: MySQL, MariaDB	214
Web Server: Nginx, Apache	214
Git	214
Package Tools	215
Docker	215
XAMPP	224
Homestead	226
Composer	226
Instalasi Composer	226
Postman	227
Penggunaan	227
Generate Dokumentasi	229
Laravel	232
Mengenal Laravel	232

Instalasi	233
Konfigurasi	235
Variabel Konfigurasi	235
Virtual Domain & Pretty URL	235
Struktur Direktori Aplikasi	237
Routing	238
Routing Web	239
Routing API (Web Service)	240
HTTP Verbs Method	241
Routing Parameter	243
Routing Name	244
Routing Group	244
Controller	247
Middleware	248
Rate Limiting	248
CORS	249
Multiple Middleware	253
Database	253
Konfigurasi	254
Migration	255
Seeding	265
Interact with Database	275
API Resources	286
Handling Error	293
Authentication	295
Konfigurasi	295
Get Authenticate User	296
Check Authenticate User	296
Protect Routing	296
Authentication Mechanism	296
Kesimpulan	305
Finishing Project	307
Intro	307
Konstanta Global	307
Layout Aplikasi	308
Layout Header	311
Layout Main Content	312
Layout Footer	314
Layout Sidebar	314
Membuat Halaman Home	318
Layout Halaman Home	318
Endpoint Random Category	322
Endpoint Top Book	324
Top Books	328
Membuat Halaman Kategori Buku	334
Endpoint Categories	334
Template & Script	336
Mendaftarkan Route Categories	338
Membuat Header Pada Parent vs Child	340
Membuat Halaman Buku	341
Endpoint Book	341
Template & Script	343
Mendaftarkan Route Books	345
Membuat Halaman Detail Kategori	346

Endpoint Detail Category	347
Template & Script	350
Mendaftarkan Route Detail Category	352
Membuat Halaman Detail Buku	353
Endpoint Detail Book	353
Template & Script	356
Mendaftarkan Route Book	358
Membuat Reusable Component	359
Membuat Component BookItem	361
Menggunakan Component BookItem	362
Implementasi State Management	363
Mendefinisikan State	364
Mendefinisikan Mutation & Action Pada State	365
Memanggil Action State Pada Component	366
Memecah State Menjadi Module Tersendiri	370
Membuat Component Alert	373
Component Snackbar	373
Module State Alert	374
Component Alert	375
Menampilkan Alert	376
Membuat Indikator Keranjang Belanja	377
Menambahkah Getters Count	378
Menggunakan Getters Count Pada Header	378
Membuat Halaman Pencarian Buku	379
Membuat Endpoint Search	380
Membuat Component Search	382
Menerapkan Component Search	384
Trigger Halaman Pencarian	384
Membuat Fitur Login	385
Endpoint Login	385
State Login / Auth	386
Memetakan State User Pada Component	387
Membuat Dynamic Component	388
Membuat Component Login	391
Menampilkan Data User Login	395
Membuat Fitur Logout	397
Endpoint Logout	397
Membuat Link Logout	398
Halaman Register	399
Endpoint Register	399
Membuat Component Register	401
Menampilkan Halaman Register	404
Halaman Keranjang Belanja	405
Link Keranjang Belanja	405
Mengupdate State Cart	405
Membuat Component Keranjang Belanja	407
Halaman Checkout	410
Endpoint Province & City	411
Endpoint Update Alamat Pengiriman	414
Membuat Halaman Checkout Part 1	415
Routing Checkout	418
Menampilkan Cart Pada Component Checkout	422
Endpoint Couriers	423
Endpoint Courier Services	424

Menampilkan Form Courier Pada Component Checkout	431
Endpoint Payment	434
Update Tombol Pay Pada Component Checkout	440
Halaman Pembayaran	443
Integrasi Payment Gateway (Experimental)	447
Apa itu Payment Gateway?	447
Persiapan Integrasi	447
Registrasi Midtrans	451
Integrasi Midtrans	454
Membuat Halaman Profile	457
Link Profile di SideBar	457
Layout Halaman Profile	459
Membuat Halaman Histori Belanja	460
Endpoint My Order	460
Link My Order di SideBar	461
Layout Halaman My Order	462
Source Code	465
Kesimpulan	465
Deployment	466
Intro	466
Diskon Hosting & VPS	466
Persiapan	466
Persiapan Aplikasi Web Frontend	466
Persiapan Aplikasi Web Service	467
Konfigurasi	467
Matikan Mode Debug	467
Proses Deployment	469
Deployment Aplikasi Web Frontend	469
Deployment Aplikasi Web Service	476
Membuat Sub Domain	476
Membuat & Mengimport Database	478
Mengunggah File Aplikasi	483
Kesimpulan	491

Mengenal Vue

Intro

Pada bab ini, kamu akan diajak mengenal Vue, mengapa memilih Vue, tools apa saja yang diperlukan dan bagaimana cara menggunakannya secara mudah dan sederhana.

Apa itu Vue?

Vue (dibaca: view) merupakan salah satu dari sekian banyak pustaka (library) pada bahasa pemrograman Javascript yang digunakan untuk membangun tampilan antarmuka pengguna (user interface) dari suatu aplikasi berbasis web khususnya untuk aplikasi berbasis halaman tunggal atau *single page application* (SPA).

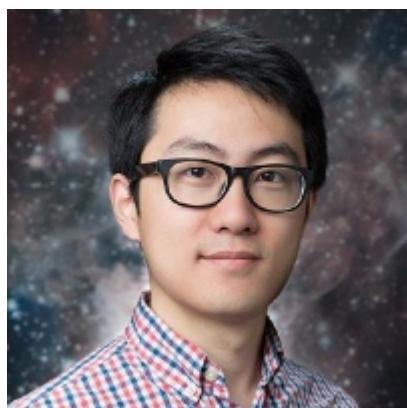


Vue sebagaimana Javascript (JS) memang awalnya didesain untuk kebutuhan web, namun seiring perkembangan teknologi yang mendukung JS, maka saat ini Vue juga mulai dapat digunakan untuk mengembangkan aplikasi berbasis desktop dan mobile.

Situs resmi Vue bisa kita jumpai pada alamat <http://vuejs.org>, adapun link githubnya pada alamat <https://github.com/vuejs>.

Sejarah Vue

Awalnya, Vue merupakan proyek pribadi Evan You (<http://evanyou.me>) ketika masih bekerja di Google Creative Labs pada tahun 2013. Di sana, Evan terlibat dalam pembuatan berbagai prototipe tampilan antarmuka pengguna menggunakan pustaka AngularJs (versi 1).



Hal inilah yang kemudian menginspirasi Evan untuk membuat suatu pustaka sendiri dengan gaya Angular namun menggunakan pendekatan API (Application Programming Interface) yang lebih sederhana.

Vue mengusung konsep web component dan virtual DOM sebagaimana React namun dengan pendekatan yang lebih natural, siapapun yang telah mengenal dasar HTML, Javascript & CSS akan dengan mudah dan cepat dalam menguasai serta mengadopsi Vue.

Pada Februari 2014, Vue pertama kali dipublikasikan dan langsung mendapatkan sambutan yang luar biasa pada minggu pertamanya, hingga membuat Evan terpacu untuk lebih serius lagi dalam mengembangkannya. Oktober 2015, Vue versi 1.0 dipublikasikan yang menandakan Vue siap digunakan untuk production. Diawal tahun 2016, Evan mulai bekerja penuh waktu untuk mengelola Vue berkat banyaknya dukungan atau sponsor yang ia dapat melalui situs Patreon (<https://www.patreon.com/evanyou>).

Salah satu sponsor utama yang sekaligus meningkatkan branding dari Vue adalah Taylor Otwell (<http://taylorotwell.com>), founder Laravel PHP Framework (<https://laravel.io>) dimana kemudian menjadikan Vue sebagai pustaka Javascript untuk Laravel.

Saat buku ini diupdate, Vue telah mencapai versi 2 dengan berbagai perbaikan dan penambahan fitur baru. Versi ini menggunakan engine berbeda untuk menangani Virtual DOM namun tetap ringan dan cepat, kita bisa menggunakan template HTML atau JSX sebagaimana yang umum dipakai di React. Manajemen state didukung secara official melalui Vuex, dan secara natural telah mendukung server side rendering.

Meskipun demikian, secara umum Vue masih tetap menjaga kompatibilitas dengan versi sebelumnya.

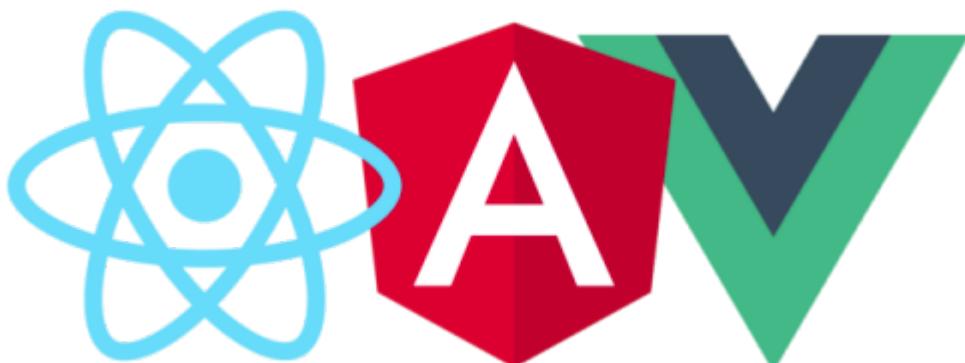
Mengapa Memilih Vue?

Jika dilihat dari karakteristik penggunanya, React menarik bagi mereka yang menyukai functional programming. Angular menarik bagi developer yang terbiasa bermain di bahasa pemrograman Java atau C#. Sedangkan Vue menarik bagi mereka yang menyukai classic HTML, CSS & JavaScript. Hal inilah yang menjadikan alasan mengapa Vue banyak mengambil hati para web developer (termasuk penulis 😊).

Berikut ini, penulis akan mencoba merangkum tentang beberapa alasan mengapa banyak developer memilih Vue.

Framework Javascript Populer

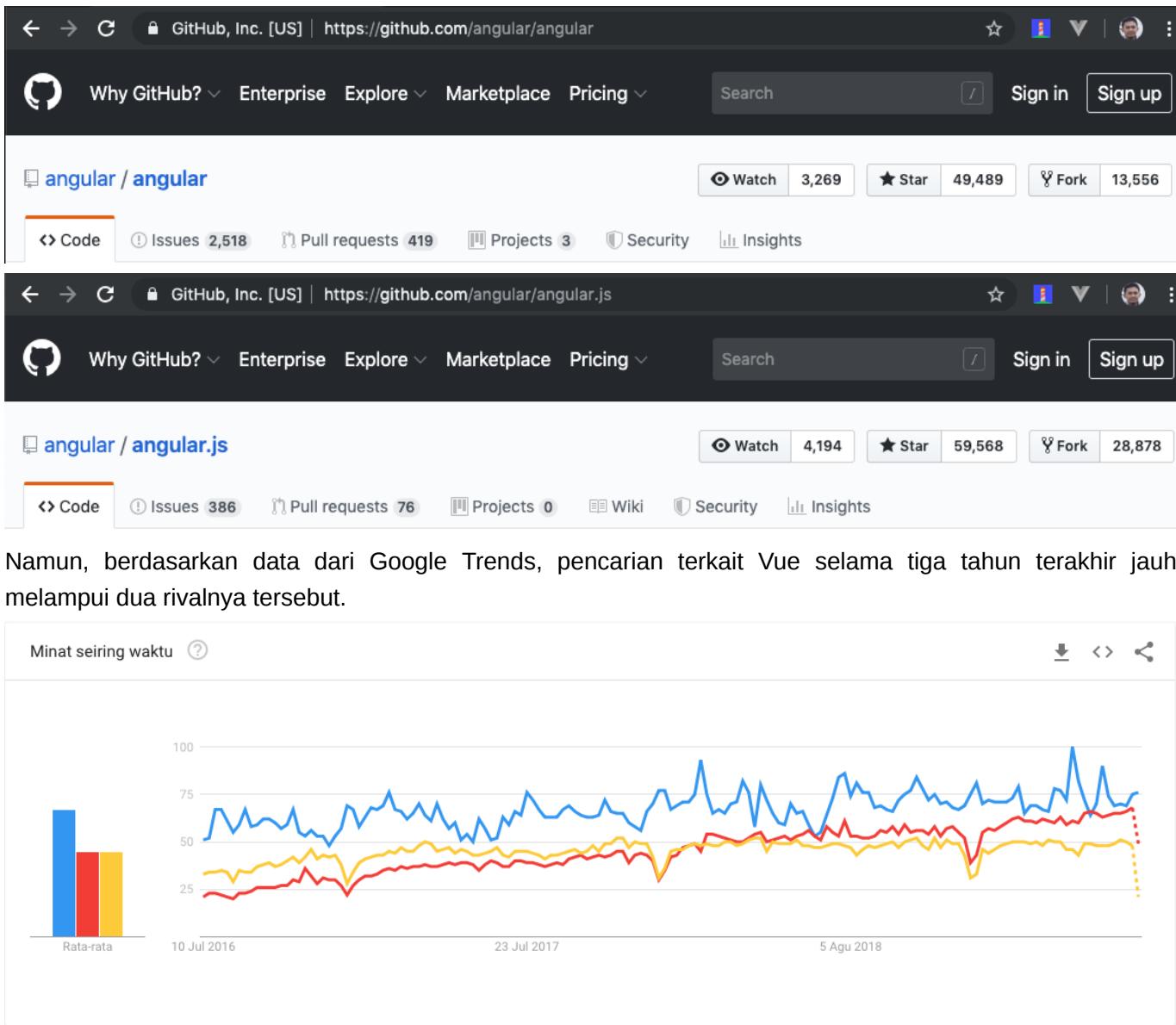
Tidak diragukan lagi, Vue merupakan framework Javascript modern yang cukup populer disamping React & Angular.



Berdasarkan data dari Github (Juli 2019), jumlah star (bintang) di akun Githubnya (<https://github.com/vuejs/vue>) mencapai lebih dari 140 ribu user dan di-fork oleh sekitar 20 ribu user, meski jumlah ini sebelas duabelas dengan perolehan React dan jauh di atas Angular (js & io)

A screenshot of a GitHub repository page. The URL in the address bar is https://github.com/vuejs/vue. The page shows the repository name "vuejs / vue". It has 143,253 stars and 20,723 forks. The repository has 220 issues and 94 pull requests. There are tabs for Code, Issues, Pull requests, Projects, Wiki, Security, and Insights.

A screenshot of a GitHub repository page. The URL in the address bar is https://github.com/facebook/react. The page shows the repository name "facebook / react". It has 132,333 stars and 24,516 forks. The repository has 6,650 issues and 132,333 pull requests. There are tabs for Code, Issues, Pull requests, Projects, Wiki, Security, and Insights.



Sumber: <https://trends.google.com/trends/explore?date=2016-07-10%202019-07-10&q=vue,react,angular>
 Tentu saja hal ini hanyalah salah satu parameter atau tolok ukur dari kepopuleran suatu pustaka atau framework.

Didukung Banyak Pustaka

Salah satu kelebihan dari Vue ini adalah didukung oleh banyak pustaka, sehingga cukup memudahkan bagi developer untuk bekerja dengan Vue. Berbagai pustaka yang mendukung dan menggunakan Vue bisa kita jumpai pada tautan ini <https://github.com/vuejs/awesome-vue>.

Bukan One Man Show

Saat ini Vue sudah mencapai versi 2, bukan lagi proyek pribadi, sebab core developer-nya sudah terdiri dari belasan orang (<https://vuejs.org/v2/guide/team.html>), belum lagi kontributornya di Github yang cukup banyak.

Digunakan Perusahaan Besar

Tidak hanya digunakan oleh perorangan, beberapa perusahaan atau web besar juga telah menggunakan Vue diantaranya: Adobe, Alibaba, Xiaomi, Line, Nintendo, Gitlab, Laravel dsb, selengkapnya di <https://madewithvuejs.com>.

Mudah Dipelajari

Pendekatan yang ditawarkan Vue cukup sederhana dan tidak banyak memperkenalkan konsep baru, sehingga siapapun dengan latar belakang pengetahuan web dasar (HTML, CSS, Javascript) akan mudah

menggunakan dan mengadopsi Vue.

Yap, jika kamu masih baru dalam dunia web programming, maka memang butuh usaha lebih karena tidak akan dibahas secara detail pada buku ini. Penulis berasumsi bahwa kamu sudah memiliki pengetahuan tentang itu, dan jika belum maka gunakan referensi lain terkait web dasar.

Mudah Diintegrasikan dengan Pustaka Lain

Jika kamu sudah menggunakan pustaka lain pada aplikasi saat ini maka kamu tidak perlu khawatir untuk mengintegrasikannya dengan Vue. Apakah kamu tetap ingin menggunakan JQuery misalnya, maka itu tidak menjadi masalah berarti.

Dukungan Official untuk Pengembangan Aplikasi Enterprise

Berbeda dengan React, Vue mendukung dan mengembangkan sendiri secara resmi pustaka-pustaka yang digunakan untuk membangun aplikasi skala besar, seperti routing (Vue Router), state management (Vuex), server side rendering, dsb. Namun hal ini tidak membuat kita sulit untuk menggunakan pustaka lain yang mungkin biasa kita gunakan, seperti Redux, Mobx, dsb.

Fitur Utama

Berikut ini beberapa fitur utama yang dimiliki Vue.

Virtual DOM

DOM singkatan dari Document Object Model merupakan model yang menggambarkan halaman HTML atau XML. DOM berbentuk struktur hirarki pohon yang menghubungkan masing-masing elemen HTML/XML (node). Contoh.

```

1 <html>
2 <head>
3   <title>Contoh</title>
4 </head>
5 <body>
6   <h1> Halo </h1>
7   <p> Test </p>
8 </body>
9 </html>
```

Kode HTML di atas jika dilihat dari sudut pandang DOM memiliki root node `html`, node `html` memiliki dua child node yaitu `head` dan `body`. Node `head` memiliki satu child yaitu `title`, sedangkan node `body` memiliki dua child yaitu `h1` dan `p`.

```

1      html
2      |
3      |      |
4      head     body
5      |      |
6      title   |      |
7          h1     p
```

Javascript memiliki kemampuan untuk mengakses dan memanipulasi semua DOM tersebut secara langsung.

```

1 const h1s = Array.from(document.getElementsByTagName('h1'))
2
3 console.log(h1s[0]); // <h1> Halo </h1>
```

Namun alih-alih memanipulasinya secara langsung, Vue memilih pendekatan lain yaitu membuat abstraksi objek virtual dari DOM kemudian memanipulasinya baru kemudian merender atau menampilkan hasilnya. Pendekatan ini lebih efektif dan cepat dibandingkan langsung memanipulasi DOM-nya sebagaimana yang dilakukan pustaka lain semisal JQuery.

Component Base

Vue menggunakan pendekatan berbasis komponen, dimana setiap tampilan atau bagian dari tampilan merupakan komponen. Melalui pendekatan ini, tampilan yang kompleks dapat dipecah menjadi beberapa bagian dan setiap bagian itu bisa digunakan kembali pada bagian lainnya. Hal ini akan membuat kode lebih efisien dan bersih. Kode komponen pada Vue ditulis menggunakan kode Javascript sebagai sebuah object.

Template

Berkaitan dengan poin sebelumnya, template merupakan kode yang dijadikan dasar dari suatu komponen dan biasanya berupa kode-kode HTML biasa. Penulisan template pada Vue bisa sangat fleksibel dan out of the box. Kita bisa tulisahkan suatu template menjadi satu dengan kode komponennya seperti React, atau dipisahkan menggunakan tag template tag HTML yang id-nya telah didaftarkan, bisa juga dipisahkan pada file tersendiri yang umumnya menggunakan ekstensi Vue, dsb.

Modularity

Komponen pada Vue bisa dipecah menjadi modul-modul kecil. Hal ini akan memudahkan developer dalam pengembangan atau pengelolaan kodennya terutama pada proyek aplikasi skala besar.

Reactivity

Secara default, Vue mendukung reactivity yaitu perubahan data pada suatu bagian tertentu akan secara interaktif mempengaruhi bagian yang lain. Fitur ini akan memudahkan developer dalam mengembangkan aplikasi karena cukup dengan fokus pada flow data dan template.

Routing

Routing merupakan kebutuhan untuk pembuatan aplikasi enterprise karena menyangkut bagaimana suatu halaman pada aplikasi tersebut diakses oleh pengguna melalui web browser. Meski bukan pada core-nya, namun Vue menyediakan pustaka yang didukung secara resmi untuk menangani routing aplikasi, yaitu Vue router <https://router.vuejs.org>.

State Management

Oleh karena vue berbasis komponen, maka diperlukan pendekatan terpusat untuk menyimpan state atau data aplikasi yang bisa dibaca dan dimodifikasi oleh semua komponen yang membutuhkannya. State management juga bukan core pada Vue seperti halnya routing, namun pustaka yang menangani state ini juga didukung secara resmi yaitu vuex <https://vuex.vuejs.org>.

Development Tools

Sebagai sebuah pustaka Javascript biasa, untuk mengembangkan aplikasi berbasis Vue sebenarnya developer hanya membutuhkan code editor untuk menulis kode programnya, serta web browser untuk menampilkan hasilnya.

Tidak ada pilihan spesifik, silahkan gunakan code editor favoritmu, misalnya: Visual Studio Code (penulis menggunakan ini), Sublime, Netbeans, Notepad++, Intelij Idea, dsb.

Adapun untuk web browser pun juga bebas, bisa Google Chrome (penulis menggunakan ini), Mozilla Firefox, Safari, bahkan Microsoft IE (versi 9 atau later) 😊.

Android	Firefox	Chrome	IE	iPhone	Edge	Safari
6.0	58	65	9	10	16	8
			7	10.12	10	10.10
			8	11	10	8.1

Ya, untuk fase awal ini, dua tools ini dulu yang harus kamu siapkan dan penulis yakin semua itu sudah tersedia di komputermu. Sebenarnya banyak tools lain yang perlu juga digunakan namun secara bertahap saja ya 😊, sebab penulis tidak ingin kamu pusing di awal, khawatir kalah sebelum berperang.

Penulis juga ingin menunjukkan kepadamu tentang seberapa sederhananya Vue, awalnya sih 😊. Bagaimana? sepakat?

Instalasi & Konfigurasi

Sebagai sebuah pustaka JS, maka kita perlu menambahkannya ke dalam halaman HTML kita sebelum kita menggunakananya. Saat buku ini ditulis, versi terbaru Vue adalah 2.6.10 (Juli 2019). Untuk melihat versi terbaru dan sebelumnya, silahkan kunjungi tautan berikut <https://github.com/vuejs/vue/releases>.

Pustaka Vue terbagi menjadi dua, yaitu mode development (filenya tidak dikompres) dan mode production (file dikompres). Sangat disarankan menggunakan mode development saat mengembangkan aplikasi menggunakan Vue sebab semua informasi umum (warning) jika terjadi kesalahan kode akan dimunculkan.

File Vue yang akan kita tambahkan ke dalam halaman HTML bisa kita unduh ke lokal (sehingga tidak membutuhkan koneksi internet lagi) atau ditautkan langsung dengan server pustaka Vue (CDN). Kita bisa mengunduh pustaka Vue versi development pada tautan berikut <https://vuejs.org/js/vue.js>, Adapun versi production bisa kita jumpai pada tautan ini <https://vuejs.org/js/vue.min.js>.

Sebagaimana umumnya pustaka javascript, untuk menambahkan ke halaman HTML kita maka cukup dengan kode berikut.

```
1 | <script src="lib/vue.js"></script>
```

Jika kita memilih menautkan langsung ke server, maka kita bisa gunakan tautan ini <https://cdn.jsdelivr.net/npm/vue@2.6.10/dist/vue.js>

Catatan: sesuaikan dengan versi saat ini. Silahkan cek versi yang tersedia pada tautan ini <https://cdn.jsdelivr.net/npm/vue>

```
1 | <script src="https://cdn.jsdelivr.net/npm/vue@2.6.10/dist/vue.js">
</script>
```

Catatan: pastikan bahwa ketika aplikasi akan dilaunching (production) maka ubah vue.js menjadi vue.min.js untuk tujuan keamanan dan performa. Namun untuk pengembangan, tetap disarankan menggunakan mode development.

Untuk mengembangkan aplikasi skala besar, maka instalasi Vue disarankan dengan menggunakan package manager seperti NPM (penulis menggunakan ini) atau YARN. Disamping itu, Vue membuat tools CLI <https://cli.vuejs.org> yang akan memudahkan kita dalam membuat scaffolding projek aplikasi (manajemen kode & tools serta konfigurasi saat pengembangan aplikasi). Topik ini akan dibahas tuntas pada bagian berikutnya.

Hello World

Cara klasik belajar untuk mulai belajar suatu bahasa pemrograman atau pustaka baru adalah dengan cara membuat kode untuk menampilkan teks "hello world" menggunakan bahasa atau pustaka tersebut. Apabila kita bisa membuatnya maka konon selanjutnya akan lebih mudah. Cara ini akan kita gunakan untuk mengawali belajar Vue pada buku ini.

Disini, kita akan buktikan seberapa natural Vue bagi kamu yang sudah terbiasa dengan HTML, CSS & JS. Mari kita mulai dengan membuat file HTML dengan nama index.html (tentu saja kamu boleh menggunakan nama lain) sebagaimana kode HTML pada umumnya.

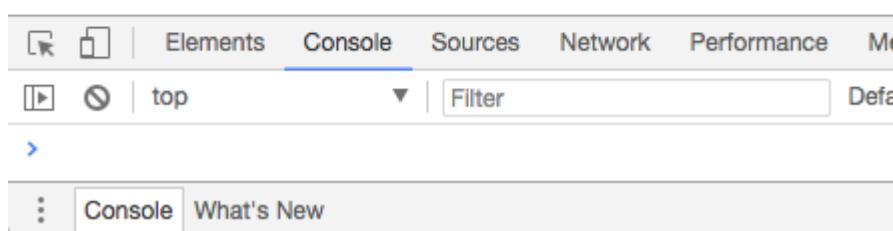
```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Belajar Vue</title>
5  </head>
6  <body>
7      <div id="app">
8          <h1>Hello world</h1>
9      </div>
10 </body>
11 </html>
```

Kemudian jalankan file ini pada browser, maka hasilnya sebagai berikut.



Hello world



Maka pada browser akan muncul teks "Hello world". Apakah ini cukup natural? Mudah sekali bukan? Oh bukan, penulis hanya bercanda, itu bukan Vue, itu hanya HTML biasa 😊. Karenanya, mari kita ubah kode di atas (di dalam tag HTML body) menjadi sebagai berikut.

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Belajar Vue</title>
5      <script src="lib/vue.js"></script>
6  </head>
7  <body>
```

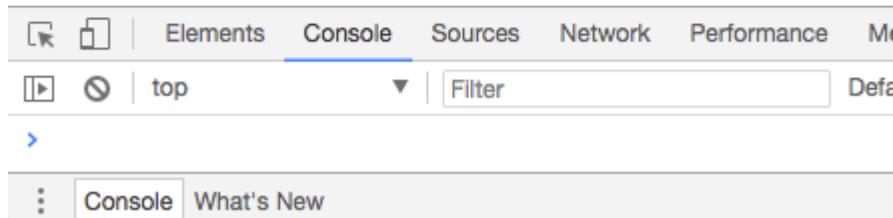
```

8 <div id="app">
9   <h1>{{ message }}</h1>
10 </div>
11 <script type="text/javascript">
12 var vm = new Vue({
13   el: '#app',
14   data: {
15     message: 'Hello world!'
16   }
17 })
18 </script>
19 </body>
20 </html>
```

Kode di atas akan menghasilkan tampilan sebagai berikut.



Hello world



Wah kok ribet sekali? Hanya untuk menuliskan HTML di web browser kodennya sepanjang itu! Apa kelebihannya?

Baik, pada contoh ini memang tidak ada kelebihannya, bahkan tidak disarankan untuk menggunakan kode ini jika hanya untuk menampilkan teks statis pada browser. Namun dari kode sederhana ini, kita akan belajar tentang bagaimana Vue tersebut bekerja.

- Pertama Kita butuh HTML untuk menjalankan kode-kode Vue, karena kita tahu bahwa Vue hanyalah sebuah pustaka Javascript yang tugasnya memanipulasi tampilan HTML.
- Kedua Kita perlu menambahkan (include) pustaka Vue ke HTML sebagaimana yang telah dijelaskan pada bagian **Instalasi** karena Vue merupakan pustaka Javascript

```
1 <script src="lib/vue.js"></script>
```

```

1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Belajar Vue</title>
5          <script src="lib/vue.min.js"></script>
6      </head>
7      <body>

```

Catatan: pustaka Vue tidak harus diletakkan di dalam elemen head, bisa juga di dalam body.

- Ketiga Kita perlu membuat kontainer (mount point) berupa elemen HTML, untuk menandai bahwa di dalam elemen tersebut nantinya hasil kompilasi Vue akan ditampilkan atau dimuat. Sebagai penanda, kita perlu tambahkan atribut id pada tag tersebut yang nantinya akan didefinisikan pada saat inisiasi objek Vue.

```

1  <div id="app">
2      ...
3  </div>

```

Catatan: Nilai dari atribut id tidak harus app, bebas saja, tergantung definisi di saat inisiasi objek Vue.

- Keempat Kita perlu menggunakan double kurung kurawal (mustache) untuk menandai bahwa teks tersebut merupakan variabel yang akan dimanipulasi oleh Vue, model seperti ini telah umum digunakan diberbagai template engine.

```

1  <div id="app">
2      <h1>{{ message }}</h1>
3  </div>

```

Di samping itu, kita juga bisa menggabungkannya dengan teks statis.

```

1  <h1>Pesan: {{ message }}</h1>

```

atau menggunakan operasi Javascript untuk menggabungkan dua teks (string) tersebut.

```

1  <h1>{{ 'Pesan: ' + message }}</h1>

```

- Kelima Kita perlu membuat instance/objek baru untuk class Vue, yang tentunya ditulis dengan menggunakan Javascript.

```

1  var vm = new Vue({
2      el: '#app',
3      data: {
4          message: 'Hello world!'
5      }
6  })

```

Objek Vue yang dibuat ini disimpan ke dalam variabel bernama `vm` (nama bebas) untuk memudahkan kita nantinya dalam mengakses objek ini. Objek Vue pada kode di atas menggunakan dua properti yaitu `el` dan `data`. Properti `el` menunjukkan id dari elemen HTML yang akan dijadikan sebagai target atau tempat ditampilkannya hasil manipulasi data dan template.

Di samping itu bisa juga kita gunakan perintah `vm.$mount('#app')` untuk mengarahkan mount point Vue pada saat runtime.

```

1 var vm = new Vue({
2   data: {
3     message: 'Hello world!'
4   }
5 )
6
7 vm.$mount('#app')

```

Properti berikutnya adalah data yang berbentuk objek, dimana di dalamnya terdapat key `message` dengan nilai ‘Hello world!’ yang merupakan representasi dari variabel. Key atau variabel dalam properti data inilah yang akan mengubah kode template `{{ message }}` (lihat poin keempat) menjadi teks “Hello world!”. Dengan kata lain, jika kita mengubah nilai dari variabel `message` ini menjadi misalnya “Hello Vuejs!” maka tentu tampilan yang kita lihat pada browser juga akan berubah sesuai teks tersebut. Sederhana sekali bukan?

Mungkin kamu melihatnya sederhana, namun Vue telah melakukan hal yang mungkin lebih dari yang kamu bayangkan. Vue diam-diam telah menghubungkan antara DOM dengan data atau variabel, dimana sekarang keduanya menjadi reaktif.

Catatan: kita bisa mengakses properti dan variabel dalam objek `vm` tersebut, misalnya untuk mengakses properti `el` atau `data` maka kita bisa gunakan perintah `vm.$el` atau `vm.$data`. Adapun untuk mengakses variabel `message` dalam properti `data` kita bisa gunakan perintah `vm.$data.message` atau langsung `vm.message` (tanpa tanda dollar).

Menguji Reaktifitas

Pada bagian awal ini, mari kita bereksperimen untuk menguji sifat reaktif dari Vue dengan cara yang sederhana. Kita akan menggunakan console pada browser (penulis menggunakan Chrome).

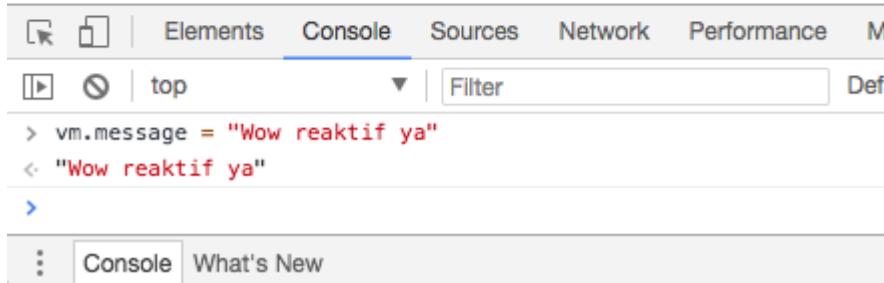
Pada console, mari kita ubah variabel `message`, dengan menggunakan perintah berikut berikut:

```
1 vm.message = "Wow reaktif ya"
```

Tekan enter dan lihat apa yang terjadi?



Wow reaktif ya



Yap, tanpa refresh maka seketika itu juga teks yang muncul di halaman browser berubah sesuai dengan teks yang kita sematkan pada variabel message di console.

Pada kondisi nyata, tentu saja perubahan data tidak dilakukan dengan menggunakan console pada browser melainkan melalui cara-cara yang natural yaitu menggunakan perintah JS yang dijalankan misalnya melalui event onclick button, input dari user, dsb.

Kesimpulan

Vue merupakan pustaka JS yang bekerja memanipulasi elemen HTML menggunakan teknik virtual DOM HTML sehingga lebih cepat prosesnya dibandingkan langsung memanipulasi DOM-nya. Untuk menggunakannya pada aplikasi, maka pustaka Vue cukup diincludekan menggunakan element HTML script sebagaimana umumnya pustaka JS.

Sekarang kita telah mengenal Vue dan bagaimana cara kerja Vue. Penulis berharap, kamu benar-benar memahami apa yang telah kita bahas pada bab ini, jika belum maka sebaiknya kamu mengulanginya lagi dan sekali lagi hingga benar-benar faham.

Untuk memperkuat pemahaman kamu, maka pada bab selanjutnya, akan dibahas mengenai dasar-dasar Vue yang tentunya lebih dari sekedar hello world 😊.

Bagaimana? Penasaran?

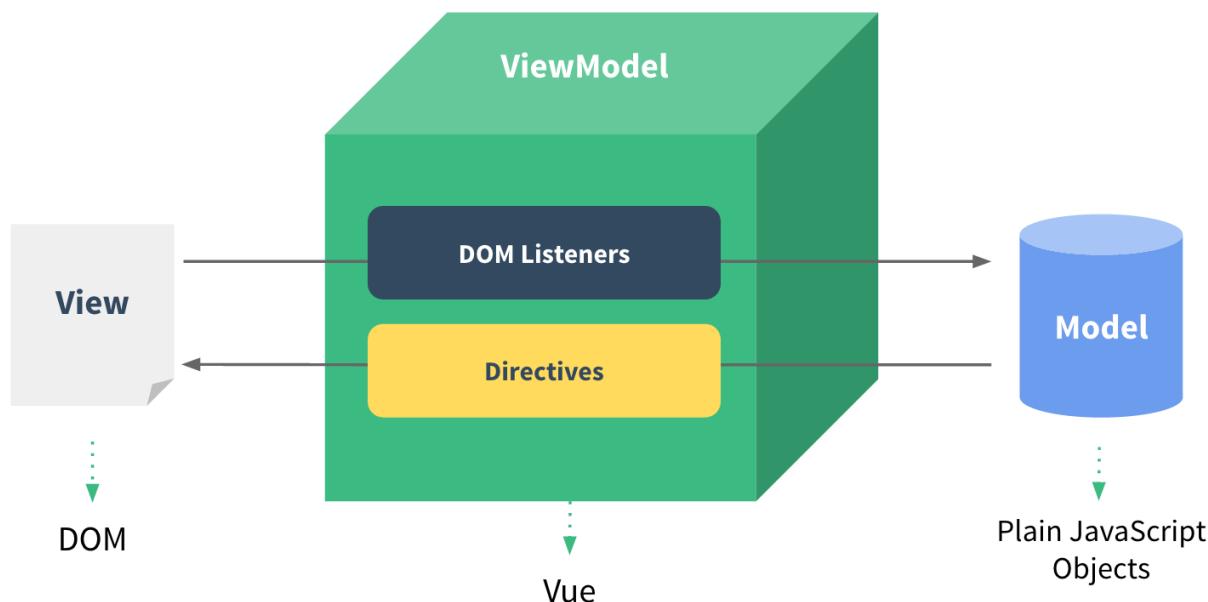
Dasar-Dasar Vue

Intro

Setelah kita mengenal Vue dan mengetahui bagaimana Vue bekerja melalui latihan menampilkan teks “Hello world”, maka kini saatnya kita menyelami kode-kode dasar pada Vue.

Objek Vue

Meski tidak benar-benar strict mengikuti pattern MVVM (https://en.wikipedia.org/wiki/Model_View_Model), namun desain dari Vue sebagiannya terinspirasi dari sana.



Teknisnya, Vue fokus pada layer ViewModel dari pattern MVVM. ViewModel merupakan penghubung antara View (DOM/tampilan) dan Model (objek data). Manipulasi pada View dipantau oleh DOM Listeners kemudian diterima oleh ViewModel untuk digunakan mengupdate Model, demikian juga perubahan data pada Model akan diteruskan oleh ViewModel ke View melalui Directives.

Pada Vue, objek Vue inilah yang berperan menjadi ViewModel pada konsep MVVM. Oleh karena itu, kita mulai dengan membedah objek Vue, yaitu objek utama yang harus ada jika kita menggunakan Vue.

Inisiasi Objek Vue

Sebagaimana yang telah dicontohkan sebelumnya, bahwa setiap aplikasi Vue dimulai dengan menginisiasi objek Vue.

```

1 var vm = new Vue({
2   // options
3 })

```

Variabel `vm` (singkatan dari ViewModel) pada contoh di atas mengacu pada objek Vue, meski sebenarnya kita tidak harus membuat variabel baru (`vm`) untuk menyimpan objek dari Vue kecuali jika memang dibutuhkan. Sedangkan `options` bisa kita isi dengan properti-properti yang telah didefinisikan oleh Vue seperti `data`, `methods`, `components`, dsb.

Setiap sub bagian dari Vue (baca: komponen), memiliki skema yang kurang lebih sama. Di mana objek Vue di atas sebagai parent atau root, kemudian dibawahnya dengan struktur hirarki pohon bisa kita buat komponen dan sub komponennya. Komunikasi data antara komponen parent dengan komponen anaknya secara native menggunakan props.

Bisa dikatakan bahwa aplikasi yang dibangun dengan Vue merupakan kesatuan dari berbagai komponen Vue di dalamnya. kamu pernah mendengar istilah **web component**?, jika pernah maka Vue sebagiannya menggunakan konsep itu.

Penjelasan lebih lengkap mengenai komponen akan dibahas secara bertahap pada bagian selanjutnya.

Properti el

Sebagaimana penjelasan terdahulu bahwa properti el menunjuk ke tempat dimana hasil kompilasi dari template dan data akan dimuat.

```

1 var vm = new Vue({
2   el: '#app',
3   ...
4 })

```

Melalui properti ini `vm.$el`, objek Vue dapat memanipulasi DOM atau View dalam konsep MVVM.

Jika kita jalankan perintah `console.log(vm.$el)` maka akan menampilkan DOM atau template HTML.

Properti Data

Pada latihan "Hello world", kita juga telah menyinggung sedikit tentang data yaitu bagaimana cara mendefinisikan dan menggunakannya. Dimana, data merupakan salah satu properti pada objek Vue yang digunakan untuk mendefinisikan variabel-variabel yang akan digunakan pada aplikasi kita atau dalam konsep MVVM, data merupakan Model.

```

1 var vm = new Vue({
2   ...
3   data: {
4     message: 'Hello world!'
5   }
6 })

```

Pada kode di atas terlihat bahwa properti `data` berbentuk objek yang bertindak sebagai kontainer dari variabel `message`. Kita bisa juga menambahkan variabel lain dengan cara sebagai berikut.

```

1 ...
2 data: {
3   message: 'Hello world!',
4   teks: 'Semangat Pagi Indonesia'
5 }
6 ...

```

`teks` merupakan nama variabel baru yang kita tambahkan dan bernilai kalimat "Semangat Pagi Indonesia". Tentunya variabel baru ini bisa kita tampilkan dengan mudah di browser sebagai berikut.

```

1 <div id="app">
2   <h1>{{ message }}</h1>
3

```

```
4 | <p>{{ teks }}</p>
| </div>
```

Tampilan pada browser akan seperti di bawah ini.



Hello world!

Semangat Pagi Indonesia

Variabel pada properti `data` bisa kita definisikan dalam berbagai tipe data seperti string, integer, boolean, array, dan object. Berikut ini contohnya.

```
1 | data: {
2 |   name: 'Hafid', // string
3 |   age: 33, // integer
4 |   gender: true, // boolean (Pria)
5 |   hobby: ['coding','sleeping'], // array
6 |   children: {
7 |     1: 'Ammar',
8 |     2: 'Faqih',
9 |     3: 'Syamil'
10 |   } // object
11 | }
```

Sebagaimana yang telah dijelaskan pada bagian pertama, properti data yang kita definisikan ini bisa kita manipulasi isinya pada saat runtime (aplikasi berjalan), disamping itu properti data ini otomatis memiliki sifat reactive sehingga perubahan nilai pada variabel data akan memicu render ulang pada view.

Sifat reactive pada data ini hanya terjadi jika variabel pada properti data telah kita definisikan saat objek Vue dibuat, atau dengan kata lain, variabel yang didefinisikan pada saat runtime tidak akan memiliki sifat reactive.

Properti data ini bisa kita akses secara langsung dari dalam objek Vue dengan menggunakan `this.$data` atau `this.message`, sedangkan jika diakses dari luar objek Vue menggunakan nama variabel objeknya `vm.$data` atau `vm.message`.

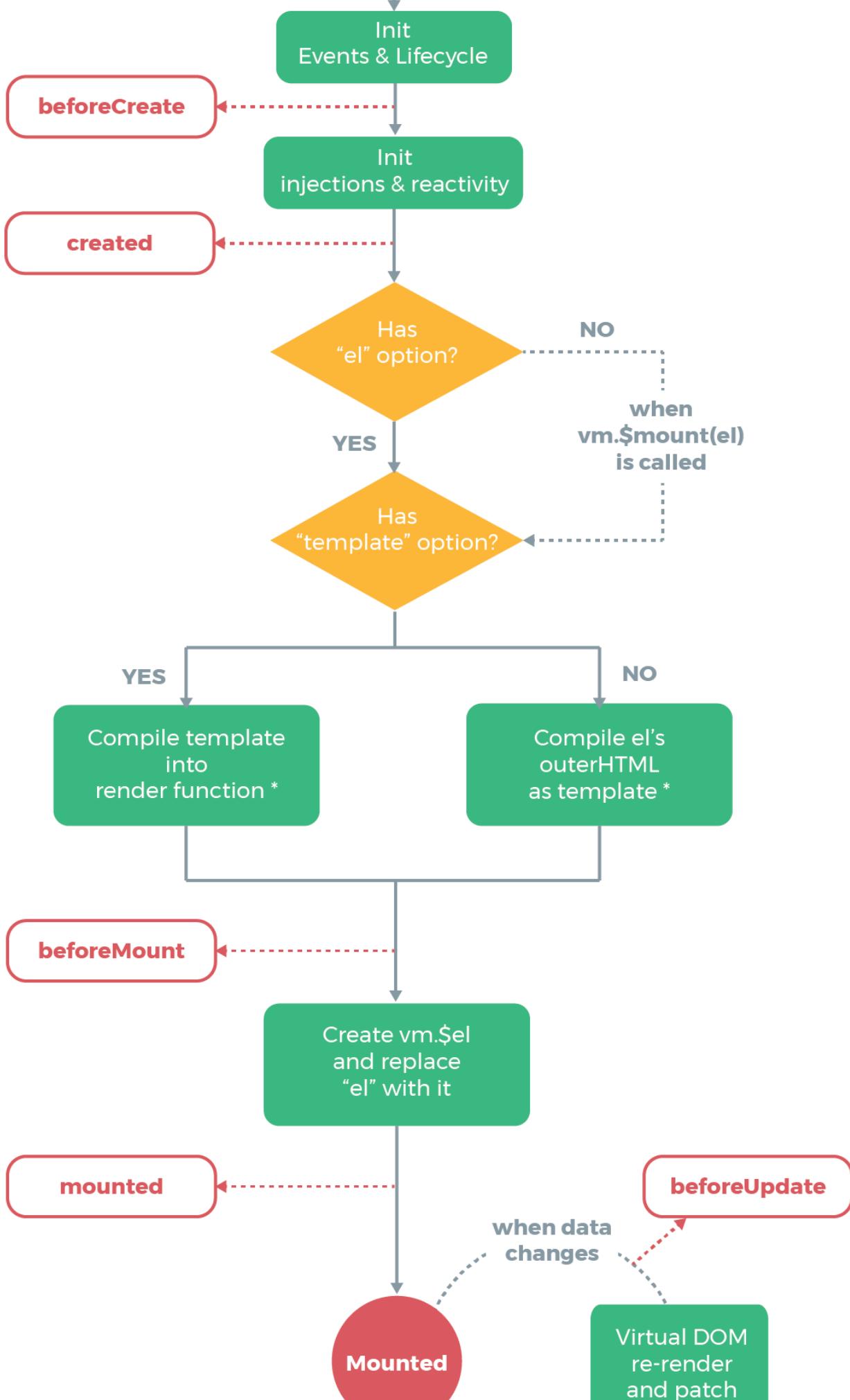
Siklus Objek Vue

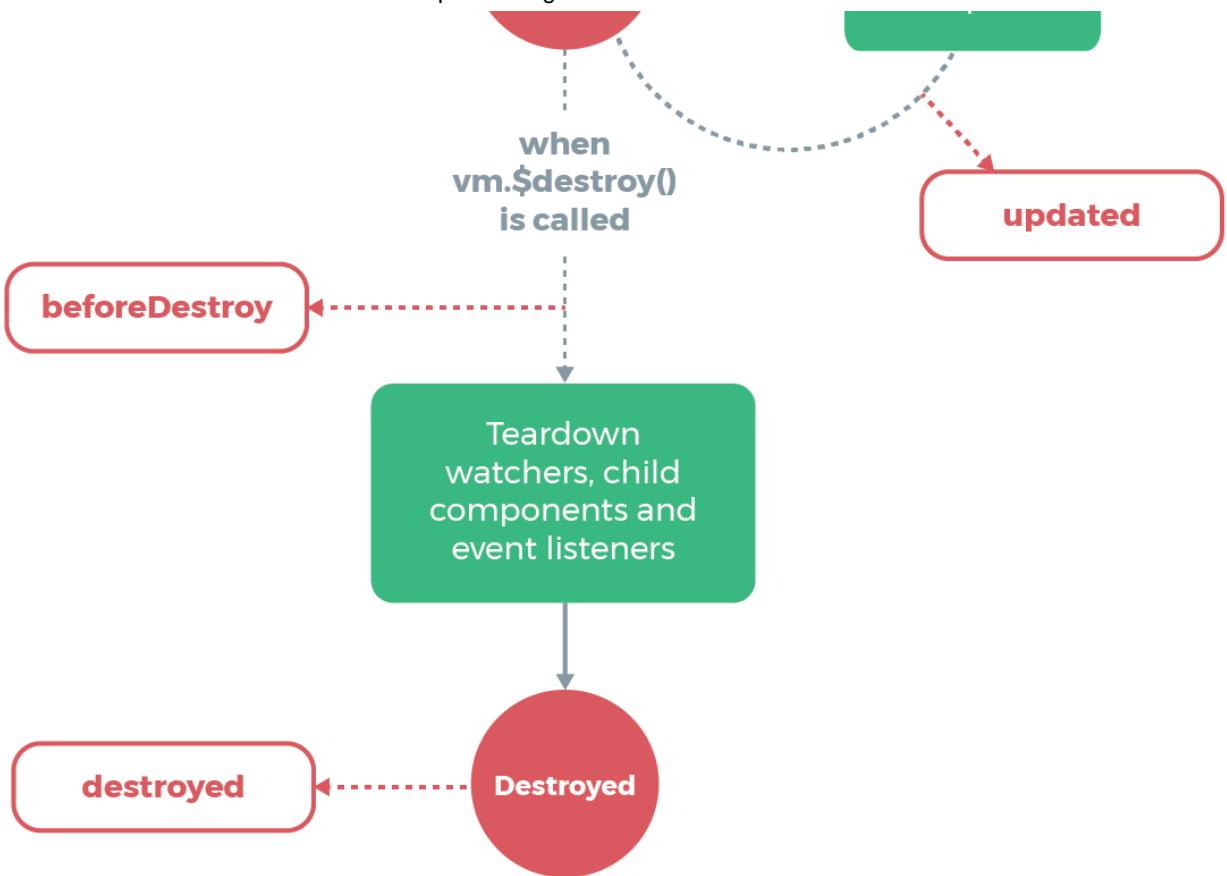
Sebelum kita mengeksplor lebih dalam lagi, ada satu hal yang perlu kita ketahui terlebih dahulu untuk memudahkan kita dalam memahami Vue secara holistik yaitu siklus objek Vue.

Objek Vue yang kita definisikan sebenarnya memiliki siklus hidup atau life cycle mulai dari ketika objek itu diciptakan, view dirender, data dimuat, hingga objek itu dihapus. Dimana pada setiap tahap siklus tersebut terdapat hook yang bisa kita manfaatkan untuk menjalankan suatu perintah tertentu.

Perhatikan gambar siklus objek Vue berikut.

`new Vue()`





* template compilation is performed ahead-of-time if using a build step, e.g. single-file components

Terdapat 8 hooks pada siklus tersebut yang digambarkan dengan shape berborder merah. 8 Siklus ini bisa dikelompokkan menjadi 4 bagian.

create

1. beforeCreate yaitu hook sesaat setelah objek Vue dan komponennya diinisialisasi. Properti data belum dapat diakses atau digunakan pada hook ini.
2. created yaitu hook ketika objek Vue telah selesai diciptakan. Pada hook ini, sifat reactivity pada properti data juga sudah didefinisikan sehingga kita sudah diizinkan untuk mengakses dan memanipulasi data. Properti computed yang digunakan untuk memonitor perubahan data juga sudah berjalan. Jika aplikasi membutuhkan request data dari server maka hook ini adalah hook yang tepat untuk melakukannya.

Berikut ini contoh kode untuk menggunakan kedua hook ini yaitu dengan method beforeCreate dan created.

```

1 var vm = new Vue({
2   el: '#app',
3   data: {
4     message: 'Hello world!',
5   },
6   beforeCreate () {
7     console.log('before create: '+
8       'message = ' + this.message)
9   },

```

```

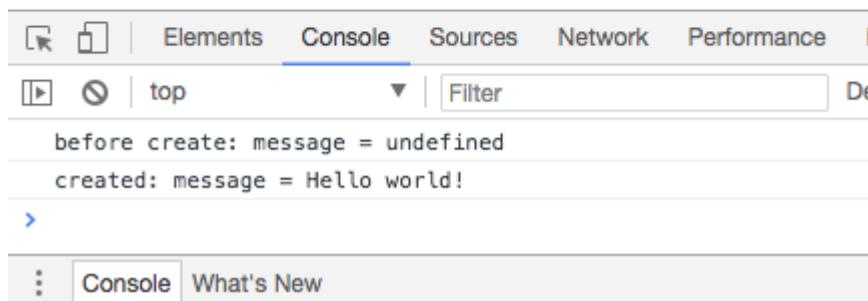
10     created () {
11         console.log('created: '+
12             'message = ' + this.message)
13     },
14 );

```

Pada kode di atas, kedua hook sama-sama digunakan untuk mengakses variabel message.



Hello world!



Pada gambar di atas terlihat bahwa hook created bisa mengakses variabel message sebaliknya beforeCreate tidak bisa.

Apa yang terjadi pada properti data dapat kita lihat juga dengan cara tambahkan kode hook created menjadi sebagai berikut.

```

1  created () {
2      console.log('created: '+
3          'message = ' + this.message)
4      console.log(this.$data)
5  },

```

Properti data dapat diakses secara langsung menggunakan tanda \$ (dollar) di depan data.



Hello world!

```

before create: message = undefined
created: message = Hello world!
▼ {__ob__: _e} ⓘ
  message: (...)

▶ __ob__: _e {value: {...}, dep: oe, vmCount: 1}
▶ get message: f ()
▶ set message: f (e)
▶ __proto__: Object

```

Console What's New

Pada gambar di atas terlihat bahwa properti data bertipe observer yang artinya nilainya senantiasa dievaluasi oleh browser (reactive).

Catatan: penulisan fungsi atau method created() pada kode di atas merupakan shorthand atau bentuk penulisan singkat dari fungsi. Bentuk panjangnya seperti berikut ini.

```

1  created: function () {
2      console.log('created: '+
3          'message = ' + this.message)
4      console.log(this.$data)
5  },

```

Meskipun kita dapat menggunakan format penulisan ES6, namun arrow function tidak disarankan untuk digunakan pada kasus ini, sebab keyword this di dalam arrow function tidak menunjuk ke objek Vue melainkan ke objek di atasnya (pada kasus ini objek window).

Berikut ini contoh kode untuk melihat bahwa this pada dua jenis fungsi merujuk ke konteks yang berbeda.

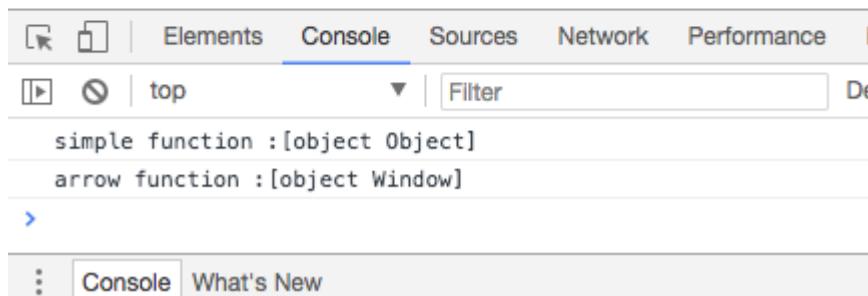
```

1  beforeCreate () {
2      console.log('simple function :'+this)
3  },
4  created: () => {
5      console.log('arrow function :'+this)
6  },

```



Hello world!



Gambar di atas menunjukkan bahwa this pada fungsi biasa merujuk ke objek Vue sedangkan pada arrow function merujuk ke objek Window.

Kembali ke topik, meskipun hook created sudah bisa mengakses properti data namun Virtual DOM & template (komponen) belum bisa diakses.

mount

3. beforeMount yaitu hook ketika template dicompile.
4. mounted yaitu hook ketika elemen (properti el) telah diinisialisasi, data telah dimuat dan view telah dirender.

Untuk mencoba kedua hook ini mari kita tambahkan kode sebagai berikut (hapus kode hook sebelumnya).

```

1 beforeMount() {
2     console.log('before mount: '+
3         'el = ' + this.$el.textContent)
4 },
5 mounted() {
6     console.log('mounted: '+
7         'el = ' + this.$el.textContent)
8 },

```

Pada kode di atas, kita akan mencoba mengakses DOM dengan menggunakan fungsi bawaan Javascript yaitu `textContent` untuk menampilkan konten teks yang adalah di dalam el atau dalam hal ini `#app`.

Berikut ini hasilnya.



Hello world!

```
before mount: el =
{{ message }}

mounted: el = Hello world!
>
```

Console What's New

Kedua hook ini dapat mengakses DOM, namun pada hook beforeMount, data belum dirender dengan template, sedangkan hook mounted sudah.

update

5. beforeUpdate yaitu hook yang terjadi setelah mounted dan hanya terjadi jika ada perubahan data yang mengakibatkan render ulang. Tepatnya, hook ini terjadi sebelum view dirender ulang.
6. updated yaitu hook yang terjadi setelah beforeUpdate yaitu setelah view dirender ulang.

Untuk mengujinya, mari kita tambahkan kode berikut (hapus hook sebelumnya)

```
1 beforeUpdate() {
2     console.log('before update: '+
3         'el = ' + this.$el.textContent)
4 },
5 updated() {
6     console.log('update: '+
7         'el = ' + this.$el.textContent)
8 },
```

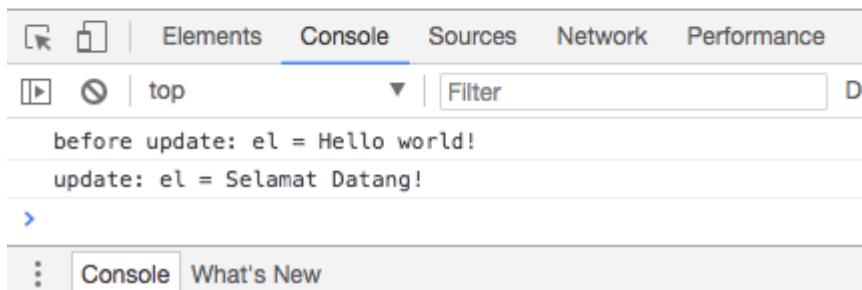
Oleh karena hook ini terjadi karena adanya manipulasi atau perubahan nilai data pada saat runtime, maka kita perlu sebuah perintah untuk mengubah variabel message. Di bawah objek Vue kita tambahkan perintah berikut.

```
1 vm.message = 'Selamat Datang!'
```

Kode diatas berfungsi mengubah variabel message yang sebelumnya Hello world menjadi Selamat datang



Selamat Datang!



Pada gambar di atas terlihat bahwa pada hook beforeUpdate, variabel message masih bernilai Hello world sedangkan pada hook updated, variabel message telah berubah menjadi Selamat datang. Hook beforeUpdate ini setara dengan method watch.

```

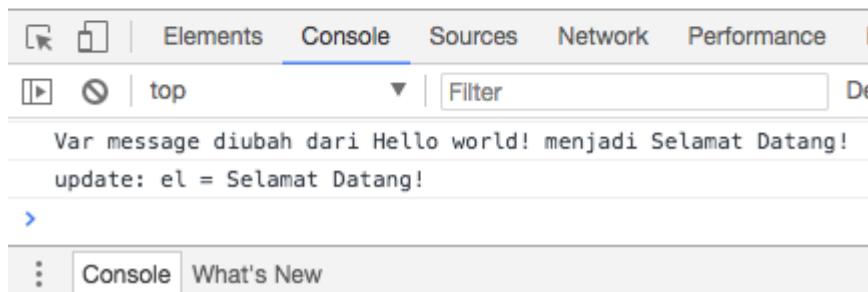
1 var vm = new Vue({
2   el: '#app',
3   data: {
4     message: 'Hello world!',
5   },
6   beforeUpdate() {
7     console.log('before update: '+
8       'el = ' + this.$el.textContent)
9   },
10  updated() {
11    console.log('update: '+
12      'el = ' + this.$el.textContent)
13  },
14)
15
16 vm.$watch('message', function (newValue, oldValue) {
17   console.log('Var message diubah dari '+oldValue+
18     ' menjadi '+newValue)
19 }
20
21 vm.message = 'Selamat Datang!'

```

Hasilnya sebagai berikut:



Selamat Datang!



destroy

7. beforeDestroy yaitu hook yang terjadi sebelum component dihapus.
8. destroyed yaitu hook yang terjadi setelah objek Vue dihapus.

Untuk menguji hook ini, sebenarnya kita perlu mengenal terlebih dahulu komponen. Karena sejak versi 2 ini objek utama Vue "tidak bisa" dihapus, namun kita masih bisa mensimulasikannya melalui method `vm.$destroy()`.

Berikut ini contoh kodenya.

```

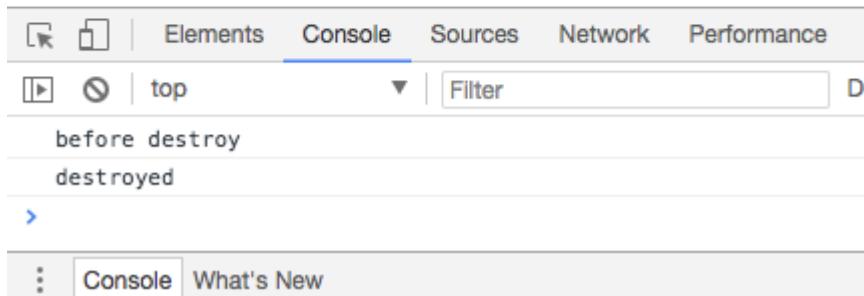
1 var vm = new Vue({
2   ...
3   beforeDestroy () {
4     console.log('before destroy')
5   },
6   destroyed () {
7     console.log('destroyed')
8   },
9 }
10 vm.$destroy()
11

```

Hasilnya sebagai berikut.



Hello world!



Penulisan Template

Pada Vue, template merupakan kode yang menjadi dasar dari suatu tampilan, umumnya kode template ditulis dengan menggunakan bahasa HTML. Variabel pada template ditulis dengan menggunakan tanda kurung kurawal (mustache). Template dan variabel (data) dicompile oleh Vue menjadi Virtual DOM sebelum akhirnya dirender atau ditampilkan dalam bentuk HTML DOM. Ketika terjadi perubahan data maka akan memicu render ulang dari DOM.

Bentuk-bentuk data terkait hubungannya dengan template ada beberapa macam, dan hal itu menuntut perlakuan yang berbeda.

Data Teks

Umumnya data dalam bentuk teks biasa, maka cara penulisannya menggunakan mustache tags atau double kurung kurawal.

```
1 | <h1>{{ message }}</h1>
```

Variabel dalam mustache tags ini akan diubah sesuai dengan variabel pada properti data yang didefinisikan di objek Vue. Ketika variabel message pada objek Vue diubah nilainya pada saat runtime maka secara otomatis variabel message pada template juga akan berubah nilainya. Hal ini telah kita uji coba pada pembahasan sebelumnya.

Untuk mencegah perubahan nilai variabel template pada saat runtime, kita bisa gunakan directive v-once.

```
1 | <h1 v-once>{{ message }}</h1>
```

Dengan menggunakan directive ini, maka variabel message pada template seolah menjadi sebuah konstanta yang tidak dapat diubah pada saat runtime, nilainya sesuai dengan ketika pertama kali didefinisikan di objek Vue.

Data Raw HTML

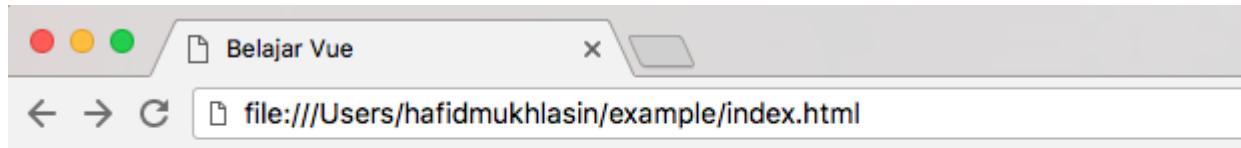
Adakalanya variabel data yang kita ingin tampilkan tidak dalam bentuk teks biasa namun dalam bentuk kode HTML, sebagai contoh variabel message bernilai sebagai berikut

```

1 var vm = new Vue({
2   el: '#app',
3   data: {
4     message: "<span style='color:red'>Hello World!</a>",
5   },
6 })

```

Apabila kita jalankan maka hasilnya kode HTML akan muncul di browser.



Hello World!

Oleh karena itu, jika kita menggunakan data dalam bentuk HTML maka kita perlu ubah templatanya menggunakan directive v-html

```

1 <h1 v-html="message"></h1>

```

Hasilnya sebagai berikut:



Hello World!

Peringatan: pastikan data HTML yang ingin ditampilkan menggunakan directive v-html ini adalah data yang terpercaya, sebab sangat berpotensi menjadi celah keamanan XSS Vulnerability

Data Attribute

Misalnya kita ingin menggunakan class CSS yang dinamis pada template, maka kita tidak bisa menggunakan kode berikut.

```

1 <style type="text/css">
2   .title{
3     color: green;
4   }
5 </style>
6
7 <div id="app">
8   <h1 class="{{ class_h1 }}> {{ message }} </h1>
9 </div>
10
11 <script type="text/javascript">
12   Vue.config.silent = true
13   var vm = new Vue({

```

```

14     el: '#app',
15     data: {
16       message: "Hello world!",
17       class_h1: "title"
18     },
19   })
20 </script>

```

Hal ini dikarenakan mustache tidak dapat digunakan didalam atribut HTML. Oleh karena itu kita harus menggunakan directive v-bind

```

1 <h1 v-bind:class="class_h1"> {{ message }} </h1>

```

Hasilnya.



Hello world!

JS Expression

Sebagaimana yang dicontohkan sebelumnya bahwa template Vue mendukung kode-kode JS.

```

1 {{ 'Pesanan : ' + message }}
2
3 {{ 'Diskon : ' + total * 10% }}
4
5 {{ ok ? 'YES' : 'NO' }}
6
7 {{ message.split(' ').reverse().join(' ') }}

```

Atau jika dalam bentuk atribut HTML maka penulisannya sebagai berikut.

```

1 <h1 v-bind:id="'product-' + index"></h1>

```

Informasi: Javascript expressions hanya dapat menjalankan kode Javascript dasar dan tidak dapat digunakan untuk mendefinisikan nilai.

Contoh-contoh berikut tidak akan berjalan.

```

1 <!-- ini adalah statement, bukan expression -->
2 {{ var a = 1 }}
3
4 <!-- gunakan ternary expressions -->
5 {{ if (ok) { return message } }}

```

Properti Template

Pada contoh kode sebelumnya, template ditulis jadi satu dengan kode HTML-nya.

```

1 <div id="app">
2   <h1>{{ message }}</h1>
3 </div>

```

Namun Vue juga menyediakan cara lain untuk mendefinisikan template yaitu menyatu dengan object Vue melalui properti template.

```

1 ...
2 <div id="app"></div>
3
4 <script type="text/javascript">
5 var vm = new Vue({
6   el: '#app',
7   data: {
8     message: 'Hello world!'
9   },
10  template: "<h1>{{ message }}</h1>"
11 }
12 </script>
13 ...

```

Template ini akan dicompile dengan data oleh Vue sebelum ditampilkan pada elemen HTML dengan id app. Alternatif lain kita bisa juga menggunakan method render. Sesuai dengan namanya, method ini berfungsi menampilkan konten yang didefinisikan.

```

1 var vm = new Vue({
2   el: '#app',
3   data: {
4     message: 'Hello world!'
5   },
6   render (createElement) {
7     return createElement('h1', this.message)
8   },
9 ...

```

Pada kode di atas, method render mengembalikan fungsi createElement untuk menciptakan elemen HTML h1 yang berisi nilai dari variabel message. Melalui method render ini, dengan bantuan babel (pustaka Javascript untuk transform code), kita bisa gunakan format penulisan JSX layaknya React.

fungsi createElement tidak hanya dapat menciptakan elemen HTML saja namun juga dapat menciptakan elemen component (component akan dibahas lebih lanjut pada bab selanjutnya) yang kita buat.

Misalnya untuk menciptakan elemen Hello.

```

1 render (createElement) {
2   return createElement(Hello);
3 }

```

Penulisan nama fungsi createElement ini kemudian dialiaskan menjadi h, sehingga kita bisa menulisnya sebagai berikut.

```

1 render (h){
2     return h(App);
3 }
```

Atau jika ditulis menggunakan es6 arrow function menjadi sebagai berikut.

```

1 render: h => h(App)
```

Informasi: Mengapa createElement disingkat menjadi h? konon kata om Evan You (<https://github.com/vuejs/babel-plugin-transform-vue-jsx/issues/6>) itu singkatan dari "hyperscript" yang umumnya digunakan dalam banyak implementasi virtual-dom. "Hyperscript" sendiri merupakan "script yang menggenerate struktur HTML" sebab HTML adalah akronim dari "hyper-text markup language".

Vue lebih merekomendasikan penggunaan template untuk sebagian besar kasus. Namun pada suatu kasus yang membutuhkan kekuatan lebih dari Javascript adakalanya kita perlu menggunakan render ini. Contohnya adalah untuk menampilkan component utama di mana di dalam component tersebut akan dibuat sub component.

Catatan: jika properti template dan render dua-duanya ada maka properti template akan diabaikan.

Properti Methods, Computed, & Filters

Pada objek Vue juga ada properti methods, computed dan filter yang terkadang cukup membingungkan para pemula tentang kapan saat yang tepat menggunakannya karena ketiganya sama-sama berisi fungsi-fungsi. Meskipun telah jelas definisinya namun ketiga properti ini terkadang dapat menyelesaikan kasus yang sama.

Properti Methods

Properti methods dapat berisi fungsi-fungsi (Javascript tentunya) yang dapat **dipanggil** disemua tempat pada aplikasi berbasis Vue. Jika ada action atau event yang memanggil suatu fungsi, maka fungsi tersebut cocok dikategorikan sebagai methods. Contoh: fungsi untuk mengevaluasi suatu nilai, menampilkan pesan, mengubah variabel, dsb.

Lihat contoh berikut.

```

1 var vm = new Vue({
2     el: '#app',
3     data: {
4         counter: 0
5     },
6     methods: {
7         increment () {
8             this.counter++
9         }
10    }
11 })
```

Adapun templatanya sebagai berikut.

```

1 <div id="app">
2     <h1>{{ counter }}</h1>
3 
```

```
4 | <button onclick="vm.increment()"> + </button>
| </div>
```

Fungsi increment pada contoh di atas dijalankan saat button diklik (event onclick). Fungsi tersebut mengubah nilai variabel counter menjadi increment 1 (+1), hal ini akan men-trigger terjadinya render ulang terhadap template sehingga tampilan counter berubah sesuai dengan nilainya.

Properti Computed

Properti computed berisi fungsi-fungsi yang nilainya akan senantiasa dievaluasi ketika terjadi perubahan variabel data yang menjadi dependensinya. Fungsi pada computed umumnya mengembalikan nilai (return value).

Contoh

```
1 | var vm = new Vue({
2 |   el: '#app',
3 |   data: {
4 |     firstName: 'Hafid',
5 |     lastName: 'Mukhlasin'
6 |   },
7 |   computed: {
8 |     fullName: function () {
9 |       return this.firstName + ' ' + this.lastName
10 |     }
11 |   }
12 | })
```

Adapun templatanya sebagai berikut.

```
1 | <div id="app">
2 | {{ fullName }}
3 | </div>
```

Fungsi fullName pada contoh diatas mengembalikan nilai berupa gabungan string antara variabel firstName dan lastName. Fungsi ini akan di cache oleh Vue sehingga nilainya akan selalu merujuk ke nilai sebelumnya kecuali jika ada perubahan variabel firstName dan lastName yang menjadi dependensinya.

Catatan: Meskipun bentuknya fungsi namun fungsi pada computed tidak memiliki parameter dan oleh Vue tidak dianggap sebagai fungsi. Artinya kita tidak bisa memanggilnya `this.fullName()` melainkan `this.fullName` layaknya variabel. Jadi methods ini lebih tepat digunakan sebagai variabel yang nilainya berasal dari variabel lain.

Contoh implementasi computed yang kurang tepat, misalnya computed untuk mendapatkan waktu saat ini.

```
1 | var vm = new Vue({
2 |   el: '#app',
3 |   data: {
4 |     computed: {
5 |       now: function () {
6 |         return Date.now()
7 |       }
8 |     }
9 |   }
10 | })
```

```

9  })
10
11 setInterval(()=>{
12     vm.now
13 }, 1000)

```

Adapun pada template.

```

1 <div id="app">
2     <h1>{{ now }}</h1>
3 </div>

```

Fungsi now mengembalikan waktu timestamp saat ini, adapun setInterval akan memanggil fungsi now setiap detik. Namun kalau kode ini dijalankan maka tampilan waktu saat ini tidak akan berubah sama sekali. Mengapa itu bisa terjadi? bukankah timestamp (`Date.now()`) akan berubah tiap miliseconds? Yap timestamp memang akan berubah tiap saat, fungsi now yang dipanggil setiap detik akan selalu mengambil nilai dari cache karena tidak ada variabel reactive yang menjadi dependensi dari fungsi tersebut. Yap, `Date.now()` bukan reactive variabel bagi Vue.

Properti Filters

Disamping methods dan computed, Objek Vue juga memiliki properti filters yang dapat berisi fungsi untuk digunakan memanipulasi tampilan atau format teks pada template. Filters ditulis dengan menggunakan simbol | atau “pipe”.

Contoh penggunaan filters adalah untuk mengubah bentuk teks menjadi huruf kapital.

```

1 <h1>{{ message | upper }}</h1>

```

Fungsi `upper` dideklarasikan pada objek Vue properti filters,

```

1 var vm = new Vue({
2     el: '#app',
3     data: {
4         message: 'Hello world!',
5     },
6     filters: {
7         upper (text) {
8             return text.toUpperCase()
9         }
10    }
11 })

```

Apabila dijalankan maka pada browser akan muncul teks `HELLO WORLD!` dengan huruf kapital.

Catatan: berbeda dengan methods yang bisa dipanggil dengan cara `vm.upper()` atau computed `vm.fullName`, fungsi filters tidak bisa dipanggil dari objek Vue.

Bukankah kita juga bisa menggunakan methods dan computed untuk menyelesaikan kasus ini? mari kita coba!

```

1 var vm = new Vue({
2   el: '#app',
3   data: {
4     message: 'Hello world!',
5   },
6   filters: {
7     upper (text) {
8       return text.toUpperCase()
9     }
10  },
11  methods: {
12    upper (text) {
13      return text.toUpperCase()
14    }
15  },
16  computed: {
17    messageUpperCase () {
18      return this.message.toUpperCase()
19    }
20  }
21 })

```

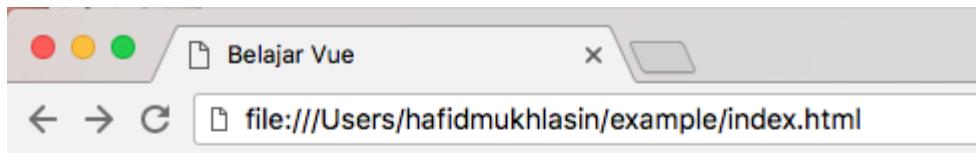
Pada template, kita panggil ketiganya.

```

1 <!-- filters -->
2 <h1>Filters: {{ message | upper }}</h1>
3
4 <!-- methods -->
5 <h1>Methods: {{ upper(message) }}</h1>
6
7 <!-- computed -->
8 <h1>Computed: {{ messageUpperCase }}</h1>

```

Mari kita lihat hasilnya:



Filters: HALO APA KABAR?

Methods: HALO APA KABAR?

Computed: HALO APA KABAR?

```
Elements Console Sources Network Performance Memory
top Filter Default level:
> vm.message = "Halo apa kabar?"
< "Halo apa kabar?"
```

Ketiganya menghasilkan tampilan yang sama serta reaktif terhadap perubahan variabel data. Namun dari sisi fleksibilitas tentu computed akan tereliminasi sebab pada kasus diatas fungsi uppercase hanya bisa digunakan spesifik pada variabel message saja. Artinya jika ada variabel lain yang ingin diubah bentuknya menjadi kapital maka kita harus membuat fungsi baru pada computed.

Lantas kenapa filters? sebab filters sesuai dengan peruntukannya yaitu memanipulasi suatu teks yang diberikan pada suatu template, adapun methods umumnya digunakan ketika ada event yang memanggilnya.

Argumen Pada Filters ~4

Fungsi pada filters juga bisa menggunakan argumen parameter. Di mana parameter pertama adalah teks yang ingin dimanipulasi.

```
1 var vm = new Vue({
2   el: '#app',
3   data: {
4     price: 500000,
5   },
6   filters: {
7     formatCurrency (value, currency) {
8       var formatter = new Intl.NumberFormat('id-ID', {
9         style: 'currency',
10        currency: currency,
11        minimumFractionDigits: 2,
12      });
13      return formatter.format(value)
14    }
15  },
16})
```

Pada template bisa kita panggil sebagai berikut:

```
1 <h1> {{ price | formatCurrency('USD') }} </h1>
2 <h1> {{ price | formatCurrency('IDR') }} </h1>
```

Hasilnya sebagai berikut.



US\$500.000,00

Rp500.000,00

Chaining Filters ~4

Filters juga dimungkinkan untuk digunakan secara berurutan sekaligus atau chain.

```

1  var vm = new Vue({
2      el: '#app',
3      data: {
4          message: 'Hello',
5      },
6      filters: {
7          upper (text) {
8              return text.toUpperCase()
9          },
10         reverse (text) {
11             return text.split(' ').reverse().join(' ')
12         }
13     }
14 })

```

Pada template sebagai berikut.

```

1 <div id="app">
2     <h1> upper: {{ message | upper }} </h1>
3     <h1> reverse: {{ message | reverse }} </h1>
4     <h1> upper & reverse: {{ message | upper | reverse }} </h1>
5 </div>

```

Hasilnya:

upper: HELLO

reverse: olleH

upper & reverse: OLLEH

Pada contoh ketiga, tampilan variabel message di-filters oleh fungsi upper menghasilkan HELLO, lalu hasilnya tersebut kemudian di-filters oleh fungsi reverse menghasilkan OLLEH.

Deklarasi Filters Secara Terpisah ~4

Deklarasi filter juga bisa dipisah pada objek tersendiri

```

1  Vue.filter('upper', function (value) {
2      return value.toUpperCase()
3  })
4
5  var vm = new Vue({
6      // ...
7  })

```

Kesimpulan

Pada bab ini kita telah belajar tentang bagaimana konsep MVVM yang diterapkan oleh Vue dimana objek Vue bertindak sebagai ViewModel, properti el sebagai View, dan properti data sebagai Model. Objek Vue bertindak sebagai penghubung antara View dan Model.

Vue memiliki siklus hidup atau lifecycle mulai dari saat objek Vue dibuat (create), objek Vue dimuat (mount), objek Vue diupdate, hingga objek Vue dihapus (destroy), dimana masing-masing memiliki hook yang bisa kita manfaatkan untuk menjalankan perintah tertentu.

Penulisan template pada view tergantung dari bentuk data yang ingin kita tampilkan dan pada posisi mana ditampilkan. Apakah data dalam bentuk teks biasa atau html.

Properti methods digunakan untuk fungsi yang bisa dipanggil melalui suatu event, computed digunakan sebagai variabel bayangan yang nilainya bergantung pada variabel data, sedangkan filters digunakan untuk memanipulasi tampilan dari suatu teks.

Pada bab selanjutkan kamu akan diajak menyelami lebih dalam mengenai directive yang sebenarnya sudah disinggung pada bab ini. Beberapa varian directive serta bagaimana cara penggunaannya akan dibahas tuntas pada bab selanjutnya.

Jangan lupa senyum ya 😊

Directive

Intro

Sebenarnya kode directive telah kita gunakan pada bab sebelumnya, nah pada bab ini kita akan secara mendalam memahami tentang apa itu direktive.

Mengenal Directive

Directive merupakan atribut khusus yang disematkan pada elemen atau markup HTML sebagai penanda bahwa elemen DOM tersebut akan dikenai perlakuan tertentu oleh Vue. Directive berbentuk ekspresi Javascript yang secara reaktif menerapkan efek tertentu ke elemen DOM ketika nilai ekspresinya berubah. Penulisan atribut directive diawali dengan prefix `v-`, hal ini terinspirasi oleh prefix `ng-` pada Angular.

Pada bagian sebelumnya, kita juga sudah sedikit menyinggung tentang directive yaitu pada pembahasan tentang konsep MVVM serta pembahasan mengenai template.

v-html

Merupakan directive yang digunakan untuk menampilkan data berupa kode HTML

```
1 | <p v-html="message"></p>
```

v-once

Merupakan directive yang digunakan agar nilai variabel pada template tidak bisa diubah-ubah lagi.

```
1 | <p v-once>{{ message }}</p>
```

v-text

Merupakan directive yang digunakan untuk menampilkan string biasa, fungsinya sama dengan mustache atau double kurung kurawal.

```
1 | <p v-text="message"></p>
2 | <!-- sama dengan -->
3 | <p>{{ message }}</p>
```

v-show

Merupakan directive yang digunakan untuk menampilkan atau menyembunyikan suatu elemen DOM. Directive ini membutuhkan variabel bertipe boolean.

```
1 | <p v-show="displayMessage">{{ message }}</p>
```

Ketika variabel `displayMessage` bernilai true maka teks `message` akan terlihat di browser, sebaliknya jika jika variabel `displayMessage` bernilai false maka teks `message` tidak akan terlihat di browser. Proses on/off pada directive ini menggunakan properti `display` pada CSS. Artinya, apabila kita lakukan inspect element dengan menggunakan browser maka elemen tersebut tetap ter-render namun tidak terlihat di browser karena `display: none` menggunakan CSS.

Catatan: v-show tidak mendukung elemen `<template>`

v-if

Hampir sama dengan v-show, v-if merupakan directive yang digunakan untuk merender atau tidak merender suatu elemen DOM (conditional rendering).

```
1 | <h1 v-if="renderTitle">{{ title }}</h1>
```

Jika variabel `renderTitle` bernilai false maka teks variabel `title` tidak akan dirender sehingga jika kita lakukan inspect elemen, maka elemen tersebut memang benar-benar tidak ada dibrowser. Hal ini berbeda dengan directive sebelumnya, yang tampil dan tidaknya hanya melalui CSS.

Kita bisa menggunakan elemen template atau div atau apapun untuk blok konten

```
1 | <template v-if="content">
2 |   <h1>Title</h1>
3 |   <p>Paragraph 1</p>
4 |   <p>Paragraph 2</p>
5 | </template>
```

atau

```
1 | <div v-if="content">
2 |   <h1>Title</h1>
3 |   <p>Paragraph 1</p>
4 |   <p>Paragraph 2</p>
5 | </div>
```

Ketika menggunakan `v-if` maka kita juga bisa menggunakan `v-else` sebagai blok pengecualian

```
1 | <h1 v-if="renderTitle">{{ title }}</h1>
2 | <h1 v-else>Untitled</h1>
```

Lebih dari itu, Vue juga menyediakan `v-else-if` sebagai blok pengecualian bersyarat jika variabel yang diuji memiliki kemungkinan nilai lebih dari dua.

```
1 | <div id="app">
2 |   <div v-if="nilai === 'A'">
3 |     Sempurna
4 |   </div>
5 |   <div v-else-if="nilai === 'B'">
6 |     Bagus
7 |   </div>
8 |   <div v-else-if="nilai === 'C'">
9 |     Cukup
10 |    </div>
11 |    <div v-else>
12 |      Kurang
13 |    </div>
14 |  </div>
15 |
16 |  <script>
17 |    var vm = new Vue({
```

```

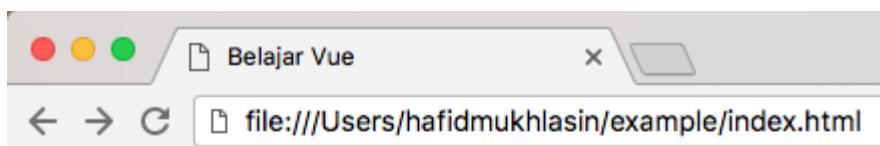
18     el: '#app',
19     data: {
20       nilai: "B",
21     },
22   })
23 </script>

```

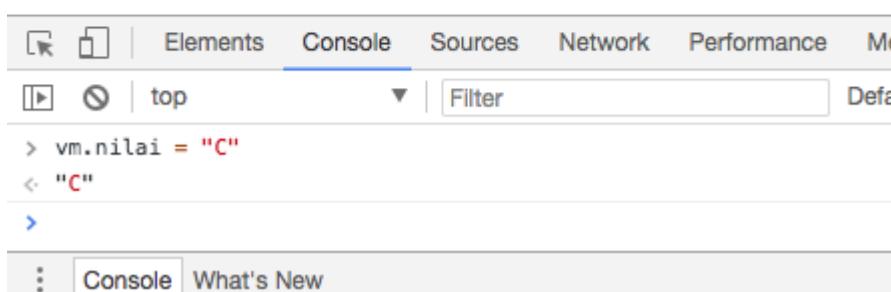


Bagus

Lebih dari itu, directive ini juga tetap memiliki sifat reactive. Sebagai contoh jika pada saat runtime, variabel nilai kita ubah menjadi "C" (`vm.nilai="C"`) maka View akan dirender ulang.



Cukup



v-on

Merupakan directive yang berperan sebagai sebuah event listener pada elemen HTML/komponen Vue. Directive ini bertugas memantau aktifitas (aksi) yang dilakukan terhadap suatu elemen HTML/komponen Vue. Contoh penggunaan.

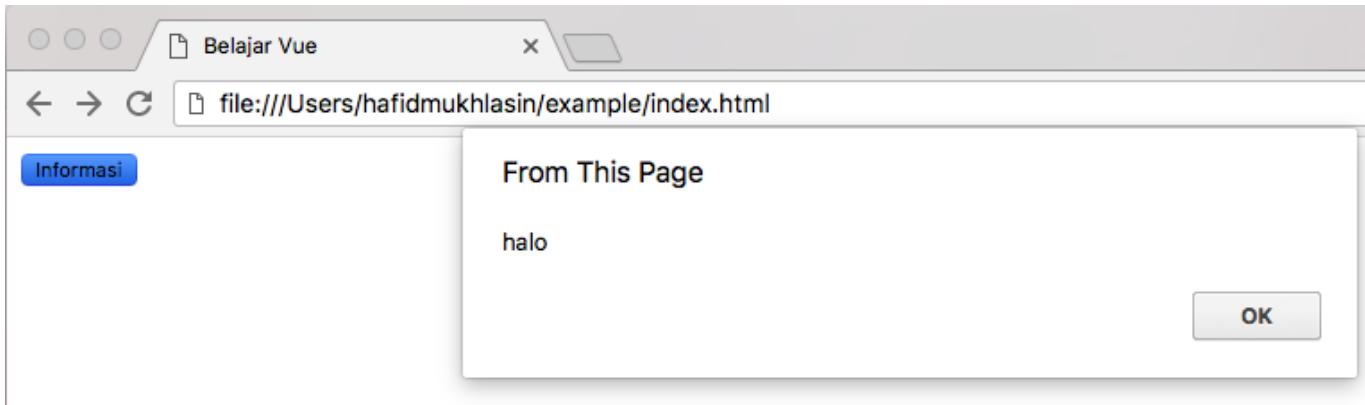
```

1 <button v-on:click="info('halo')">
2   Informasi
3 </button>

```

Catatam: `info()` adalah method yang harus kita deklarasikan dalam Vue, lihat pembahasan berikutnya.

Kode pada directive ini akan dijalankan ketika button `Informasi` diklik.



Directive v-on:click pada elemen HTML dikasus ini setara dengan event onclick native HTML biasa

```

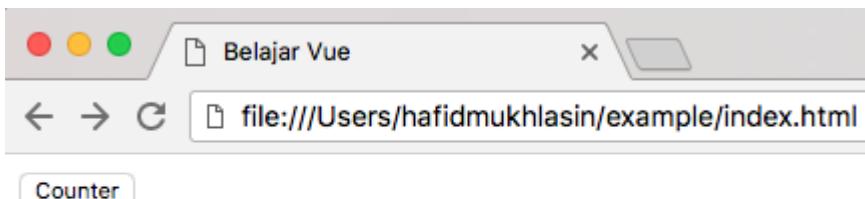
1 <button onclick="window.alert('halo')">
2   Informasi
3 </button>
```

Directive ini juga dapat kita manfaatkan untuk memanipulasi variabel data.

```

1 <div id="app">
2   <button v-on:click="counter += 1">
3     Counter
4   </button>
5   <p>Button di atas telah diklik sebanyak {{ counter }} kali.</p>
6 </div>
7
8 <script>
9 var example1 = new Vue({
10   el: '#app',
11   data: {
12     counter: 0
13   }
14 })
15 </script>
```

Ketika button diklik maka akan terjadi increment (penambahan 1) pada nilai dari variabel counter.



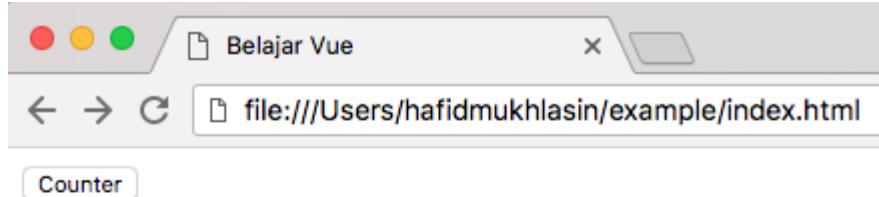
Button di atas telah diklik sebanyak 4 kali.

Hal ini tidak akan terjadi jika kita menggunakan event onclick native HTML.

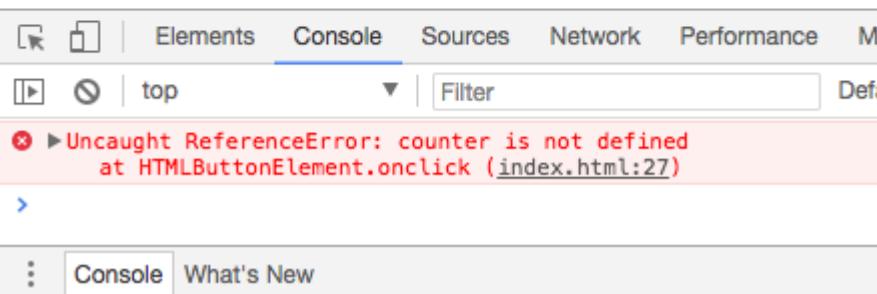
```

1 <button onclick="counter += 1">
2   Counter
3 </button>
```

Ketika button diklik maka akan muncul error, dimana variabel counter tidak dikenali.



Button di atas telah diklik sebanyak 0 kali.



Solusinya memang ada, yaitu menggunakan variabel vm

```

1 <button onclick="vm.counter += 1">
2   Counter
3 </button>
```

Namun, meskipun demikian, tetap disarankan menggunakan directive Vue.

Directive ini juga dapat digunakan untuk memanggil methods pada object Vue. Methods merupakan salah satu properti dalam objek Vue sebagaimana data dan el yang dapat berisi kumpulan fungsi yang digunakan pada aplikasi.

Berikut contoh pendefinisian dari methods.

```

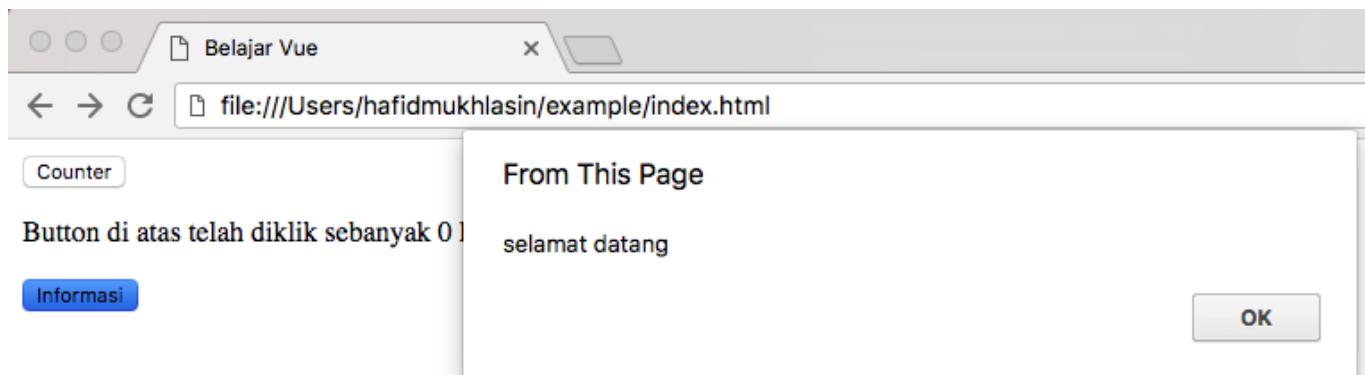
1 var vm = new Vue({
2   el: '#app',
3   data: {
4     counter: 0
5   },
6   methods: {
7     info (text) {
8       alert(text)
9     }
10   }
11 })
```

Untuk memanggil fungsi info, pada directive v-on, cukup dengan menuliskan nama fungsinya diikuti dengan parameter fungsinya. Berikut ini contoh implemetasinya pada button Informasi.

```

1 <button v-on:click="info('selamat datang')">
2   Informasi
3 </button>
```

Hasilnya, ketika button Informasi diklik maka Vue akan mengeksekusi fungsi `info` beserta parameter `selamat datang` dimana fungsi tersebut akan menampilkan alert yang berisi sesuai parameter yang dikirimkan.



Directive `v-on` juga dapat menangkap event native HTML melalui variabel `$event` yang dilewatkan sebagai parameter. Misalnya kita ingin mencegah event melakukan normal flow pada elemen anchor (link) HTML.

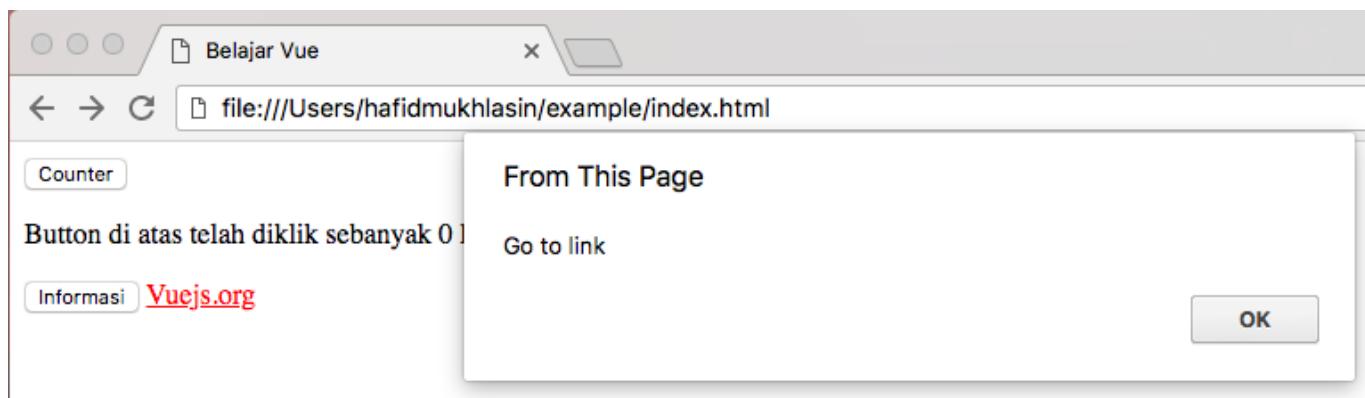
Berikut ini simulasinya.

```
1 <a href="http://vuejs.org" v-on:click="link()">
2   Vuejs.org
3 </a>
```

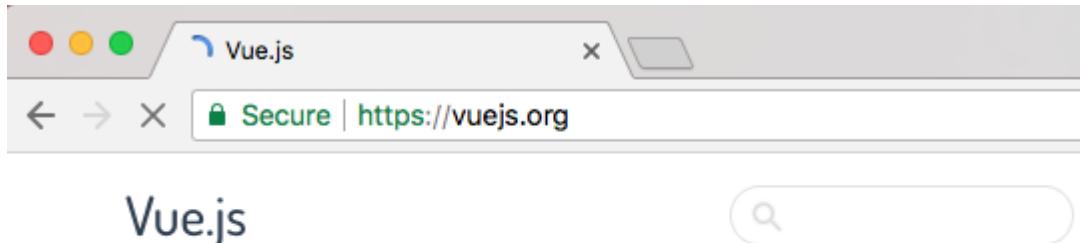
Tambahkan fungsi `link()` pada methods.

```
1 ...
2   methods: {
3     info (text) {
4       alert(text)
5     },
6     link () {
7       alert('Go to link')
8     }
9   }
10 ...
```

Hasilnya ketika link Vuejs.org diklik maka akan muncul alert.



Kemudian ketika button OK pada popup alert diklik maka halaman akan diredirect ke website Vuejs.



Connecting...

The Pro

Kita bisa mencegah redirect dengan menggunakan perintah `event.preventDefault()`, caranya, tambahkan parameter `$event` pada pemanggilan fungsi link di template.

```

1 <a href="http://vuejs.org" v-on:click="link($event)">
2   Vuejs.org
3 </a>
```

Kemudian gunakan event native HTML pada fungsi link di methods

```

1 ...
2   methods: {
3     ...
4       link (event) {
5         alert('Go to link')
6         event.preventDefault()
7     }
8   }
9 ...
```

Silahkan dicoba dan lihat hasilnya bahwa setelah alert muncul maka halaman tidak diredirect ke website Vuejs sesuai nilai yang tertera di atribut href.

Catatan: native event bisa kita gunakan untuk mengakses nilai dari atribut elemen HTML.

```
1 alert('Go to link '+ event.target.value)
```

Menariknya, Vue punya cara lain untuk mengatasi hal ini dengan cara yang lebih mudah yaitu melalui Event Modifier. Ada beberapa modifier untuk v-on, namun yang terkait dengan `preventDefault` adalah `.prevent`.

Berikut ini contoh implementasinya.

```

1 <a href="http://vuejs.org" v-on:click.prevent="info('Go to link')">
2   Vuejs.org
3 </a>
```

Dengan menambahkan modifier `.prevent` setelah directive `v-on:click` maka setelah alert muncul, halaman tidak akan di-redirect ke website Vue.

Disamping itu kita bisa juga membatasi agar misalnya suatu button atau link hanya boleh diklik sekali saja. Hal ini bisa kita lakukan dengan menggunakan modifier once.

```

1 <button v-on:click.once="info('selamat datang')">
2   Informasi
3 </button>
```

Pada contoh di atas, button Informasi hanya akan bereaksi ketika pertama kali diklik, kemudian button akan mengabaikan klik berikutnya.

Selain directive `v-on:click` ada beberapa directive lain yang bisa kita gunakan, diantaranya:

- `v-on:mouseover` ketika mouse berada di area elemen.
- `v-on:mouseenter` ketika mouse masuk ke area elemen.
- `v-on:mouseout` ketika mouse keluar dari area elemen.
- `v-on:mousedown` sama dengan `v-on:click`.
- `v-on:keyup` ketika keyboard up pada elemen (biasanya digunakan pada elemen input).
- `v-on:keydown` ketika keyboard down pada elemen (biasanya digunakan pada elemen input).
- `v-on:submit` ketika form di submit.

Demikian juga untuk modifiernya juga bermacam macam.

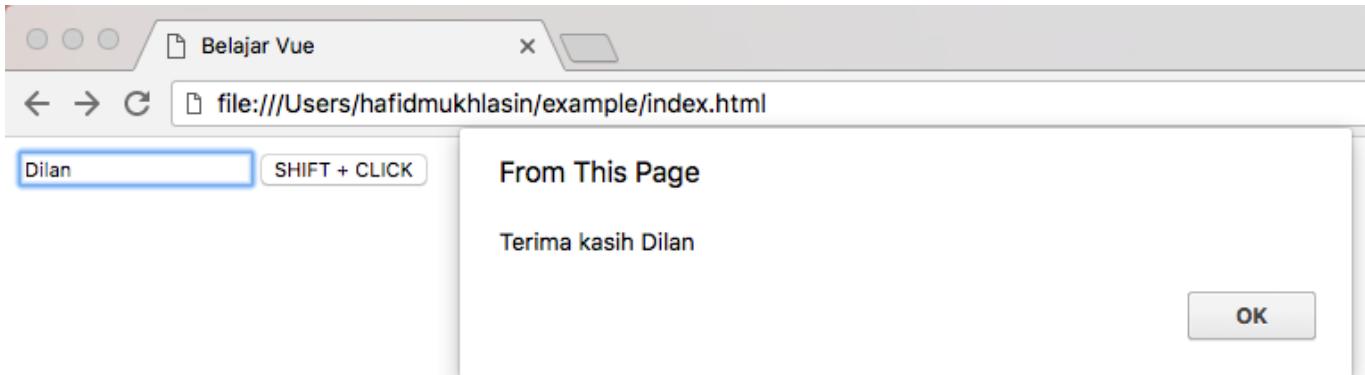
- `.enter` modifier ini akan bereaksi ketika keyboard `Enter` ditekan.
- `.tab` modifier ini akan bereaksi ketika keyboard `Tab` ditekan.
- `.delete` modifier ini akan bereaksi ketika keyboard `Delete` atau `Backspace` ditekan.
- `.esc` modifier ini akan bereaksi ketika keyboard `Escape` ditekan.
- `.space` modifier ini akan bereaksi ketika keyboard `Spasi` ditekan.
- `.native` modifier ini akan listen native event pada elemen root dari komponen.
- `.ctrl` modifier ini akan bereaksi ketika keyboard `Ctrl` ditekan.
- `.alt` modifier ini akan bereaksi ketika keyboard `Alt` ditekan.
- `.shift` modifier ini akan bereaksi ketika keyboard `Shift` ditekan.
- dsb.

Berikut ini contoh penggunaanya.

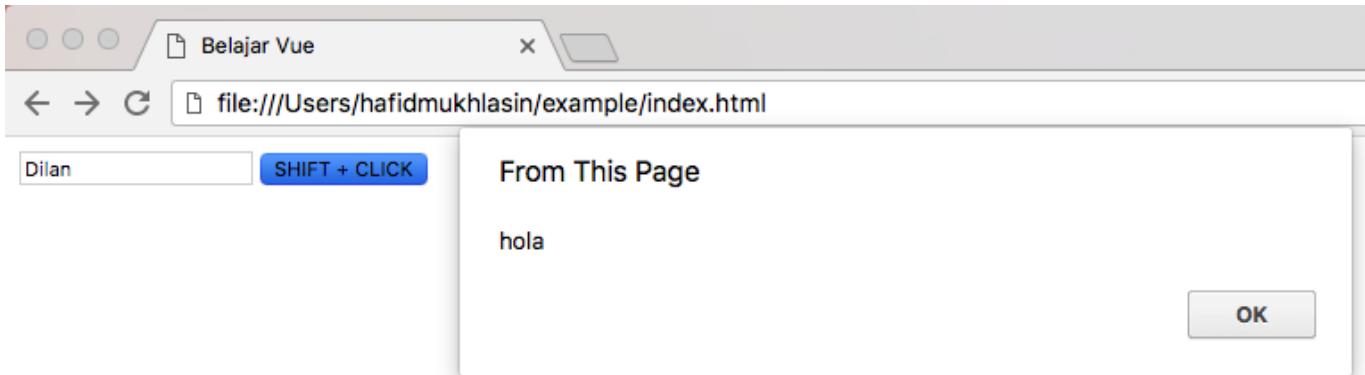
```

1 <!-- mencegah reload halaman saat event submit -->
2 <form v-on:submit.prevent="onSubmit"></form>
3
4 <!-- ketika ditekan enter maka akan menjalankan fungsi submit -->
5 <input v-on:keyup.enter="info('Terima kasih ' + $event.target.value)">
6
7 <!-- Shift + Click -->
8 <button v-on:click.ctrl="info('hola')">
9   SHIFT + CLICK
10 </button>
```

Mari kita lihat hasilnya, ketika fokus pada elemen input dan tombol keyboard `Enter` ditekan maka akan menampilkan alert sebagai berikut.



Demikian juga ketika button "SHIFT + CLICK" di klik berbarengan dengan tombol keyboard Shift maka juga akan muncul alert sebagai berikut.



Catatan: kita bisa menggabungkan beberapa directive dan modifiernya sekaligus dalam satu elemen dengan cara sebagai berikut.

```

1 <!-- object syntax (2.4.0+) -->
2 <button v-on="{ mousedown: doThis, mouseup: doThat }"></button>
3
4 <!-- chain modifiers -->
5 <button v-on:click.stop.prevent="doThis"></button>
```

Modifier .exact

Sejak versi 2.5.0+, Vue menambahkan modifier .exact untuk memastikan bahwa event hanya akan dijalankan ketika key tersebut saja yang diklik.

```

1 <!-- akan dijalankan ketika CTRL ditekan meskipun saat yang bersamaan Alt
2 atau Shift juga diklik -->
3 <button @click.ctrl="onClick">A</button>
4
5 <!-- hanya akan dijalankan ketika CTRL diklik dan tidak mengklik key lain
-->
6 <button @click.ctrl.exact="onCtrlClick">A</button>
```

Modifier Mouse Button

Nah kita juga bisa menangani klik pada mouse dengan modifier berikut: .left, .right, dan .middle.

```

1 <!-- akan dijalankan ketika pengguna mengklik kanan mousenya pada button
2 ini -->
```

```
<button @click.right="onRightClick">A</button>
```

Catatan: penulisan directive v-on: dapat disingkat menjadi @, contoh:

```
1 | <button @click="info('halo')">Info</button>
```

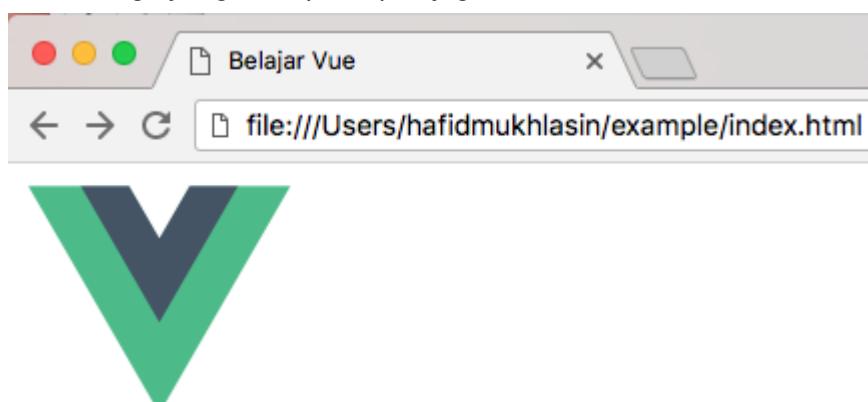
v-bind

Directive ini berfungsi untuk mem-binding atribut HTML atau komponen agar nilainya terupdate secara reactive sesuai dengan datanya, kebalikan dari v-on.

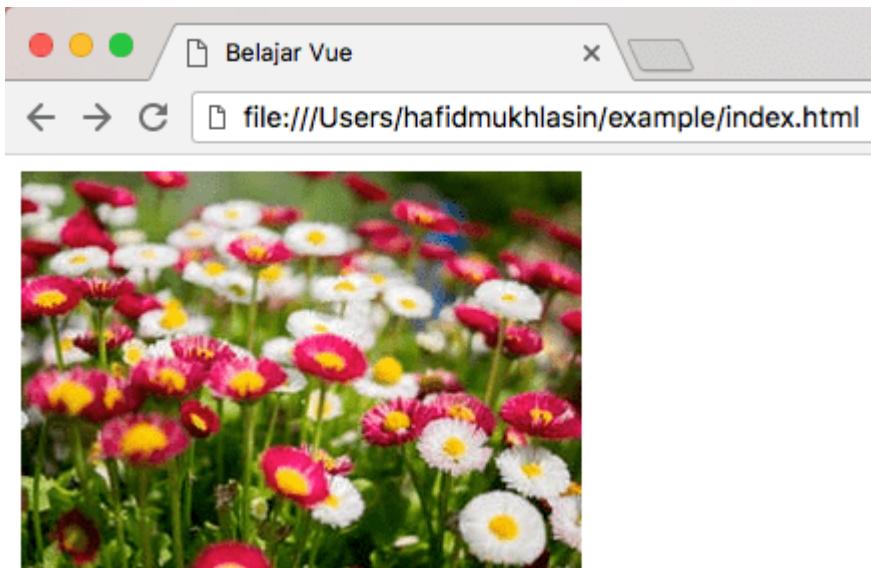
Contoh:

```
1 | <div id="app">
2 |   
3 | </div>
4 |
5 | <script>
6 | var vm = new Vue({
7 |   el: '#app',
8 |   data: {
9 |     imageSrc: 'logo-vue.png',
10 |   }
11 | })
12 |
13 | setTimeout(()=>{
14 |   vm.imageSrc = 'flowers.jpg'
15 | }, 3000);
16 | </script>
```

Pada contoh di atas, attribut src mem-binding variabel imageSrc, sehingga nilai dari attribut src tersebut mengikuti nilai dari variabel imageSrc. Demikian juga ketika nilai variabel imageSrc diganti secara runtime maka image yang ditampilkan pun juga akan berubah karena variabel imageSrc diubah.



setelah tiga detik maka gambar akan berubah



Tentu saja kita bisa gabungkan dengan teks statis. Misalnya:

```
1 
```

Kita juga bisa mem-binding satu elemen dengan dua atau lebih variabel data sekaligus, yaitu dengan menggunakan kurung siku []

```
1 <div v-bind:class="[classA, classB]"></div>
```

Boolean expression juga dimungkinkan

```
1 <div v-bind:class="{ red: isRed }"></div>
2 <div v-bind:style="{ fontSize: size + 'px' }"></div>
```

Sebagaimana directive `v-on`, penulisan directive `v-bind` juga bisa menggunakan object syntax misalnya untuk mem-binding beberapa atribut.

```
1 <img v-bind="{ id: imageID, src: imageSrc }" />
```

```
1 var vm = new Vue({
2   el: '#app',
3   data: {
4     imageID: 'image1',
5     imageSrc: 'logo-vue.png',
6   }
7 })
```

Catatan: penulisan directive `v-bind:` dapat disingkat menjadi `:`, contoh:

```
1 <a :href="url"> Website VueJS </a>
```

Dynamic Argument

Sejak versi 2.6.0, kita bisa menggunakan argumen dinamis pada sebuah directive. Contoh:

```
1 | <a v-bind:[nama_atribut]="url"> ... </a>
```

atau tentu saja kita tulis dengan

```
1 | <a :[nama_atribut]="url"> ... </a>
```

Nah pada contoh di atas, variabel nama_atribut akan secara dinamis dievaluasi sebagai JS expression, dan hasil dari evaluasi itu akan digunakan sebagai nilai akhir dari argumen direktif tersebut. Misalnya pada Vue kita set data nama_atribut bernilai "href", maka binding directive ini akan sama dengan kode v-bind:href.

Cara ini juga berlaku untuk directive event handler, contoh:

```
1 | <a v-on:[nama_event]="info('hallo')"> ... </a>
```

Jika variabel nama_event di set pada properti data bernilai "click" maka itu akan sama dengan kode v-on:click.

Kesimpulan

Pada bab ini kita telah belajar directive dan berbagai macam variannya yang akan berguna untuk memanipulasi tampilan dari aplikasi kita. Directive cukup powerfull untuk memantau perubahan data pada View dan memanipulasinya. Ada dua directive utama yang kita bahas yaitu v-on untuk event listener dan v-bind untuk binding DOM dan data.

Pada bab selanjutnya, kita masih akan bersinggungan dengan directive terutama yang berkaitan dengan data list (v-for), dan form (v-model).

Tetap semangat ya!

List

Intro

Data dalam bentuk list atau daftar yang bisa berupa array , objek atau collection (array dari objek) bisa kita tampilkan dengan mudah menggunakan Vue.

Menampilkan Data Array

Sebagai contoh, misalnya kita mempunyai data judul buku dalam bentuk array.

```

1  books : [
2      'C++ High Performance',
3      'Mastering Linux Security and Hardening', 'Python Programming
4      Blueprints',
5      'Mastering PostgreSQL 10'
6  ]

```

Atau jika kita masukkan dalam struktur data pada objek Vue, kira-kira seperti berikut.

```

1  var vm = new Vue({
2     el: '#app',
3     data: {
4         books : [
5             'C++ High Performance',
6             'Mastering Linux Security and Hardening',
7             'Python Programming Blueprints',
8             'Mastering PostgreSQL 10'
9         ]
10    }
11 })

```

Data tersebut ingin kita tampilkan menggunakan tag HTML list (li). Maka pada template Vue kita cukup mendefinisikannya sebagai berikut.

```

1  <div id="app">
2      <ul>
3          <li v-for="book in books">
4              {{ book }}
5          </li>
6      </ul>
7  </div>

```

Vue mempunyai directive `v-for` yang berfungsi untuk melakukan perulangan sebanyak elemen data yang ada pada variabel `books`. Sedangkan `book` (tanpa s) merupakan elemen (item satuan) dari array `books` yang bisa langsung ditampilkan tentunya dengan menggunakan `mustache` `{{ }}`

Mari kita lihat hasilnya.

The screenshot shows a browser window with the title "Belajar Vue". The address bar displays "file:///Users/hafidmukhlasin/example/index.html". The main content area shows a bulleted list of books:

- C++ High Performance
- Mastering Linux Security and Hardening
- Python Programming Blueprints
- Mastering PostgreSQL 10

Below the browser window is the Chrome DevTools interface, specifically the "Console" tab. The console output shows the word "top" followed by a blank line, indicating no recent console activity.

v-for Menggunakan Tag Template

Kode di atas bisa kita tulis dengan menggunakan tag template sebagai berikut.

```

1 <div id="app">
2   <ul>
3     <template v-for="book in books">
4       <li> {{ book }} </li>
5     </template>
6   </ul>
7 </div>
```

v-for Menggunakan Index

Index dari suatu array yang kita tampilkan melalui v-for bisa kita gunakan dengan menambahkan argumen kedua sebagai berikut

```

1 <li v-for="(book, index) in books">
2   {{ index+1 }}. {{ book }}
3 </li>
```

Parameter index didefinisikan untuk menampung key dari array books. Oleh karena index dari suatu array dimulai dari 0 maka kita juga bisa menggunakan ekspresi matematika untuk memanipulasinya supaya index dimulai dari angka 1. Berikut hasilnya.

The screenshot shows a browser window with the title "Belajar Vue". The address bar displays "file:///Users/hafidmukhlasin/example/index.html". The main content area shows a numbered list of books:

1. C++ High Performance
2. Mastering Linux Security and Hardening
3. Python Programming Blueprints
4. Mastering PostgreSQL 10

- 1. C++ High Performance
- 2. Mastering Linux Security and Hardening
- 3. Python Programming Blueprints
- 4. Mastering PostgreSQL 10

Selain `in`, kita bisa juga menggunakan delimiter `of` pada directive `v-for`.

```
1 <li v-for="(food, index) of foods">
```

Menampilkan Data Objek

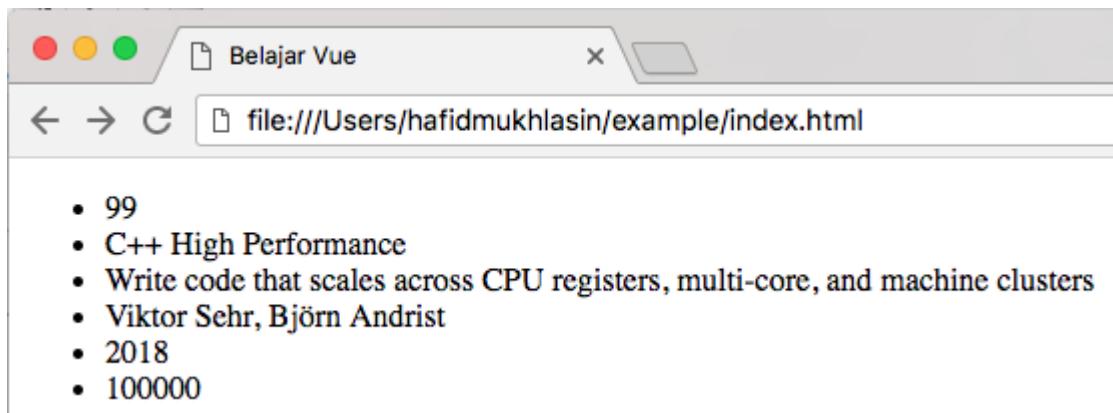
Sebagaimana array, data objek juga bisa kita tampilkan menggunakan directive `v-for`

```
1 book: {
2   id: 99,
3   title: 'C++ High Performance',
4   description: 'Write code that scales across CPU registers, multi-core,
5   and machine clusters',
6   authors: 'Viktor Sehr, Björn Andrist',
7   publish_year: 2018,
8   price: 100000,
9 }
```

Adapun kode untuk templatanya sebagai berikut.

```
1 <li v-for="value of book">
2   {{ value }}
3 </li>
```

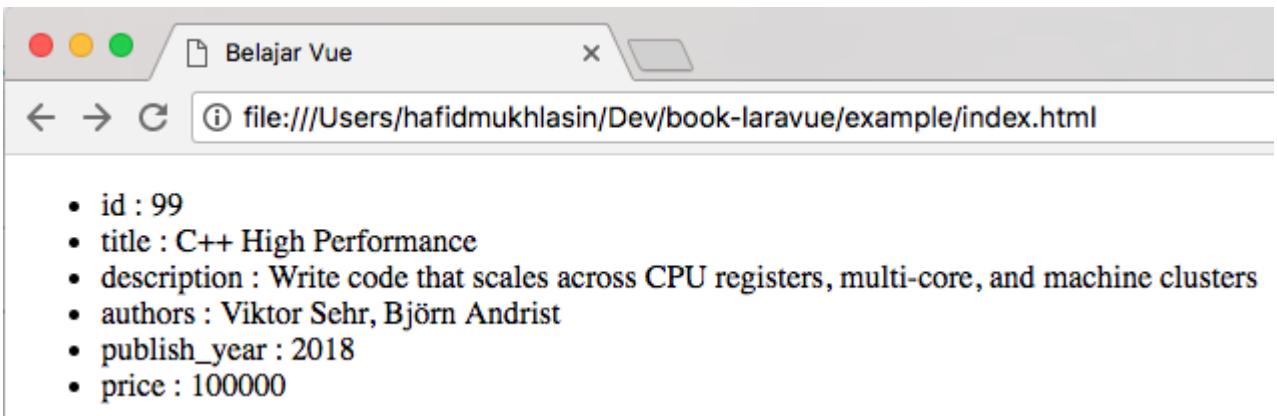
Berikut ini hasilnya.



Kita juga bisa menambahkan argumen kedua untuk key, seperti berikut.

```
1 <li v-for="(value, key) of book">
2   {{ key }} : {{ value }}
3 </li>
```

Berikut ini hasilnya.

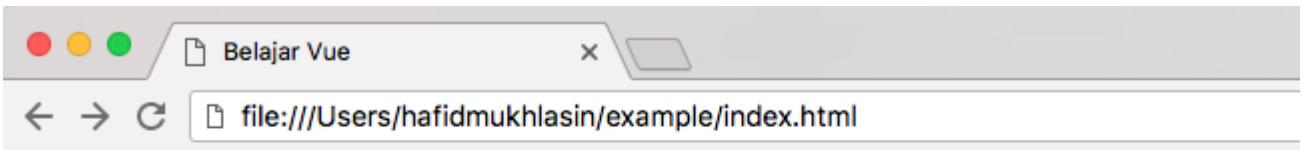


The screenshot shows a browser window titled "Belajar Vue". The address bar displays the URL "file:///Users/hafidmukhlasin/Dev/book-laravue/example/index.html". The page content is a list of book details:

- id : 99
- title : C++ High Performance
- description : Write code that scales across CPU registers, multi-core, and machine clusters
- authors : Viktor Sehr, Björn Andrist
- publish_year : 2018
- price : 100000

Adapun argumen ketiga yang bisa kita tambahkan akan menjadi index dari objek tersebut.

```
1 <li v-for="(value, key, index) of book">
2   {{ index+1 }}. {{ key }} : {{ value }}
3 </li>
```



The screenshot shows a browser window titled "Belajar Vue". The address bar displays the URL "file:///Users/hafidmukhlasin/example/index.html". The page content is a numbered list of book details:

- 1. id : 99
- 2. title : C++ High Performance
- 3. description : Write code that scales across CPU registers, multi-core, and machine clusters
- 4. authors : Viktor Sehr, Björn Andrist
- 5. publish_year : 2018
- 6. price : 100000

Menampilkan Data Collection

Pada kasus nyata, seringkali kita dapati data tidak dalam bentuk array sederhana ataupun objek, melainkan dalam bentuk yang lebih kompleks semisal array dari objek atau dalam format JSON (Javascript Object Notation).

Perhatikan contoh data list berikut.

```
1 books : [
2   {
3     id: 99,
4     title: 'C++ High Performance',
5     description: 'Write code that scales across CPU registers, multi-
6     core, and machine clusters',
7     authors: 'Viktor Sehr, Björn Andrist',
8     publish_year: 2018,
9     price: 100000,
10    image: 'c++-high-performance.png'
11  },
12  {
13    id: 100,
14    title: 'Mastering Linux Security and Hardening',
15    description: 'A comprehensive guide to mastering the art of
16    preventing your Linux system from getting compromised',
17    authors: 'Donald A. Tevault',
```

```

18     publish_year: 2018,
19     price: 125000,
20     image: 'mastering-linux-security-and-hardening.png'
21   },
22   {
23     id: 101,
24     title: 'Mastering PostgreSQL 10',
25     description: 'Master the capabilities of PostgreSQL 10 to
26 efficiently manage and maintain your database',
27     authors: 'Hans-Jürgen Schönig',
28     publish_year: 2016,
29     price: 90000,
30     image: 'mastering-postgresql-10.png'
31   },
32   {
33     id: 102,
34     title: 'Python Programming Blueprints',
35     description: 'How to build useful, real-world applications in the
36 Python programming language',
37     authors: 'Daniel Furtado, Marcus Pennington',
38     publish_year: 2017,
      price: 75000,
      image: 'python-programming-blueprints.png'
    },
]

```

Misalnya data tersebut ingin kita tampilkan dalam bentuk HTML tabel, maka melalui pendekatan yang sama dengan sebelumnya kita bisa menyusun templatenya sebagai berikut.

```

1 <div id="app">
2   <table border=1>
3     <tr v-for="book of books">
4       <td>
5         
6       </td>
7     </td>
8     <td>
9       title: {{ book.title }} <br>
10      description: {{ book.description }} <br>
11      authors: {{ book.authors }} <br>
12      price: {{ book.price }}
13     </td>
14   </tr>
15 </table>
16 </div>

```

Mari kita bedah satu persatu kode di atas.

Pertama, directive `v-for` kita letakkan di elemen `tr` pada `table` karena elemen itulah yang akan di-looping.

Pilihan lain kita bisa juga menggunakan elemen `<template>`

```

1 <template v-for="book of books">
2   <tr>

```

```

3 ...
4 </template>

```

Kedua, pada kolom pertama tabel ini kita akan tampilkan cover buku menggunakan kode ``. Supaya nilai dari atribut src dari elemen image menjadi dinamis sesuai dengan data books, maka (sebagaimana yang telah kita bahas pada bab terdahulu) kita perlu menambahkan directive `v-bind` pada atribut tersebut. `v-bind:src` atau disingkat menjadi `:src`.

Oleh karena variabel `book` yang dihasilkan dari perulangan variabel `books` berbentuk objek, maka kita bisa panggil setiap item didalamnya dengan menggunakan titik diikuti nama keynya.

```

1 book.title // judul buku
2 book.image // nama file cover buku

```

Karena lokasi cover buku pada tutorial ini ada dalam direktori `images/vue/books` maka kita bisa tambahkan definisi direktori tersebut pada atribut `src`.

Ketiga, sebagaimana poin kedua, pada kolom kedua dari tabel, bisa kita tampilkan detail bukunya.

```

1 <td>
2   title: {{ book.title }} <br>
3   description: {{ book.description }} <br>
4   authors: {{ book.authors }} <br>
5   price: {{ book.price }}
6 </td>

```

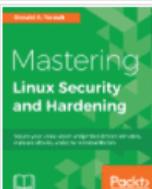
Boleh juga kita gunakan directive `v-for` lagi sebab variabel `book` berbentuk objek.

```

1 <td>
2   <template v-for="(value, key) of book">
3     {{ key }} : {{ value }} <br>
4   </template>
5 </td>

```

Berikut ini hasilnya.

Belajar Vue	
	file:///Users/hafidmukhlasin/example/index.html
	<p>id : 99 title : C++ High Performance description : Write code that scales across CPU registers, multi-core, and machine clusters authors : Viktor Sehr, Björn Andrist publish_year : 2018 price : 100000 image : c++-high-performance.png</p>
	<p>id : 100 title : Mastering Linux Security and Hardening description : A comprehensive guide to mastering the art of preventing your Linux system from getting compromised authors : Donald A. Tevault publish_year : 2018 price : 125000 image : mastering-linux-security-and-hardening.png</p>
	<p>id : 101 title : Mastering PostgreSQL 10 description : Master the capabilities of PostgreSQL 10 to efficiently manage and maintain your database authors : Hans-Jürgen Schönig publish_year : 2016 price : 90000 image : mastering-postgresql-10.png</p>
	<p>id : 102 title : Python Programming Blueprints description : How to build useful, real-world applications in the Python programming language authors : Daniel Furtado, Marcus Pennington publish_year : 2017 price : 75000 image : python-programming-blueprints.png</p>

Atribut Key

Terkait dengan metode menampilkan list, Vue menyarankan agar se bisa mungkin menggunakan atribut key pada tag HTML yang ikut dalam perulangan v-for. Key tersebut berperan sebagai penanda unik, sehingga Vue bisa melakukan tracking perubahan atas setiap tag HTML dari elemen list yang di-render.

Nilai atribut key sebenarnya bisa kita dapat darimana saja asal unik, misal:

- index dari array
- key atau properti dari objek

Berikut ini contohnya:

```

1 <li v-for="(book, index) of books" v-bind:key="index">
2   {{ index+1 }}. {{ book }}
3 </li>
```

Pada contoh di atas, nilai atribut key berasal dari argumen index. Kita perlu menambahkan directive v-bind untuk mem-binding nilai dari atribut key yang dinamis sesuai dengan hasil perulangan v-for.

Tips: penulisan dari `v-bind:key` bisa disingkat menjadi `:key` saja (shorthand).

Membatasi v-for menggunakan v-if

Adakalanya kita hanya ingin menampilkan data dengan kriteria tertentu saja, misalnya menampilkan data buku yang harganya lebih besar sama dengan 100 ribu.

Berdasarkan data varabel `books`, yang memenuhi kriteria tersebut hanya dua buku yaitu buku dengan judul "C++ High Performance" dan "Mastering Linux Security and Hardening".

Penerapannya di Vue sebagai berikut.

```

1 <ul>
2   <li v-for="(book, index) of books" :key="index" v-
3     if="book.price>=100000">
4       {{ book.title }}
5     </li>
</ul>

```

Kode v-if melakukan pengujian apakah harga buku lebih besar sama dengan 100000, jika benar maka data buku ditampilkan, demikian sebaliknya.



- C++ High Performance
- Mastering Linux Security and Hardening

Fungsi v-if pada v-for ini mirip dengan filter data, "kelemahan"-nya adalah index dari array jadi tidak berurutan.

Perubahan (mutation) Data Pada Array

Data pada contoh sebelumnya merupakan data statis yang tidak atau belum ada perubahan. Namun sebagaimana yang telah dijelaskan diawal bahwa data pada Vue bersifat observer artinya perubahannya senantiasa dipantau dan bisa menjadi pemicu untuk perubahan yang lain. Sebagai contoh, apabila data dihubungkan dengan template maka perubahan data akan menyebabkan perubahan pada tampilan atau DOM melalui rendering ulang.

Catatan: Kaidah pada Vue ini tetap mengikuti aturan main pada Javascript. Sebagai contoh, perubahan data berbentuk array harus menggunakan fungsi-fungsi yang tersedia di Javascript sehingga kita tidak bisa langsung mengeset data array secara langsung menggunakan index-nya.

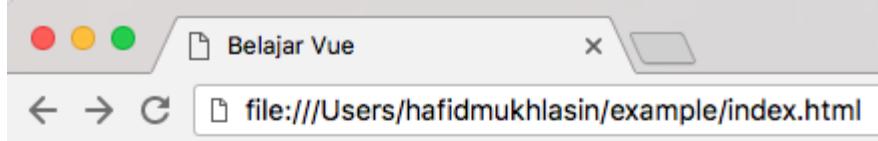
Asumsinya, kita menggunakan data array books.

```

1 books : [
2   'C++ High Performance',
3   'Mastering Linux Security and Hardening', 'Python Programming
4   Blueprints',
5   'Mastering PostgreSQL 10'
]

```

Kita akan coba melakukan manipulasi melalui console.



The screenshot shows the Chrome DevTools Console tab with the following interactions:

```

Elements Console Sources Network Performance M
▶ top Filter
> vm.books[4] = "Mastering PHP 7"
< "Mastering PHP 7"
> vm.books[1] = "Mastering Hacking"
< "Mastering Hacking"
>

```

Pada contoh di atas, meski kita menambahkan data elemen baru pada variabel books (Mastering PHP 7), namun tampilan daftar buku tetap tidak berubah atau tidak dirender ulang. Demikian juga perubahan data judul buku pada index pertama dari "Mastering Linux Security and Hardening" menjadi "Masterng Hacking" pun juga tidak bersifat observe atau tidak reaktif.

Untuk mengatasi hal ini, kita perlu menggunakan fungsi-fungsi built-in Javascript untuk memanipulasi data berbentuk array. Fungsi-fungsi tersebut yaitu push(), pop(), shift(), unshift(), sort(), reverse(), dan splice().

push() & pop()

Fungsi push digunakan untuk menambahkan data elemen baru pada suatu array pada posisi index terakhir. Sebaliknya fungsi pop untuk menghapus elemen terakhir dari suatu array. Misalnya:



The screenshot shows the Chrome DevTools Console tab with the following interaction:

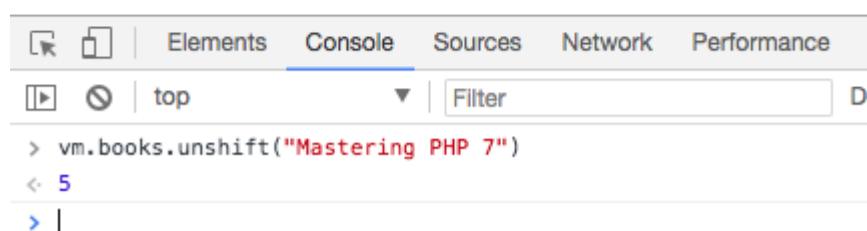
```

▶ top Filter
> vm.books.push("Mastering PHP 7")
< 5
>

```

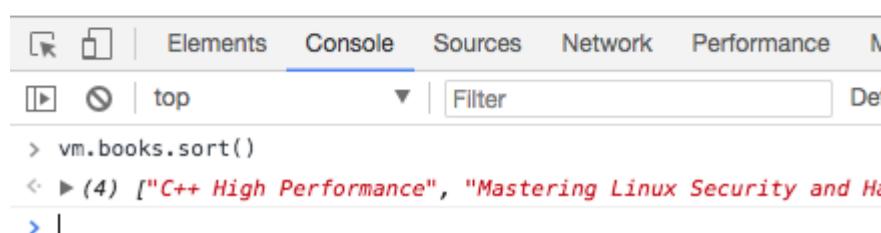
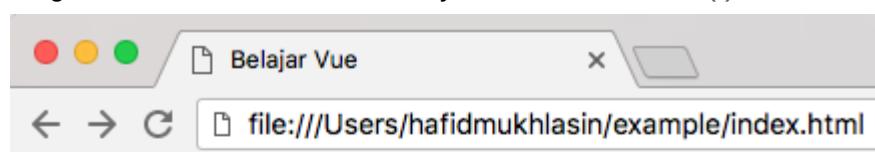
unshift() & shift()

Fungsi unshift digunakan untuk menambahkan data elemen baru pada suatu array pada posisi index pertama (0). Sebaliknya fungsi shift untuk menghapus elemen pertama dari suatu array. Misalnya: `vm.books.unshift('Mastering PHP 7')`



sort() & reverse()

Fungsi sort digunakan untuk mengurutkan data elemen pada suatu array secara ascending, sedangkan fungsi reverse melakukan sebaliknya. `vm.books.sort()`



splice()

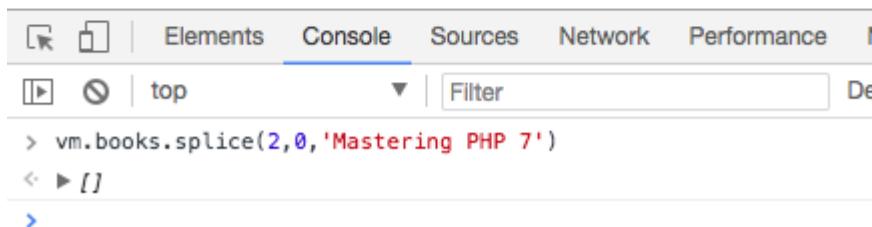
Fungsi splice ini multi fungsi, bisa digunakan untuk menambahkan data elemen baru pada suatu array. Misalnya: `vm.books.splice(2, 0, 'Mastering PHP 7')`

Perintah tersebut akan menambahkan data elemen baru yaitu 'Mastering PHP 7' pada index ke dua dari array.

- parameter pertama menujukkan posisi index dari data yang akan ditambahkan.
- adapun parameter kedua menunjukkan jumlah elemen pada array yang akan dihapus.



- C++ High Performance
- Mastering Linux Security and Hardening
- Mastering PHP 7
- Python Programming Blueprints
- Mastering PostgreSQL 10

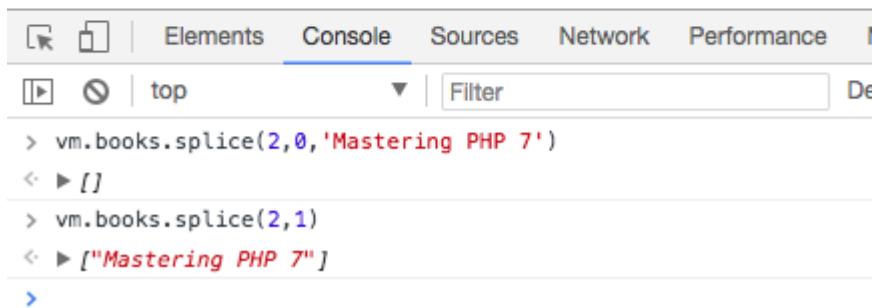


Fungsi splice juga bisa digunakan untuk menghapus elemen dari suatu array pada index tertentu. Misalnya: `vm.books.splice(2,1)`

Perintah tersebut akan menghapus elemen array pada index ke-dua sebanyak satu elemen.

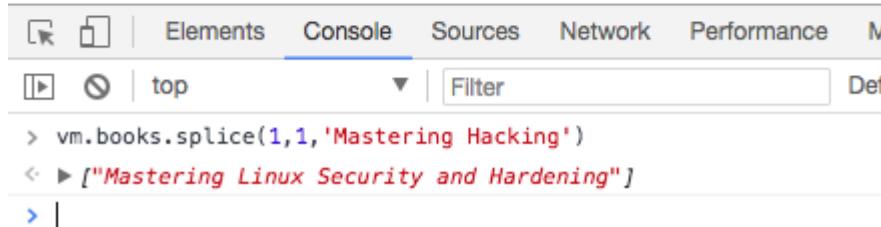


- C++ High Performance
- Mastering Linux Security and Hardening
- Python Programming Blueprints
- Mastering PostgreSQL 10



Perintah splice pun bisa digunakan untuk mengubah elemen dari suatu array pada index tertentu. Misalnya: `vm.books.splice(1,1,'Mastering Hacking')`

Perintah tersebut akan menghapus elemen array index ke-satu sekaligus menambahkan elemen baru pada index ke-satu juga. (ingat: index array dimulai dari 0)



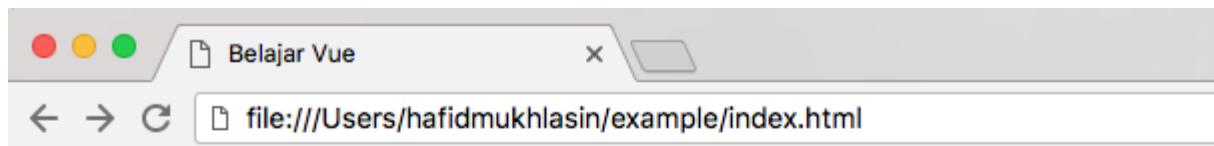
Disamping fungsi-fungsi di atas, terdapat beberapa fungsi built-in lagi terkait array yang bisa kita gunakan untuk memanipulasi elemen pada array namun sifatnya tidak melakukan perubahan langsung pada current array melainkan mengembalikan data array baru. Diantaranya: filter(), concat() dan slice().

Satu contoh, fungsi filter digunakan untuk mem-filter elemen dari array berdasarkan kriteria tertentu. Misalnya mengambil judul buku (dari books) yang diawali dengan huruf M.

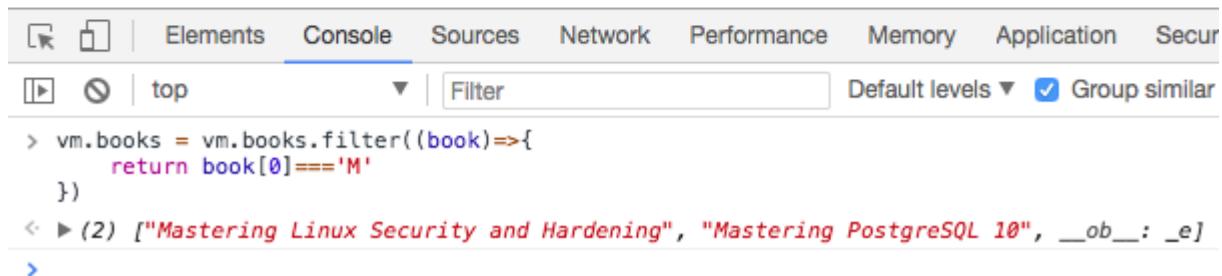
```

1  vm.books = vm.books.filter((book)=>{
2      return book[0]==='M'
3  })

```



- Mastering Linux Security and Hardening
- Mastering PostgreSQL 10



Implementasi pada Vue biasanya dengan menggunakan properti computed. Properti ini berupa fungsi-fungsi yang nilainya senantiasa dipantau atau observe sesuai dengan perubahan data.

Berikut ini contoh kodennya

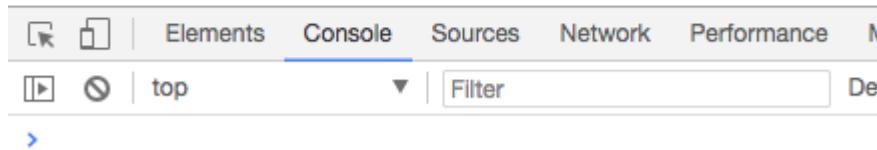
```

1 <div id="app">
2   <ul>
3     <li v-for="(book, index) of booksPrefixM" :key="index">
4       {{ book }}
5     </li>
6   </ul>
7 </div>
8
9 <script type="text/javascript">
10 var vm = new Vue({
11   el: '#app',
12   data: {
13     books : [
14       'C++ High Performance',
15       'Mastering Linux Security and Hardening', 'Python Programming
16 Blueprints',
17       'Mastering PostgreSQL 10'
18     ]
19   },
20   computed: {
21     booksPrefixM() {
22       return this.books.filter((book)=>{
23         return book[0]=='M'
24       })
25     }
26   }
27 })
</script>
```

Berikut ini hasilnya



- Mastering Linux Security and Hardening
- Mastering PostgreSQL 10



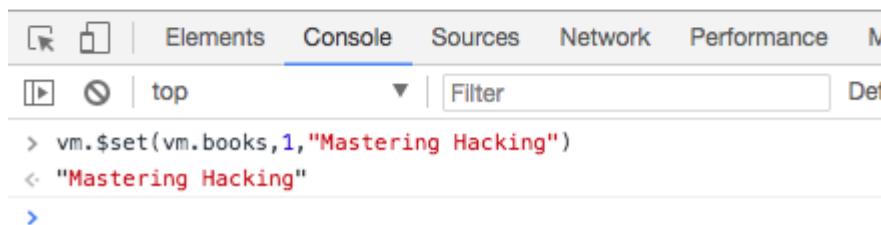
fungsi set pada Vue

Sebenarnya Vue juga menyediakan fungsi built-in untuk mengatasi masalah keterbatasan pada Javascript ini, jika pada contoh sebelumnya untuk mengubah data pada suatu elemen array menggunakan fungsi splice maka sebenarnya kita bisa juga melakukannya dengan fungsi set Vue.

```
Vue.set(data_array, index_yangakan_diganti, nilai_baru)
```



- C++ High Performance
- Mastering Hacking
- Python Programming Blueprints
- Mastering PostgreSQL 10



Perubahan Data Pada Objek

Seperti halnya array, objek pun juga perlu fungsi khusus untuk melakukan perubahan data.

Asumsi, kita gunakan data objek berikut.

```
1 book: {
2   id: 99,
3   title: 'C++ High Performance',
4   description: 'Write code that scales across CPU registers, multi-core,
5   and machine clusters',
6   authors: 'Viktor Sehr, Björn Andrist',
7   publish_year: 2018,
8   price: 100000,
 }
```

Untuk melakukan penambahan data properti pada objek, kita bisa gunakan fungsi set bawaan Vue.

```
1 | Vue.set(object, key, value)
```

atau

```
1 | vm.$set(object, key, value)
```

The screenshot shows a browser window with the title "Belajar Vue". The address bar displays "file:///Users/hafidmukhlasin/example/index.html". The content area lists the following book details:

- id: 99
- title: C++ Low Performance
- description: Write code that scales across CPU registers, multi-core, and machine clusters
- authors: Viktor Sehr, Björn Andrist
- publish_year: 2018
- price: 100000

The screenshot shows the Chrome DevTools Console tab. The console output shows:

```
> vm.$set(vm.book, 'title', 'C++ Low Performance')
<- "C++ Low Performance"
>
```

Namun, jika properti yang ingin kita tambahkan lebih dari satu maka kita bisa gunakan cara berikut.

```
1  vm.book = Object.assign({}, vm.book, {
2    umur: 27,
3    status: 'Perjaka'
4  })
```

The screenshot shows a browser window with the title "Belajar Vue". The address bar displays "file:///Users/hafidmukhlasin/example/index.html". The content area lists the following book details:

- id: 66
- title: C++ High Performance
- description: Write code that scales across CPU registers, multi-core, and machine clusters
- authors: Viktor Sehr, Björn Andrist
- publish_year: 2018
- price: 75000

The screenshot shows the Chrome DevTools Console tab. The console output shows:

```
> vm.book = Object.assign({}, vm.book, {
  'id': 66,
  'price': 75000
})
<- > {...}
>
```

Kesimpulan

Pada bab ini kita telah belajar bagaimana menampilkan data dalam bentuk list (array, object atau array of object) dengan berbagai macam bentuk variasinya. Menggunakan filter untuk menampilkan data tertentu saja. Tidak hanya itu, kita juga belajar bagaimana memanipulasi data dalam bentuk list supaya reaktif. List ini akan cukup banyak kita jumpai implementasinya dalam kasus nyata nanti.

Setelah belajar bagaimana cara menampilkan data maka pada bab selanjutnya kita akan belajar tentang bagaimana cara menangani input data dari user melalui form dan sejenisnya.

Keep on the track!

Form

Intro

Pada bab ini kita akan belajar tentang bagaimana menangani input data dari user melalui form serta bagaimana memanipulasi tampilan datanya.

Catatan: untuk latihan pada bab ini, save as file html yang berisi kode Vue kita menjadi form.html

Input Binding

Form HTML memiliki berbagai jenis field input seperti text, password, radio, dsb yang peruntukannya tentu berbeda tergantung dari data yang ingin ditangkap.

```

1 <input name="username" type="text">
2 <input name="password" type="password">
3 <input name="gender" type="radio">
```

Terkait dengan input binding ini, yang kita butuhkan adalah two way data binding, di mana nilai dari field input terhubung dengan data secara dua arah. Artinya perubahan field input yang dilakukan oleh user akan menyebabkan perubahan variabel data, sebaliknya perubahan variabel data akan menyebabkan perubahan pada field input.

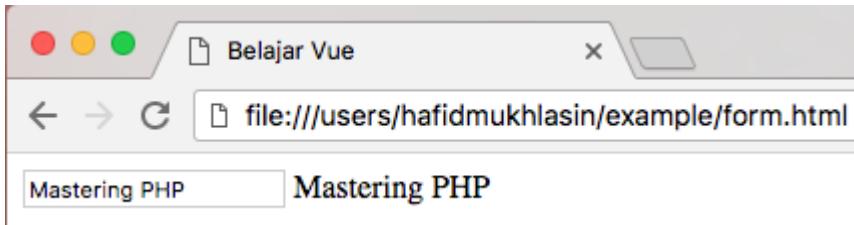
Berdasarkan, bahasan tentang directive, mungkin kita segera bisa menemukan triknya. Ya, yang kita butuhkan dua directive sekaligus v-on sebagai listener perubahan field input, dan v-bind yang bertugas mem-bind perubahan variabel data untuk diterapkan pada field input.

```

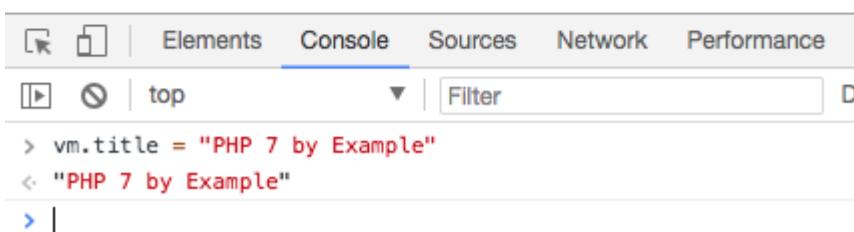
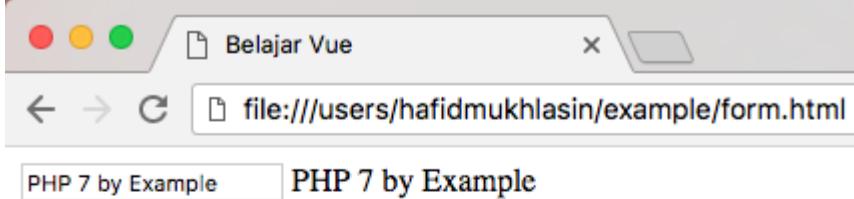
1 <div id="app">
2   <form>
3     <input type="text" name="title" :value="title" @input="title =
4       $event.target.value" />
5
6     {{ title }}
7   </form>
8 </div>
9
10 <script>
11 var vm = new Vue({
12   el: '#app',
13   data: {
14     title: "Mastering "
15   },
16 })
```

Atribut value di-bind dan event oninput di-listen. Kode :value="title" menunjukkan bahwa nilai dari field input ini di-bind dengan variabel title, sedangkan kode @input="title = \$event.target.value" artinya ketika field diinput maka nilai variabel title akan diubah sesuai isian user.

Berikut ini hasilnya.



Ketika field input diubah maka variabel title juga berubah.



Ketika variabel title diubah via console `vm.title = "PHP 7 by Example"` maka nilai dari field input juga akan mengikuti.

Catatan: metode ini akan dipakai ketika kita bermain dengan komponen.

Karenanya, untuk mengatasi "kerumitan" ini, Vue memperkenalkan directive `v-model` yang bertugas melakukan two way data binding tersebut. Berikut ini contohnya.

```
1 <form>
2   <input type="text" name="title" v-model="title" placeholder="masukkan
3   judul">
4     {{ title }}
5     <br><br>
6     <textarea name="description" v-model="description"
7     placeholder="masukkan deskripsi"></textarea>
      {{ description }}
</form>
```

Catatan: pastikan selalu menggunakan atribut `name` pada setiap field yang digunakan, disamping merupakan best practice juga akan berguna nanti pada bahasan selanjutnya. Nilai atribut `name` tidak harus sama dengan nilai dari atribut `v-model`, hanya saja karena keduanya identik maka sebaiknya disamakan saja.

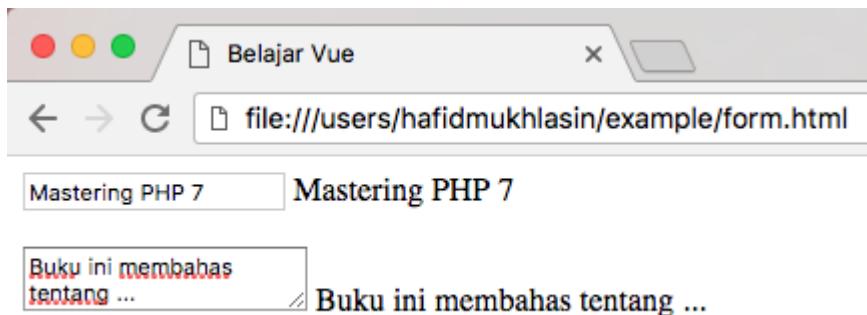
Tentu saja kita perlu tambahkan dua variabel yaitu `title` dan `description`.

```

1  data: {
2    title: '',
3    description: ''
4  },

```

Hasilnya sebagai berikut.



Text

Pada contoh di atas, elemen field input text, password dan textarea menerima data dalam bentuk teks atau string, sehingga tipe data pada variabel datanya adalah string juga.

Directive v-model juga memiliki modifier, diantaranya adalah `number` dan `trim`.

Modifier `number` digunakan untuk memastikan input data dari user bertipe numeric. Sedangkan modifier `trim` digunakan untuk menghapus spasi putih diawal atau akhir dari string.

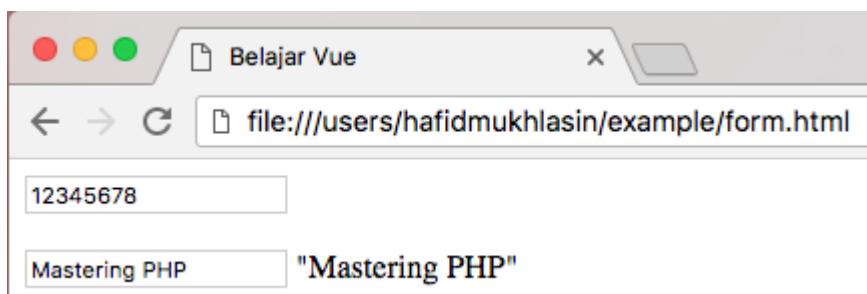
```

1  <input type="number" name="price" v-model.number="price">
2  <br><br>
3  <input type="text" name="title" v-model.trim="title"
   placeholder="masukkan judul"> "{{ title }}"

```

Catatan: tambahkan variabel price.

Mari kita lihat hasilnya.



Field input radion bertipe data teks.

```

1  Paket <br>
2  <input type="radio" name="bukuA" value="A" v-model="paket">
3  <label for="bukuA">Buku Laravel</label>
4  <br>
5  <input type="radio" name="bukuB" value="B" v-model="paket">
6  <label for="bukuB">Buku Vue</label>
7  <br>
8  <input type="radio" name="bukuAB" value="AB" v-model="paket">
9  <label for="bukuAB">Buku Laravel + Vue</label>

```

```

10 | <br>
11 | <span>Pilihan paket: {{ paket }}</span>

```

```

1 | data: {
2 |   paket: ''
3 |

```

Paket

- Buku Laravel
- Buku Vue
- Buku Laravel + Vue

Pilihan paket: AB

Field input select (single select) juga bertipe data teks. (tambahkan variabel category)

```

1 <select name="category" v-model="category">
2   <option disabled value="">Please select one</option>
3   <option>Graphics Programming</option>
4   <option>Mobile Application Development</option>
5   <option>Virtual and Augmented Reality</option>
6 </select>

```

Mobile Application Development

Catatan: Jika menggunakan field select, sebaiknya menambahkan elemen option disabled dengan nilai kosong sebagaimana contoh di atas untuk mengatasi masalah pada IOS.

Pada contoh di atas, atribut value pada elemen option tidak didefinisikan sehingga nilai dari variabel category mengikuti text diantara elemen option. Namun apabila value didefinisikan maka yang digunakan adalah value tersebut.

```

1 <select name="category" v-model="category">
2   <option disabled value="">Please select one</option>
3   <option value="01">Graphics Programming</option>
4   <option value="02">Mobile Application Development</option>
5   <option value="03">Virtual and Augmented Reality</option>
6 </select>
7 <span>Selected: {{ category }}</span>

```

Boolean

Input data bertipe data boolean (true atau false) biasanya terjadi pada field input checkbox tunggal. Contoh pada input data gender.

```
1 Gender :
2 <input type="checkbox" name="checkbox" v-model="gender">
3 <label for="checkbox">{{ gender }}</label>
```

```
1 data: {
2     gender: true
3 }
```

Field input select tunggal dengan dua option juga bisa bertipe boolean. Perhatikan contoh berikut.

```
1 <select name="gender" v-model="gender">
2     <option value="false">Wanita</option>
3     <option value="true">Pria</option>
4 </select>
5 <span>Selected: {{ gender }}</span>
```

Array

Input data bertipe array terjadi pada field input dengan kemungkinan pilihan lebih dari satu seperti checkbox multiple dan select multiple.

```
1 Hobi <br>
2 <input type="checkbox" name="hobby1" value="nonton" v-model="hobbies">
3 <label for="hobby1">Nonton</label>
4 <input type="checkbox" name="hobby2" value="jalan" v-model="hobbies">
5 <label for="hobby2">Jalan</label>
```

```

6 <input type="checkbox" name="hobby3" value="makan" v-model="hobbies">
7 <label for="hobby3">Makan</label>
8 <br>
9 <span>Pilihan hobi: {{ hobbies }}</span>
10
11 <hr>
12
13 <select v-model="categories" multiple>
14   <option value="01">Graphics Programming</option>
15   <option value="02">Mobile Application Development</option>
16   <option value="03">Virtual and Augmented Reality</option>
17 </select>
18 <span>Selected: {{ categories }}</span>

```

Adapun definisi dari variabel data harus sebagai array.

```

1 var vm = new Vue({
2   el: '#app',
3   data: {
4     hobbies: [],
5     categories: []
6   }
7 })

```



Tampilan yang berulang dengan pola tertentu seperti option pada field input select, tentu saja bisa kita generate menggunakan directive `v-for` sebagaimana yang telah dibahas pada bagian terdahulu.

```

1 data: {
2   categories: [],
3   options: [
4     { text: 'Graphics Programming', value: '01' },
5     { text: 'Mobile Application Development', value: '02' },
6     { text: 'Virtual and Augmented Reality', value: '03' }
7   ]
8 }

```

```

1 <select name="categories" v-model="categories" multiple>
2   <option v-for="option in options" :value="option.value">
3     {{ option.text }}

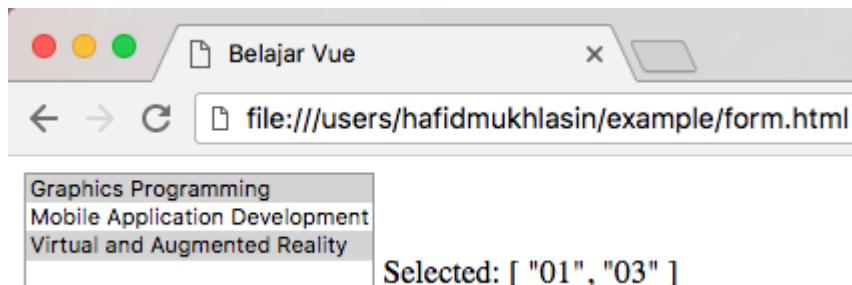
```

```

4 </option>
5 </select>
6 <span>Selected: {{ categories }}</span>

```

Atribut value dibind.



Filtering Data List

Sebelum kita gunakan form untuk submit data, ada satu materi terkait dengan list namun sengaja dibahas pada bab ini karena terkait dengan form input. Sederhana sekali sebenarnya karena secara umum konsepnya telah dibahas pada bab list tersebut.

Materi ini adalah membuat field pencarian atau filtering data buku dalam bentuk list. Pada template kita perlu satu filed input dan list.

```

1 <div id="app">
2   <input type="text" v-model="keyword" placeholder="ketikkan kata
3   kunci">
4   <ul>
5     <li v-for="(book, index) of filterBooks" :key="index">
6       {{ book }}
7     </li>
8   </ul>
9 </div>

```

Adapun kode Javascript-nya sebagai berikut.

```

1 var vm = new Vue({
2   el: '#app',
3   data: {
4     keyword: '',
5     books : [
6       'C++ High Performance',
7       'Mastering Linux Security and Hardening', 'Python Programming
8 Blueprints',
9       'Mastering PostgreSQL 10'
10      ],
11    },
12   computed: {
13     filterBooks() {
14       return this.books.filter((book)=>{
15         return book.includes(this.keyword)
16       })
17     }
18   }

```

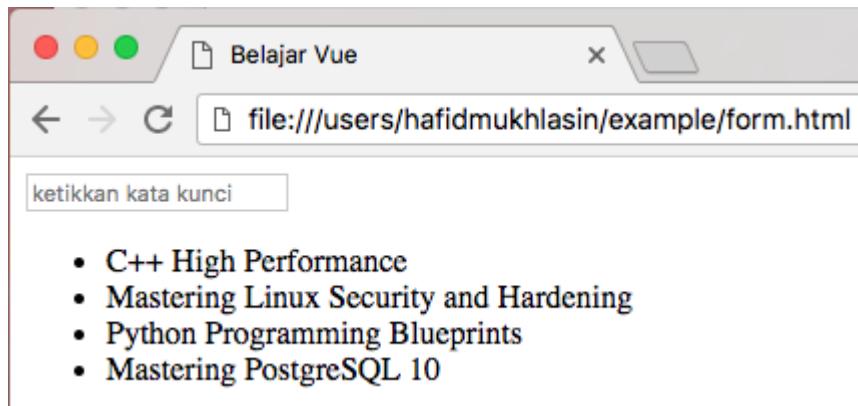
```

    }
})

```

Ada dua variabel yaitu keyword untuk menampung kata kunci dan books untuk menampung data buku. Disamping itu ada satu fungsi pada properti computed yaitu filterBooks. Fungsi filterBook akan melakukan pengecekan adakah item di variabel books yang berisi variabel keyword menggunakan method bawaan Javascript yaitu `includes`.

Hasilnya sebagai berikut.



Ketika diinput teks



Handling Submit Form & Validation

Pertanyaan: Apa fungsi elemen HTML form jika field bisa lewat begitu saja melalui v-model tanpa perlu submit?

Sebagaimana kita ketahui bahwa form memiliki event submit yang umumnya digunakan untuk mengirimkan data (yang berasal dari field input user) ke server atau ke bagian lain dari aplikasi.

Sudah menjadi konsensus bahwa sebuah form yang berisi beberapa field input data membutuhkan button submit yang menandai bahwa semua data yang diisi melalui field input tersebut siap dikirimkan.

Berikut ini contoh template form input data buku.

```

1  <form @submit="submitForm($event)" action="http://example.com/add-
2   product" method="post">
3     <label>Title:</label>
4     <input type="text" v-model="title" />
5
6     <label>Description:</label>
7     <textarea v-model="description"></textarea>
8
9     <label>Authors:</label>
10    <input type="text" v-model="authors">
11

```

```

12 <label>Price:</label>
13 <input type="number" v-model.number="price">
14
15 <label>Categories:</label>
16 <select v-model="categories" multiple>
17   <option v-for="option in options" :value="option.value">
18     {{ option.text }}
19   </option>
20 </select>
21
22 <label></label>
23 <input type="submit" value="Submit">
</form>

```

Pada elemen form `<form @submit="submitForm" action="http://example.com/add-product" method="post">`, kita menggunakan 3 atribut directive `v-on:submit` atau `@submit` (yang akan dijalankan ketika form disubmit), action (endpoint pengiriman data), dan method (metode pengiriman data apakah get atau post).

Jika kita menggunakan Vue untuk membuat aplikasi SPA (Single Page Application) maka dua atribut terakhir yaitu action dan method menjadi *unfaedah* 😊. Oleh karena itu, atribut sekaligus directive `@submit` yang kita jadikan tumpuan untuk dibahas pada bagian ini.

Directive ini pada contoh diatas memanggil fungsi `submitForm` yang mana pada fungsi ini kita bisa gunakan untuk misalnya: memvalidasi isian dari user, melakukan kalkulasi jika diperlukan, mengirimkan data isian tersebut ke server melalui http client, dsb.

```

1 var vm = new Vue({
2   el: '#app',
3   data: {
4     title: '',
5     description: '',
6     authors: '',
7     price: 0,
8     categories: [],
9     options: [
10       { text: 'Graphics Programming', value: '01' },
11       { text: 'Mobile Application Development', value: '02' },
12       { text: 'Virtual and Augmented Reality', value: '03' }
13     ]
14   },
15   methods: {
16     submitForm(event){
17       console.log(event)
18
19       // kode validasi
20
21       // kode kirim ke server
22
23       // kode status informasi
24       alert('Terima kasih')
25
26       // block redirect ke action
27       event.preventDefault()

```

```

28     }
29 }
30 })

```

Kode `event.preventDefault()` pada methods `submitForm` berfungsi untuk mencegah agar form tidak diredirect ke alamat yang didefinisikan di atribut `action`, hal ini dikarenakan pengiriman data tidak melalui cara normal HTML melainkan melalui Javascript (http client). Disamping cara ini, kita bisa juga menggunakan modifier `prevent` pada directive `@submit`

```

1 <form @submit.prevent="submit"...

```

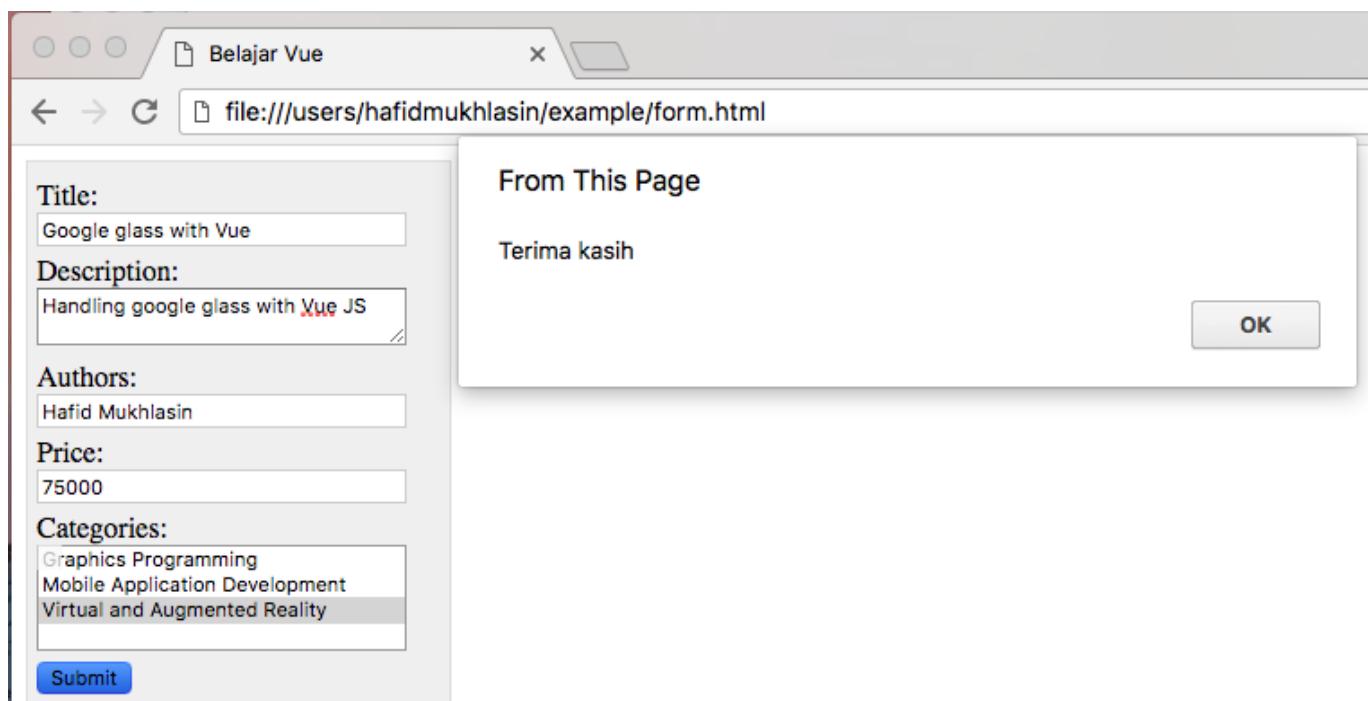
Supaya tampilan form lebih rapi, untuk sementara kita menggunakan style CSS berikut (letakkan pada elemen head HTML).

```

1 <style>
2   form {
3     border: 1px solid #ddd;
4     padding: 5px;
5     width: 225px;
6     background: #efefef;
7   }
8   label {
9     display: block;
10    margin-top: 5px;
11  }
12
13  input, textarea, select, option {
14    min-width: 200px;
15  }
16 </style>

```

Mari kita lihat hasilnya.



Selanjutnya, kita akan fokus pada bagian methods `submitForm()`.

Validasi Data

Proses validasi adalah proses memastikan setiap isian yang diinput oleh user melalui form tersebut sesuai dengan persyaratan minimal yang kita tentukan. Misalnya: title harus diisi dan minimal 3 karakter, description boleh tidak diisi namun kalau diisi maka tidak boleh lebih dari 500 karakter, price tidak boleh minus, dsb. Jika ditemukan terdapat isian yang tidak memenuhi syarat minimal maka akan dimunculkan pesan peringatan berupa alert serta dicatat sebagai error (counter). Pada bagian akhir kemudian dicek secara total apakah terdapat error (error lebih dari 0) atau tidak (error = 0), jika tidak ada error maka ditampilkan pesan terima kasih dan kode untuk mengirim data tersebut ke server.

```

1 submitForm(event){
2     let error = 0
3
4     if(this.title.length < 3){
5         error++
6         alert('Title minimal 3 karakter!')
7     }
8     else if(this.description.length > 500){
9         error++
10        alert('Description maximal 500 karakter!')
11    }
12    else if(this.authors.length < 3){
13        error++
14        alert('Authors minimal 3 karakter!')
15    }
16    else if(this.price < 0){
17        error++
18        alert('Price tidak boleh minus!')
19    }
20    else if(this.categories.length === 0){
21        error++
22        alert('Pilih minimal 1 category!')
23    }
24
25    if( error === 0 ){
26        alert('Terima kasih telah mengisi data dengan benar!')
27        // kirim data ke server
28    }
29
30    event.preventDefault()
31 }
```

Berikut hasilnya jika ada field yang tidak memenuhi minimal persyaratan

Title: Google glass with Vue

Description: Handling google glass with Vue JS

Authors:

Price: 75000

Categories:

- Graphics Programming
- Mobile Application Development
- Virtual and Augmented Reality

From This Page

Authors minimal 3 karakter!

OK

Dan berikut ini jika semua field telah memenuhi persyaratan.

Title: Google glass with Vue

Description: Handling google glass with Vue JS

Authors: Hafid Mukhlasin

Price: 75000

Categories:

- Graphics Programming
- Mobile Application Development
- Virtual and Augmented Reality

From This Page

Terima kasih telah mengisi data dengan benar!

OK

Kita bisa juga menambahkan method `setFocus` pada input yang belum memenuhi syarat. Sehingga akan memudahkan dari sisi user. Caranya dengan menambahkan directive `ref` sebagai penanda unik pada field input.

```
1 | <input type="text" name="title" ref="title" v-model="title">
```

Melalui directive tersebut kemudian bisa kita akses dengan kode berikut.

```
1 | if(this.title.length < 3){
2 |   error++
3 |   this.$refs.title.focus()
4 |   alert('Title minimal 3 karakter!')
5 | }
```

Bisa juga dengan kode ini `this.$refs.title.select()`.

Catatan: tambahkan juga directive `ref` pada form supaya lebih mudah nantinya menangani elemen form tersebut.

Supaya lebih user friendly, pesan `error` tidak kita munculkan dalam bentuk `alert`, melainkan dalam bentuk teks di browser. Kita bisa kombinasikan dengan teknik list untuk menampilkan `error` yang akan kita tampung dalam bentuk array.

Mari kita ubah kode konstruktor Vue menjadi sebagai berikut.

```

1  var vm = new Vue({
2    el: '#app',
3    data: {
4      title: 'Google Glass with VueJS',
5      description: 'Control Google Glass with VueJS',
6      authors: 'Hafid Mukhlasin',
7      price: 75000,
8      categories: [],
9      options: [
10        { text: 'Graphics Programming', value: '01' },
11        { text: 'Mobile Application Development', value: '02' },
12        { text: 'Virtual and Augmented Reality', value: '03' }
13      ],
14      errors: []
15    },
16    methods: {
17      submitForm(event){
18        this.errors = []
19        if(this.title.length < 3){
20          this.errors.push('Title minimal 3 karakter!')
21          this.$refs.title.select()
22        }
23        if(this.description.length > 500){
24          this.errors.push('Description maximal 500 karakter!')
25          this.$refs.description.select()
26        }
27        if(this.authors.length < 3){
28          this.errors.push('Authors minimal 3 karakter!')
29          this.$refs.authors.select()
30        }
31        if(this.price < 0){
32          this.errors.push('Price tidak boleh minus!')
33          this.$refs.price.select()
34        }
35        if(this.categories.length === 0){
36          this.errors.push('Pilih minimal 1 category!')
37          this.$refs.categories.focus()
38        }
39
40        if( this.errors.length === 0 ){
41          alert('Terima kasih telah mengisi data dengan benar!')
42          // kirim data ke server
43        }

```

```
44     }
45   }
46 })
```

Kemudian pada template, kita ubah menjadi sebagai berikut.

```
1 <form ref="formBook" @submit.prevent="submitForm($event)"
2   action="http://example.com/add-product" method="post">
3
4   <p v-if="errors.length">
5     <b>Please correct the following error(s):</b>
6     <ul>
7       <li v-for="error in errors">{{ error }}</li>
8     </ul>
9   </p>
10
11   <label>Title:</label>
12   <input name="title" ref="title" type="text" v-model="title">
13
14   <label>Description:</label>
15   <textarea name="description" ref="description" v-model="description">
16 </textarea>
17
18   <label>Authors:</label>
19   <input name="authors" ref="authors" type="text" v-model="authors">
20
21   <label>Price:</label>
22   <input name="price" ref="price" type="number" v-model.number="price">
23
24   <label>Categories:</label>
25   <select name="categories" ref="categories" v-model="categories"
26   multiple>
27     <option v-for="option in options" :value="option.value">
28       {{ option.text }}
29     </option>
30   </select>
31
32   <label></label>
33   <input type="submit" value="Submit">
34 </form>
```

Kode di atas akan menghasilkan tampilan berikut.

Please correct the following error(s):

- Authors minimal 3 karakter!
- Pilih minimal 1 category!

Title:
Google glass with Vue

Description:
Handling google glass with Vue JS

Authors:

Price:
75000

Categories:

Graphics Programming
Mobile Application Development
Virtual and Augmented Reality

Submit

Catatan: tutorial ini hanya menunjukkan tentang bagaimana validasi itu bekerja, selanjutnya tentu kamu bisa gunakan berbagai cara untuk memastikan bahwa input data user sesuai dengan yang diharapkan. Disamping cara manual ini, ada beberapa pustaka Vue yang bisa kita gunakan untuk memudahkan kita melakukan validasi data form, diantaranya:

- <https://github.com/monterail/vuelidate>
- <http://vee-validate.logaretm.com>

Peringatan: jangan lupa bahwa validasi ini hanya dari sisi client yang sangat rawan untuk dimanipulasi oleh user "nakal". Oleh karena itu validasi dari sisi server, merupakan hal mutlak yang harus kita lakukan untuk memastikan bahwa apa yang diinput oleh user itu sesuai dengan harapan kita.

Prepare Data Submit

Setelah validasi form dilakukan, langkah berikutnya adalah mempersiapkan data sebelum dikirimkan ke server. Kita bisa menggunakan object FormData (<https://developer.mozilla.org/en-US/docs/Web/API/FormData/FormData>) untuk mem-packing data hasil isian form menjadi sebuah object.

```

1 let formData = new FormData();
2 // tambahkan satu persatu field
3 formData.append('username', 'Chris');
```

Berikut ini implementasi pada kasus kita.

```

1 if( this.errors.length === 0 ){
2     alert('Terima kasih telah mengisi data dengan benar!')
3     // persiapkan data
4     let formData = new FormData()
5     formData.append('title', this.title)
6     formData.append('description', this.description)
7     formData.append('authors', this.authors)
8     formData.append('price', this.price)
9     formData.append('categories', this.categories)
10
11     // kirim data ke server
12 }
```

Objek formData inilah yang akan dikirimkan ke server atau diproses lebih lanjut.

Di samping cara di atas yaitu manual satu persatu dalam memasukkan data field, FormData juga menyediakan cara cepat untuk memasukkan semua data field yang dimiliki form ke dalam objek formData. Berikut yang kita dapat dari dokumentasi FormData.

```

1 let myForm = document.getElementById('myForm');
2 formData = new FormData(myForm);
```

Dengan sedikit modifikasi maka implementasi pada kasus kita menjadi sebagai berikut

```

1 // persiapkan data
2 let formBook = this.$refs.formBook
3 formData = new FormData(formBook);
4 // kirim data ke server
```

Selesai.

Catatan: untuk bisa menggunakan cara singkat ini syaratnya satu yaitu field pada form harus ada atribut **name**-nya.

Send Data To Server

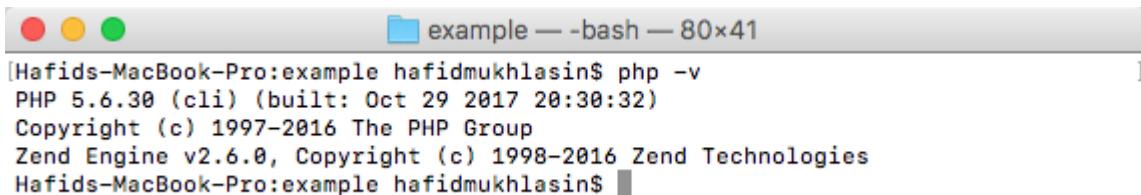
Setelah data di-*bundle* dalam satu objek formData, maka data siap untuk dikirim ke server. Pada bagian ini, kita akan mensimulasikan pengiriman data form ke server menggunakan PHP native. Oleh karenanya siapkan web server (nginx atau apache) serta PHP untuk dapat mencoba simulasi ini, atau bisa juga dengan menggunakan PHP built-in server.

Catatan: pada bab ini tidak akan dijelaskan tentang bagaimana instalasi PHP, dan Web Server. kamu bisa menggunakan panduan lain untuk menginstalasinya atau merujuk langsung ke website resmi PHP (<https://php.net>)

Pada tutorial ini, kita akan menggunakan built-in web server bawaan PHP. Oleh karena itu, pastikan PHP sudah terinstalasi dengan baik sehingga bisa diakses melalui terminal (command prompt atau powershell pada Windows).

Jalankan perintah

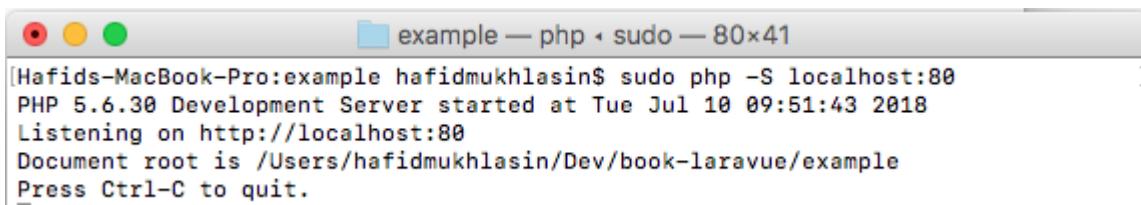
```
1 php -v
```



```
[Hafids-MacBook-Pro:example hafidmukhlasin$ php -v
PHP 5.6.30 (cli) (built: Oct 29 2017 20:30:32)
Copyright (c) 1997-2016 The PHP Group
Zend Engine v2.6.0, Copyright (c) 1998-2016 Zend Technologies
Hafids-MacBook-Pro:example hafidmukhlasin$ ]
```

Kemudian jalankan PHP built-in server dengan menggunakan format perintah berikut:

```
php -S localhost:80
```



```
[Hafids-MacBook-Pro:example hafidmukhlasin$ sudo php -S localhost:80
PHP 5.6.30 Development Server started at Tue Jul 10 09:51:43 2018
Listening on http://localhost:80
Document root is /Users/hafidmukhlasin/Dev/book-laravel/example
Press Ctrl-C to quit.
] ]
```

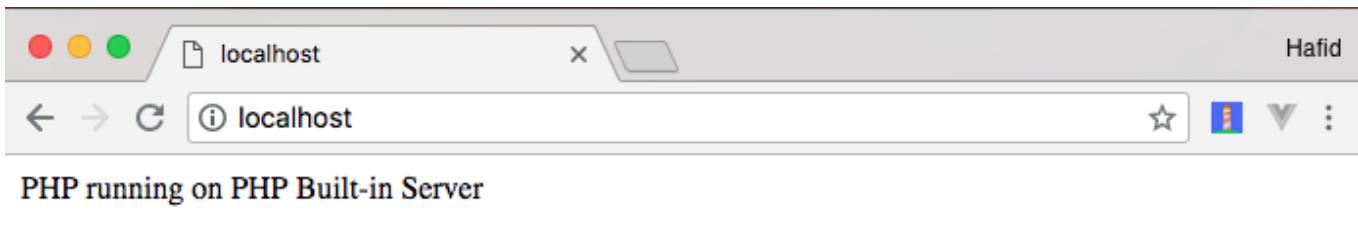
Catatan: 80 adalah nomer port dari webserver, kita bebas mengubahnya dengan nomer lain yang sedang tidak digunakan.

Perintah ini akan menjalankan web server dan menjadikan current directory sebagai web root.

Untuk menguji apakah kode PHP dapat berjalan dengan baik, maka buat file index.php pada current directory (pada contoh ini penulis meletakkan pada direktori yang sama dengan file form.html atau kode Vue). Isinya sebagai berikut.

```
1 <?php
2 echo "PHP running on PHP Built-in Server";
```

Jalankan pada browser alamat <http://localhost:80>, maka hasilnya.



Jika berhasil, maka selanjutnya kita akan membuat kode PHP yang bertugas sebagai *endpoint* penerima data kiriman dari form dan mengembalikan data tersebut dalam bentuk array. Berikut ini kodenya.

```
1 <?php
2 // untuk mencegah error akibat CORS
3 header("Access-Control-Allow-Origin: *");
4 header('Access-Control-Allow-Credentials: true');
5 header("Access-Control-Allow-Methods: GET, POST, OPTIONS");
6
7 echo "SIMULASI KIRIM DATA FORM <hr>";
8
9 // menampilkan data yang dikirimkan dengan method post
10 print_r($_POST);
```

Kemudian coba akses melalui browser.

SIMULASI KIRIM DATA FORM

Array ()

Catatan: fungsi CORS pada contoh ini digunakan agar Vue melalui pustaka HTTP client dapat mengakses file PHP yang diletakkan pada server yang berbeda dengan server dimana Vue di-hosting. Selengkapnya silakan baca di <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>

Untuk lebih mudahnya, pindahkan juga atau atur agar file-file Vue yang telah kita buat sebelumnya juga berada pada direktori yang sama dengan file index.php

Langkah selanjutnya adalah membuat kode Javascript/Vue untuk mengirimkan data ke server. Ada berbagai cara untuk melakukan request data ke server, diantaranya dengan engine XMLHTTP, fungsi fetch (fungsi native Javascript), dan pustaka axios.

Pada bagian ini, kita hanya akan membahas tentang cara request ke server dengan menggunakan XMLHTTP. Jadi XMLHTTP adalah engine yang digunakan untuk menangani permintaan data ke dan dari server. Engine ini cukup populer digunakan terutama untuk menjalankan AJAX.

```

1 if( this.errors.length === 0 ){
2     //alert('Terima kasih telah mengisi data dengan benar!')
3
4     // persiapkan data
5     let formBook = this.$refs.formBook
6     formData = new FormData(formBook);
7
8     // kirim data ke server
9     let xhttp = new XMLHttpRequest() // create objek XMLHttpRequest
10
11    // definisikan fungsi ketika terjadi perubahan state
12    xhttp.onreadystatechange = function() {
13        // state ini menunjukkan data terkirim dan diterima server dengan
14        baik
15        if (this.readyState == 4 && this.status == 200) {
16            // respon text dari server
17            console.log(this.responseText)
18        }
19    }
20    // sesuaikan dengan lokasi file index.php di lokasi komputer kamu
21    xhttp.open("POST", "http://localhost/index.php", true)
22
23    // bisa juga langsung nama filenya jika berada dalam satu folder yang
24    sama
25    // xhttp.open("POST", "index.php", true)
26
27    // kirim objek formData
28    xhttp.send(formData)
29 }
```

Mari kita ujicoba kode di atas pada browser.

Handling File Upload

Pada sisi client, penanganan field bertipe file pada form pada dasarnya hampir sama saja dengan field bertipe lain.

Sebagai simulasi, kita gunakan kode sebelumnya. Pada template form tambahkan field input bertipe file.

```
1 <label>Cover:</label>
2 <input name="cover" ref="cover" type="file">
```

Kemudian pada kode Javascript-nya tambahkan kode berikut

```
1 // get file yang dibrowse user
2 let cover = this.$refs.cover.files[0]
3 // tambahkan ke object formData
4 formData.append("cover", cover);
```

Dengan cara ini maka file akan terkirim ke server. Untuk mensimulasikannya, edit file `index.php` dan tambahkan kode berikut.

```
1 // ...
2 print_r($_FILES['cover']);
```

Hasilnya.

The screenshot shows a web browser window titled "Belajar Vue". The address bar displays "localhost/form.html". The page contains a form for adding a book entry. The fields are as follows:

- Title:** Google glass with Vue
- Description:** Handling google glass with Vue JS
- Authors:** Hafid Mukhlasin
- Price:** 75000
- Categories:** A dropdown menu with three options: Graphics Programming, Mobile Application Development (which is selected), and Virtual and Augmented Reality.
- Cover:** A file input field showing "Choose File flowers.jpg" and a "Submit" button.

Ketika form disubmit maka pada console log akan terlihat sebagai berikut.

The screenshot shows a web browser window with the title "Belajar Vue". The address bar shows "localhost/form.html". The page content is a form for adding a book. The form fields are:

- Title:** Google glass with Vue
- Description:** Handling google glass with Vue JS
- Authors:** Hafid Mukhlasin
- Price:** 75000
- Categories:** Graphics Programming, Mobile Application Development, Virtual and Augmented Reality
- Cover:** Choose File flowers.jpg

Below the form, the browser's developer tools are open, specifically the Console tab. The console output shows the submitted data as an array of objects:

```
SIMULASI KIRIM DATA FORM <hr>Array
(
    [title] => Google glass with Vue
    [description] => Handling google glass with Vue JS
    [authors] => Hafid Mukhlasin
    [price] => 75000
    [categories] => 02
)
Array
(
    [name] => flowers.jpg
    [type] => image/jpeg
    [tmp_name] => /private/var/tmp/phpRoRyzc
    [error] => 0
    [size] => 16119
)
```

Kesimpulan

Form merupakan media yang digunakan user untuk berinteraksi dengan aplikasi atau web. User dapat menginput data dalam bentuk teks, array, atau file. Validasi merupakan hal yang harus kita lakukan sebab kita tidak tau apakah user benar-benar memasukkan data sesuai dengan harapan kita atau tidak, sekaligus dalam rangka keamanan data.

Pengiriman data ke server menggunakan Javascript membutuhkan tools http client, pada bab ini dibahas mengenai penggunaan tools XMLHTTP yang biasa digunakan untuk AJAX. Pilihan lainnya, kita bisa gunakan Fetch (native Javascript) atau yang direkomendasikan Vue yaitu pustaka Axios.

Pada bab berikutnya, kita akan belajar tentang komponen yang merupakan bagian dari Vue yang cukup penting untuk dipelajari guna memudahkan kita dalam mengembangkan aplikasi yang kompleks.

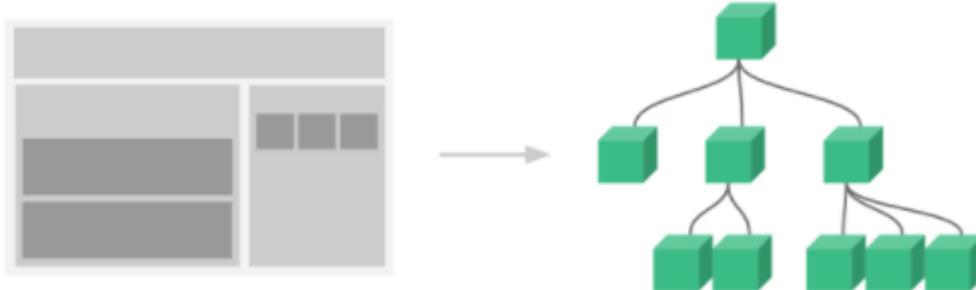
Ayoo lebih semangat lagi!

Component

Intro

Pada bab ini kita akan belajar tentang konsep *component* pada Vue dan penggunaannya pada aplikasi. Konsep *component* pada Vue mirip dengan konsep [Web Components](#).

Sebuah aplikasi Vue bisa dibangun dari beberapa component yang memiliki strukturnya hirarki



Sebagai contoh: sebuah aplikasi blog bisa terdiri dari tiga component utama yaitu header, content, dan footer. Pada component content memiliki dua sub component (child) yaitu sidebar dan main content. Component sidebar memiliki sub component lagi yaitu link navigasi, dst.

Catatan: untuk latihan pada bab ini, save as file html yang berisi kode Vue kamu menjadi component.html

Component Dasar

Component merupakan sub class/objek dari Vue yang bisa kita buat untuk berbagai tujuan misalnya memecah kompleksitas kode, reusabilitas kode, dan modularitas kode. Setiap component memiliki lifecycle yang sama dengan objek Vuenya.

Untuk membuat sebuah component baru, cukup dengan kode menjalankan method `Vue.component('nama-component', { /* options */ })`.

```

1  Vue.component('hello-world', {
2      data () {
3          return {
4              message: 'Hello world!'
5          }
6      },
7      template: '<h1> {{ message }}</h1>'
8  })
9
10 var vm = new Vue({
11     el: '#app',
12 })

```

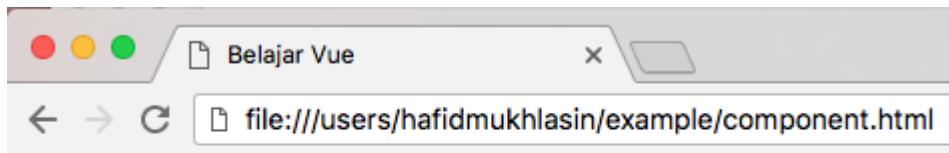
Pada contoh kode di atas, kita mencoba membuat sebuah component sederhana bernama `hello world`. Component tersebut dapat kita gunakan pada template utama Vue sebagai berikut.

```

1  <div id="app">
2      <hello-world></hello-world>
3  </div>

```

Apabila kita jalankan maka hasilnya sebagai berikut.



Hello world!

Catatan: tentu kamu boleh saja menggunakan web server supaya file component.html dapat diakses melalui protokol http.

Penamaan Component

Terdapat dua format penamaan component pada Vue

1. kebab-case

Format kebab-case artinya menggunakan huruf lowercase dan antar kata dipisah dengan tanda - (dash).

```
1 | Vue.component('my-component-name', { /* ... */ })
```

Ketika kita mendefinisikan component dengan menggunakan format kebab-case, maka komponen tersebut dapat diakses menggunakan custom element <my-component-name>.

2. PascalCase

Format PascalCase artinya menggunakan huruf capital pada huruf pertama setiap kata di mana antar kata tidak dipisah.

```
1 | Vue.component('MyComponentName', { /* ... */ })
```

Ketika kita mendefinisikan component dengan menggunakan format PascalCase, maka kita bisa memilih untuk mengakses komponen tersebut dengan cara <my-component-name> atau <MyComponentName>. Namun hanya elemen kebab-case yang dianggap valid oleh DOM.

Component Registration

Terdapat dua cara untuk meregister component pada Vue supaya bisa digunakan yaitu global dan local.

Global Component

Register component secara global akan membuat component tersebut bisa digunakan oleh semua objek utama (root) Vue, cara meregister component secara global sebagaimana contoh di atas yaitu menggunakan method

```
1 | Vue.component('my-component-name', { /* ... */ })
```

Contoh kita memiliki tiga component Global dan dua objek Vue.

```
1 | Vue.component('component-a', {
2   template: `<p>Component A</p>`
3 })
4 | Vue.component('component-b', {
5   template: `<p>Component B</p>`
```

```

6  })
7  Vue.component('component-c', {
8    template: `<p>Component C</p>`
9  )
10
11 new Vue({ el: '#app1' })
12 new Vue({ el: '#app2' })

```

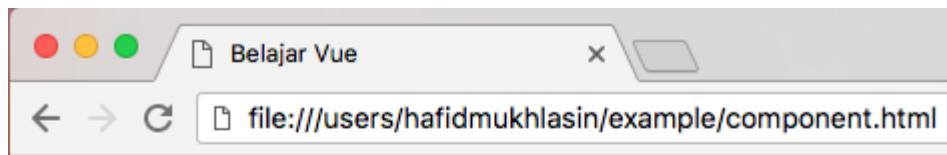
Implementasi pada template

```

1 <div id="app1">
2   <h1>Objek Vue 1</h1>
3   <component-a></component-a>
4   <component-b></component-b>
5 </div>
6
7 <div id="app2">
8   <h1>Objek Vue 2</h1>
9   <component-a></component-a>
10  <component-c></component-c>
11 </div>

```

Hasilnya



Objek Vue 1

Component A

Component B

Objek Vue 2

Component A

Component C

Local Component

Register component secara local artinya component tersebut diregister pada suatu objek Vue dan hanya bisa digunakan pada objek tersebut saja.

Untuk mendefinisikan component local cukup dengan menggunakan object Javascript biasa

```

1 var ComponentA = {
2   template: `<p>Component A</p>`
3 }
4

```

```

5  var ComponentB = {
6    template: `<p>Component B</p>`
7  }
8
9  var ComponentC = {
10   template: `<p>Component C</p>`
11 }

```

Adapun untuk meregister pada objek vue menggunakan properti `components`.

```

1 new Vue({
2   el: '#app',
3   components: {
4     'component-a': ComponentA,
5     'component-b': ComponentB,
6     'component-c': ComponentC
7   }
8 })

```

Pada template

```

1 <div id="app">
2   <component-a></component-a>
3   <component-b></component-b>
4   <component-c></component-c>
5 </div>

```

Berikut ini hasilnya.



Component A

Component B

Component C

Sebaliknya jika kita register suatu component ke objek Vue pertama saja sedangkan objek Vue kedua tidak.

```

1 var ComponentA = {
2   template: `<p>Component A</p>`
3 }
4
5 new Vue({
6   el: '#app1',
7   components: {
8     'component-a': ComponentA,
9   }
10 })
11

```

```

12 new Vue({
13   el: '#app2'
14 })

```

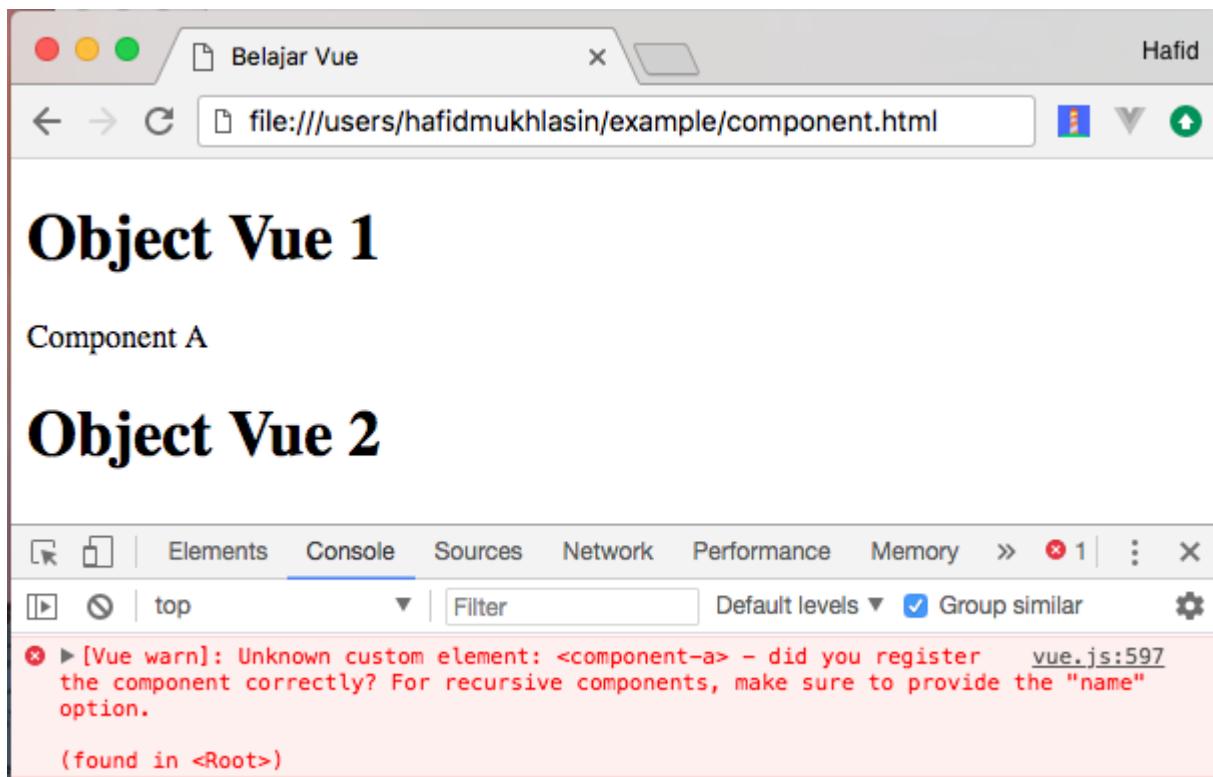
Sedangkan pada template objek Vue kedua juga menggunakan component tersebut

```

1 <div id="app1">
2   <h1>Object Vue 1</h1>
3   <component-a></component-a>
4 </div>
5
6 <div id="app2">
7   <h1>Object Vue 2</h1>
8   <component-a></component-a>
9 </div>

```

Maka hasilnya akan muncul error seperti berikut ini



Deklarasi Properti Data

Strukturnya components secara umum hampir sama dengan objek utama Vue, salah satu yang membedakan adalah definisi properti data pada component tidak lagi sebagai object melainkan sebagai sebuah fungsi.

```

1 data: {
2   message: 'Hello world'
3 }
4
5 // menjadi
6 data () {
7   return {
8     message: 'Hello world!'
9   }

```

```
10 |     }
  }
```

Tujuannya supaya data pada masing-masing component bisa berdiri sendiri, jika tidak maka ketika terjadi perubahan pada suatu objek dari component X, maka data pada objek lain pada component X juga akan berubah.

Reusable Component

Component ini kemudian dapat digunakan berkali-kali atau reusable.

```
1 <div id="app">
2   <hello-world></hello-world>
3   <hello-world></hello-world>
4   <hello-world></hello-world>
5 </div>
```

Component Lanjutan

Kirim Data ke Component

Vue mempunyai mekanisme untuk mengirimkan atau mengeset suatu data pada component yaitu dengan menggunakan properti `props`

Untuk mendefinisikan props, pada component caranya cukup dengan menambahkan properti `props`.

```
1 props: [ 'nama_props1', 'nama_props2' ],
```

Kemudian pada template dari component tersebut kita bisa akses props tersebut layaknya properti data.

```
1 template : `<div> {{ nama_props1 }} </div>`
```

Sebagai contoh, kita mempunyai component book, dengan tiga props yaitu title, description dan image.

```
1 var BookComponent = {
2   data () {
3     return {
4       classCard: 'card'
5     }
6   },
7   props: [ 'title', 'description', 'image' ],
8   template :
9     <div :class="classCard">
10      <h3>{{ title }}</h3>
11      
12      <p v-html="description"></p>
13    </div>
14  ,
15}
```

Catatan: template harus memiliki elemen root tunggal (wrapper), pada contoh di atas, rootnya adalah elemen `<div>`

Seperti biasa, kita perlu daftarkan component book pada objek utama Vue.

```

1 new Vue({
2   el: '#app',
3   components: {
4     'book': BookComponent,
5   }
6 })

```

Pada bagian template utama, kita bisa deklarasikan component book dan sekaligus mendefinisikan nilai dari masing-masing props, sebagai berikut.

```

1 <div id="app">
2   <book
3     title="C++ High Performance"
4     description="Write code that scales across CPU registers, multi-
5     core, and machine clusters"
6     image = "c++-high-performance.png"
7   >
8   </book>
</div>

```

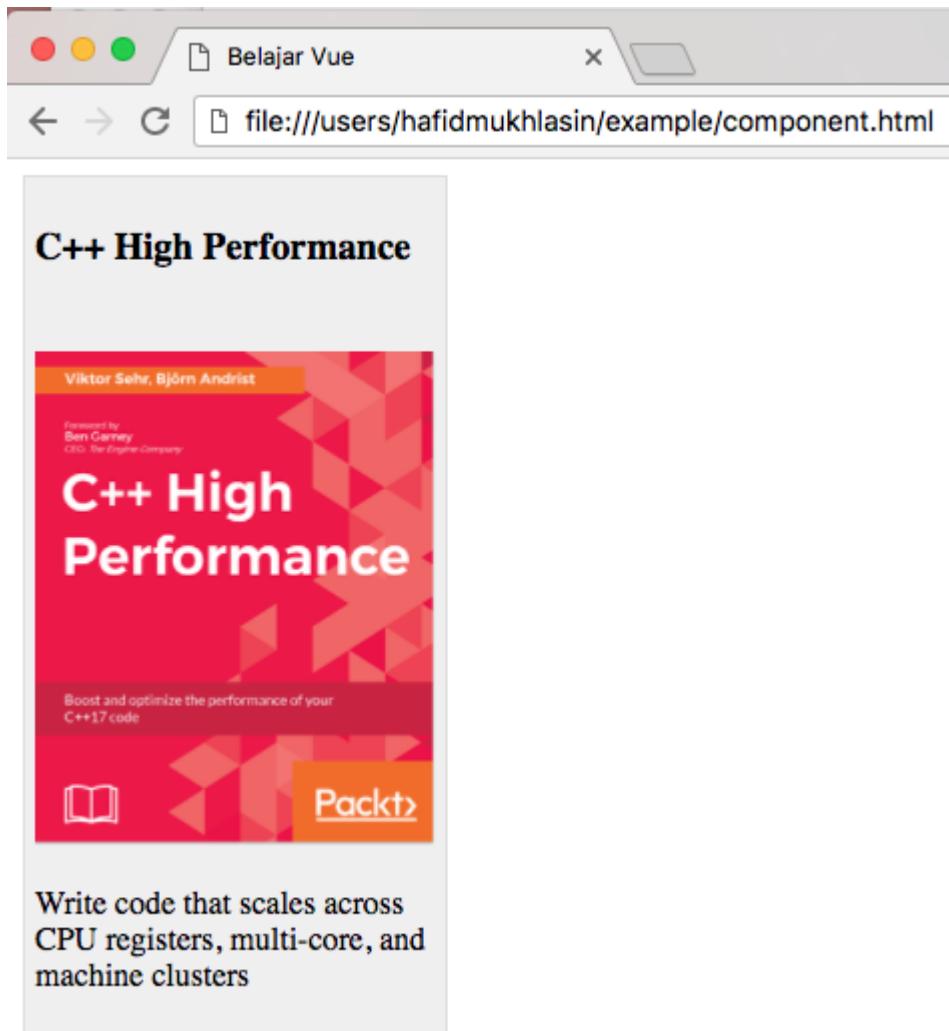
Untuk merapikan tampilan, untuk sementara kita akan buat style CSS sederhana (letakkan pada elemen style pada head)

```

1 .card {
2   background: #efefef;
3   border: 1px solid #ddd;
4   margin-right: 5px;
5   padding: 5px;
6   width: 200px;
7   float: left;
8 }
9
10 h3{
11   min-height: 45px;
12 }

```

Hasilnya.



Nilai dari variable title, description, dan image dikirimkan atau dilewatkan dari luar component akan muncul atau dapat diakses dari dalam component layaknya properti data.

Directive Pada Component

Component pada Vue bisa diibaratkan sebagai elemen HTML di mana ada atribut berupa directive yang bisa diaplikasikan padanya. Sebagai contoh, kita akan menampilkan data dalam bentuk list menggunakan elemen berupa component. Adapun directive yang kita gunakan adalah v-for dan v-bind atau :.

Pada bagian sebelumnya, pendefinisian nilai props dari component dilakukan secara hardcode langsung pada componentnya,

```
1 | <book title='judul buku'></book>
```

Nah dengan konsep ini, maka kita bisa mendefinisikan nilai props title secara lebih dinamis menggunakan properti data.

```
1 | <book :title='judul'></book>
```

Props title kita bind ke variabel `judul` (nama variabelnya bisa bebas saja), cukup dengan menambahkan directive v-bind atau :.

Nah kemudian, pada objek utama Vue kita perlu tambahkan data `judul` sebagai berikut:

```
1 | var vm = new Vue({
2 |   el: '#app',
```

```

3   components: {
4     'book': BookComponent,
5   },
6   data: {
7     judul: 'judul buku'
8   }
9 })

```

Oke, mari kita praktikkan ke contoh yang lebih nyata. Masih dengan component book,

```

1 // deklarasi component book, lihat contoh sebelumnya
2 // ...
3
4 // deklarasi object vue dengan data books
5 var vm = new Vue({
6   el: '#app',
7   components: {
8     'book': BookComponent,
9   },
10  data: {
11    books : [
12      {
13        id: 99,
14        title: 'C++ High Performance',
15        description: 'Write code that scales across CPU
registers, multi-core, and machine clusters',
16        authors: 'Viktor Sehr, Björn Andrist',
17        publish_year: 2018,
18        price: 100000,
19        image: 'c++-high-performance.png'
20      },
21      {
22        id: 100,
23        title: 'Mastering Linux Security and Hardening',
24        description: 'A comprehensive guide to mastering the art
of preventing your Linux system from getting compromised',
25        authors: 'Donald A. Tevault',
26        publish_year: 2018,
27        price: 125000,
28        image: 'mastering-linux-security-and-hardening.png'
29      },
30      {
31        id: 101,
32        title: 'Mastering PostgreSQL 10',
33        description: 'Master the capabilities of PostgreSQL 10 to
efficiently manage and maintain your database',
34        authors: 'Hans-Jürgen Schönig',
35        publish_year: 2016,
36        price: 90000,
37        image: 'mastering-postgresql-10.png'
38      },
39      {
40        id: 102,
41        title: 'Mastering MySQL 8.0',
42        description: 'A practical guide to MySQL 8.0',
43        authors: 'David Litchfield',
44        publish_year: 2018,
45        price: 110000,
46        image: 'mastering-mysql-8.0.png'
47      }
48    ]
49  }
50})

```

```

44     title: 'Python Programming Blueprints',
45     description: 'How to build useful, real-world
46     applications in the Python programming language',
47     authors: 'Daniel Furtado, Marcus Pennington',
48     publish_year: 2017,
49     price: 75000,
50     image: 'python-programming-blueprints.png'
51   },
52 ]
53 )

```

Properti data books berbentuk array dari objek, silakan merujuk kembali ke bab yang membahas tentang List. Kemudian pada template utama, kita akan gunakan v-for untuk merender data books di atas, sebagai berikut.

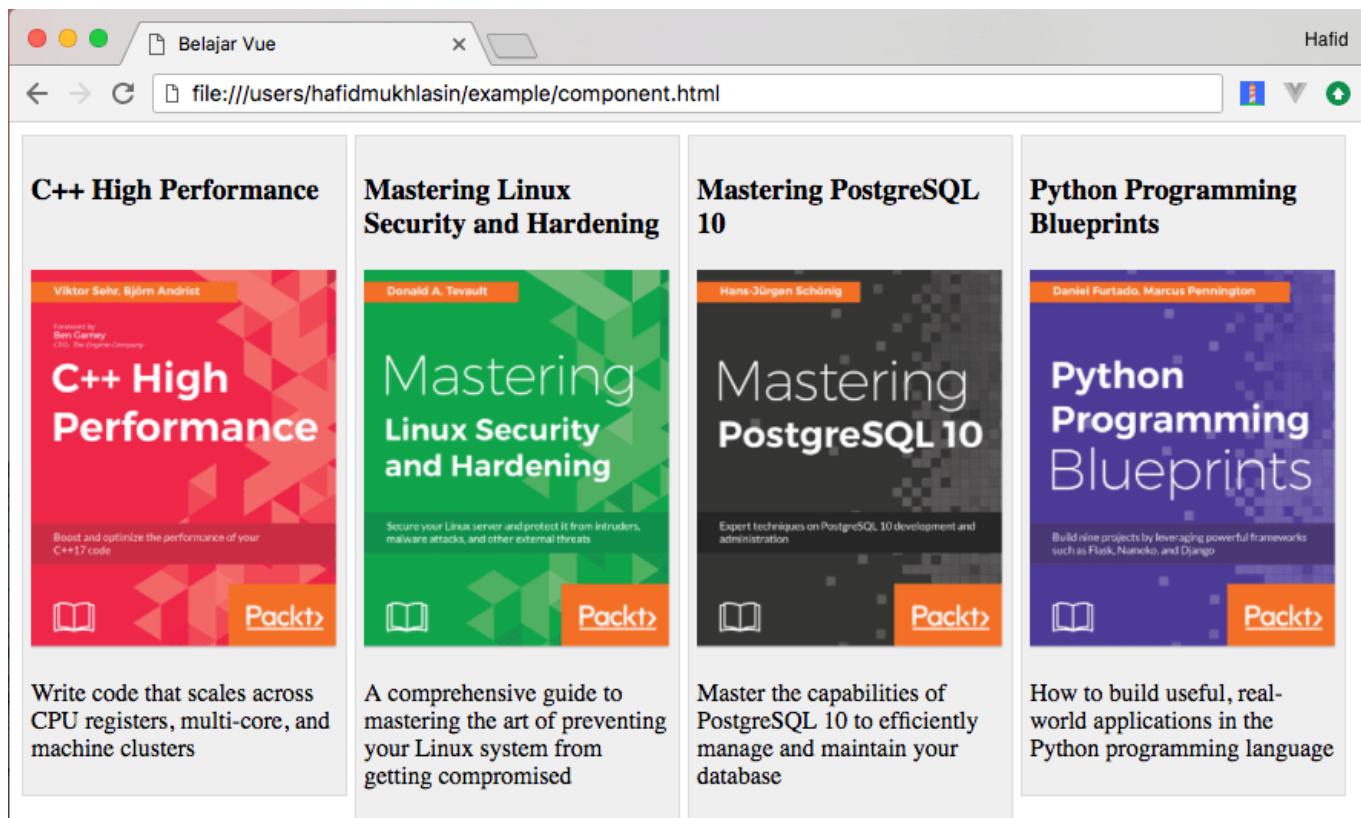
```

1 <div id="app">
2   <book
3     v-for="book in books"
4       :key="book.id"
5       :title="book.title"
6       :description="book.description"
7       :image = "book.image"
8     >
9   </book>
10 </div>

```

Sebagaimana penjelasan sebelumnya, maka semua atribut untuk props kita perlu tambahkan directive v-bind atau disingkat : karena nilainya dinamis mengikuti objek book pada v-for.

Hasilnya sebagai berikut.



Supaya kode lebih rapi dan mudah dibaca, tentunya kita bisa saja mengubah tipe data yang kita kirimkan ke component melalui props, di mana sebelumnya teks biasa menjadi objek buku.

Untuk melakukan hal itu maka, deklarasi component book harus kita ubah menjadi seperti berikut.

```

1 var BookComponent = {
2   data () {
3     return {
4       classCard: 'card'
5     }
6   },
7   props: [ 'book' ],
8   template : `
9     <div :class="classCard">
10       <h3>{{ book.title }}</h3>
11       
12       <p v-html="book.description"></p>
13     </div>
14   `
15 }
```

Karenanya, sekarang kode template utamanya cukup seperti berikut.

```

1 <div id="app">
2   <book
3     v-for="book in books"
4     :key="book.id"
5     :book="book"
6     >
7   </book>
8 </div>
```

Hasilnya pada browser tentu saja sama, namun kodingan akan terlihat lebih rapi dan mudah dibaca.

Update Data Parent From Component

Kebalikan dari bagian sebelumnya, kita juga diizinkan untuk mengupdate data pada objek parent melalui component. Hal ini bisa kita lakukan dengan menggunakan perintah `$emit('event', 'nilai')`.

Event pada kode di atas adalah custom event handler pada parent component yang akan dilisten oleh component tersebut. Sedangkan nilai dapat berisi perintah yang akan dijalankan dalam bentuk JS Expression.

Mari coba kita simulasikan hal itu melalui contoh berikut ini. Kita masih akan menggunakan kode sebelumnya, namun kita perlu tambahkan variabel baru, misalnya `selectedBook` yang nantinya akan digunakan untuk menampung data judul buku yang dipilih atau diklik oleh pengguna.

```

1 var vm = new Vue({
2   el: '#app',
3   data: {
4     books: [ /* ... */ ],
5     selectedBook: '',
6   },
7   components: {
8     'book': BookComponent,
```

```
9     },
10    })
```

Kemudian pada template, kita akan tambahkan kode untuk menampilkan data `selectedBook` dan juga kita akan tambahkan custom event yang berfungsi mengubah nilai dari data `selectedBook`

```
1 <div id="app">
2   <h1>Selected: {{ selectedBook }}</h1>
3   <book
4     v-for="book in books"
5       :key="book.id"
6       :book="book"
7       @selected="selectedBook = $event"
8     >
9   </book>
10 </div>
```

Pada contoh di atas, custom event yang kita gunakan bernama `@selected` (bebas saja, kamu bisa gunakan `@pilihan` atau yang lain). Sedangkan nilainya, kita gunakan perintah `selectedBook = $event`, yang berarti bahwa nilai `selectedBook` nantinya akan diisi dengan nilai dari `$event`.

Nah nilai dari `$event` nantinya adalah nilai yang didapat dari perintah `$emit` yang di eksekusi di dalam component `book`.

Sekarang waktunya kita ngoprek component `book`. Pada template component, kita tambahkan button yang berfungsi mentrigger event `selected` yang kita definisikan di atas.

```
1 var BookComponent = {
2   data () {
3     return {
4       classCard: 'card'
5     }
6   },
7   props: [ 'book' ],
8   template : `
9     <div :class="classCard">
10      <h3>{{ book.title }}</h3>
11
12      
13      <p v-html="book.description"></p>
14
15      <button @click="$emit('selected', book.title)"> Select
16    </button>
17  `,
18}
```

Dari kode di atas, kita cukup fokus pada bagian ini:

```
1 <button @click="$emit('selected', book.title)"> Select </button>
```

Nilai dari book.title atau judul buku akan di-emit atau di set ke event selected. Sehingga ketika kita merujuk ke kode sebelumnya maka dengan cara ini, nilai dari selectedBook akan sama dengan nilai dari book.title.

Mari kita lihat hasilnya pada browser sebagai berikut.

The screenshot shows a browser window with the title 'Belajar Vue'. The address bar shows 'file:///Users/hafidmukhlasin/example/component.html'. The main content area displays four book cards:

- C++ High Performance** by Viktor Sehr, Björn Andrist. Description: Boost and optimize the performance of your C++17 code. Button: Select
- Mastering Linux Security and Hardening** by Donald A. Tewault. Description: Secure your Linux server and protect it from intruders, malware attacks, and other external threats. Button: Select (highlighted with a red border)
- Mastering PostgreSQL 10** by Hans-Jürgen Schöning. Description: Expert techniques on PostgreSQL 10 development and administration. Button: Select
- Python Programming Blueprints** by Daniel Furtado, Marcus Pennington. Description: Build nine projects by leveraging powerful frameworks such as Flask, Nameko, and Django. Button: Select

Ketika button pada suatu buku diklik maka judul buku tersebut akan dikirimkan ke parent component.

Two Way Data Binding on Component

Pada pembahasan yang lalu, kita telah membahas tentang two way data binding pada kasus elemen input dengan menggunakan directive v-model. Pada bagian ini kita akan membahas mengenai hal tersebut namun untuk diterapkan pada component untuk input data.

Ingat bahwa kode v-model memiliki padanan sebagai berikut.

```
1 <input v-model="searchText">
```

setara dengan kode berikut.

```
1 <input
2   :value="searchText"
3   @input="searchText = $event.target.value"
4 >
```

Yap, pada kode kedua, v-bind bertugas memastikan bahwa attribut value bernilai sama dengan variabel searchText. Sedangkan v-on bertugas mengupdate variabel searchText sesuai data yang diinput oleh user.

Melalui pendekatan ini, kita bisa terapkan pada component custom input

```

1 <custom-input
2   v-bind:value="searchText"
3   v-on:input="searchText = $event"
4 ></custom-input>

```

Di dalam component ini elemen input harus melakukan dua hal:

- Bind atribut value ke value props sehingga sama nilainya dengan componentnya.
- Ketika event input, emit event input dengan value baru.

Berikut ini kode pada component.

```

1 Vue.component('custom-input', {
2   props: ['value'],
3   template: `
4     <input
5       :value="value"
6       @input="$emit('input', $event.target.value)"
7     >
8   `
9 })

```

Nah, dengan deklarasi component seperti itu, maka kita sekarang bisa gunakan cara yang lebih singkat pada template utama kita yaitu

```

1 <custom-input v-model="searchText"></custom-input>

```

Content Distribution with Slots

Pada bagian terdahulu tentang mengirimkan data ke Component, telah dibahas mengenai cara mengirimkan data ke component dengan menggunakan `props`, nah pada bagian ini kita akan belajar mengirimkan konten ke dalam component.

Misalnya kita memiliki component bernama `information`

```

1 Vue.component('information', {
2   template: `
3     <div class="card">
4       <strong>Informasi</strong>
5     </div>
6   `})

```

Sehingga untuk mengaksesnya pada template cukup dengan `<information></information>`, Nah kita tidak bisa serta merta menambahkan sesuatu konten di antara tag `information` tersebut layaknya tag html biasa.

Sehingga kode template berikut akan menampilkan error.

```

1 <information>Halo apa kabar?</information>

```

Lalu bagaimana kita bisa menambahkan konten di dalam component? caranya, kita harus deklarasikan menggunakan elemen `slot`.

```

1 Vue.component('information', {
2   template: `
3     <div class="card">
4       <strong>Informasi</strong>
5       <hr>
6       <slot></slot>
7     </div>
8   `
9 })

```

Kemudian component tersebut dapat kita panggil sebagai berikut.

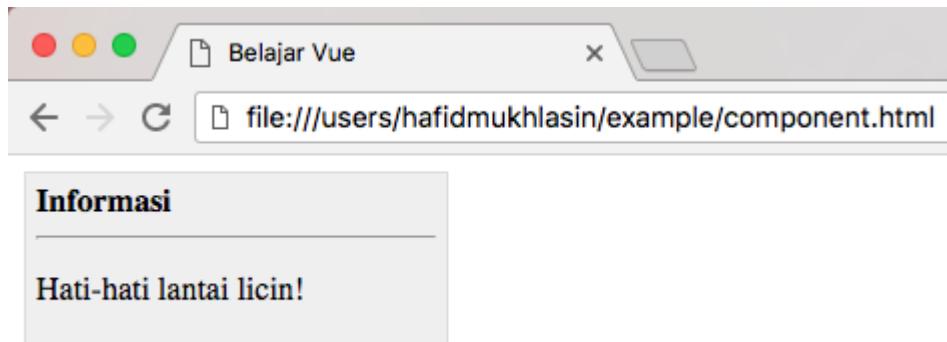
```

1 <div id="app">
2   <information>
3     <p>Hati-hati lantai licin!</p>
4   </information>
5 </div>

```

Nah, pada contoh di atas, elemen `<slot></slot>` akan `di-replace` dengan konten `<p>Hati-hati lantai licin!</p>`

Berikut hasilnya.



Dibandingkan dengan `props`, penggunaan `slot` lebih fleksible yang memungkinkan pengguna component mengirimkan konten berupa HTML ataupun template ke dalam component, sehingga tampilan component akan sangat mudah dikustomisasi.

Catatan: kode slots ini diperkenalkan sejak versi 2.6.0, menggantikan atribut slot dan slot-scope yang sekarang deprecated.

Fallback Slots

Kita juga diizinkan memberikan konten default untuk suatu slot yang tidak diset di parent, caranya dengan menambahkan konten di dalam tag slot.

```

1 <div class="card">
2   <strong>Informasi</strong>
3   <hr>
4   <slot>Tanpa informasi</slot>
5 </div>

```

Sehingga jika component ini dipanggil menyertakan konten dari slot misalnya:

```
1 <information></information>
```

Maka ketika dirender akan menampilkan konten Tanpa informasi
Sedangkan jika kita definisikan konten slotnya sebagai berikut:

```
1 <information>
2   <p>Hati-hati lantai licin!</p>
3 </information>
```

Maka ketika dirender akan menampilkan konten <p>Hati-hati lantai licin!</p>

Penamaan Slot

Bagaimana jika konten yang akan kita kirimkan lebih dari satu bagian? misal pada component `information` kita memiliki dua bagian konten di mana bagian pertama berisi konten judul dari informasi dan konten kedua berisi isi dari informasi, kodenya sebagai berikut

```
1 <div class="card">
2   <slot>Judul Informasi</strong>
3   <hr>
4   <slot>Isi Informasi</slot>
5 </div>
```

Lalu bagaimana mendefinisikan dua konten slot tersebut dari template utama?

Yap, tentu cara biasa tidak bisa, karenanya Vue mempunyai cara untuk mendefinisikan masing-masing konten slot tersebut yaitu dengan memberikan atribut `name` pada tag slot.

```
1 <div class="card">
2   <slot name="judul"></strong>
3   <hr>
4   <slot name="isi"></slot>
5 </div>
```

Sehingga kode componentnya menjadi

```
1 Vue.component('information', {
2   template: `
3     <div class="card">
4       <slot name="judul"></strong>
5       <hr>
6       <slot name="isi"></slot>
7     </div>
8   `
9 })
```

Lalu bagaimana cara mendefinisikan konten masing-masing slot component tersebut pada template utama? Kita bisa melakukannya dengan menggunakan elemen template dengan attribut `v-slot` yang bernilai nama dari slot yang dituju.

```

1 <information>
2   <template v-slot="judul">
3     <h1>Info Penting!<h1>
4   </template>
5   <template v-slot="isi">
6     <p>Waspada, banyak curanmor!</p>
7   </template>
8 </information>

```

Maka jika dirender akan menghasilkan kode HTML berikut:

```

1 <div class="card">
2   <h1>Info Penting!<h1>
3   <hr>
4   <p>Waspada, banyak curanmor!</p>
5 </div>

```

Scoped Slot

Lalu bagaimana jika di dalam slot tadi, kita ingin mengakses property data dari component tersebut. Misal untuk fallback

```

1 Vue.component('alert', {
2   data () {
3     return {
4       defaultAlert: 'Awas Bahaya',
5     }
6   },
7   template: `
8     <div>
9       <slot>{{ defaultAlert }}</strong>
10      </div>
11    `
12 })

```

Namun hal ini tidak bisa kita lakukan, data defaultAlert tidak bisa diakses pada slot secara langsung karena pada hakikatnya slot ini adalah konten dari parentnya. Nah untuk menghadirkan properti data tersebut pada slot maka kita perlu membinding variabelnya pada slot. Berikut ini contohnya.

```

1 <div>
2   <slot :defaultJudul="defaultAlert">{{ defaultAlert }}</strong>
3 </div>

```

Gimana? masuk akal kan? 😊

Nah lebih dari itu, kita juga bisa mengakses variabel yang dibinding tersebut dari parent component untuk didistribusikan kembali pada konten slot,

```

1 <alert>
2   <template v-slot:default="slotData">
3     <h1>{{ slotData.defaultAlert }}</h1>

```

```

4   </template>
5 </alert>
```

Catatan: slotData hanyalah sebuah nama variabel, kamu bisa gunakan nama lain. Sedangkan default adalah nama dari slot (karena pada slot component alert tidak didefinisikan namanya maka kita bisa pakai default).

Bisa juga kita tuliskan sebagai berikut, dimana variabel slotData dialiaskan dengan variabel sd.

```

1 <alert>
2   <template v-slot:default="{ sd: slotData }">
3     <h1>{{ sd.defaultAlert }}<h1>
4   </template>
5 </alert>
```

Pada contoh di atas yang hanya memiliki sebuah slot maka kita bisa tuliskan v-slot langsung pada componentnya tanpa elemen template lagi

```

1 <alert v-slot:default="slotData">
2   <h1>{{ slotData.defaultAlert }}<h1>
3 </alert>
```

atau disingkat menjadi

```

1 <alert v-slot="slotData">
2   <h1>{{ slotData.defaultAlert }}<h1>
3 </alert>
```

Dynamic Slot Names

Sejak versi 2.6.0+, kita juga bisa menggunakan nama slot dinamis melalui v-slot.

```

1 <alert>
2   <template v-slot:[dynamicSlotName]>
3     ...
4   </template>
5 </alert>
```

Named Slots Shorthand

Jika directive v-on dapat dituliskan secara singkat dengan @ dan v-bind dengan :, maka v-slot menggunakan simbol #. Sehingga kita bisa gunakan kode berikut ini.

```

1 <alert>
2   <template #default="slotData">
3     <h1>{{ slotData.defaultAlert }}<h1>
4   </template>
5 </alert>
```

Single File Component

ES6 memungkinkan kita bisa membuat deklarasi component dalam file yang terpisah sehingga file utama tidak menjadi terlalu panjang. Hal ini bisa dilakukan dengan fitur import dan export.

Buat file bernama `BookComponent.js` yang berisi deklarasi dari component book yang telah kita bahas sebelumnya.

```

1  export const BookComponent = {
2      data () {
3          return {
4              classCard: 'card'
5          }
6      },
7      props: [ 'book' ],
8      template : `
9          <div :class="classCard">
10             <h3>{{ book.title }}</h3>
11             
12             <p v-html="book.description"></p>
13             <button @click="$emit('selected', book.title)"> Select
14         </button>
15     </div>
16 `}
```

Kode di atas meng-export konstanta `BookComponent`. Kemudian pada file utama, kita import file `BookComponent.js` di atas.

```

1  <script type="module">
2
3  import { BookComponent } from './BookComponent.js'
4
5  var vm = new Vue({
6      el: '#app',
7      components: {
8          'book': BookComponent,
9      },
10     data: {
11         selectedBook: '',
12         books : [
13             /* ... */
14         ]
15     }
16 })
17 </script>
```

Ada dua bagian penting pada kode di atas yaitu

```

1  <script type="module">
2
3  import { BookComponent } from './BookComponent.js'
```

Mari kita lihat hasilnya.

The screenshot shows a browser window with the title "Belajar Vue". The address bar contains the URL "file:///users/hafidmukhlasin/example/component.html". The developer tools are open, with the "Console" tab selected. A red error message is visible: "Access to Script at 'file:///users/hafidmukhlasin/example/BookComponent.js' from origin 'null' has been blocked by CORS policy: Invalid response. Origin 'null' is therefore not allowed access." Other tabs in the developer tools include Elements, Sources, Network, Performance, and Memory.

Ops, terjadi error. Sebagaimana yang telah lalu bahwa cara ini hanya akan berjalan dengan baik jika kita mengaksesnya melalui protocol http.

The screenshot shows a browser window with the title "Belajar Vue". The address bar contains the URL "localhost/component.html". The page displays a list of four books with their covers and descriptions. Each book has a "Select" button below it.

Book Title	Description	Action
C++ High Performance	Viktor Sehr, Björn Andrist Boost and optimize the performance of your C++17 code	
Mastering Linux Security and Hardening	Donald A. Tevault Secure your Linux server and protect it from intruders, malware attacks, and other external threats	
Mastering PostgreSQL 10	Hans-Jürgen Schönig Expert techniques on PostgreSQL 10 development and administration	
Python Programming Blueprints	Daniel Furtado, Marcus Pennington Build nine projects by leveraging powerful frameworks such as Flask, Nameko, and Django	

Jika kita klik salah satu tombol maka hasilnya.

The screenshot shows a browser window with the address bar containing 'localhost/component.html'. The main content area has a heading 'Selected: Mastering Linux Security and Hardening' enclosed in a red border. Below this are four book covers from Packt Publishing:

- C++ High Performance** by Viktor Sehr, Björn Andrist. Description: Write code that scales across CPU registers, multi-core, and machine clusters. Call-to-action: Select.
- Mastering Linux Security and Hardening** by Donald A. Tevault. Description: A comprehensive guide to mastering the art of preventing your Linux system from getting compromised. Call-to-action: Select.
- Mastering PostgreSQL 10** by Hans-Jürgen Schönig. Description: Master the capabilities of PostgreSQL 10 to efficiently manage and maintain your database. Call-to-action: Select.
- Python Programming Blueprints** by Daniel Furtado, Marcus Pennington. Description: How to build useful, real-world applications in the Python programming language. Call-to-action: Select.

Selected: Mastering Linux Security and Hardening

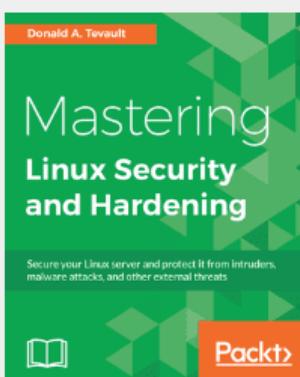
C++ High Performance



Write code that scales across CPU registers, multi-core, and machine clusters

Select

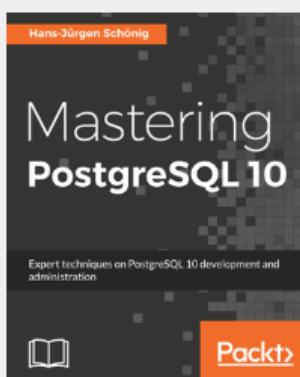
Mastering Linux Security and Hardening



A comprehensive guide to mastering the art of preventing your Linux system from getting compromised

Select

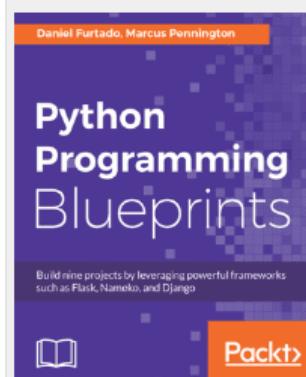
Mastering PostgreSQL 10



Master the capabilities of PostgreSQL 10 to efficiently manage and maintain your database

Select

Python Programming Blueprints



How to build useful, real-world applications in the Python programming language

Select

Perhatian: best practice terkait single file component pada Vue akan kita bahas secara bertahap pada bagian selanjutnya sebab membutuhkan beberapa konsep yang belum dibahas hingga bagian ini. Adapun tutorial single file component pada bagian ini hanya membahas dari sisi konsepnya. Atau silahkan merujuk dokumentasi resminya di <https://vuejs.org/v2/guide/single-file-components.html>

Dynamic Components

Ide dari dynamic component ini adalah kita bisa memuat (meload) suatu component yang sudah diregister secara dinamis, atau hanya akan kita muat jika kita perlukan saja.

Misalnya kita memiliki dua component yaitu list dan detail.

```

1  var list = {
2      template: `
3          <div class="card">
4              <strong>Bahasa Pemrograman</strong>
5              <ul>
6                  <li>Javascript</li>
7                  <li>PHP</li>
8                  <li>Java</li>
9              </ul>
10         </div>
11     `
12 }
13
14 var detail = {
15     template: `
16         <div class="card">
```

```

17     <strong>PHP</strong>
18     <p>
19         PHP adalah singkatan dari PHP Hypertext Preprocessor.
20     </p>
21   </div>
22   '
23 }
24
25 var vm = new Vue({
26   el: '#app',
27   components: {
28     'list': list,
29     'detail': detail,
30   },
31 })

```

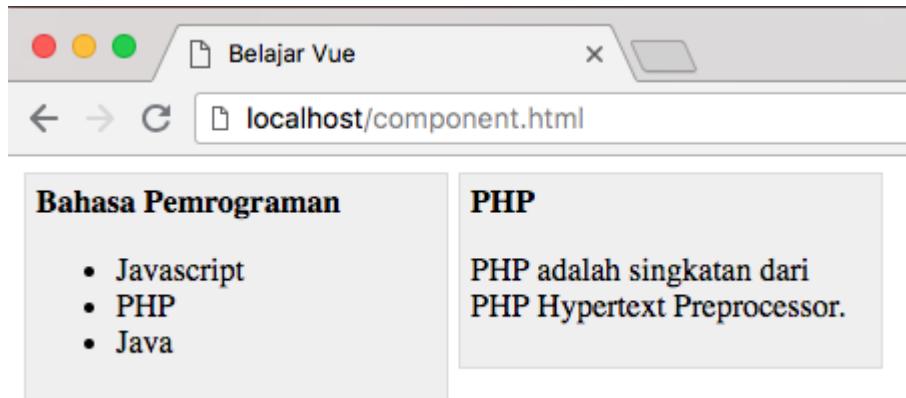
Lalu pada template kita tampilkan semua.

```

1 <div id="app">
2   <list></list>
3   <detail></detail>
4 </div>

```

Hasilnya



Nah, kita bisa memuat component secara dinamis dengan menggunakan elemen `<component>`, elemen ini memiliki directive `v-bind:is` atau `:is` untuk memantau pergantian component yang dimuat.

```

1 <component :is="currentComponent"></component>

```

Baik, untuk mensimulasikannya mari kita edit kode kita pada objek Vue, tambahkan variabel `currentComponent`, seperti berikut ini.

```

1 var vm = new Vue({
2   el: '#app',
3   data: {
4     currentComponent: 'list'
5   },
6   components: {
7     'list': list,
8     'detail': detail,

```

```

9     },
10    })

```

Lalu pada template ubah menjadi.

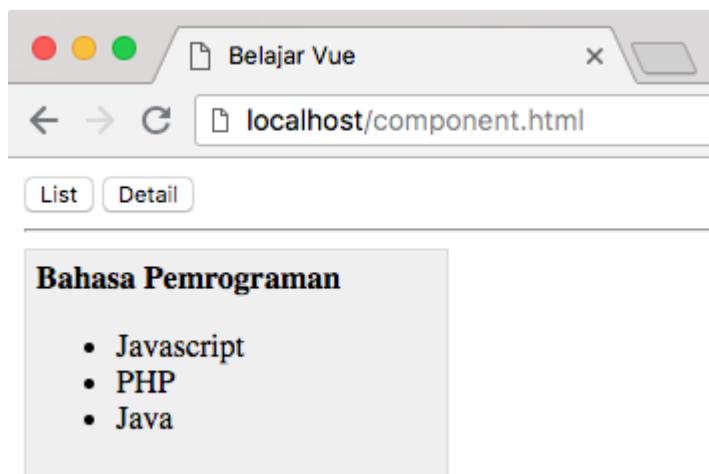
```

1 <div id="app">
2   <button @click="currentComponent = 'list'">List</button>
3   <button @click="currentComponent = 'detail'">Detail</button>
4   <hr>
5   <component :is="currentComponent"></component>
6 </div>

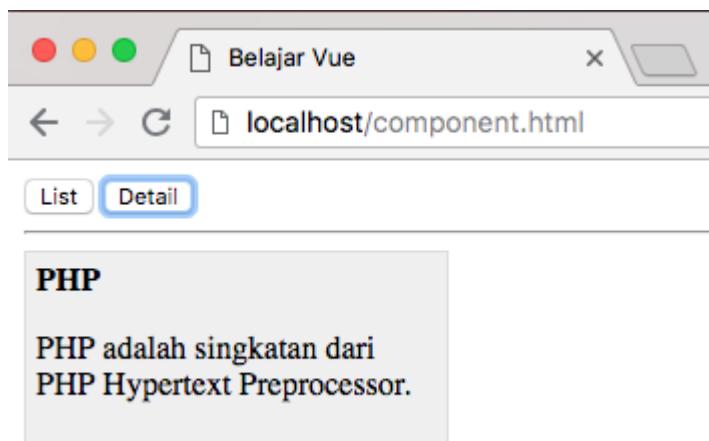
```

Selain ditambahkan elemen component, kita juga tambahkan dua button untuk switcher component atau mengubah variabel currentComponent, supaya dinamis.

Mari kita lihat hasilnya.



Ketika button Detail diklik maka component detail dimuat atau ditampilkan.



Transition Effect

Vue menyediakan cara yang sederhana untuk memberikan efek animasi transisi. Pada bagian sebelumnya yaitu dynamic component, kita dapat menambahkan efek animasi atas pergantian component supaya pergantiannya terasa lembut dan menarik.

Caranya dengan menggunakan elemen `<transition>`. Ubah template menjadi sebagai berikut.

```

1 <div id="app">
2   <button @click="currentComponent = 'list'">List</button>

```

```

3   <button @click="currentComponent = 'detail'">Detail</button>
4   <hr>
5   <transition name="fade" mode="out-in">
6       <component :is="currentComponent"></component>
7   </transition>
8 </div>

```

Kemudian tambahkan style css berikut.

```

1 .fade-enter-active, .fade-leave-active {
2     transition: opacity .5s;
3 }
4 .fade-enter, .fade-leave-to /* .fade-leave-active below version 2.1.8 */ {
5     opacity: 0;
6 }

```

Dan lihat hasilnya 😊, ketika button diklik maka konten current component akan menghilang perlahan dan berganti secara perlahan dengan component berikutnya. Keren kan?

Untuk explore lebih dalam, kamu bisa belajar dari tautan ini <https://vuejs.org/guide/transitions.html>.

Mixins

Mixins (bukan micin) merupakan cara pada Vue untuk mendefinisikan suatu kumpulan fungsi atau option yang akan digunakan pada aplikasi atau component tertentu. Ketika objek Vue atau component menggunakan mixins maka semua option dari mixin tersebut akan digabungkan ke dalam component yang menggunakan tersebut.

Contoh definisi mixins.

```

1 var MixinHello = {
2     created: function () {
3         this.hello()
4     },
5     methods: {
6         hello: function () {
7             console.log('hello from mixin!')
8         }
9     }
10 }

```

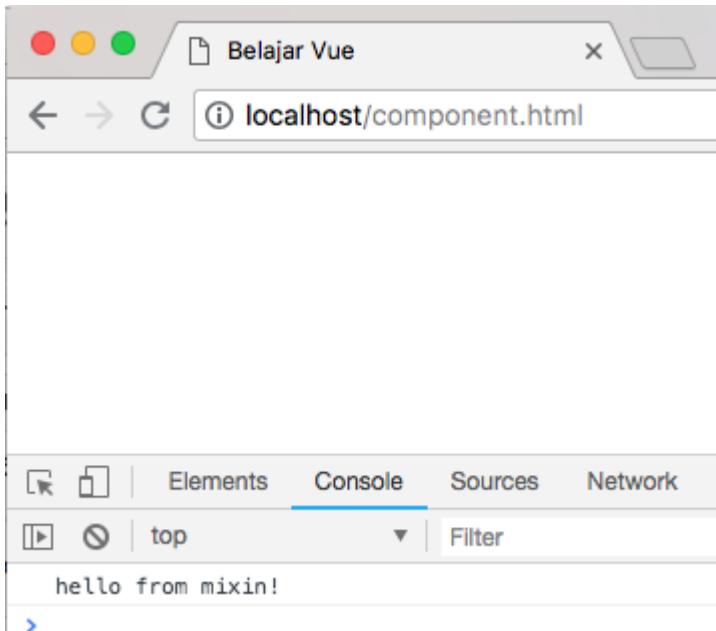
Defini tersebut jika digunakan pada Vue objek seperti berikut.

```

1 var vm = new Vue({
2     el: '#app',
3     mixins: [
4         MixinHello
5     ]
6 })

```

Lihat hasilnya pada browser.



Properti created dan methods pada mixins melebur ke dalam objek Vue, sehingga objek Vue memiliki behavior yang sama dengan mixinnya.

Catatan: sebagaimana component, deklarasi mixin juga dapat diletakkan pada file berbeda dengan objek utamanya. Disamping itu, dengan cara yang sama dengan di atas, mixin juga dapat digunakan dalam component.

Deklarasi mixin pada object Vue atau component pada contoh di atas termasuk deklarasi local. Mixins seperti juga component dapat dideklarasikan secara global. Berikut ini contoh deklarasi global mixin.

```

1 // inject a handler for `text` custom option
2 Vue.mixin({
3   created: function () {
4     let text = this.$options.text
5     if (text) {
6       console.log(text)
7     }
8   }
9 })
10
11 new Vue({
12   text: 'hello!'
13 })
14 // => "hello!"
```

Catatan: hati-hati menggunakan global mixins, sebab akan mempengaruhi setiap object atau component pada Vue.

Plugins

Plugins biasanya digunakan sebagai wrapper untuk menambahkan atau mendaftarkan suatu fitur global pada Vue, misalnya plugin untuk menambahkan:

- global methods atau properties. contoh: vue-custom-element
- global assets: directives/filters/transitions. contoh: vue-touch
- component options menggunakan global mixin. contoh: vue-router

- methods untuk objek Vue melalui Vue.prototype.

Deklarasi Plugins

Berikut ini contoh deklarasi plugin.

```

1  MyPlugin.install = function (Vue, options) {
2      // 1. add global method or property
3      Vue.myGlobalMethod = function () {
4          // something logic ...
5      }
6
7      // 2. add a global asset
8      Vue.directive('my-directive', {
9          bind (el, binding, vnode, oldVnode) {
10             // something logic ...
11         }
12         ...
13     })
14
15     // 3. inject some component options
16     Vue.mixin({
17         created: function () {
18             // something logic ...
19         }
20         ...
21     })
22
23     // 4. add an instance method
24     Vue.prototype.$myMethod = function (methodOptions) {
25         // something logic ...
26     }
27 }
```

Menggunakan Plugin

Suatu plugin digunakan melalui method `Vue.use()`:

```

1  // calls `MyPlugin.install(Vue)`
2  Vue.use(MyPlugin)
3
4  // atau jika ada options
5  Vue.use(MyPlugin, { someOption: true })
```

Catatan: Kode `Vue.use` secara otomatis mencegah duplikasi plugin.

Kesimpulan

Sebagai sub class dari Vue, dengan menggunakan component akan memungkinkan kita memecah kode aplikasi yang kompleks menjadi beberapa bagian sehingga memudahkan dalam pengelolaannya. Disamping itu menggunakan component akan membuat kode kita lebih efisien dan menghindari pengulangan menulis kode, karena component bisa digunakan berkali-kali.

Adapun mixins digunakan untuk mendefinisikan options yang umum digunakan dibanyak component, sehingga tidak terjadi duplikasi atau perulangan kode. Kemudian untuk mendistribusikannya pada komponen bisa dengan menggunakan plugins.

Pada bab berikutnya, kita akan belajar tentang routing aplikasi yaitu suatu mekanisme bagaimana user akan mengakses halaman dari aplikasi kita. Konsep dynamic component yang dibahas pada bab ini juga akan dibahas dengan lebih lanjut menggunakan routing.

Aku yakin kamu pasti bisa!

Routing

Intro

Pada bab ini kita akan belajar tentang routing yaitu bagaimana mekanisme akses aplikasi berbasis Vue. Untuk menangani routing, Vue secara official membuat pustaka bernama Vue Router. Pustaka ini dapat diintegrasikan secara baik tentunya dengan Vue untuk membuat **Single Page Applications**. Kunjungi alamat webnya di <https://router.vuejs.org>.

Features

Vue Router memiliki beberapa fitur diantaranya:

- Route/view bertingkat
- Modular, konfigurasinya berbasis component
- Mendukung params, query, wildcards pada route
- Mendukung efek transisi saat perpindahan halaman / route
- Menangani pengontrolan akses dengan baik
- Link routenya otomatis terhubung dengan CSS class active.
- Mendukung HTML5 history mode atau hash mode, dengan auto-fallback di IE9
- Mendukung fitur scroll dan kustomisasinya

Installation

Sebagai sebuah pustaka Javascript seperti Vue, Vue Router juga perlu ditambahkan ke dalam halaman HTML kita. File pustaka bisa kita unduh ke lokal (sehingga tidak membutuhkan koneksi internet lagi) atau ditautkan langsung dengan server pustaka Vue (CDN).

Kita bisa mengunduh pustaka Vue Router pada tautan ini <https://unpkg.com/vue-router>, atau bisa juga menunjuk ke versi spesifiknya <https://unpkg.com/vue-router@3.0.7/dist/vue-router.js>.

Informasi: cek versi terbaru pada tautan ini <https://github.com/vuejs/vue-router/releases>. Tambahkan `.min` untuk mendapatkan versi production `vue-router.min.js`.

Pustaka Vue Router ditambahkan setelah pustaka Vue.

```
1 <script src="lib/vue.js"></script>
2 <script src="lib/vue-router.js"></script>
```

Untuk mengembangkan aplikasi skala besar, maka untuk instalasi Vue Router disarankan kita menggunakan package manager seperti NPM (penulis menggunakan ini) atau YARN. Lebih dari itu, untuk memudahkan kita membuat scaffolding projek aplikasi (manajemen kode & tools serta konfigurasi saat pengembangan aplikasi), Vue membuat tools CLI <https://cli.vuejs.org>. Topik ini akan dibahas pada bagian berikutnya.

Getting Started

Agar tidak bercampur dengan file-file latihan sebelumnya, mari kita buat satu folder tersendiri untuk belajar routing, misalnya folder `routing`. Buat file `index.html` dan includenya pustaka Vue dan Vue Router.

Untuk bekerja dengan Vue Router, ada dua component penting yang akan kita gunakan yaitu `router-link` dan `router-view`. `router-link` berfungsi menggenerate menu link, sedangkan `router-view` berfungsi sebagai tempat penampung component yang ditampilkan. Disamping kita kita perlu mendeklarasikan class `VueRouter` dan mendefinisikan routing aplikasi kita.

Lalu bagaimana cara kerjanya?

Ketika suatu link yang digenerate dari `router-link` tersebut diklik sehingga URL pada browser berubah maka aksi tersebut akan ditangkap dan diproses oleh Vue Router untuk dilakukan pengecekan apakah ada route yang sesuai, jika ada maka component akan dimuat didalam `router-view`, hal ini mirip dengan konsep dynamic component.

Berikut ini contoh cara mendeklarasikan `router-link` dan `router-view` pada template.

```

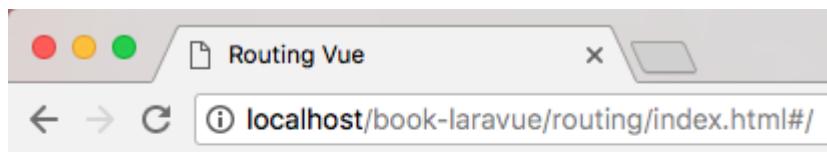
1 <div id="app">
2   <p>
3     <router-link to="/">Home</router-link>
4     <router-link to="/about">About</router-link>
5   </p>
6
7   <router-view></router-view>
8 </div>
```

Kode di atas akan menampilkan dua buah link yaitu Home dengan routing `/` dan About dengan routing `/about`. Selanjutkan kita perlu mendeklarasikan class `VueRouter` dan mendefinisikan routing aplikasi kita.

```

1 // definisikan konfigurasi component
2 const Home = { template: '<div>Halaman Home</div>' }
3 const About = { template: '<div>Halaman About</div>' }
4
5 // mapping route path dengan componentnya, dibaca dari atas ke bawah
6 const routes = [
7   { path: '/', component: Home, alias: '/home' },
8   { path: '/about', component: About }
9 ]
10
11 // register routing aplikasi kita pada objek dari class VueRouter
12 const router = new VueRouter({
13   routes // bentuk pendek dari `routes: routes`
14 })
15
16 // register objek router pada objek Vue
17 var vm = new Vue({
18   el: '#app',
19   router,
20 })
```

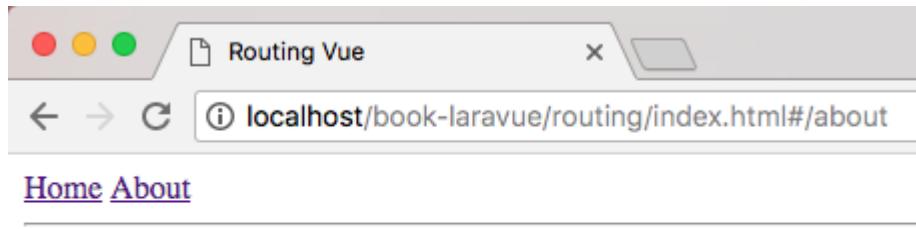
Mari kita ujicoba pada browser.



[Home](#) [About](#)

Halaman Home

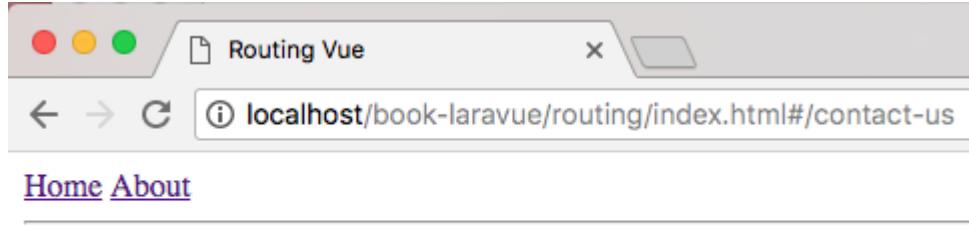
Pada saat pertama kali diakses maka routing pada URL address akan mengarah ke / yang telah kita mapping dengan component Home. Oleh karenanya, ketika pertama kali dibuka maka component Home akan tampil di router-view. Adapun kemudian, ketika link About diklik maka router-view akan menampilkan component About



Halaman About

Mudah sekali bukan?

Bagaimana jika path tidak ditemukan? misalnya user mengakses URL /contact-us



Jika hal itu terjadi maka tidak akan ada component yang dimuat karena tidak ada path dalam mapping yang sesuai. Untuk mengatasi hal ini, maka kita perlu menghandle URL apapun dan meredirectnya ke URL / jika tidak ada yang sesuai. Gunakan path * yang akan sesuai dengan url apapun

```

1 const routes = [
2   { path: '/', component: Home, alias: '/home' },
3   { path: '/about', component: About },
4   { path: '*', redirect: '/' }
5 ]

```

Jadi ketika kita mengakses URL /contact-us maka Vue Router akan meredirect ke URL /.

Dengan menggunakan Vue Router ini maka seluruh routing yang pernah kita klik akan tersimpan di-*history* browser sehingga ketika user mengklik button back, URL akan sesuai dengan routing yang terakhir kali diakses.

Catatan: current route bisa kita akses dalam objek Vue dengan perintah `this.$route.path` atau pada template dengan perintah `{{ $route.path }}`.

Dynamic Routing

Kita bisa menyertakan parameter pada routing sehingga menjadikan routing menjadi dinamis. Misalnya path book/1 untuk buku dengan id 1, path book/2 untuk buku dengan id 2, dst. Maka sebenarnya kita bisa tuliskan seperti berikut ini.

```

1 { path: '/book/1', component: Book },
2 { path: '/book/2', component: Book },
3 { path: '/book/n', component: Book },

```

Hanya saja jika jumlah data bukunya sangat banyak maka tentu sangat tidak efisien jika kita harus definisikan satu persatu. Oleh karena itu, Vue Router menyediakan cara untuk mengatasi hal ini, yaitu dengan menggunakan parameter yang diawali dengan simbol :. Berikut ini contohnya:

```

1 { path: '/book/:id', component: Book }

```

Parameter yang diawali dengan : tersebut (atau dalam hal ini :id) kemudian pada objek Vue atau pada component Book dapat diakses dengan perintah `this.$route.params.nama_parameter`nya atau dalam hal ini `this.$route.params.id`.

Untuk mensimulasikannya, mari kita coba buat dua component lagi yaitu BooksComponent yang berisi daftar buku dan BookComponent yang berisi detail buku.

Membuat BooksComponent

Component ini berfungsi menampilkan daftar buku. Supaya lebih rapi pada tutorial ini deklarasi component buku akan letakkan pada file terpisah.

Buat file BooksComponent.js pada direktori yang sama dengan file index.html

```

1 export const BooksComponent = {
2     data () {
3         return {
4             books: [
5                 {
6                     id: 99,
7                     title: 'C++ High Performance'
8                 },
9                 {
10                    id: 100,
11                    title: 'Mastering Linux Security and Hardening'
12                },
13                {
14                    id: 101,
15                    title: 'Mastering PostgreSQL 10'
16                },
17                {
18                    id: 102,
19                    title: 'Python Programming Blueprints'
20                },
21            ]
22        }
23    },
24    template: `
25        <div>
26            <h1>Daftar Buku</h1>
27            <ul>
28                <li v-for="book of books">
29                    <router-link :to="'/book/' + book.id">
30                        {{ book.title }}

```

```

31          </router-link>
32      </li>
33  </ul>
34 </div>
35
36 }

```

Component ini akan menampilkan daftar judul buku menggunakan directive perulangan v-for, di mana pada setiap itemnya akan ditampilkan router-link yang URL pathnya memiliki format /book/BOOK_ID

Pada file utama `index.html`, import component BooksComponent sebagai berikut:

```

1 <script type="module">
2 import { BooksComponent } from './BooksComponent.js'

```

Selanjutnya, mari kita daftarkan component ini pada mapping routing yang telah kita sebelumnya.

```

1 const routes = [
2     { path: '/', component: Home, alias: '/home' },
3     { path: '/about', component: About },
4     { path: '/books', component: BooksComponent },
5     { path: '*', redirect: '/' }
6 ]

```

Kemudian, tentu saja pada template kita tambahkan router-link untuk mengakses component ini.

```

1 <div id="app">
2     <router-link to="/">Home</router-link>
3     <router-link to="/about">About</router-link>
4     <router-link to="/books">Books</router-link>
5     <hr>
6     <router-view></router-view>
7 </div>

```

Lihat hasilnya. Maka akan tampil link daftar judul buku



[Home](#) [About](#) [Books](#)

Daftar Buku

- [C++ High Performance](#)
- [Mastering Linux Security and Hardening](#)
- [Mastering PostgreSQL 10](#)
- [Python Programming Blueprints](#)

Jika kita klik salah satu link misalnya buku berjudul Mastering PostgreSQL 10 dengan link <http://localhost/book-laravel/routing/index.html#/book/101> maka akan diredirect ke component Home, karena path book tidak sesuai dengan salah satu dari daftar mapping routing.

Membuat BookComponent

Setelah membuat BooksComponent yang berfungsi menampilkan link daftar judul buku. Maka selanjutkan kita perlu membuat satu component lagi untuk menampilkan data detail buku sesuai dengan buku yang dipilih pada BooksComponent.

Misalnya jika diklik link buku Mastering PostgreSQL 10 maka akan menampilkan detail dari buku tersebut.

Buat file BookComponent.js.

```

1  export const BookComponent = {
2      data () {
3          return {
4              books: [
5                  {
6                      id: 99,
7                      title: 'C++ High Performance',
8                      description: 'Write code that scales across CPU
9 registers, multi-core, and machine clusters',
10                     authors: 'Viktor Sehr, Björn Andrist',
11                     publish_year: 2018,
12                     price: 100000,
13                     image: 'c++-high-performance.png'
14                 },
15                 {
16                     id: 100,
17                     title: 'Mastering Linux Security and Hardening',
18                     description: 'A comprehensive guide to mastering the
19 art of preventing your Linux system from getting compromised',
20                     authors: 'Donald A. Tevault',
21                     publish_year: 2018,
22                     price: 125000,
23                     image: 'mastering-linux-security-and-hardening.png'
24                 },
25                 {
26                     id: 101,
27                     title: 'Mastering PostgreSQL 10',
28                     description: 'Master the capabilities of PostgreSQL
29 10 to efficiently manage and maintain your database',
30                     authors: 'Hans-Jürgen Schönig',
31                     publish_year: 2016,
32                     price: 90000,
33                     image: 'mastering-postgresql-10.png'
34                 },
35                 {
36                     id: 102,
37                     title: 'Python Programming Blueprints',
38                     description: 'How to build useful, real-world
39 applications in the Python programming language',
40                     authors: 'Daniel Furtado, Marcus Pennington',

```

```

41         publish_year: 2017,
42         price: 75000,
43         image: 'python-programming-blueprints.png'
44     },
45 ],
46 }
47 },
48 computed: {
49     book() {
50         return this.books.filter((book)=>{
51             return book.id === parseInt(this.$route.params.id)
52         })[0]
53     }
54 },
55 template: `
56     <div v-if="book">
57         <h1>Buku {{ book.title }}</h1>
58         <ul>
59             <li v-for="(num, value) of book">
60                 {{ num +' : '+ value }} <br>
61             </li>
62         </ul>
63     </div>
64 `

}

```

Pada kode di atas, tepatnya properti computed, data books yang berformat list difilter dengan cara membandingkan id buku book.id dan parameter id yang dilewatkan melalui URL this.\$route.params.id.

Pada file utama index.html, import component BookComponent,

```

1 <script type="module">
2 import { BooksComponent } from './BooksComponent.js'
3 import { BookComponent } from './BookComponent.js'

```

Daftarkan component ini pada mapping routing.

```

1 const routes = [
2     { path: '/', component: Home, alias: '/home' },
3     { path: '/about', component: About },
4     { path: '/books', component: BooksComponent },
5     { path: '/book/:id', component: BookComponent },
6     { path: '*', redirect: '/' }
7 ]

```

Perhatikan path book/:id, ini adalah path dinamis dimana parameter id bersifat dinamis sesuai dengan link pada router-linknya. Nah, untuk mengaksesnya kita bisa gunakan kode this.\$route.params.id. Nama parameternya bebas, tidak harus id untuk data id. Artinya pada contoh di atas, boleh saja kita ganti dengan path book/:kode, maka untuk mengakses parameternya menjadi this.\$route.params.kode.

Mari kita lihat hasilnya. Ketika salah satu judul buku diklik maka akan menampilkan detail dari bukunya.

Home About Books

Buku Mastering PostgreSQL 10

- 101 : id
- Mastering PostgreSQL 10 : title
- Master the capabilities of PostgreSQL 10 to efficiently manage and maintain your database : description
- Hans-Jürgen Schönig : authors
- 2016 : publish_year
- 90000 : price
- mastering-postgresql-10.png : image

Programmatic Navigation

Disamping menggunakan element router-link yang menampilkan link untuk menuju ke path tertentu, kita juga dapat menjalankan method dari objek Vue Router untuk meredirect halaman ke path tertentu.

Method tersebut adalah method push()

```

1  router.push( /* location */ )
2
3  // jika di dalam objek Vue atau component, tambahkan this.$
4  this.$router.push( /* location */ )

```

Method ini berfungsi sama dengan ketika kita mengklik link , oleh karena itu klik setara dengan memanggil router.push(...).

Berikut ini contoh variasi pemanggilan method ini.

```

1  // literal string path
2  router.push('/home')
3
4  // object
5  router.push({ path: '/home' })
6
7  // named route /user/123
8  router.push({ name: 'user', params: { userId: 123 }})
9
10 // with query, resulting in /register?plan=private
11 router.push({ path: 'register', query: { plan: 'private' }})

```

Catatan: jangan bingung dengan \$route vs \$router, di mana \$route mengembalikan routing saat ini, sedangkan \$router adalah objek router yang bisa kita gunakan untuk menjalankan fungsi push() dan go().

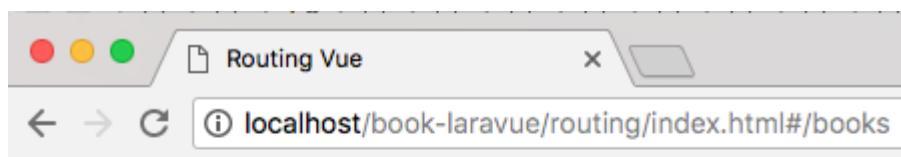
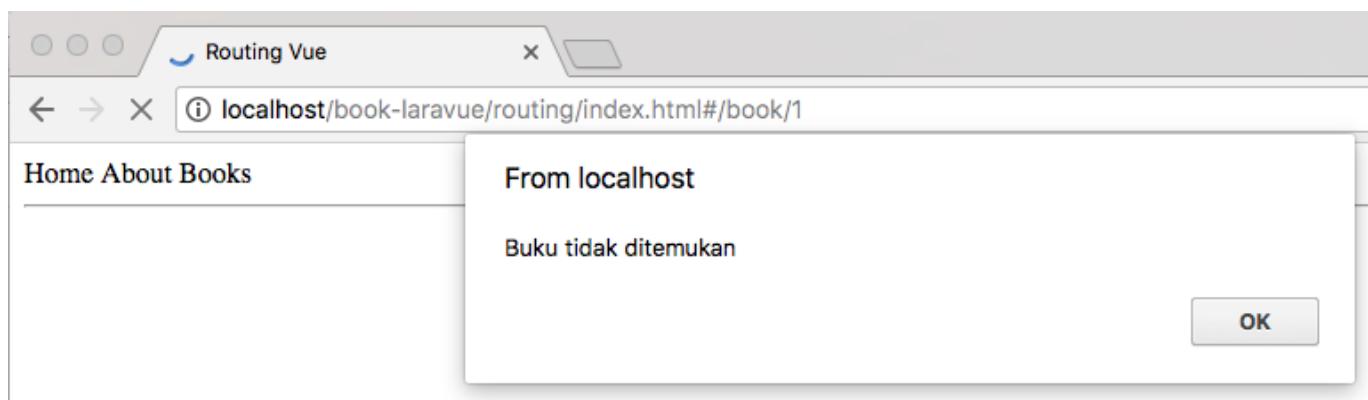
Untuk mencoba method ini, kita akan gunakan contoh sebelumnya, di mana jika pada detail buku atau component BookComponent ternyata buku dengan id yang dikirimkan tidak ditemukan maka halaman akan diredirect ke path books atau component BooksComponent.

```

1  computed: {
2    book() {
3      let book = this.books.filter((book)=>{
4        return book.id === parseInt(this.$route.params.id)
5      })
6
7      // jika buku tidak ditemukan
8      if (book.length==0){
9        // redirect ke path books
10       alert("Buku tidak ditemukan")
11       this.$router.push("/books")
12     }
13     else{
14       return book[0]
15     }
16   }
17 },

```

Mari kita uji coba, akses URL ini <http://localhost/book-laravue/routing/index.html#/book/1> di mana tidak ada buku dengan id 1, maka halaman akan menampilkan pesan bahwa buku tidak ada kemudian diredirect ke halaman daftar buku BooksComponent



Daftar Buku

- [C++ High Performance](#)
- [Mastering Linux Security and Hardening](#)
- [Mastering PostgreSQL 10](#)
- [Python Programming Blueprints](#)

Kita juga bisa mengakses history URL dengan menggunakan method go.

```

1 // go forward by one record, the same as history.forward()
2 router.go(1)
3

```

```

4 // go back by one record, the same as history.back()
5 router.go(-1)
6
7 // go forward by 3 records
8 router.go(3)

```

Sebagai catatan: method router.push, router.replace dan router.go adalah wrapper dari Browser History APIs yaitu window.history.pushState, window.history.replaceState dan window.history.go.

Penamaan Routing

Untuk mengidentifikasi suatu route kita bisa menggunakan nama, dibanding menggunakan path-nya. Tambahkan key `name` pada mapping route

```

1 const routes = [
2     { path: '/', component: Home, alias: '/home' },
3     { path: '/about', component: About },
4     { path: '/books', component: BooksComponent },
5     { path: '/book/:id', name: 'book', component: BookComponent },
6     { path: '*', redirect: '/' }
7 ]

```

Untuk membuat link ke route tersebut kita bisa gunakan kode berikut:

```

1 <router-link :to="{ name: 'book', params: { bookId: 123 } }>Mastering
Vue</router-link>

```

Atau jika menggunakan method push kita bisa gunakan kode berikut.

```

1 router.push({ name: 'book', params: { bookId: 123 } })

```

Kedua contoh diatas akan mengarahkan halaman ke path /book/123.

Mengirimkan Props ke Component Routing

Kita bisa mengirimkan props kepada component routing, melalui properti `props` pada saat mapping routes. Pada contoh terdahulu, kita akan sedikit ubah component BookComponent untuk mensimulasikan penggunaan props ini.

Buka file BookComponent.js, tambahkan properti `props` dengan nilai `id`, dan gunakan untuk memfilter data books dengan cara mengganti `this.$route.params.id` menjadi `this.id`

```

1 export const BookComponent = {
2     data () {
3         return {
4             books: [ /* */ ],
5         }
6     },
7     props: [ 'id' ],
8     computed: {
9         book() {
10             let book = this.books.filter((book)=>{

```

```

11         return book.id === parseInt(this.id)
12     })
13
14     ...
15     }
16 },
17 ...
18 }
```

Kemudian pada mapping route, tambahkan key props yang bernilai true

```

1 const routes = [
2     { path: '/', component: Home, alias: '/home' },
3     { path: '/about', component: About },
4     { path: '/books', component: BooksComponent },
5     { path: '/book/:id', name: 'book', component: BookComponent, props:
6         true },
7     { path: '*', redirect: '/' }
]
```

Lebih sederhana bukan?

Transitions Effect

Sebagaimana bahasan pada dynamic component, pada routing ini kita juga bisa menerapkan atau menambahkan efek transisi antara route dengan sedikit animasi. Caranya masih sama yaitu menggunakan elemen `<transition>`.

Pada template kita bungkus router-view dengan transition

```

1 <transition name="slide" mode="out-in">
2     <router-view></router-view>
3 </transition>
```

Tambahkan sedikit css berikut.

```

1 .fade-enter-active, .fade-leave-active {
2     transition: opacity .5s;
3 }
4 .fade-enter, .fade-leave-to {
5     opacity: 0;
6 }
```

Dan lihatlah keajaibannya 😊.

Bisa juga kita definisikan efek transisi ini per component. Caranya, bungkus template pada component dengan transition.

```

1 const Home = {
2     template: `
3         <transition name="slide">
4             <div> Halaman Home </div>
5         </transition>
6     
```

```

6
7 }

```

Navigation Guards

Vue Router memungkinkan kita menolak atau mengizinkan akses ke suatu route. Misalkan ada halaman yang hanya boleh diakses ketika user sudah login.

Ada tiga cara untuk mendefinisikan navigation guards, yaitu: secara global, per-route, atau dalam component.

Setiap fungsi guard mempunyai tiga argumen:

- to: Route: target route (path) di mana halaman akan diredirect.
- from: Route: current route asal.
- next: Function: fungsi ini harus dipanggil untuk menyelesaikan hook. Aksinya tergantung pada argumen yang dilewatkan via next.
 - next(): navigasi akan dilanjutkan.
 - next(false): navigasi akan dibatalkan.
 - next('/') or next({ path: '/' }): redirect ke route lain.
 - next(error): (2.4.0+) navigasi akan dibatalkan dan error akan dikirimkan ke callbacks via router.onError().

Catatan: Pastikan selalu memanggil fungsi next atau hook tidak akan selesai

Global

Berikut ini sintak untuk mendeklarasikan route guard secara global artinya guard akan berlaku untuk semua route

```

1 const router = new VueRouter({ ... })
2
3 // Guard yang diimplementasikan disini akan dijalankan sebelum route
4 dituju
5 router.beforeEach((to, from, next) => {
6   // ...
7 })
8
9 // akan dijalankan setelah route dituju tapi guard disini tidak akan
10 berefek pada routing
11 router.afterEach((to, from) => {
12   // ...
13 })

```

Per Route

Guard juga bisa diimplementasikan untuk routing tertentu saja. Berikut ini sintaksnya. s

```

1 routes: [
2   {
3     path: '/home',
4     component: Home,
5     beforeEnter: (to, from, next) => {
6       // ...
7     }
8   },

```

```

9   // ...
10 ]

```

Dalam Component

Kita juga bisa mendeklarasikan guard di dalam component. Sebagai berikut:

```

1 const Home = {
2   template: `...`,
3   beforeRouteEnter (to, from, next) {
4     // dipanggil sebelum route dituju & sebelum component yang dituju itu
5     // dibuat
6     // sehingga kita tidak bisa mengakses `this` component
7   },
8   beforeRouteUpdate (to, from, next) {
9     // dipanggil ketika route yang merender component diubah, `this` bisa
10    // diakses
11  },
12  beforeRouteLeave (to, from, next) {
13    // dipanggil ketika akan meninggalkan current route
14  }
}

```

Prevent Leave Accident

Pada bagian berikutnya kita telah belajar di mana kita meletakkan implementasi guard. Sekarang kita akan coba mengimplementasikan contoh penggunaan navigation guard ini untuk hal yang simple yaitu untuk mencegah pengguna keluar dari suatu halaman hanya karena salah klik (tidak sengaja).

Nah, pada contoh ini kita akan menampilkan pesan konfirmasi untuk memastikan bahwa pengguna memang benar-benar ingin keluar atau berpindah dari halaman tersebut.

Kita akan mencoba melalui deklarasi dalam component menggunakan hook `beforeRouteLeave`.

Pada BookComponent, tambahkan properti ini.

```

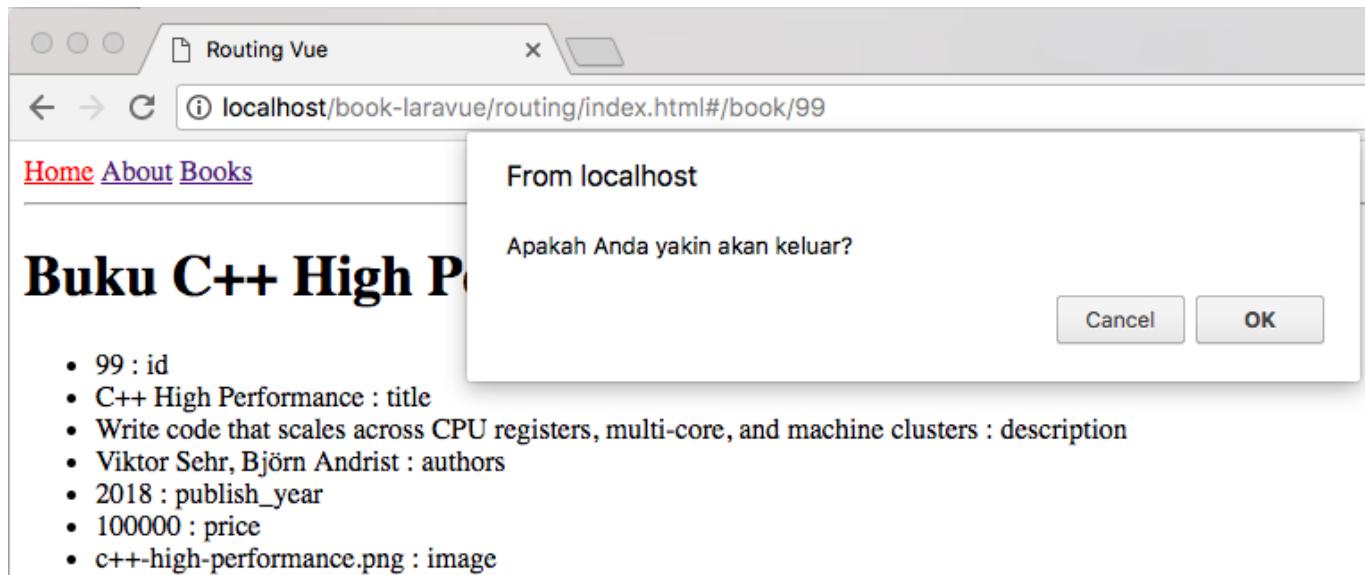
1 export const BookComponent = {
2   data () {
3     return {
4       books: [ /* */ ],
5     }
6   },
7   props: ['id'],
8   computed: {
9     /* */
10  },
11  template: `
12    ...
13  `,
14  beforeRouteLeave (to, from, next) {
15    const answer = window.confirm('Apakah Anda yakin akan keluar?')
16    if (answer) {
17      next()
18    } else {
19      next(false)
20    }
}

```

```
21     }
22 }
```

Kode di atas, kita akan menampilkan dialog confirm. Jika pengguna setuju atau mengkonfirmasi dialog tersebut maka halaman akan diredirect keluar atau next route, jika tidak setuju maka akan tetap pada halaman tersebut next(false).

Mari kita coba, akses BookComponent, kemudian klik tombol back pada browser atau menu link lain (home misalnya), maka hasilnya sebagai berikut:



Keren kan?

Authentication Route

Contoh implementasi lain dari navigation guards, adalah kita bisa mencegah user yang tidak memiliki hak akses untuk mengakses suatu route / halaman tertentu.

Untuk melakukan hal itu, pertama kali kita perlu definisikan pada mapping route, mana saja route yang hanya boleh diakses oleh misalnya user yang sudah login. Vue Router telah menyediakan key `meta` yang bisa kita manfaatkan sebagai penanda routing.

Misalnya, route atau halaman about hanya boleh diakses oleh user yang sudah login, maka kita tambahkan saja meta login yang bernilai true (bebas saja nama metanya).

```
1 const routes = [
2     { path: '/', component: Home, alias: '/home' },
3     { path: '/about', component: About, meta: { login: true } },
4     { path: '/books', component: BooksComponent },
5     { path: '/book/:id', name: 'book', component: BookComponent, props:
6       true },
7     { path: '*', redirect: '/' }
]
```

Kemudian navigation guard bisa kita definisikan secara global.

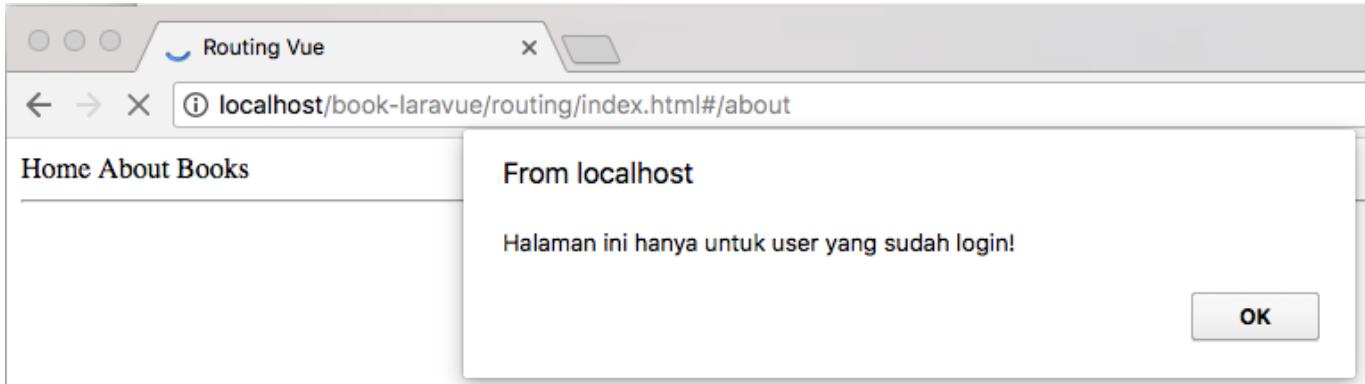
```
1 router.beforeEach((to, from, next) => {
2     if (to.matched.some(record => record.meta.login)) {
3         alert('Halaman ini hanya untuk user yang sudah login!')
4         next(false)
5     }
}
```

```

6     else{
7         next()
8     }
9 })

```

Mari kita coba dengan mengakses link about, maka hasilnya sebagai berikut:



Tentu saja kita bisa membuat skenario, misalnya ketika user sudah login maka boleh mengakses, jika belum login maka route akan diredirect ke halaman login.

```

1 if (!auth.isLoggedIn()) {
2     next('/login')
3 } else {
4     next()
5 }

```

Lho auth dari mana? 😊 auth cuman contoh saja object store pada state management vuex yang berisi data login pengguna. Pada bab selanjutnya akan kita bahas terkait hal ini.

Kesimpulan

Routing merupakan mekanisme untuk mengakses suatu halaman dari aplikasi. Halaman aplikasi merupakan sebuah component. Konsep routing ini mirip dengan konsep dynamic component. Vue menyediakan pustaka resmi untuk menangani routing yaitu Vue Router. Dengan menggunakan routing memungkinkan kita membatasi akses ke suatu halaman aplikasi, misalnya berdasarkan hak aksesnya.

Pada bab selanjutnya kita akan belajar tentang state management yaitu manajemen variabel global untuk memudahkan komunikasi antar komponen.

Sudah lebih dari separuh materi!

State Management

Intro

Pada bab ini kita akan belajar tentang state management menggunakan pustaka resmi Vue yaitu Vuex.

Mengenal State Management

Apa itu state management?

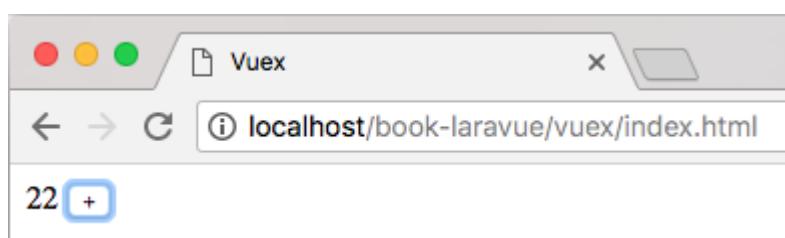
State management merupakan sentralisasi variabel data, sehingga semua component dalam aplikasi dapat mengakses dan memanipulasinya dengan aturan-aturan tertentu sehingga perubahannya dapat diprediksi.

Untuk memahami state management, berikut ini contoh aplikasi counter sederhana berbasis Vue.

```

1 new Vue({
2   el: '#app',
3   // state
4   data: {
5     counter: 0
6   },
7   // view
8   template: `
9     <div>
10       {{ counter }}
11       <button @click="increment()"> + </button>
12     </div>
13   `,
14   // actions
15   methods: {
16     increment () {
17       this.counter++
18     }
19   }
20 })

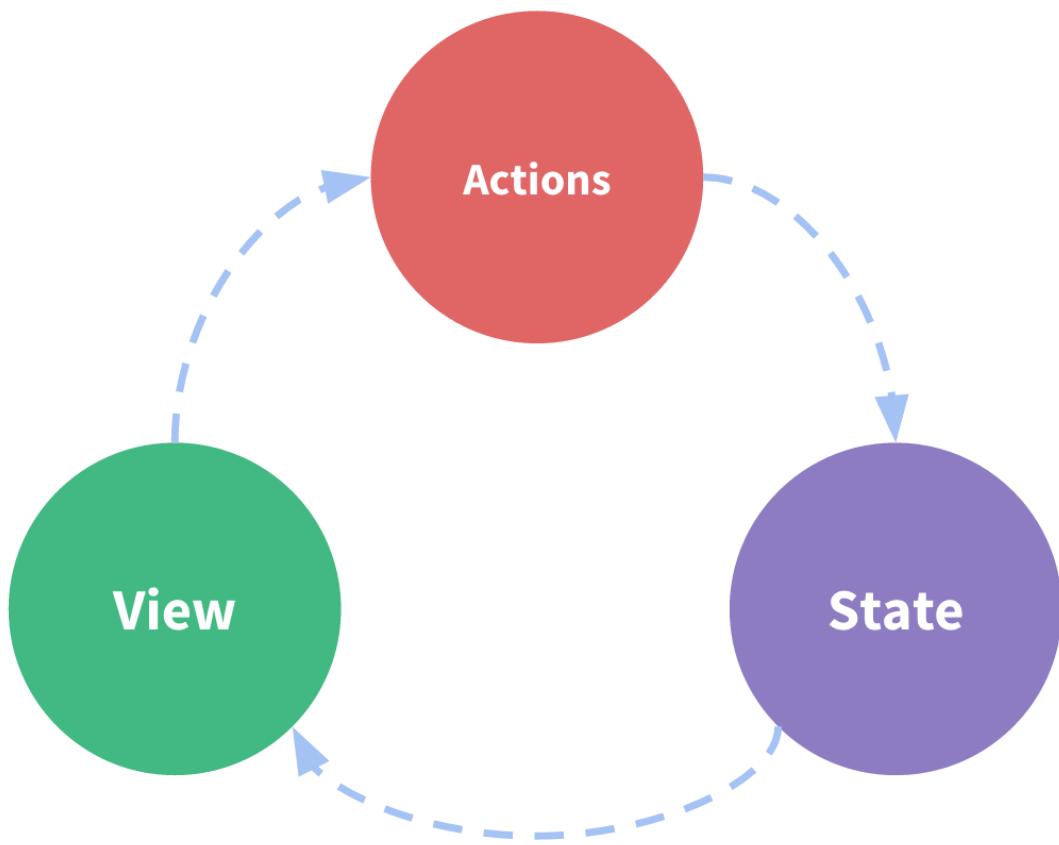
```



Kode pada aplikasi counter di atas memiliki tiga bagian utama.

- State, atau data yang dijadikan sebagai sumber utama yang digunakan oleh aplikasi;
- View, deklarasi mapping dari state, di mana dan bagaimana state akan ditampilkan;
- Actions, jalan untuk mengubah state ketika user melakukan tindakan pada view.

Berikut ini diagram yang menjelaskan konsep di atas (one way data flow).



Ketika pengguna mengklik button + maka akan menjalankan method increment yang berisi perintah untuk menambahkan nilai dari data counter secara increment. Ketika data counter berubah maka otomatis akan memicu perubahan tampilan counter pada template/view.

Terlihat sederhana memang, karena hanya satu component yang mempunya satu method untuk mengakses satu state, namun kalo kita sudah bermain dengan banyak component di mana setiap component menggunakan state yang sama dan masing-masing component tersebut dapat melakukan perubahan terhadap state tersebut maka alurnya akan menjadi kompleks.

Kita memang bisa menggunakan menggunakan props atau konsep two way data binding seperti directive v-model, namun tentu hal itu akan membuat kode kita menjadi sangat kompleks dan bisa jadi perubahan state malah tidak terprediksi dengan baik.

Oleh karena itu diperlukan sentralisasi state yang bisa diakses dan dimanipulasi oleh semua component dalam aplikasi, sehingga perubahan state dapat lebih mudah dimonitoring dan diprediksi.

Kapan Menggunakan State Management?

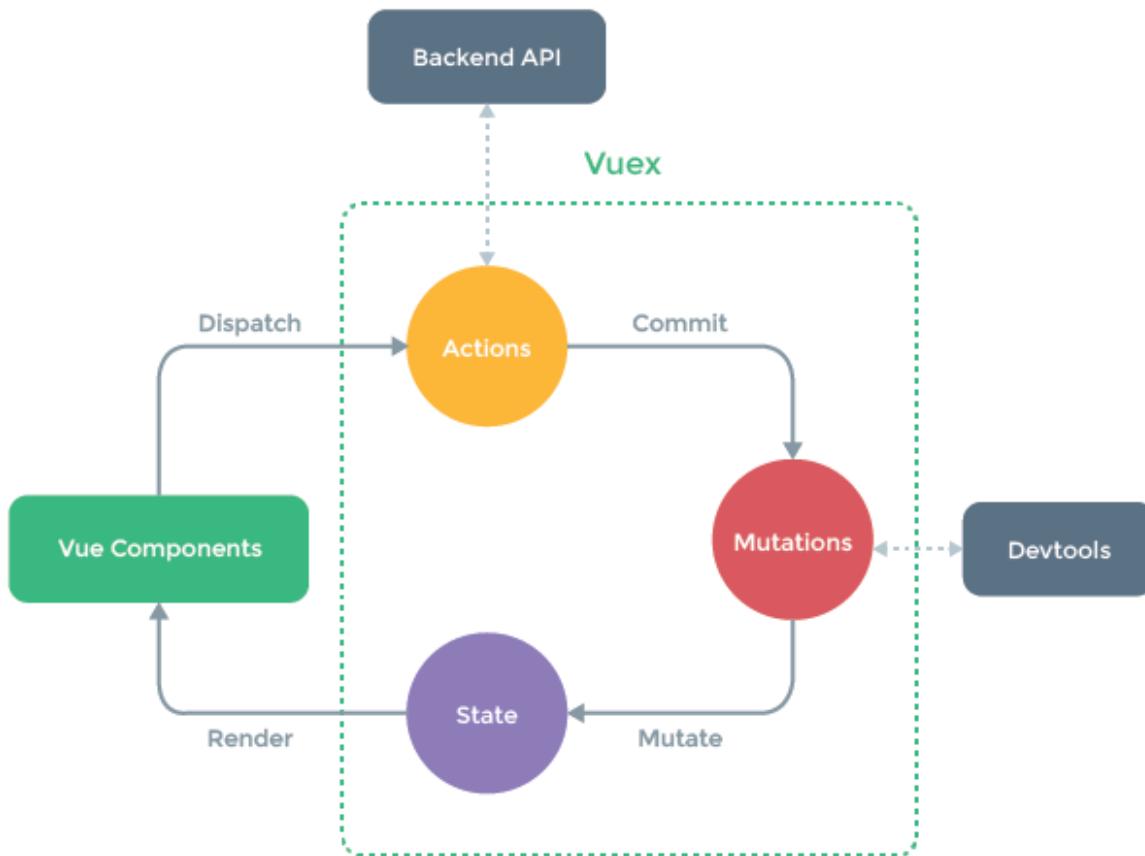
Kita sebaiknya menggunakan state management ketika aplikasi kita kompleks dengan banyak component di dalamannya serta menangani banyak state. Namun jika aplikasi yang kamu bangun sederhana maka tentunya akan overkill atau berlebihan jika state management ini diterapkan.

Pustaka State Management

Sebenarnya, implementasi state management itu out of the box, boleh saja kita membuat mekanisme sendiri atau menggunakan pustaka apa saja misalnya redux, mobx, dll. Namun terkait hal ini, Vue telah membuat pustaka state management-nya sendiri yang tentunya mendukung fitur-fitur Vue secara lebih dalam sehingga sangat disarankan menggunakan pustaka ini.

Pustaka tersebut bernama Vuex yaitu pustaka state management untuk aplikasi berbasis Vue. Ide dasar dari Vuex, terinspirasi oleh arsitektur Flux, Redux dan Elm. Pustaka Vuex juga terintegrasi dengan Vue devtools extension sehingga memudahkan kita melakukan debugging state.

Arsitektur Vuex



Pustaka Vuex menangani dan terdiri dari 3 hal utama yaitu state, mutation dan action. Cara kerja Vuex adalah:

- Mula-mula suatu component melakukan dispatch (pemanggilan) kepada suatu fungsi pada actions;
- Actions yang berisi kumpulan fungsi tersebut bertugas memanggil fungsi pada mutation;
- Fungsi-fungsi pada mutation bertugas mengupdate state.
- Perubahan pada state yang bersifat reaktif akan memicu rendering component.

Instalasi Vuex

Sebagai sebuah pustaka JS seperti Vue Router, Vuex juga perlu ditambahkan ke dalam halaman HTML kita. File pustaka bisa kita unduh ke lokal (sehingga tidak membutuhkan koneksi internet lagi) atau ditautkan langsung dengan server pustaka Vue (CDN).

Kita bisa mengunduh pustaka Vue Router pada tautan ini <https://unpkg.com/vuex>, atau bisa juga menunjuk ke versi spesifiknya <https://unpkg.com/vuex@3.1.1/dist/vuex.js>.

Informasi: cek versi terbaru pada tautan ini <https://github.com/vuejs/vuex/releases>.
Tambahkan `.min` untuk mendapatkan versi production `vux.min.js`.

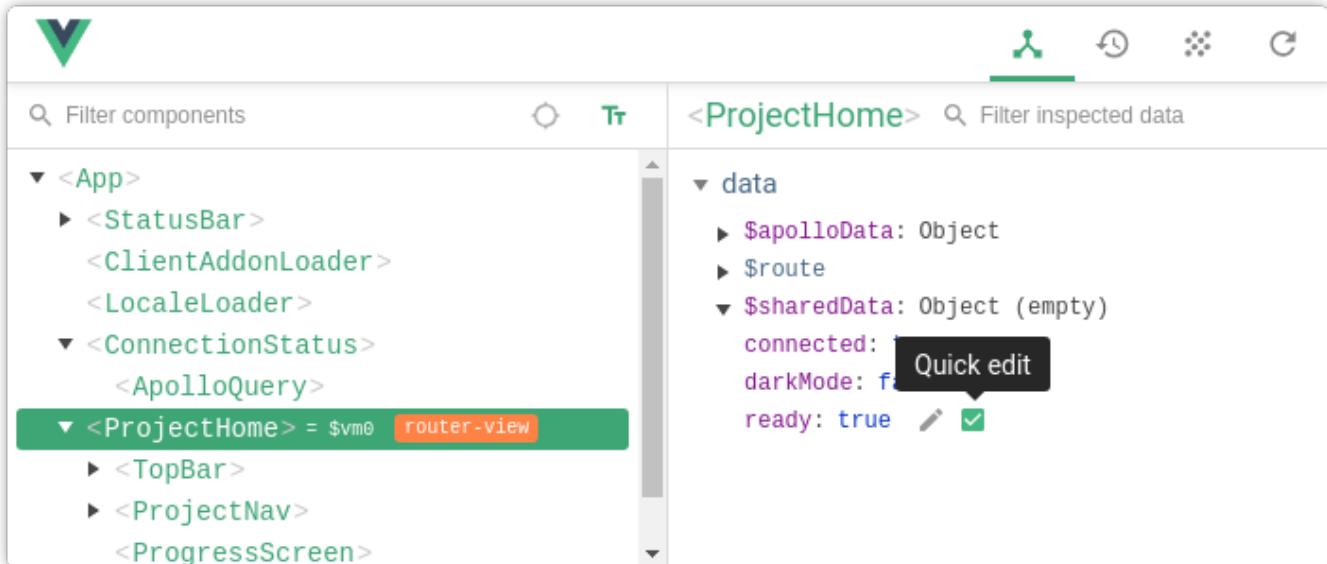
Pustaka Vuex ditambahkan setelah pustaka Vue.

```
1 <script src="lib/vue.js"></script>
2 <script src="lib/vuex.js"></script>
```

Catatan: untuk mengembangkan aplikasi skala besar, maka untuk instalasi Vuex disarankan kita menggunakan package manager seperti NPM (penulis menggunakan ini) atau YARN. Lebih dari itu, untuk memudahkan kita membuat scaffolding projek aplikasi (manajemen kode & tools serta konfigurasi saat pengembangan aplikasi), Vue membuat tools CLI <https://cli.vuejs.org>. Topik ini akan dibahas pada bagian berikutnya.

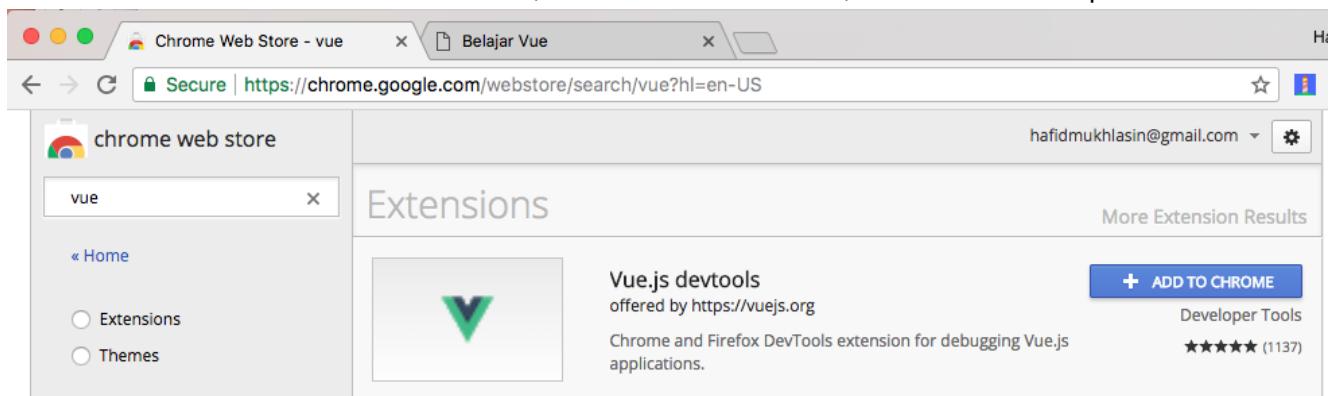
Instalasi Dev Tools

Tools dev-tools ini merupakan extension pada browser (Chrome, Firefox) untuk debugging aplikasi berbasis Vue. Jika kita bermain dengan Vuex, maka tools ini sangat disarankan.

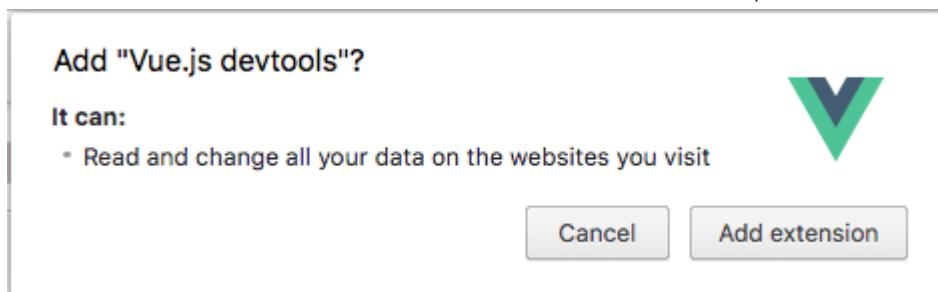


Untuk instalasi selengkapnya silakan cek tautan ini <https://github.com/vuejs/vue-devtools>, pada tutorial ini, penulis menggunakan browser chrome.

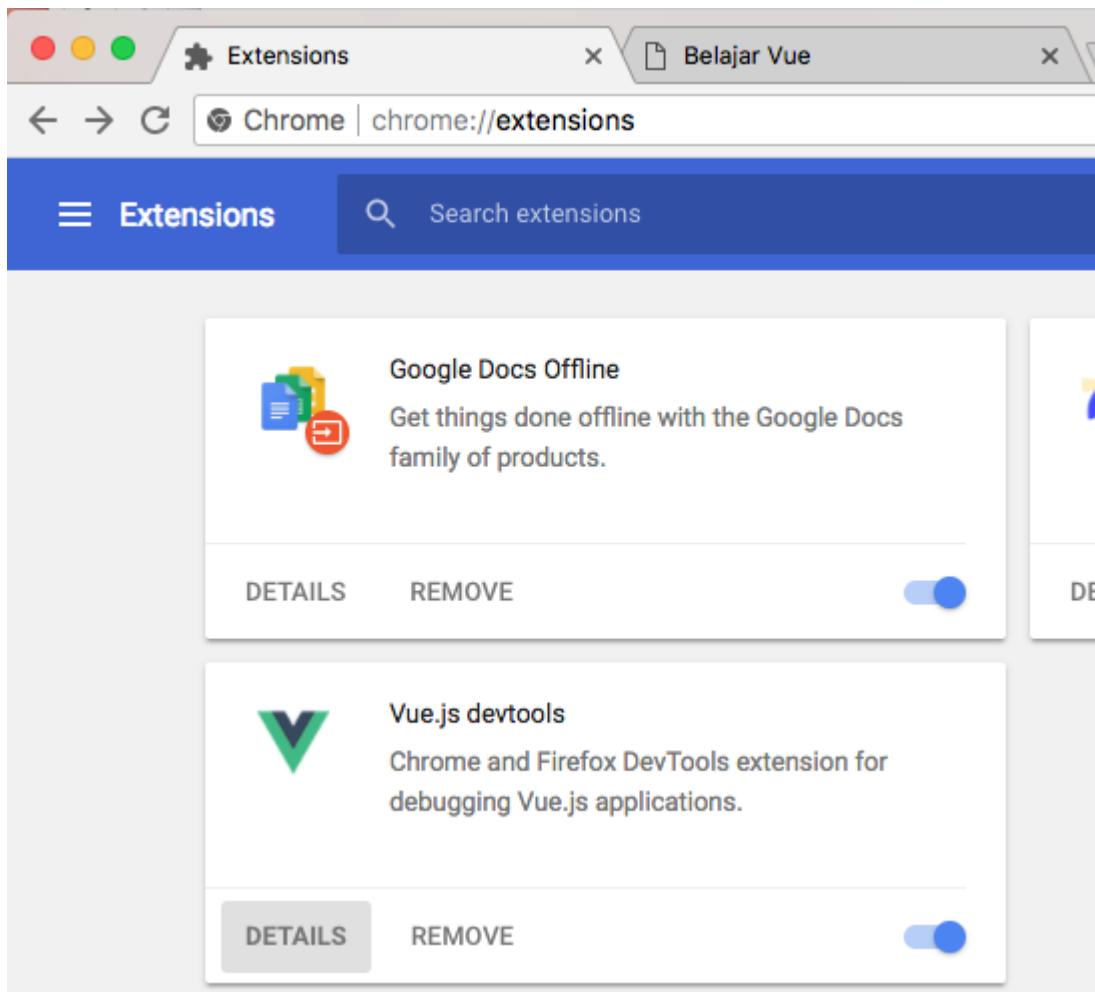
- Buka chrome web store, search Vue, maka pilih devtools



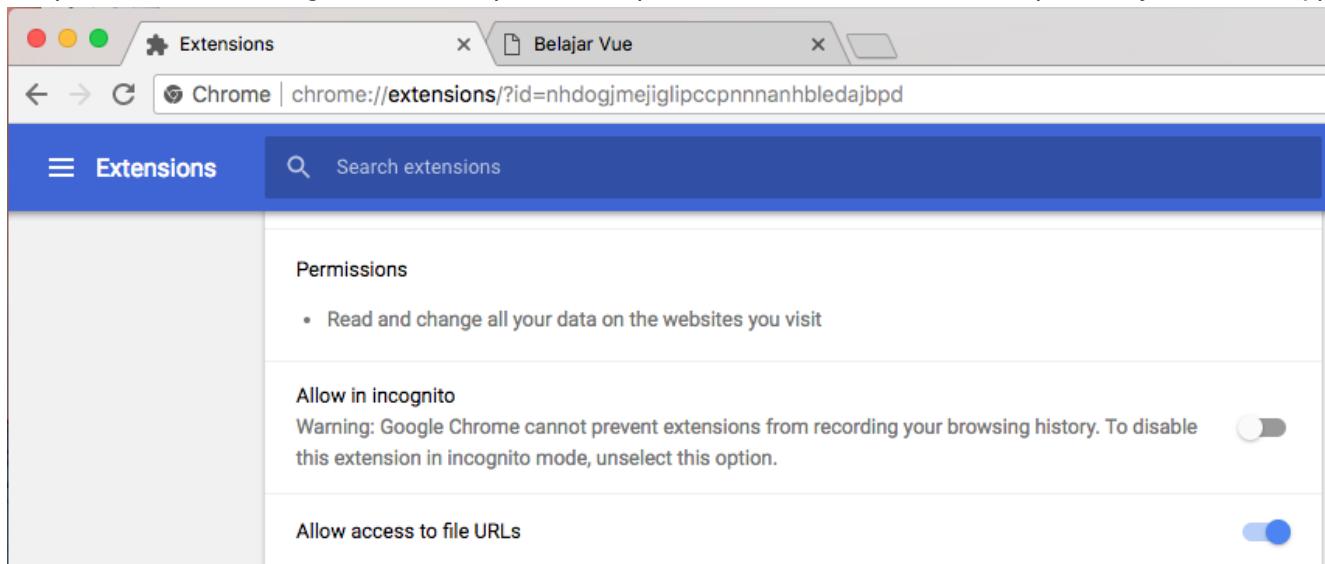
- Klik tombol Add to Chrome, lalu Add extension



- Akses chrome://extensions untuk memastikan bahwa extension telah terinstalasi



- Pilih tombol Detail pada Vue.js devtools, kemudian aktifkan Allow access to file URLs tetap bisa digunakan pada protocols file:// (defaultnya http)



Cara menggunakannya, pada Developer Tools - Browser, pilih menu Vue yaitu menu paling kanan

The screenshot shows the Chrome DevTools interface with the 'Vue' tab selected. At the top, it says 'Ready. Detected Vue 2.5.16.' Below that, there's a search bar labeled 'Filter components'. A green bar highlights the path '<Root> = \$vm0'. To the right, under the '<Root>' heading, there's a section for 'data' which includes a 'count: 0' entry.

Getting Started

State atau data pada Vuex disimpan dalam sebuah objek yang disebut dengan `store`. Tidak hanya menyimpan state, namun store juga bertanggung jawab atas perubahan state, dan perubahannya tersebut bersifat reaktif sehingga bisa digunakan untuk memicu render ulang suatu View yang menggunakan state.

Untuk mensimulasikan penggunaan Vuex, mari kita ubah aplikasi counter yang telah dibahas pada bagian awal bab ini, namun kali ini akan menggunakan Vuex.

Pertama, kita perlu tau dulu bahwa state atau data apa saja yang digunakan pada aplikasi counter? dan bagaimana perubahan (*mutation*) state-nya?

Pada kasus ini, tentu saja state yang terlibat adalah data `counter` dan perubahannya adalah `increment` dari state.

Kedua, kita buat objek Vuex store berdasarkan poin pertama di mana ada dua properti yang akan kita gunakan yaitu state dan mutation.

```

1 const store = new Vuex.Store({
2   state: {
3     counter: 0
4   },
5   mutations: {
6     increment(state){
7       state.counter++
8     }
9   }
10 })

```

Yap, analogi kode di atas mirip dengan data & method pada objek Vue.

Pada kode store di atas, kita bisa melakukan perubahan data counter melalui fungsi `increment`, caranya dengan meng-commit mutation `store.commit('increment')`. Adapun untuk mengakses data state counter, kita bisa gunakan perintah `store.state.counter`.

Ketiga, kita implementasikan perubahan state tersebut langsung pada objek Vue menggunakan store. Untuk itu, kita bisa menggunakan properti computed untuk mengakses nilai state counter, hal ini dilakukan karena state itu reactive sehingga properti computed lebih tepat digunakan untuk memonitor perubahan state, disamping itu untuk memudahkan kita menampilkannya pada template.

Tambahkan juga satu method increment yang berfungsi mengirimkan sinyal commit kepada mutations increment di store. Kira-kira kodenya menjadi sebagai berikut.

```

1 // koding VuexStore pada bagian dua di sini
2 // const store = new Vuex.Store({ ... }
3
4 new Vue({
5   el: '#app',
6   // local state as computed
7   computed: {
8     counter(){
9       return store.state.counter
10    }
11  }
12  // view
13  template: `
14    <div>
15      {{ counter }}
16      <button @click="increment()"> + </button>
17    </div>
18  `,
19  // actions
20  methods: {
21    increment () {
22      store.commit('increment')
23    },
24  },
25})

```

Catatan: pastikan pustaka Vuex telah kamu masukkan dalam file HTML.

Mari kita lihat hasilnya.

The screenshot shows the Vuex DevTools extension in a browser. The address bar displays the URL `localhost/book-laravel/vuex/index.html`. The Vue tab is active in the devtools toolbar. On the left, a list of mutations is listed with their timestamps: 'Base State' at 01:50:00, followed by three 'increment' mutations at 02:20:43, 02:20:44, and 02:20:44. The third 'increment' mutation is highlighted with a green background. On the right, the state tree is shown, with 'state' expanded to reveal 'counter: 3' and 'mutation' expanded to show 'payload: undefined' and 'type: "increment"'.

Pada gambar di atas penulis mengklik button + tiga kali, sehingga state counter bernilai 3 dan setiap perubahan statenya tercatat pada devtools Vue bahkan sampai waktunya juga tercatat, hal ini terjadi karena kita menggunakan perintah `commit store.commit('increment')` untuk melakukan increment.

Catatan: data state merupakan data sementara yang tersimpan dalam memori yang dialokasikan untuk variabel javascript (non persisted), artinya ketika browser di refresh maka data state akan tereset ke data awalnya. Ke depan jika diperlukan, kita bisa menggunakan client storage (misal: local storage) untuk menyimpan data state

Pada konsep Vuex, perubahan state sebaiknya hanya dilakukan oleh mutations supaya lebih mudah dalam tracking perubahan state. Meskipun perubahan state diluar mutation tetap bisa dilakukan, misalnya dengan perintah `store.state.counter++` namun hal ini melanggar pattern Vuex dan akan menyulitkan kita sendiri ketika aplikasi kita telah kompleks.

Jika kita aktifkan mode strict pada Vuex, maka perubahan state diluar mutation akan menimbulkan pesan warning pada console browser, tambahkan properti `strict: true` pada store.

```

1 const store = new Vuex.Store({
2     strict: true,
3     state: {
4         counter: 0
5     },
6     ...
7 })

```

Kemudian ubah method increment menjadi sebagai berikut.

```

1 new Vue({
2   ...
3   // actions
4   methods: {
5     increment () {
6       //store.commit('increment')
7       store.state.counter++
8     },
9   },
10 })

```

Mari kita lihat hasilnya.

The screenshot shows a browser window with the title 'Vuex'. The address bar contains 'localhost/book-laravel/vuex/index.html'. The main content area of the browser is blank. Below the browser is the Chrome DevTools interface, specifically the 'Console' tab. The console output shows two error messages:

- Vue warn]**: Error in callback for watcher "function () { return this._data.\$\$state }": "Error: [vuex] Do not mutate vuex store state outside mutation handlers." [vue.js:597](#)
- Error: [vuex]**: Do not mutate vuex store state outside mutation handlers. [vue.js:1743](#)

The stack traces for these errors are listed below each message, tracing back through various Vue.js and vuex.js functions.

Disamping itu perubahan state tidak tercatat pada devtools Vue

The screenshot shows a browser window with the title 'Vuex'. The address bar contains 'localhost/book-laravel/vuex/index.html'. Below the address bar is a toolbar with icons for refresh, back, forward, and search. The main content area is a developer tools interface for Vuex. At the top, there are tabs for 'Elements', 'Console', 'Sources', 'Network', 'Performance', 'Memory', and 'Vue'. The 'Vue' tab is active. Below the tabs is a search bar with placeholder 'Filter mutations' and a red circular icon. To the right of the search bar is another search bar with placeholder 'Filter inspected state' and a download icon. The main pane is divided into two sections: 'Base State' (timestamp: 01:50:00) and 'state' (mutation: counter: 0). A sidebar on the left has a green header 'Base State'.

Perhatian: jangan aktifkan mode strict pada saat production! Mode ini akan memonitor kondisi state secara menyeluruh untuk mendeteksi perubahan state diluar mutation, tentu beban aplikasi akan bertambah.

Mengakses Store Via Component

Untuk mengakses store via objek Vue kita memang bisa menggunakan perintah `store.state.counter` atau `store.commit('increment')`, sebab Vuex store dideklarasikan dalam satu file yang sama dengan objek Vue. Lalu bagaimana jika kita menggunakan component yang terpisah filenya dengan Vuex store?

Mari kita simulasikan, buat component baru misalnya bernama `Hello` yang berfungsi menampilkan state counter (nama filenya: `Hello.js`)

```

1  export const Hello = {
2      template: `
3          <p>
4              State counter pada hello :
5              {{ counter }}
6          </p>
7      `,
8      computed: {
9          counter(){
10             return store.state.counter
11         }
12     }
13 }
```

Kemudian pada file utama, kita import component ini.

```
1 <script type="module">
2 import { Hello } from "./Hello.js"
```

Lalu pada objek Vue, registerkan component Hello serta ubah properti template dengan menambahkan elemen <hello>, adapun kode lainnya masih tetap sama.

```
1 new Vue({
2   el: '#app',
3   components: {
4     'hello': Hello
5   },
6   ...
7   // view
8   template: `
9     <div>
10       {{ counter }}
11       <button @click="increment()"> + </button>
12       <hello></hello>
13     </div>
14   `,
15   ...
16 })
17 </script>
```

Hasilnya akan muncul error yang menyebutkan bahwa varaiel store tidak didefinisikan di dalam component Hello.

The screenshot shows a browser window titled "Vuex" with the URL "localhost/book-laravel/vuex/index.html". The developer tools console tab is active, showing the following error messages:

- [Vue warn]: Error in render: "ReferenceError: store is not defined"** (vue.js:597)
found in
---> <Hello>
<Root>
- ReferenceError: store is not defined** (vue.js:1743)
at VueComponent.counter (Hello.js:9)
at Watcher.get (vue.js:3140)
at Watcher.evaluate (vue.js:3247)
at Proxy.computedGetter (vue.js:3503)
at Proxy.eval (eval at createFunction (vue.js:10667), <anonymous>:2:71)
at VueComponent.Vue._render (vue.js:4535)
at VueComponent.updateComponent (vue.js:2788)
at Watcher.get (vue.js:3140)
at new Watcher (vue.js:3129)
at mountComponent (vue.js:2795)

At the bottom of the console, it says "vue-devtools Detected Vue v2.5.16" and "backend.js:1".

Adapun perubahan state pada devtools tetap tercatat karena perintah commit state dilakukan di objek Vue bukan di component Hello

The screenshot shows a browser window with the URL localhost/book-laravel/vuex/index.html. The browser title bar says "Vuex". The bottom of the browser has developer tools tabs: Elements, Console, Sources, Network, Performance, Memory, and Vue. The Vue tab is selected. On the left, there's a list of mutations with timestamps: "Base State" at 01:50:00, "increment" at 02:53:43, and another "increment" at 02:53:43 which is highlighted in green. On the right, it shows the state tree with "state": { "counter": 2 } and a mutation object: { "payload": undefined, "type": "increment" }. There are also buttons for filtering mutations and state, and icons for saving and sharing.

Untuk mengatasi hal ini, Vuex menyediakan mekanisme untuk menginjek atau meregister Vuex store pada objek Vue sehingga bisa digunakan pada seluruh component dibawah objek Vue tersebut. Hal ini sebenarnya hampir sama dengan register component secara local.

```

1 new Vue({
2   el: '#app',
3   store, // tambahkan ini atau store: store
4   components: {
5     'hello': Hello
6   },
7   ...
8 }
9 )

```

Kemudian pada component, kita bisa gunakan perintah `this.$store` untuk mengakses store.

```

1 this.$store.state.counter

```

Masalah selesai 😊.

Getters

Vuex store sebenarnya juga memiliki properti getters sehingga pemanggilan state tidak langsung menembak state-nya (`store.state.counter`) melainkan melalui perantaraan fungsi getter di store tersebut.

```

1 var store = new Vuex.Store({
2   ...
3   getters: {
4     counter: state => state.counter
5
6     /*
7      counter(state){
8        return state.counter
9      }
10    */
11  }
12})

```

Untuk memanggilnya pada computed, kita bisa menggunakan perintah `store.getters.counter`. Oleh karenanya kita bisa ubah cara pemanggilannya pada properti computed

```

1 computed: {
2   counter(){
3     return store.getters.counter
4   }
5 },

```

Bukankah sama saja atau bahkan overkoding kalo kita menggunakan getters? Ya ini supaya kode Vuex kita lebih terstruktur saja, di samping itu kadangkala implementasi getters ini tidak hanya mereturn satu state saja melainkan beberapa state sekaligus dan bisa juga ditambahkan dengan operasi tertentu sehingga akan lebih efisien jika kita buatkan fungsi getters sendiri.

Mutations

Mutations merupakan kumpulan fungsi untuk memanipulasi state atau bisa juga disebut sebagai setter. Tiap fungsi mutations memiliki minimal dua hal yaitu type dan handler, di mana type merupakan nama fungsinya dan handler merupakan state.

```

1 increment (state) {
2   state.counter++
3 }
4
5 // atau versi es6 arrow function
6 increment: state => state.counter++
7
8 // untuk mengakses
9 store.commit('increment')

```

Kita juga dizinkan menambahkan argumen (payload) pada fungsi ini.

```

1 increment (state, n) {
2   state.counter += n
3 }
4
5 // untuk mengaksesnya
6 store.commit('increment', 10)

```

Catatan: fungsi dalam mutation seharusnya bersifat synchronous supaya mudah dalam memantau perubahan state.

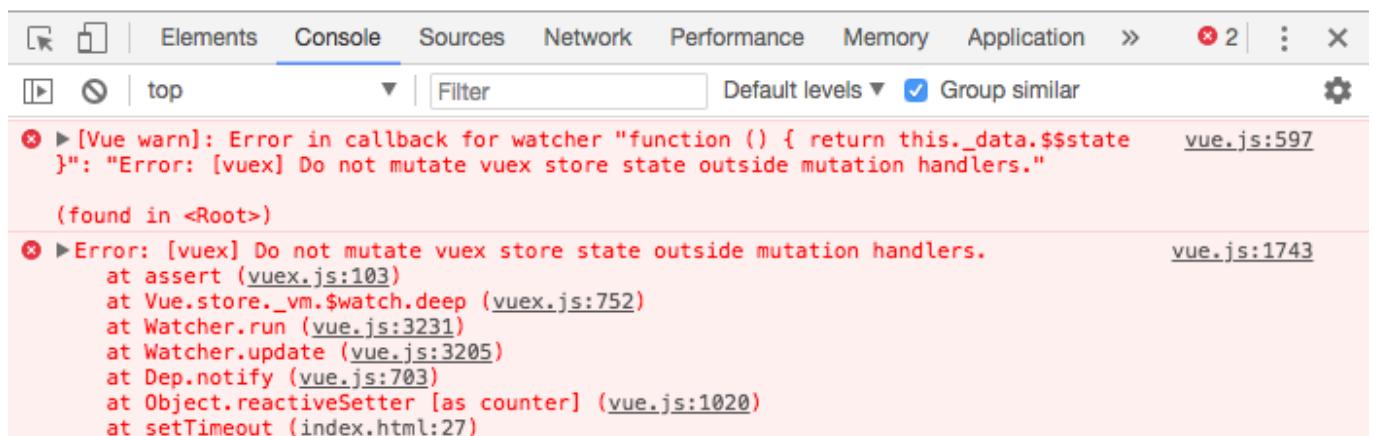
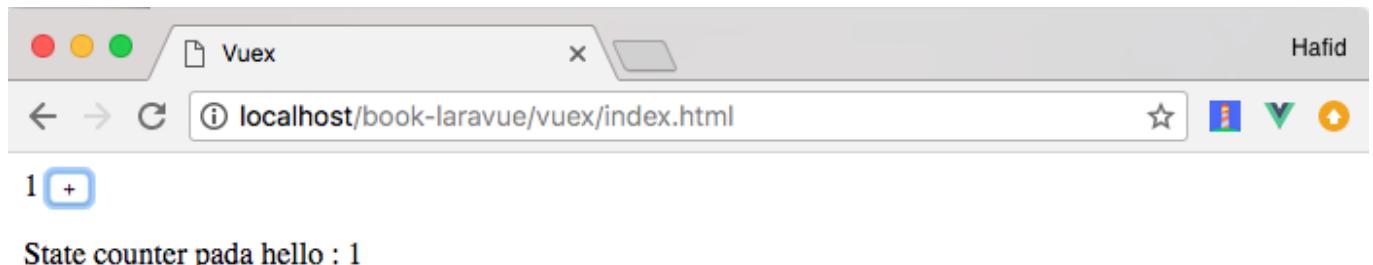
Berikut ini contoh simulasi asynchronous fungsi setName dengan menggunakan setTimeout.

```

1 mutations: {
2     //increment: state => state.counter++
3     increment(state) {
4         setTimeout(()=>{
5             state.counter++
6         }, 1000)
7     }
8 },

```

Hasilnya akan muncul error dan state tidak berubah.



Artinya, perubahan state pada mutation harus dilakukan pada saat itu juga, tidak boleh menunggu barang satu detik pun

Info: menunggu itu berat, biar aku aja 😊

Actions

Vuex juga memiliki properti actions. Actions sebenarnya mirip dengan mutations, namun perbedaannya adalah:

- Actions bertugas meng-commit mutations.
- Actions mendukung operasi asynchronous.

```

1 var store = new Vuex.Store({
2   strict: true,
3   state: {
4     counter: 0
5   },
6   mutations: {
7     increment: state => state.counter++
8   },
9   actions: {
10     increment(context){
11       context.commit('increment')
12     }
13     // atau
14     /*
15     increment: (context) => {
16       context.commit('increment')
17     }
18     atau
19     increment: ({commit}) => {
20       commit('increment')
21     }
22     */
23   },
24   getters: {
25     counter: state => state.counter
26   }
27 }
28 )
29

```

Lalu cara memanggilnya pada objek Vue atau component, yang sebelumnya menggunakan perintah `store.commit('increment')` maka kita ubah menjadi `store.dispatch('increment')`.

```

1 new Vue({
2   el: '#app',
3   store,
4   components: {
5     'hello': Hello
6   },
7   computed: {
8     counter(){
9       return store.getters.counter
10    }
11  },
12  // view
13  template: `
14    <div>
15      {{ counter }}
16      <button @click="increment()"> + </button>
17      <hello></hello>
18    </div>
19  `,

```

```

20     methods: {
21         increment () {
22             // store.commit('increment')
23             store.dispatch('increment')
24         },
25     },
26 }

```

Asynchronous Actions

Berbeda dengan mutation, fungsi-fungsi pada actions bisa dibuat asynchronous, misalnya pada kasus pemanggilan data dari server yang tentu membutuhkan waktu. Caranya, kita menggunakan Promise sebagai return value dari fungsi pada actions.

Wah, apa itu promise? promise artinya janji, ini sebuah term di JS yang memungkinkan kita menunggu suatu proses yang dilakukan secara asynchronous dan kita dipastikan akan mendapatkan feedback balik dari proses itu, baik berhasil maupun gagal.

Adapun kerangka dari objek Promise adalah sebagai berikut.

```

1   return new Promise((resolve, reject) => {
2       //if success
3       resolve()
4       // if error
5       reject()
6   })

```

OK, untuk mensimulasikan asynchronous action maka kita bisa gunakan data JSON buku (dummy) sebagaimana yang telah digunakan pada bab-bab terdahulu.

```

1  [
2      {
3          "id": 99,
4          "title": "C++ High Performance",
5          "description": "Write code that scales across CPU
6          registers, multi-core, and machine clusters",
7          "authors": "Viktor Sehr, Björn Andrist",
8          "publish_year": 2018,
9          "price": 100000,
10         "image": "c++-high-performance.png"
11     },
12     {
13         "id": 100,
14         "title": "Mastering Linux Security and Hardening",
15         "description": "A comprehensive guide to mastering the
16         art of preventing your Linux system from getting compromised",
17         "authors": "Donald A. Tevault",
18         "publish_year": 2018,
19         "price": 125000,
20         "image": "mastering-linux-security-and-hardening.png"
21     },
22     {
23         "id": 101,
24         "title": "Mastering PostgreSQL 10",

```

```

25     "description": "Master the capabilities of PostgreSQL 10
26     to efficiently manage and maintain your database",
27     "authors": "Hans-Jürgen Schönig",
28     "publish_year": 2016,
29     "price": 90000,
30     "image": "mastering-postgresql-10.png"
31   },
32   {
33     "id": 102,
34     "title": "Python Programming Blueprints",
35     "description": "How to build useful, real-world
36 applications in the Python programming language",
37     "authors": "Daniel Furtado, Marcus Pennington",
38     "publish_year": 2017,
39     "price": 75000,
40     "image": "python-programming-blueprints.png"
41   }
42 ]

```

Data JSON ini kemudian bisa kita hosting ke <https://jsonbin.io> untuk sekedar uji coba saja.

Pada website <https://jsonbin.io>, masukkan data JSON tersebut pada kolom input yang disediakan lalu klik button Create maka akan digenerate API dari data tersebut yang bisa kita akses (dalam hal ini) melalui Access URL -> <http://api.jsonbin.io/b/5b42acd44d5ea95c8ba22392>

Kita bisa mencoba API tersebut dengan mengaksesnya melalui browser atau tools Postman.

```
[{"image": "c++-high-performance.png", "price": 100000, "publish_year": 2018, "authors": "Viktor Sehr, Bj\u00f6rn Andrist", "description": "Write code that scales across CPU registers, multi-core, and machine clusters", "title": "C++ High Performance", "id": 99}, {"image": "mastering-linux-security-and-hardening.png", "price": 125000, "publish_year": 2018, "authors": "Donald A. Tevault", "description": "A comprehensive guide to mastering the art of preventing your Linux system from getting compromised", "title": "Mastering Linux Security and Hardening", "id": 100}, {"image": "mastering-postgresql-10.png", "price": 90000, "publish_year": 2016, "authors": "Hans-J\u00fcrgen Sch\u00f6nig", "description": "Master the capabilities of PostgreSQL 10 to efficiently manage and maintain your database", "title": "Mastering PostgreSQL 10", "id": 101}, {"image": "python-programming-blueprints.png", "price": 75000, "publish_year": 2017, "authors": "Daniel Furtado, Marcus Pennington", "description": "How to build useful, real-world applications in the Python programming language", "title": "Python Programming Blueprints", "id": 102}]
```

Setelah data json siap, langkah berikutnya kita siapkan kerangka store menjadi sebagai berikut.

```
1 var store = new Vuex.Store({
2     strict: true,
3     state: {
4         books: []
5     },
6     mutations: {
7         setBooks(state, books){
8             state.books = books
9         }
10    },
11    actions: {
12        getBooks ({ commit }) {
13            return new Promise((resolve, reject) => {
14                //if success
15                resolve()
16                // if error
17                reject()
18            })
19        }
20    },
21    getters: {
22        books: state => state.books
23    }
24 })
```

Kita siapkan dulu kerangka state books, mutations setBooks, action getBooks dan getters books.

Kita masih akan menggunakan pustaka XMLHttpRequest untuk merequest data JSON buku dan kodennya tersebut akan kita bungkus menggunakan Promise pada actions getBooks() agar mendukung asynchronous. Kira-kira kode action getBooks menjadi sebagai berikut:

```
1 getBooks ({ commit }) {
2     return new Promise((resolve, reject) => {
3         var xhr = new XMLHttpRequest();
4         xhr.open("GET",
5         "http://api.jsonbin.io/b/5b42acd44d5ea95c8ba22392");
6         xhr.onload = function () {
7             if (this.status >= 200 && this.status < 300) {
8                 commit('setBooks', JSON.parse(xhr.response) )
```

```

9     resolve(xhr.response);
10    } else {
11        reject({
12            status: this.status,
13            statusText: xhr.statusText
14        });
15    }
16  };
17  xhr.onerror = function () {
18      reject({
19          status: this.status,
20          statusText: xhr.statusText
21      });
22  };
23  xhr.send();
24 }

```

Kode di atas akan merequest data buku pada jsonbin.io dengan metode GET `xhr.open("GET", "http://api.jsonbin.io/b/5b42acd44d5ea95c8ba22392")`. Kemudian jika permintaan data sukses `if (this.status >= 200 && this.status < 300)` maka data tersebut akan dicommit ke mutation `setBooks`, tentunya karena data berformat JSON maka kita perlu parsing atau konversi ke bentuk objek atau array Javascript menggunakan `JSON.parse`

```

1 commit('setBooks', JSON.parse(xhr.response) )
2 resolve(xhr.response);

```

Kemudian pada objek Vue atau component, tepatnya pada hook created kita bisa melakukan dispatch action `getBooks` dan pada computed kita perlu buat fungsi `books` yang memanggil getters `books` pada store, sebagai berikut.

```

1 new Vue({
2   el: '#app',
3   store,
4   computed: {
5     books(){
6       return store.getters.books
7     }
8   },
9   created() {
10     store.dispatch('getBooks')
11     .then((response) => {
12       console.log('result: ', response)
13     })
14     .catch((error) => {
15       console.log('error: ', error)
16     })
17   }
18 })

```

Selanjutnya pada template, kita gunakan teknik list rendering.

```

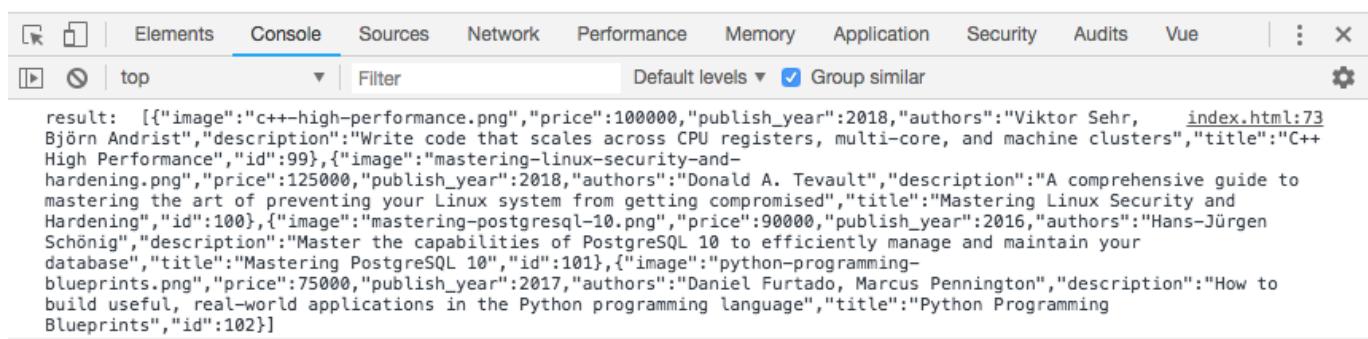
1 <div id="app">
2   <ul v-for="book in books">
3     <li>{{ book.title }}</li>
4   </ul>
5 </div>

```

Waktunya ujicoba.



- C++ High Performance
- Mastering Linux Security and Hardening
- Mastering PostgreSQL 10
- Python Programming Blueprints



Yeay! berhasil.

Menangani Two Way Data Binding

Pada bab sebelumnya kita telah membahas two way data binding dengan menggunakan directive v-model, di mana v-model merujuk ke variabel pada properti data. Kita tau bahwa sifat dari v-model itu ada dua yaitu getter dan setter, artinya ketika kita mengisi teks pada field input maka variabel data akan diset sesuai dengan teks isian kita (setter), sebalik jika variabel data diubah maka isian field input juga akan mengikuti perubahan pada variabel data tersebut (getter).

Lalu bagaimana menghubungkan dengan state padahal state pada objek di definisikan sebagai getter di properti computed bukan di properti data?

Baik, mari kita simulasikan dengan membuat state `name` dan field inputnya

```

1 var store = new Vuex.Store({
2   strict: true, // supaya warning muncul
3   state: {
4     name: 'Hafid'
5   },
6   mutations: {
7     setName: (state, name) => {
8       state.name = name
9     }
10   },
11   /*
12   // boleh juga fungsi biasa

```

```

13     setName (state, name) {
14         state.name = name
15     }
16     */
17 },
18 getters: {
19     name: state => state.name
20 }
21 )
22
23 new Vue({
24     el: '#app',
25     store,
26     computed: {
27         name(){
28             return store.getters.name
29         }
30     },
31     template: `
32         <div>
33             <input v-model='name'>
34         </div>
35     `,
36 })

```

Mari kita lihat hasilnya..

Hafid

Elements Console Sources Network Performance Memory Application >

top Filter Default levels ▾ Group similar

Wah berhasil!

Tapi!! ketika field input kita ketikkan suatu teks ternyata muncul error pada console.

Intinya properti name tidak memiliki setter. Yap, karena properti computed itu secara default sifatnya hanya getter saja.

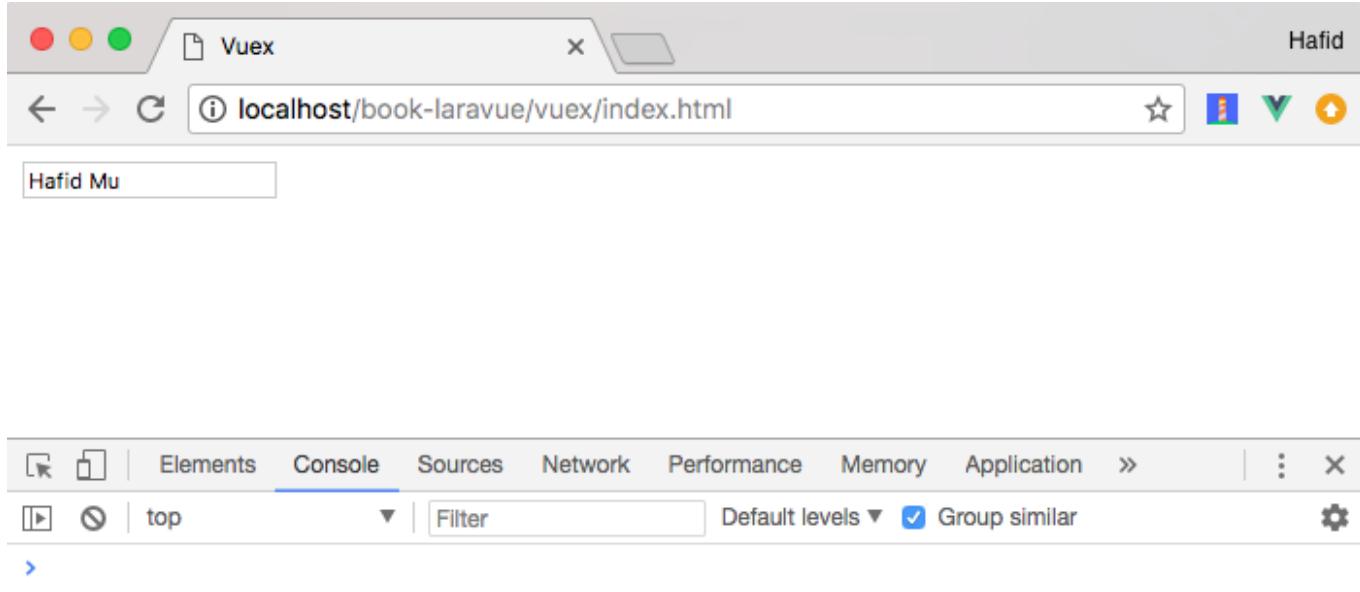
Oleh karena itu, kita perlu buat atau definisikan getter dan setter pada computed name, seperti ini bentuknya.

```

1 // ...
2 new Vue({
3   el: '#app',
4   computed: {
5     name: {
6       get () {
7         return store.getters.name
8       },
9       set (value) {
10         store.commit('setName', value)
11       }
12     }
13   },
14   template: `
15     <div>
16       <input v-model='name'>
17     </div>
18   `,
19 })

```

Mari kita lihat hasilnya



Tidak ada lagi error, dan lebih dari itu, jika kita lihat di devtools Vue maka perubahan state akan tercatat.

Mutation	Time
Base State	01:50:00
setName	03:59:19
setName	03:59:19
setName	03:59:30

Filter mutations:

Filter inspected state:

state

```
name: "Hafid Mu"
```

mutation

```
payload: "Hafid Mu"
type: "setName"
```

Kesimpulan

State management merupakan manajemen data atau state aplikasi secara terpusat sehingga memudahkan sharing data dan manipulasinya antar component dalam aplikasi. Setiap perubahan state hanya boleh dilakukan secara langsung oleh mutation, sehingga memudahkan dalam tracking state. Operasi pada mutation harus bersifat synchronous, sedangkan operasi pada actions bisa asynchronous menggunakan Promise atau async/await.

Jika mengikuti best practice-nya maka interaksi terhadap state pada suatu component hanya sebatas dispatch action dan getters saja, bukan langsung mengakses state atau mutationnya.

Masih ada beberapa topik menarik terkait state management yang belum dibahas pada bab ini, beberapa diantaranya akan dibahas secara bertahap pada bagian selanjutnya.

Jika pada bab pertama sampai dengan bab ini kita belajar tentang dasar-dasar Vue dan bagaimana menggunakan fitur-fiturnya maka pada bab selanjutnya kita akan belajar tentang bagaimana mempersiapkan berbagai hal terkait pengembangan projek aplikasi berbasis Vue.

Sebentar lagi itu gak bakal lama

Scaffolding Application

Intro

Pada bab ini kita akan belajar tentang bagaimana mempersiapkan pengembangan projek aplikasi berbasis Vue yang meliputi gambaran umum projek aplikasi, instalasi dan konfigurasi tools-tool yang diperlukan dalam pengembangan, hingga penyusunan kerangka aplikasi.

Sebelum kita memulai maka kita akan membuat satu folder baru pada PC kita misalnya bernama `vue-projects`, di mana pada folder inilah nantinya projek-projek aplikasi berbasis Vue akan kita tempatkan, sehingga akan memudahkan kita dalam mengelolanya.

Briefing Projek

Projek aplikasi Vue yang penulis angkat untuk menjadi studi kasus pada buku ini adalah aplikasi toko buku berbasis mobile web. Projek studi kasus ini dipilih karena umumnya bisnis prosesnya telah sama-sama kita ketahui sehingga diharapkan pembahasan bisa lebih fokus kepada hal-hal teknisnya.

Secara umum, bisnis proses dari projek ini adalah user aplikasi dapat melihat tampilan daftar buku, melakukan pencarian buku berdasarkan kategori tertentu, melihat detail dari buku, hingga memutuskan untuk memasukkan buku tersebut ke keranjang belanjanya atau mencari buku yang lain.

Setelah belanjanya dirasa cukup maka user dapat melakukan checkout pemesanan yang kemudian memasukkan informasi profil dan alamat pengiriman barang. Namun jika user sudah pernah melakukan pemesanan sebelumnya, maka user bisa login dan memilih data alamat pengiriman barang yang pernah dimasukkan sebelumnya atau memasukkan alamat baru.

Setelah proses pemesanan selesai maka user akan melihat total belanja dan kemana harus transfer uangnya. Pada projek ini, proses pembayaran tidak di-cover dalam pembahasan buku ini, sehingga pembayaran dilakukan secara manual dan belum terintegrasi.

Projek aplikasi ini berbasis mobile web artinya aplikasi ini berbasis web sebagaimana umumnya namun fokus pada tampilan versi mobile, sehingga performa dan user interface benar-benar diperhatikan agar mobile friendly. Di samping itu aplikasi ini dapat diinstalasi pada gadget user layaknya native aplikasi sehingga akan memudahkan user untuk membuka aplikasi ini dilain kesempatan.

Projek ini hanya fokus pada bisnis proses dari sisi user sedangkan terkait manajemen buku, manajemen pemesanan yang dilakukan oleh admin atau pemilik toko buku tidak dicover dalam buku ini melainkan dibahas pada buku Laravel yang di-launching bersamaan dengan buku ini. Dengan kata lain, projek yang kita bangun ini adalah projek toko buku dari sisi *frontend*.

Fitur Utama Aplikasi

Berdasarkan uraian di atas, dapat disimpulkan bahwa fitur utama yang akan kita bangun untuk projek aplikasi ini meliputi:

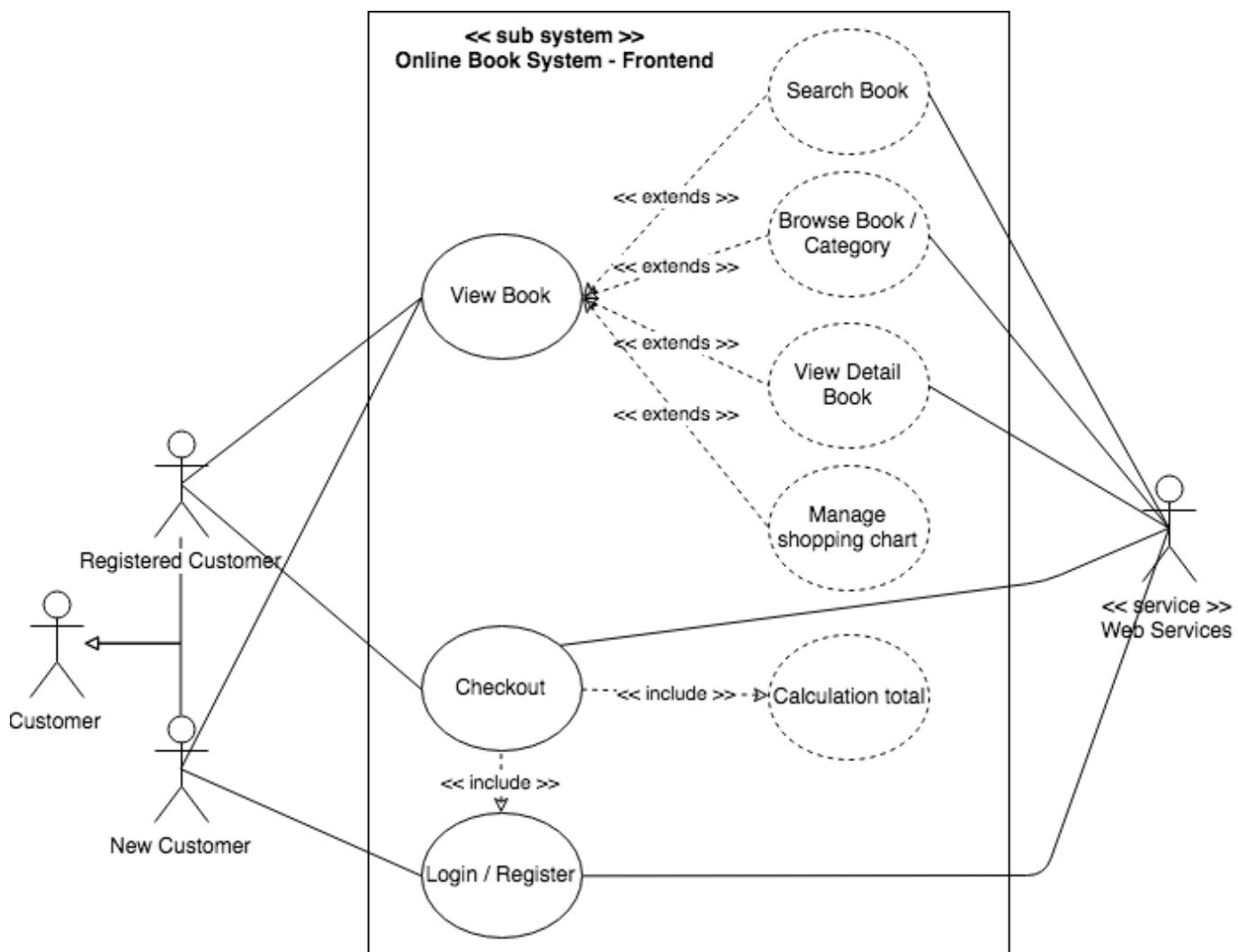
- Tampilan daftar buku (list & detail)
- Keranjang belanja
- Keanggotaan
- Checkout Pemesanan
- Single Page Application (SPA) berkaitan dengan kenyamanan user interface di mobile
- Progressive Web Application (PWA) untuk mengatasi peforma di mobile

Unified Model Language

Untuk lebih memudahkan kita dalam melihat projek ini secara holistic view, maka berikut ini diagram use case yang digambarkan menggunakan format unified model language (UML).

Use Case Diagram

Diagram ini menggambarkan hubungan antara actor (pemeran atau user) dan action (tindakan) yang terlibat dalam sistem.



Catatan: Use case di atas hanya sub sistem (bagian frontend) dari sistem toko buku

Terdapat dua actor utama dalam sistem ini yang berinteraksi langsung dengan sistem yaitu **Customer** dan **Web Service**. Actor **Customer** dibagi menjadi dua yaitu **New Customer** (user yang belum login) dan **Registered Customer** (user yang sudah login). **Registered Customer** dapat melakukan action **View Book** (lihat buku) dan **Checkout**, sedangkan **New Customer** dapat melakukan action **View Book** dan **Login/Register**.

Action **View Book** memiliki beberapa sub action yaitu **Search Book** (pencarian buku), **Browse Book/Category** (pemilihan buku/kategori buku), dan **Manage Shopping Cart** (penambahan data buku ke keranjang belanja, edit dan hapus). Semua sub action **View Book** kecuali **Manage Shopping Cart** terhubung dengan actor **Web Service** sebagai sebuah sub sistem lain yang menyediakan data buku. Adapun data keranjang belanja pada **Manage Shopping Cart** disimpan dalam browser state.

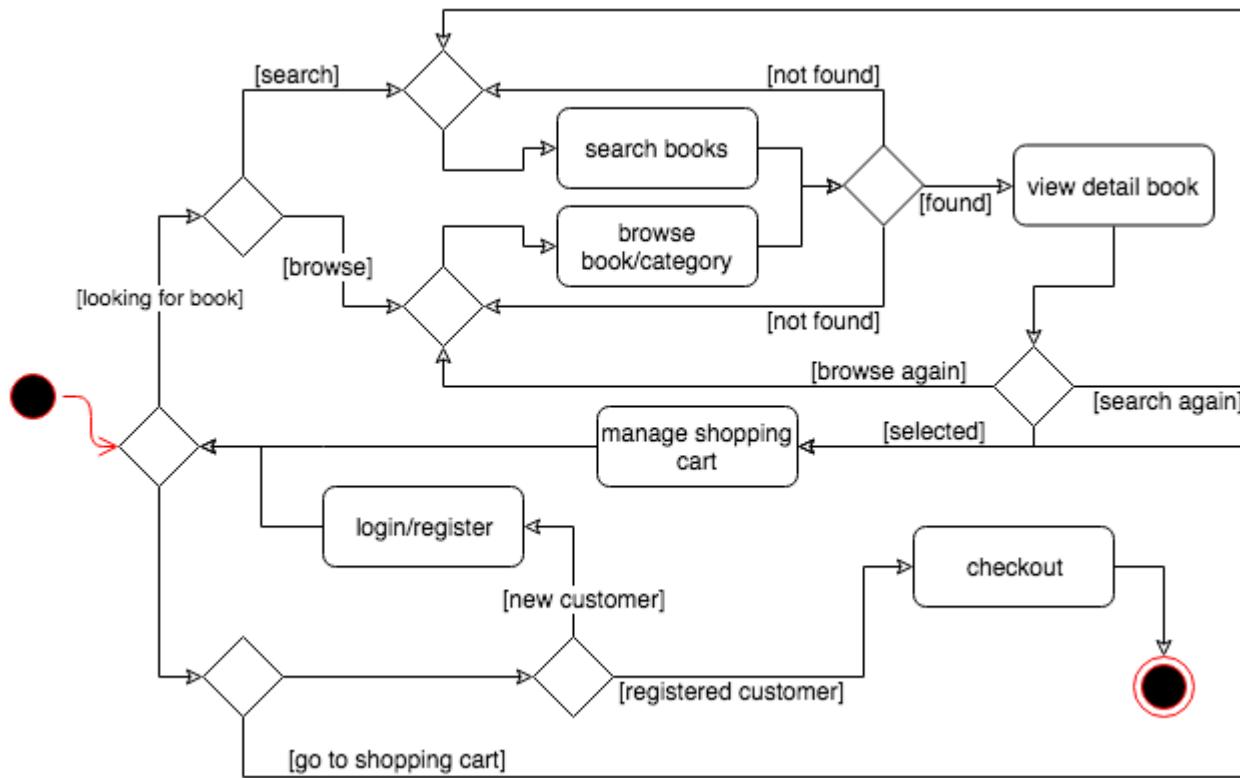
Action **Checkout** memiliki sub action yang sudah include yaitu **Calculation Total** (perhitungan total bayar). Action ini juga terhubung ke **Web Service** yang menjembatani pengiriman data pemesanan buku ke server.

Action terakhir adalah **Login/Register** yang harus dilakukan oleh **Customer** ketika dia hendak melakukan action **Checkout**. Action ini terhubung dengan **Web Service** yang melakukan proses autentikasi atau registrasi data customer.

Catatan: web service adalah aplikasi di sisi backend yang berfungsi menyediakan dan memproses data (buku, pemesanan, dsb) pada sistem aplikasi ini.

Activity Diagram

Diagram ini menggambarkan alur dari setiap aktivitas yang dilakukan pada sistem ini.



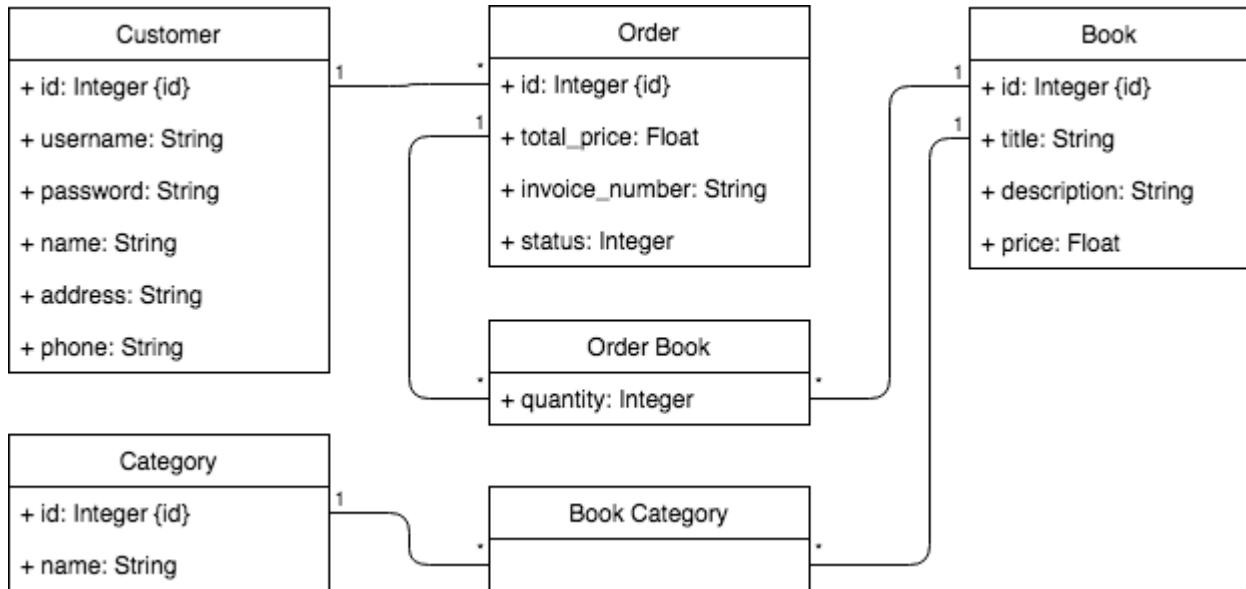
Mula-mula customer dapat melakukan beberapa pilihan aktivitas yaitu search books, browse book/category, manage shopping cart dan login/register. Setelah melakukan aktivitas search books dan browse book/category, jika buku ditemukan maka customer akan melakukan aktivitas view detail book, namun jika buku tidak ditemukan maka customer dapat melakukan kegiatan sebelumnya (looking for book).

Setelah melakukan aktivitas view detail book, customer kemudian dapat memutuskan apakah hendak memesan buku tersebut atau tidak, jika memesan buku maka aktivitas berikutnya masuk ke manage shopping cart, namun jika tidak jadi memesan maka customer dapat melakukan aktivitas pencarian buku kembali.

Jika sudah selesai belanja, maka customer baru atau yang belum login maka akan melakukan aktivitas login/register, sedangkan yang sudah login maka akan melakukan aktivitas checkout.

Class Diagram

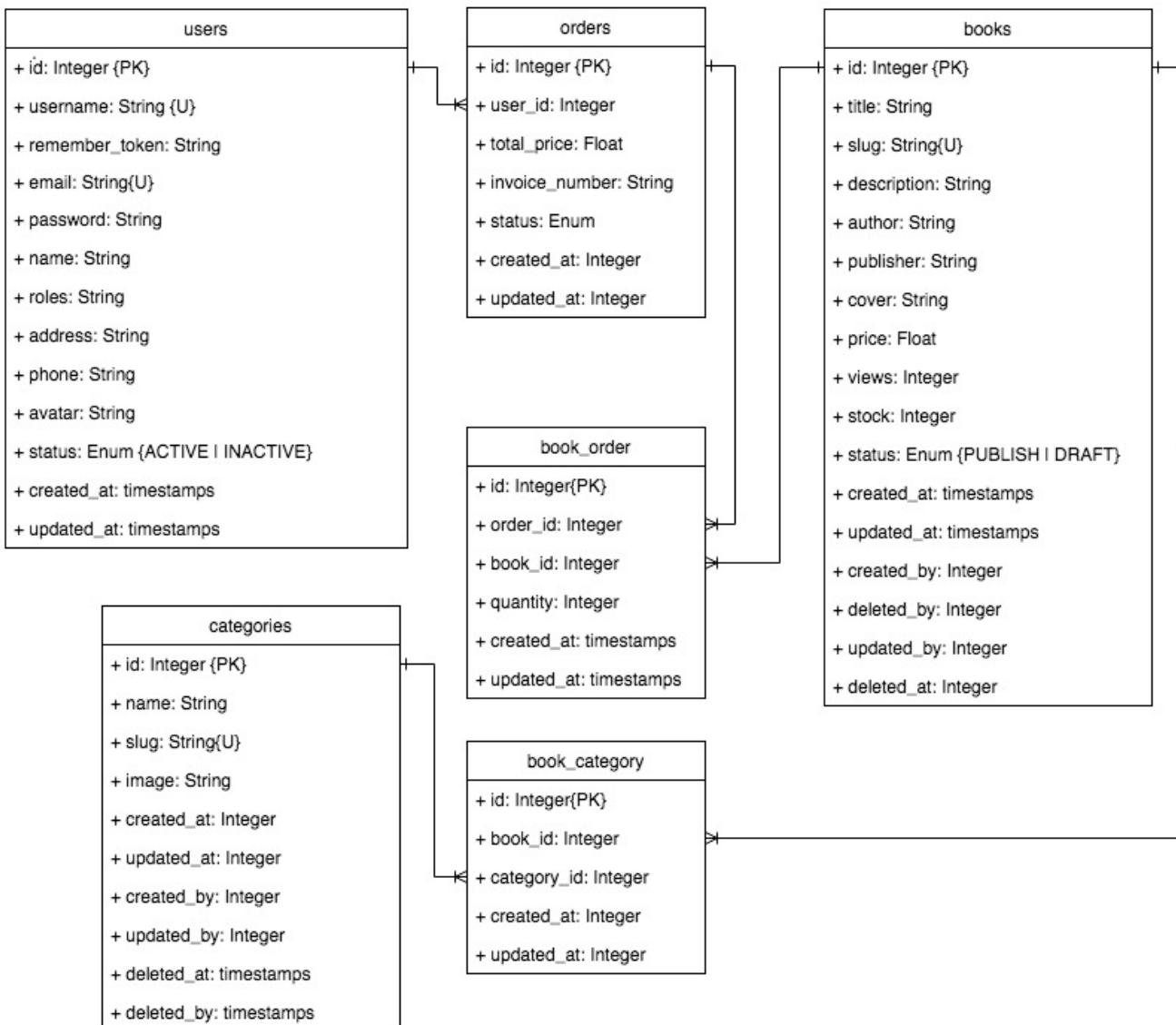
Diagram ini menggambarkan objek-objek & properti utama yang terlibat di dalam sistem beserta hubungannya satu sama lain.



Desain Database

Berdasarkan diagram class yang telah diuraikan sebelumnya serta beberapa penyesuaian maka dapat desain database dapat digambarkan dalam entity relationship diagram (ERD) berikut.

books
+ id: Integer {PK}
+ title: String
+ slug: String{U}
+ description: String
+ author: String
+ publisher: String
+ cover: String
+ price: Float
+ weight: Integer
+ views: Integer
+ stock: Integer
+ status: Enum {PUBLISH DRAFT}
+ created_at: timestamps
+ updated_at: timestamps
+ created_by: Integer
+ deleted_by: Integer
+ updated_by: Integer
+ deleted_at: Integer



Pada contoh ini, penulis menggunakan nama tabel dalam format huruf kecil (lowercase) dan berbentuk plural (jamak).

Preparing Project

Setelah kita mengetahui ruang lingkup, proses bisnis aplikasi serta gambaran umum dari projek yang akan kita bangun, maka kini saatnya kita mulai mempersiapkan tools-tools yang diperlukan selama pengembangan dan menyusun kerangka aplikasi.

Command Line Tools

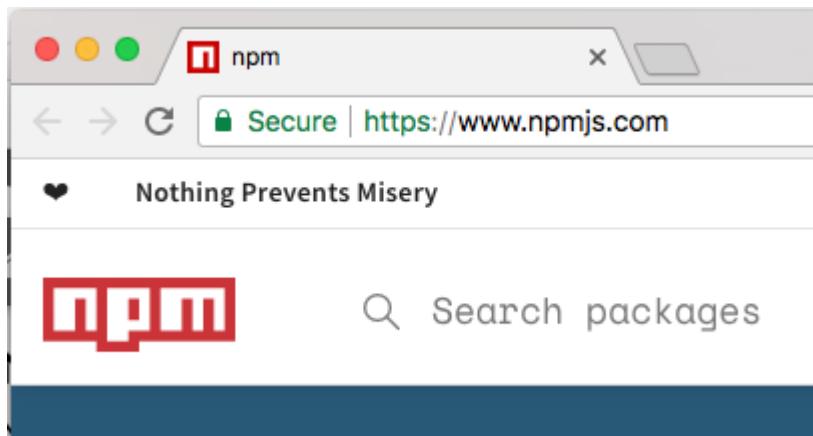
Setelah ini kita akan banyak menggunakan tools berbasis command line, baik untuk menginstalasi pustaka, mengupdate atau menjalankannya. Nah, pada buku ini penulis menggunakan terminal bawaan dari MacOS, bagi kamu pengguna linux tentu tools yang sama bisa kamu gunakan. Bagi pengguna Windows, penulis menyarankan untuk menggunakan powershell bukan command prompt karena secara umum perintah yang digunakan setara dengan perintah pada terminal.

Pilihan lain yang juga sering penulis gunakan pada real projek adalah menggunakan terminal yang terintegrasi dengan Visual Code, dan atau untuk pengguna Windows bisa coba <http://cmder.net>

Package Manager for JS

Package manager merupakan tools berbasis command line (terminal) untuk memudahkan kita dalam mengelola (baca: instalasi, pengecekan dependensi, updating, sharing) pustaka JS yang digunakan pada

aplikasi kita. Saat ini, terdapat dua package manager populer dalam dunia JS yaitu NPM (<https://www.npmjs.com>) dan Yarn (<https://yarnpkg.com>).

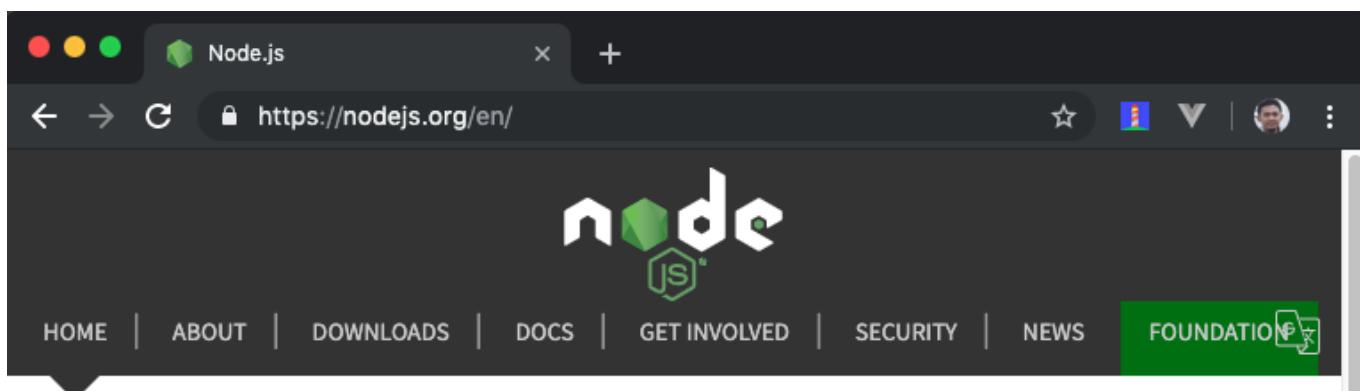


Tools package manager ini diperlukan apabila aplikasi kita menggunakan banyak pustaka yang akan cukup menghabiskan waktu jika kita mengelolanya secara manual. Di samping itu banyak pustaka-pustaka yang ada saat ini hanya menjelaskan cara instalasi menggunakan package manager saja.

Catatan: pada buku ini hanya akan dibahas tentang NPM, adapun penggunaan Yarn sebenarnya kurang lebih sama dengan NPM, silakan merujuk ke website resminya.

Instalasi NodeJS & NPM

Tools npm berbasis NodeJS (<https://nodejs.org>) dan didistribusikan dalam satu paket. NodeJS sendiri merupakan Javascript runtime yang dibangun di atas Chrome's V8 Javascript engine. Oleh karena itu, untuk melakukan instalasi NPM maka kita cukup dengan menginstalasi NodeJS, silakan merujuk ke website resmi di <https://nodejs.org/en>.



Node.js® is a JavaScript runtime built on [Chrome's V8 JavaScript engine](#).

Download for macOS (x64)

10.16.0 LTS

Recommended For Most Users

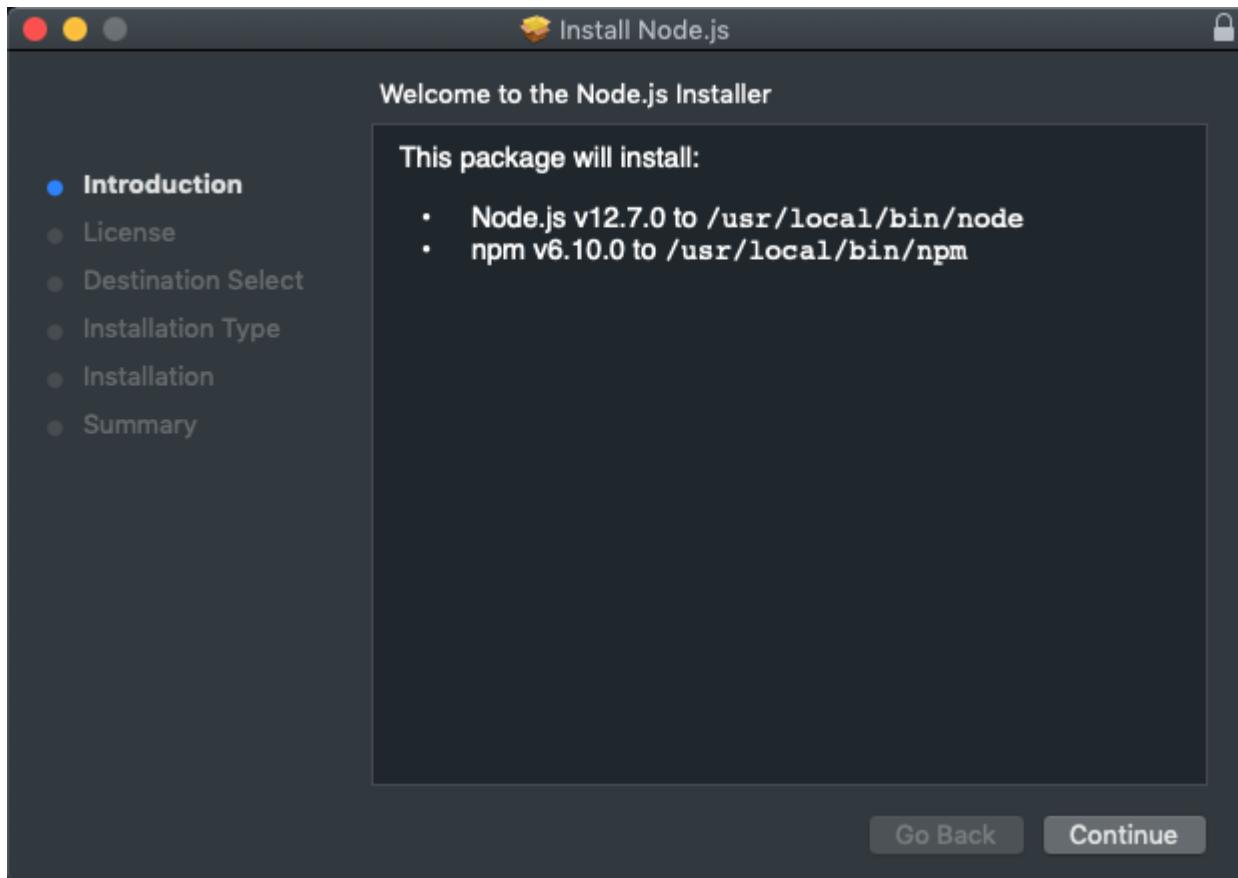
12.7.0 Current

Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

Or have a look at the [Long Term Support \(LTS\) schedule](#).



Setelah melakukan instalasi NodeJS maka untuk memastikannya telah terinstalasi dengan baik, jalankan perintah berikut pada terminal.

```
node -v
```

Pastikan juga NPM telah terinstalasi dengan baik melalui perintah berikut.

```
npm -v
```

The terminal window shows the output of the commands:

```
1. bash
Hafids-MacBook-Pro:vue-projects hafidmukhlasin$ node -v
v12.7.0
Hafids-MacBook-Pro:vue-projects hafidmukhlasin$ npm -v
6.10.0
```

Jika kamu telah menginstalasi NPM sebelumnya maka untuk memastikan bahwa NPM yang kamu instalasi tersebut merupakan versi terbaru, maka kamu bisa menggunakan perintah berikut.

```
npm install npm@latest -g
```

```
1. bash
Hafids-MacBook-Pro:vue-projects hafidmukhlasin$ sudo npm install npm@latest -g
Password:
/usr/local/bin/npx -> /usr/local/lib/node_modules/npm/bin/npx-cli.js
/usr/local/bin/npm -> /usr/local/lib/node_modules/npm/bin/npm-cli.js
+ npm@6.10.2
added 6 packages from 4 contributors, removed 11 packages and updated 35 packages in 9.847s

New patch version of npm available! 6.10.0 → 6.10.2
Changelog: https://github.com/npm/cli/releases/tag/v6.10.2
Run npm install -g npm to update!
```

Catatan: jika terdapat error ketika instalasi pustaka dsb mungkin disebabkan karena cache terdahulu, karenanya untuk mengatasinya, kamu bisa jalankan perintah `npm cache clean --force` atau karena hak akses, jika perlu jalankan perintah di atas menggunakan mode administrator (`sudo`)

Instalasi Vue melalui NPM

Sebagaimana yang disebutkan diawal bahwa pustaka Vue bisa juga diinstalasi dengan mudah melalui tools NPM. Pada contoh ini, kita akan menginstalasinya pada folder `npm-vue`

`npm install vue`

```
1. bash
Hafids-MacBook-Pro:vue-projects hafidmukhlasin$ mkdir npm-vue
Hafids-MacBook-Pro:vue-projects hafidmukhlasin$ cd npm-vue/
Hafids-MacBook-Pro:npm-vue hafidmukhlasin$ npm install vue
npm WARN saveError ENOENT: no such file or directory, open '/Users/hafidmukhlasin/Dev/vue-projects/npm-vue/package.json'
npm notice created a lockfile as package-lock.json. You should commit this file
npm WARN enoent ENOENT: no such file or directory, open '/Users/hafidmukhlasin/ev/vue-projects/npm-vue/package.json'
npm WARN npm-vue No description
npm WARN npm-vue No repository field.
npm WARN npm-vue No README data
npm WARN npm-vue No license field.

+ vue@2.6.10
added 1 package from 1 contributor and audited 1 package in 2.245s
found 0 vulnerabilities
```

Lokasi pustaka vue hasil instalasi di atas dapat kita jumpai pada direktori `./node_modules/vue/dist/`. Tapi kok ada banyak sekali file disana? jangan khawatir, berikut ini akan penulis jabarkan tentang file-file tersebut.

File-file dalam folder `dist` (distribution) dapat dipetakan pada tabel berikut ini.

Version	UMD	CommonJS	ES Module
---------	-----	----------	-----------

Version	UMD	CommonJS	ES Module
Full	vue.js	vue.common.js	vue.esm.js
Runtime-only	vue.runtime.js	vue.runtime.common.js	vue.runtime.esm.js
Full (production)	vue.min.js		
Runtime-only (production)	vue.runtime.min.js		

Keterangan:

- **Compiler**: kode yang bertanggungjawab terhadap proses kompilasi template strings pada Vue menjadi fungsi render pada Javascript.
- **Runtime**: kode yang bertanggungjawab terhadap pembuatan objek Vue, rendering dan patching virtual DOM.
- **Full**: kode di dalamnya berisi compiler dan runtime
- **UMD**: Kode ini dapat langsung digunakan di browser melalui elemen `<script>`. Default file Vue dari unpkg CDN di <https://unpkg.com/vue> adalah runtime + compiler UMD (`vue.js`).
- **CommonJS**: CommonJS digunakan jika kita menggunakan bundler versi lama seperti `browserify` atau `webpack 1`. Default file untuk bundler ini (`pkg.main`) adalah runtime CommonJS (`vue.runtime.common.js`).
- **ES Module**: ES module (esm) digunakan jika kita menggunakan modern bundler seperti `webpack 2` atau `rollup`. Default file untuk bundler ini (`pkg.module`) adalah runtime ES Module (`vue.runtime.esm.js`).

Berdasarkan penjelasan di atas, kita bisa gunakan kode UMD yang full yaitu `vue.js` atau `vue.min.js` dengan cara meng-*include*-kan file kode tersebut melalui elemen `<script>`.

Buat file `index.html` pada direktori utama projekmu.

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Belajar Vue</title>
5      <script src=".//node_modules/vue/dist/vue.js"></script>
6  </head>
7  <body>
8      <div id="app">
9          {{ msg }}
10     </div>
11     <script>
12         var vm = new Vue({
13             el: '#app',
14             data: {
15                 msg: 'Vue on npm'
16             }
17         })
18     </script>
19 </body>
20 </html>

```

Kemudian, jalankan pada browser.



Catatan: pada pengembangan web modern, tentu kita tidak menggunakan cara ini untuk menggunakan pustaka Vue atau JS lainnya, melainkan menggunakan JS Bundler, apa itu?

Apa itu Bundler

Pada bagian sebelumnya, sedikit disinggung mengenai JS bundler, tools apa lagi itu?

Kalau kita membuat aplikasi menggunakan JS maka akan banyak file pustaka yang kita butuhkan atau masukkan pada file utama kita (`index.html`). Pada contoh yang lalu kita hanya menggunakan tiga file saja `vue.js`, `vue-router.js`, `vuex.js`, di mana `vue-router` dan `vuex` harus diletakkan setelah `vue` karena terkait dependensinya.

```

1 <script src="lib/vue.js"></script>
2 <script src="lib/vue-router.js"></script>
3 <script src="lib/vuex.js"></script>
```

Namun pada kasus nyata kita akan menggunakan banyak file pustaka berikut dependensinya yang akan cukup rumit jika kita mengaturnya secara manual.

JS bundler mengatasi hal ini dengan meletakkan file pustaka dan dependensinya dalam satu file JS, sehingga kita cukup memasukkan file tersebut saja dan semua pustaka bisa kita gunakan dalam kode kita. Hal ini mirip dengan fungsi composer autoload pada PHP.

Ada banyak pustaka JS bundler yang ada saat ini, diantaranya: webpack, browserify, rollup, parcel, dll. Konon webpack adalah yang paling populer sedangkan parcel adalah yang paling cepat.

Instalasi Vue menggunakan JS Bundler

Jika pada bagian sebelumnya kita menginstalasi Vue dengan memasukkan secara langsung pustakanya melalui elemen `<script>` maka pada bagian ini kita akan mencoba menggunakan JS bundler yang direkomendasikan untuk pengembangan projek aplikasi berbasis Javascript skala besar.

Sebagai bahan latihan, kita akan mencoba menggunakan dua pustaka JS bundler yaitu browserify dan webpack.

Browserify

Kita siapkan dulu folder untuk latihan ini.

```

1. bash
Hafids-MacBook-Pro:npm-vue hafidmukhlasin$ cd ..
Hafids-MacBook-Pro:vue-projects hafidmukhlasin$ mkdir browserify-vue
Hafids-MacBook-Pro:vue-projects hafidmukhlasin$ cd browserify-vue/
Hafids-MacBook-Pro:browserify-vue hafidmukhlasin$
```

Lalu kita masuk ke dalam folder tersebut, Kitainisialisasi projek ini dengan menjalankan perintah `npm init`, perintah ini akan menggenerate file `package.json` yang berisi profil dari projek kita

```
1. bash
package name: (browserify-vue)
version: (1.0.0)
description: Vue on Browserify
entry point: (index.js)
test command:
git repository:
keywords:
author: Hafid Muklasin
license: (ISC)
About to write to /Users/hafidmuklasin/Dev/vue-projects/browserify-vue/package.json:

{
  "name": "browserify-vue",
  "version": "1.0.0",
  "description": "Vue on Browserify",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "author": "Hafid Muklasin",
  "license": "ISC"
}

Is this OK? (yes) yes
Hafids-MacBook-Pro:browserify-vue hafidmuklasin$
```

Catatan: Kita cukup menekan enter pada setiap stepnya jika tidak ingin mengubah apapun, perhatikan juga pada properti main yang menunjukkan ke index.js, index.js ini adalah file yang nantinya akan digenerate oleh si JS bundler ini. Jadi kamu tidak perlu membuat file index.js ini secara manual.

Kemudian kita instalasi Vue melalui terminal dengan menggunakan perintah `npm install vue`. Jika kita lihat pada file package.json maka akan muncul dependencies dari pustaka yang baru kita instalasi yaitu Vue.

```
1 ...
2   "dependencies": {
3     "vue": "^2.6.10"
4   }
5 }
```

Setelah itu kita instalasi browserify, gunakan perintah `npm install browserify`

```

1. bash
Hafids-MacBook-Pro:browserify-vue hafidmukhlasin$ npm install vue
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN browserify-vue@1.0.0 No repository field.

+ vue@2.6.10
added 1 package from 1 contributor and audited 1 package in 2.73s
found 0 vulnerabilities

Hafids-MacBook-Pro:browserify-vue hafidmukhlasin$ npm install browserify
npm WARN browserify-vue@1.0.0 No repository field.

+ browserify@16.3.0
added 142 packages from 110 contributors and audited 920 packages in 18.653s
found 0 vulnerabilities

```

Maka dependecies pada package.json akan bertambah

```

1 "dependencies": {
2     "browserify": "^16.3.0",
3     "vue": "^2.6.10"
4 }
5 }
```

Buat file app.js

```

1 const Vue = require('vue/dist/vue.common')
2 new Vue({
3     el: "#app",
4     data: {
5         message: "Vue on Browserify",
6     }
7 })
```

Kode di atas akan me-require kode vue jenis commonJs (lihat pembahasan sebelumnya mengenai file jenis ini), kemudian inisialisasi objek Vue.

Pada package.json tambahkan perintah untuk mem-*bundle* kode app.js tersebut menjadi index.js

```

1 "scripts": {
2     "test": "echo \"Error: no test specified\" && exit 1",
3     "build": "browserify app.js -o index.js"
4 },
```

Catatan: perintah pada bagian build diatas bersifat opsional, jika kita definisikan maka akan mempersingkat perintah yang akan kita jalankan untuk mem-build kode kita.

Buat file index.html, *include*-kan file index.js menggunakan elemen <script>.

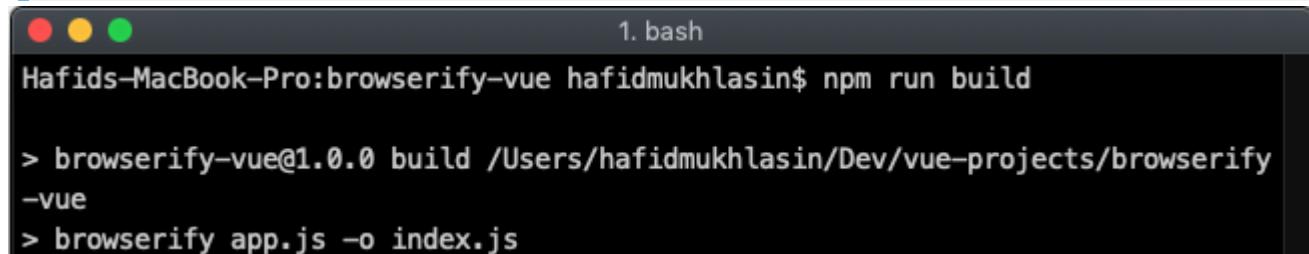
```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Belajar Vue</title>
5  </head>
6  <body>
7      <div id="app">
8          {{ message }}
9      </div>
10     <script src="index.js"></script>
11 </body>
12 </html>

```

Pada terminal, jalankan perintah `npm run build`

Catatan: perintah di atas, setara dengan perintah `browserify app.js -o index.js`, silakan cek pada `package.json` bagian `build`.



```

1. bash
Hafids-MacBook-Pro:browserify-vue hafidmukhlasin$ npm run build

> browserify-vue@1.0.0 build /Users/hafidmukhlasin/Dev/vue-projects/browserify-vue
> browserify app.js -o index.js

```

Silakan cek apakah file `index.js` sudah terbentuk, dan lihat kode didalamnya.

Kemudian, jalankan file `index.html` tersebut via browser. Maka pada browser akan muncul teks "Vue on Browserify"

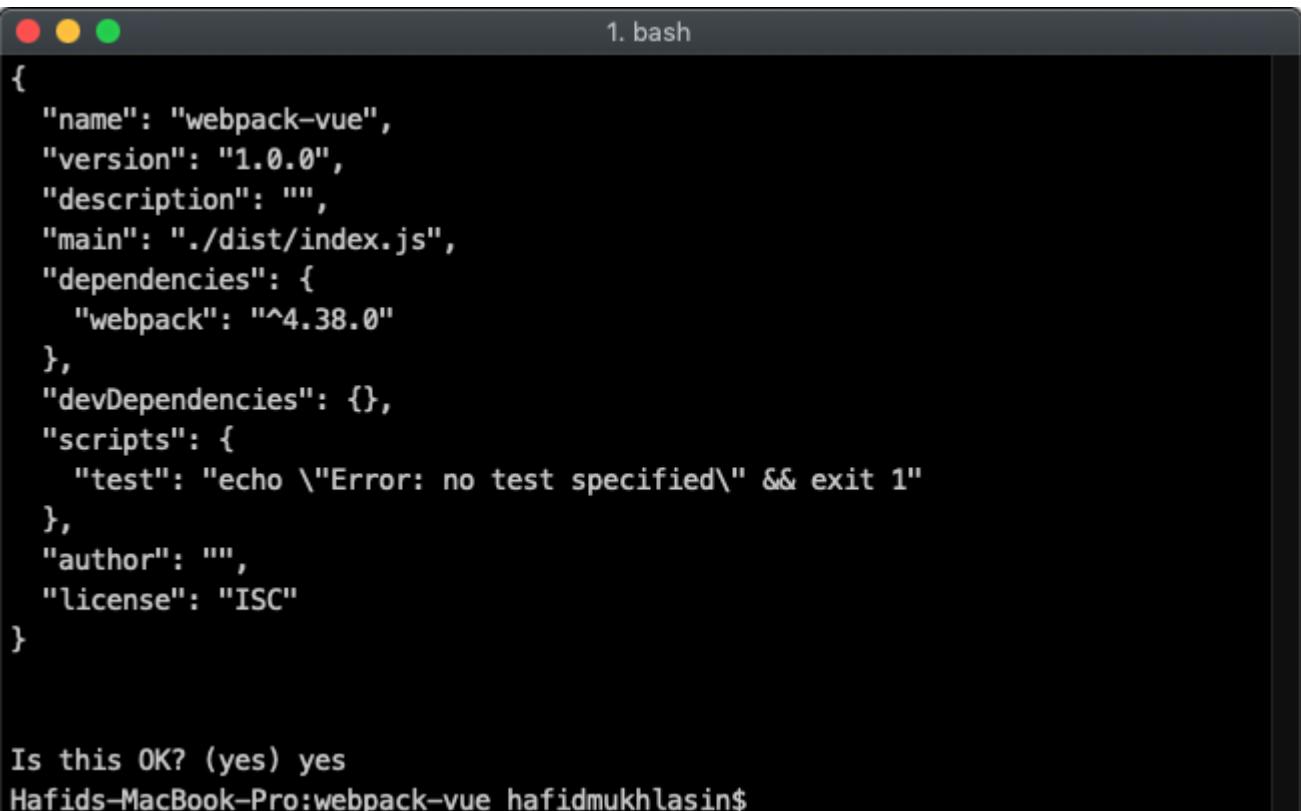
Demikianlah cara yang perlu kita lakukan jika menggunakan browserify sebagai bundler.

Webpack

Setelah mencoba JS blunder browserify, sekarang kita akan mencoba JS Blunder lain yaitu webpack. Webpack merupakan JS bundler yang cukup populer dan direkomendasikan untuk digunakan dalam pengembangan aplikasi berbasis Vue skala besar.

Langkah pertama untuk memulai latihan ini adalah membuat folder baru (misal: `webpack-vue`)

Kemudian jalankan perintah `npm init` untuk menginisialisasi projek kita via `package.json`, sebagaimana langkah yang kita lakukan ketika kita menggunakan bundler



```
{
  "name": "webpack-vue",
  "version": "1.0.0",
  "description": "",
  "main": "./dist/index.js",
  "dependencies": {
    "webpack": "^4.38.0"
  },
  "devDependencies": {},
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

Is this OK? (yes) yes
Hafids-MacBook-Pro:webpack-vue hafidmukhlasin\$

Pada kode di atas, file JS utama diletakkan pada lokasi `./dist/index.js` (tidak di root directory melainkan di folder dist supaya lebih terstruktur), di mana file ini nantinya akan di-generate oleh webpack.

Buat file index.html pada root directory yang meng-*include* file `index.js`.

```

1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Belajar Vue Webpack</title>
5    </head>
6    <body>
7      <div id="app">
8        {{ message }}
9      </div>
10     <script src="dist/index.js"></script>
11   </body>
12 </html>
```

Selanjutnya, buat file `app.js` pada directory `src`. Di sini kita mulai belajar mengelompokkan file kode aplikasi, di mana folder `src` akan berisi file-file kodingan kita, sedangkan folder `dist` (distribution) akan berisi file hasil generate webpack.

```

1 import Vue from 'vue'
2
3 new Vue({
4   el: '#app',
5   data: {
6     message: "Vue on Webpack"
7   }
8 })
```

Nantinya pada file `app.js` ini kita bisa memasukkan semua module atau component pada aplikasi kita.

Kemudian, buat file bernama `webpack.config.dev.js` pada root direktori yang berisi kode berikut:

```

1 const path = require('path')
2
3 module.exports = {
4   mode: 'development',
5   entry: './src/app.js',
6   output: {
7     path: path.resolve(__dirname, 'dist'),
8     publicPath: '/dist/',
9     filename: 'index.js'
10   },
11   resolve: {
12     alias: {
13       'vue$': 'vue/dist/vue.esm.js',
14     }
15   },
16   module: {
17     rules: [
18       {
19         test: /\.js$/,
20         exclude: /node_modules/,
21         use: {
22           loader: "babel-loader"
23         }
24       }
25     ]
26   }
27 }

```

Konfigurasi di atas untuk menginstruksikan agar webpack mem-*bundle* file `src/app.js` dan hasilnya disimpan pada file `dist/build.js`. Mode yang kita gunakan mode: 'development'.

Pada bagian `resolve`, kita perlu mengaliaskan `vue` sebagai `vue/dist/vue.esm.js` sehingga ketika kita memanggil Vue maka pustaka vue yang digunakan adalah yang versi ES module (lihat pembahasan sebelumnya).

Adapun pada bagian `module`, kita perlu menambahkan `rules` atau aturan di mana, semua file berekstensi `.js` kecuali yang ada di dalam folder `node_modules` akan dimuat dan dikonversi ke dalam versi JS yang didukung oleh browser dengan menggunakan pustaka `babel`.

Apa itu `babel`? pustaka loader untuk file JS. <https://babeljs.io>.

Catatan: file `webpack.config.dev.js` merupakan config webpack mode development, adpun untuk mode production, kamu bisa buat file config baru misalnya `webpack.config.prod.js`. Penamaan file config ini sebenarnya bebas saja.

Kemudian kita perlu membuat file konfigurasi dari pustaka `babel` bernama `.babelrc` pada root direktori.

```

1 {
2   "presets": ["env"]
3 }

```

Lakukan instalasi pustaka utama yaitu Vue dengan menggunakan perintah `npm install --save vue`. Maka dependecies pada package.json akan bertambah

```

1 "dependencies": {
2   "vue": "^2.6.10"
3 },

```

Selanjutnya, kita instalasi pustaka pendukung untuk development yaitu webpack, dan babel serta core-js (polyfill untuk standard library JS - opsional).

```

1 npm install --save-dev webpack-cli
2 npm install --save-dev babel-core babel-loader babel-preset-env
3 @babel/core
4 npm install core-js

```

Hasil dari instalasi ini akan mengupdate secara otomatis file package.json

```

1 "devDependencies": {
2   "@babel/core": "^7.5.5",
3   "babel-core": "^6.26.3",
4   "babel-loader": "^8.0.6",
5   "babel-preset-env": "^1.7.0",
6   "webpack": "^4.38.0",
7   "webpack-cli": "^3.3.6"
8 },

```

Catatan `--save` digunakan untuk menyimpan package yang dibutuhkan untuk menjalankan aplikasi. `--save-dev` digunakan untuk menyimpan package yang digunakan untuk tujuan development.

Masih pada file package.json pada key `scripts` tambahkan perintah untuk menjalankan webpack dengan konfigurasi sebagaimana yang tertuang pada file `webpack.config.dev.js`

```

1 "scripts": {
2   "test": "echo \"Error: no test specified\" && exit 1",
3   "dev": "webpack --config webpack.config.dev.js",
4 }

```

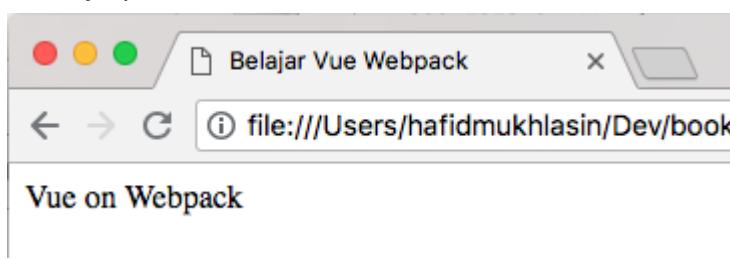
Jalankan perintah `npm run dev` pada terminal untuk megeksekusi webpack dengan konfigurasi sebagaimana file `webpack.config.dev.js`.

```
1. bash
Hafids-MacBook-Pro:webpack-vue hafidmukhlasin$ npm run dev

> webpack-vue@1.0.0 dev /Users/hafidmukhlasin/Dev/vue-projects/webpack-vue
> webpack --config webpack.config.dev.js

Hash: 916658ee4bb13be6e74a
Version: webpack 4.38.0
Time: 1156ms
Built at: 07/28/2019 7:27:34 AM
    Asset      Size  Chunks             Chunk Names
index.js   348 KiB  main  [emitted]  main
Entrypoint main = index.js
[./node_modules/webpack/buildin/global.js] (webpack)/buildin/global.js 472 bytes {main} [built]
[./src/app.js] 264 bytes {main} [built]
+ 4 hidden modules
Hafids-MacBook-Pro:webpack-vue hafidmukhlasin$
```

Hasilnya pada browser



Hore! 😊

Single File Component

Pada bab component, sudah kita bahas tentang konsep single file component (SFC) yaitu memecah suatu component menjadi file tersendiri dengan menggunakan fitur ES6 import dan export. Meski tujuan telah tercapai namun cara tersebut ada batasannya salah satunya adalah file component hanya bisa berisi kode JS saja sehingga kurang fleksibel. Di samping itu kita perlu mendefinisikan `type` dari elemen `<script>` sebagai module.

Untuk mengatasi hal ini, Vue memperkenalkan cara yang lebih fleksible dengan menggunakan pustaka `vue-loader` yang berfungsi memuat file component yang defaultnya berakhiran `.vue`. Pustaka ini merupakan loader untuk webpack layaknya babel.

Pertama kita perlu instalasi pustaka ini

```
npm install --save-dev vue-loader
```

Karena SFC pada vue tidak hanya berisi kode JS, namun juga template dan css maka kita perlu menginstalasi pustaka yang bisa menangani template dan css yaitu `vue-template-compiler`, `css-loader`, dan `css-style-loader`.

Gunakan perintah berikut pada terminal untuk menginstall ketiga pustaka tersebut.

```
npm install --save-dev vue-template-compiler css-loader vue-style-loader
```

Cek package.json

```
1 "devDependencies": {
2     "@babel/core": "^7.5.5",
3     "babel-core": "^6.26.3",
```

```

4 "babel-loader": "^8.0.6",
5 "babel-preset-env": "^1.7.0",
6 "css-loader": "^3.1.0",
7 "vue-loader": "^15.7.1",
8 "vue-style-loader": "^4.1.2",
9 "vue-template-compiler": "^2.6.10",
10 "webpack": "^4.38.0",
11 "webpack-cli": "^3.3.6"
12 },

```

Kemudian update file `webpack.config.dev.js` dengan menambahkan konfigurasi vue-loader.

```

1 const path = require('path')
2 const VueLoaderPlugin = require('vue-loader/lib/plugin')

3
4 module.exports = {
5   mode: 'development',
6   entry: './src/app.js',
7   output: {
8     path: path.resolve(__dirname, 'dist'),
9     publicPath: '/dist/',
10    filename: 'index.js'
11  },
12  resolve: {
13    alias: {
14      'vue$': 'vue/dist/vue.esm.js',
15    }
16  },
17  module: {
18    rules: [
19      {
20        test: /\.js$/,
21        exclude: /node_modules/,
22        use: {
23          loader: "babel-loader"
24        },
25      },
26      {
27        test: /\.vue$/,
28        loader: 'vue-loader'
29      },
30      {
31        test: /\.css$/,
32        use: [
33          'vue-style-loader',
34          'css-loader'
35        ]
36      }
37    ],
38  },
39  plugins: [
40    new VueLoaderPlugin()
41

```

```
42 |     ]
  }
```

Kemudian kita coba buat sebuah component pada file terpisah (SFC). Buat file `Hello.vue` pada direktori `src`.

```
1 <template>
2   <div class="hello">{{ text }}</div>
3 </template>
4
5 <script>
6 export default {
7   data () {
8     return {
9       text: 'Hello dari vue-loader!'
10    }
11  }
12 }
13 </script>
14
15 <style>
16 .hello {
17   color: blue;
18 }
19 </style>
```

Kode `Hello.vue` di atas hanya memberikan contoh bahwa kita dapat menggunakan tiga jenis kode sekaligus yaitu template Vue (HTML), kode JS dan kode CSS.

Ubah file `src/app.js` dengan mendaftarkan component Hello.

```
1 import Vue from 'vue'
2 import Hello from './Hello.vue'
3
4 new Vue({
5   el: '#app',
6   data: {
7     message: "Vue on Webpack"
8   },
9   components: { Hello }
10 })
```

Kemudian tambahkan elemen component Hello pada `index.html`

```
1 <div id="app">
2   {{ message }}
3   <hello></hello>
4 </div>
```

Jalankan perintah `npm run dev` pada terminal untuk mengeksekusi webpack. Kemudian jalankan file `index.html` pada browser. Lihat hasilnya..

The screenshot shows a Mac OS X style browser window. The title bar says 'Belajar Vue Webpack'. The address bar shows the path '/Users/hafidmukhlasin/Dev'. The main content area displays the text 'Vue on Webpack' and 'Hello dari vue-loader!'

Yes, berhasil!

Web Server Development

Untuk menguji coba aplikasi biasanya kita langsung jalankan file index.html pada browser melalui protocol `file://` atau pada contoh terdahulu, kita menggunakan web server (nginx atau apache) sehingga bisa diakses melalui protocol `http`.

Nah, webpack dalam hal ini juga menyediakan web server untuk tujuan development.

```
npm install --save-dev webpack-dev-server
```

The terminal window shows the command `npm install --save-dev webpack-dev-server` being run. The output indicates that the package was installed successfully, with 195 packages added from 131 contributors and audited 9510 packages in 25.704s. A warning message states 'WARN webpack@1.0.0 No repository field.'

```
[Hafids-MacBook-Pro:webpack-vue hafidmukhlasin$ npm install --save-dev webpack-de]
v-server
npm WARN webpack@1.0.0 No repository field.

+ webpack-dev-server@3.1.4
added 195 packages from 131 contributors and audited 9510 packages in 25.704s
found 0 vulnerabilities
```

Kemudian pada package.json key scripts, ubah `webpack` menjadi `webpack-dev-server`.

```
1 "scripts": {
2     "test": "echo \"Error: no test specified\" && exit 1",
3     "dev": "webpack-dev-server --config webpack.config.dev.js"
4 },
```

Kemudian jalankan lagi perintah `npm run dev`, perintah ini disamping mem-bundle file juga menjalankan web server development.

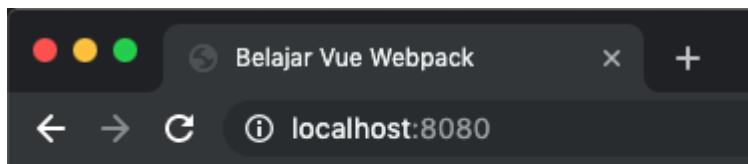
The terminal window shows the command `npm run dev` being run. The output shows the webpack-dev-server starting up, indicating the project is running at `http://localhost:8080`. It also lists the asset sizes and chunk names.

```
1. node
Hafids-MacBook-Pro:webpack-vue hafidmukhlasin$ npm run dev

> webpack-vue@1.0.0 dev /Users/hafidmukhlasin/Dev/vue-projects/webpack-vue
> webpack-dev-server --config webpack.config.dev.js

i [wds]: Project is running at http://localhost:8080
i [wds]: webpack output is served from /dist/
i [wds]: Content not from webpack is served from /Users/hafidmukhlasin/Dev/vue-projects/webpack-vue
i [wdm]: Hash: 03d1fff8a3576019dcc0
Version: webpack 4.38.0
Time: 2106ms
Built at: 07/28/2019 9:07:30 AM
      Asset      Size  Chunks             Chunk Names
index.js  734 KiB    main  [emitted]  main
Entrypoint main = index.js
```

Selanjutnya kita bisa mengakses melalui URL : <http://localhost:8080>



Vue on Webpack

Hello dari vue-loader!

Awesome!

Untuk mengubah port default 8080 kita bisa tambahkan pengaturanya pada file webpack.config.dev.js key devServer sebagai berikut.

```
1 module.exports = {
2   entry: './src/app.js',
3   output: {
4     ...
5   },
6   module: {
7     ...
8   },
9   ...
10  devServer: {
11    port: 9000
12  }
13 }
```

Alternatif lain bisa kita tambahkan parameter --port pada scripts package.json.

```
webpack-dev-server --config webpack.conf.dev.js --port 9000
```

Maka selanjutnya kita bisa mengakses aplikasi dengan menggunakan URL <http://localhost:9000>

Keren kan?

Hot Reload

Umumnya, setiap kali kita mengedit kode pada component .vue maka untuk melihat perubahan yang yang kita lakukan, maka pada browser kita harus merefresh atau reload (F5) dahulu halaman browser kita. Nah, lagi-lagi Vue memanjakan kita para developer sehingga ketika kita melakukan perubahan kode maka saat itu juga hasilnya bisa kita lihat pada browser tanpa perlu merefreshnya secara manual.

Pustaka untuk me-refresh otomatis tersebut adalah [vue-hot-reload-api](https://github.com/vuejs/vue-hot-reload-api) (<https://github.com/vuejs/vue-hot-reload-api>). Untuk menginstalasinya jalankan perintah berikut.

```
npm install --save-dev vue-hot-reload-api
```

```
webpack-vue — bash — 80x40
[Hafids-MacBook-Pro:webpack-vue hafidmukhlasin$ npm install --save-dev html-webpack-plugin
npm WARN webpack-vue@1.0.0 No repository field.

+ html-webpack-plugin@3.2.0
added 37 packages from 56 contributors and audited 9585 packages in 15.453s
found 0 vulnerabilities
```

Kemudian pada webpack.config.dev.js, kita update konfigurasinya menjadi sebagai berikut.

```

1 const path = require('path')
2 const VueLoaderPlugin = require('vue-loader/lib/plugin')
3 const webpack = require('webpack');
4
5 module.exports = {
6   mode: 'development',
7   entry: './src/app.js',
8   output: {
9     path: path.resolve(__dirname, 'dist'),
10    publicPath: '/dist/',
11    filename: 'build.js'
12  },
13  resolve: {
14    alias: {
15      'vue$': 'vue/dist/vue.esm.js',
16    }
17  },
18  module: {
19    rules: [
20      {
21        test: /\.js$/,
22        exclude: /node_modules/,
23        use: {
24          loader: "babel-loader"
25        },
26      },
27      {
28        test: /\.vue$/,
29        loader: 'vue-loader'
30      },
31      {
32        test: /\.css$/,
33        use: [
34          'vue-style-loader',
35          'css-loader'
36        ]
37      }
38    ]
39  },
40  plugins: [
41    new VueLoaderPlugin(),
42    new webpack.HotModuleReplacementPlugin(),
43  ],
44  devServer: {
45    hot: true
46  }
47}

```

Tekan **CTRL + C** untuk menghentikan perintah sebelumnya lalu jalankan perintah `npm run dev` dan lihat hasilnya pada browser.

The screenshot shows a browser window titled "Belajar Vue Webpack" at "localhost:8080". The page content says "Vue on Webpack" and "Hello dari vue-loader!, baru diedit udah langsung muncul". Below the browser is a developer tools console tab labeled "Console". The log output is as follows:

```

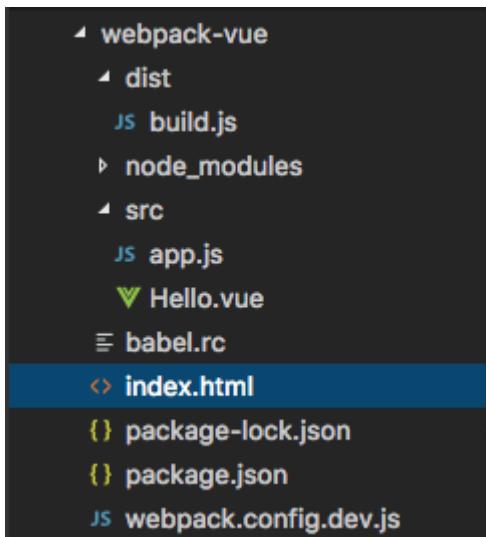
[HMR] Waiting for update signal from WDS...
You are running Vue in development mode.
Make sure to turn on production mode when deploying for production.
See more tips at https://vuejs.org/guide/deployment.html
[WDS] Hot Module Replacement enabled.
[WDS] Live Reloading enabled.
[2] [WDS] App updated. Recompiling...
[WDS] App hot update...
[HMR] Checking for updates on the server...
[HMR] Updated modules:
▶ [HMR] - ./src/Hello.vue?vue&type=script&lang=js&
[HMR] - ./src/Hello.vue?vue&type=script&lang=js&
[HMR] - ./src/Hello.vue
[HMR] App is up to date.

```

Ketika kita melakukan perubahan pada component Hello.vue maka otomatis akan dilakukan kompilasi ulang dan perubahannya seketika bisa dilihat pada browser. Cek hal itu pada browser console, dimana hot reload akan mengecek setiap saat perubahan disisi server kemudian melakukan compile ulang.

Gokil kan?

Dari hasil koding kita di atas menghasilkan struktur direktori aplikasi sebagai berikut.



Catatan: selengkapnya mengenai hot reload, kamu bisa kunjungi halaman berikut <https://vue-loader.vuejs.org/guide/hot-reload.html>

Fitur-fitur webpack tidak hanya berhenti sampai di sini saja, fitur-fitur lain diantaranya: integrasi dengan code testing, lint (formatter), progressive web app (PWA), dsb.

Vue Command Line Interface (CLI)

Sadar akan rumitnya pengaturan JS bundler dan semua tools terkait pengembangan, maka lagi-lagi Vue membuat sebuah tools yang akan memudahkan kita melakukan hal itu semua. Tools itu bernama Vue CLI

yaitu Command Line Interface sederhana untuk scaffolding projek aplikasi berbasis Vue. Hal ini terinspirasi dari [create-react-app](#).

catatan: dengan menggunakan Vue CLI ini maka semua tools yang kita bahas pada bagian sebelumnya akan tetap digunakan namun integrasinya mudah sehingga nyaris tanpa effort. Vue CLI dibangun di atas webpack dan webpack-dev-server

Saat ini Vue CLI sudah mencapai versi 3 dengan berbagai perubahan dan fitur baru salah satunya adalah menyediakan user interface berbasis web untuk menyiapkan projek kita.

Vue CLI versi ini menggunakan plugins arsitektur yang bisa ditambahkan dan dikurangi, plugin tersebut telah diprekonfigurasi menurut kaidah best practice sehingga user bisa menambahkan plugins tanpa perlu banyak melakukan konfigurasi manual lagi. Beberapa plugins yang disediakan antara lain: Typescript, PWA, Vuex, Vue Router, ESLint, unit testing dll.

Cara menginstalasi pustaka ini dengan menggunakan perintah berikut.

```
npm install -g @vue/cli
```

atau

```
yarn global add @vue/cli
```

catatan: gunakan sudo atau mode administrator untuk menjalankan perintah di atas. Kita instal secara global supaya perintah `vue` (cli) bisa digunakan dimanapun.

Untuk memastikan bahwa Vue CLI berhasil terinstalasi dengan baik maka jalankan perintah berikut yang seharusnya akan menampilkan versi dari Vue CLI.

```
vue --version
```

Pada saat tulisan disusun, versi vue-cli adalah 3.9.3

Create New Project

Untuk membuat projek baru dengan menggunakan vue-cli ini maka cukup jalankan perintah dengan format berikut.

```
vue create <project-name>
```

Contoh:

```
vue create myproject
```

Perintah di atas akan menampilkan panduan interaktif (wizard) untuk menginstalasi dan mempersiapkan tools-tools yang diperlukan untuk pengembangan aplikasi berbasis Vue. Hasilnya akan diletakkan pada folder `myproject` sebagaimana parameter yang dituliskan pada perintah `vue create`.

Catatan: Gunakan perintah sudo atau mode administrator untuk menjalankan perintah di atas.

Minimal ada dua pilihan preset (pre konfigurasi), apakah default atau pilih secara manual fiturnya (pilihan lain muncul karena kita bisa melakukan kustomisasi terhadap instalasi). Jika kita pilih default preset maka sebuah projek baru akan dibuat dan pustaka yang umum digunakan untuk pengembangan aplikasi berbasis Vue akan diinstalasi.

Pertama, kita coba dulu yang default preset.

```
1. node
Hafids-MacBook-Pro:vue-projects hafidmukhlasin$ vue create myproject

Vue CLI v3.9.3
? Please pick a preset:
  vueshop-template (vue-router, vuex, stylus, babel, pwa, eslint, unit-mocha,
e2e-cypress)
  vueshop-template-2 (vue-router, vuex, stylus, babel, pwa, eslint, unit-mocha
)
❯ default (babel, eslint)
  Manually select features
```

catatan: gunakan tombol panah keatas/kebawah pada keyboard untuk perpindah pilihan dan tekan enter untuk menggunakan pilihan itu.

tldr;

```
1. bash
↳ Running completion hooks...

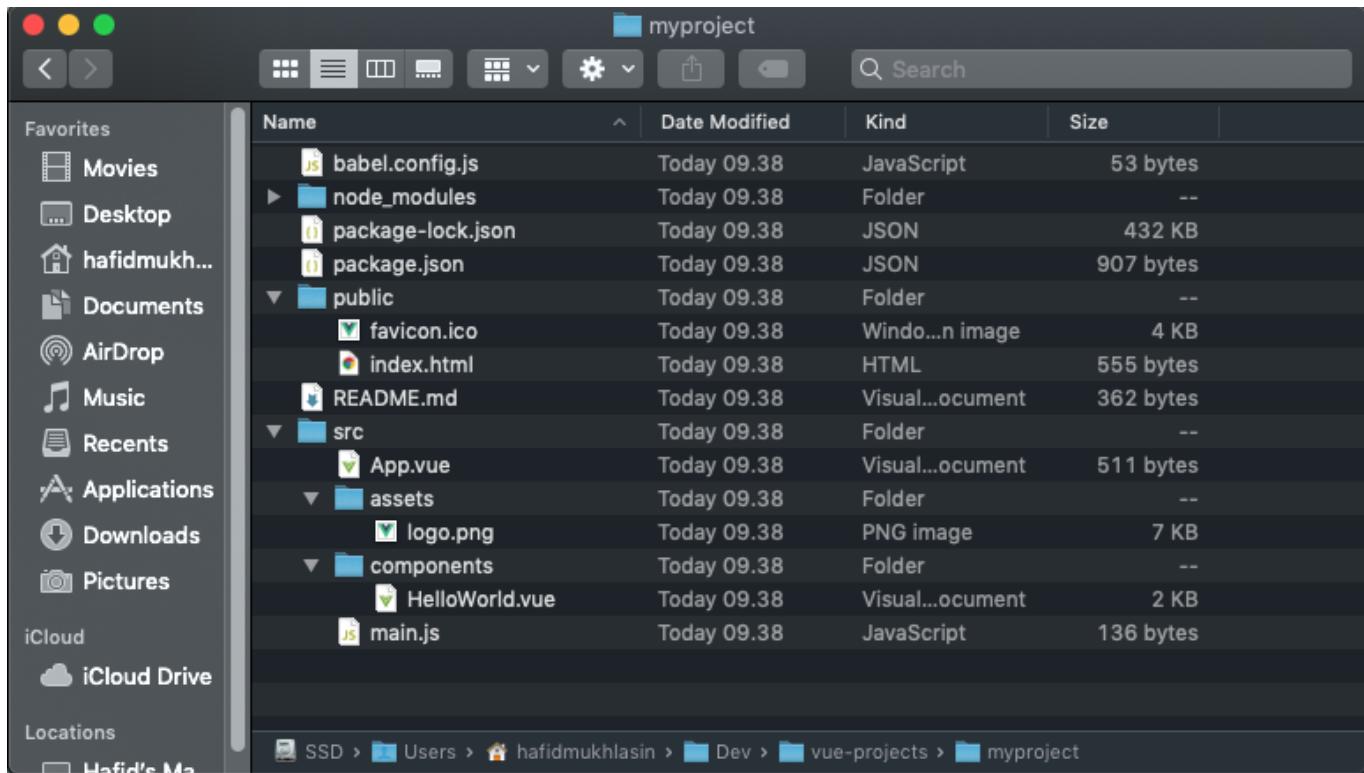
📄 Generating README.md...

🎉 Successfully created project myproject.
👉 Get started with the following commands:

$ cd myproject
$ npm run serve

Hafids-MacBook-Pro:vue-projects hafidmukhlasin$
```

Sebelum kita melangkah lebih jauh, mari kita perhatikan struktur direktori projek yang dibuat dengan vue-cli default preset.



Terdapat 3 folder utama yaitu:

- `node_modules` yaitu folder untuk menampung pustaka yang terinstalasi untuk projek ini, terdapat lebih dari 80 pustaka di dalamnya diantaranya: webpack, babel, browserify, eslint, dsb. Beberapa pustaka telah kita bahas sebelumnya, beberapa yang lain sebagiannya akan kita bahas berikutnya.
- `src` yaitu folder untuk menampung source code kita, seperti component, assets (image, icon), template, store, dsb.
- `public` yaitu folder untuk menampung file `index.html` atau mount point dari aplikasi kita nanti, ada juga file `favicon` di sini.

Di samping itu terdapat 3 file yaitu:

- `package.json` yaitu file yang berisi profil dari projek, konfigurasi serta daftar pustaka yang digunakan.
- `package-lock.json` yaitu file yang berisi daftar pustaka yang telah terinstalasi.
- `babel.config.js` yaitu file yang berisi konfigurasi dari pustaka babel.

Pada projek ini kita sudah dibuatkan contoh template aplikasinya, sudah ada contoh component-nya. Tugas kita hanya meneruskan tanpa perlu berpusing-pusing dalam mengkonfigurasinya.

Mari kita buka file `index.html` pada folder `public`.

```

1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width,initial-scale=1.0">
7      <link rel="icon" href="<%= BASE_URL %>favicon.ico">
8      <title>myproject</title>
9    </head>
10   <body>
11     <noscript>
12       <strong>We're sorry but myproject doesn't work properly without
13       JavaScript enabled. Please enable it to continue.</strong>
14     </noscript>

```

```

15 | <div id="app"></div>
16 |   <!-- built files will be auto injected -->
17 | </body>
</html>

```

Pada kode di atas bagian link rel icon, menggunakan kode `<%= BASE_URL %>` untuk menampilkan URL dasar. Di sana juga terdapat kontainer utama aplikasi `<div id="app"></div>`, yang menarik adalah tidak ada file Javascript yang di-include-kan di sana. Yap, file JS tersebut nantinya akan diinjeksi secara otomatis ke dalam file index.html ini ketika projek di jalankan.

File berikutnya adalah main.js pada folder `src`.

```

1 import Vue from 'vue'
2 import App from './App.vue'
3
4 Vue.config.productionTip = false
5
6 new Vue({
7   render: h => h(App)
8 }).$mount('#app')

```

Tentu kamu tidak asing dengan kode di atas? itu hanya sedikit modifikasi dari bentuk ini.

```

1 new Vue({
2   el: '#app',
3   components: { App },
4   template: '<App/>'
5 })

```

Terkait hal ini, juga telah dibahas pada bab awal. Silakan dibaca lagi ya 😊.

File berikutnya masih dalam folder `src` adalah `App.vue` yang merupakan component utama atau wrapper dari semua component yang nantinya digunakan pada projek aplikasi kita. Pada contoh ini component `HelloWorld` (`src/components/HelloWorld.vue`) diimport pada component `App.vue`, demikian juga nanti pada component lainnya.

Overall, tidak ada yang magis sebab semua masih dalam lingkup yang telah kita bahas sebelumnya.

Sekarang waktunya kita jalankan projek kita ini. Pada terminal, masuk ke folder `myproject` kemudian jalankan perintah `npm run serve`

```
1. node
Hafids-MacBook-Pro:myproject hafidmukhlasin$ npm run serve

> myproject@0.1.0 serve /Users/hafidmukhlasin/Dev/vue-projects/myproject
> vue-cli-service serve

[INFO] Starting development server...
98% after emitting CopyPlugin

[DONE] Compiled successfully in 3723ms 9:46:14 AM

App running at:
- Local: http://localhost:8080/
- Network: http://192.168.43.127:8080/

Note that the development build is not optimized.
To create a production build, run npm run build.
```

Akses pada browser <http://localhost:8080>

Welcome to Your Vue.js App

For a guide and recipes on how to configure / customize this project,
check out the [vue-cli documentation](#).

Installed CLI Plugins

[babel](#) [eslint](#)

Ini adalah aplikasi mode development yang belum dioptimalkan, untuk mendapatkan versi production maka pada terminal jalankan perintah `npm run build` (dengan sebelumnya tekan `CTRL+C` untuk keluar dari perintah sebelumnya)

```
1. bash
Hafids-MacBook-Pro:myproject hafidmukhlasin$ sudo npm run build
Password:

> myproject@0.1.0 build /Users/hafidmukhlasin/Dev/vue-projects/myproject
> vue-cli-service build

:: Building for production...

[DONE] Compiled successfully in 7073ms 9:49:50 AM

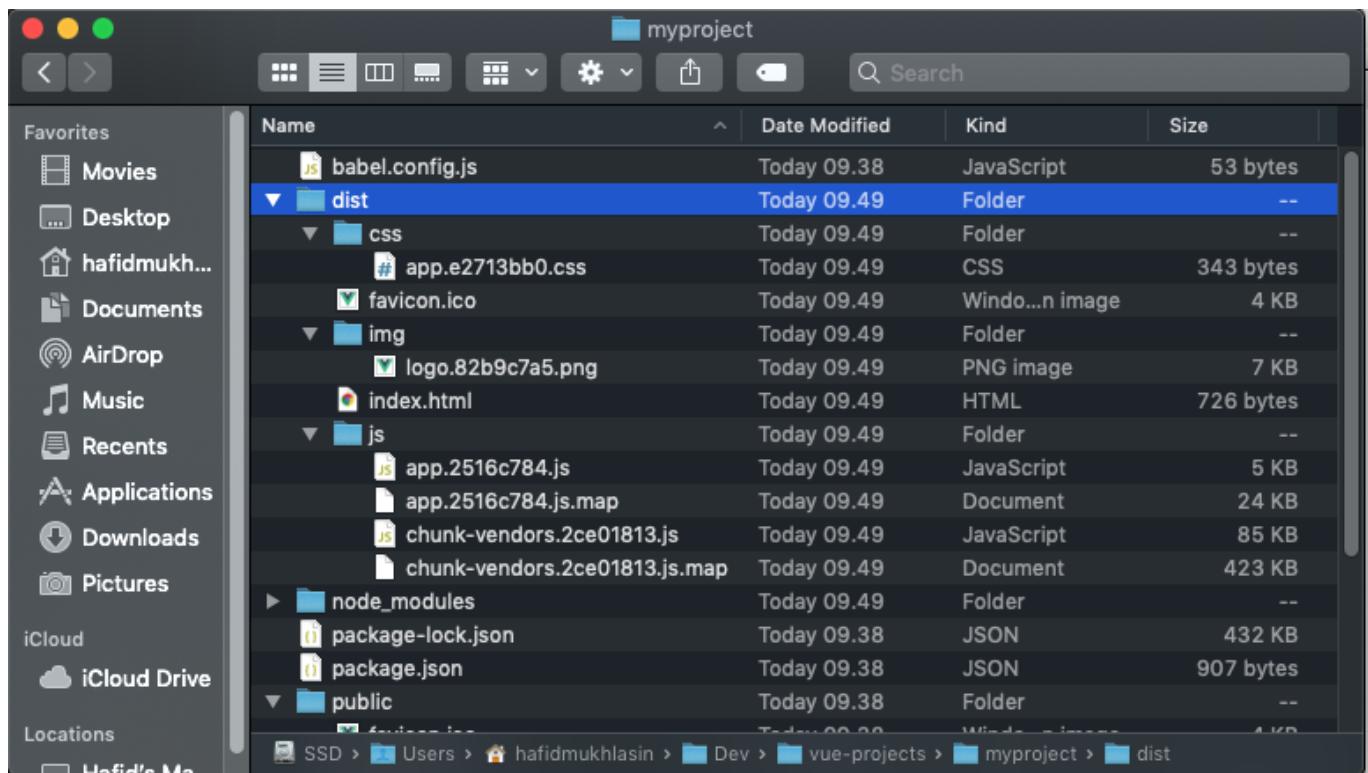
File Size Gzipped
dist/js/chunk-vendors.2ce01813.js 82.82 KiB 29.95 KiB
dist/js/app.2516c784.js 4.60 KiB 1.65 KiB
dist/css/app.e2713bb0.css 0.33 KiB 0.23 KiB

Images and other types of assets omitted.

[DONE] Build complete. The dist directory is ready to be deployed.
[INFO] Check out deployment instructions at https://cli.vuejs.org/guide/deployment.html

Hafids-MacBook-Pro:myproject hafidmukhlasin$
```

Maka akan digenerate file aplikasi yang telah dioptimalkan untuk production pada folder `dist`



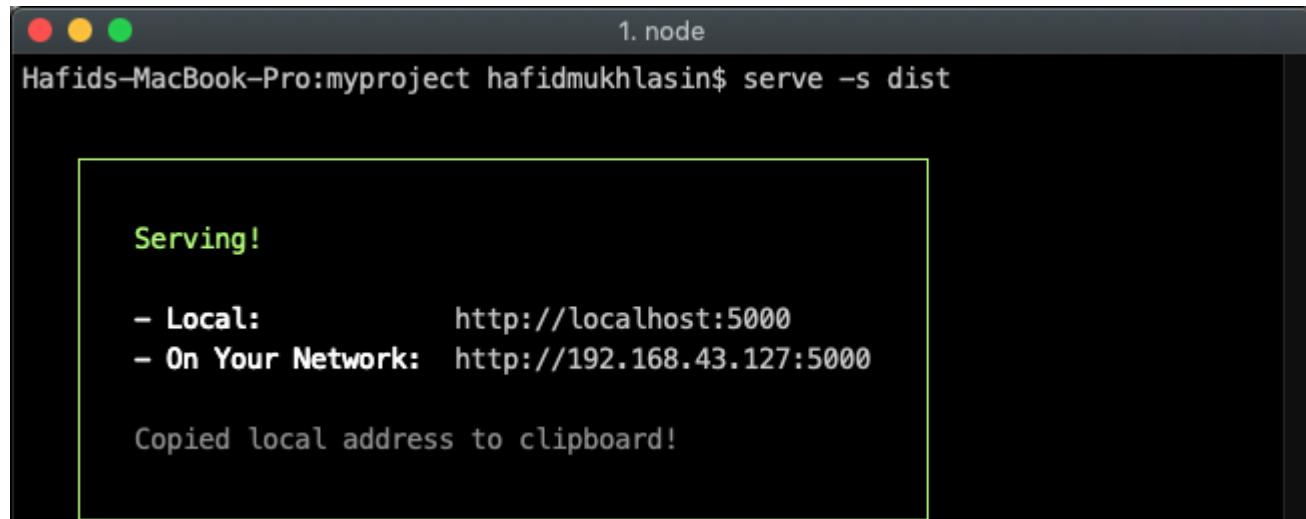
Hanya file-file dalam folder inilah yang nantinya kita deploy (unggah) ke server production. Lho.. pustaka Vue dan sebagainya apa gak perlu dideploy juga? ga perlu, sebab pustaka-pustaka tersebut telah di-*bundle* oleh webpack menjadi file `js/chunk-vendors.[xyz].js` sedangkan kode aplikasi kita pada folder `src` juga telah di-*bundle* menjadi file `js/app.[xyz].js`.

Hasil build production ini bisa kita preview file index.html-nya di local melalui browser dengan protocol file:// atau bisa juga menggunakan web server static milik NodeJS tapi perlu kita instalasi dulu 😊.

npm install -g serve

kemudian untuk menjalankannya gunakan perintah

serve -s dist



```
1. node
Hafids-MacBook-Pro:myproject hafidmukhlasin$ serve -s dist

Serving!
- Local: http://localhost:5000
- On Your Network: http://192.168.43.127:5000

Copied local address to clipboard!
```

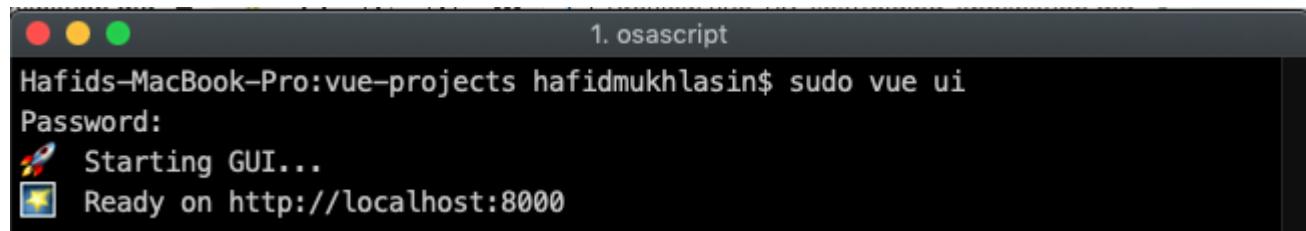
Silakan buka browser dan akses alamat <http://localhost:5000>

Yeay, viola.

Create New Project on Web Base

Disamping menyediakan user interface (UI) instalasi berbasis terminal, Vue CLI juga menyediakan graphical UI berupa tampilan berbasis web yang memudahkan bagi user pemula untuk memulai men-setup projek aplikasinya. Namun, jika kamu sudah membaca buku ini sampai dengan bab ini maka tentu GUI ini bukan untuk kamu. "Kamu udah advanced men..." 😊"

Tapi tak ada salahnya kita mencobanya, sekalian mempersiapkan scaffolding untuk proyek studi kasus kita yaitu toko buku berbasis mobile web. Caranya pada terminal jalankan perintah vue ui maka Vue CLI akan menjalankan server web yang defaultnya beralamat di <http://localhost:8000>



```
1. osascript
Hafids-MacBook-Pro:vue-projects hafidmukhlasin$ sudo vue ui
Password:
Starting GUI...
Ready on http://localhost:8000
```

Vue CLI juga sekaligus akan menjalankan web browser dan menunjuk ke alamat tersebut atau tepatnya <http://localhost:8000/dashboard>

The screenshot shows the 'Project dashboard' page of a Vue application. On the left, there's a sidebar with icons for file management, a puzzle piece, a document, settings, and a list. The main area has a large 'Welcome tips' section featuring a green circular icon with a white 'V' logo. Below it is a message: 'Welcome to your new project!'. A tooltip appears over this message, stating: 'You are looking at the project dashboard where you can put widgets. Use the 'Customize' button to add. Everything is automatically saved.' To the right, there's a 'Kill port' section with a 'Ready to kill' button and a field to enter a network port, along with a 'Kill' button. At the bottom, the browser status bar shows the path '/Users/hafidmukhlasin/Dev/vue-projects/vueshop' and the status 'Ready on http://localhost:8000'.

Untuk memulai membuat project akses alamat <http://localhost:8000/project/select>,

The screenshot shows the 'Vue Project Manager' application. The top navigation bar includes tabs for 'Projects' (selected), 'Create', and 'Import'. The main content area displays a large 'U' shaped icon and the text 'No existing projects'. The browser status bar at the bottom shows 'No project open' and the path '/Users/hafidmukhlasin/Dev/vue-projects'.

Pertama kita akan membuat projek baru. Pilih tab create. Tentukan lokasi direktori dimana projek kita akan disimpan.

Vue Project Manager

?

Projects + Create Import

Users hafidmukhlasin Dev vue-projects

^

browsify-vue

buff

myproject

+ Create a new project here

No project open /Users/hafidmukhlasin/Dev/vue-projects Ready on... 7/28/2019, 10:09:54 AM

Tekan tombol **Create a new project here**. Maka akan muncul wizard untuk membuat projek baru. Masukkan nama folder projek kita **vueshop**, pilih package manager **npm**

Create a new project

Details Presets Features Configuration

Project folder

vueshop

/Users/hafidmu.../Dev/vue-projects/vueshop

Package manager

npm

Additional options

Overwrite target folder if it exists

Cancel Next →

Klik **Next**, pilih preset manual di mana kita akan pilih sendiri plugins yang akan kita gunakan dalam projek kita.

Create a new project

≡ Details Presets  Features  Configuration

Manual

Manually select features

Remote preset

Fetch a preset from a git repository

← Previous

Next →

🏠 No project open

📁 /Users/hafidmukhlasin/Dev/vue-projects



Ready on... 7/28/2019, 10:09:54 AM



Klik Next, enable semua fitur kecuali Typescript dan E2E Testing.

Catatan: Typescript merupakan penulisan kode JS yang menggunakan kaidah static typing (pada buku ini materi tentang typescript tidak dibahas namun penulis menyarankan agar kamu mempelajari topik ini). Sedangkan E2E Testing, fitur ini untuk menjalankan end to end testing melalui browser.

Create a new project

≡ Details Presets  Features  Configuration

Babel

Transpile modern JavaScript to older versions (for compatibility) 



TypeScript

Add support for the TypeScript language 



Progressive Web App (PWA) Support

Improve performances with features like Web manifest and Service workers 



Router

Structure the app with dynamic pages 



Vuex

Manage the app state with a centralized store 



Fitur-fitur yang kita gunakan adalah:

- Babel merupakan fitur yang berfungsi mentranspile (konversi) dari modern Javascript ke versi sebelumnya untuk tujuan kompatibilitas.

- Progressive Web Application (PWA) merupakan fitur yang memungkinkan aplikasi kita nantinya tetap dapat berjalan tanpa ada koneksi ke server, serta dapat diinstalasi pada perangkat mobile layaknya native application.
- Router
- Vuex
- CSS Preprocessor merupakan fitur yang memungkinkan penulisan kode CSS dengan menggunakan cara-cara tertentu yang lebih efisien, untuk mempelajarinya lebih lanjut kamu bisa merujuk ke pustaka terkait seperti: [sass](#), [less](#), dan [stylus](#).
- Linter/Formatter, fitur ini digunakan untuk mengecek format penulisan kita agar sesuai dengan standard penulisan Javascript serta untuk mencegah kemungkinan error.
- Unit Testing, fitur ini untuk menjalankan unit testing
- Use config files, fitur ini akan memecah file konfigurasi berdasarkan pustakanya.

Create a new project

≡ Details ✅ Presets  Features ⚙ Configuration

CSS Pre-processors
Add support for CSS pre-processors like Sass, Less or Stylus [More Info](#) 

Linter / Formatter
Check and enforce code quality with ESLint or Prettier [More Info](#) 

Unit Testing
Add a Unit Testing solution like Jest or Mocha [More Info](#) 

E2E Testing
Add an End-to-End testing solution to the app like Cypress or Nightwatch [More Info](#) 

Use config files
Use specific configuration files (like '.babelrc') instead of using 'package.json'. 

[← Previous](#) [Next →](#)



Klik Next,

Pada tab configuration

- pilih history mode aktif, untuk pretty URL
- pilih stylus untuk CSS pre-processor
- pilih ESLint with error prevention only untuk linter/formatter (cocok untuk pemula).
- pilih Lint on save yaitu pengecekan otomatis (linter) dilakukan ketika kita menyimpan file.

- Untuk unit testing kita pilih Mocha + Chai,

Create a new project

Details Presets Features Configuration

Pick a CSS pre-processor:
PostCSS, Autoprefixer and CSS Modules are supported by default.

Stylus

Pick a linter / formatter config:
Checking code errors and enforcing an homogeneous code style is recommended.

ESLint with error prevention only

Pick additional lint features:

Lint on save
 Lint and fix on commit

Pick a unit testing solution:

Mocha + Chai

← Previous Create Project

Kemudian klik Create Project maka pada browser akan muncul tampilan prompt untuk menyimpan preset baru, sehingga jika nantinya kita ingin membuat projek baru dengan konfigurasi yang sama bisa digunakan kembali.

Save as a new preset

Preset name

vueshop-template

Save the features and configuration into a new preset

Cancel Continue without saving Create a new preset

Tekan tombol Create a new preset.

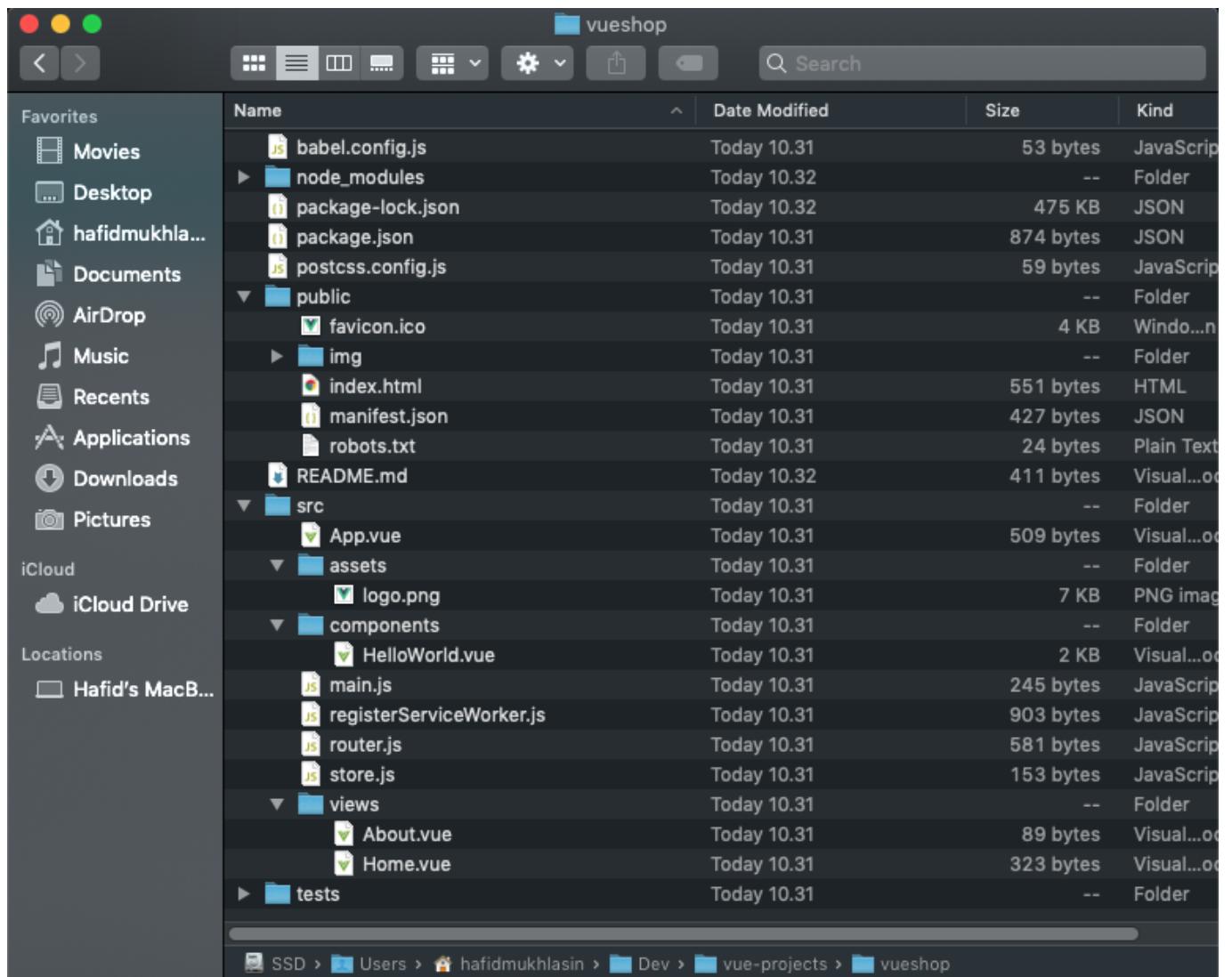


Installing Vue CLI plugins. This might take a while...

Di mana sebenarnya dibelakang layar alias di terminal, vue-cli menjalankan perintah untuk membuat projek baru

```
Hafids-MacBook-Pro:vue-projects hafidmukhlasin$ sudo vue ui
Starting GUI...
Ready on http://localhost:8000
Creating project in /Users/hafidmukhlasin/Dev/vue-projects/vueshop.
Initializing git repository...
(Progress bar) extract:resolve-from: sill extract resolve-from@^3.0.0
```

Jika create projek selesai, mari kita lihat struktur direktori projek yang telah dibuat.



Secara umum sama dengan struktur projek pada preset default yang telah kita buat sebelumnya (myproject) kecuali penambahan file karena ada beberapa fitur tambahan.

Router adalah fitur yang membedakan dengan projek sebelumnya. Buka file src/router.js.

```
1 import Vue from 'vue'
2 import Router from 'vue-router'
```

```

3 import Home from './views/Home.vue'
4
5 Vue.use(Router)
6
7 export default new Router({
8   mode: 'history',
9   base: process.env.BASE_URL,
10  routes: [
11    {
12      path: '/',
13      name: 'home',
14      component: Home
15    },
16    {
17      path: '/about',
18      name: 'about',
19      // route level code-splitting
20      // this generates a separate chunk (about.[hash].js) for this route
21      // which is lazy-loaded when the route is visited.
22      component: () => import(/* webpackChunkName: "about" */ 
23      './views/About.vue')
24    }
25  ]
26})

```

Berdasarkan konfigurasi di atas, terdapat dua routing yaitu /home dengan component Home (views/Home.vue) dan /about dengan component About (views/About.vue). Pada routing /about dicontohkan tentang component yang dimuat dengan konsep lazyload.

Lazy load merupakan design pattern di mana kita bisa menginisialisasi terlebih dahulu komponen yang akan digunakan, namun komponen itu hanya akan dimuat ketika dibutuhkan saja.

Adapun layout dari aplikasi bisa kita jumpai pada file src/App.vue yang juga berperan sebagai wrapper dari seluruh component Vue pada aplikasi.

```

1 <template>
2   <div id="app">
3     <div id="nav">
4       <router-link to="/">Home</router-link> |
5       <router-link to="/about">About</router-link>
6     </div>
7     <router-view/>
8   </div>
9 </template>
10 <style lang="stylus">
11 #app
12   font-family 'Avenir', Helvetica, Arial, sans-serif
13   -webkit-font-smoothing antialiased
14   -moz-osx-font-smoothing grayscale
15   text-align center
16   color #2c3e50
17
18 #nav
19   padding 30px

```

```

20   a
21     font-weight bold
22     color #2c3e50
23     &.router-link-exact-active
24       color #42b983
25
</style>

```

Pada kode di atas terdapat dua router-link untuk menu Home dan About. Adapun router-view sebagai target view dari component yang dimuat. Sedangkan pada blok bawah terdapat style CSS yang ditulis dengan bahasa stylus yaitu style penulisan CSS dengan menggunakan indentasi.

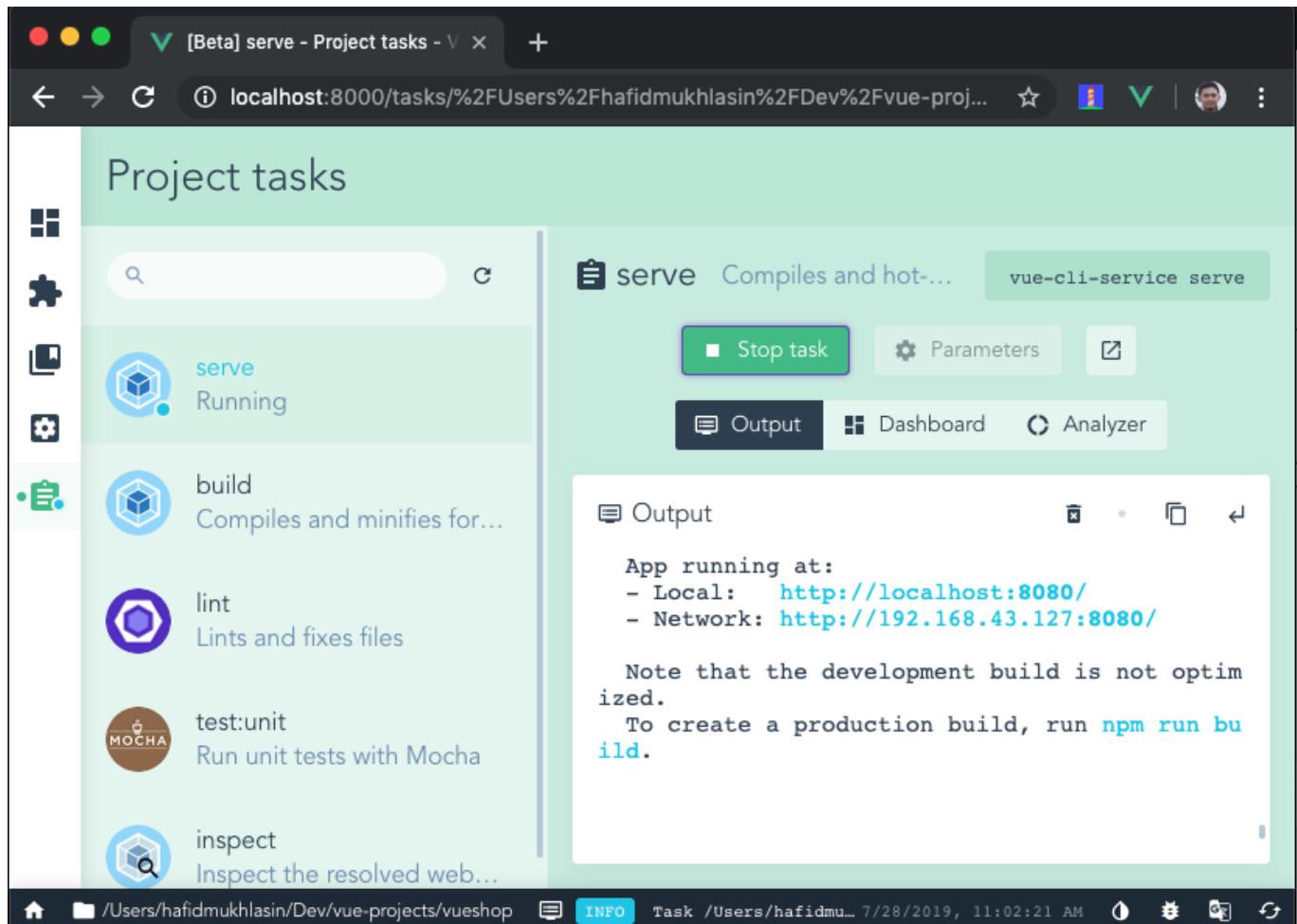
Pada views/Home.vue, dimuat components/HelloWorld.vue yang berisi link.

```

1  <template>
2    <div class="home">
3      
4      <HelloWorld msg="Welcome to Your Vue.js App"/>
5    </div>
6  </template>
7
8  <script>
9    // @ is an alias to /src
10   import HelloWorld from '@/components/HelloWorld.vue'
11
12  export default {
13    name: 'home',
14    components: {
15      HelloWorld
16    }
17  }
18 </script>

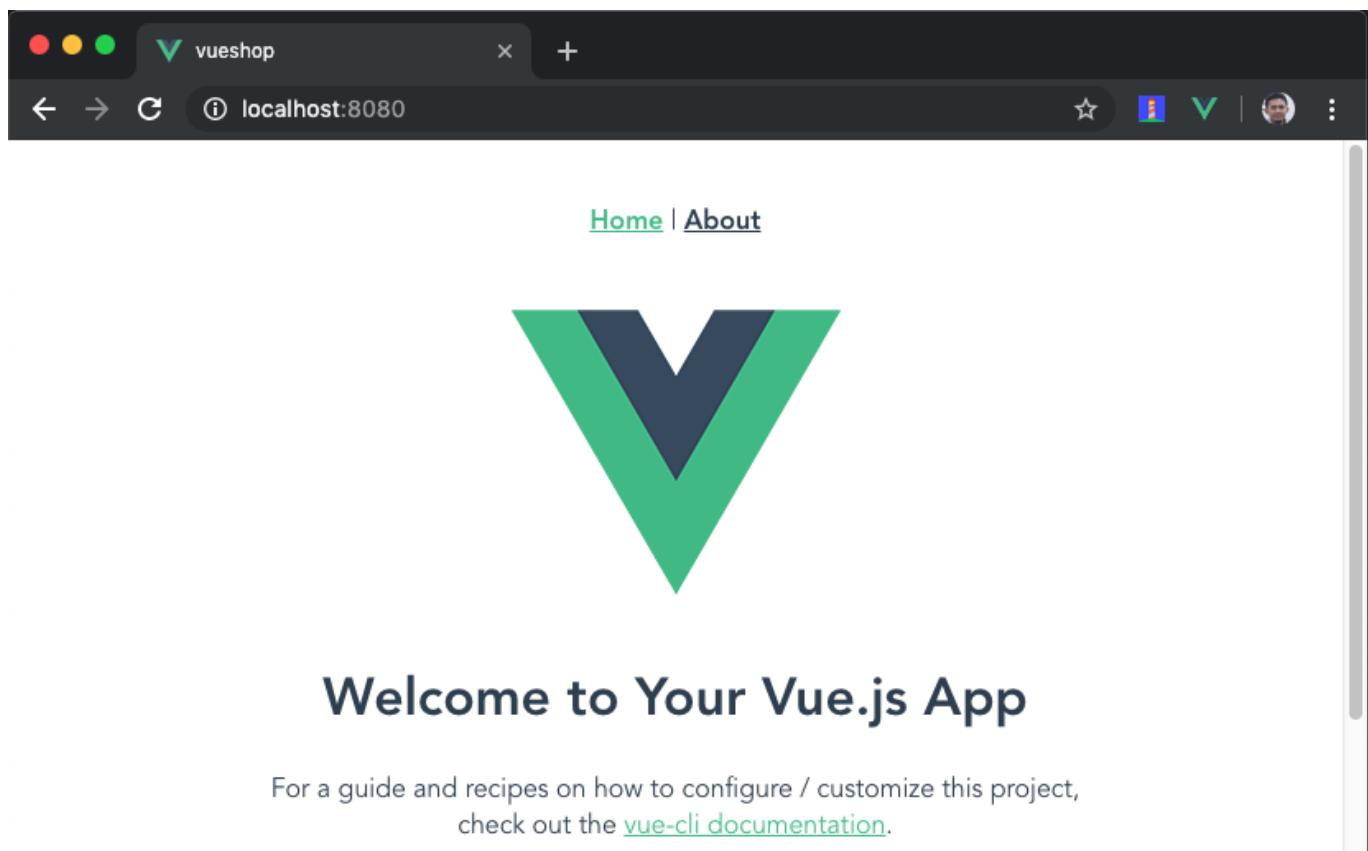
```

Sekarang waktunya menjalankan projek kita. Pada sidebar menu web vue-ui, pilih project task, lalu pilih serve, lalu klik tombol Run Task.



Catatan: perintah di atas, setara dengan perintah `npm run serve`

Hasilnya bisa kita lihat di browser pada alamat `http://localhost:8080`.



Klik halaman About maka akan muncul tampilan berikut.

This is an about page

Menambahkan Plugin Baru

Sebagaimana yang telah dijelaskan sebelumnya bahwa Vue CLI menggunakan konsep plugins untuk menambahkan fitur tertentu pada projek yang telah kita buat sebelumnya. Ada dua plugin yang kita akan tambahkan pada projek vueshop kita yaitu axios dan vuertify.

Plugin Axios

Axios "Promise based HTTP client for the browser and node.js" merupakan pustaka Javascript untuk http client yang cukup populer. Pada bagian yang lalu kita telah belajar menggunakan engine XMLHttpRequest pada browser untuk melakukan http request ke dan dari server, Axios menggunakan engine tersebut untuk melakukannya serta menambahkan kemampuan Javascript Promise sehingga lebih asynchronous friendly.

Fitur-fitur lain yang dimiliki oleh Axios antara lain: requestnya bisa di-intercept atau dibatalkan, responsenya otomatis dapat ditransform ke bentuk JSON, proteksi terhadap serangan XSRF. Kita bisa kunjungi repository officialnya pada tautan ini <https://github.com/axios/axios>

Umumnya browser modern telah mendukung Axios.

Latest ✓	8+ ✓				

Firefox	Chrome	IE	Edge	Safari
61 ✓	67 ✓	9 ✗	17 ✓	9 10.11 ✓
		10 ✓		
		11 ✓		

Untuk menginstalasi pustaka Axios sebagai plugins Vue CLI pada projek kita maka bisa melalui vue ui <http://localhost:8000/plugins>,

Project plugins

Installed plugins

- @vue/cli-service
version 3.0.1 latest 3.0.1 ★ Official ✓ Installed local service for vue-cli projects [More Info](#)
- @vue/cli-plugin-babel
version 3.0.1 latest 3.0.1 ★ Official ✓ Installed babel plugin for vue-cli [More Info](#)
- @vue/cli-plugin-eslint
version 3.0.1 latest 3.0.1 ★ Official ✓ Installed eslint plugin for vue-cli [More Info](#)
- @vue/cli-plugin-pwa
version 3.0.1 latest 3.0.1 ★ Official ✓ Installed pwa plugin for vue-cli [More Info](#)
- @vue/cli-plugin-unit-mocha
version 3.0.1 latest 3.0.1 ★ Official ✓ Installed mocha unit testing plugin for vue-cli [More Info](#)

klik button + Add plugin kemudian cari plugin Axios dengan kata kunci Axios yang pada tutorial ini menggunakan vue-cli-plugin-axios buatan @canhkieu, lalu instal.

Add a plugin

Search Configuration Files changed

axios

vue-cli-plugin-axios 0.0.4
Vue-cli-3 plugin: axios [Download 17.8K](#) [canhkieu](#) [Install](#)

Search by algolia Browse local plugin Cancel Install vue-cli-plugin-axios

17.7K [canhkieu](#)

Installing vue-cli-plugin-axios...

Pada tab configuration, klik Finish installation

The screenshot shows a browser window titled '[Beta] Add a plugin - Vue CLI'. The URL is 'localhost:8000/plugins/add'. The page content is titled 'Add a plugin' and shows 'Installation of vue-cli-plugin-axios'. It features a large checkmark icon and the text 'No configuration'. At the bottom are two buttons: 'Cancel' and 'Finish installation'.

Di akhir instalasi akan muncul konfirmasi bahwa file package.json akan dimodifikasi melalui penambahan dua pustaka yaitu Axios & vue-cli-plugin-axios.

The screenshot shows a browser window titled '[Beta] Add a plugin - Vue CLI'. The URL is 'localhost:8000/plugins/add'. The page content is titled 'Add a plugin' and shows 'Files changed' with a count of 4. It lists 'package-lock.json' and 'package.json'. At the bottom are two buttons: 'Commit changes' and 'Skip'.

Silakan

di-commit

perubahan

tersebut.

Commit changes X

Enter a commit message

 install axios

Record changes to the repository

Cancel

 Commit

Action pada vue ui ini mentrigger instalasi pustaka Axios pada terminal.

```
[^CHafids-MacBook-Pro:vueshop hafidmukhlasin$ vue ui
  Starting GUI...
  Ready on http://localhost:8000
+ vue-cli-plugin-axios@0.0.4
added 1 package from 1 contributor and audited 14306 packages in 16.337s
found 0 vulnerabilities

No prompts found for vue-cli-plugin-axios

  Invoking generator for vue-cli-plugin-axios...
  Installing additional dependencies...

added 1 package from 1 contributor and audited 14311 packages in 13.433s
found 0 vulnerabilities

  Running completion hooks...

✓ Successfully invoked generator for plugin: vue-cli-plugin-axios
  The following files have been updated / added:

    src/plugins/axios.js
    package-lock.json
    package.json
    src/main.js

  You should review these changes with git diff and commit them.
```

Apa yang dilakukan plugin Axios ini? plugin ini selain menginstalasi juga melakukan prekonfigurasi dengan menjadikannya sebagai sebuah plugin pada projek Vue kita sehingga bisa langsung digunakan.

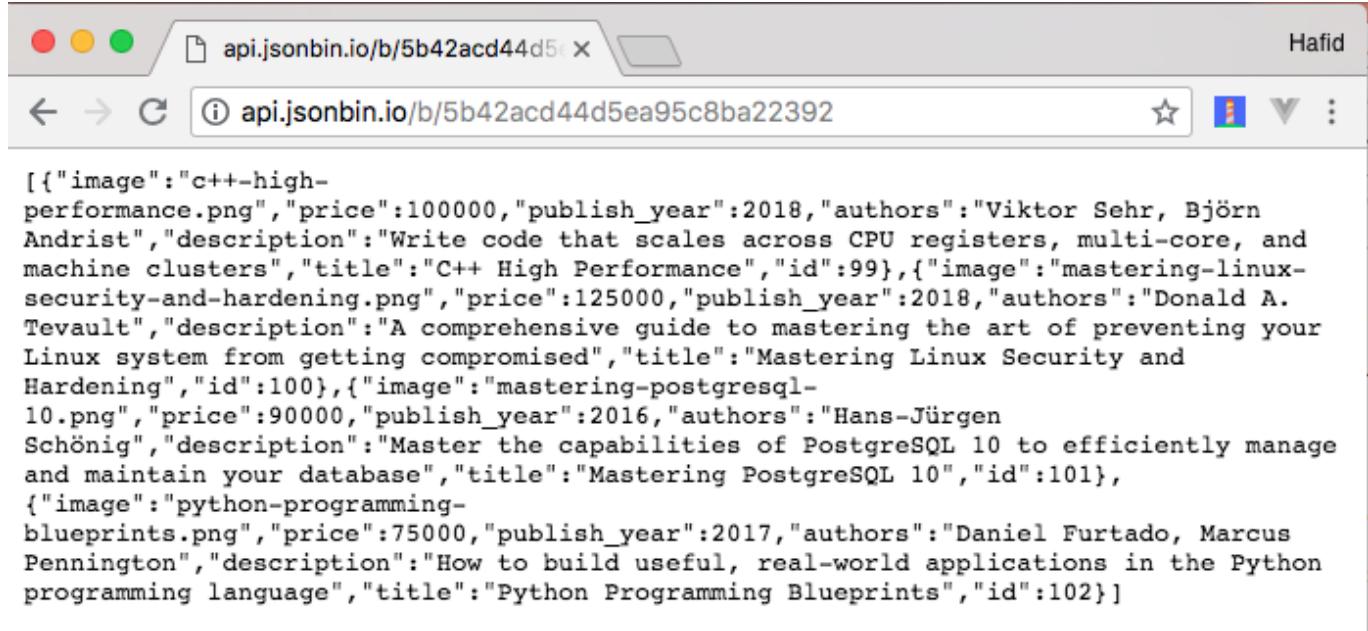
Plugin Axios bisa kita jumpai pada lokasi `src/plugins/axios.js`, di mana kita nantinya bisa melakukan konfigurasi jika diperlukan. Sedangkan pada file `src/main.js` yang merupakan file di mana objek Vue diciptakan telah *di-import* plugin tersebut.

Oleh karena itu, untuk menggunakannya maka kita bisa gunakan perintah `this.axios`

```
1  this.axios.get(URL_API_WEB_SERVICE)
2    .then(function (response) {
3      // handle success
4    })
5    .catch(function (error) {
6      // handle error
7    })
```

Sebagai contoh, kita akan gunakan data JSON buku sebagaimana yang telah digunakan pada bab terdahulu.

<http://api.jsonbin.io/b/5b42acd44d5ea95c8ba22392>



```
[{"image": "c++-high-performance.png", "price": 100000, "publish_year": 2018, "authors": "Viktor Sehr, Bj\u00f6rn Andr\u00e4st", "description": "Write code that scales across CPU registers, multi-core, and machine clusters", "title": "C++ High Performance", "id": 99}, {"image": "mastering-linux-security-and-hardening.png", "price": 125000, "publish_year": 2018, "authors": "Donald A. Tevault", "description": "A comprehensive guide to mastering the art of preventing your Linux system from getting compromised", "title": "Mastering Linux Security and Hardening", "id": 100}, {"image": "mastering-postgresql-10.png", "price": 90000, "publish_year": 2016, "authors": "Hans-J\u00fcrgen Sch\u00f6nig", "description": "Master the capabilities of PostgreSQL 10 to efficiently manage and maintain your database", "title": "Mastering PostgreSQL 10", "id": 101}, {"image": "python-programming-blueprints.png", "price": 75000, "publish_year": 2017, "authors": "Daniel Furtado, Marcus Pennington", "description": "How to build useful, real-world applications in the Python programming language", "title": "Python Programming Blueprints", "id": 102}]
```

Maka pada contoh ini, kita bisa tambahkan hook mounted pada component HelloWorld untuk sekedar menguji plugin axios. Berikut contoh penggunaanya.

```
1  export default {
2      name: 'HelloWorld',
3      props: {
4          msg: String
5      },
6      mounted(){
7          this.axios.get('http://api.jsonbin.io/b/5b42acd44d5ea95c8ba22392')
8              .then((response) => {
9                  console.log(response.data)
10             })
11             .catch((error) => {
12                 console.log(error)
13             })
14         }
15     }
```

Perintah di atas hanya akan menampilkan data json pada console browser. Buka web projek vueshop kita <http://localhost:8080>

For a guide and recipes on how to configure / customize this project, check out the [vue-cli documentation](#).

Installed CLI Plugins

- [babel](#)
- [pwa](#)
- [eslint](#)
- [unit-mocha](#)

Essential Links

- [Core Docs](#)
- [Forum](#)
- [Community Chat](#)
- [Twitter](#)
- [News](#)

Ecosystem

Lebih sederhana dibandingkan menggunakan XMLHTTP kan?

Plugin Vuetify

Vuetify merupakan framework user interface yang bertanggung jawab terhadap tampilan antarmuka aplikasi berbasis Vue. Vuetify yang menggunakan konsep Google material desain ini berupa kumpulan style dan component Vue yang akan memudahkan developer dalam mengembangkan aplikasi. Selengkapnya kita bisa mengunjungi website resminya <https://vuetifyjs.com>.



Sebenarnya ada banyak framework user interface yang mendukung Vue seperti [Element](#), [Quasar](#), [Bootstrap Vue](#), [Buefy](#), [Framework7](#), namun kemudian penulis memilih Vuetify karena beberapa sebab, diantaranya:

- Component yang berbasis mobilennya cukup lengkap
- Desainnya berbasis Material Design
- Komunitas yang cukup besar (bukan terbesar)
- Fiturnya cukup lengkap, seperti Progressive Web Application (PWA), Server-Side Rendering (SSR) dan Single Page Application melalui Vue Router
- Kustomisasi theme relatif mudah
- Terdapat plugin untuk Vue-CLI 3 (official)
- Mendukung berbagai web browser modern baik desktop maupun mobile.

Cara instalasinya jika menggunakan vue-ui, hampir sama dengan instalasi plugins Axios.

Add a plugin

Search Configuration Files changed

vuetify

vue-cli-plugin-vuetify 0.1.6
Plugin for vue cli 3 30.7K vuetifyjs

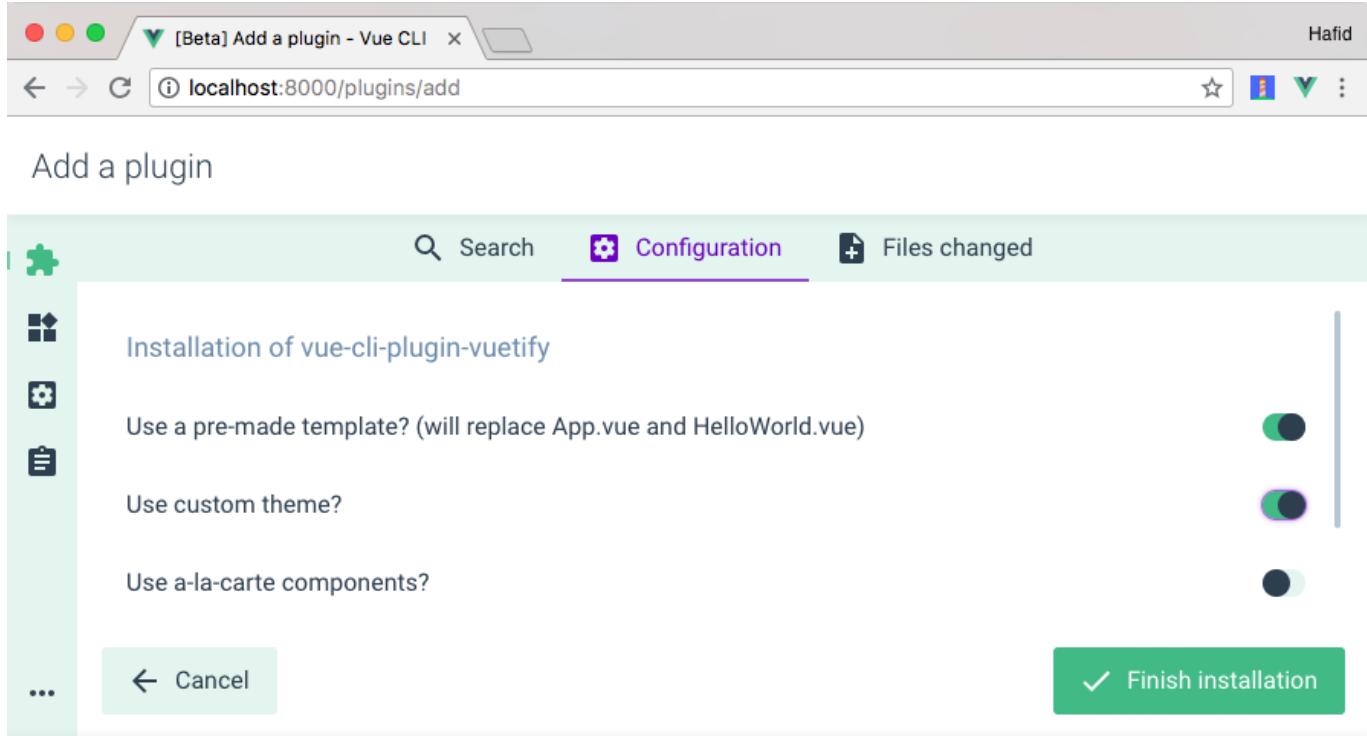
vue-cli-plugin-vuetify-cordova 0.0.13
Vuetify Cordova plugin. Make the world easy 5.7K canhkieu

Cancel Search by algolia Install vue-cli-plugin-vuetify

Catatan: jika menggunakan terminal, kita bisa menginstalasi dengan menjalankan perintah berikut `vue add vuetify`

Installing vue-cli-plugin-vuetify...

Untuk configuration, kita menggunakan default. Jika kita enable `A-la-carte components` maka kita dapat memilih component Vuetify tertentu saja yang diimport, namun pada tutorial ini kita import semua component Vuetify. Kita juga menggunakan premade template, dimana vuetify akan mencontohkan pada kita cara penggunaannya.



Intalasi ini akan mentriger proses instalasi di terminal.

```
vueshop — root@8a48a85542c9: /var/www/larashop-api — node /usr/local/bin/vue ui — 91...
+ vue-cli-plugin-vuetify@0.1.6
added 1 package from 1 contributor and audited 14312 packages in 16.737s
found 0 vulnerabilities

⚡ Invoking generator for vue-cli-plugin-vuetify...
📦 Installing additional dependencies...

added 2 packages from 2 contributors and audited 14316 packages in 20.511s
found 0 vulnerabilities

⚓ Running completion hooks...

✓ Successfully invoked generator for plugin: vue-cli-plugin-vuetify
The following files have been updated / added:

src/plugins/vuetify.js
babel.config.js
package-lock.json
package.json
public/index.html
src/App.vue
src/assets/logo.png
src/main.js
src/views/Home.vue

You should review these changes with git diff and commit them.
```

Terdapat beberapa perubahan yang harus kita commit.

The screenshot shows a web application interface for managing code changes. At the top, a header bar includes a title "[Beta] Add a plugin - Vue CLI", a user name "Hafid", and navigation links for back, forward, and search. Below the header is a toolbar with icons for search, configuration, and files changed. A sidebar on the left contains icons for file operations like add, move, and delete. The main area displays a list of files changed, with "babel.config.js" being the most recent. The code editor shows the following content:

```
1      1      module.exports = {  
2      -     presets: [  
3      -       '@vue/app'  
]
```

Below the code editor are two buttons: a green "Commit changes" button and a light blue "Skip" button. The status bar at the bottom shows the path "/Users/hafidmukhasin/Dev/vue-projects/vueshop" and a message "You should review these changes... 8/19/2018, 2:47:56 PM".

A modal window titled "Commit changes" is open. It contains a text input field with the placeholder "Enter a commit message" and a suggestion "install vuetyfy". Below the input field is a button labeled "Record changes to the repository". At the bottom of the modal are two buttons: "Cancel" and a green "Commit" button.

Waktunya uji coba di browser,
Mari kita coba akses melalui web browser.

The screenshot shows a web browser window with the address bar displaying 'localhost:8080'. The page title is '[Beta] serve - Project tasks - vueshop'. The main content features the Vuetify logo (a large blue 'V' composed of several triangles) and the heading 'Welcome to Vuetify'. Below the heading, there is a call to action: 'For help and collaboration with other Vuetify developers, please join our online [Discord Community](#)'. At the bottom, there is a section titled 'What's next?' with three links: 'Explore components', 'Select a layout', and 'Frequently Asked Questions'.

VUETIFY MATERIAL DESIGN

LATEST RELEASE

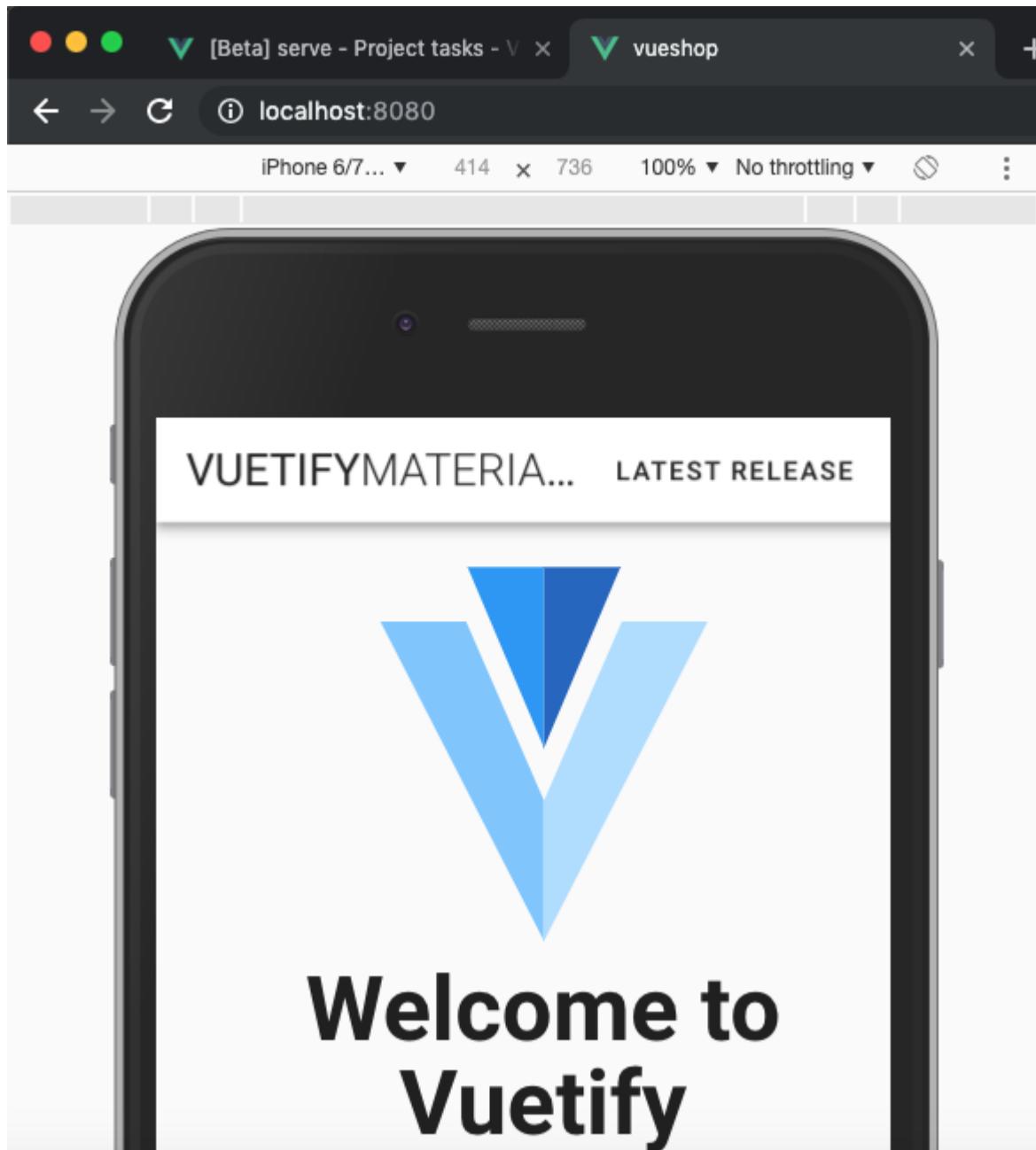
Welcome to Vuetify

For help and collaboration with other Vuetify developers,
please join our online [Discord Community](#).

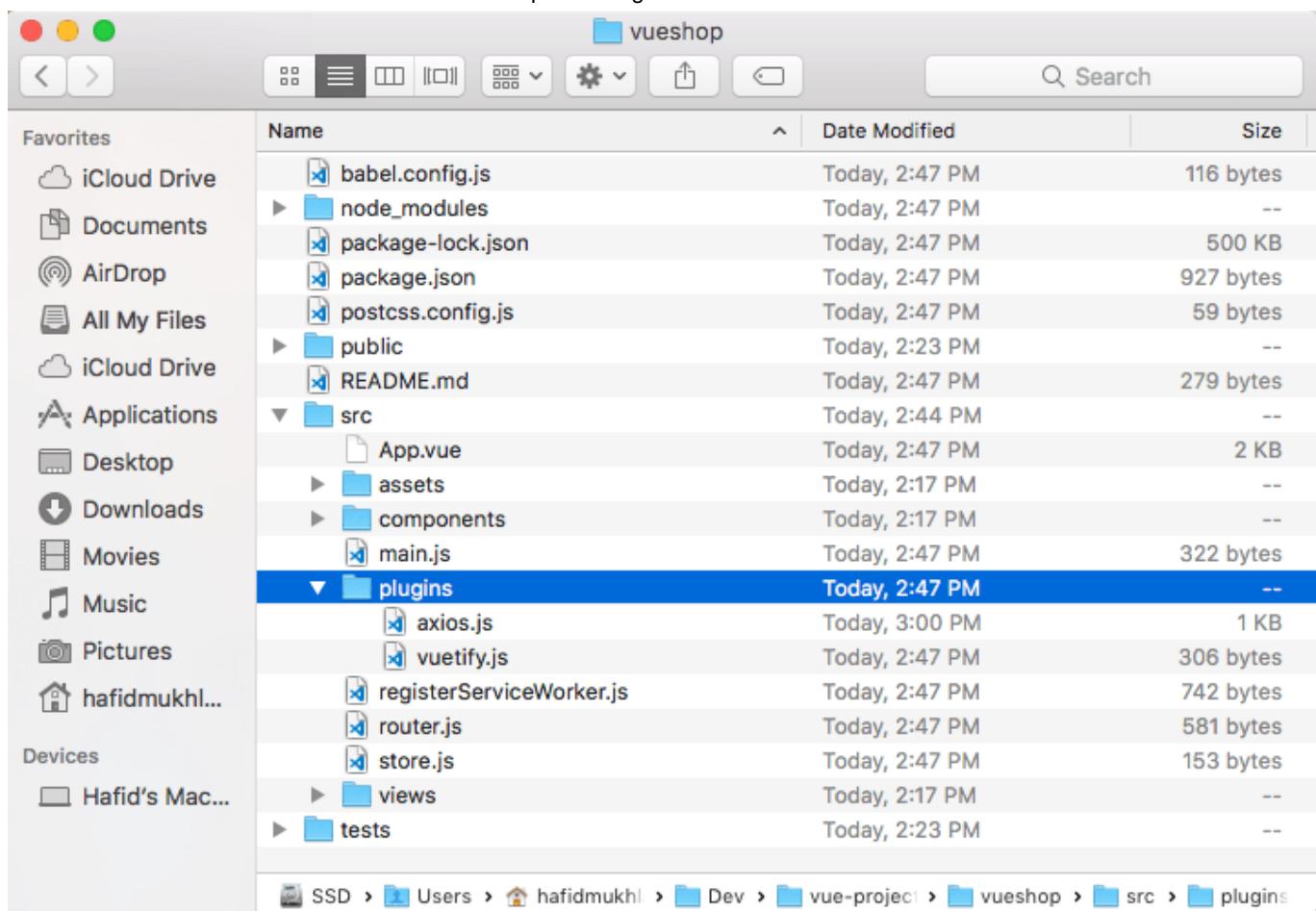
What's next?

[Explore components](#) [Select a layout](#) [Frequently Asked Questions](#)

Sedangkan jika diakses menggunakan preview web mobile sebagai berikut.



Tidak ada yang berubah dari struktur direktori kecuali penambahan file pada direktori `src/plugins`



Pengaturan theme atau warna pada vuetify terdapat pada file src/plugins/vuetify.js selengkapnya kamu bisa kunjungi <https://vuetifyjs.com/en/customization/theme>

Di sana, kita bisa menggunakan tools theme generator pada tautan ini <https://vuetifyjs.com/en/theme-generator>, untuk memilih warna yang kita suka.

Untuk memperlancar pemahaman kamu tentang Vuetify, maka penulis sarankan agar kamu membaca lebih banyak dokumentasi resminya serta mencoba beberapa componentnya. Bagaimana cara pengaturannya dan fitur-fitur apa yang dimiliki. Hal ini karena pada bab selanjutnya kita akan banyak menggunakan component Vuetify untuk membangun aplikasi studi kasus kita.

Kesimpulan

Sebelum membangun sebuah projek aplikasi, kita perlu persiapkan scaffolding-nya mulai dari struktur direktori, component dan pustaka yang akan digunakan hingga tools-tools yang digunakan untuk membantu pengembangan Aplikasi.

Vue mempunyai Vue CLI yang memudahkan kita membuat scaffolding aplikasi berbasis Vue dan menginstalasi plugin-plugin yang dibutuhkan. Vue CLI juga menyediakan tampilan user interface yang berbasis web untuk membuat scaffolding tersebut.

Bab selanjutnya kita akan belajar tentang cara membuat web service sendiri dengan menggunakan sebuah framework PHP populer yaitu Laravel.

Ayoo..! sedikit lagi

Web Service

Intro

Pada bab ini kita akan belajar tentang *web service* dan bagaimana interaksinya dengan Vue.

Mengenal Web Service

Meski pada bagian sebelumnya kita telah sedikit mencicipi *web service* namun alangkah baiknya kita memahami definisi dan cara kerja dari *web service*. Oke cekidot!.

Definisi Web Service

Web service merupakan standard yang digunakan untuk pertukaran data antar aplikasi atau sistem berbasis web. Standard ini diperlukan karena masing-masing aplikasi bisa jadi memiliki format data yang berbeda, ditulis dengan bahasa pemrograman yang berbeda, dan berjalan pada *platform* berbeda. Dengan adanya standard tersebut akan memungkinkan dua aplikasi berinteraksi satu sama lain dengan baik.

Pada contoh terdahulu, dengan menggunakan *tools* web <http://jsonbin.io> kita bisa membuat service penyedia data buku berformat JSON dan tentunya berbasis web.

Standard Web Service

Setidaknya ada dua standard *web service* yang bisa digunakan yaitu SOAP dan REST. Namun yang umum digunakan di dunia web adalah REST, karenanya pada tutorial ini kita akan fokus terhadap standard REST. REST merupakan singkatan dari *REpresentational State Transfer* yaitu standard dalam arsitektur web yang menggunakan protokol HTTP untuk pertukaran data.

Konsep REST pertama kali diperkenalkan oleh Roy Fielding pada tahun 2000. Secara sederhana konsep ini dapat dijelaskan bahwa REST server menyediakan jalur untuk akses *resource* atau data, sedangkan REST *client* melakukan akses *resource* dan kemudian menampilkan atau menggunakannya. *Resource* yang dihasilkan sebenarnya berupa teks, namun formatnya bisa bermacam-macam tergantung kebutuhan, umumnya adalah JSON dan XML.

Web services yang berbasis arsitektur REST kemudian dikenal sebagai RESTful Web Services.

Method Web Service

Dalam mengakses sebuah *resource*, REST juga menggunakan konsep *uniform resource identifier* (URI) di mana ada *method* yang digunakan, secara *default* adalah GET. Berikut ini *method-method* yang mendukung REST:

- GET, cocok untuk *resource* yang hanya perlu dibaca saja (*read only*)
- PUT, cocok digunakan untuk membuat/*create resource* baru.
- DELETE, cocok digunakan untuk menghapus suatu *resource*.
- POST, cocok digunakan untuk memodifikasi suatu *resource*.
- OPTIONS, cocok digunakan untuk mendapatkan operasi yang di-*support* pada *resource*.

Cara Kerja Web Service

Cara kerja *web service* berbasis REST cukup sederhana yaitu mula-mula suatu *client* mengirimkan permintaan data melalui protokol HTTP (HTTP *request*) pada *endpoint* (alamat URL dari suatu *resource*) tertentu dan kemudian server merespon permintaan tersebut (HTTP *response*).

Adapun komponen dari HTTP *request* adalah:

- Verb, HTTP *method* yang digunakan misalnya GET, POST, DELETE, PUT dll.
- URI, *endpoint* untuk mengidentifikasi lokasi *resource* pada server.
- HTTP version, menunjukkan versi dari HTTP yang digunakan, contoh HTTP v1.1.

- *Request header*, berisi *meta* data untuk HTTP *request*. Contoh, jenis *client/browser*, format yang didukung oleh *client*, format dari *body* pesan, pengaturan *cache* dll.
- *Request body*, konten dari *resource*.

Sedangkan komponen dari HTTP *response* adalah:

- Status/*response code*, mengindikasikan status server terhadap *resource* yang di-*request*. misal : 404, artinya *resource* tidak ditemukan dan 200 artinya OK.
- HTTP *version*, menunjukkan versi dari HTTP yang digunakan, contoh HTTP v1.1.
- *Response header*, berisi *meta* data untuk HTTP *response*. Contoh: jenis *server*, panjang konten, jenis konten, waktu *response*, dll
- *Response body*, konten dari *resource* yang diberikan.

HTTP Response Code

Sebagaimana yang telah disebutkan sebelumnya bahwa setiap terjadi *request* ke server melalui protokol HTTP maka akan mengembalikan respon yang salah satunya adalah kode respon. Kode ini mengindikasikan status server terhadap *resource* yang di-*request* tersebut.

Secara umum kode respon tersebut dapat dikelompokkan menjadi lima kelompok.

Kelompok Kode	Keterangan
1xx	<i>Informational response</i>
2xx	<i>Success</i>
3xx	<i>Redirection</i>
4xx	<i>Client errors</i>
5xx	<i>Server errors</i>

Berikut ini beberapa kode respon yang umum terjadi ketika bermain dengan *web service*.

Code	Status	Keterangan
200	OK	<i>Request resource berhasil</i>
301	<i>Move Permanently</i>	<i>Redirect ke resource lain</i>
304	<i>Not Modified</i>	<i>Resource belum berubah</i>
400	<i>Bad Request</i>	Terjadi kesalahan pada <i>request client</i>
401	<i>Unauthorized</i>	Belum <i>login</i> sebagai <i>authorize user</i>
403	<i>Forbidden</i>	Server menolak akses ke <i>resource</i>
404	<i>Not found</i>	<i>Resource</i> tidak ditemukan
405	<i>Method not allowed</i>	<i>Method</i> salah, misal: harusnya menggunakan POST tapi <i>request</i> menggunakan GET
500	<i>Server error</i>	<i>Server error</i>

Selengkapnya, kamu bisa baca pada tautan ini https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

States pada Web Service

Dalam arsitektur REST, seharusnya tidak boleh ada penyimpanan state atau session untuk mengidentifikasi suatu client pada suatu request, artinya setiap request dari client harus bersifat independen dan dianggap sebagai request baru. Hal ini disebut sebagai stateless. Tujuan utama dari stateless adalah memudahkan peningkatan (scale-up) concurrent access terhadap web service.

Oleh karena aplikasi bisa jadi tetap membutuhkan penanda suatu client, misalnya resource X hanya boleh diakses oleh client yang telah terdaftar, maka kemudian digunakan mekanisme tertentu untuk membedakan suatu request apakah berasal dari client yang sudah terdaftar atau belum. Mekanisme yang umum digunakan adalah dengan menggunakan token.

Mula-mula client melakukan request data login, kemudian server melakukan pengecekan untuk memastikan data login tersebut valid. Jika data login valid maka server menggenerate dan mengembalikan suatu token dimana token tersebut untuk mengidentifikasi client yang telah login. Pada request berikutnya, client dapat menyisipkan token tersebut sehingga server penerima dapat mengenali client yang melakukan request tersebut.

Persiapan Tools Pengembangan

Ada beberapa tools yang perlu kita persiapkan pada komputer kita untuk mengembangkan web service. Pada tutorial ini kita akan belajar membuat web service menggunakan bahasa pemrograman PHP dan framework Laravel. Oleh karenanya kita perlu persiapkan lingkungan kerja yang mendukung framework laravel.

Bahasa Pemrograman: PHP

PHP merupakan bahasa pemrograman populer yang bersifat server side atau dijalankan di sisi server, cocok untuk digunakan mengembangkan aplikasi berbasis web termasuk juga web service baik projek sekala kecil maupun besar.

PHP dapat diinstalasi pada semua platform OS, silakan merujuk ke tautan berikut <http://php.net/install>

Buku ini tidak akan membahas secara mendalam tentang PHP, oleh karenanya jika kamu masih belum lancar dalam menggunakan bahasa ini silakan merujuk ke dokumentasinya langsung pada tautan berikut: <http://php.net/download-docs.php>

Database Server: MySQL, MariaDB

Untuk menyimpan dan mengelola data buku pada toko buku kita, maka kita perlu suatu media penyimpanan yang pada hal ini disebut sebagai database. Pada tutorial ini, kita akan menggunakan salah satu database yang cukup populer dan sering disandingkan dengan PHP yaitu MySQL <https://www.mysql.com>.

MySQL merupakan relational database management system yaitu sistem manajemen database yang mendukung relasional data antar tabelnya.

Buku ini juga tidak akan membahas tentang dasar-dasar database MySQL melainkan langsung penerapannya. Oleh karenanya jika kamu ingin mempelajari lebih dalam lagi tentang database ini bisa mengunjungi dokumentasi resminya pada tautan berikut: <https://dev.mysql.com/doc/>

Web Server: Nginx, Apache

Web server merupakan software yang bertugas melayani request dari client yang dalam hal ini request dilakukan menggunakan browser dengan protokol http/s. PHP sendiri sebenarnya mempunyai built-in web server yang pada bab sebelumnya telah kita gunakan.

`php -S localhost`

Namun untuk production sebaiknya kita gunakan web server yang sudah teruji seperti misalnya Nginx dan Apache.

Git

Git merupakan tools untuk mengelola source code aplikasi seperti: menduplikasi projek, mencatat perubahan kode, melakukan pelacakan histori perubahan kode, mengunggah kode, dsb.



Untuk menginstalasi tools git silakan merujuk ke tautan berikut <https://git-scm.com/downloads>, dimana git menyediakan paket instalasi yang cukup mudah.

Cek versi git.

```
book — root@c5ac5cd7e2ae: /var/www — bash — 74x32
[Hafids-MacBook-Pro:book hafidmukhlasin$ git --version
git version 2.14.3 (Apple Git-98)
Hafids-MacBook-Pro:book hafidmukhlasin$ ]
```

Package Tools

Kita dapat menginstalasi tools atau software (PHP, MySQL/MariaDB, Web Server) yang dibutuhkan dalam pengembangan ini secara terpisah namun untuk lebih mudahnya kita bisa gunakan package tools yang telah menyediakan semua tools yang dibutuhkan dalam satu paket instalasi.

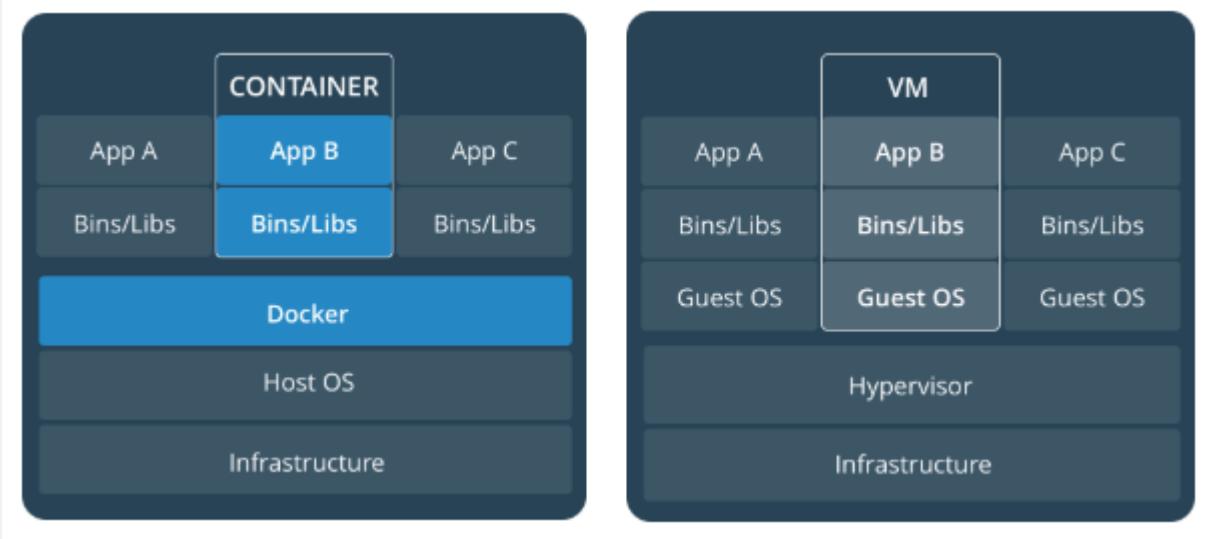
Ada banyak paket yang bisa kita gunakan, misalnya: Docker (Laradock), XAMPP, dan Homestead. Tentu saja jangan kamu install semua, melainkan pilih salah satu saja, adapun penulis menggunakan Docker.

Docker

Apa itu docker? [docker](#) merupakan tools virtualisasi berbasis container (wadah) yang mengizinkan kita membuat paket-paket aplikasi beserta pustaka yang dibutuhkannya dalam sebuah container yang terisolasi satu sama lainnya.



Docker berbeda dengan virtual machine yang memvirtualiasi sistem operasi secara utuh, docker hanya memvirtualiasi dilevel aplikasi.



Docker saat ini tidak hanya tersedia untuk sistem operasi berbasis linux namun juga secara native bisa berjalan di OS Windows melalui hyper-V (Windows 10 Pro).

Keuntungan jika kamu menggunakan Docker adalah kamu bisa menyamakan environment saat development dengan environment di server saat production. Permasalahan yang sering terjadi misalnya kode tidak jalan ketika sudah di deploy ke server padahal di development running well ini bisa di hindari. Contoh lain extension tertentu tidak kompatibel atau lupa belum diinstall juga tidak akan terjadi, karena environment server di development benar-benar sama dengan production. Di samping itu, environment komputer kita akan lebih bersih dari pengaturan terkait tools-tools yang dibutuhkan untuk menjalankan aplikasi.

Oleh karena itu, penulis menyarankan agar kamu sebisa mungkin menggunakan Docker.

Instalasi Docker

Silakan ikuti panduan instalasi docker pada tautan berikut:

- MacOS <https://docs.docker.com/docker-for-mac/install/>
- Windows <https://docs.docker.com/docker-for-windows/install/>
- Linux Ubuntu <https://docs.docker.com/install/linux/docker-ce/ubuntu/>

Setelah melakukan instalasi, jalankan docker



Kemudian untuk memastikan docker terinstalasi dengan baik sekaligus mengecek versi dari docker, jalankan perintah docker `-v` pada terminal.

```
book — root@c5ac5cd7e2ae: /var/www — -bash — 74x32
[Hafids-MacBook-Pro:book hafidmukhlasin$ docker -v
Docker version 18.03.1-ce, build 9ee9f40
Hafids-MacBook-Pro:book hafidmukhlasin$ ]
```

Instalasi Laradock

Laradock awalnya merupakan prekonfigurasi docker untuk pengembangan projek berbasis Laravel framework, namun kemudian laradock ini dapat digunakan untuk framework PHP apapun. Penulis merekomendasikan kamu untuk menggunakan laradock.

Setelah docker berhasil terinstalasi dengan baik, langkah selanjutnya adalah menginstalasi Laradock (<http://laradock.io/>)

Namun, untuk memudahkan kita kedepannya, kita akan buat terlebih dahulu folder bernama `laravel-projects` di mana pada folder ini kita akan meletakkan projek-projek berbasis laravel.

```
[Hafids-MacBook-Pro:Dev hafidmukhlasin$ mkdir laravel-projects
[Hafids-MacBook-Pro:Dev hafidmukhlasin$ cd laravel-projects
[Hafids-MacBook-Pro:laravel-projects hafidmukhlasin$ ]
```

Kemudian, pada terminal jalankan perintah berikut untuk mengcloning projek laradock.

```
1 | git clone https://github.com/Laradock/laradock.git
```

```
[Hafids-MacBook-Pro:laravel-projects hafidmukhlasin$ git clone https://github.com/Laradock/laradock.git
Cloning into 'laradock'...
remote: Counting objects: 7963, done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 7963 (delta 7), reused 9 (delta 1), pack-reused 7944
Receiving objects: 100% (7963/7963), 7.59 MiB | 471.00 KiB/s, done.
Resolving deltas: 100% (4201/4201), done.
[Hafids-MacBook-Pro:laravel-projects hafidmukhlasin$ ]
```

Masuk ke direktori laradock, kemudian copy dan rename file `env-example` menjadi `.env`. Jalankan perintah berikut.

```
[Hafids-MacBook-Pro:laravel-projects hafidmukhlasin$ cd laradock/
[Hafids-MacBook-Pro:laradock hafidmukhlasin$ cp env-example .env
[Hafids-MacBook-Pro:laradock hafidmukhlasin$ ]
```

File `.env` ini berisi pengaturan default atau prekonfigurasi yang dilakukan oleh laradock. Salah satunya pengaturan lokasi di mana projek aplikasi kita nantinya disimpan.

```
1 | # Point to the path of your applications code on your host
2 | APP_CODE_PATH_HOST=...
```

`.. /` artinya file projek aplikasi kita bisa disimpan pada folder yang sejajar dengan folder laradock. Sehingga struktur folder "laravel-projects" akan menjadi sebagai berikut.

- laravel-projects
 - laradock
 - project1
 - project2

Menjalankan Container

Kemudian kita bisa menjalankan container nginx, mysql, phpmyadmin dan workspace menggunakan perintah `docker-compose up` berikut.

```
1 | docker-compose up -d nginx mysql phpmyadmin workspace
```

```
[Hafids-MacBook-Pro:laradock hafidmukhlasin$ docker-compose up -d nginx mysql php]
myadmin workspace
Recreating laradock_mysql_1      ... done
Recreating laradock_workspace_1   ... done
Recreating laradock_docker-in-docker_1 ... done
Recreating laradock_php-fpm_1     ... done
Recreating laradock_phpmyadmin_1  ... done
Recreating laradock_nginx_1       ... done
Hafids-MacBook-Pro:laradock hafidmukhlasin$
```

Perintah ini akan menjalankan container nginx (php-fpm), mysql, phpmyadmin (tools manajemen database mysql) dan workspace (area kerja kita).

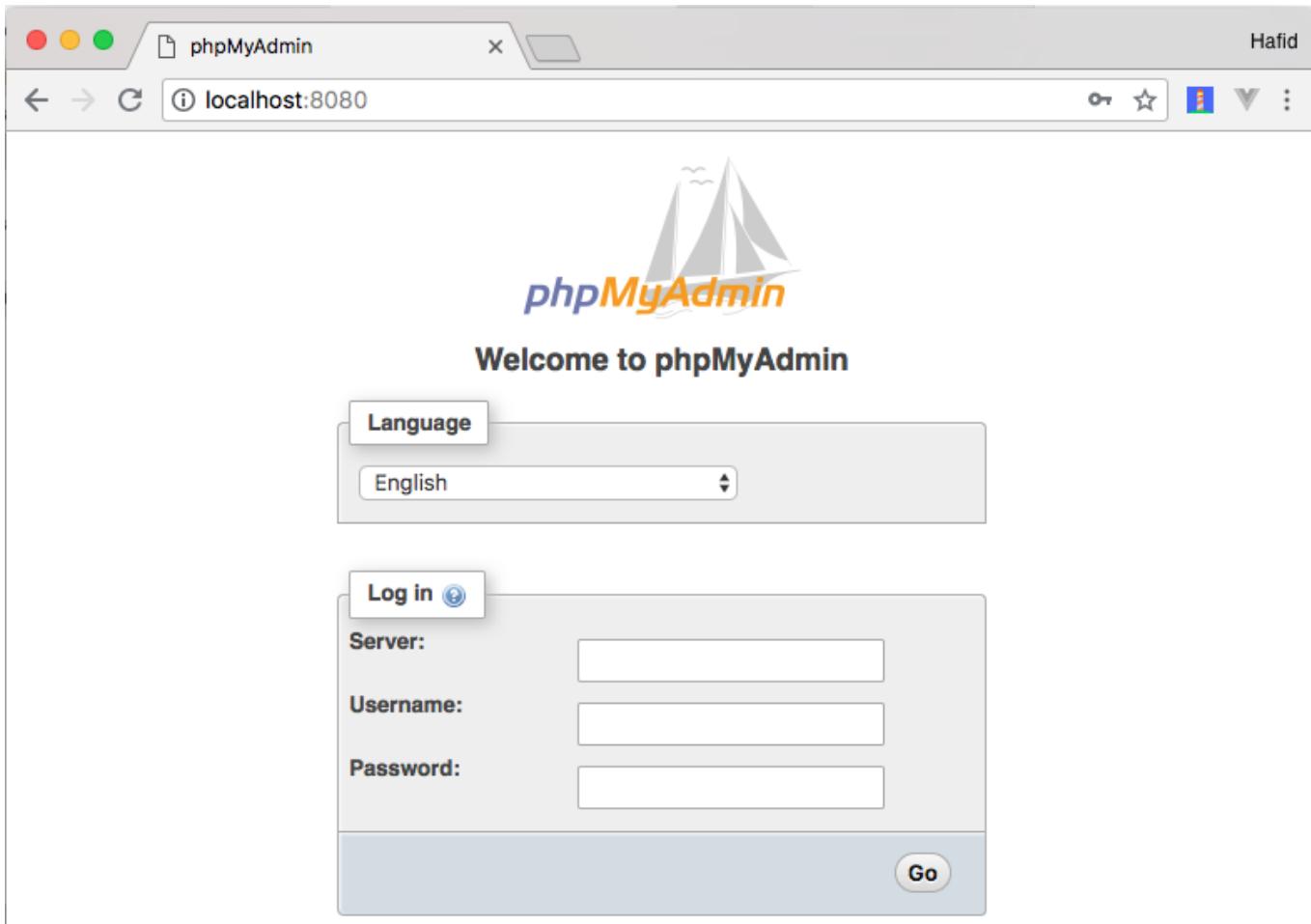
Setiap container merujuk ke image dari aplikasi yang dibungkusnya, misalnya container mysql berisi image aplikasi mysql yang oleh docker dihosting pada repositori docker image di <https://hub.docker.com> (tepatnya di https://hub.docker.com/_/mysql/). Oleh karena itu, ketika pertama kali kita menjalankan `docker-compose up` maka docker akan mengunduh file image jika dilocal belum ada yang tentunya membutuhkan koneksi internet.

Selanjutnya kita bisa mengakses web melalui browser dengan alamat `http://localhost`,



Muncul eror `404 Not Found` karena kita memang belum membuat projek dan kita juga belum melakukan pengaturan pada konfigurasi nginx untuk dapat mengakses projek kita.

Sedangkan phpmyadmin dapat kita akses dengan cara yang sama namun menggunakan port 8080, `http://localhost:8080`.



Secara default, isian server bisa kita isi dengan `mysql`, username `root` dan password `root`, hal ini sebagaimana pengaturan pada file `.env`

```

1 MYSQL_VERSION=latest
2 MYSQL_DATABASE=default
3 MYSQL_USER=default
4 MYSQL_PASSWORD=secret
5 MYSQL_PORT=3306
6 MYSQL_ROOT_PASSWORD=root
7 MYSQL_ENTRYPOINT_INITDB=./mysql/docker-entrypoint-initdb.d

```

Catatan: jika kamu gagal login ke mysql via phpmyadmin maka lakukan langkah pada bagian "Mengatasi Bug pada MySQL".

Mengatasi Bug Pada MySQL

Pada saat buku ini ditulis, terdapat bug terkait mysql versi 8.0, selengkapnya silakan baca pada tautan berikut: <https://github.com/laradock/laradock/issues/1392>, maka sebaiknya kita *downgrade* versi dari mysql yang sebelumnya `latest` menjadi versi `5.7`. Untuk melakukannya, edit file `.env`.

```

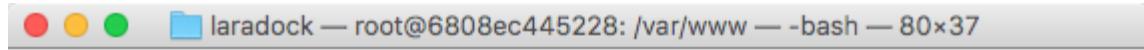
1 # MYSQL_VERSION=latest
2 MYSQL_VERSION=5.7

```

Kemudian, masih pada file `.env`, ubah lokasi di mana data mysql disimpan, dari `~/.laradock/data` menjadi misalnya: `~/.laradock/data2`.

```
1 # DATA_PATH_HOST=~/.laradock/data
2 DATA_PATH_HOST=~/.laradock/data2
```

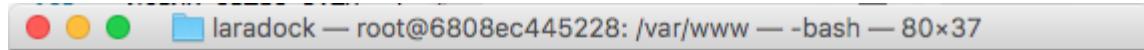
Pada terminal folder laradock jalankan perintah docker-compose down untuk memastikan semua container tidak sedang berjalan.



```
[Hafids-MacBook-Pro:laradock hafidmukhlasis$ docker-compose down
Stopping laradock_nginx_1 ... done
Stopping laradock_phpmyadmin_1 ... done
Stopping laradock_php-fpm_1 ... done
Stopping laradock_workspace_1 ... done
Stopping laradock_mysql_1 ... done
Stopping laradock_docker-in-docker_1 ... done
Removing laradock_nginx_1 ... done
Removing laradock_phpmyadmin_1 ... done
Removing laradock_php-fpm_1 ... done
Removing laradock_workspace_1 ... done
Removing laradock_mysql_1 ... done
Removing laradock_docker-in-docker_1 ... done
Removing network laradock_frontend
Removing network laradock_backend
Removing network laradock_default
Hafids-MacBook-Pro:laradock hafidmukhlasis$ ]
```

Lalu build ulang container mysql dengan menjalankan perintah berikut.

```
1 docker-compose build mysql
```



```
[Hafids-MacBook-Pro:laradock hafidmukhlasis$ docker-compose build mysql
Building mysql
Step 1/9 : ARG MYSQL_VERSION=latest
Step 2/9 : FROM mysql:${MYSQL_VERSION}
--> 66bc0f66b7af
Step 3/9 : LABEL maintainer="Mahmoud Zalt <mahmoud@zalt.me>"
--> Using cache
--> d2fc5f6600f0
--> 
```

Solusi yang lain adalah dengan menggunakan fitur upgrade dari mysql untuk mengatasi masalah ini. Caranya, masuk ke container mysql

```
1 docker exec -it mysql bash
```

Kemudian login ke mysqlnya, lalu jalankan perintah berikut.

```
1 mysql -u root -p
2 mysql> SET GLOBAL innodb_fast_shutdown = 1;
```

Ketika exit atau tekan CTRL+C untuk keluar dari container mysql, kemudian jalankan perintah mysql upgrade berikut.

```
1 mysql_upgrade -u root -p
```

Dan silakan dicoba login lagi.

Catatan: bagian ini tidak perlu kamu lakukan jika memang pada saat kamu instalasi laradock secara normal kamu sudah bisa login ke phpmyadmin. Mungkin perlu saya koreksi, bahwa permasalahan ini terjadi salah satunya karena perbedaan penggunaan plugin authentication password dimana pada versi 5.7 ke bawah secara default mysql menggunakan mysql_native_password sedangkan pada versi 8.0 ke atas yang digunakan adalah caching_sha2_password

Masuk Ke Workspace Container

Kita diizinkan masuk ke dalam container, layaknya meremote suatu server. Caranya, masuk masuk direktori laradock, kemudian jalankan perintah berikut.

```
1 | docker-compose exec workspace bash
```



Direktori /var/www pada container workspace ini sama dengan direktori yang diatur pada file .env tepatnya pada APP_CODE_PATH_HOST=.../ yaitu secara default sejajar dengan direktori folder laradock. Penulis menyebutnya sebagai webroot

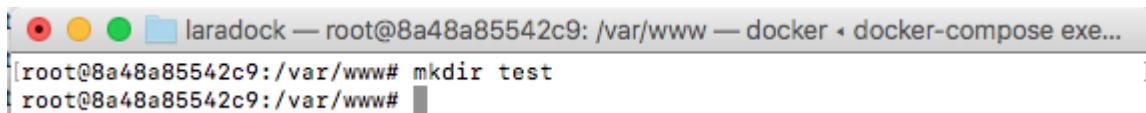
Catatan: webroot pada komputer penulis terletak di ~/Dev/laravel-projects/.

Sehingga jika pada direktori /var/www/ kita buat suatu folder baru (mkdir) maka folder tersebut dapat kita jumpai juga pada direktori webroot.

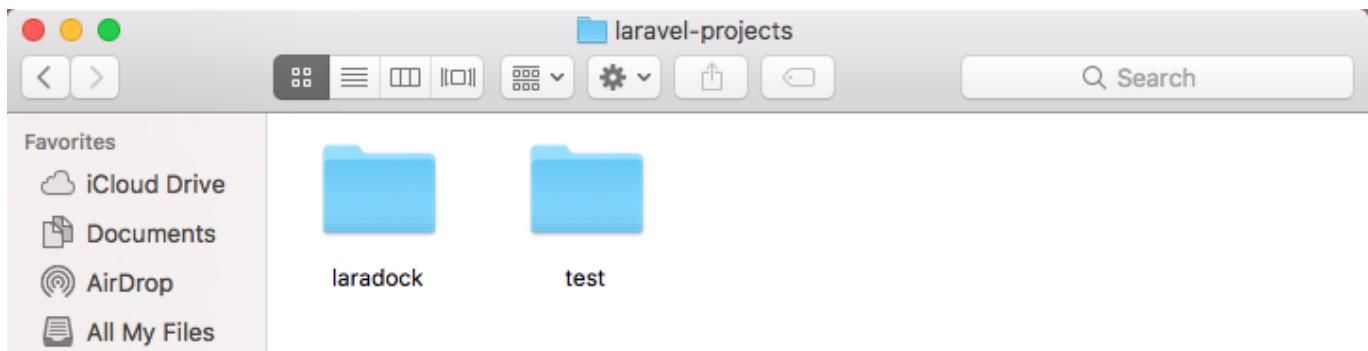
Untuk keluar dari workspace, ketik perintah exit.

Uji Coba Membuat Projek Baru

Pada terminal workspace, buat folder test menggunakan perintah mkdir.



Buka direktori webroot menggunakan file explorer (finder), maka akan kita jumpai folder bernama test



Untuk sekedar menguji coba, pada folder test tersebut buat file index.php yang isinya sebagai berikut.

```
1 | <?php phpinfo(); ?>
```

Kode di atas digunakan untuk menampilkan informasi terkait konfigurasi PHP.

Kemudian untuk mengaksesnya pada browser, kita perlu ubah konfigurasi pada nginx yang terletak di /laradock/nginx/sites/default.conf

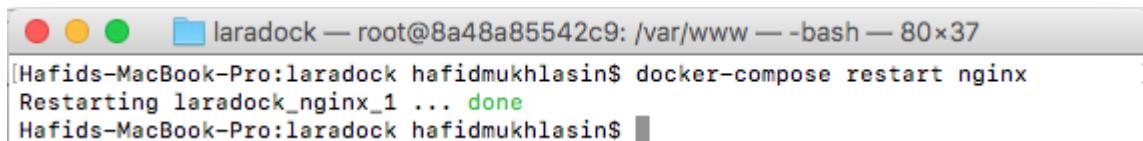
Catatan: Pada komputer penulis terletak di ~/Dev/laravel-projects/laradock/nginx/sites/default.conf, silakan disesuaikan.

Ubah pengaturan root dari /var/www/public menjadi /var/www

```
1 # root /var/www/public;
2 root /var/www;
```

Karena kita telah melakukan perubahan pada konfigurasi nginx, maka kita perlu merestart container nginx supaya perubahan tersebut diterapkan.

```
1 docker-compose restart nginx
```



```
[Hafids-MacBook-Pro:laradock hafidmukhlasin$ docker-compose restart nginx
Restarting laradock_nginx_1 ... done
Hafids-MacBook-Pro:laradock hafidmukhlasin$ ]
```

Lalu pada browser, kita bisa mengaksesnya pada alamat <http://localhost/test>



System	Linux 4a446a9769c7 4.9.87-linuxkit-aufs #1 SMP Wed Mar 14 15:12:16 UTC 2018 x86_64
Build Date	Apr 18 2017 19:24:18
Configure Command	'./configure' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/cgi' '--enable-ftp' '--enable-mbstring' '--enable-mysqlnd' '--with-curl' '--with-libedit' '--with-enable-fpm' '--with-fpm-user=www-data' '--with-fpm-group=www-data'
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	/usr/local/etc/php/php.ini
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d

Kita juga bisa membuat virtual domain di local, misalnya coba.test merujuk ke <http://localhost/test>, caranya duplikasi file konfigurasi nginx default.conf dengan nama coba.conf, lalu modifikasi kodennya pada bagian berikut.

```
1 listen 80;
2 listen [::]:80;
3
4 server_name coba.test;
5 # root /var/www/public;
6 root /var/www/test;
```

```
coba.conf *
1 server {
2
3     # listen 80 default_server;
4     listen 80;
5     # listen [::]:80 default_server ipv6only=on;
6     listen [::]:80;
7
8     # server_name localhost;
9     server_name coba.test;
10    # root /var/www;
11    root /var/www/test;
12    index index.php index.html index.htm;
--
```

Kemudian daftarkan domain coba.test pada file hosts.

```
1 | 127.0.0.1 coba.test
```

```
sites — root@c764ad5e8447: /var/www — vi /etc/hosts — 80x24
##  
# Host Database  
#  
# localhost is used to configure the loopback interface  
# when the system is booting. Do not change this entry.  
##  
127.0.0.1      localhost  
255.255.255.255 broadcasthost  
::1            localhost  
127.0.0.1      coba.test
```

Catatan: pada MacOS dan Linux, file host terletak pada /etc/hosts, sedangkan pada Windows biasanya terletak pada C:\Windows\System32\drivers\etc\hosts. File ini harus diedit menggunakan user admin, misal jika menggunakan MacOS/Debian gunakan perintah sudo vi /etc/hosts, sedangkan jika menggunakan Windows maka bisa menggunakan perintah notepad C:\Windows\System32\drivers\etc\hosts

Setelah itu restart container nginx, docker-compose restart nginx, kemudian akses http://coba.test pada browser.

PHP Version 7.1.4	
System	Linux 81e62fa23007 4.9.87-linuxkit-aufs #1 SMP Wed Mar 14 15:12:16 UTC 2018 x86_64
Build Date	Apr 18 2017 19:24:18
Configure Command	'./configure' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/cgi' '--enable-ftp' '--enable-mbstring' '--enable-mysqli' '--with-curl' '--with-libedit' '--with-enable-fpm' '--with-fpm-user=www-data' '--with-fpm-group=www-data'
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	/usr/local/etc/php/php.ini

XAMPP



XAMPP

XAMPP yang beralamat di <https://www.apachefriends.org> merupakan paket instalasi yang terdiri dari Apache, MariaDB, PHP, dan PHPMyAdmin. Tersedia untuk OS Mac, Linux dan Windows.

[Download](#)

[Click here for other versions](#)

[XAMPP for Windows](#)

7.2.7 (PHP 7.2.7)

[XAMPP for Linux](#)

7.2.7 (PHP 7.2.7)

[XAMPP for OS X](#)

XAMPP-VM (PHP 7.2.7)

Untuk instalasinya cukup mudah karena XAMPP menyediakan installer yang bisa kita unduh pada tautan berikut: <https://www.apachefriends.org/download.html>.

Pilih versi terbaru yang sesuai dengan requirement Laravel 6.0, yaitu PHP minimal versi 7.2

XAMPP mempunyai control panel berbasis user interface untuk menjalankan Apache, MariaDB dan PHP. Oleh karenanya, setelah instalasi XAMPP berhasil, pastikan service Apache, mariaDB dan PHP, kamu aktifkan pada control panel tersebut.

Lokasi `webroot` pada XAMPP berbeda-beda tergantung OS dan lokasi instalasinya, pada Windows biasanya terletak di `C:\xampp\htdocs`, pada Linux Ubuntu terletak pada `/opt/lamp/htdocs` sedangkan pada MacOS terdapat `/Applications/XAMPP/htdocs` atau `/Applications/XAMPP/xamppfiles/htdocs`.

Web dapat diakses melalui URL `http://localhost`, sedangkan PHPMyAdmin dapat diakses melalui `http://localhost/phpmyadmin`, dengan username `root` dan tanpa password.

Untuk menguji coba XAMPP sebagaimana jika kita menggunakan Docker, maka kita bisa buat projek baru dengan membuat folder `test` pada direktori `webroot`, kemudian buat file `index.php` di dalam folder `test` tersebut yang berisi kode berikut.

```
1 | <?php phpinfo() ?>
```

Kemudian melalui web browser kita bisa mengakses alamat `http://localhost/test`, jika berhasil maka browser akan menampilkan informasi terkait konfigurasi PHP.

Kita juga bisa membuat virtual host, di mana karena XAMPP menggunakan web server Apache maka konfigurasinya pun ada di Apache. Lokasi file konfigurasinya bisa sangat bervariasi tergantung dari versi XAMPP dan sistem operasi yang kamu gunakan.

Buka file `httpd.conf`, pada Windows biasanya terletak di `C:\xampp\apache\conf\httpd.conf` atau jika di MacOS dapat dijumpai pada direktori `/Applications/XAMPP/xamppfiles/etc/httpd.conf` atau pada Ubuntu `/opt/lampp/etc/httpd.conf`, cari kata "Virtual Host" dan pastikan konfigurasi virtual host tidak di-comment.

- Pada Windows

```
1 | # Virtual hosts
2 | Include C:\xampp\apache\conf\extra\httpd-vhosts.conf
```

- Pada MacOS

```
1 | # Virtual hosts
2 | Include /Applications/XAMPP/etc/extra/httpd-vhosts.conf
```

- Pada Ubuntu

```
1 | # Virtual hosts
2 | Include etc/extra/httpd-vhosts.conf
```

Kemudian tambahkan konfigurasi berikut pada file `httpd-vhosts.conf`.

```
1 | <VirtualHost *:80>
2 |   DocumentRoot C:\xampp\htdocs\test
3 |   ServerName coba.test
4 |   <Directory "C:\xampp\htdocs\test">
5 |     Options Indexes FollowSymLinks
6 |     AllowOverride All
7 |     Require all granted
8 |   </Directory>
9 | </VirtualHost>
```

Catatan: sesuaikan lokasi directory `test`-nya.

Jangan lupa daftarkan virtual domain tersebut pada file `hosts` (pada bagian sebelumnya telah dibahas mengenai file ini).

```
1 | 127.0.0.1 coba.test
```

Restart Apache (XAMPP) melalui control panel,

Modules	Service	Module	PID(s)	Port(s)	Actions
		Apache	5764 6352	80, 443	Stop Admin Config Logs
		MySQL	4916	3306	Stop Admin Config Logs
		FileZilla			Start Admin Config Logs

Kemudian pada browser akses alamat <http://coba.test>.

Homestead

Homestead merupakan prekonfigurasi virtual machine Vagrant (<https://www.vagrantup.com>) untuk pengembangan aplikasi berbasis Laravel. Tools ini didukung secara resmi oleh Laravel. Untuk instalasinya silakan ikuti tautan resminya pada tautan berikut: <https://laravel.com/docs/6.0/homestead>

Composer

Composer merupakan PHP dependency manager yang bertugas mencari sumber pustaka yang ditentukan pada repository server (Packagist) serta dependensinya, melakukan download dan instalasi pustaka tersebut, serta mencatat pustaka yang telah diinstalasi. Secara umum fungsinya sama dengan NPM namun hanya untuk PHP.



Instalasi Composer

Jika kita menggunakan laradock, maka Composer sudah otomatis terinstal dan dapat langsung kita gunakan. Sebenarnya konfigurasinya dapat kita lihat pada file .env

```
WORKSPACE_COMPOSER_GLOBAL_INSTALL=true
```

```
laradock — root@6808ec445228: /var/www — docker • docker-compose.exe...
[Hafids-MacBook-Pro:laradock hafidmukhlasin$ docker-compose exec workspace bash]
[root@6808ec445228:/var/www# composer -v
Do not run Composer as root/super user! See https://getcomposer.org/root for details
[...]
```

Composer version 1.6.3 2018-01-31 16:28:17

Sedangkan jika Anda menggunakan XAMPP maka untuk menginstalasinya pada OS Mac atau Linux, Anda bisa merujuk ke dokumentasi resminya <https://getcomposer.org/download/>

Pada terminal, jalankan perintah berikut.

```
1 | php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
2 | php -r "if (hash_file('SHA384', 'composer-setup.php') ===
3 | '544e09ee996cdf60ece3804abc52599c22b1f40f4323403c44d44fdfdd586475ca9813a85
4 | 8088ffbc1f233e9b180f061') { echo 'Installer verified'; } else { echo
'Installer corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"
```

atau bisa juga menggunakan CURL

```
1 | curl -sS https://getcomposer.org/installer | php
```

Intinya perintah diatas akan mengunduh file composer.phar dari alamat <https://getcomposer.org/installer>

Selanjutnya kita bisa gunakan composer pada current direktori dengan perintah

`php composer.phar`

Nah, supaya perintah ini dapat dijalankan pada direktori manapun maka kita perlu mendaftarkannya pada environment variabel PATH (silakan googling jika belum tau tentang ini).

Pada contoh ini, file composer.phar dipindahkan ke lokasi /usr/local/bin/composer

`mv composer.phar /usr/local/bin/composer`

Kemudian tutup dan buka lagi terminal (restart), maka selanjutnya kita bisa menggunakan composer dengan cukup menjalankan perintah `composer`

Untuk memastikan bahwa composer sudah terinstal dengan benar, coba jalankan perintah berikut.

`composer -v`

Catatan: instalasi pada OS Windows lebih mudah lagi menggunakan installer yang bisa diunduh pada tautan berikut <https://getcomposer.org/Composer-Setup.exe>

Postman

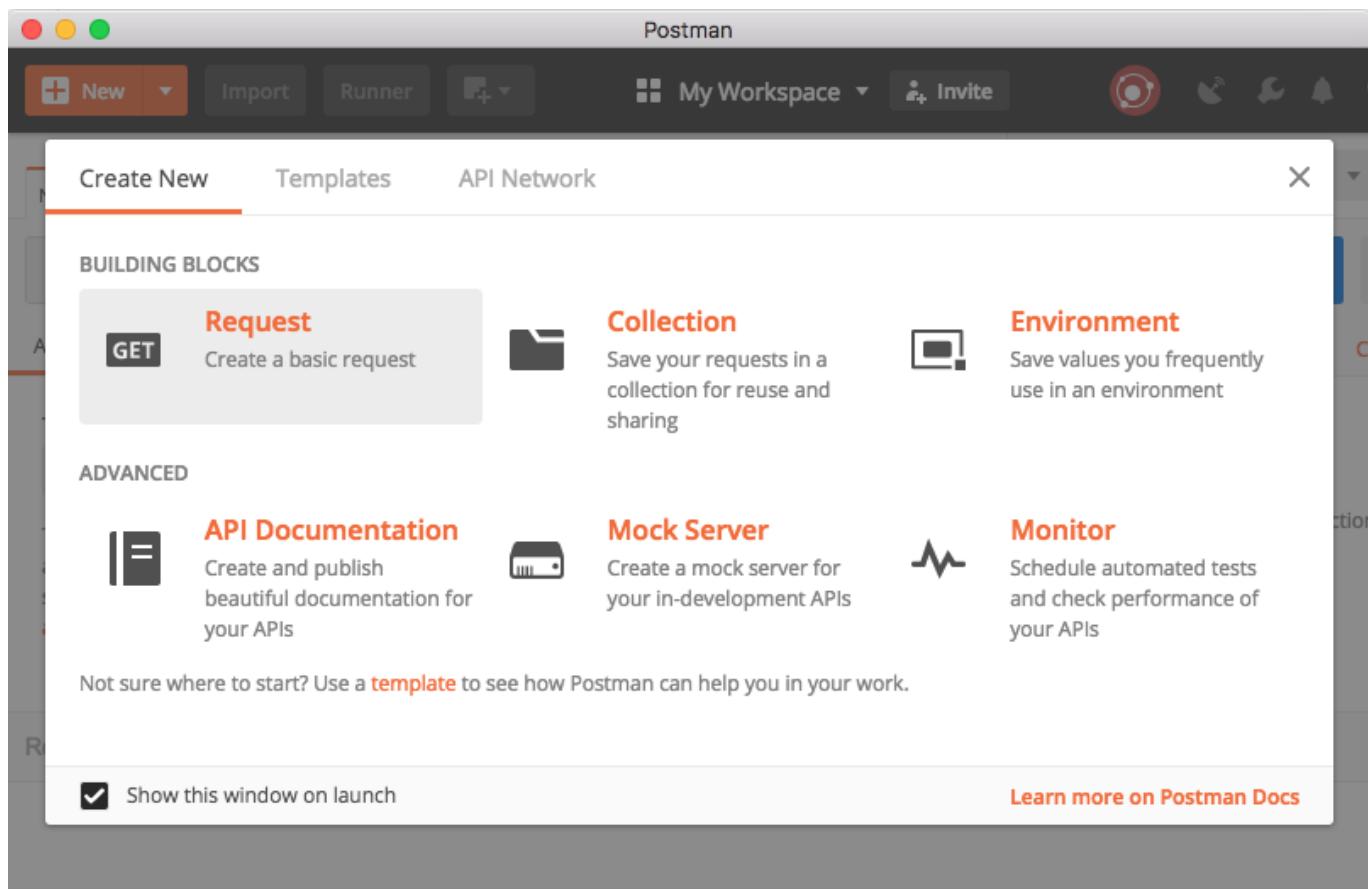
Postman adalah tools berbasis user interface yang digunakan untuk menguji coba suatu web service. Sehingga untuk sekedar mencoba apakah web service yang kita buat berjalan dengan baik atau tidak maka kita cukup menggunakan tools ini tanpa perlu membuat aplikasi clientnya dulu.



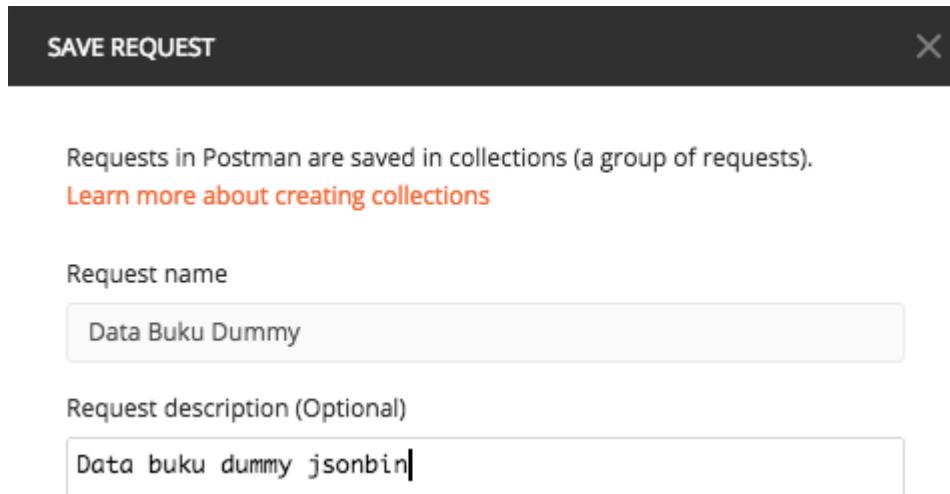
Postman tersedia untuk semua platform OS: Mac, Windows dan Linux. Untuk instalasinya silakan unduh file installernya pada tautan berikut: <https://www.getpostman.com/apps>

Penggunaan

Setelah menginstalasi Postman, jalankan Postman lalu pada halaman login, kamu bisa skip saja melalui menu bagian bawah. Maka akan muncul popup Create New.

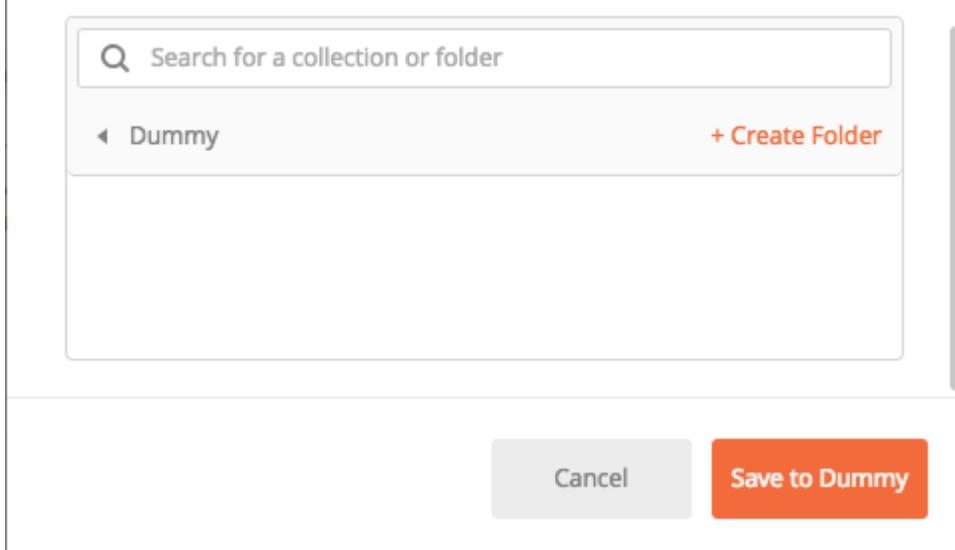


Pilih Request dan masukkan deskripsi dari URL web service yang ingin kita daftarkan. Tuliskan nama dan deskripsinya.



Lalu scroll ke bawah, pilih collection untuk mengelompokkan URL web service tersebut. Pada contoh ini penulis menambahkan collection **Dummy** melalui + Create Folder

Select a collection or folder to save to:



Kemudian Save to Dummy. Maka kita akan dibawa ke halaman utama, pilih HTTP method (defaultnya GET), masukkan URL web service yang ingin kita akses, misalnya URL yang sebelumnya kita gunakan <http://api.jsonbin.io/b/5b42acd44d5ea95c8ba22392>. Lalu klik tombol Send. Maka respon dari web service akan ditampilkan pada bagian bawah

```

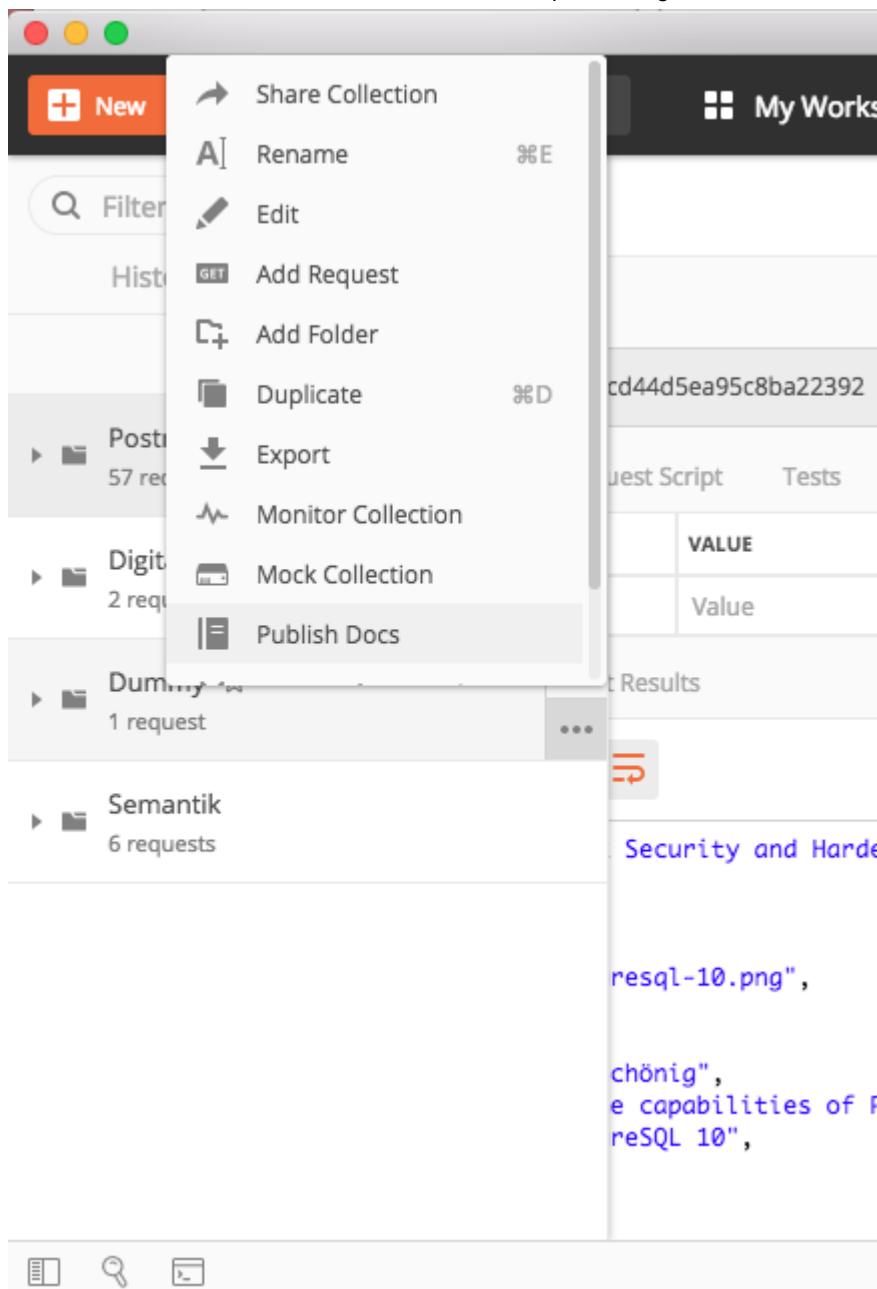
1 [ { 
2   "image": "c++-high-performance.png",
3   "price": 100000,
4   "publish_year": 2018,
5   "authors": "Viktor Sehr, Björn Andrist",
6   "description": "Write code that scales across CPU registers, multi-core, and machine clusters",
7   "title": "C++ High Performance",
8   "id": 99
9 },
10 ],
11 [
12   {
13     "image": "mastering-linux-security-and-hardening.png",
14     "price": 125000,
15   }
16 ]

```

Generate Dokumentasi

Postman memiliki fitur ciamik untuk menggenerate dokumentasi dari web service kita.

Pada panel kiri menu collection, klik tanda . . . lalu pada popup yang muncul pilih Publish Docs



Pilih Publish Collection.

The screenshot shows the 'Publish Collection' dialog box from the Postman application. At the top, it says 'DUMMY' and 'PUBLISH COLLECTION'. Below that, the title 'Publish Collection' is displayed. A section labeled 'Dummy' follows. Under 'Choose an environment', there is a dropdown menu set to 'No Environment'. To its right, a note states: 'The shared environment will be made available with the collection's public documentation'. Next, under 'Select a custom domain', another dropdown menu is set to 'No custom domain'. To its right, a note states: 'The published collection will be made available at the selected domain'. Below these sections, there is a checkbox labeled 'Allow other Postman users to discover this collection through the API network'. Underneath this, a red link says 'Show Custom Styling Options ▾'. At the bottom of the dialog, there are two buttons: 'Cancel' and a large orange 'Publish Collection' button. To the right of the 'Publish Collection' button, a note says: 'Completing this step make your collection available at a public URL.'

Maka kita akan diarahkan ke halaman yang menginfokan bahwa dokumentasi dari web service kita telah digenerate.

The screenshot shows the Postman interface with a published collection titled 'DUMMY'. The collection has been successfully published, indicated by a green checkmark icon. The URL for the published collection is displayed as <https://documenter.getpostman.com/view/255653/RWTfzMtG>. There are buttons to edit or unpublish the collection.

DUMMY ➔ EDIT PUBLISHED COLLECTION

Your collection is published!

Dummy, No Environment

Published URL

<https://documenter.getpostman.com/view/255653/RWTfzMtG>

[Edit published collection](#)

[Unpublish this collection](#)

Lalu akses URL yang dihasilkan.

The screenshot shows a browser window displaying the published collection 'Dummy'. The URL is <https://documenter.getpostman.com/view/255653/RWTfzMtG>. The page content includes an introduction and a 'GET Data Buku Dummy' endpoint. The endpoint details include the URL <http://api.jsonbin.io/b/5b42acd44d5ea95c8ba22392> and the response 'Data buku dummy jsonbin'. On the right side, there is an 'Example Request' section with a curl command:

```
curl --request GET \
--url http://api.jsonbin.io/b/5b42acd44d5ea95c8ba22392
```

Fitur lain yang dimiliki oleh Postman adalah automated testing untuk web service yang telah kita daftarkan.

Laravel

Laravel merupakan sebuah framework PHP yang sangat populer digunakan untuk mengembangkan aplikasi berbasis web.

Bagian ini hanya akan membahas sekilas tentang Laravel secara umum, karena fokus pada pembahasan tentang web service. Adapun pembahasan secara lebih detail tentang Laravel bisa melalui buku Laravel yang kami tulis dan terbitkan bersamaan dengan buku ini, atau melalui sumber lain.

Catatan: pada tutorial ini kita menggunakan Laravel versi 6.0.

Mengenal Laravel

Framework ini cukup mudah digunakan, serta memiliki dukungan yang baik terhadap berbagai teknologi terkini seperti Javascript. Laravel diinisiasi dan dimaintain oleh Taylor Otwell.

Hubungan antara Laravel dengan Vue sendiri cukup baik, di mana projek Vue sejak awal didukung penuh oleh Taylor Otwell, bahkan laravel telah menyediakan prekonfigurasi jika kita ingin menggunakan Vue sebagai framework frontend.

Instalasi

Sebelum menginstalasi laravel, pastikan kita telah menyiapkan environmentnya yaitu web server dan PHP, serta tools untuk instalasi pustaka berbasis PHP yaitu Composer.

Pastikan juga PHP dan extension-nya memenuhi spesifikasi untuk dapat menginstalasi Laravel versi 6.0 sebagai berikut.

- PHP >= 7.2.0
- BCMath PHP Extension
- OpenSSL PHP Extension
- PDO PHP Extension
- Mbstring PHP Extension
- Tokenizer PHP Extension
- XML PHP Extension
- Ctype PHP Extension
- JSON PHP Extension

Cara termudah menginstalasi laravel adalah dengan menggunakan tools Composer melalui perintah `create-project`. Pada terminal masuk ke direktori `webroot`

Ingat: Jika menggunakan XAMPP maka pada terminal (powershell) masuk ke direktori `htdocs`, sedangkan jika menggunakan laradock berarti kamu harus masuk ke workspace melalui perintah `docker-compose exec workspace bash`

```
● ● ● laradock — root@8a48a85542c9: /var/www — docker • docker-compose exe...
[Hafids-MacBook-Pro:Dev hafidmukhlasin$ cd laravel-projects/laradock/
[Hafids-MacBook-Pro:laradock hafidmukhlasin$ docker-compose exec workspace bash
root@8a48a85542c9:/var/www#
```

Lalu, jalankan perintah berikut untuk menginstalasi laravel.

```
composer create-project --prefer-dist laravel/laravel=6.0 larashop-api
```

Catatan: `larashop-api` adalah nama folder lokasi projek ini akan disimpan.

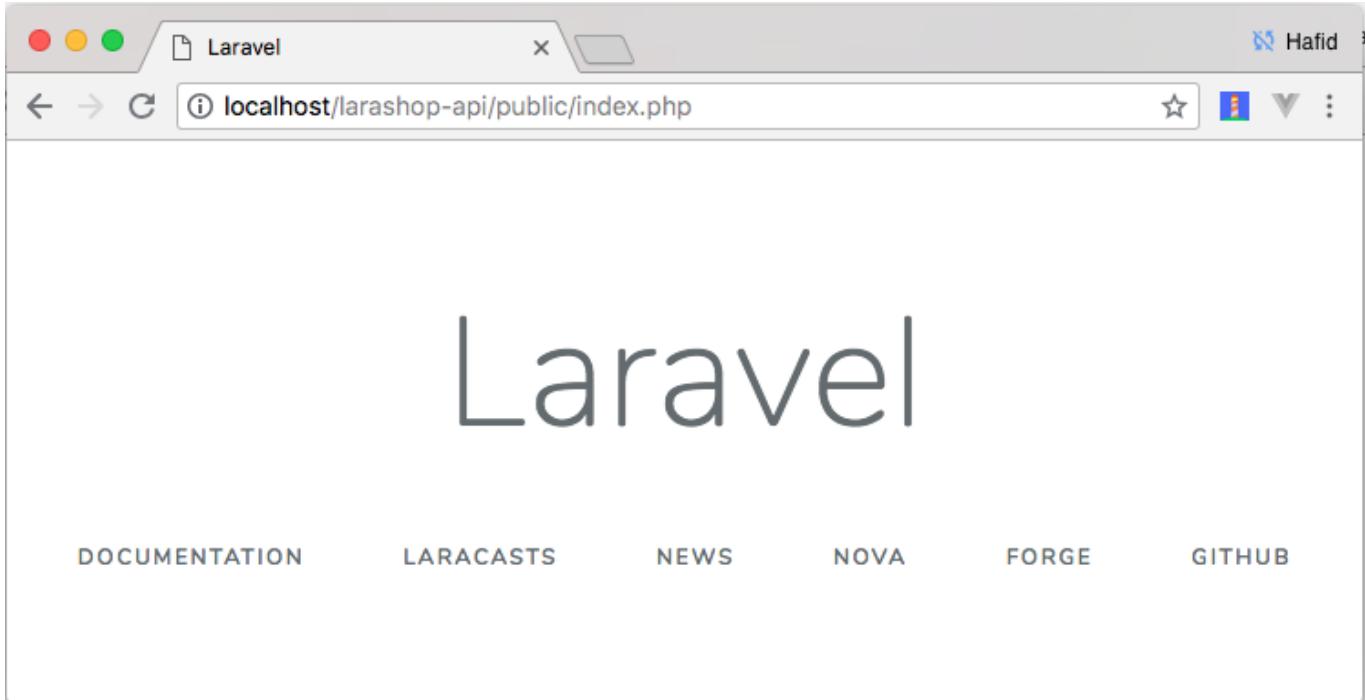
Perintah ini akan mengunduh file-file code laravel dari repository kemudian diinstalasi ke sistem kita. Adapun waktu yang dibutuhkan untuk instalasi ini sangat bergantung pada koneksi internet yang kita miliki, namun kita bisa menambahkan parameter `-vvv` untuk melihat detail dari setiap progres yang sedang berlangsung.

```

lardock — root@28de49479092: /var/www — docker + docker-compose exec...
[Hafids-MacBook-Pro:laradock hafidmukhlasin$ docker-compose exec workspace bash      ]
[root@28de49479092:/var/www# composer create-project --prefer-dist laravel/laravel ]
larashop-api
Do not run Composer as root/super user! See https://getcomposer.org/root for details
ls
Installing laravel/laravel (v5.7.0)
  - Installing laravel/laravel (v5.7.0): Downloading (100%)
Created project in larashop-api
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 71 installs, 0 updates, 0 removals
  - Installing vlucas/phpdotenv (v2.5.1): Downloading (100%)
  - Installing symfony/css-selector (v4.1.4): Downloading (100%)
  - Installing tijsverkoyen/css-to-inline-styles (2.2.1): Downloading (100%)
  - Installing symfony/polyfill-php72 (v1.9.0): Downloading (100%)
  - Installing symfony/polyfill-mbstring (v1.9.0): Downloading (100%)
  - Installing symfony/var-dumper (v4.1.4): Downloading (100%)
  - Installing symfony/routing (v4.1.4): Downloading (100%)
  - Installing symfony/process (v4.1.4): Downloading (100%)
  - Installing symfony/polyfill-ctype (v1.9.0): Downloading (100%)
tldr;
sebastian/global-state suggests installing ext-uopz (*)
PHPUnit/PHP-Coverage suggests installing ext-xdebug (^2.6.0)
PHPUnit/PHPUnit suggests installing PHPUnit/PHPUnit-Invoker (^2.0)
PHPUnit/PHPUnit suggests installing ext-soap (*)
PHPUnit/PHPUnit suggests installing ext-xdebug (*)
Writing lock file
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover
Discovered Package: beyondcode/laravel-dump-server
Discovered Package: fideloper/proxy
Discovered Package: laravel/tinker
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Package manifest generated successfully.
> @php artisan key:generate
Application key [base64:K9MFukfG8flpHEtwWRdzzkuGG/P5nThKDqXRNFitCI4=] set successfully.
root@28de49479092:/var/www#

```

Setelah berhasil kita instalasi, maka kita bisa mengaksesnya pada browser melalui alamat <http://localhost/larashop-api/public/index.php>



Catatan: laravel menggunakan folder public untuk file-file yang boleh diakses user melalui web browser. Adapun file index.php dalam folder tersebut merupakan mount point (file yang pertama kali diakses) dari aplikasi Laravel kita.

Konfigurasi

Variabel Konfigurasi

Laravel menggunakan file `.env` untuk menyimpan variabel konfigurasinya, seperti nama aplikasi, key aplikasi, URL aplikasi, dan koneksi database.

```

1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:K9MFukfG8f1pHETwWRdzzkuGG/P5nThKDqXRNFiTCI4=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 DB_CONNECTION=mysql
8 DB_HOST=127.0.0.1
9 DB_PORT=3306
10 DB_DATABASE=homestead
11 DB_USERNAME=homestead
12 DB_PASSWORD=secret

```

Silakan disesuaikan dengan environment sistem yang sudah kamu persiapkan sebelumnya.

`APP_KEY` merupakan key yang digunakan Laravel untuk mengenkripsi data yang digunakan pada aplikasi, sebagai contoh session data. Untuk keamanan, jalankan perintah `php artisan key:generate` untuk menggenerate ulang `APP_KEY` ketika aplikasi hendak dideploy.

Adapun untuk konfigurasi lebih lanjut, kita bisa tuliskan pada file-file dalam folder config.

Virtual Domain & Pretty URL

Buatlah virtual domain `larashop-api.test` sebagaimana langkah yang telah dijelaskan pada bagian sebelumnya.

Jika kita menggunakan laradock (web server Nginx) maka kita bisa melihat contoh konfigurasinya pada lokasi laradock/nginx/sites/laravel.conf.example

Silakan copy file konfigurasi tersebut dengan nama misalnya larashop-api.conf

```
larashop-api.conf
1 server {
2
3     listen 80;
4     listen [::]:80;
5
6     server_name larashop-api.test;
7     root /var/www/larashop-api/public;
8     index index.php index.html index.htm;
9
10    location / {
11        try_files $uri $uri/ /index.php$is_args$args;
12    }
13
14 }
```

```
server {
2
3     listen 80;
4     listen [::]:80;
5
6     server_name larashop-api.test;
7     root /var/www/larashop-api/public;
8     index index.php index.html index.htm;
9
10    location / {
11        try_files $uri $uri/ /index.php$is_args$args;
12    }
13
14 ...}
```

Pada kode di atas, bagian location.

```
try_files $uri $uri/ /index.php$is_args$args;
```

atau boleh juga ditulis seperti berikut

```
try_files $uri $uri/ /index.php?$query_string;
```

digunakan untuk mendukung pretty URL (URL yang SEO Friendly atau ramah mesin pencari) pada aplikasi kita. Sehingga untuk mengakses aplikasi kita, maka kita tidak perlu menyertakan index.php pada URL. Cukup dengan URL berikut.

<http://larashop-api.test>

Jika kita menggunakan XAMPP atau web server Apache maka pada file httpd-vhosts.conf, tambahkan konfigurasi yang kurang lebih sebagai berikut.

```
<VirtualHost *:80>
1 DocumentRoot C:\xampp\htdocs\larashop-api\public
2
```

```

3  ServerName larashop-api.test
4  <Directory "C:\xampp\htdocs\larashop-api\public">
5      Options Indexes FollowSymLinks
6      AllowOverride All
7      Require all granted
8  </Directory>
9  </VirtualHost>
```

Supaya aplikasi kita mendukung pretty URL (URL yang SEO friendly) maka jika menggunakan XAMPP atau web server Apache, pastikan kita mengaktifkan module `mod_rewrite` pada PHP sehingga file `.htaccess` yang sudah disertakan oleh Laravel dalam folder `public` akan bekerja.

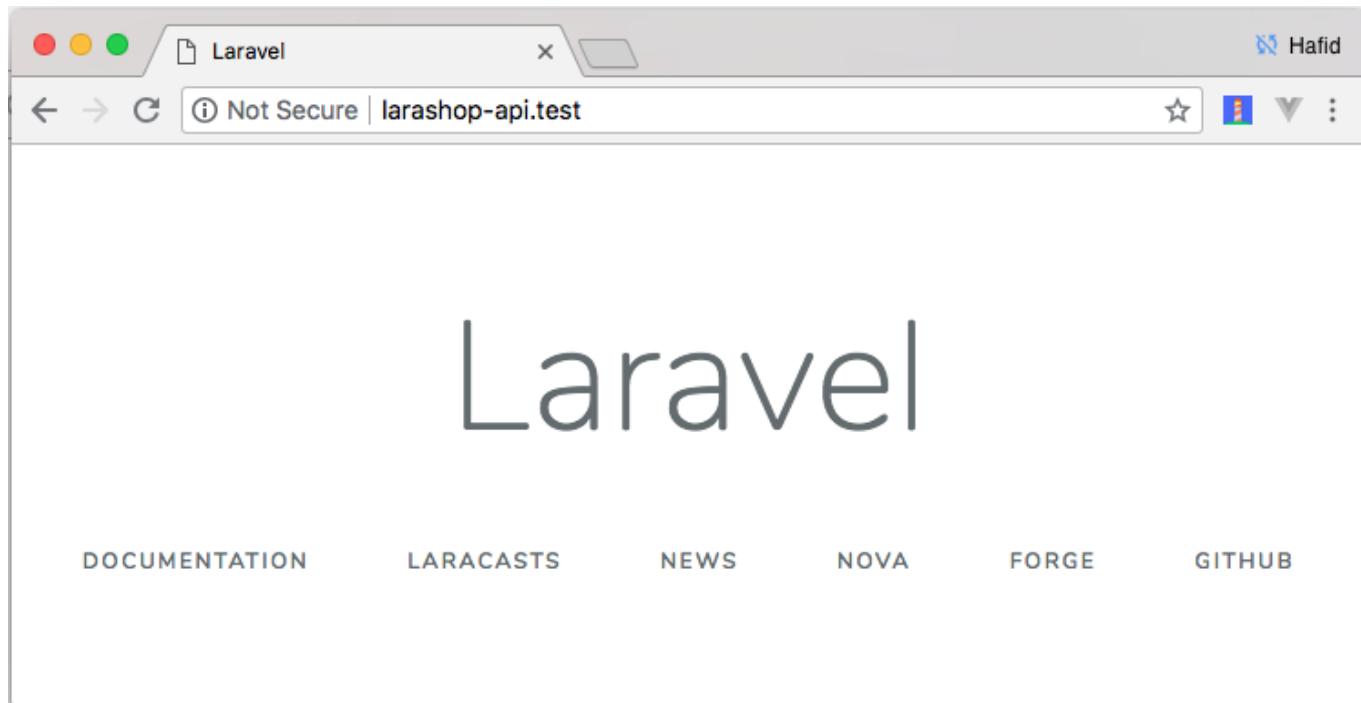
Kemudian, daftarkan virtual domain `larashop-api.test` pada file hosts.

```

## 
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting. Do not change this entry.
##
127.0.0.1      localhost
255.255.255.255 broadcasthost
::1            localhost
127.0.0.1      coba.test
127.0.0.1      larashop-api.test
~
```

Yang terakhir jangan lupa restart web server supaya konfigurasi yang kita lakukan dapat diaplikasikan.

Pada laradock jalankan perintah `docker-compose restart nginx` atau pada XAMPP control panel stop dan start lagi service Apache. Lalu jika semua sudah kita lakukan maka pada browser, silakan akses `http://larashop-api.test`



horee!

Struktur Direktori Aplikasi

Struktur direktori projek aplikasi yang digenerate oleh Laravel adalah sebagai berikut.

The screenshot shows a Mac OS X Finder window with the title bar "larashop-api". The left sidebar lists "Favorites" including iCloud Drive, Documents, AirDrop, All My Files, Applications, Desktop, Downloads, Movies, Music, Pictures, and a folder for "hafidmukhl...". The main pane displays a table of files and folders:

Name	Date Modified	Size	Kind
app	Today, 4:40 AM	--	Folder
artisan	Today, 3:02 AM	2 KB	TextEd...ument
bootstrap	Today, 3:02 AM	--	Folder
composer.json	Today, 3:02 AM	1 KB	JSON
composer.lock	Today, 3:13 AM	144 KB	Document
config	Today, 3:02 AM	--	Folder
database	Today, 4:42 AM	--	Folder
package.json	Today, 3:02 AM	1 KB	JSON
phpunit.xml	Today, 3:02 AM	1 KB	XML text
public	Today, 3:02 AM	--	Folder
readme.md	Today, 3:02 AM	4 KB	Visual...cument
resources	Today, 4:43 AM	--	Folder
routes	Today, 3:02 AM	--	Folder
server.php	Today, 3:02 AM	563 bytes	PHP script
storage	Today, 4:43 AM	--	Folder
tests	Today, 3:02 AM	--	Folder
vendor	Today, 3:13 AM	--	Folder
webpack.mix.js	Today, 3:02 AM	549 bytes	JavaScript

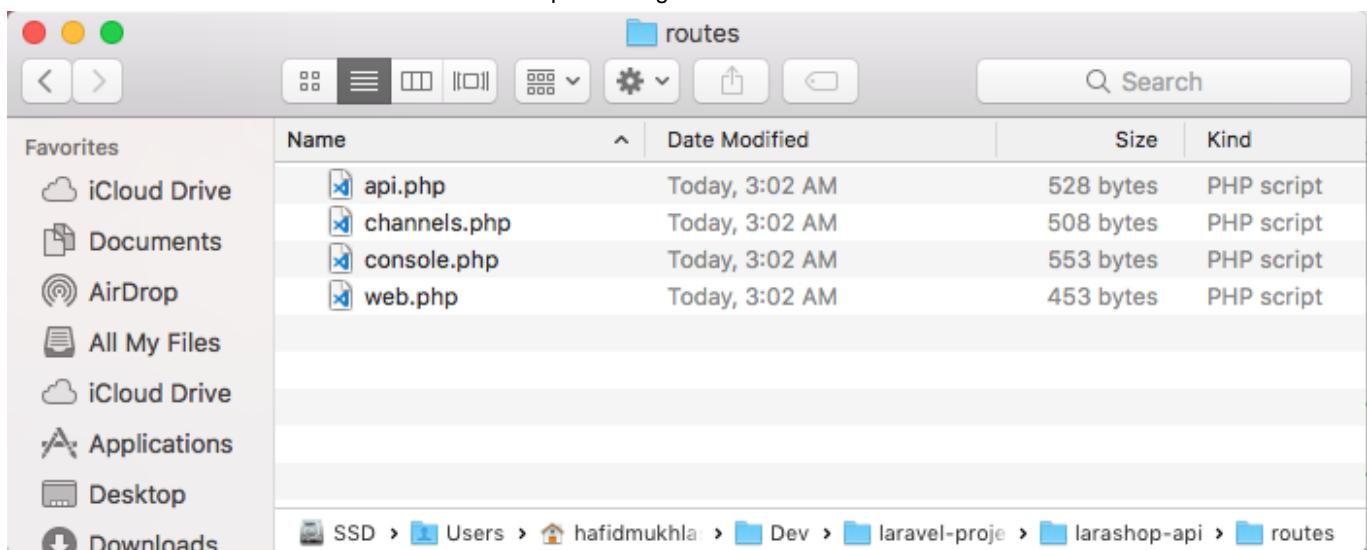
The path at the bottom of the window is: SSD > Users > hafidmukhlasin > Dev > laravel-projects > larashop-api.

Struktur tersebut dapat dijelaskan sebagai berikut.

- **app** -> folder ini berisi logic dari aplikasi kita seperti controller, middleware dan model.
- **bootstrap** -> folder ini berisi file-file yang diakses pertama kali setelah index dan config.
- **config** -> folder ini berisi konfigurasi aplikasi.
- **database** -> folder ini berisi file migration, dan seeding database.
- **public** -> folder ini berisi file-file yang boleh diakses oleh user melalui web browser.
- **resources** -> folder ini berisi asset yang digunakan pada aplikasi seperti css, js, image, dan view (blade template)
- **routes** -> folder ini berisi file-file yang digunakan untuk mengatur routing aplikasi.
- **storage** -> folder ini digunakan sebagai tempat penyimpanan cache, session, dan log aplikasi. Pastikan folder ini dapat ditulisi oleh PHP.
- **test** -> folder ini berisi kode-kode testing
- **vendor** -> folder ini berisi pustaka yang digunakan oleh aplikasi.

Routing

Routing merupakan bagian sentral dari aplikasi yaitu bagaimana aplikasi membaca URL yang diketikkan atau dikirimkan oleh user ketika mengakses aplikasi kita, memprosesnya serta meresponnya. Konfigurasi routing pada Laravel disimpan dalam folder **routes**.



Terdapat empat file yaitu:

- `api.php` -> konfigurasi routing khusus untuk web service
- `web.php` -> konfigurasi routing untuk aplikasi web biasa
- `channels.php` -> konfigurasi routing untuk broadcast message seperti notification dan chat.
- `console.php` -> konfigurasi routing untuk diakses pada console

Routing Web

Sebelum kita membahas lebih spesifik tentang routing web service yang konfigurasinya terdapat pada file `api.php`, kita akan coba preview sekilas tentang routing pada aplikasi web biasa. Mari kita buka file `web.php` maka kita akan dapati kode routing berikut.

```

1 Route::get('/', function () {
2     return view('welcome');
3 });

```

Kode routing di atas maksudnya adalah ketika URL yang direquest oleh user sama dengan / misal `http://larashop-api.test/` maka Laravel akan memanggil view atau tampilan `welcome` yang terdapat pada direktori `resources/views/welcome.blade.php`. Kode pada file `welcome.blade.php` merupakan kode HTML yang akan dicompile menggunakan template engine blade bawaan Laravel.

Merujuk hal tersebut maka kita bisa menambahkan kode routing berikut.

```

1 Route::get('hello', function () {
2     return view('hello');
3 });

```

Kemudian buat file `hello.blade.php` pada direktori `resources/views/`

```

1 <!doctype html>
2 <html>
3     <body>
4         <h1>Hello kakak! aku bisa mainan Laravel nih!</h1>
5     </body>
6 </html>

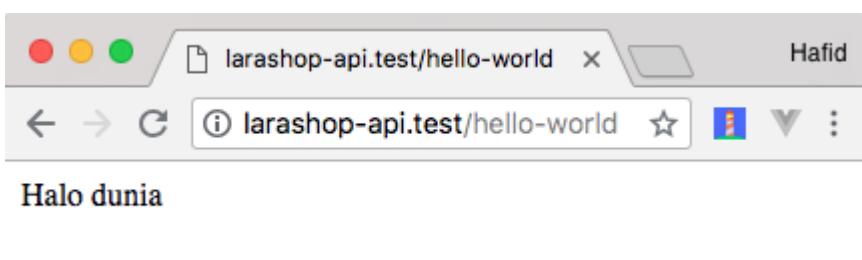
```

Lalu pada browser, kita bisa mengaksesnya menggunakan alamat URL: `http://larashop-api.test/hello`



Atau untuk sekedar menampilkan teks, kita juga bisa langsung me-return string tanpa view.

```
1 Route::get('hello-world', function () {
2     return 'Halo dunia';
3 });
```



Routing web ini terhubung dengan group middleware (mekanisme untuk filtering http request) yang menyediakan fitur seperti session dan proteksi CSRF.

Routing API (Web Service)

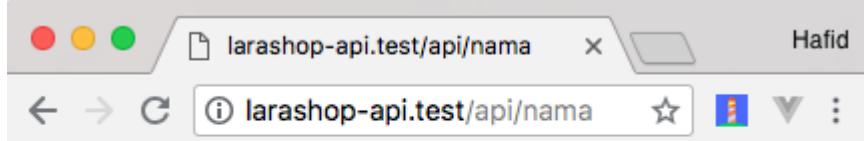
Perbedaan utama dengan routing web dengan routing api ini adalah sifatnya yang stateless (tanpa session). Sebagaimana yang telah dijelaskan sebelumnya bahwa kode routing untuk web service disimpan dalam file api.php, yang jika kita buka file tersebut maka terdapat kode berikut.

```
1 Route::middleware('auth:api')->get('/user', function (Request $request) {
2     return $request->user();
3 });
```

Nah sebelum kita memahami maksud dari kode di atas, mari kita membuat aturan routing baru pada file tersebut. Misalnya:

```
1 Route::get('nama', function () {
2     return 'Namaku, Larashop API';
3 });
```

Intinya ketika user mengakses URL nama maka akan tampil teks Namaku, Larashop API. Namun tidak seperti routing web, routing api ini menggunakan prefix api pada URL sehingga untuk mengakses routing nama di atas menggunakan alamat URL <http://larashop-api.test/api/nama>



Tentu seharusnya untuk mengujinya kita gunakan Postman.

HTTP Verbs Method

Routing pada laravel juga mendukung HTTP verbs method selain GET yaitu POST, PUT, PATCH, DELETE, OPTIONS. Cara deklarasinya juga sama.

```

1 Route::get($uri, $callback);
2 Route::post($uri, $callback);
3 Route::put($uri, $callback);
4 Route::patch($uri, $callback);
5 Route::delete($uri, $callback);
6 Route::options($uri, $callback);

```

Ketika suatu route kita deklarasikan menggunakan method POST maka kita tidak akan bisa mengaksesnya menggunakan method lain. Contoh:

```

1 Route::post('umur', function () {
2     return 17;
3 });

```

Ketika kita akses menggunakan browser yang notabene menggunakan GET maka akan menampilkan error.

The screenshot shows a browser window with the URL `larashop-api.test/api/umur`. The title bar says "Whoops! There was an error." The main content area displays the following error message:

```

Symfony \ Component
\ HttpKernel \
Exception \
MethodNotAllowedHttpException

No message

G ⌂ ⌂
```

Below this, there is a sidebar titled "Application frames (1)" with a link "All frames (27)". A single frame is expanded, showing the stack trace:

```

26 Symfony\Component\HttpKernel\Exception\MethodNotAllowedHttpException
.../vendor/laravel/framework/src/Illuminate/Routing/RouteCollection.php:255
```

The right side of the screen shows the source code of the `RouteCollection.php` file from Laravel's framework, specifically the `methodNotAllowed` method:

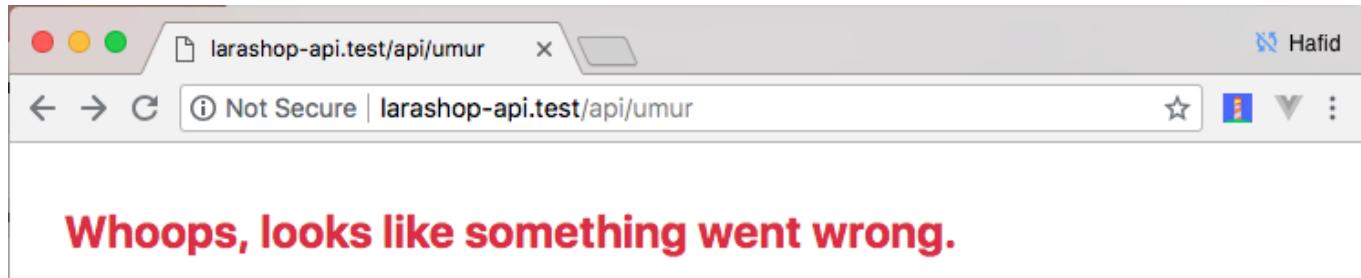
```

245.     /**
246.      * Throw a method not allowed HTTP exception.
247.      *
248.      * @param array $others
249.      * @return void
250.      *
251.      * @throws \Symfony\Component\HttpKernel\Exception\MethodNotAllowedHttpException
252.     */
253.     protected function methodNotAllowed(array $others)
254.     {
255.         throw new MethodNotFoundException($others);
256.     }
257. 
258.     /**
259.      * Get routes from the collection by method.
260.      *
261.      * @param string|null $method
262.      * @return array
263.      */
264. 
```

Tampilan error menyeramkan ini muncul karena setting `APP_DEBUG` pada file `.env` kita set bernilai true, namun jika nilainya false.

```
1 | APP_DEBUG=false
```

Maka akan muncul tampilan error yang lebih ramah sebagai berikut.



Untuk mengakses routing `umur` di atas kita bisa menggunakan Postman, dengan method POST.

The screenshot shows the Postman interface. The address bar shows `http://larashop-api.test`. The request type is set to `POST`, and the URL is `http://larashop-api.test/api/umur`. The "Params" tab is selected. At the bottom, there are buttons for "Pretty", "Raw", and "Preview".

Adakalanya kita ingin membuat routing yang bisa diakses oleh beberapa method sekaligus, maka kita bisa gunakan fungsi `match`.

```

1 Route::match(['get', 'post'], 'test', function () {
2     //
3 });
4

```

Maka routing `/test` sekarang dapat diakses menggunakan method GET dan dapat pula diakses dengan POST.

Contoh lain, jika kita ingin agar suatu routing bisa diakses dengan method apapun maka kita bisa gunakan fungsi `any`.

```

1 Route::any('foo', function () {
2     //
3 });

```

Routing Parameter

Kita juga dapat menyisipkan parameter pada routing, sehingga bisa membuat respon yang dinamis sesuai dengan parameter yang dikirimkan oleh user. Contoh:

```

1 Route::get('category/{id}', function ($id) {
2     $categories = [
3         1 => 'Programming',
4         2 => 'Desain Grafis',
5         3 => 'Jaringan Komputer',
6     ];
7     $id = (int) $id;
8     if($id==0) return 'Silakan pilih kategori';
9     else return 'Anda memilih kategori <b>' . $categories[$id] . '</b>';
10 });

```

Mari kita akses dengan menggunakan URL `http://larashop-api.test/api/category/0`

Maka akan muncul respon teks. Silakan pilih kategori

Nah ketika kita akses menggunakan URL `http://larashop-api.test/api/category/1` maka akan menampilkan respon teks Anda memilih kategori Programming

Namun jika kita mengakses URL `http://larashop-api.test/api/category` maka akan muncul error bahwa halaman tidak ditemukan.

Solusinya, kita bisa membuat parameter id pada routing tersebut optional. Caranya, tambahkan tanda tanya ? pada parameter dan berikan nilai default pada fungsi.

```

1 Route::get('category/{id?}', function ($id=null) {
2     // ...
3 });

```

Menariknya, parameter pada routing juga mendukung Regular Expression (regex), sehingga ketika URL yang diberikan tidak match dengan regex yang didefinisikan maka akan tertolak. Caranya dengan menambahkan fungsi `where`.

```

1 Route::get('book/{id}', function () {
2     return 'buku angka';

```

```
3 | }))->where('id', '[0-9]+');
```

Pada contoh di atas, routing akan match jika parameter id bernilai numerik atau angka.

GET ▾

<http://larashop-api.test/api/book/123>

Pretty Raw Preview

buku angka

```
1 | Route::get('book/{title}', function ($title) {
2 |     return 'buku abjad';
3 | })->where('title', '[A-Za-z]+');
```

Pada contoh di atas, routing akan match jika parameter id bernilai abjad.

GET ▾

<http://larashop-api.test/api/book/abc>

Pretty Raw Preview

buku abjad

Meski URL routingnya sama kita bisa membedakan berdasarkan parameter yang diberikan. keren kan?

Catatan: regular expression atau regex adalah standard umum yang tidak terbatas atau tergantung pada suatu bahasa pemrograman / framework tertentu, silakan googling mengenai hal itu karena buku ini tidak akan membahas spesifik tentang regex.

Routing Name

Kita juga bisa memberikan nama pada suatu routing sehingga memudahkan kita dalam menyebutnya.

```
1 | Route::get('book/{id}', function ($id) {
2 |     //
3 | })->name('book');
```

Sehingga untuk menggunakannya kita cukup memanggil namanya saja, misal

```
1 | // Generating URLs...
2 | $url = route('book', ['id' => 1]);
3 |
4 | // Generating Redirects...
5 | return redirect()->route('book', [
6 |     'id' => 1
7 | ]);
```

Routing Group

Kita bisa mengelompokkan suatu aturan routing menggunakan group berdasarkan beberapa hal misalnya Prefix, Sub-Domain.

Route Prefixes

Route prefix ini digunakan untuk mengelompokkan route berdasarkan prefix sehingga pada URL akan ada tambahan prefix di depan route. Sedikit telah kita singgung sebelumnya bahwa routing api juga menggunakan fitur prefix ini sehingga secara default untuk mengakses routing api kita harus mengawalinya dengan prefix `api`.

Contoh kita ingin mengelompokkan route web service berdasarkan versinya.

```

1 Route::prefix('v1')->group(function () {
2     Route::get('books', function () {
3         // Match dengan "/v1/books"
4     });
5
6     Route::get('categories', function () {
7         // Match dengan "/v1/categories"
8     });
9 });

```

Selanjutnya, untuk mengakses resource tersebut menggunakan URL:

- `http://larashop-api.test/api/v1/books`
- `http://larashop-api.test/api/v1/categories`

Lalu di mana pengaturan prefix api yang secara default mengubah aturan routing?

Penambahan prefix ini didefinisikan pada class `App\Providers\RouteServiceProvider`, tepatnya pada fungsi `mapApiRoutes()`

```

1 protected function mapApiRoutes()
2 {
3     Route::prefix('api')
4         ->middleware('api')
5         ->namespace($this->namespace)
6         ->group(base_path('routes/api.php'));
7 }

```

Jika method `prefix('api')` di atas kita hapus, menjadi sebagai berikut.

```

1 protected function mapApiRoutes()
2 {
3     Route::middleware('api')
4         ->namespace($this->namespace)
5         ->group(base_path('routes/api.php'));
6 }

```

Maka URL web service kita tanpa prefix `api` lagi.

Misalnya sebelumnya `http://larashop-api.test/api/book/abc` menjadi `http://larashop-api.test/book/abc`

GET ▾ http://larashop-api.test/book/abc

Pretty Raw Preview

buku abjad

Route Sub-Domain

Route sub-domain memungkinkan kita menangani routing sub domian, di mana dengan menggunakan fitur ini maka sub domain bisa diperlakukan seperti parameter biasa.

Misalnya kita ingin membuat sub domain berdasarkan kategori.

```
1 Route::domain('{category}.larashop.id')->group(function () {
2     Route::get('book/{id}', function ($category, $id) {
3         //
4     });
5 });
```

Setelah kita banyak melakukan konfigurasi routing, kita bisa jalankan perintah berikut untuk melihat routing yang tersedia pada aplikasi kita.

```
1 php artisan route:list
```

Domain	Method	URI	Name	Action	Middleware
	GET HEAD	/		Closure	web
	GET HEAD	book/{id}		Closure	api
	GET HEAD	book/{title}		Closure	api
	GET HEAD	category/{id?}		Closure	api
	GET HEAD	hello		Closure	web
	GET HEAD	hello-world		Closure	web
	GET HEAD	nama		Closure	api
	GET POST HEAD	test		Closure	api
	POST	umur		Closure	api
	GET HEAD	user		Closure	api,auth:api
	GET HEAD	v1/books		Closure	api
	GET HEAD	v1/categories		Closure	api

Nah, karena kita memang hanya akan menggunakan routing api, maka kita bisa jadikan matikan routing webnya. Caranya masih pada class RouteServiceProvider, tepatnya pada fungsi map(). Comment saja mapWebRoutes()

```
1 public function map()
2 {
3     $this->mapApiRoutes();
4     // $this->mapWebRoutes(); // <= ini
5 }
```

Hasilnya.

Domain	Method	URI	Name	Action	Middleware
	GET HEAD	book/{id}		Closure	api
	GET HEAD	book/{title}		Closure	api
	GET HEAD	category/{id?}		Closure	api
	GET HEAD	nama		Closure	api
	GET POST HEAD	test		Closure	api
	POST	umur		Closure	api
	GET HEAD	user		Closure	api,auth:api
	GET HEAD	v1/books		Closure	api
	GET HEAD	v1/categories		Closure	api

Controller

Controller mempunyai hubungan erat dengan routing. Di mana kita bisa meletakkan logic atau bisnis proses pada controller sehingga definisi routing tidak tercampur dengan logic.

Secara default, class controller disimpan pada direktori app/Http/Controllers.

Sebagai contoh kita akan membuat controller BookController, caranya pada terminal jalankan perintah berikut.

```
1 | php artisan make:controller BookController
```

Maka akan digenerate file BookController.php pada direktori app/Http/Controllers

```
1 | <?php
2 |
3 | namespace App\Http\Controllers;
4 |
5 | use Illuminate\Http\Request;
6 |
7 | class BookController extends Controller
8 |
9 | {
10 | }
```

Lalu pada class BookController tersebut kita tambahkan fungsi

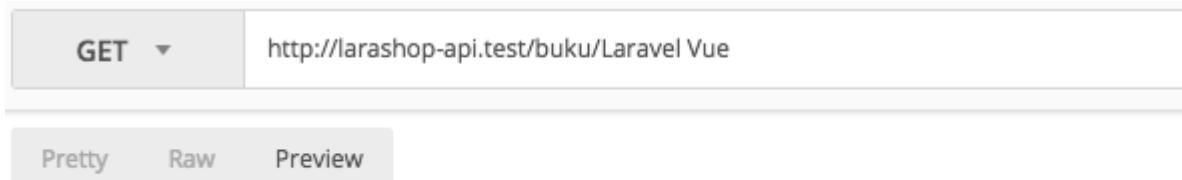
```
1 | public function cetak($judul)
2 |
3 | {
4 |     return $judul;
```

Catatan: merupakan *best practice* memberikan nama class controller dengan akhiran Controller. Apakah harus demikian? jawabannya tidak harus.

Kemudian pada konfigurasi api.php, kita tambahkan.

```
1 | Route::get('buku/{judul}', 'BookController@cetak');
```

Mari kita coba melalui postman dengan mengakses URL <http://larashop-api.test/buku/Laravel Vue>



Laravel Vue

Dengan cara ini maka kode routing pada file api.php akan lebih rapi.

Middleware

Middleware merupakan mekanisme di Laravel yang memungkinkan kita melakukan filtering atau pengecekan terhadap HTTP request yang masuk ke aplikasi kita. Sebagai contoh implementasinya untuk mengecek apakah request yang masuk berasal dari user yang sudah login atau belum, jika sudah login boleh lanjut ke action yang dimaksud, namun jika belum maka akan dilempar ke halaman login.

Kode Middleware dapat kita jumpai pada direktori app/Http/Middleware, yang didalamnya terdapat beberapa Middleware bawaan Laravel seperti otentikasi dan proteksi CSRF, namun kita juga bisa menambahkan sendiri.

Rate Limiting

Laravel mempunyai middleware yang bertugas membatasi akses ke suatu routing tertentu yaitu throttle. Middleware throttle menerima dua parameter untuk menentukan jumlah maksimal request yang dizinkan dalam setiap menitnya. Sebagai contoh routing buku maksimal hanya boleh diakses sebanyak 10 kali dalam setiap menitnya:

```
1 | Route::middleware('throttle:10,1')->group(function () {
2 |     Route::get('buku/{judul}', 'BookController@cetak');
3 | });
```

Ketika diakses lebih dari itu maka akan muncul error Too many request.

GET <http://larashop-api.test/buku/Laravel Vue> Params

Pretty Raw Preview

429

Sorry, you are making too many requests to our servers.

[GO HOME](#)

Secara default, `throttle` telah didefinisikan pada `App\Http\Kernel.php`

```

1 protected $middlewareGroups = [
2     'web' => [
3         ...
4     ],
5     'api' => [
6         'throttle:60,1',
7         'bindings',
8     ],
9 ];

```

Artinya secara default, routing apapun pada Laravel hanya bisa diakses oleh satu user maksimal satu kali per detiknya.

CORS

CORS adalah singkatan dari Cross-Origin Resource Sharing, pada bab sebelumnya kita telah menyinggung tentang CORS, di mana by default, Javascript tidak diizinkan melakukan HTTP request ke domain server berbeda. Misal: aplikasi Vue kita di domain `vueshop.id` kemudian mengakses web service Laravel di `api.larashop.id`. Nah untuk mengatasi hal ini, kita bisa menambahkan kode untuk terkait CORS pada middleware.

Pertama kita generate middleware, pada terminal gunakan perintah.

```
1 | php artisan make:middleware Cors
```

Catatan: jangan lupa perintah ini harus dieksekusi di workspace (jika menggunakan laradock) pada folder larashop-api

```
[Hafids-MacBook-Pro:laradock hafidmukhlasin$ docker-compose exec workspace bash
root@8a48a85542c9:/var/www#
[root@8a48a85542c9:/var/www/larashop-api# php artisan make:middleware Cors
Middleware created successfully.
root@8a48a85542c9:/var/www/larashop-api# ]]
```

Perintah di atas akan menggenerate template middleware Cors pada direktori app\Http\Middleware\Cors.php, update file tersebut menjadi sebagai berikut.

```
1 | <?php
2 | namespace App\Http\Middleware;
3 | use Closure;
4 | class Cors
5 | {
6 |     public function handle($request, Closure $next)
7 |     {
8 |         return $next($request)
9 |             ->header('Access-Control-Allow-Origin', '*')
10 |             ->header('Access-Control-Allow-Credentials', true)
11 |             ->header('Access-Control-Allow-Methods', 'GET, POST, PUT,
12 | DELETE, OPTIONS')
13 |             ->header('Access-Control-Allow-Headers', 'X-Requested-With,
14 | Content-Type, X-Token-Auth, Authorization')
15 |             ;
|     }
| }
```

Setelah itu kita bisa daftarkan pada file /app/Http/Kernel.php tepatnya di properti routeMiddleware

```
1 | protected $routeMiddleware = [
2 |     ...
3 |     'cors' => \App\Http\Middleware\Cors::class
4 | ];
```

Selanjutnya kita bisa gunakan pada routing.

```
1 | Route::middleware(['cors'])->group(function () {
2 |     Route::get('buku/{judul}', 'BookController@cetak');
3 | });
```

Sebagai contoh, kita coba buka kembali projek vue yang telah kita siapkan untuk studi kasus yaitu vueshop pada direktori vue-projects. Pada contoh ini kita akan gunakan axios untuk merequest <http://larashop-api.test/buku/PHP MySQL>

Kita akan gunakan axios untuk merequest resource web service tersebut pada view Home.vue tepatnya di vueshop/src/views/Home.vue. Tambahkan kode berikut.

```

1 <script>
2   export default {
3     created () {
4       window.axios.get('http://larashop-api.test/buku/PHP MySQL')
5         .then((response) => {
6           console.log(response)
7         })
8         .catch((error) => {
9           console.log(error)
10        })
11      }
12    }
13  </script>

```

Kode di atas akan menampilkan hasil request pada console log.

Lalu pada terminal, jalankan perintah `npm run serve`

```

vueshop — root@8a48a85542c9: /var/www/larashop-api — node -e npm TERM_PROGRAM=A
[Hafids-MacBook-Pro:Dev hafidmukhlasin$ cd vue-projects/
[Hafids-MacBook-Pro:vue-projects hafidmukhlasin$ cd vueshop/
[Hafids-MacBook-Pro:vueshop hafidmukhlasin$ npm run serve

> vueshop@0.1.0 serve /Users/hafidmukhlasin/Dev/vue-projects/vueshop
> vue-cli-service serve

[INFO] Starting development server...
98% after emitting CopyPlugin

[DONE] Compiled successfully in 6963ms

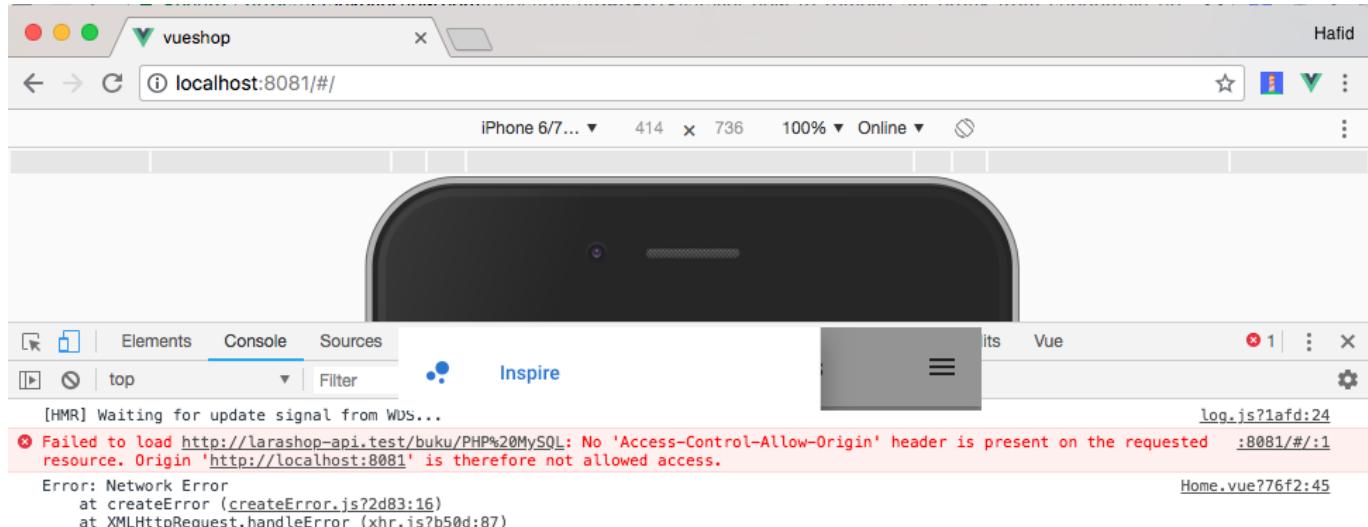
App running at:
- Local:  http://localhost:8081/
- Network: http://192.168.43.127:8081/

Note that the development build is not optimized.
To create a production build, run npm run build.

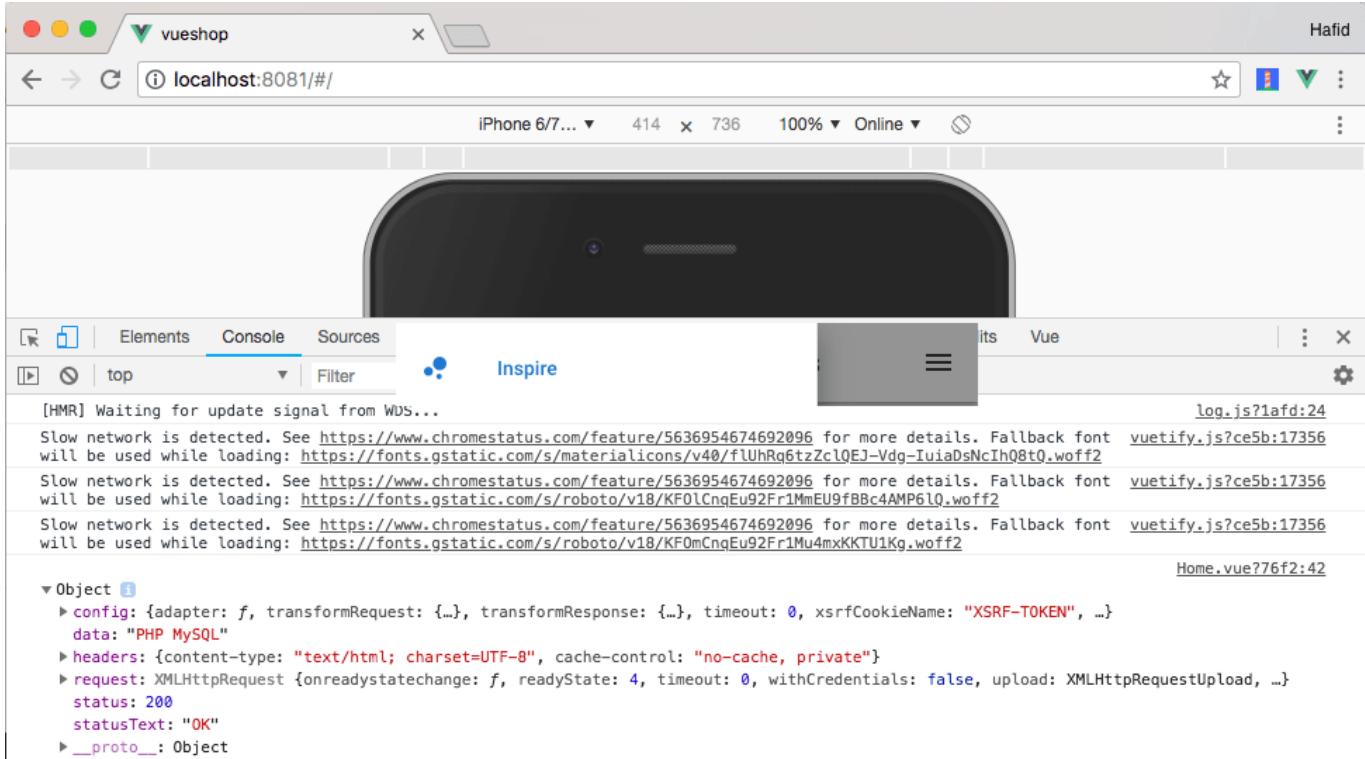
```

Kemudian kita bisa akses melalui web browser pada alamat `http://localhost:8081`

Jika middleware Cors tidak kita pasang maka akan muncul error pada console log browser.



Namun setelah kita pasang maka pada consol akan ditampilkan data Testing hasil dari request tersebut.



Cara ini sebenarnya sudah cukup untuk mengatasi kasus CORS, namun jika ada pengiriman data berformat Json atau yang lebih kompleks dari client maka data tersebut harus dikirimkan dalam bentuk encoded, misalnya pada Javascript gunakan class FormData.

Penulis menyarankan agar tidak menggunakan cara ini untuk production, sebaiknya gunakan pustaka CORS yang bagus seperti `laravel-cors` (<https://github.com/barryvdh/laravel-cors>).

```
1 | composer require barryvdh/laravel-cors
```

```

root@28de49479092:/var/www/larashop-api# composer require barryvdh/laravel-cors
Do not run Composer as root/super user! See https://getcomposer.org/root for details
Using version ^0.11.2 for barryvdh/laravel-cors
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 2 installs, 0 updates, 0 removals
- Installing asm89/stack-cors (1.2.0): Downloading (100%)
- Installing barryvdh/laravel-cors (v0.11.2): Downloading (100%)
Writing lock file
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover
Discovered Package: barryvdh/laravel-cors
Discovered Package: beyondcode/laravel-dump-server
Discovered Package: fideloper/proxy
Discovered Package: laravel/tinker
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Package manifest generated successfully.

```

Lalu pada file `/app/Http/Kernel.php` tepatnya di properti `routeMiddleware`, sesuaikan menjadi

```

1 | protected $routeMiddleware = [
2 |     // ... middleware lain
3 |     'cors' => \Barryvdh\Cors\HandleCors::class,
4 | ];

```

Supaya otomatis tanpa mendeklarasikan secara manual di routing maka cors bisa kita tambahkan pada Kernel yaitu di bagian middlewareGroups.

```

1 protected $middlewareGroups = [
2     'api' => [
3         'throttle:60,1',
4         'bindings',
5         'cors', // <= ini
6     ],
7 ];

```

Multiple Middleware

Kita dapat mengimplementasikan multiple middleware sekaligus dengan menggunakan array. Contohnya sebagai berikut.

```

1 Route::middleware(['throttle:10,1', 'cors'])->group(function () {
2     Route::get('buku/{judul}', 'BookController@cetak');
3 });

```

atau bisa juga nested.

```

1 Route::middleware(['throttle:10,1'])->group(function () {
2     Route::middleware(['cors'])->group(function () {
3         Route::get('buku/{judul}', 'BookController@cetak');
4     });
5 });

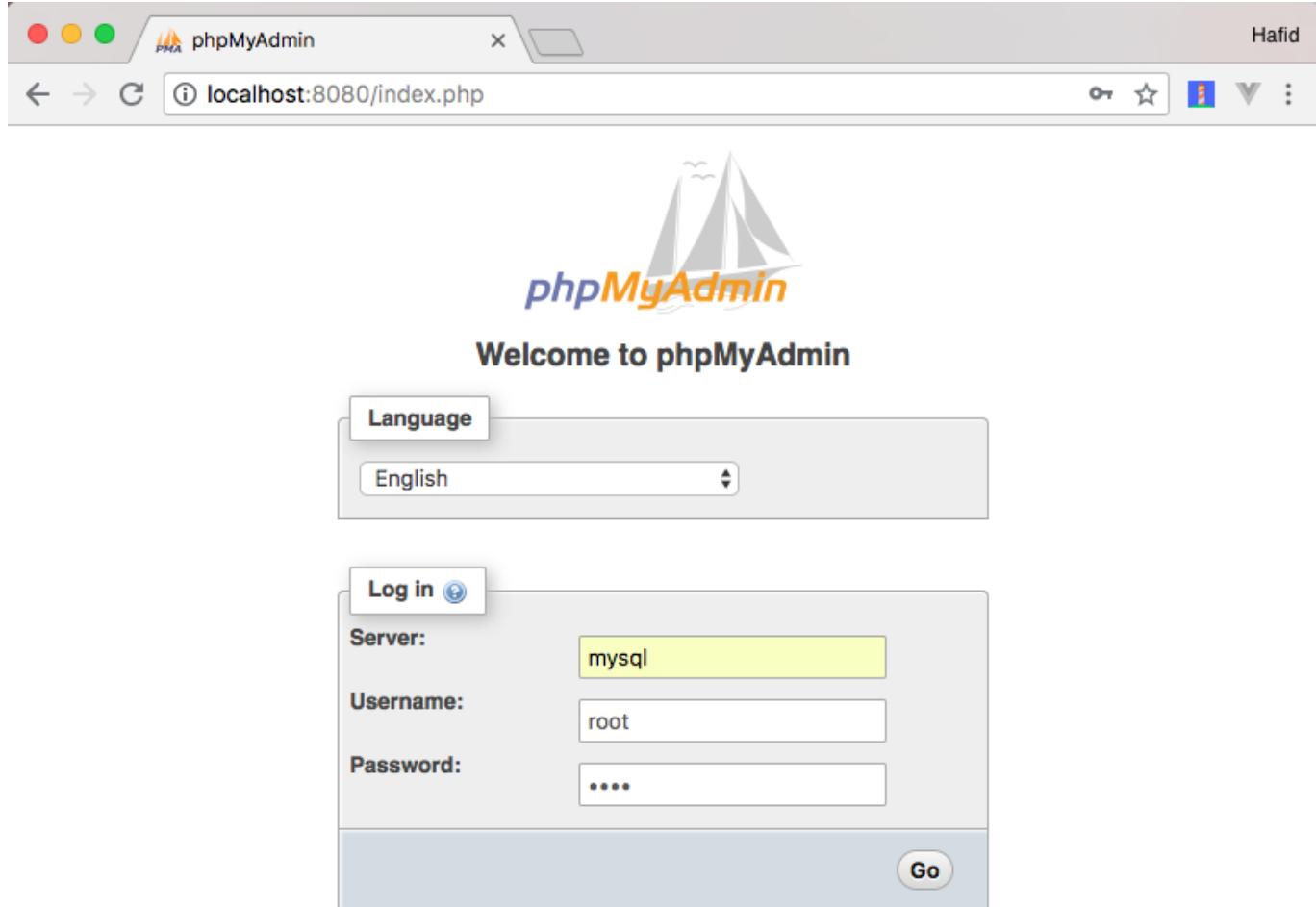
```

Database

Laravel akan memudahkan kita untuk berinteraksi dengan database menggunakan raw SQL, Query Builder atau Eloquent ORM. Laravel mendukung empat database, yaitu:

- MySQL/MariaDB
- PostgreSQL
- SQLite
- SQL Server

Sebelum kita melangkah lebih jauh, mari kita buat dahulu database larashop melalui phpmyadmin.



Pada tab Databases masukkan nama databasenya yaitu larashop, kemudiak klik tombol create

The screenshot shows the "Databases" section of the phpMyAdmin interface. On the left, there's a sidebar with icons for "New", "Recent", and "Favorites". The main panel shows a list of existing databases: "default", "information_schema", "mysql", "performance_schema", and "sys". A "Create database" button is visible. In the input field, "larashop" is typed. To the right of the input field is a dropdown menu set to "utf8_general_ci" and a "Create" button.

Konfigurasi

Sesuaikan konfigurasi database pada file .env laravel

```
1 DB_CONNECTION=mysql
2 DB_HOST=mysql
3 DB_PORT=3306
4 DB_DATABASE=larashop
5 DB_USERNAME=root
6 DB_PASSWORD=root
```

silakan disesuaikan jika menggunakan XAMPP, umumnya menggunakan DB_HOST = localhost dan default password rootnya adalah root.

Migration

Migration seperti version control untuk database yang mengizinkan kita dengan mudah memodifikasi atau meng-share skema database pada projek aplikasi kita, sehingga perubahan dari skema database dapat terlacak.

Pada direktori `database/migrations` kita akan jumpai file migration bawaan Laravel yaitu:

- `2014_10_12_000000_create_users_table.php`
- `2014_10_12_100000_create_password_resets_table.php`

Penamaan file migration ini ada aturannya yaitu dimulai dari waktu pembuatan, action, dan nama table-nya.

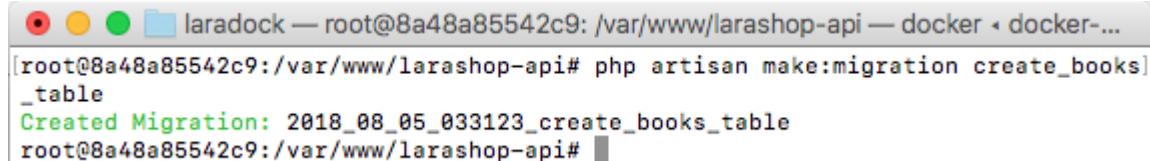
Generate Migration

Kita dapat membuat kode migration melalui fitur `artisan make:migration`. Perintah tersebut akan menggenerate kode dasar migration yang kemudian dapat kita lengkapi.

Tabel Books

Misalnya kita akan membuat tabel bernama `books` maka gunakan perintah berikut.

```
1 | php artisan make:migration create_books_table
```



```
[root@8a48a85542c9:/var/www/larashop-api# php artisan make:migration create_books_table
Created Migration: 2018_08_05_033123_create_books_table
root@8a48a85542c9:/var/www/larashop-api# ]
```

Perintah di atas akan menggenerate file migration pada direktori `database/migrations/2018_08_05_033123_create_books_table.php`

```
1 | <?php
2 | use Illuminate\Support\Facades\Schema;
3 | use Illuminate\Database\Schema\Blueprint;
4 | use Illuminate\Database\Migrations\Migration;
5 |
6 | class CreateBooksTable extends Migration
7 |
8 |     public function up() {
9 |         Schema::create('books', function (Blueprint $table) {
10 |             $table->increments('id');
11 |             $table->timestamps();
12 |         });
13 |     }
14 |
15 |     public function down() {
16 |         Schema::dropIfExists('books');
17 |     }
18 | }
```

Nah pada fungsi `up` tersebut dapat kita tambahkan skema tabel `books` untuk menggenerate tabel `books`, adapun fungsi `down` sebaliknya yaitu untuk menghapus tabel `books`.

- `$table->increments('id');` akan menggenerate field dengan nama id bertipe integer atau angka dan auto increment sekaligus bertindak sebagai primary key.
- `$table->timestamps();` akan menggenerate dua field bernama created_at dan updated_at bertipe integer.

Sebelum kita jalankan, mari kita update fungsi up pada tabel book sesuai dengan skema database yang telah kita bahas pada bab sebelumnya.

books
+ id: Integer {PK}
+ title: String
+ slug: String{U}
+ description: String
+ author: String
+ publisher: String
+ cover: String
+ price: Float
+ weight: Integer
+ views: Integer
+ stock: Integer
+ status: Enum {PUBLISH DRAFT}
+ created_at: timestamps
+ updated_at: timestamps
+ deleted_at: Integer
+ created_by: Integer
+ updated_by: Integer
+ deleted_by: Integer

Berdasarkan skema di atas, maka kode migrationnya kurang lebih sebagai berikut.

```

1 public function up()
2 {
3     Schema::create('books', function (Blueprint $table) {
4         $table->increments('id');
5         $table->string('title');
6         $table->string('slug')->unique();
7         $table->text('description')->nullable();
8         $table->string('author')->nullable();
9         $table->string('publisher')->nullable();
10        $table->string('cover')->nullable();
11        $table->float('price')->unsigned()->default(0);
12        $table->float('weight')->unsigned()->default(0);
13        $table->integer('views')->unsigned()->default(0);
14        $table->integer('stock')->unsigned()->default(0);
15        $table->enum('status', ['PUBLISH', 'DRAFT'])->default('PUBLISH');
16        $table->timestamps(); // created_at, updated_at
17        $table->softDeletes(); // deleted_at

```

```

18     $table->integer('created_by')->nullable();
19     $table->integer('updated_by')->nullable();
20     $table->integer('deleted_by')->nullable();
21 }
22 }
```

Keterangan:

- `string()` akan menggenerate field bertipe teks pendek, sedangkan `text()` digunakan untuk teks yang panjang;
- `integer()` akan menggenerate field bertipe integer;
- `float()` akan menggenerate field bertipe float atau decimal;
- `enum()` akan menggenerate field bertipe enumerasi;
- `unique()` akan menggenerate field dengan key unik;
- `nullable` akan menggenerate field yang boleh tidak diisi karena defaultnya field harus diisi;
- `unsigned()` akan menggenerate field numerik yang nilainya selalu positif atau minimal 0;
- `default()` untuk mendefinisikan nilai default dari suatu field jika dikosongkan;
- `softDeletes()` akan menggenerate field `deleted_at`.

Tabel Users

Meski tabel user telah disediakan migrationnya oleh Laravel, namun ada sedikit perubahan untuk menyesuaikan dengan skema database projek kita. Migration bawaan Laravel tidak perlu dihapus, cukup kita buat migration baru untuk memodifikasinya.

```
1 | php artisan make:migration alter_users_table --table=users
```

```

● ● ● laradock — root@8a48a85542c9: /var/www/larashop-api — docker ↵ docker-...
root@8a48a85542c9:/var/www/larashop-api# php artisan make:migration alter_users_
table --table=users
Created Migration: 2018_08_05_050155_alter_users_table
root@8a48a85542c9:/var/www/larashop-api#
```

Mari kita update hasil generate-nya menjadi sesuai dengan tabel users.

users	
+ id: Integer {PK}	+
+ remember_token: String	
+ email: String(U)	
+ password: String	
+ name: String	
+ roles: String	
+ address: String	
+ city_id: Integer	
+ province_id: Integer	
+ phone: String	
+ avatar: String	
+ status: Enum {ACTIVE INACTIVE}	
+ created_at: timestamps	
+ updated_at: timestamps	

```

1  <?php
2  use Illuminate\Support\Facades\Schema;
3  use Illuminate\Database\Schema\Blueprint;
4  use Illuminate\Database\Migrations\Migration;
5
6  class AlterUsersTable extends Migration
7  {
8      public function up()
9      {
10         Schema::table('users', function (Blueprint $table) {
11             $table->text('roles')->nullable();
12             $table->text('address')->nullable();
13             $table->integer('city_id')->nullable();
14             $table->integer('province_id')->nullable();
15             $table->string('phone')->nullable();
16             $table->string('avatar')->nullable();
17             $table->enum('status', ['ACTIVE', 'INACTIVE'])-
18             >default('ACTIVE');
19         });
20     }
21
22     public function down()
23     {
24         Schema::table('users', function (Blueprint $table) {
25             $table->dropColumn("roles");
26             $table->dropColumn("address");
27             $table->dropColumn("city_id");
28             $table->dropColumn("province_id");
29             $table->dropColumn("phone");
30             $table->dropColumn("avatar");
31             $table->dropColumn("status");

```

```

32     });
33 }

```

Field city_id dan province_id nantinya akan merujuk ke tabel cities dan provinces yang datanya akan kita peroleh dari API RajaOngkir.com (untuk menghitung ongkos kirim)

Tabel Categories

Tabel ini untuk menyimpan data kategori buku, berikut ini skemanya.

categories
+ id: Integer {PK}
+ name: String
+ slug: String{U}
+ image: String
+ status: Enum {PUBLISH DRAFT}
+ created_at: Integer
+ updated_at: Integer
+ created_by: Integer
+ updated_by: Integer
+ deleted_at: timestamps
+ deleted_by: timestamps

```

1 | php artisan make:migration create_categories_table

```

Maka kode migration fungsi up-nya akan menjadi sebagai berikut.

```

1 public function up()
2 {
3     Schema::create('categories', function (Blueprint $table) {
4         $table->increments('id');
5         $table->string('name');
6         $table->string('slug')->unique();
7         $table->string('image')->nullable();
8         $table->enum('status', ['PUBLISH', 'DRAFT'])->default('PUBLISH');
9         $table->timestamps(); // created_at, updated_at
10        $table->softDeletes(); // deleted_at
11        $table->integer('created_by')->nullable();
12        $table->integer('updated_by')->nullable();
13        $table->integer('deleted_by')->nullable();
14    });
15 }

```

Tabel Book Category

Tabel ini untuk menyimpan data relasi antara tabel books dan categories, berikut ini skemanya.

book_category
+ id: Integer{PK}
+ book_id: Integer
+ category_id: Integer
+ created_at: Integer
+ updated_at: Integer

```
1 | php artisan make:migration create_book_category_table
```

Maka kode migration fungsi up-nya akan menjadi sebagai berikut.

```
1 | public function up()
2 | {
3 |     Schema::create('book_category', function (Blueprint $table) {
4 |         $table->increments('id');
5 |         $table->integer('book_id');
6 |         $table->integer('category_id');
7 |         $table->timestamps();
8 |     });
9 | }
```

Tabel Orders

Tabel ini untuk menyimpan data pemesanan buku, berikut ini skemanya.

orders
+ id: Integer {PK}
+ user_id: Integer
+ total_price: Float
+ invoice_number: String
+ courier_service: String
+ status: Enum
+ created_at: Integer
+ updated_at: Integer

Adapun field status bertipe enum dengan nilai: SUBMIT, PROCESS, FINISH, dan CANCEL

```
1 | php artisan make:migration create_orders_table
```

Maka kode migration fungsi up-nya akan menjadi sebagai berikut.

```
1 | public function up()
2 | {
3 |     Schema::create('orders', function (Blueprint $table) {
4 |         $table->increments('id');
5 |         $table->integer('user_id');
6 |         $table->float('total_price');
```

```

7 |     $table->string('invoice_number');
8 |     $table->string('courier_service')->nullable();
9 |     $table->enum('status', ['SUBMIT', 'PROCESS', 'FINISH',
10 |      'CANCEL'])->default('SUBMIT');
11 |     $table->timestamps();
12 |
13 };
```

Tabel Book Order

Tabel ini untuk menyimpan data relasi antara tabel books dan orders, berikut ini skemanya.

book_order
+ id: Integer{PK}
+ order_id: Integer
+ book_id: Integer
+ quantity: Integer
+ created_at: timestamps
+ updated_at: timestamps

```
1 | php artisan make:migration create_book_order_table
```

Maka kode migration fungsi up-nya akan menjadi sebagai berikut.

```

1 | public function up()
2 | {
3 |     Schema::create('book_order', function (Blueprint $table) {
4 |         $table->increments('id');
5 |         $table->integer('book_id');
6 |         $table->integer('order_id');
7 |         $table->integer('quantity');
8 |         $table->timestamps();
9 |     });
10 | }
```

Tabel Provinces

Tabel ini untuk menyimpan data provinces yang didapat dari API RajaOngkir.com, berikut ini skemanya.

provinces
+ id: Integer{PK}
+ province: String

```
1 | php artisan make:migration create_provinces_table
```

Maka kode migration fungsi up-nya akan menjadi sebagai berikut.

```

1 | public function up()
2 | {
```

```

3 Schema::create('provinces', function (Blueprint $table) {
4     $table->increments('id');
5     $table->string('province');
6 });
7 }
```

Tabel Cities

Tabel ini untuk menyimpan data cities yang didapat dari API RajaOngkir.com, tabel ini juga berelasi dengan tabel provinces melalui field province_id, berikut ini skemanya.

cities
+ id: Integer{PK}
+ province_id: Integer
+ city: String
+ type: String
+ postal_code: String

```

1 | php artisan make:migration create_cities_table
```

Maka kode migration fungsi up-nya akan menjadi sebagai berikut.

```

1 public function up()
2 {
3     Schema::create('cities', function (Blueprint $table) {
4         $table->increments('id');
5         $table->integer('province_id');
6         $table->string('city');
7         $table->string('type');
8         $table->string('postal_code');
9     });
10 }
```

Execute Migration

Untuk menjalankan migration pada terminal jalankan perintah berikut.

```

1 | php artisan migrate
```

```
[root@28de49479092:/var/www/larashop-api# php artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table
Migrating: 2018_08_05_033123_create_books_table
Migrated: 2018_08_05_033123_create_books_table
Migrating: 2018_08_05_050155_alter_users_table
Migrated: 2018_08_05_050155_alter_users_table
Migrating: 2018_08_06_205410_create_categories_table
Migrated: 2018_08_06_205410_create_categories_table
Migrating: 2018_08_06_212931_create_orders_table
Migrated: 2018_08_06_212931_create_orders_table
Migrating: 2018_08_06_213012_create_book_category_table
Migrated: 2018_08_06_213012_create_book_category_table
Migrating: 2018_08_06_213037_create_book_order_table
Migrated: 2018_08_06_213037_create_book_order_table
Migrating: 2018_08_06_213134_create_provinces_table
Migrated: 2018_08_06_213134_create_provinces_table
Migrating: 2018_08_06_213147_create_cities_table
Migrated: 2018_08_06_213147_create_cities_table]
```

Perintah di atas akan mengeksekusi fungsi up pada migration.

Mari kita lihat hasil migration ini pada database larashop melalui phpmyadmin.

Table	Action
books	Browse Structure Search Insert Empty Drop
book_category	Browse Structure Search Insert Empty Drop
book_order	Browse Structure Search Insert Empty Drop
categories	Browse Structure Search Insert Empty Drop
cities	Browse Structure Search Insert Empty Drop
migrations	Browse Structure Search Insert Empty Drop
orders	Browse Structure Search Insert Empty Drop
password_resets	Browse Structure Search Insert Empty Drop
provinces	Browse Structure Search Insert Empty Drop
users	Browse Structure Search Insert Empty Drop
10 tables	Sum

Tabel `migrations` digunakan untuk menyimpan histori dari migration yang pernah dilakukan, hal ini untuk kebutuhan rollback.

Lihat struktur tabel `users` yang dihasilkan

Server: mysql » Database: larashop » Table: users

[Browse](#) [Structure](#) [SQL](#) [Search](#) [Insert](#) [Export](#) [Import](#) [More](#)

[Table structure](#) [Relation view](#)

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int(10)		UNSIGNED	No	None		AUTO_INCREMENT
2	name	varchar(255)	utf8mb4_unicode_ci		No	None		
3	email	varchar(255)	utf8mb4_unicode_ci		No	None		
4	password	varchar(255)	utf8mb4_unicode_ci		No	None		
5	remember_token	varchar(100)	utf8mb4_unicode_ci		Yes	None		
6	created_at	timestamp			Yes	None		
7	updated_at	timestamp			Yes	None		
8	roles	text	utf8mb4_unicode_ci		Yes	None		
9	address	text	utf8mb4_unicode_ci		Yes	None		
10	city_id	int(11)			Yes	None		
11	province_id	int(11)			Yes	None		
12	phone	varchar(255)	utf8mb4_unicode_ci		Yes	None		
13	avatar	varchar(255)	utf8mb4_unicode_ci		Yes	None		
14	status	enum('ACTIVE', 'INACTIVE')	utf8mb4_unicode_ci		No	ACTIVE		

Lihat struktur tabel books yang dihasilkan

Server: mysql » Database: larashop » Table: books

[Browse](#) [Structure](#) [SQL](#) [Search](#) [Insert](#) [Export](#) [Import](#) [More](#)

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int(10)		UNSIGNED	No	None		AUTO_INCREMENT
2	title	varchar(255)	utf8mb4_unicode_ci		No	None		
3	slug	varchar(255)	utf8mb4_unicode_ci		No	None		
4	description	text	utf8mb4_unicode_ci		Yes	None		
5	author	varchar(255)	utf8mb4_unicode_ci		Yes	None		
6	publisher	varchar(255)	utf8mb4_unicode_ci		Yes	None		
7	cover	varchar(255)	utf8mb4_unicode_ci		Yes	None		
8	price	double(8,2)		UNSIGNED	No	0.00		
9	weight	double(8,2)		UNSIGNED	No	0.00		
10	views	int(10)		UNSIGNED	No	0		
11	stock	int(10)		UNSIGNED	No	0		
12	status	enum('PUBLISH', 'DRAFT')	utf8mb4_unicode_ci		No	PUBLISH		
13	created_at	timestamp			Yes	None		
14	updated_at	timestamp			Yes	None		
15	deleted_at	timestamp			Yes	None		
16	created_by	int(11)			Yes	None		
17	updated_by	int(11)			Yes	None		
18	deleted_by	int(11)			Yes	None		

Sebaliknya, kita bisa menghapus atau rollback tabel hasil migration dengan menggunakan perintah berikut.

```
1 | php artisan migrate:rollback
```

Perintah di atas akan menjalankan fungsi down pada migration. Sedangkan untuk memperbarui tabel atau menjalankan fungsi down() kemudian up() sekaligus, kita bisa gunakan perintah berikut.

```
1 | php artisan migrate:refresh
```

Atau gunakan perintah

```
1 | php artisan migrate:fresh
```

untuk menghapus semua tabel dalam database dan menjalankan migration.

Seeding

Jika migration fokus menangani skema database dan struktur tabelnya maka seeding fokus pada data yang tersimpan pada tabelnya. Umumnya seeding digunakan untuk menginject data dummy untuk keperluan testing aplikasi. Semua class seeder disimpan pada direktori database/seeds.

Generate Seeder

Kita dapat membuat seeder melalui fitur `artisan make:seeder`.

Tabel Users

Misalkan kita akan membuat seeder untuk tabel `users` maka gunakan perintah berikut.

```
1 | php artisan make:seeder UsersTableSeeder
```

```
root@8a48a85542c9:/var/www/larashop-api# php artisan make:seeder UsersTableSeeder
Seeder created successfully.
root@8a48a85542c9:/var/www/larashop-api#
```

Perintah di atas akan menggenerate tabel `UsersTableSeeder.php` pada direktori `database/seeds/`. Modifikasi file tersebut menjadi sebagai berikut.

```
1 | <?php
2 | use Illuminate\Database\Seeder;
3 | use Illuminate\Support\Facades\DB;
4 |
5 | class UsersTableSeeder extends Seeder
6 |
7 |     public function run() {
8 |         DB::table('users')->insert([
9 |             'name' => 'Anwar',
10 |             'email' => 'anwar@gmail.com',
11 |             'password' => bcrypt('123456'),
12 |             'roles' => json_encode(['CUSTOMER']),
13 |             'status' => 'ACTIVE',
14 |         ],
15 |         [
16 |             'name' => 'Budi',
17 |             'email' => 'budi@gmail.com',
```

```

18     'password' => bcrypt('123456'),
19     'roles' => json_encode(['CUSTOMER']),
20     'status' => 'ACTIVE',
21   ]);
22   // dst
23 }
24 }
```

Kode di atas untuk menginsert dua data dummy users. Sebenarnya kita juga bisa menggunakan pustaka Faker (fzaninotto/faker) untuk membuat data dummy, kebetulan pustaka ini sudah otomatis terinstalasi bersama Laravel.

```

1 public function run()
2 {
3     $users = [];
4     $faker = Faker\Factory::create();
5     for($i=0;$i<5;$i++){
6         $avatar_path = '/var/www/larashop-api/public/images/users';
7         $avatar_fullpath = $faker->image( $avatar_path, 200, 250,
8         'people', true, true, 'people');
9         $avatar = str_replace($avatar_path . '/' , '', $avatar_fullpath);
10        $users[$i] = [
11            'name'      => $faker->name,
12            'email'     => $faker->unique()->safeEmail,
13            'password'  => bcrypt('123456'),
14            'roles'     => json_encode(['CUSTOMER']),
15            'avatar'    => $avatar,
16            'status'    => 'ACTIVE',
17            'created_at'=> Carbon\Carbon::now(),
18        ];
19    }
20    DB::table('users')->insert($users);
}
```

Yang menarik, faker juga dapat menggenerate image avatar. Pada contoh di atas avatar akan digenerate pada direktori `/var/www/larashop-api/public/images/users` (sesuaikan dengan lokasi direktori di komputermu) tapi syaratnya kamu harus membuat folder `users` terlebih dulu pada direktori `public/image/users`

```

1 mkdir public/images
2 mkdir public/images/users
```

Catatan: Jika kamu menggunakan XAMPP maka tentu direktorinya dapat kamu sesuaikan yaitu di `C:\xampp\htdocs\public\images\users`

Jika tertarik menjelajah faker secara lebih mendalam, kamu bisa kunjungi official githubnya di <https://github.com/fzaninotto/Faker>

Oleh karena users dapat memiliki lebih dari satu role maka pada field roles kita menggunakan data dalam format array, adapun fungsi `json_encode` untuk mengkonversi array menjadi string agar bisa disimpan pada database. Untuk membacanya berarti harus didecode kembali.

Adapun field created_at diisi dengan menggunakan class Carbon dengan fungsi now yang mengembalikan timestamp, dengan menggunakan class ini kita bisa lebih mudah mendapatkan dan mengolah data dalam format waktu.

Tabel Books

Kemudian buat juga seeder untuk tabel Books.

```
1 | php artisan make:seeder BooksTableSeeder
```

Lalu tambahkan data dummy pada seeder tabel Books atau BooksTableSeeder.php

```
1 | <?php
2 |
3 | use Illuminate\Database\Seeder;
4 | use Illuminate\Support\Facades\DB;
5 |
6 | class BooksTableSeeder extends Seeder
7 |
8 | {
9 |     /**
10 |      * Run the database seeds.
11 |      *
12 |      * @return void
13 |     */
14 |     public function run()
15 |     {
16 |         DB::table('books')->insert([
17 |             'title' => 'C++ High Performance',
18 |             'slug' => 'c++-high-performance',
19 |             'description' => 'Write code that scales across CPU
registers, multi-core, and machine clusters',
20 |             'author' => 'Viktor Sehr, Björn Andrist',
21 |             'publisher' => 'Packtpub',
22 |             'cover' => 'c++-high-performance.png',
23 |             'price' => 100000,
24 |             'weight' => 0.5,
25 |             'status' => 'PUBLISH',
26 |         ], [
27 |             'title' => 'Mastering Linux Security and Hardening',
28 |             'slug' => 'mastering-linux-security-and-hardening',
29 |             'description' => 'A comprehensive guide to mastering the art
of preventing your Linux system from getting compromised',
30 |             'author' => 'Donald A. Tevault',
31 |             'publisher' => 'Packtpub',
32 |             'cover' => 'mastering-linux-security-and-hardening.png',
33 |             'price' => 125000,
34 |             'weight' => 0.5,
35 |             'status' => 'PUBLISH',
36 |         ], [
37 |             'title' => 'Mastering PostgreSQL 10',
38 |             'slug' => 'mastering-postgresql-10',
39 |             'description' => 'Master the capabilities of PostgreSQL 10 to
efficiently manage and maintain your database',
40 |             'author' => 'Hans-Jürgen Schönig',
41 |         ]
42 |     }
43 | }
```

```

43     'publisher' => 'Packtpub',
44     'cover' => 'mastering-postgresql-10.png',
45     'price' => 90000,
46     'weight' => 0.5,
47     'status' => 'PUBLISH',
48   ], [
49     'title' => 'Python Programming Blueprints',
50     'slug' => 'c++-high-performance',
51     'description' => 'How to build useful, real-world
52 applications in the Python programming language',
53     'author' => 'Daniel Furtado, Marcus Pennington',
54     'publisher' => 'Packtpub',
55     'cover' => 'python-programming-blueprints.png',
56     'price' => 75000,
57     'weight' => 0.5,
58     'status' => 'PUBLISH',
59   ]);
60 }
61 }

```

Tentu, kita juga bisa menggunakan faker.

```

1 public function run()
2 {
3     $books = [];
4     $faker = Faker\Factory::create();
5     $image_categories = ['abstract', 'animals', 'business', 'cats',
6 'city', 'food',
7     'nature', 'technics', 'transport'];
8     for($i=0;$i<25;$i++){
9         $title = $faker->sentence(mt_rand(3, 6));
10        $title = str_replace('.', ' ', $title);
11        $slug = str_replace(' ', '-', strtolower($title));
12        $category = $image_categories[mt_rand(0, 8)];
13        $cover_path = '/var/www/larashop-api/public/images/books';
14        $cover_fullpath = $faker->image( $cover_path, 300, 500,
15 $category, true, true, $category);
16        $cover = str_replace($cover_path . '/' , '',
17 $cover_fullpath);
18        $books[$i] = [
19             'title' => $title,
20             'slug' => $slug,
21             'description' => $faker->text(255),
22             'author' => $faker->name,
23             'publisher' => $faker->company,
24             'cover' => $cover,
25             'price' => mt_rand(1, 10) * 50000,
26             'weight' => 0.5,
27             'status' => 'PUBLISH',
28             'created_at' => Carbon\Carbon::now(),
29         ];
30     }
31 }

```

```
    DB::table('books')->insert($books);
}
```

Jangan lupa buat dulu folder books pada direktori public/image

```
1 | mkdir public/images/books
```

Catatan: str_slug adalah helpers Laravel untuk mengkonversi string menjadi format slug, selengkapnya: <https://laravel.com/docs/6.0/helpers>

Tabel Categories

Buat seeder untuk tabel Categories.

```
1 | php artisan make:seeder CategoriesTableSeeder
```

Lalu tambahkan data dummy pada seeder tabel Categories atau CategoriesTableSeeder.php di fungsi run() dengan menggunakan faker.

```
1 | public function run()
2 |
3 |     $categories = [];
4 |     $faker = Faker\Factory::create();
5 |     $image_categories = ['abstract', 'animals', 'business', 'cats',
6 |     'city', 'food',
7 |     'nature', 'technics', 'transport'];
8 |     for($i=0;$i<8;$i++){
9 |         $name = $faker->unique()->word();
10 |        $name = str_replace('.', '', $name);
11 |        $slug = str_replace(' ', '-', strtolower($name));
12 |        $category = $image_categories[mt_rand(0, 8)];
13 |        $image_path = '/var/www/larashop-api/public/images/categories';
14 |        $image_fullpath = $faker->image( $image_path, 500, 300,
15 |        $category, true, true, $category);
16 |        $image = str_replace($image_path . '/' , '', $image_fullpath);
17 |        $categories[$i] = [
18 |            'name' => $name,
19 |            'slug' => $slug,
20 |            'image' => $image,
21 |            'status' => 'PUBLISH',
22 |            'created_at' => Carbon\Carbon::now(),
23 |        ];
24 |    }
25 |    DB::table('categories')->insert($categories);
}
```

Jangan lupa buat folder categories pada direktori public/image

```
1 | mkdir public/images/categories
```

Tabel Provinces & Cities

Untuk tabel provinces dan city ini data akan kita ambil dari RajaOngkir.com.

RajaOngkir (RO) ini adalah website perantara yang menyediakan layanan pengecekan ongkos kirim dan resi untuk beberapa ekspedisi pengiriman, misalnya: JNE, TIKI, POS, dll. RO sendiri bukan berasal dari perusahaan ekspedisi dan tidak berafiliasi dengan perusahaan ekspedisi, adapun data yang mereka dapatkan konon melalui grabing data otomatis pada website ekspedisi.

Layanan RO gratis untuk paket starter yang dapat kita gunakan untuk mengecek ongkos kirim dari tiga ekspedisi yaitu JNE, TIKI, dan POS. Adapun layanan pengecekan resi ada di paket Basic dan Pro yang berbayar, oleh karenya jika kamu ingin membuat website toko online komersial ya harusnya bayar paket Basic dan Pro.

Pada tutorial ini kita akan mencoba versi gratisnya dulu, untuk itu kita perlu register ke <https://rajaongkir.com/> untuk mendapatkan API KEY. API KEY ini kita gunakan untuk mengakses data ke layanan RO.

Silakan lakukan register pada RO, nanti kamu akan mendapatkan email konfirmasi. Jika sudah register silakan login dengan menggunakan username password saat kamu register tadi. Jika sudah maka buka halaman ini <https://rajaongkir.com/akun/panel> untuk melihat API KEY.

The screenshot shows the RajaOngkir API Key page. At the top, there's a navigation bar with links for Beranda, Integrasi, Download, Bantuan, and a user profile. On the left, a sidebar has links for Profil, API Key (which is selected and highlighted in green), Upgrade, Ubah perujuk, and Ubah password. The main content area is titled 'API Key' and contains the text: 'Gunakan API Key ini untuk menggunakan API RajaOngkir. Untuk informasi lebih lanjut mengenai cara menggunakan API RajaOngkir, silakan baca [dokumentasi](#)'. Below this is a large text input field containing the API key value 'a23e...'. A red box highlights this value. At the bottom of the page, there's a warning message: 'Peringatan: API key Anda berfungsi layaknya username dan password. Jaga baik-baik API key Anda!'

Nah selanjutnya, dengan bekal API KEY gratisan ini kita bisa mendapatkan:

- data nama provinsi dan kota di Indonesia
- mengecek ongkos kirim untuk ekspedisi JNE, TIKI dan POS

Untuk mendapatkan data provinsi kita bisa gunakan URL ini

https://api.rajaongkir.com/starter/province?key=API_KEY

```

1 {
2   "rajaongkir": {
3     "query": {
4       "key": "a23e...1a55405c"
5     },
6     "status": {
7       "code": 200,
8       "description": "OK"
9     },
10    "results": [
11      {
12        "province_id": "1",
13        "province": "Bali"
14      },
15      {
16        "province_id": "2",
17        "province": "Bangka Belitung"
18      }
19    ]
20  }
21}
  
```

Buat seeder untuk tabel provinces.

```
1 php artisan make:seeder ProvincesTableSeeder
```

Lalu tambahkan data province dari API RajaOngkir pada seeder tabel Provinces atau file ProvincesTableSeeder.php di fungsi run().

```

1 public function run()
2 {
3     $url_province = "https://api.rajaongkir.com/starter/province?
4 key=API_KEY";
5     $json_str = file_get_contents($url_province);
6     $json_obj = json_decode($json_str);
7     $provinces = [];
8     foreach($json_obj->rajaongkir->results as $province){
9         $provinces[] = [
10             'id' => $province->province_id,
11             'province' => $province->province
12         ];
13     }
14     DB::table('provinces')->insert($provinces);
15 }
  
```

Untuk mendapatkan data kota kita bisa gunakan URL ini

https://api.rajaongkir.com/starter/city?key=API_KEY

```

1 { "rajaongkir": { "query": { "key": "a23...51a55405c" }, "status": { "code": 200, "description": "OK" } }, "results": [ { "city_id": "1", "province_id": "21", "province": "Nangroe Aceh Darussalam (NAD)", "type": "Kabupaten", "city_name": "Aceh Barat", "postal_code": "23681" } ] }

```

Buat seeder untuk tabel cities.

```
1 | php artisan make:seeder CitiesTableSeeder
```

Lalu tambahkan data kota dari API RajaOngkir pada seeder tabel cities atau file CitiesTableSeeder.php di fungsi run().

```

1 public function run()
2 {
3     $url_city = "https://api.rajaongkir.com/starter/city?key=API_KEY";
4     $json_str = file_get_contents($url_city);
5     $json_obj = json_decode($json_str);
6     $cities = [];
7     foreach($json_obj->rajaongkir->results as $city){
8         $cities[] = [
9             'id'          => $city->city_id,
10            'province_id' => $city->province_id,
11            'city'        => $city->city_name,
12            'type'        => $city->type,
13            'postal_code' => $city->postal_code,
14        ];
15    }
16    DB::table('cities')->insert($cities);
17 }

```

Register Seeder

Setelah kita membuat seeder untuk semua tabel maka langkah selanjutnya adalah mendaftarkan seeder, caranya melalui file database/seeds/DatabaseSeeder.php. Pada fungsi run, masukkan semua seeder yang telah kita buat.

```

1 <?php
2 use Illuminate\Database\Seeder;

```

```

3 class DatabaseSeeder extends Seeder
4 {
5     public function run()
6     {
7         $this->call(UsersTableSeeder::class);
8         $this->call(BooksTableSeeder::class);
9         $this->call(CategoriesTableSeeder::class);
10        $this->call(ProvincesTableSeeder::class);
11        $this->call(CitiesTableSeeder::class);
12    }
13 }
14 }
```

Execute Seeder

Setiap kali kita selesai menulis seeder maka kita perlu meng-generate kembali autoloader dari Composer menggunakan perintah `dump-autoload`.

```
1 | composer dump-autoload
```

Setelah itu baru kita bisa jalankan seeder menggunakan perintah `artisan db:seed` atau bisa juga kita jalankan hanya class seeder tertentu saja menggunakan parameter `--class`

```

1 | php artisan db:seed
2 |
3 | php artisan db:seed --class=UsersTableSeeder
```

```

laradock — root@8a48a85542c9: /var/www/larashop-api — docker ✧ docker-...
root@8a48a85542c9:/var/www/larashop-api# composer dump-autoload
Do not run Composer as root/super user! See https://getcomposer.org/root for details
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover
Discovered Package: fideloper/proxy
Discovered Package: laravel/tinker
Discovered Package: nunomaduro/collision
Package manifest generated successfully.
```

Perintah seeder ini bisa juga kita kombinasikan dengan migration.

```
1 | php artisan migrate:refresh --seed
```

Perintah di atas akan melakukan rollback migration, kemudian mengeksekusi migration dan yang terakhir melakukan seeding data.

```
[root@8a48a85542c9:/var/www/larashop-api# php artisan migrate:refresh --seed
Rolling back: 2018_08_06_213147_create_cities_table
Rolled back: 2018_08_06_213147_create_cities_table
Rolling back: 2018_08_06_213134_create_provinces_table
Rolled back: 2018_08_06_213134_create_provinces_table
Rolling back: 2018_08_06_213037_create_book_order_table
Rolled back: 2018_08_06_213037_create_book_order_table
Rolling back: 2018_08_06_213012_create_book_category_table
Rolled back: 2018_08_06_213012_create_book_category_table
Rolling back: 2018_08_06_212931_create_orders_table
Rolled back: 2018_08_06_212931_create_orders_table
Rolling back: 2018_08_06_205410_create_categories_table
Rolled back: 2018_08_06_205410_create_categories_table
Rolling back: 2018_08_05_050155_alter_users_table
Rolled back: 2018_08_05_050155_alter_users_table
Rolling back: 2018_08_05_033123_create_books_table
Rolled back: 2018_08_05_033123_create_books_table
Rolling back: 2014_10_12_100000_create_password_resets_table
Rolled back: 2014_10_12_100000_create_password_resets_table
Rolling back: 2014_10_12_000000_create_users_table
Rolled back: 2014_10_12_000000_create_users_table
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table
Migrating: 2018_08_05_033123_create_books_table
Migrated: 2018_08_05_033123_create_books_table
Migrating: 2018_08_05_050155_alter_users_table
Migrated: 2018_08_05_050155_alter_users_table
Migrating: 2018_08_06_205410_create_categories_table
Migrated: 2018_08_06_205410_create_categories_table
Migrating: 2018_08_06_212931_create_orders_table
Migrated: 2018_08_06_212931_create_orders_table
Migrating: 2018_08_06_213012_create_book_category_table
Migrated: 2018_08_06_213012_create_book_category_table
Migrating: 2018_08_06_213037_create_book_order_table
Migrated: 2018_08_06_213037_create_book_order_table
Migrating: 2018_08_06_213134_create_provinces_table
Migrated: 2018_08_06_213134_create_provinces_table
Migrating: 2018_08_06_213147_create_cities_table
Migrated: 2018_08_06_213147_create_cities_table
Seeding: UsersTableSeeder
Seeding: BooksTableSeeder
Seeding: CategoriesTableSeeder
Seeding: ProvincesTableSeeder
Seeding: CitiesTableSeeder
root@8a48a85542c9:/var/www/larashop-api# ]
```

Berikut ini contoh data pada tabel books hasil generate faker.

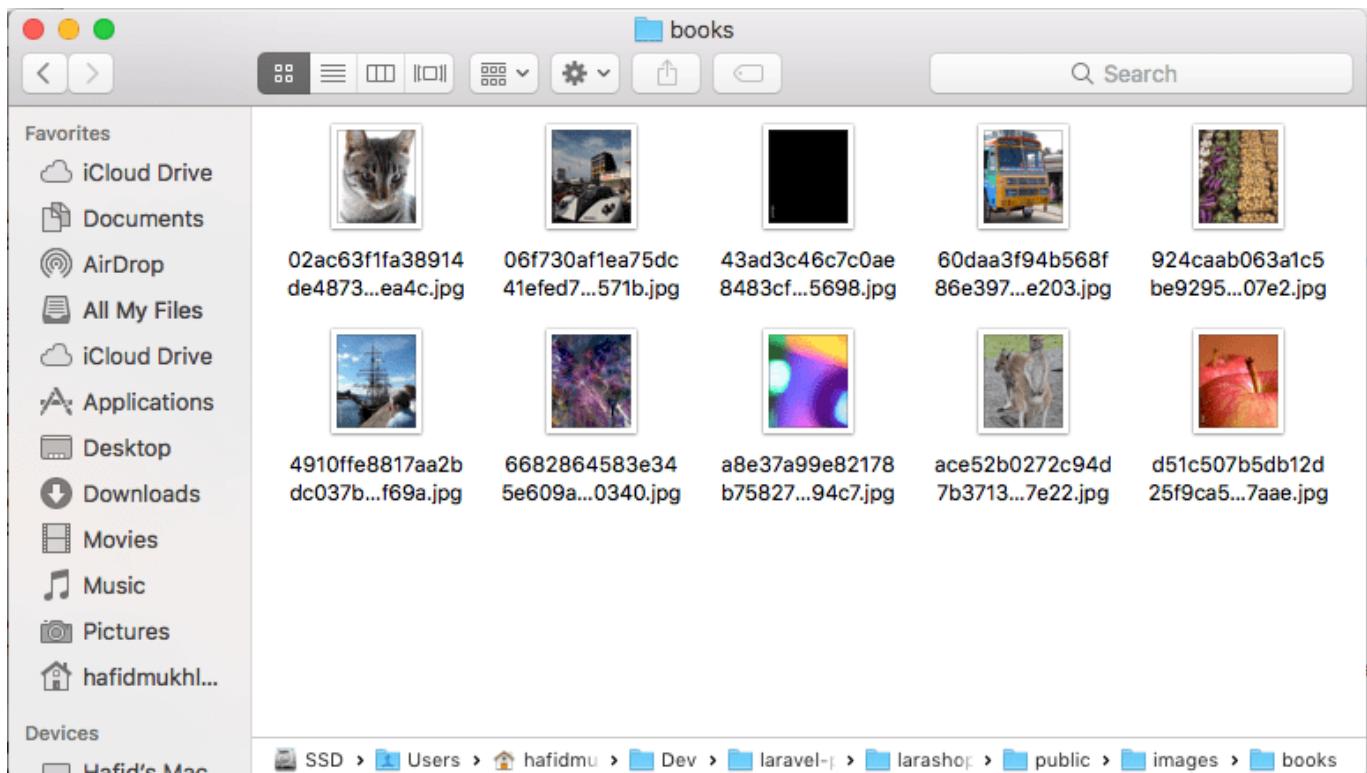
localhost:8080 / mysql / larash X Hafid

localhost:8080/sql.php?db=larashop&table=books&pos=0

Server: mysql » Database: larashop » Table: books

id	title	slug	description	author	publisher	cover	price	weight	views	stock	status
1	Ut tempore doloremque	ut-tempore-doloremque	Sint et unde suscipit. Sed officia delectus nes...	Forest Grimes	Funk, Aufderhar and Kuvalis	ace52b0272c94d7b371355f743087e22.jpg	500000.00	0.50	0	1	PENDING
2	Eligendi praesentium aut voluptatem molestias ut	eligendi-praesentium-aut-voluptatem-molestias-ut	Consequuntur velit iure est omnis dolores Do...	Seth Tromp	Yost PLC	d51c507b5db12d25f9ca590e19c67aae.jpg	200000.00	0.50	0	1	PENDING
3	Nemo fugiat repellat consequatur	nemo-fugiat-repellat-consequatur	Nisi qui voluptate sit possimus quis eos. Ut conse...	Crystal Daugherty	Heaney Inc	924caab063a1c5be92954b4f9f8f07e2.jpg	200000.00	0.50	0	1	PENDING
4	Rerum et nobis mollitia	rerum-et-nobis-mollitia	Rem quod libero expedita dolor aspernatur iure nec...	Dr. Alf Auer MD	Schneider, Stoltenberg and Leffler	43ad3c46c7c0ae8483cf7c1fea285698.jpg	450000.00	0.50	0	1	PENDING
5	Fugit voluptibus delectus in	fugit-voluptibus-delectus-in	Et explicabo adipisci dolore ab qui quam ex accusa...	Miss Cassidy Cremin MD	Yost-Rosenbaum	06f730af1ea75dc41efed7f8b5ce571b.jpg	150000.00	0.50	0	1	PENDING

Sedangkan berikut ini contoh file cover buku hasil dari generate faker.



Interact with Database

Ada setidaknya tiga cara yang bisa kita lakukan untuk berinteraksi dengan database yang telah terhubung dengan Laravel. Tiga cara itu adalah raw query, query builder dan ORM eloquent.

Raw Query

Raw query atau query mentah dapat dijalankan dengan menggunakan class DB (`Illuminate\Support\Facades\DB`). DB menyediakan fungsi untuk setiap jenis query yaitu: `select`, `update`, `insert`, `delete`, dan `statement`.

Untuk mencobanya, mari kita buka lagi BookController. Tambahkan fungsi baru misalnya `index()`.

```

1 public function index()
2 {
3     $books = DB::select('select * from books');
4     return $books;
5 }
```

Jangan lupa tambahkan kode use DB; pada bagian atas. Lho mengapa bukan use Illuminate\Support\Facades\DB;? Yap, si Laravel telah mengaliaskannya sehingga class DB yang dimaksud adalah Illuminate\Support\Facades\DB. Konfigurasi alias ada pada file config/app.php. Lalu pada routing api.php, kita hapus routing-routing yang pernah kita buat sebelumnya, kemudian kita ubah menjadi sebagai berikut.

```

1 Route::prefix('v1')->group(function () {
2     Route::get('books', 'BookController@index');
3 });
```

Untuk menguji cobanya, gunakan tools postman, akses URL resource tersebut yaitu <http://larashop-api.test/v1/books>, dan hasilnya:

The screenshot shows a Postman interface with a GET request to `http://larashop-api.test/v1/books`. The response is displayed in 'Pretty' JSON format, which looks like this:

```

1 {
2     "id": 28,
3     "title": "Test updateOrCreate",
4     "slug": "test-updateorcreate",
5     "description": null,
6     "author": null,
7     "publisher": null,
8     "cover": null,
9     "price": 0,
10    "weight": 0,
11    "views": 0,
12    "stock": 1,
13    "status": "PUBLISH",
14    "created_at": "2018-08-08 01:16:09",
15    "updated_at": "2018-08-08 01:16:09",
16    "deleted_at": null,
17    "created_by": null,
18    "updated_by": null,
19    "deleted_by": null
20 }
```

Fungsi select akan selalu mengembalikan array, sedangkan return array ketika diakses via HTTP request akan memberikan respon berformat JSON.

Kita bisa juga menggunakan parameter where pada query select untuk memfilter data, di mana parameter tersebut nilainya sesuai dengan dengan parameter yang dikirimkan user melalui routing.

Pada controller, tambahkan fungsi view().

```

1 public function view($id)
2 {
3     $book = DB::select('select * from books where id = ?', [ $id ]);
4     return $book;
5 }
```

Di samping menggunakan tanda ?, kita juga bisa membinding nama variabel.

```
1 | $book = DB::select('select * from books where id = :id', ['id' => $id]);
```

Kode di atas tentu lebih mudah dibaca dibandingkan dengan menggunakan tanda ?.

Catatan: nama variabel binding tidak harus sama dengan nama variabel yang dikirimkan.

Lalu pada routing api.php, tambahkan routing baru yang mengarah ke function view.

```
1 | Route::prefix('v1')->group(function () {
2 |     Route::get('books', 'BookController@index');
3 |     Route::get('book/{id}', 'BookController@view')->where('id', '[0-9]+');
4 |     // <== tambahkan ini
5 | });
```

Akses menggunakan URL v1/book/2, maka hasilnya:

The screenshot shows a Postman request to `http://larashop-api.test/v1/book/2`. The response is a JSON object representing a book with ID 2, containing fields like title, slug, description, author, publisher, cover, price, weight, views, stock, status, created_at, updated_at, deleted_at, created_by, updated_by, and deleted_by.

```

1 [ 
2 { 
3   "id": 2,
4   "title": "Labore accusamus qui similique qui qui",
5   "slug": "labore-accusamus-qui-similique-qui-qui",
6   "description": "Quia omnis voluptas quo excepturi asperiores odio. Aspernatur est porro sunt delectus.",
7   "author": "Dr. Geo Paucek I",
8   "publisher": "McKenzie-Walter",
9   "cover": "e0988c6c7c86133047da410550a657f4.jpg",
10  "price": 50000,
11  "weight": 0.5,
12  "views": 0,
13  "stock": 1,
14  "status": "PUBLISH",
15  "created_at": "2018-08-07 19:51:41",
16  "updated_at": null,
17  "deleted_at": null,
18  "created_by": null,
19  "updated_by": null,
20  "deleted_by": null
21 }
  ]

```

Untuk fungsi query yang lain seperti insert, update, delete, dan statement bisa dilihat pada contoh berikut.

```
1 | DB::insert('insert into books (title, slug) values (?, ?)', ['Learn
2 | VueJS', 'learn-vuejs']);
3 |
4 | $affected = DB::update('update books set price = ? where id = ?', [125000,
5 | 3]);
6 |
7 | $deleted = DB::delete('delete from books where id=?', [5]);
8 |
9 | DB::statement('drop table books');
```

Database transaction juga didukung dengan menggunakan kode berikut.

```

1 DB::transaction(function () {
2     DB::insert('insert into books (title) values (?)', ['Learn VueJS']);
3
4     $affected = DB::update('update books set price = ? where id = ?',
5     [125000, 3]);
6
7     $deleted = DB::delete('delete from books where id=?', [5]);
8 });

```

Atau bisa juga menggunakan cara manual `DB::beginTransaction()`; untuk rollback transaction `DB::rollBack()`; dan untuk commit transaction menggunakan `DB::commit()`;

Query Builder

Cara lain yang bisa kita lakukan untuk berinteraksi dengan database adalah dengan menggunakan query builder. Sebenarnya query builder ini hampir sama dengan raw query namun sedikit lebih singkat kodennya.

Catatan: silakan terapkan kode berikut pada Controller, sebagaimana contoh pada raw query. (nama function-nya bebas saja)

```

1 // select * from books
2 $books = DB::table('books')->get();
3
4 // select * from books where id = 3
5 $books = DB::table('books')->where('id', 3)->first();
6
7 // select title from books where id = 3
8 $title = DB::table('books')->where('id', 3)->value('title');
9
10 // mereturn collection data
11 // select id, title from books
12 $books = DB::table('books')->pluck('id', 'title');
13 foreach ($books as $id => $title) {
14     echo $title;
15 }
16
17 // select count(*) from books
18 $count_books = DB::table('books')->count();
19
20 // select max(price) from books
21 $max_book_price = DB::table('books')->max('price');
22
23 // select average(price) from books
24 $avg_book_price = DB::table('books')->avg('price');
25
26 // insert into books (title) values ('Learn VueJS')
27 DB::table('books')->insert(
28     ['title' => 'Learn VueJS', 'slug' => 'learn-vuejs']
29 );
30
31 // update books set price = 125000 where id =3
32 DB::table('books')
33     ->where('id', 3)

```

```
Licensed to Riva Pebrian - lvhotpebrian@gmail.com - 085324454424 at 22/10/2019 20:40:19
34     ->update(['price' => 125000]);
35
36 // delete from books where id=5
37 DB::table('books')->where('id', '=', 5)->delete();
```

Selengkapnya mengenai query builder dapat kamu baca melalui tautan ini <https://laravel.com/docs/6.0/queries>.

ORM Eloquent

Cara ketiga berinteraksi dengan database adalah menggunakan ORM Eloquent. Eloquent merupakan ORM atau object relational models yang digunakan oleh Laravel untuk mengimplementasikan konsep ActiveRecord. Setiap tabel dalam database mempunyai keterkaitan dengan "Model" yang digunakan untuk berinteraksi (melakukan query) dengan tabel tersebut seperti query select, insert, dan delete.

Membuat Model Eloquent

Mari kita buat sebuah model Eloquent pada directory `app`, di mana model tersebut extend dengan class `Illuminate\Database\Eloquent\Model`. Untuk membuatnya gunakan perintah `artisan make:model` supaya lebih mudah:

```
1 | php artisan make:model Book
```

Catatan: tambahkan parameter `-m` atau `--migration` jika kita ingin sekaligus membuat kode untuk database migrationnya

```
1 | php artisan make:model Book --migration
2 |
3 | php artisan make:model Book -m
```

Hanya saja pada bagian sebelumnya kan kita sudah membuat database migration.

```
● ○ ● laradock — root@8a48a85542c9: /var/www/larashop-api — docker ↵ docker-...
[root@8a48a85542c9:/var/www/larashop-api# php artisan make:model Book
Model created successfully.
root@8a48a85542c9:/var/www/larashop-api# ]
```

Berikut ini hasilnya pada `app\Book.php`

```
1 | <?php
2 | namespace App;
3 | use Illuminate\Database\Eloquent\Model;
4 | class Book extends Model
5 | {
6 |     //
7 | }
```

Buat juga model untuk tabel yang lain.

```
root@8a48a85542c9:/var/www/larashop-api# php artisan make:model Category
Model created successfully.
root@8a48a85542c9:/var/www/larashop-api# php artisan make:model City
Model created successfully.
root@8a48a85542c9:/var/www/larashop-api# php artisan make:model Province
Model created successfully.
root@8a48a85542c9:/var/www/larashop-api# php artisan make:model Order
Model created successfully.
root@8a48a85542c9:/var/www/larashop-api# php artisan make:model BookOrder
Model created successfully.
root@8a48a85542c9:/var/www/larashop-api# php artisan make:model BookCategory
Model created successfully.
```

Catatan: hasil dari generate tabel ini hanya berupa kerangka model saja, oleh karena itu perlu kita lengkapi.

Konvensi Model Eloquent

Beberapa konvensi atau kesepakatan dalam pembuatan atau penulisan model eloquent.

1. Nama Class & Nama Tabel

Nama class menggunakan bentuk singular (tunggal), seperti: Book, User, Category. Namun tabel database yang direpresentasikannya menggunakan bentuk plural seperti: books, users, categories. Jika tidak mengikuti konvensi tersebut maka kita harus definisikan secara spesifik pada model dengan menggunakan properti table, misalnya nama tabelnya 'my_books'

```
1 | protected $table = 'my_books';
```

2. Primary Key

Eloquent berasumsi bahwa setiap tabel memiliki primary key pada field id. Nah jika primary key tabel kita bukan id maka kita harus definisikan spesifik. Di samping itu, eloquent juga berasumsi bahwa primary key pada setiap tabel bertipe integer dan autoincrement, di mana eloquent akan melakukan casting secara otomatis primary key menjadi integer. Nah jika bukan integer dan tidak autoincrement maka perlu kita definisikan spesifik seperti berikut.

```
1 | protected primaryKey = 'code'; // string
2 | public $incrementing = false;
3 | protected $keyType = 'string';
```

3. Timestamps

Secara default, eloquent juga berasumsi setiap tabel memiliki field created_at dan updated_at. Namun apabila nama field kita bukan created_at dan updated_at maka kita bisa definisikan secara spesifik sebagai berikut.

```
1 | const CREATED_AT = 'creation_date';
2 | const UPDATED_AT = 'last_update';
```

Tapi, ada kalanya tabel kita tidak punya dua field itu maka kita perlu mendefinisikan properti \$timestamps bernilai false. Pada kasus kita, contohnya ada pada tabel provinces dan cities, dimana kedua tabel itu tidak mempunyai dua field tersebut.

```
1 | public $timestamps = false;
```

Berdasarkan konvensi di atas, berikut ini model untuk tabel-tabel yang kita susun:

Catatan: terdapat satu tambahan properti pada class model yaitu `$fillable` yang digunakan untuk mendefinisikan field-field mana yang bisa diisi nilainya secara batch, tentang properti `$fillable` akan dijelaskan lagi pada bagian selanjutnya.

- Model Book

```

1  class Book extends Model
2  {
3      protected $fillable = [
4          'title', 'slug', 'description', 'author', 'publisher',
5          'cover', 'price', 'weight', 'stock', 'status'
6      ];
7  }

```

- Model Category

```

1  class Category extends Model
2  {
3      protected $fillable = [
4          'name', 'slug', 'image', 'status'
5      ];
6  }

```

- Model BookCategory

```

1  class BookCategory extends Model
2  {
3      protected $table = 'book_category';
4      protected $fillable = [
5          'book_id', 'category_id', 'invoice_number', 'status'
6      ];
7  }

```

- Model Order

```

1  class Order extends Model
2  {
3      protected $fillable = [
4          'user_id', 'total_price', 'invoice_number', 'status'
5      ];
6  }

```

- Model BookOrder

```

1  class BookOrder extends Model
2  {
3      protected $table = 'book_order';
4
5      protected $fillable = [

```

```

6     'book_id', 'order_id', 'quantity'
7 ];
8 }
```

- Model User

Model ini masih sama dengan defaultnya, hanya saja kita tambahkan beberapa filed pada properti \$fillable sesuai dengan desain database yang kita buat.

```

1 protected $fillable = [
2     'name', 'email', 'password', 'username', 'roles',
3     'address', 'city_id', 'province_id', 'phone', 'avatar', 'status'
4 ];
```

- Model Province

```

1 class Province extends Model
2 {
3     public $timestamps = false;
4 }
```

- Model City

```

1 class City extends Model
2 {
3     public $timestamps = false;
4 }
```

Query Pada Model Eloquent

Setelah kita membuat model eloquent maka kita bisa gunakan untuk melakukan query data dengan mudah.

Catatan: silakan terapkan kode berikut pada Controller, sebagaimana contoh pada raw query.

```

1 // select * from books
2 $books = \App\Book::all();
3 foreach ($books as $book) {
4     echo $book->title;
5 }
6
7 // select * from books where status = 'PUBLISH' limit 10 order by title
8 desc
9 $books = \App\Book::where('status', 'PUBLISH')
10    ->orderBy('title', 'asc')
11    ->limit(10)
12    ->get();
```

Setiap fungsi pada eloquent, baik all() maupun get() akan mengembalikan data berbentuk *collection*. Dengan bentuk tersebut kita bisa menjalankan berbagai macam fungsi seperti filter, reject, random, dsb. Lihat contoh berikut.

```

1 // jangan ambil buku yang statusnya draft
2 $published_books = $books->reject(function ($book) {
3     return $book->status=='DRAFT';
4 });
5
6 // ambil buku yang statusnya publish
7 $published_books = $books->filter(function ($book) {
8     return $book->status=='PUBLISH';
9 });
10
11 // ambil 2 items buku secara random
12 $books->random(2)->all();

```

Catatan: fungsi keren lain pada *collection* yang bisa kita gunakan bisa dilihat pada tautan ini
<https://laravel.com/docs/6.0/eloquent-collections#available-methods>

Fungsi-fungsi pada *query builder* juga bisa kita gunakan di sini.

```

1 $books = \App\Book::all();
2 // ambil record pertama
3 $first_book = $books->first();
4
5 // ambil nilai dari atribut title pada record pertama
6 $title = $books->first()->value('title');

```

Selain menggunakan fungsi `first()` untuk mendapatkan *record* pertama saja, kita juga bisa gunakan fungsi `find()` sebagai berikut.

```

1 // select * from books where id = 1
2 $book = \App\Book::find(1);
3 echo $book->title;

```

Di samping itu, fungsi `find` juga bisa digunakan untuk mengambil beberapa *record* berdasarkan id atau primary key-nya.

```

1 $books = \App\Book::find([1, 2, 3]);

```

Kita bisa gunakan fungsi `findOrFail()` atau `firstOrFail()` untuk mengantisipasi *record* yang tidak ditemukan karena akan mengembalikan HTTP 404.

```

1 $book = \App\Book::findOrFail(1);
2
3 $book = \App\Book::where('status', '=', 'DRAFT')->firstOrFail();

```

Berikut ini contoh penggunaan fungsi *aggregat* seperti: count, max, avg

```

1 $count = \App\Book::count();
2 $max_price = \App\Book::max('price'); // nilai maksimal
3 $avg_price = \App\Book::avg('price'); // rata-rata

```

Query lain seperti insert, update, delete pun dengan mudah bisa kita lakukan.

```

1 // insert into books (title, slug) values ('Learn VueJS', 'learn-vuejs')
2 $book = new \App\Book;
3 $book->title = 'Learn VueJS';
4 $book->slug = str_slug($book->title);
5 $book->save();

```

Catatan: str_slug adalah fungsi bawaan Laravel untuk merubah teks menjadi format URL slug.

```

1 // update books set price = 125000 where id =3
2 $book = \App\Book::find(3);
3 $book->price = 125000;
4 $book->save();
5
6 // attau bisa juga
7 \App\Book::where('id', 3)
8     ->update([
9         'price' => 125000
10    ]);
11
12 // delete from books where id = 2
13 $book = \App\Book::find(2);
14 $book->delete()
15
16 // delete from books where id = 2
17 \App\Book::destroy(2);
18
19 // delete from books where id = 2 or 5
20 \App\Book::destroy([2,5]);
21
22 // delete from books where id = 2
23 \App\Book::where('id', 2)->delete();

```

Batch Insert Pada Model Eloquent

Batch insert atau menambahkan data secara massal (*mass assignment*) tidak bisa langsung kita terapkan karena secara default si Eloquent mencegah hal ini. Oleh karenanya kita perlu meng-enable fitur ini melalui properti `$fillable` pada model Eloquent. Properti ini berisi array dari field-field yang boleh diisi dengan cara *mass assignment*.

```

1 protected $fillable = ['title', 'slug'];

```

Kebalikan dari properti ini adalah properti `guarded` yang melakukan blacklist terhadap field-field yang tidak boleh diisi secara massal. Nah menggunakan properti ini kita bisa membuat semua field dapat diisi secara massal dengan mendefinisikannya sebagai empty array.

```

1 protected $guarded = [];

```

Maka kemudian kita bisa gunakan

```

1 $book = \App\Book::create(
2     ['title' => 'Judul 01', 'slug' => 'judul-01']
3 );
4
5 $book = \App\Book::create(
6     ['title' => 'Judul 02', 'slug' => 'judul-02'],
7     ['title' => 'Judul 03', 'slug' => 'judul-03'],
8 );
9
10 // update judul buku dengan id 26
11 $book = \App\Book::find(26);
12 $book->fill(
13     ['title' => 'Judul 01 - updated']
14 );

```

Fitur lain yang menarik adalah fitur `updateOrCreate` yaitu update data namun jika data tidak ditemukan maka create/insert.

```

1 $book = \App\Book::updateOrCreate(
2     ['id' => '28', 'title' => 'Test updateOrCreate', 'slug' => 'test-
3 updateorcreate'],
4     ['price' => 99]
5 );

```

Soft Delete

Soft delete merupakan fitur pada model Eloquent untuk menghidden data dari suatu tabel tanpa benar-benar menghapusnya. Data yang dihapus dengan soft delete ini hanya akan diberikan flag atau tanda bahwa dia telah dihapus melalui field `deleted_at`.

Untuk mengimplementasikan softdelete pada model maka pastikan kita telah menambahkan field `deleted_at` pada model eloquent kita, kemudian gunakan trait `Illuminate\Database\Eloquent\SoftDeletes`, dan tambahkan `deleted_id` pada properti `$dates`.

```

1 use Illuminate\Database\Eloquent\Model;
2 use Illuminate\Database\Eloquent\SoftDeletes;
3
4 class Book extends Model
5 {
6     use SoftDeletes;
7     protected $dates = ['deleted_at'];
8
9     protected $fillable = [
10         'title', 'slug', 'description', 'author', 'publisher',
11         'cover', 'price', 'weight', 'stock', 'status'
12     ];
13 }

```

Tambahkan untuk semua model yang menggunakan softdelete.

```

1 use Illuminate\Database\Eloquent\Model;
2 use Illuminate\Database\Eloquent\SoftDeletes;

```

```

3 class Category extends Model
4 {
5     use SoftDeletes;
6     protected $dates = ['deleted_at'];
7
8     protected $fillable =
9         ['name', 'slug', 'image', 'status']
10    ];
11
12
13 }
```

Ketika kita menggunakan softdelete maka fungsi `delete()` tidak akan menghapus record, kecuali hanya mengupdate field `deleted_at` saja. Adapun untuk menghapus record secara permanen atau dihapus recordnya dari tabel maka gunakan fungsi `forceDelete()`.

```

1 $book = \App\Book::find(1);
2 $book->forceDelete();
```

Demikian juga ketika kita melakukan query select maka record yang field `deleted_at`-nya telah terisi atau tidak kosong akan otomatis dikecualikan. Nah, jika kita ingin memasukkan record yang telah dihapus dengan metode soft delete maka bisa menggunakan fungsi `withTrashed()`, sebaliknya jika hanya ingin menampilkan record yang sudah dihapus maka gunakan fungsi `onlyTrashed()`.

```

1 // semua record buku
2 $books = \App\Book::withTrashed()->get();
3
4 // semua record buku yang sudah dihapus
5 $books = \App\Book::onlyTrashed()->get();
```

Untuk me-*restore* data yang sudah terhapus, kita bisa gunakan fungsi `restore()`.

```
1 \App\Book::withTrashed()->restore();
```

API Resources

Ketika kita membangun web service maka kita perlu merubah data yang berasal dari Eloquent menjadi berformat JSON sebagai respon kepada client. Laravel mempunyai mekanisme untuk melakukannya yaitu dengan menggunakan fitur resources ini. Resource merupakan sebuah class yang dapat kita generate scaffoldingnya menggunakan perintah `artisan make:resource`. Hasil generate akan disimpan dalam direktori `app/Http/Resources`

Berikut ini perintah untuk menggenerate kerangka resource Book.

```
1 php artisan make:resource Book
```

Maka file `Book.php` akan terlihat sebagai berikut.

```

1 <?php
2 namespace App\Http\Resources;
3 use Illuminate\Http\Resources\Json\JsonResource;
```

```

4 class Book extends JsonResource
5 {
6     public function toArray($request)
7     {
8         return parent::toArray($request);
9     }
10}

```

Fungsi `toArray()` di atas akan menampilkan semua field dari tabel. Namun kita bisa mendefinisikan secara spesifik, misalnya:

```

1 public function toArray($request)
2 {
3     return [
4         'id' => $this->id,
5         'title' => $this->title,
6         'created_at' => $this->created_at,
7         'updated_at' => $this->updated_at,
8     ];
9 }

```

Kemudian pada `route/api.php`, tambahkan routing ke resource tersebut, misalnya.

```

1 use App\Book;
2 use App\Http\Resources\Book as BookResource;
3
4 Route::get('/book', function () {
5     return new BookResource(Book::find(1));
6 });

```

Bungkus query book dengan class `BookResource`. Tentu boleh juga jika kita masukkan kode di atas pada controller fungsi view supaya route lebih rapi. Buka `BookController`, pada fungsi view, ubah menjadi sebagai berikut.

```

1 public function view($id)
2 {
3     //$book = DB::select('select * from books where id = ?', [ $id ]);
4     $book = new BookResource(Book::find($id));
5     return $book;
6 }

```

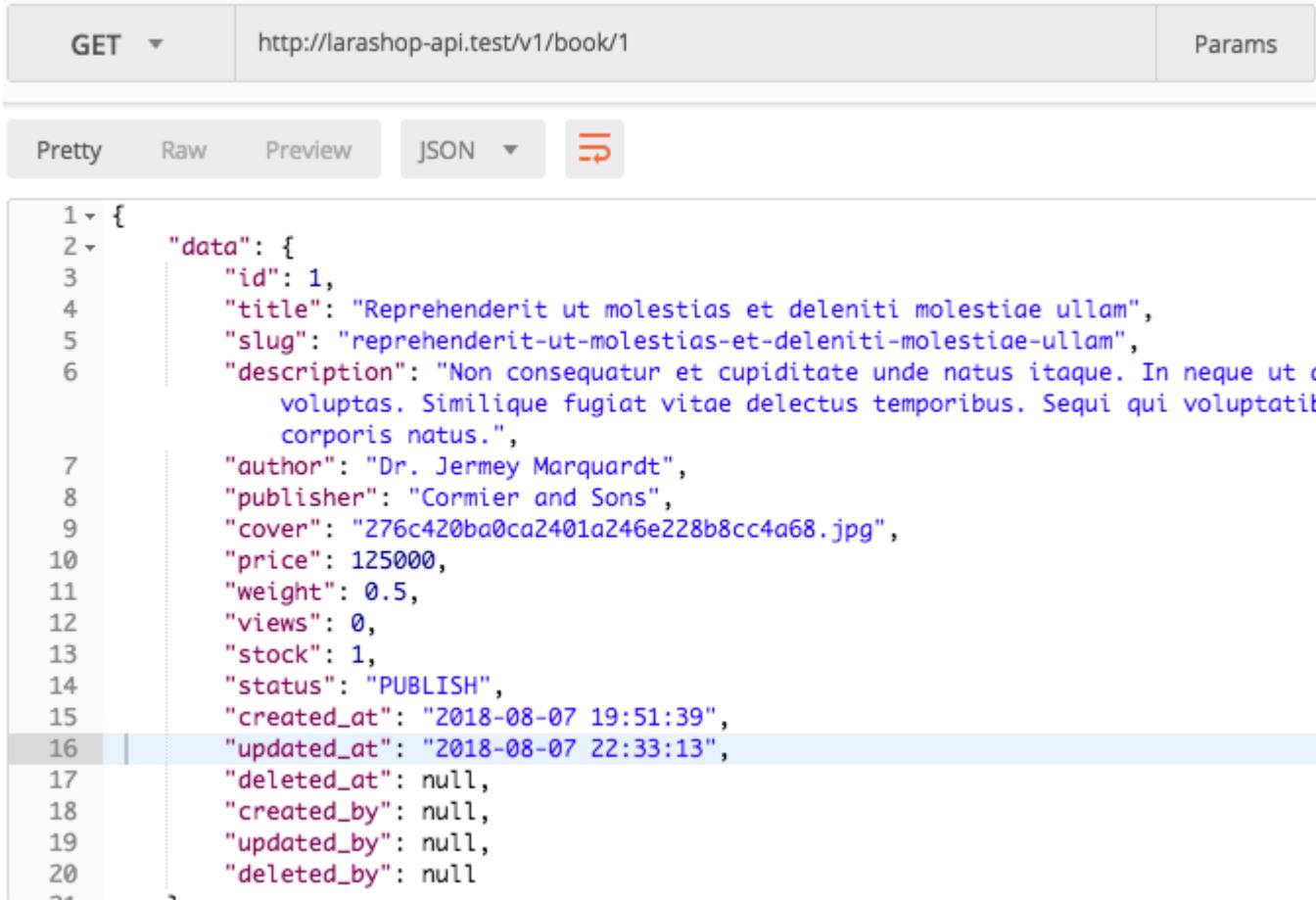
Lalu akses dengan route sesuai dengan `route/api.php` sebelumnya.

```

1 Route::prefix('v1')->group(function () {
2     Route::get('books', 'BookController@index');
3     Route::get('book/{id}', 'BookController@view')->where('id', '[0-9]+');
4 });

```

Gunakan format URL endpoint ini `/v1/book/1`

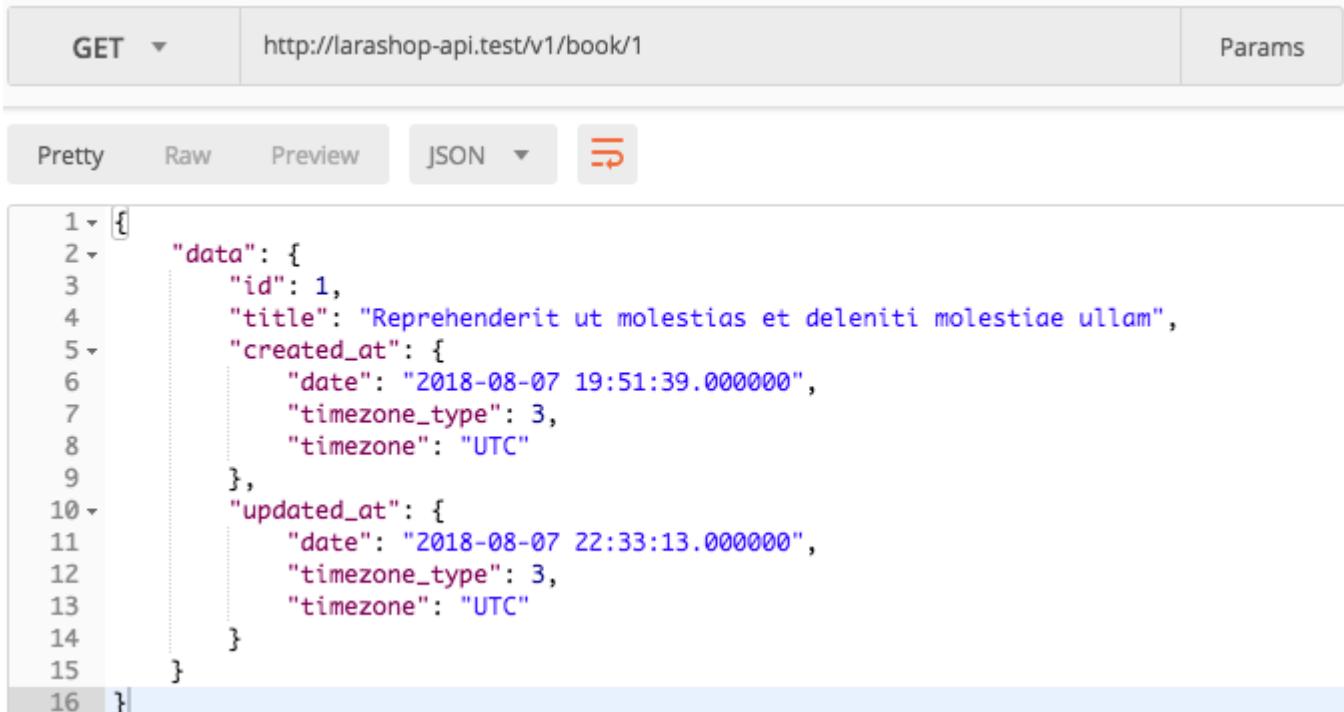


```

1 - {
2 -   "data": {
3 -     "id": 1,
4 -     "title": "Reprehenderit ut molestias et deleniti molestiae ullam",
5 -     "slug": "reprehenderit-ut-molestias-et-deleniti-molestiae-ullam",
6 -     "description": "Non consequatur et cupiditate unde natus itaque. In neque ut a voluptas. Similique fugiat vitae delectus temporibus. Sequi qui voluptatibus corporis natus.",
7 -     "author": "Dr. Jermey Marquardt",
8 -     "publisher": "Cormier and Sons",
9 -     "cover": "276c420ba0ca2401a246e228b8cc4a68.jpg",
10 -    "price": 125000,
11 -    "weight": 0.5,
12 -    "views": 0,
13 -    "stock": 1,
14 -    "status": "PUBLISH",
15 -    "created_at": "2018-08-07 19:51:39",
16 -    "updated_at": "2018-08-07 22:33:13",
17 -    "deleted_at": null,
18 -    "created_by": null,
19 -    "updated_by": null,
20 -    "deleted_by": null
21 -  }

```

Maka response dari web service kita akan berformat seperti di atas yaitu data buku di wrap dengan key data. Atau jika fungsi `toArray()` kita definisikan spesifik fieldnya maka akan tampil sebagai berikut.



```

1 - [
2 -   {
3 -     "data": {
4 -       "id": 1,
5 -       "title": "Reprehenderit ut molestias et deleniti molestiae ullam",
6 -       "created_at": {
7 -         "date": "2018-08-07 19:51:39.000000",
8 -         "timezone_type": 3,
9 -         "timezone": "UTC"
10 -       },
11 -       "updated_at": {
12 -         "date": "2018-08-07 22:33:13.000000",
13 -         "timezone_type": 3,
14 -         "timezone": "UTC"
15 -       }
16 -     }

```

Pada contoh sebelumnya, resource berbentuk data tunggal. Bagaimana jika resource berbentuk collection? Untuk membuat resource collection kita bisa melakukan langkah yang sama, hanya saja pada perintah artisan kita tambahkan parameter `--collection` atau bisa juga tambahkan akhiran (suffix) `Collection` (pilih salah satu). Contoh:

```

1 php artisan make:resource Books --collection
2
3 php artisan make:resource BookCollection

```

Perintah perintah pertama akan menghasilkan file di app/Http/Resources/Books.php sedangkan perintah kedua di app/Http/Resources/BookCollection.php, sebenarnya keduanya sama saja alias hanya beda nama classnya, oleh karena itu pilih salah satu.

Berikut penampakannya.

```

1 <?php
2 namespace App\Http\Resources;
3 use Illuminate\Http\Resources\Json\ResourceCollection;
4 class Books extends ResourceCollection
5 {
6     public function toArray($request)
7     {
8         return parent::toArray($request);
9     }
10 }

```

Buka BookController, pada fungsi index(), ubah kodennya menjadi sebagai berikut.

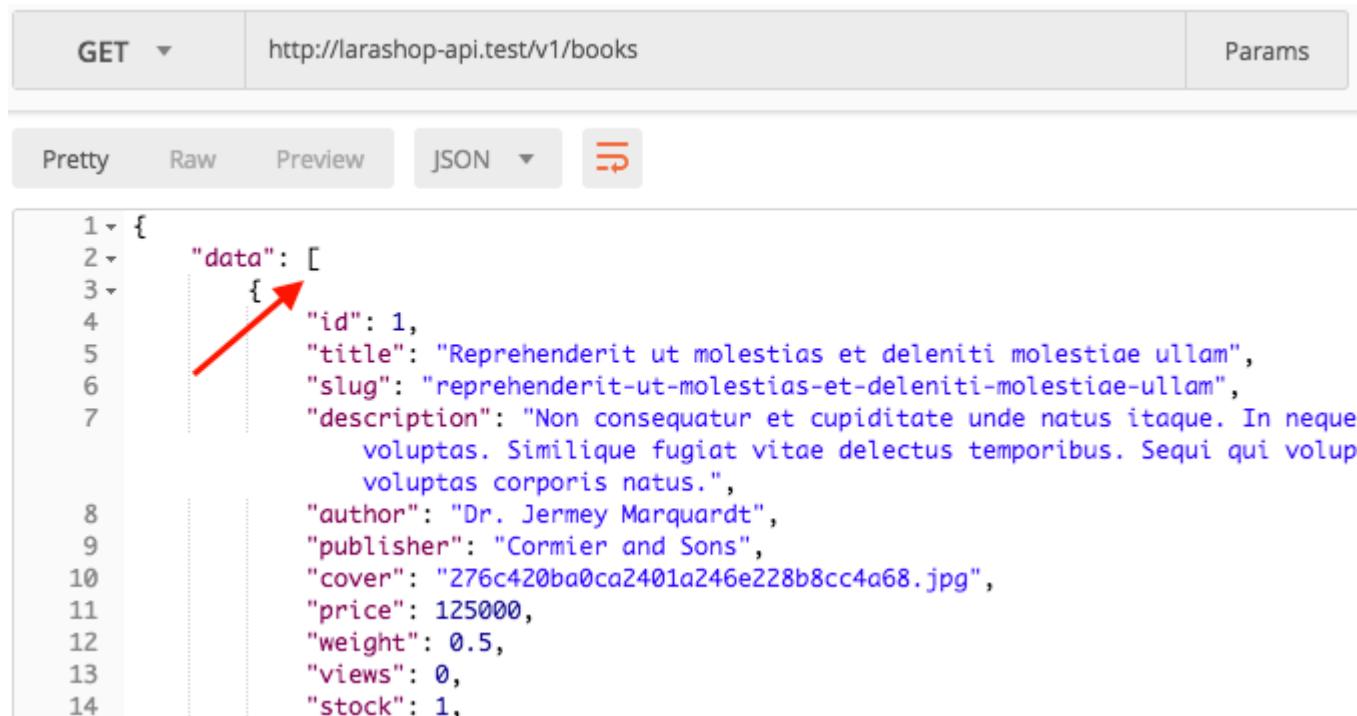
```

1 public function index(){
2     $books = new BookCollectionResource(Book::get());
3     return $books;
4 }

```

Boleh Book::get() atau Book::all(). Jangan lupa tambahkan kode use App\Http\Resources\Books as BookCollectionResource di bagian atas controller.

Mari kita lihat hasilnya:



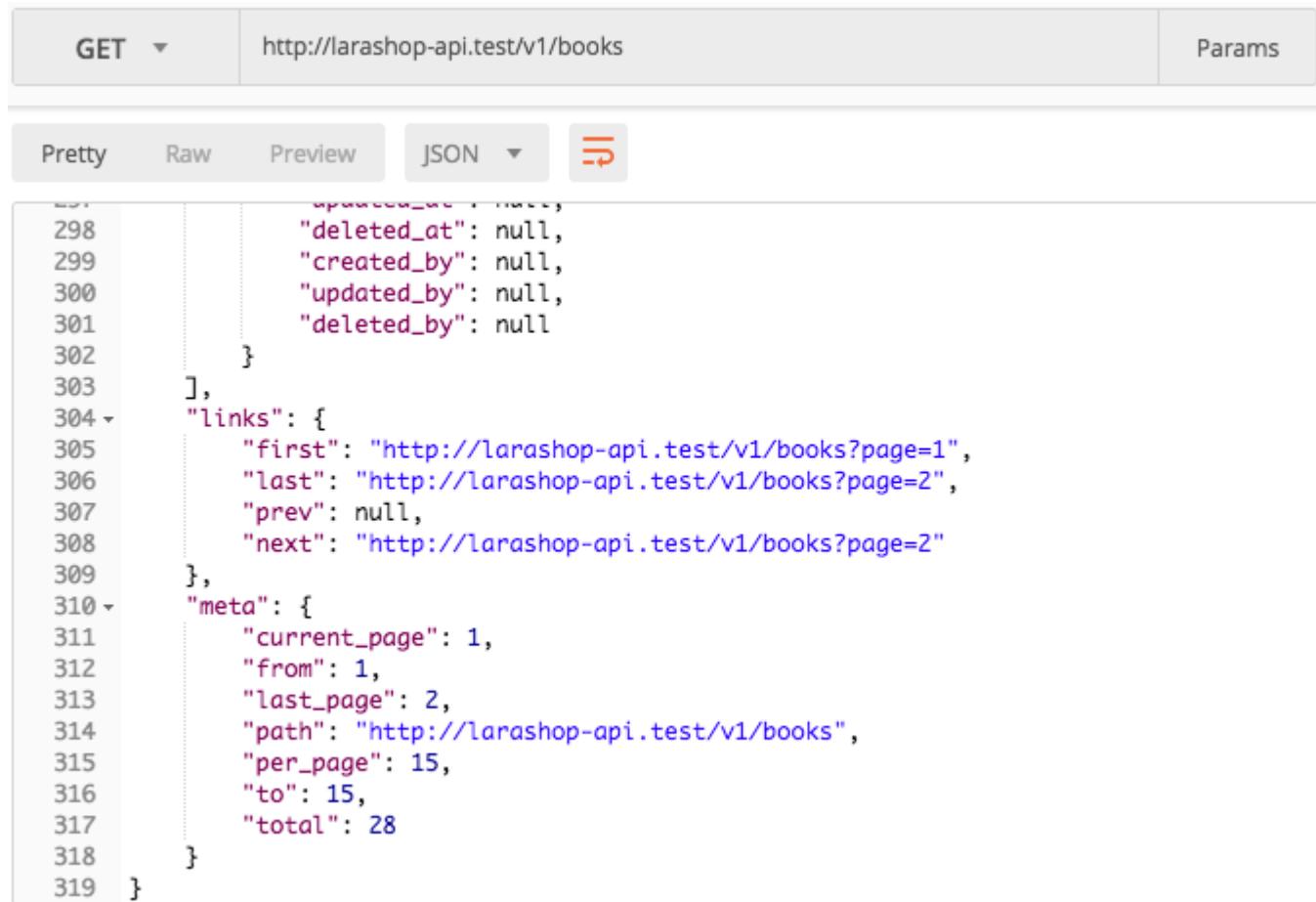
```

1 { "data": [
2 { "id": 1,
3 "title": "Reprehenderit ut molestias et deleniti molestiae ullam",
4 "slug": "reprehenderit-ut-molestias-et-deleniti-molestiae-ullam",
5 "description": "Non consequatur et cupiditate unde natus itaque. In neque
6         voluptas. Similique fugiat vitae delectus temporibus. Sequi qui volup
7         voluptas corporis natus.",
8 "author": "Dr. Jeremy Marquardt",
9 "publisher": "Cormier and Sons",
10 "cover": "276c420ba0ca2401a246e228b8cc4a68.jpg",
11 "price": 125000,
12 "weight": 0.5,
13 "views": 0,
14 "stock": 1,

```

Secara umum hampir sama dengan single data, yang membedakan hanya tanda array yang merupakan array dari data buku.

Nah, tidak berhenti sampai di sini saja, namun resource juga mendukung pagination. Caranya dengan mengubah perintah `Book::get()` menjadi `Book::paginate()`. Maka jika kita coba akses, pada response akan ditambahkan dua atribut baru yaitu links dan meta.



```

298     "deleted_at": null,
299     "created_by": null,
300     "updated_by": null,
301     "deleted_by": null
302   }
303 ],
304   "links": {
305     "first": "http://larashop-api.test/v1/books?page=1",
306     "last": "http://larashop-api.test/v1/books?page=2",
307     "prev": null,
308     "next": "http://larashop-api.test/v1/books?page=2"
309   },
310   "meta": {
311     "current_page": 1,
312     "from": 1,
313     "last_page": 2,
314     "path": "http://larashop-api.test/v1/books",
315     "per_page": 15,
316     "to": 15,
317     "total": 28
318   }
319 }

```

Dari info tersebut, pengguna API kita akan mengetahui informasi tentang paging dari resource yang kita sediakan.

Fitur pagination ini sebenarnya bukan fitur dari resource, melainkan fitur dari QueryBuilder dan Eloquent. Terdapat dua macam fungsi pagination yaitu `paginate()` dan `simplePaginate()`. Perbedaannya, `simplePaginate()` hanya akan menampilkan informasi halaman sebelum dan sesudah saja.

Sebagai contoh, jika kita ingin membatasi record yang muncul pada setiap halamannya hanya 5 record (defaultnya 15 record), maka tambahkan parameter jumlah record yang muncul dalam fungsi paginate.

```
1 | Book::paginate(5)
```

Gimana? mudah sekali bukan?

Nah sejak versi 5.6, lagi-lagi Laravel memperkenalkan fitur yang ciamik untuk membuat web service, yaitu apa yang disebut dengan `Resource Controller`.

Apa itu? Resource Controller adalah controller berbasis resource yang mempercepat kita dalam membuat kerangka controller yang mendukung Create Read Update dan Delete, hanya dengan satu baris perintah, Wow!!.

Caranya, jalankan perintah

```
1 | php artisan make:controller CategoryController --resource
```

Atau bisa juga kita definisikan spesifik modelnya.

```
1 | php artisan make:controller CategoryController --resource --model=Category
```

Hasilnya sebagai berikut. (komentar dan whitespace sudah penulis dihapus):

```

1 | <?php
2 | namespace App\Http\Controllers;
3 | use App\Category;
4 | use Illuminate\Http\Request;
5 | class CategoryController extends Controller
6 |
7 |     public function index()
8 |     {
9 |         //
10|     }
11|
12|     public function create()
13|     {
14|         //
15|     }
16|
17|     public function store(Request $request)
18|     {
19|         //
20|     }
21|
22|     public function show(Category $category)
23|     {
24|         //
25|     }
26|
27|     public function edit(Category $category)
28|     {
29|         //
30|     }
31|
32|     public function update(Request $request, Category $category)
33|     {
34|         //
35|     }
36|
37|     public function destroy(Category $category)
38|     {
39|         //
40|     }
41| }
```

Yap, kode di atas fungsinya masih kosong, artinya kita perlu koding sendiri.. (agak kecewa sih penulis juga). Tapi apa yang membedakan?

Dengan menggunakan resource controller ini maka konfigurasi routing pada `api.php` lebih sederhana tanpa perlu mendaftarkan satu persatu fungsi pada controller. Cukup dengan menggunakan sintaks berikut.

```
1 | Route::resource('categories', 'CategoryController');
```

Atau untuk mendaftarkan lebih dari satu controller resource, kita bisa gunakan format array.

```
1 | Route::resources([
2 |     'categories' => 'CategoryController',
3 |     'books' => 'BookController',
4 | ]);
```

Untuk melihat daftar routing yang telah kita definisikan, laravel menyediakan perintah `php artisan route:list`.

Verb	URI	Action	Route Name
GET	/categories	index	categories.index
POST	/categories	store	categories.store
GET	/categories/{category}	show	categories.show
GET	/categories/{category}/edit	edit	categories.edit
PUT/PATCH	/categories/{category}	update	categories.update
DELETE	/categories/{category}	destroy	categories.destroy

Ketika menggunakan resource controller, kita bisa juga menggunakan method tertentu saja dengan menggunakan fungsi `only` atau `except`.

```
1 | Route::resource('categories', 'CategoryController')->only([
2 |     'index', 'show'
3 | ]);
4 |
5 | // atau
6 |
7 | Route::resource('categories', 'CategoryController')->except([
8 |     'create', 'store', 'update', 'destroy'
9 | ]);
```

Sebenarnya method `create` dan `update` itu ditujukan untuk menampilkan form `create` atau `update` data pada aplikasi web, sedangkan untuk web service, kita bisa langsung gunakan method `store` saja. Nah alih-alih kita menggunakan fungsi `only` atau `except` untuk mengecualikan kedua method di atas, Laravel menyediakan fungsi `apiResource` pada route untuk melakukan hal tersebut.

```
1 | Route::apiResources([
2 |     'categories' => 'CategoryController',
3 |     'books' => 'BookController',
4 | ])
```

Meskipun pada controller terdapat method `create` dan `update` namun dengan menggunakan `apiResource` ini maka method tersebut tidak akan digunakan. Namun bisa juga kita menghilangkan dua method tersebut dari controller, caranya pada saat membuat controller gunakan perintah berikut.

```
1 | php artisan make:controller CategoryController --api
```

Handling Error

Pada bagian sebelumnya, error yang muncul tidak ramah dengan web service yang umumnya hanya dapat membaca response format JSON. Oleh karena itu error pada web service ini perlu penanganan khusus. Penanganan error ini dapat kita jumpai pada file `app/Exceptions/Handler.php`.

Response 404 atau resource tidak ditemukan merupakan error yang seringkali muncul. Hal ini bisa disebabkan karena halaman tersebut tidak ditemukan, atau URL yang diberikan salah, atau resource memang tidak ada.

Terdapat lima class terkait hal ini yaitu

- `Illuminate\Database\Eloquent\ModelNotFoundException;`
- `Symfony\Component\HttpFoundation\Exception\NotFoundHttpException;`
- `Symfony\Component\HttpFoundation\Exception\MethodNotAllowedHttpException;`
- `Illuminate\Validation\ValidationException;`
- `Illuminate\Database\QueryException;`

Untuk menangani error tersebut buka file `Handler.php`, update fungsi render menjadi sebagai berikut.

```
1 | public function render($request, Exception $exception)
2 | {
3 |     // baca konfigurasi apakah aplikasi menggunakan mode production atau
4 |     // development
5 |     $debug = config('app.debug');
6 |     $message = '';
7 |     $status_code = 500;
8 |     // cek jika eksepsinya dikarenakan model tidak ditemukan
9 |     if ($exception instanceof ModelNotFoundException) {
10 |         $message = 'Resource is not found';
11 |         $status_code = 404;
12 |     }
13 |     // cek jika eksepsinya dikarenakan resource tidak ditemukan
14 |     elseif ($exception instanceof NotFoundHttpException) {
15 |         $message = 'Endpoint is not found';
16 |         $status_code = 404;
17 |     }
18 |     // cek jika eksepsinya dikarenakan method tidak diizinkan
19 |     elseif ($exception instanceof MethodNotAllowedHttpException) {
20 |         $message = 'Method is not allowed';
21 |         $status_code = 405;
22 |     }
23 |     // cek jika eksepsinya dikarenakan kegagalan validasi
24 |     else if ($exception instanceof ValidationException) {
25 |         $validationErrors = $exception->validator->errors()-
26 | >getMessages();
27 |         $validationErrors = array_map(function($error) {
28 |             return array_map(function($message) {
29 |                 return $message;
30 |             }, $error);
31 |         }, $validationErrors);
32 |         $message = $validationErrors;
33 |         $status_code = 405;
```

```

34 }
35 // cek jika eksepsinya dikarenakan kegagalan query
36 else if ($exception instanceof QueryException) {
37     if ($debug) {
38         $message = $exception->getMessage();
39     } else {
40         $message = 'Query failed to execute';
41     }
42     $status_code = 500;
43 }
44 $rendered = parent::render($request, $exception);
45 $status_code = $rendered->getStatusCode();
46 if (empty($message)) {
47     $message = $exception->getMessage();
48 }
49 $errors = [];
50 if ($debug) {
51     $errors['exception'] = get_class($exception);
52     $errors['trace'] = explode("\n", $exception->getTraceAsString());
53 }
54 return response()->json([
55     'status' => 'error',
56     'message' => $message,
57     'data' => null,
58     'errors' => $errors,
59 ], $status_code);
}

```

Jangan lupa, sebelumnya kita harus **use** terlebih dahulu lima class tersebut.

```

1 use Illuminate\Database\Eloquent\ModelNotFoundException;
2 use Symfony\Component\HttpKernel\Exception\NotFoundHttpException;
3 use Symfony\Component\HttpKernel\Exception\MethodNotAllowedHttpException;
4 use Illuminate\Validation\ValidationException;
5 use Illuminate\Database\QueryException;

```

Kode diatas akan mengembalikan error dalam format JSON apabila model atau page tidak ditemukan. Mari kita ujicoba dengan mengakses URL yang memang tidak ada.

GET ▾ http://larashop-api.test/v1/url-palsu Params

Pretty Raw Preview

```
{"data":{"message":"Endpoint not found","status_code":404}}
```

Adapun untuk error terkait autentication maka kita harus membuat fungsi `unauthenticated()` tersendiri dengan parameter class `Illuminate\Auth\AuthenticationException`.

```

1 protected function unauthenticated($request, AuthenticationException
2 $exception)
3 {

```

```

4     return response()->json([
5         'status'      => 'error',
6         'message'    => 'Unauthenticated',
7         'data'        => null
8     ], 401);
}

```

Hasilnya.

GET	http://larashop-api.test/user	Params
Pretty	Raw	Preview

```
{"data": {"message": "Unauthenticated.", "status_code": 400}}
```

Authentication

Pada bagian sebelumnya, kita telah membuat web service yang resource-nya bisa diakses oleh siapapun yang mengetahui URL endpoint, cara tersebut cocok untuk resource yang bebas diakses oleh publik, umumnya bersifat readonly saja.

Nah, untuk resource yang sifatnya bisa dimodifikasi maka kita perlu menggunakan mekanisme tertentu agar hanya user yang berwenang saja yang bisa melakukan modifikasi tersebut, inilah yang disebut dengan authentication. Laravel membuat implementasi authentication menjadi sangat sederhana. Konfigurasi dari authentication dapat kita jumpai pada `config/auth.php`, di sana terdapat dua pengaturan yaitu "guards" dan "providers".

Guards mengatur bagaimana user di cek authentication-nya pada setiap request. Sedangkan providers mengatur bagaimana atau dimana data user tersebut disimpan.

Konfigurasi

Untuk melakukan konfigurasi, caranya: edit file `config/auth.php` sebagai berikut.

```

1  'defaults' => [
2      'guard' => 'api', // <== update ini
3      'passwords' => 'users',
4  ],
5  'guards' => [
6      'web' => [
7          'driver' => 'session',
8          'provider' => 'users',
9      ],
10
11     'api' => [
12         'driver' => 'token', // <== pastikan ini
13         'provider' => 'users',
14     ],
15 ]

```

Pada web service api, driver yang digunakan adalah token karena stateless (tanpa session). Sedangkan providernya adalah users (model eloquent `App\Users`).

Catatan: abaikan guards untuk web, karena pada materi ini kita fokus ke API web service

Untuk menggunakan fungsi-fungsi authentication, Laravel menyediakan class Illuminate\Support\Facades\Auth.

Get Authenticate User

Untuk mengambil objek user yang sudah login kita bisa gunakan kode berikut.

```
1 $user = Auth::user();
2 // Get user id
3 $id = Auth::id();
```

Check Authenticate User

Untuk mengecek apakah user sudah login atau belum maka bisa gunakan kode berikut.

```
1 if (Auth::check()) {
2     // The user is logged in...
3 }
```

Protect Routing

Untuk memproteksi route tertentu yang hanya boleh diakses oleh user yang sudah login, maka kita bisa gunakan middleware berikut.

```
1 Route::get('profile', function () {
2     // Hanya untuk user yang sudah login
3 })->middleware('auth:api');
```

Authentication Mechanism

Ada berbagai mekanisme authentication web service yang bisa kita terapkan pada Laravel, misalnya yang sederhana menggunakan token, atau yang lebih kompleks menggunakan oauth2.

Token Field

Kembali ke pemahaman awal tentang web service yang bersifat stateless yaitu tidak ada penyimpanan session atau penanda user di server sehingga setiap request akan independen terhadap request lainnya. Contoh: pada request pertama kita berhasil login, maka pada request kedua tidak otomatis dianggap sebagai user yang sudah login. Oleh karena itu kita perlu sesuatu yang menandakan bahwa request kedua tadi sudah login. Penanda itulah yang disebut dengan token.

Jadi, ketika user melakukan login pada request pertama, maka akan diresponse dengan token unik oleh Laravel. Token yang berfungsi sebagai penanda user tersebut akan disertakan setiap kali melakukan request berikutnya, sehingga pada request berikutnya itu Laravel tau bahwa request tersebut berasal dari authenticate user.

Untuk mengimplementasikannya, kita perlu menambahkan satu field yaitu `api_token` dengan tipe data string dan panjang 60 karakter pada tabel users (jumlah karakternya bebas saja sebenarnya). Kita bisa tambahkan manual atau menggunakan migration.

Jalankan perintah.

```
1 php artisan make:migration --table=users adds_api_token_to_users_table
```

Kemudian pada file migration kita tambahkan kode berikut.

```

1 public function up()
2 {
3     Schema::table('users', function (Blueprint $table) {
4         $table->string('api_token', 60)->unique()->nullable();
5     });
6 }
7
8 public function down()
9 {
10    Schema::table('users', function (Blueprint $table) {
11        $table->dropColumn(['api_token']);
12    });
13}

```

Jalankan migration

```
1 php artisan migrate
```

```

root@8a48a85542c9:/var/www/larashop-api# cd larashop-api
root@8a48a85542c9:/var/www/larashop-api# php artisan make:migration --table=users adds_api_token_to_users_table
Created Migration: 2018_08_18_211244_adds_api_token_to_users_table
root@8a48a85542c9:/var/www/larashop-api# php artisan migrate
Migrating: 2018_08_18_211244_adds_api_token_to_users_table
Migrated: 2018_08_18_211244_adds_api_token_to_users_table
root@8a48a85542c9:/var/www/larashop-api#

```

Pada model User, kita akan tambahkan fungsi untuk menggenerate token.

```

1 public function generateToken()
2 {
3     $this->api_token = str_random(60);
4     $this->save();
5     return $this->api_token;
6 }

```

Kode di atas menggunakan helpers `str_random(60)` untuk menggenerate teks random sebanyak 60 karakter dan disimpan pada field `api_token` yang telah kita buat sebelumnya.

Kemudian untuk menangani authentication, kita akan buat satu controller baru yaitu `AuthController`. Sebenarnya Laravel sudah menyediakan controller untuk authentication pada direktori `App\Http\Controllers\Auth`, namun itu diperuntukkan bagi web normal, sedangkan untuk web service sebaiknya kita buat sendiri.

```
1 php artisan make:controller AuthController
```

Pada `AuthController` ini kita akan buat fungsi login, register dan logout.

```

1 namespace App\Http\Controllers;
2 use Illuminate\Http\Request;
3 class AuthController extends Controller
4 {

```

```

5  public function login(Request $request)
6  {
7  }
8
9
10 public function register(Request $request)
11 {
12 }
13
14
15 public function logout(Request $request)
16 {
17 }
18
19 }
```

Pada fungsi login, kita tidak bisa menggunakan fungsi bawaan Laravel yaitu Auth::attempt() untuk mengecek data login user sebab fungsi tersebut hanya untuk web normal yang menggunakan session. Oleh karena itu kita akan menggunakan query biasa untuk mengecek apakah user tersebut terdaftar atau tidak.

Pada fungsi login ini, kita menggunakan dua class yaitu Illuminate\Support\Facades\Hash dan App\User.

```

1 use Hash;
2 use App\User;
```

Pertama kali, data login yaitu email dan password akan divalidasi. Kemudian jika valid maka data tersebut digunakan untuk mencari data user berdasarkan emailnya. Jika data user ditemukan maka password dari user di cek dengan menggunakan fungsi Hash::check. Jika password cocok maka token baru akan digenerate dan akhirnya mengembalikan respon data user yang login menggunakan fungsi fungsi toArray().

```

1 public function login(Request $request)
2 {
3     $user = User::where('email', '=', $request->email)->firstOrFail();
4     $status = "error";
5     $message = "";
6     $data = null;
7     $code = 401;
8     if($user){
9         // jika hasil hash dari password yang diinput user sama dengan
10        password di database user maka
11         if (Hash::check($request->password, $user->password)) {
12             // generate token
13             $user->generateToken();
14             $status = 'success';
15             $message = 'Login sukses';
16             // tampilkan data user menggunakan method toArray
17             $data = $user->toArray();
18             $code = 200;
19         }
20     else{
```

```

1          $message = "Login gagal, password salah";
2      }
3  }
4 else{
5     $message = "Login gagal, username salah";
6 }
7
8 return response()->json([
9     'status' => $status,
10    'message' => $message,
11    'data' => $data
12 ], $code);
13

```

Catatan: jika pernah menggunakan Laravel sebelumnya mungkin kamu bertanya "mengapa tidak menggunakan fungsi login & attempt bawaan laravel guards?", yaps itulah, sayangnya fungsi fungsi itu hanya available untuk Laravel versi web bukan versi API. 😞. Oleh karena itu terpaksa kita harus bikin manual.

Lalu pada routing tambahkan route login.

```

1 Route::prefix('v1')->group(function () {
2     // ...
3     Route::post('login', 'AuthController@login');
4
5     // tambahkan sekalian untuk register dan logout :
6     Route::post('register', 'AuthController@register');
7     Route::post('logout', 'AuthController@logout');
8 });

```

Dengan menggunakan postman, mari kita coba login menggunakan URL <http://larashop-api.test/v1/login> dengan method POST serta parameter body email dan password.

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> email	mcglynn.rey@example.org	
<input checked="" type="checkbox"/> password	123456	

Hasilnya sebagai berikut.

Pretty Raw Preview JSON 

```

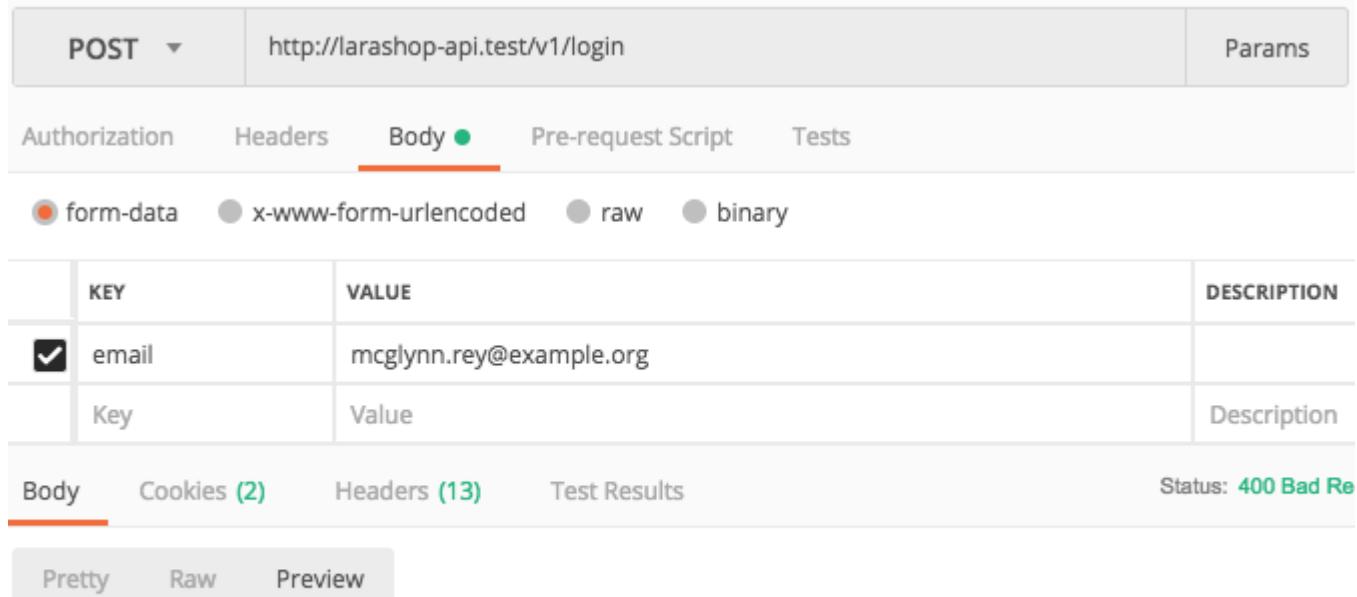
1  {
2    "data": {
3      "id": 3,
4      "name": "Timmothy Howell",
5      "email": "mcglynn.rey@example.org",
6      "created_at": "2018-08-07 19:51:32",
7      "updated_at": "2018-08-18 22:15:17",
8      "username": "ole.boyle",
9      "roles": ["CUSTOMER"],
10     "address": null,
11     "city_id": null,
12     "province_id": null,
13     "phone": null,
14     "avatar": "2c923b1d8e436b1fec19ef48e3edab4.jpg",
15     "status": "ACTIVE",
16     "api_token": "mM0nBNctiyrFKMrzAoBr4pZu0MaTML6pYRCjdCkqqkEZgxiP82y3PSFQw8gJ"
17   }
18 }
```

Supaya lebih straightforward maka kita bisa validasi terlebih dahulu request dari user. Caranya, pada fungsi login, kita bisa tambahkan kode berikut di awal fungsi .

```

1 $this->validate($request, [
2   'email' => 'required',
3   'password' => 'required',
4 ]);
```

Kode di atas akan melakukan validasi di mana email dan password harus diisi. Mari kita coba dengan mengirimkan parameter body email saja tanpa password maka akan muncul error.



The screenshot shows the Postman interface with a failed login attempt. The request is a POST to `http://larashop-api.test/v1/login`. The 'Body' tab is selected, showing a form-data key 'email' with value 'mcglynn.rey@example.org'. The response status is 400 Bad Request, and the JSON body is `{"data": {"message": "The given data was invalid.", "status_code": 400}}`.

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> email	mcglynn.rey@example.org	
Key	Value	Description

`{"data": {"message": "The given data was invalid.", "status_code": 400}}`

Mari kita lihat hasilnya, ketika password salah.

POST ▾ http://larashop-api.test/v1/login Params

Authorization Headers Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> email	mcglynn.rey@example.org	
<input checked="" type="checkbox"/> password	123	
Key	Value	Description

Body Cookies (2) Headers (13) Test Results Status: 400 Bad Request

Pretty Raw Preview

```
{"data": {"message": "Username atau password salah", "status_code": 400}}
```

Setelah fungsi login selesai, maka kita akan membuat fungsi untuk register. Sedikit berbeda dengan fungsi login dimana pada fungsi register ini kita akan menggunakan class `Validator` (`Illuminate\Support\Facades\Validator`) karena kita mengharapkan respon baliknya. Berikut ini kerangkanya, dan kamu bisa lihat bagaimana penggunaan Validatornya.

```

1 public function register(Request $request)
2 {
3     $validator = Validator::make($request->all(), [
4         'name' => 'required|string|max:255', // name harus diisi teks
5         // dengan panjang maksimal 255
6         'email' => 'required|string|email|max:255|unique:users', // email harus unik pada tabel users
7         'password' => 'required|string|min:6', // password minimal 6 karakter
8     ]);
9     if ($validator->fails()) { // fungsi untuk ngecek apakah validasi
10        gagal
11        // validasi gagal
12    }
13    else{
14        // validasi sukses
15    }
16 }
```

Ketika validasi gagal maka kita bisa tampilkan errornya dengan cara berikut.

```

1 if ($validator->fails()) {
2     $errors = $validator->errors();
3     return response()->json([
4         'data' => [
5             'message' => $errors,
6         ]
6     ])
6 }
```

```

7     ], 400);
8 }
```

Mari kita lihat hasilnya.

The screenshot shows a POST request to `http://larashop-api.test/v1/register`. The 'Body' tab is selected, showing form-data fields: email (hafid@gmail.org), password (1234), and name (Hafid). The response status is 400 Bad Request, with the message: {"data": {"message": {"password": ["The password must be at least 6 characters."]}, "status_code": 400}}.

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> email	hafid@gmail.org	
<input checked="" type="checkbox"/> password	1234	
<input checked="" type="checkbox"/> name	Hafid	
Key	Value	Description

{"data": {"message": {"password": ["The password must be at least 6 characters."]}, "status_code": 400}}

Jika validasi berhasil maka kita bisa create users baru dan jika berhasil maka loginkan user tersebut. Untuk password pada kode sebelumnya kita menggunakan fungsi bcrypt, namun jika mengikuti best practice Laravel maka kita bisa gunakan fungsi make yang terdapat pada class Hash (Illuminate\Support\Facades\Hash).

```

1 $validator = Validator::make($request->all(), [
2     'name' => 'required|string|max:255',
3     'email' => 'required|string|email|max:255|unique:users',
4     'password' => 'required|string|min:6',
5 ]);
6
7     $status = "error";
8     $message = "";
9     $data = null;
10    $code = 400;
11    if ($validator->fails()) {
12        $errors = $validator->errors();
13        $message = $errors;
14    }
15    else{
16        $user = \App\User::create([
17            'name' => $request->name,
18            'email' => $request->email,
19            'password' => Hash::make($request->password),
20            'roles' => json_encode(['CUSTOMER']),
21        ]);
22    }
23 }
```

```

22 if($user){
23     // Auth::login($user);
24     $user->generateToken();
25     $status = "success";
26     $message = "register successfully";
27     $data = $user->toArray();
28     $code = 200;
29 }
30 else{
31     $message = 'register failed';
32 }
33 }
34
35 return response()->json([
36     'status' => $status,
37     'message' => $message,
38     'data' => $data
39 ], $code);

```

Berikut ini tampilan jika, register berhasil.

The screenshot shows a Postman interface. The method is set to POST, the URL is <http://larashop-api.test/v1/register>, and the 'Send' button is highlighted. Below the URL, there's a table for parameters:

	KEY	VALUE	DESCRIPTION	...
<input checked="" type="checkbox"/>	email	hafid@gmail.org		
<input checked="" type="checkbox"/>	password	123456		
<input checked="" type="checkbox"/>	name	Hafid		
	Key	Value	Description	

Below the table, the 'Body' tab is selected, showing the JSON response:

```
{"data":{"name":"Hafid","email":"hafid@gmail.org","roles":["CUSTOMER"],"updated_at":"2018-08-18 23:59:17","created_at":"2018-08-18 23:59:17","id":6,"api_token":"um4PikXSve4IH5Qi5HuTGX4YVJOU1W6wlVGtUQ7KvDMUmexxRbMGDoXwxpMb"}}
```

At the bottom, the status is shown as 200 OK and the time as 140 ms.

Baik, setelah login dan register, maka berikutnya bagaimana caranya kita mengakses suatu resource yang diperuntukkan bagi authenticate user?

Caranya adalah pada setiap request terhadap authenticate resource maka sertakan bearer token atau dalam hal ini api_token pada header.

Sebagai contoh, kita akan menggunakan routing yang telah by default sudah ada di file api.php yaitu

```

1 Route::middleware('auth:api')->get('/user', function (Request $request) {
2     return $request->user();
3 });

```

Di mana route /user di atas dilindungi oleh middleware auth:api. Untuk mengakses route tersebut, pertama, pada tab authorization, pilih type Bearer Token, kemudian masukkan nilai dari api_token pada kolom Token.

The authorization header will be automatically generated when you send the request. [Learn more about](#)

api_token

um4PikXSve4IH5Qi5HuTGX4YVJOU1W6wlVGtUQ7KvDMUmexxR...

Jika token benar maka akan menampilkan data user sebagaimana definisi routing di atas.

KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets
Authorization	Bearer um4PikXSve4IH5Qi5HuTGX4YVJOU1W6wlVGtUQ7KvDMUmexxRbMG...				
Key	Value	Description			

Status: 200 OK Time: 78 ms Size: 577 B

```
{"id":6,"name":"Hafid","email":"hafid@gmail.org","created_at":"2018-08-18 23:59:17","updated_at":"2018-08-18 23:59:17","roles":["CUSTOMER"],"address":null,"city_id":null,"province_id":null,"phone":null,"avatar":null,"status":"ACTIVE","api_token":"um4F
```

Jika gagal maka akan menampilkan json message 'Unauthenticated'.

KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets
Authorization	Bearer um4PikXSve4IH5Qi5HuTGX4YVJOU1W6wlVGtUQ7KvDMUmexxRbMG...				
Key	Value	Description			

Status: 401 Unauthorized Time: 77 ms Size: 331 B

```
{"data":{"message":"Unauthenticated","status_code":401}}
```

Terakhir pada AuthController, kita akan selesaikan fungsi logout. Tentu yang logout adalah yang sudah login. Karenanya, routing logout juga termasuk routing dilindungi authentication.

```

1 public function logout(Request $request)
2 {
3     $user = Auth::user();
4     if ($user) {
5         $user->api_token = null;
6         $user->save();
7     }
8     return response()->json([
9         'status' => 'success',
10        'message' => 'logout berhasil',
11        'data' => null
12    );
13 }
```

```
12     ], 200);
13 }
```

Untuk routing, sekalian kita rapikan. Di dalam routing dengan prefix v1, kita bagi menjadi dua yaitu routing yang sifatnya public artinya siapapun boleh mengakses resourceny dan routing yang sifatnya private atau hanya user yang berwenang saja yang boleh mengakses resourceny. Pada routing private ini kita kelompokkan menggunakan middleware auth:api.

```
1 Route::prefix('v1')->group(function () {
2     // public
3     Route::post('login', 'AuthController@login');
4     Route::post('register', 'AuthController@register');
5     // ...
6
7     // private
8     Route::middleware('auth:api')->group(function () {
9         Route::post('logout', 'AuthController@logout');
10        //...
11    });
12});
```

Catatan: silakan hapus dulu definisi *routing-routing* terdahulu hasil latihan kita.

Waktunya mencoba, silakan akses routing /logout dengan header bearer token.

KEY	VALUE	DESCRIPTION
Authorization	Bearer um4Pil0Sve4IH5Qi5HuTGX4YVJOU1W6wlVGtUQ7KvDMUmexxRbMGD...	
Key	Value	Description

```
{"data": {"message": "logout berhasil", "status_code": 200}}
```

Kesimpulan

Web service merupakan standard yang digunakan untuk pertukaran data antar aplikasi atau sistem berbasis web. Standard ini diperlukan karena masing-masing aplikasi bisa jadi memiliki format data yang berbeda, ditulis dengan bahasa pemrograman yang berbeda, dan berjalan pada *platform* berbeda. Dengan adanya standard tersebut akan memungkinkan dua aplikasi berinteraksi satu sama lain dengan baik.

Pada bab ini kita menggunakan Laravel sebagai framework PHP untuk membantu kita dalam membangun web service. Beberapa hal utama yang telah kita pelajari yaitu eloquent yang merupakan object relational models untuk memudahkan kita berinteraksi dengan database, routing untuk menangani URL dari user, serta middleware untuk grouping URL, versioning dan authentication.

Pada bab selanjutnya yang merupakan puncak dari bab pada buku ini kita akan membahas hampir setiap langkah dalam pengembangan aplikasi toko buku berbasis mobile web.

Meski berat, tetap semangat!

Finishing Project

Intro

Pada bab ini kita akan menyelesaikan projek studi kasus yang kita angkat yaitu toko buku berbasis mobile web. Karena mobile web maka user interface yang kita buat akan fokus untuk pengguna mobile. Adapun framework yang akan kita gunakan untuk menangani user interface adalah Vuetify.

Source code lengkap dari studi kasus ini dapat kamu jumpai pada tautan berikut:

- <https://github.com/laravel-vue-book/vueshop>
- <https://github.com/laravel-vue-book/larashop-api>

Catatan: vueshop merupakan kode projek Vue-nya, sedangkan larashop-api adalah kode web service Laravel-nya. Sebaiknya kamu ikuti panduan pada bab ini dulu sebelum merujuk ke source code lengkapnya supaya pemahaman kamu lebih baik lagi.

Konstanta Global

Terdapat sedikit perbedaan antara Vue CLI versi 3 dan versi sebelumnya terkait dengan pendefinisian konstanta yang akan digunakan secara global pada aplikasi. Misalnya konstanta URL web service, bisa kita bedakan antara production dan development. Vue menggunakan file `.env` untuk mendefinisikan konstanta global tersebut (sebagaimana Laravel juga menggunakan file tersebut).

Selengkapnya, kamu bisa baca pada tautan ini <https://cli.vuejs.org/guide/mode-and-env.html#modes>

Terdapat beberapa jenis varian file `.env` sesuai dengan peruntukannya.

- `.env` akan diaplikasikan ke semua kondisi
- `.env.development` akan diaplikasikan ke kondisi development saja `npm run serve`
- `.env.production` akan diaplikasikan ke kondisi production saja `npm run build`
- `.env.*.local` tidak di push ke github

Merujuk hal tersebut, maka pada kasus ini, untuk kebutuhan pengembangan aplikasi (development) kita perlu membuat sebuah file `.env.development.local` pada directory root Vue yang isinya adalah konstanta yang akan kita gunakan pada aplikasi. Adapun penulisan nama konstanta harus diawali dengan `VUE_APP_` misalnya sebagai berikut.

```
1 VUE_APP_NAME=Vueshop
2 VUE_APP_BACKEND_URL=http://larashop-api.test
3 VUE_APP_API_URL=http://larashop-api.test
```

Konstanta pada file `.env` ini dapat kita akses dan gunakan pada semua component menggunakan format perintah `process.env.NAMA_VARIABEL`. Untuk mengujinya, kita akan coba tambahkan kode berikut pada hook mounted di component manapun pada Vue.

```
1 mounted(){
2     console.log(process.env)
3 }
```

Lalu jalankan perintah `npm run serve` maka hasilnya akan sebagai berikut.

```
[HMR] Waiting for update signal from log.js?1afdf:24
WDS...
App.vue?234e:66
{VUE_APP_NAME: "Vueshop", VUE_APP_BACKEND_URL: "http://larashop-api.test", VUE_APP_API_URL: "http://larashop-api.test", NODE_ENV: "development", BASE_URL: "/"
" } ①
  BASE_URL: "/"
  NODE_ENV: "development"
  VUE_APP_API_URL: "http://larashop-api.test"
  VUE_APP_BACKEND_URL: "http://larashop-api.test"
  VUE_APP_NAME: "Vueshop"
▶ __proto__: Object
```

Demikian juga dengan environment production, kita bisa tambahkan file `env.production` dengan nilai yang berbeda, misalnya sebagai berikut.

```
1 VUE_APP_NAME=Vueshop
2 VUE_APP_BACKEND_URL=http://api.larashop.id
3 VUE_APP_API_URL=http://api.larashop.id
```

Kemudian jalankan perintah `npm run build` dan lihat hasilnya.

Layout Aplikasi

Layout merupakan kerangka utama tampilan aplikasi yang akan jadi acuan untuk membuat halaman-halaman lain pada aplikasi.

Secara umum, layout pada aplikasi kita ini dibagi menjadi 4 bagian.

1. Header pada posisi paling atas, juga berperan sebagai navigasi utama.
2. Konten utama pada posisi tengah di bawah menu utama,
3. Footer pada posisi paling bawah.
4. Sidebar pada posisi overlay, defaultnya pada layar yang kecil akan tersembunyi di sisi kiri yang bisa ditampilkan melalui menu di header.

Pada Vuetify terdapat konsep untuk mengimplementasikan layout ini, hal itu bisa kamu jumpai pada [tautan ini](#). Pada tautan tersebut dijelaskan bahwa sebuah layout aplikasi vuetify harus dibangun di atas component utama `v-app` sebagai parent dari semua component vuetify. `v-app` juga berperan untuk menjamin bahwa aplikasi vuetify kita memiliki kompatibilitas antar browser.

Pada konsep scaffolding Vue CLI, kode untuk layout aplikasi bisa kita jumpai pada file `src/App.vue`, yaitu sebuah component yang akan dimuat pertama kali. Merujuk pada pembagian layout di atas dan konsep layout pada vuetify maka kode pada file `App.vue` akan tampak menjadi sebagai berikut.

```
1 <template>
2   <v-app>
3
4     <v-app-bar app>
5       Header
6     </v-app-bar>
7
8     <v-navigation-drawer app>
9       Side Menu
10      </v-navigation-drawer>
11
12      <v-content>
13        <v-container fluid>
```

```

14      <!-- jika menggunakan vue-router -->
15      <router-view></router-view>
16      </v-container>
17      </v-content>
18
19      <v-footer app>
20          Footer
21      </v-footer>
22
23      </v-app>
24  </template>
25  <script>
26  export default {
27      name: 'App'
28  }
29  </script>

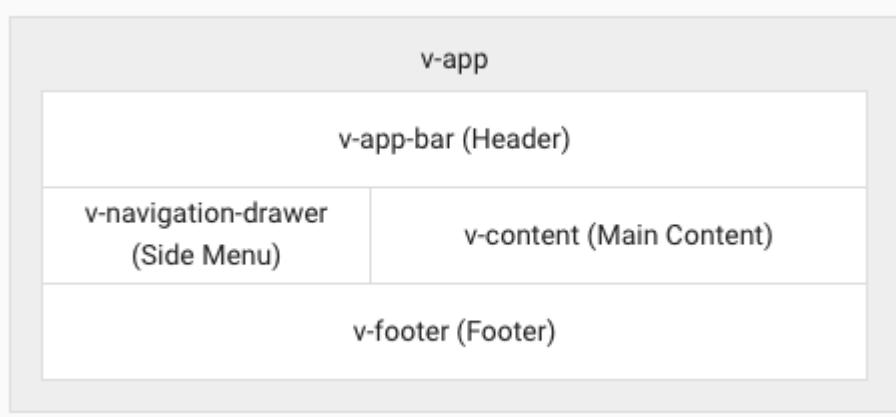
```

Terdapat empat component utama di bawah v-app yaitu:

- v-app-bar sebagai wrapper header aplikasi
- v-navigation-drawer sebagai wrapper dari side menu
- v-content sebagai wrapper konten utama aplikasi
- v-footer sebagai warapper footer aplikasi, dan

catatan: Component yang diawali dengan **v-** merupakan component vuetyf.

Sebagai gambaran, berikut ini skema dari layout di atas.



Mari kita jalankan perintah `npm run serve` pada projek root. atau bisa melalui vue ui dengan menjalankan tombol run task. Kemudian pada browser akses alamat `http://localhost:8080`

Header



Welcome to Vuetify

For help and collaboration with other Vuetify developers,
please join our online [Discord Community](#)

What's next?

[Explore components](#)

[Select a layout](#)

[Frequently Asked Questions](#)

Footer

Jika kita buka melalui layar desktop, maka tampilan sidebar menu akan muncul.

Side Menu

Header



Welcome to Vuetify

For help and collaboration with other Vuetify developers,
please join our online [Discord Community](#)

Footer

Oke, lalu apa langkah berikutnya?

Berikutnya kita oprek perbagian layout tersebut supaya terlihat seperti aplikasi sungguhan 😊.

Catatan: sebagai inspirasi bagimana koding di dalam masing-masing bagian layout, kamu bisa merujuk ke [tautan ini](#) untuk melihat contoh kode layout aplikasi berbasis vuetify.

Layout Header

Layout ini menggunakan component `v-app-bar`, sekali lagi silakan kamu merujuk ke [dokumentasi component ini](#) untuk melihat konfigurasinya secara lebih lengkap.

Catatan: component `v-app-bar` ini memiliki fungsi yang sama dengan component `v-toolbar` yang banyak kita gunakan pada buku versi sebelumnya.

Berikut ini kode dasar dari component ini yang akan menampilkan icon navigasi dan judul aplikasi.

```

1 <v-app-bar app color="primary" dark>
2   <v-app-bar-nav-icon></v-app-bar-nav-icon>
3
4   <v-toolbar-title>Vueshop</v-toolbar-title>
5 </v-app-bar>
```

Silakan terapkan kode di atas pada file `App.vue`, dan lihat hasilnya pada browser.



Berikutnya kita akan coba tambahkan sebuah button (component `v-btn`) yang memiliki icon (component `v-icon`) keranjang belanja pada pojok kanan, adapun iconnya kita akan gunakan default dari vuetyf yaitu material desain icon (mdi) yang kamu bisa lihat daftar icon pada [tautan ini](#).

```

1 <v-app-bar app color="primary" dark>
2   <v-app-bar-nav-icon></v-app-bar-nav-icon>
3
4   <v-toolbar-title>Vueshop</v-toolbar-title>
5
6   <!-- pemisah konten -->
7   <v-spacer></v-spacer>
8
9   <v-btn icon>
10    <v-icon>mdi-cart</v-icon>
11  </v-btn>
12 </v-app-bar>
```

Lihat hasilnya pada browser.



Pada icon keranjang belanja ini, kita bisa tambahkan sebuah penanda apabila keranjang belanja telah ada isinya, umumnya dengan menampilkan teks jumlah barang yang dipesan. Untuk itu, kita bisa gunakan component `v-badge`.

```

1 <v-btn icon>
2   <v-badge color="orange" overlap>
3     <template v-slot:badge>
```

```

4   <span>3</span>
5   </template>
6   <v-icon>mdi-cart</v-icon>
7   </v-badge>
8 </v-btn>

```

Pada kode di atas, warna dari badge kita set sebagai orange melalui atribut color, sedangkan text dari badge kita set menggunakan slot. Adapun atribut overlap kita gunakan untuk membuat tampilan teksnya menempel ke icon. Lihat hasilnya pada browser.



Kemudian, kita akan coba tambahkan tepat pada bagian bawah dari app-bar ini kolom pencarian dengan menggunakan component `text-field`.

Supaya kolom pencarian ini cukup untuk masuk dalam app-bar maka kita bisa gunakan atribut extended untuk menambah baris baru pada app-bar yang akan kita isi dengan component `text-field`.

```

1 <v-app-bar app color="primary" dark extended>
2 ...
3 ...
4 ...
5   <v-text-field slot="extension"></v-text-field>

```



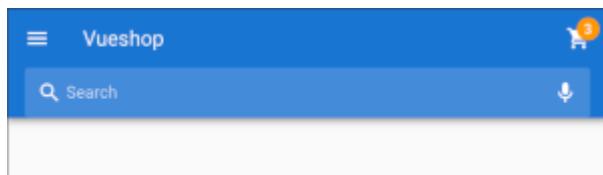
Untuk mempercantiknya kita bisa tambahkan placeholder, icon search & mic pada text field tersebut.

```

1 <v-text-field
2   slot="extension"
3   hide-details
4   append-icon="mdi-microphone"
5   flat
6   label="Search"
7   prepend-inner-icon="mdi-magnify"
8   solo-inverted
9 ></v-text-field>

```

Nah hasilnya bisa kita lihat di browser, kurang lebih sebagai berikut.



Oke silakan bereksperimen dengan berbagai konfigurasi yang bisa kamu terapkan pada component app-bar ini. Jika perlu, tirulah header dari aplikasi-aplikasi mobile populer untuk mengasah kemampuanmu.

Layout Main Content

Layout ini digunakan untuk menampilkan konten utama dari aplikasi, atau content yang akan secara dinamis berubah sesuai dengan halaman yang dibuka oleh pengguna.

Pada contoh sebelumnya kodenya sebagai berikut:

```
1 <v-content>
2   <v-container fluid>
3     <router-view></router-view>
4   </v-container>
5 </v-content>
```

Component v-content ini ukurannya akan menyesuaikan content di dalamnya. Component v-container digunakan untuk memberikan spasi margin (gutter) sehingga nantinya content utama tidak terlalu melebar dengan layar.

Adapun component router-view sebagai tempat untuk menampilkan view berdasarkan routing tertentu (lihat kembali materi routing). Untuk sekedar mencoba apakah dynamic view pada router-view benar-benar berjalan maka kita bisa tambahkan atribut `to` pada button keranjang belanja di layout header.

```
1 <v-btn icon to="/about">
```

Routing `/about` sengaja kita gunakan hanya untuk mencoba fungsi ini saja karena kebetulan sudah dibuatkan oleh Vue ketika kita generate projek ini.

Silakan coba dibrowser dan klik button keranjang belanja, maka konten utama akan berpindah ke view About.

Perpindahan antar konten atau halaman bisa juga kita buat agar lebih *smooth* dengan menambahkan efek transisi yang juga telah disediakan oleh vuertify.

Ada 12 jenis efek transisi yang telah disediakan oleh vuertify.

- slide-x-transition
- slide-x-reverse-transition
- slide-y-transition
- slide-y-reverse-transition
- scroll-x-transition
- scroll-x-reverse-transition
- scroll-y-transition
- scroll-y-reverse-transition
- scale-transition
- fade-transition
- expand-transition
- expand-x-transition

Untuk menggunakan efek ini pada component adalah dengan menambahkan atribut atau prop transition pada component dengan nilai di atas.

```
1 <v-menu transition="slide-y-transition">
```

atau untuk component router view dilakukan dengan cara mewrap dengan component transition di atas.

```
1 <v-slide-y-transition>
2   <router-view></router-view>
3 </v-slide-y-transition>
```

Silakan dicoba dan rasakan efeknya 😊.

Layout Footer

Sebagaimana umumnya aplikasi, kita bisa menggunakan footer ini untuk menampilkan identitas atau copyright dari aplikasi, menampilkan link-link penting, informasi tentang aplikasi dll.

Pada contoh ini, kita hanya akan menampilkan teks tahun saat ini dan teks nama aplikasi yaitu Vueshop. Supaya terkesan lebih kuat maka kita bisa bungkus v-footer ini dengan component v-card.

```

1 <v-card>
2   <v-footer absolute app>
3     <v-card-text class="text-center">
4       &copy; {{ new Date().getFullYear() }} – <strong>Vueshop</strong>
5     </v-card-text>
6   </v-footer>
7 </v-card>
```

Hasilnya kurang lebih sebagai berikut.

© 2019 – Vueshop

Layout Sidebar

Sidebar ini akan kita gunakan untuk menampilkan menu-menu dari aplikasi. Posisinya akan kita letakkan di sisi kiri dan pada layar kecil akan tersembunyi, untuk menampilkannya kita akan gunakan button `v-app-bar-nav-icon` pada header sebagai triggernya.

Pertama yang harus kita lakukan adalah membuat properti data untuk menyimpan status sidebar tersebut apakah tampil atau disembunyikan, defaultnya disembunyikan.

Pada file `src/App.vue` kita tambahkan properti data bernama `drawer` dengan nilai `false`.

```

1 ...
2 <script>
3 export default {
4   name: 'App',
5   data: () => ({
6     drawer: false
7   })
8 };
9 </script>
```

Kemudian masih pada `App.vue`, kita tambahkan kode pada button `v-app-bar-nav-icon` agar ketika diklik akan mengubah data `drawer` ini menjadi `true` jika dia `false` atau menjadi `false` jika dia `true`.

```

1 <v-app-bar-nav-icon @click.stop="drawer = !drawer"></v-app-bar-nav-icon>
```

Kemudian jangan lupa pada component `v-navigation-drawer`, pasang data `drawer` melalui prop `v-model`. Jika `drawer` bernilai `true` maka akan memicu component ini untuk ditampilkan.

```

1 <v-navigation-drawer app v-model="drawer">
```

Saatnya mencoba.



Ketika button navigasi diklik maka dari sisi kiri akan muncul sidebar tersebut.

Setelah secara fungsional berjalan, kini saatnya kita mendesain layout dari sidebar menu ini.

Supaya tidak terkesan flat maka sidebar ini akan kita bungkus dengan component v-card.

```

1 <v-card>
2   <v-navigation-drawer app v-model="drawer">
3     Sidebar Menu
4   </v-navigation-drawer>
5 </v-card>
```

Kemudian, pada sidebar ini, kita akan gunakan untuk menampilkan menu aplikasi kita dalam bentuk list. Untuk membuatnya, kita bisa gunakan component **v-list**, di mana masing-masing item dari list tersebut dapat berisi icon, teks dan link.

Namun sebelumnya, kita definisikan dulu menu aplikasi kita pada properti data (file `src/App.vue` bagian script)

```

1 data: () => ({
2   drawer: false,
3   menus: [
4     { title: 'Home', icon: 'mdi-home', route: '/' },
5     { title: 'About', icon: 'mdi-account', route: '/about' },
6   ],
7 })
```

Baru setelah itu, kita buat templatanya, masih pada `src/App.vue`, kita tambahkan kode berikut di dalam component **navigation-drawer**.

```

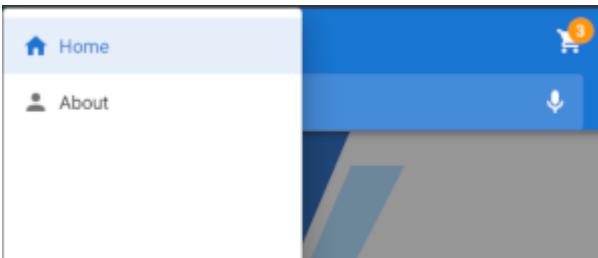
1 <v-list>
2   <v-list-item
3     v-for="(item, index) in menus"
4       :key="`menu-`+index"
5       :to="item.route"
6     >
7       <v-list-item-icon>
8         <v-icon left>{{ item.icon }}</v-icon>
9       </v-list-item-icon>
```

```

10
11      <v-list-item-content>
12          <v-list-item-title>{{ item.title }}</v-list-item-title>
13      </v-list-item-content>
14  </v-list-item>
15</v-list>

```

Mari kita lihat hasilnya:



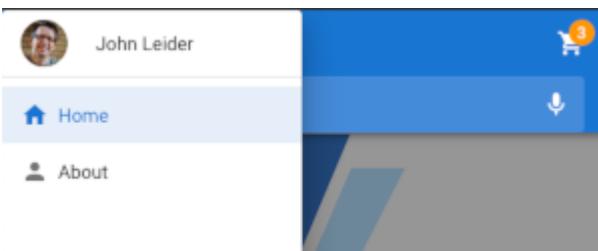
Silakan coba klik salah satu menunya, maka halaman akan berpindah sesuai dengan menu route-nya. Selanjutnya, kita bisa tambahkan menu-menu lainnya.

Di sidebar ini, kita juga akan menampilkan user yang sedang login berupa avatar dan nama user tersebut. Di mana posisinya kita letakkan di atas menu aplikasi.

```

1  <v-list-item>
2      <v-list-item-avatar>
3          <v-img src="https://randomuser.me/api/portraits/men/78.jpg"></v-img>
4      </v-list-item-avatar>
5      <v-list-item-content>
6          <v-list-item-title>John Leider</v-list-item-title>
7      </v-list-item-content>
8  </v-list-item>
9
10 <v-divider></v-divider>

```



Kedepan, untuk mendeteksi user sudah login atau belum, kita akan gunakan data dari vuex. Namun pada bagian ini, kita bisa simulasikan dahulu dengan menggunakan properti data untuk mendefinisikan status login user dan directive v-if pada component untuk menampilkan atau menyembunyikan content tersebut berdasarkan status login tersebut.

Untuk melakukannya, kita tambahkan pada properti data sebuah variabel baru bertipe boolean, misalnya: guest.

```

1  data: () => ({
2      drawer: false,
3      menus: [
4          { title: 'Home', icon: 'mdi-home', route: '/' },
5          { title: 'About', icon: 'mdi-account', route: '/about' },
6      ],

```

```
7   guest: true, // <= tambahkan ini
8 })
```

Kemudian pada component yang menampilkan user login di atas, kita tambahkan directive v-if yang nilainya negasi dari data guest (bukan guest).

```
1 <v-list-item v-if="!guest">
2   <v-list-item-avatar>
```

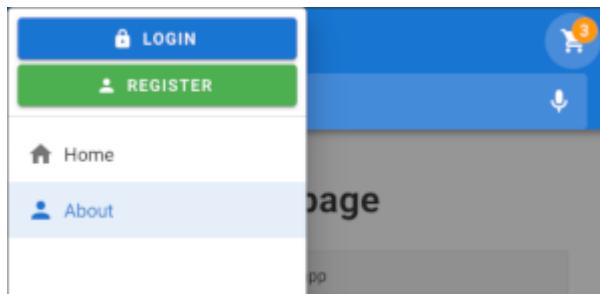
Ketika dijalankan, tampilan user login tidak akan muncul, karena defaultnya data guest di set true.

Lalu bagaimana jika user belum login? maka akan kita tampilkan button login dan register.

```
1 <div class="pa-2" v-if="guest">
2   <v-btn block color="primary" class="mb-1">
3     <v-icon left>mdi-lock</v-icon>
4     Login
5   </v-btn>
6   <v-btn block color="success">
7     <v-icon left>mdi-account</v-icon>
8     Register
9   </v-btn>
10 </div>
```

Jika kita perhatikan ada class `pa-2` dan `mb-1` apa itu? class tersebut merupakan class yang disediakan oleh vuetyfy untuk mengontrol padding dan margin. `pa` maskudnya padding all, sedangkan angka 2 artinya 2 pixel. sedangkan `mb` artinya margin bottom. Selengkapnya, kamu bisa merujuk ke tautan ini <https://vuetyfyjs.com/en/styles/spacing>.

Mari kita lihat hasil kreasi kita.

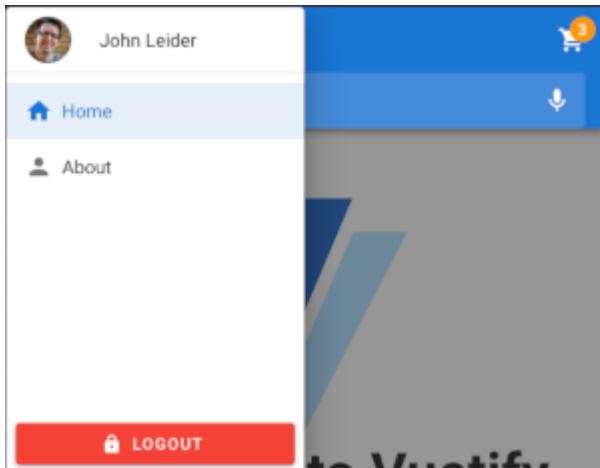


Gimana? sudah seperti aplikasi beneran atau belum nih?

Ah iya, hampir lupa, kita perlu tambahkan satu button lagi, ada login ada register, kenapa logout belum? hehe Nah, supaya tidak membosankan, maka button logout tidak kita letakkan di atas menu aplikasi, melainkan akan kita letakkan di bawah menu aplikasi. Triknya dengan menggunakan directive `v-slot:append` seperti berikut.

```
1 <template v-slot:append v-if="!guest">
2   <div class="pa-2">
3     <v-btn block color="red" dark>
4       <v-icon left>mdi-lock</v-icon>
5       Logout
6     </v-btn>
7   </div>
8 </template>
```

Untuk melihat hasilnya, maka ubah dulu data guest menjadi false.



Cee.. ile.. manis banget kan? 😊

Membuat Halaman Home

Halaman home adalah halaman ketika pengguna mengakses aplikasi kita. Menyusun halaman ini gampang-gampang susah karena ini adalah halaman yang cukup menentukan apakah user akan terus berada pada aplikasi kita ataukah justru meninggalkannya. Oleh karena itu, sebisa mungkin kita menampilkan konten yang memang paling diharapkan oleh user terhadap aplikasi kita. Adapun konten lain bisa kita letakkan pada halaman lain.

Pada projek ini, halaman home akan menampilkan dua kategori buku yang diambil secara random dalam bentuk grid, di mana setiap itemnya akan ditampilkan gambar dari kategori dan teks namanya (card). Pada bagian atas sebelah kanan heading, kita akan tampilkan teks link `All Categories` yang akan mengarahkan user ke halaman daftar lengkap dari kategori buku .

Pada bagian bawahnya, akan menampilkan empat buku terpopuler dalam bentuk list grid atau card juga, di mana setiap itemnya akan ditampilkan gambar dari buku, teks judul, dan harganya. Pada bagian atas sebelah kanan heading, juga akan kita tampilkan teks link `All Books` yang akan mengarahkan user ke halaman daftar seluruh buku yang tersedia.

Layout Halaman Home

Pertama, kita akan membuat layout untuk menampilkan kategori buku di mana pada tiap itemnya berupa gambar kategori dan teks nama kategorinya. Itu artinya kita perlu membuat properti data list kategori buku.

Buka file `src/views/Home.vue` pada bagian script tambahkan properti data kategori buku dan set nilainya menggunakan data dummy.

```

1 <script>
2 export default {
3   data: () => ({
4     categories: [
5       {
6         id: 1,
7         image: 'https://via.placeholder.com/150',
8         name: 'Ekonomi',
9         slug: 'ekonomi'
10      },
11      {
12        id: 2,
13        image: 'https://via.placeholder.com/150',

```

```

14     name: 'Agama',
15     slug: 'agama'
16   },
17 ]
18 })
19 ;
20 </script>

```

List kategori buku ini kemudian di render pada templatanya,

```

1 <template>
2   <div>
3     <!-- template categories -->
4     <v-container class="ma-0 pa-0" grid-list-sm>
5       <div class="text-right">
6         <v-btn small text to="/categories" class="blue--text">
7           All Categories <v-icon>mdi-chevron-right</v-icon>
8         </v-btn>
9       </div>
10      <v-layout wrap>
11        <v-flex v-for="(category, index) in categories" :key="`category-
12 `+category.id" xs6>
13          <v-card :to="'/category/'+ category.slug">
14            <v-img
15              :src="category.image"
16              class="white--text"
17            >
18              <v-card-title
19                class="fill-height align-end"
20                v-text="category.name"
21              ></v-card-title>
22            </v-img>
23          </v-card>
24        </v-flex>
25      </v-layout>
26    </v-container>
27
28    <!-- template books -->
29
30  </div>
</template>

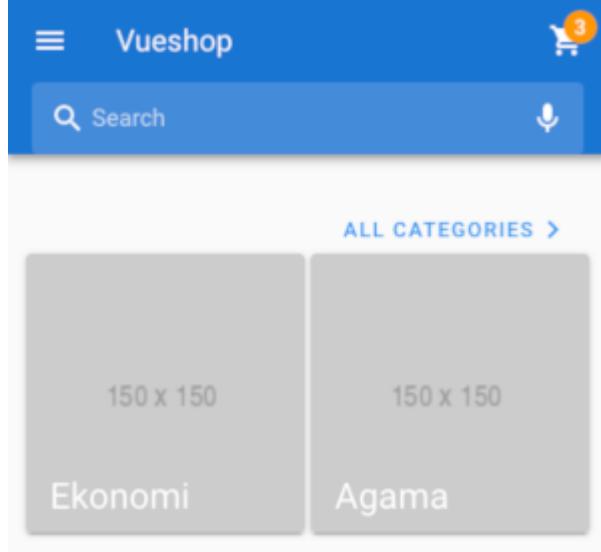
```

Pada teks All Categories di atas, kita definisikan link ke route `/categories`, route ini nantinya akan mengarahkan ke halaman yang menampilkan semua kategori buku. Demikian juga pada masing-masing item kategori juga kita definisikan link ke route `/category/{category-slug}`

Diinget-inget ya guys! jangan sampai nanti lupa membuat halaman ini 😊.

Pada component `v-card-title` classnya menggunakan `fill-height align-end` supaya area teksnya seluas gambar kategori namun posisi teksnya di akhir area teks tersebut.

Mari kita lihat hasilnya.



Selanjutnya, kita akan lakukan hal yang sama untuk data list buku.

```

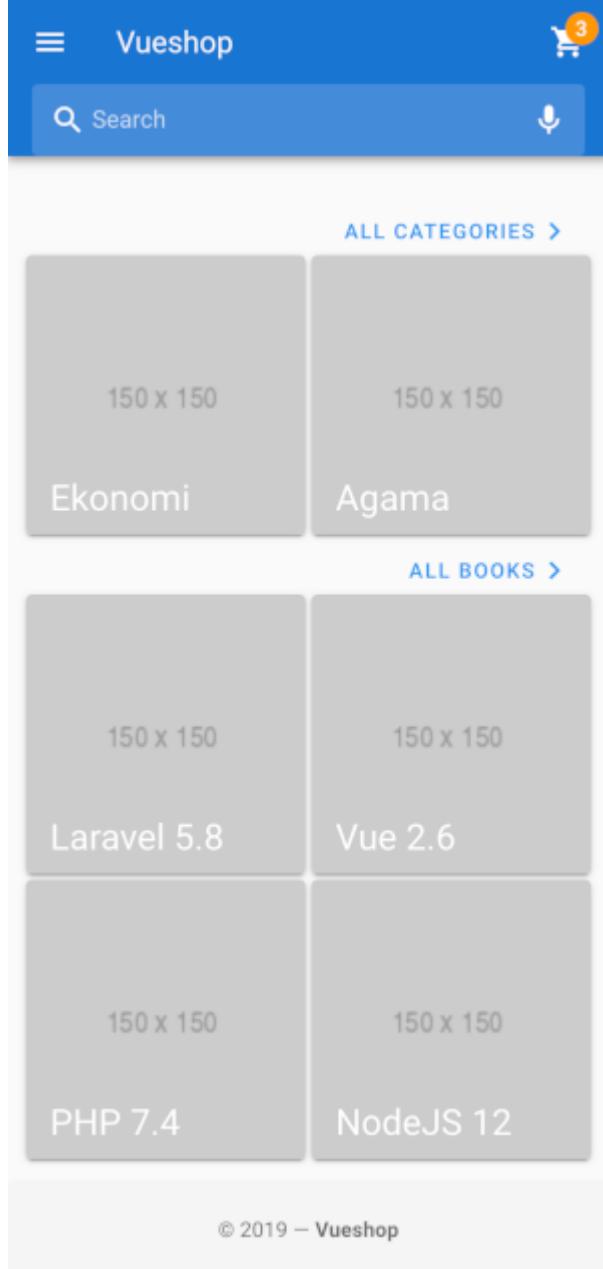
1  export default {
2    data: () => ({
3      ...
4      books: [
5        {
6          id: 1,
7          cover: 'https://via.placeholder.com/150',
8          title: 'Laravel 5.8',
9          slug: 'laravel-5-8'
10        },
11        {
12          id: 2,
13          cover: 'https://via.placeholder.com/150',
14          title: 'Vue 2.6',
15          slug: 'vue-2-6'
16        },
17        {
18          id: 3,
19          cover: 'https://via.placeholder.com/150',
20          title: 'PHP 7.4',
21          slug: 'php-7-4'
22        },
23        {
24          id: 4,
25          cover: 'https://via.placeholder.com/150',
26          title: 'NodeJS 12',
27          slug: 'nodejs-12'
28        },
29      ]
30    })
31  };

```

Kemudian data daftar buku tersebut kita petakan pada templatanya.

```
1 <v-container class="ma-0 pa-0 mt-2" grid-list-sm>
2   <div class="text-right">
3     <v-btn small text to="/books" class="blue--text">
4       All Books <v-icon>mdi-chevron-right</v-icon>
5     </v-btn>
6   </div>
7   <v-layout wrap>
8     <v-flex v-for="(book, index) in books" :key="`book-`+book.id" xs6>
9       <v-card :to="'/book/'+ book.slug">
10         <v-img
11           :src="book.cover"
12           class="white--text"
13         >
14           <v-card-title
15             class="fill-height align-end"
16             v-text="book.title"
17           ></v-card-title>
18         </v-img>
19       </v-card>
20     </v-flex>
21   </v-layout>
22 </v-container>
```

Mari kita lihat hasilnya.



Gimana? cakep kan?

Endpoint Random Category

Karena sudah memerlukan data buku dan kategorinya maka kita perlu mengambil data dari backend atau dalam hal ini web service. Berikut ini akan kita bahas satu persatu endpoint yang dibutuhkan untuk halaman home ini.

Catatan: kita pindah ke Laravel dulu ya!

Endpoint ini sifatnya publik artinya bebas diakses oleh siapapun tanpa perlu login dahulu. Sebelumnya, kita perlu buat dulu resource collection untuk model Category.

```
1 | php artisan make:resource Categories --collection
```

Perintah ini akan menggenerate file `Categories.php` pada direktori `app/Http/Resources`. Buka file `Categories.php` tersebut dan lakukan modifikasi fungsi `toArray()` supaya outputnya standard yaitu `status`, `message`, `data`. Berikut ini kodennya.

```

1 public function toArray($request)
2 {
3     return [
4         'status' => 'success',
5         'message' => 'categories data',
6         'data' => parent::toArray($request),
7     ];
8 }
```

Kemudian kita buat controller Category.

```
1 php artisan make:controller CategoryController
```

Perintah ini akan menggenerate file `CategoryController.php` pada direktori `app/Http/Controllers`. Buka file `CategoryController.php` dan buat fungsi baru yaitu `random()`.

```

1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use App\Category;
7 use App\Http\Resources\Categories as CategoryResourceCollection;
8
9 class CategoryController extends Controller
10 {
11     public function random($count)
12     {
13         $criteria = Category::select('*')
14             ->inRandomOrder()
15             ->limit($count)
16             ->get();
17         return new CategoryResourceCollection($criteria);
18     }
19 }
```

Fungsi ini akan mengembalikan data category secara random dengan jumlah tertentu sesuai dengan parameter `$count` yang dikirimkan saat fungsi ini dipanggil serta data yang dikembalikan akan dibungkus dengan resource collection `CategoryResourceCollection` supaya format datanya standard.

Catatan: Jika file-file tersebut sebelumnya sudah ada maka hapus saja dulu.

Langkah selanjutnya, fungsi `random` pada controller Category tersebut kita daftarkan pada router `api.php`.

```

1 <?php
2 use Illuminate\Http\Request;
3
4 Route::prefix('v1')->group(function () {
5     // public
6     Route::post('login', 'AuthController@login');
7     Route::post('register', 'AuthController@register');
```

```

8     Route::get('categories/random/{count}', 'CategoryController@random');
9 // <== ini ya gaes
10
11
12     // private
13     Route::middleware(['auth:api'])->group(function () {
14         Route::post('logout', 'AuthController@logout');
15     });
16 });

```

Mari kita uji coba routing `http://larashop-api.test/v1/categories/random/2` menggunakan postman. Maka hasilnya kurang lebih sebagai berikut.

```

1  {
2      "status": "success",
3      "message": "categories data",
4      "data": [
5          {
6              "id": 1,
7              "name": "suscipit",
8              "slug": "suscipit",
9              "image": "ef3312ea1fdc5f17efb006dd5bf582bf.jpg",
10             "status": "PUBLISH",
11             "created_at": "2019-07-31 04:28:16",
12             "updated_at": null,
13             "deleted_at": null,
14             "created_by": null,
15             "updated_by": null,
16             "deleted_by": null
17         },
18         {
19             "id": 5,
20             "name": "temporibus",
21             "slug": "temporibus",
22             "image": "bb3431eaf8fcb5e3def2057e5cd76ac2.jpg",
23             "status": "PUBLISH",
24             "created_at": "2019-07-31 04:28:29",
25             "updated_at": null,
26             "deleted_at": null,
27             "created_by": null,
28             "updated_by": null,
29             "deleted_by": null
30         }
31     ]
32 }

```

Lho kok udah ada isinya? ya iyalah karena pada bab sebelumnya telah kita masukkan data dummy yang digenerate menggunakan pustaka Faker dan dimasukkan ke database melalui seeder. Hayoo cek lagi ya gaes..

Endpoint Top Book

Masih pada projek Laravel, endpoint top book yang akan kita buat ini sifatnya juga publik artinya bebas diakses oleh siapapun tanpa perlu login dahulu. Sama seperti pembuatan endpoint random category, kita juga perlu buat dulu resource collection untuk model Book.

```
1 | php artisan make:resource Books --collection
```

Perintah ini akan menggenerate file Books.php pada direktori app/Http/Resources. Buka file Books.php lalu modifikasi juga fungsi toArray() menjadi sebagai berikut.

```
1 | public function toArray($request)
2 |
3 |     return [
4 |         'status'  => 'success',
5 |         'message' => 'books data',
6 |         'data'     => parent::toArray($request),
7 |     ];
8 | }
```

Kemudian kita buat controller Book dengan menggunakan perintah berikut.

```
1 | php artisan make:controller BookController
```

Perintah ini akan menggenerate file BookController.php pada direktori app/Http/Controllers, lalu buatlah fungsi baru yaitu top() pada controller ini.

```
1 | <?php
2 |
3 | namespace App\Http\Controllers;
4 |
5 | use Illuminate\Http\Request;
6 | use App\Book;
7 | use App\Http\Resources\Books as BookResourceCollection;
8 |
9 | class BookController extends Controller
10 {
11     public function top($count)
12     {
13         $criteria = Book::select('*')
14             ->orderBy('views', 'DESC')
15             ->limit($count)
16             ->get();
17         return new BookResourceCollection($criteria);
18     }
19 }
20 }
```

Fungsi ini akan mengembalikan data book yang jumlah views-nya paling besar (order by view DESC) dan jumlah tertentu sesuai dengan parameter \$count yang dikirimkan. Adapun output dari fungsi ini juga akan berformat data resource.

Langkah selanjutnya, fungsi top pada controller Book tersebut harus kita daftarkan juga pada router api.php.

```

1 <?php
2 use Illuminate\Http\Request;
3
4 Route::prefix('v1')->group(function () {
5     // public
6     Route::post('login', 'AuthController@login');
7     Route::post('register', 'AuthController@register');
8
9     Route::get('categories/random/{count}', 'CategoryController@random');
10    Route::get('books/top/{count}', 'BookController@top'); // <= ini ya
11
12    // private
13    Route::middleware(['auth:api'])->group(function () {
14        Route::post('logout', 'AuthController@logout');
15    });
16});
```

Mari kita uji coba routing `http://larashop-api.test/v1/books/top/4` menggunakan postman. Maka hasilnya kurang lebih sebagai berikut.

```

1 {
2     "status": "success",
3     "message": "books data",
4     "data": [
5         {
6             "id": 1,
7             "title": "Rerum accusamus dolores totam quis quia quisquam
8 odit",
9             "slug": "rerum-accusamus-dolores-totam-quis-quia-quisquam-
10 odit",
11             "description": "Doloremque maxime voluptatem id. Enim aut et
12 repudiandae similique architecto dolor rerum ut. Ut nisi culpa id vel.",
13             "author": "Mark Koch",
14             "publisher": "Glover and Sons",
15             "cover": "a9713db1fd7738850e55d1b76f1ebda5.jpg",
16             "price": 200000,
17             "weight": 0.5,
18             "views": 0,
19             "stock": 1,
20             "status": "PUBLISH",
21             "created_at": "2019-07-31 04:27:04",
22             "updated_at": null,
23             "deleted_at": null,
24             "created_by": null,
25             "updated_by": null,
26             "deleted_by": null
27         },
28         {
29             "id": 2,
```

```

30     "title": "Dolorem dolores distinctio quidem",
31     "slug": "dolorem-dolores-distinctio-quidem",
32     "description": "Illo quod repellat repudianda voluptate
33 corporis veniam. Quis hic quo assumenda sed alias. Voluptatibus porro
34 quisquam doloremque ut similique maxime deleniti.",
35         "author": "Mrs. Nicole Glover",
36         "publisher": "Abshire, Monahan and Kilback",
37         "cover": "1d3edab23a8f0c03fc4e9e48d15a3e84.jpg",
38         "price": 400000,
39         "weight": 0.5,
40         "views": 0,
41         "stock": 1,
42         "status": "PUBLISH",
43         "created_at": "2019-07-31 04:27:06",
44         "updated_at": null,
45         "deleted_at": null,
46         "created_by": null,
47         "updated_by": null,
48         "deleted_by": null
49     },
50     {
51         "id": 3,
52         "title": "Suscipit repellendus saepe repellat dolor beatae
53 fugiat",
54         "slug": "suscipit-repellendus-saepe-repellat-dolor-beatae-
55 fugiat",
56         "description": "Ea consequuntur voluptatibus assumenda
57 commodi. Sunt architecto molestiae rerum quam doloribus. Libero qui omnis
58 doloribus quam alias laboriosam vel. Necessitatibus aut commodi aut
59 cumque. Et ut voluptatum aut incident ut.",
60         "author": "Dan Koch",
61         "publisher": "Konopelski-Bergstrom",
62         "cover": "097609a3c9106c35d9b30c28901fdd38.jpg",
63         "price": 350000,
64         "weight": 0.5,
65         "views": 0,
66         "stock": 1,
67         "status": "PUBLISH",
68         "created_at": "2019-07-31 04:27:10",
69         "updated_at": null,
70         "deleted_at": null,
71         "created_by": null,
72         "updated_by": null,
73         "deleted_by": null
74     },
75     {
76         "id": 4,
77         "title": "Laboriosam animi quam",
78         "slug": "laboriosam-animi-quam",
79         "description": "Quasi animi repellendus quas eveniet. Hic
80 numquam repudianda ipsum omnis dolore voluptas. Quod qui et enim quia
81 et. Voluptatem nihil error perspiciatis libero quas.",
82         "author": "Prof. Sage Fadel DDS",
83         "publisher": "Doyle-Dibbert",

```

```

84     "cover": "815511f4346a669072ff052c918273e4.jpg",
85     "price": 50000,
86     "weight": 0.5,
     "views": 0,
     "stock": 1,
     "status": "PUBLISH",
     "created_at": "2019-07-31 04:27:13",
     "updated_at": null,
     "deleted_at": null,
     "created_by": null,
     "updated_by": null,
     "deleted_by": null
   }
]
}

```

Menghubungkan Layout Home dengan Endpoint Random Categories & Top Books

Setelah kode endpoint OK, maka sekarang kita berpindah ke projek Vue lagi, untuk menampilkan data random category dan top buku yang disupplay dari endpoint web service tersebut.

Pada layout Home sebelumnya kita menggunakan data dummy, nah karena data sebenarnya sudah siap digunaan melalui endpoint maka kini saatnya kita integrasikan.

Ada dua endpoint yang akan kita akses melalui aplikasi Vue yaitu:

- Random Categories => <http://larashop-api.test/v1/categories/random/2>
- Top Books => <http://larashop-api.test/v1/books/top/4>

Bagaimana cara mengaksesnya? sebagaimana yang telah dijelaskan pada bab sebelumnya, kita akan menggunakan pustaka axios untuk mengaksesnya.

```

1  this.axios.get('http://larashop-api.test/v1/categories/random/2')
2    .then((response) => {
3      let { data } = response.data
4      console.log(data)
5    })

```

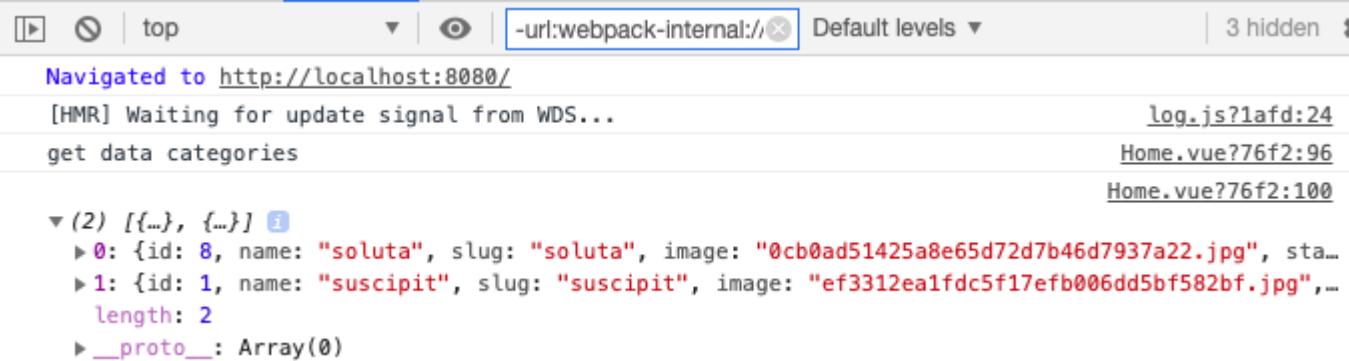
Letakkan kode di atas pada hook `created()` di file `src/views/Home.vue`.

```

1  created(){
2    console.log('get data categories')
3    this.axios.get('http://larashop-api.test/v1/categories/random/2')
4      .then((response) => {
5        let { data } = response.data
6        console.log(data)
7      })
8      .catch((error) => {
9        let { response } = error
10       console.log(response)
11     })
12   }

```

Jika kita akses halaman home maka pada console log browser akan muncul sebagai berikut:



```
Navigated to http://localhost:8080/
[HMR] Waiting for update signal from WDS...
get data categories
log.js?1afcd:24
Home.vue?76f2:96
Home.vue?76f2:100
▼(2) [{}]
▶0: {id: 8, name: "soluta", slug: "soluta", image: "0cb0ad51425a8e65d72d7b46d7937a22.jpg", sta...
▶1: {id: 1, name: "suscipit", slug: "suscipit", image: "ef3312ea1fdc5f17efb006dd5bf582bf.jpg",...
length: 2
__proto__: Array(0)
```

Nah kalo kita perhatikan penulisan URL secara hardcode pada setiap request axios tentu tidak efisien dan apabila terjadi perubahan URL misalnya dari development ke production maka kita harus mengubah kodennya satu persatu, tentu melelahkan bukan?

Axios memungkinkan kita mengeset base URLnya sehingga kita tidak perlu menyebutkan secara lengkap URL endpointnya pada setiap request.

Caranya melalui file `src/plugins/axios.js`, update menjadi sebagai berikut:

```
1 let config = {
2   baseURL: 'http://larashop-api.test/v1',
3 };
```

Atau kita bisa menggunakan konstanta global `.env` supaya lebih dinamis lagi.

```
1 let config = {
2   baseURL: process.env.VUE_APP_API_URL + '/v1',
3 };
```

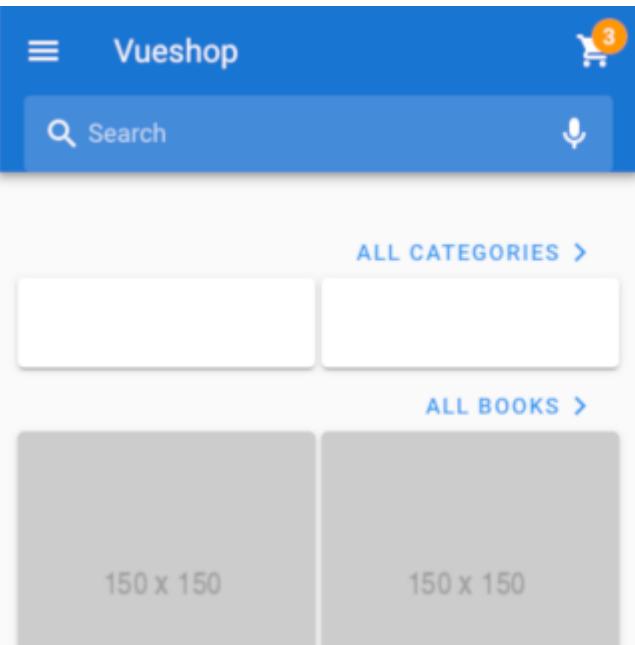
Dengan cara ini maka untuk mengakses endpoint via axios, cukup dituliskan path endpointnya saja `this.axios.get('/categories/random/2')`.

Setelah kita pastikan bahwa data endpoint berhasil diambil, maka langkah berikutnya adalah memasukkan data respon endpoint ini ke properti data categories (`this.categories`) yang sebelumnya diisi dengan dummy data, dan jangan lupa kosongkan saja inisial data categoriesnya.

```
1 data: () => ({
2   categories: [], // <== ini gaes
3   books: [
4     // ...
5   ]
6 },
7 created(){
8   console.log('get data categories')
9   this.axios.get('/categories/random/2') // <== jadi ginis
10  .then((response) => {
11    let { data } = response.data
12    this.categories = data // <== ini gaes
13  })
14  .catch((error) => {
15    let { responses } = error
16    console.log(responses)
17}
```

```
18 |     })  
    }
```

Mari kita lihat hasilnya..



```
[HMR] Waiting for update signal from WDS...
● > [Vuetify] Image load failed
src: a7b2761324edc77f1d73be36289ed82c.jpg
found in
----> <VImg>
      <RouterLink>
      <VCard>
        <Home> at src/views/Home.vue
        <VContent>
          <VApp>
            <App> at src/App.vue
            <Root>
● > [Vuetify] Image load failed
src: ff1f2dde4ea206d5e635d82a784628d3.jpg
found in
----> <VImg>
      <RouterLink>
      <VCard>
        <Home> at src/views/Home.vue
        <VContent>
          <VApp>
            <App> at src/App.vue
            <Root>
```

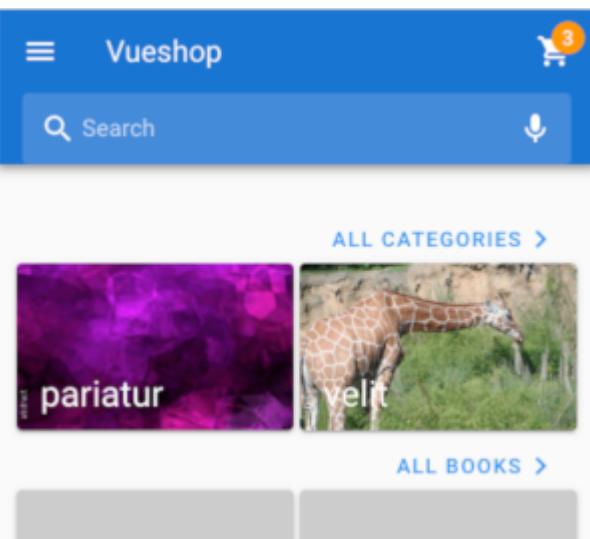
Ops, ada kesalahan terjadi, dimana tertulis "Image load failed", mengapa itu bisa terjadi?

Ya, itu terjadi karena data image yang diterima dari endpoint hanya berupa nama file image-nya saja tanpa URL lengkapnya.

Baik, karena image category di simpan di server endpoint pada folder /images/categories/ maka dari itu, kita perlu tambahkan URL lengkap ini `http://larashop-api.test/images/categories/` ketika mengeset data src ke component v-image.

```
1 <v-img  
2   :src="'http://larashop-api.test/images/categories/'+category.image"  
3   class="white--text"  
4 >
```

Hasilnya? boom!



Berhasil.

Nah, supaya kodennya lebih ringkas, maka kita bisa saja membuatkan sebuah method yang mereturn image dan URL lengkapnya.

```

1 methods: {
2     getImage(image){
3         if(image!=null && image.length>0){
4             return "http://larashop-api.test/images"+ image
5         }
6         // default image jika tidak ditemukan,
7         // letakkan image ini pada folder /public/img di project Vue
8         return "/img/unavailable.png"
9     },
10 }

```

Tentu saja pada templatennya kita ubah menjadi sebagai berikut.

```

1 <v-img
2   :src="getImage('/categories/'+category.image)"
3   class="white--text"
4 >

```

Gimana? lebih ringkas kan?

Kalau kita perhatikan bahwa URL `http://larashop-api.test` adalah URL server development yang ketika kode kita telah di deploy ke production maka URL ini tentu akan diganti dengan URL server production, nah supaya lebih dinamis kita juga bisa lakukan cara sebagaimana yang kita lakukan pada pustaka axios. Sehingga method `getImage` menjadi sebagai berikut.

```

1 getImage(image){
2     if(image!=null && image.length>0){
3         return process.env.VUE_APP_BACKEND_URL + "/images"+ image
4     }
5     return "/img/unavailable.png"
6 },

```

Keren kan?

Ehm tunggu, method `getImage` ini nantinya pasti akan banyak digunakan pada component lain yang berkaitan dengan gambar buku atau category, karena itu sebenarnya kita bisa saja mendaftarkannya sebagai global method menggunakan mixins atau yang lainnya. Pada contoh ini, penulis akan mencontohkan kepadamu menggunakan pendekatan plugins.

Buat file `helper.js` pada direktori `src/plugins`.

```

1 "use strict"
2 import Vue from 'vue'
3 const Helper = {
4     install(Vue) {
5         Vue.prototype.appName = process.env.VUE_APP_NAME
6         Vue.prototype.getImage = function (image){
7             if(image!=null && image.length>0){
8                 return process.env.VUE_APP_BACKEND_URL + "/images"+ image
9             }

```

```

10         return "/img/unavailable.png"
11     }
12   }
13 }
14
15 Vue.use(Helper)

```

Cara mendefinisikan properti atau method pada objek Vue adalah dengan menggunakan JS prototype, yaitu dengan menuliskan `Vue.prototype`. lalu diikuti dengan nama properti atau methodnya.

Jangan lupa import file helper.js ini pada `src/main.js`

```

1 import './plugins/helper'

```

Yap, alih alih menggunakan mixins, kita malah menggunakan fitur prototype pada Javascript (bukan Vue) untuk mendaftarkan atau menginjeksi property dan methods/fungsi pada objek Vue. Dengan cara ini maka properti `appName` dan method `getImage()` dapat kita akses dari seluruh component Vue.

Karena kita telah menggunakan global method `getImage()` melalui plugins maka method `getImage()` yang sudah kita deklarasikan di dalam component Home sudah tidak perlu lagi, hapus saja.

Nah, bagaimana jika di component kita menggunakan nama properti atau method yang sama dengan global properti dan method tersebut? maka tidak akan terjadi error melainkan lokal properti atau method akan mengoverride-nya.

Silakan dicoba dan rasakan kedahsyatannya.

Apa selanjutnya? yap tentu saja melanjutkan ngoprek halaman home, dimana data `books` belum menggunakan endpoint. Sampai disini sebenarnya penulis sudah tidak perlu memandu kamu lagi, karena konsepnya sama persis dengan yang sebelumnya. Tapi karena penulis lagi baik hati dan akan selalu demikian maka akan dipandu 😊.

Pada hook created, tambahkan kode untuk mengakses endpoint Top Book.

```

1 this.axios.get('/books/top/4')
2   .then((response) => {
3     let { data } = response.data
4     this.books = data
5   })
6   .catch((error) => {
7     let { responses } = error
8     console.log(responses)
9   })

```

Kemudian pada template ubah bagian v-img menjadi sebagai berikut:

```

1 <v-img
2   :src="getImage(' /books/' +book.cover)"
3   class="white--text"
4 >

```

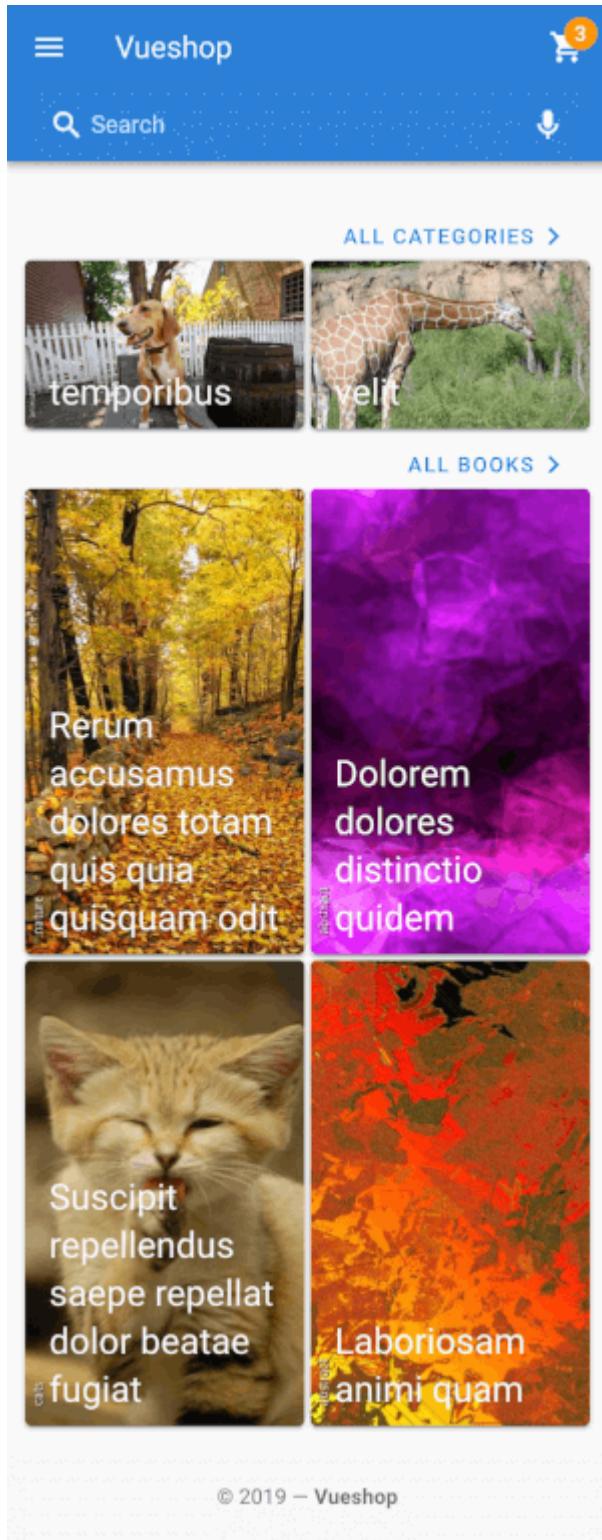
Dan pastinya kosongkan initial data books.

```

1 | data: () => ({
2 |   categories: [],
3 |   books: []
4 | })

```

Sekarang, lihat hasilnya:



Mantap kan?

Oh iya, karena appName juga telah kita definikan di plugin helper maka kita juga bisa langsung pakai layaknya properti data biasa. Misalnya pada App.vue, kita bisa terapkan pada toolbar.

```

1 | <v-toolbar-title to="/">{{ appName }}</v-toolbar-title>

```

Oke juga kan?

Pada halaman home ini, terdapat 4 jenis link yang harus kita buat halamannya juga yaitu:

1. Link ke route `/categories` yang akan mengarahkan ke halaman daftar kategori buku.

```
1 <v-btn small text to="/categories" class="blue--text">
2   All Categories <v-icon>mdi-chevron-right</v-icon>
3 </v-btn>
```

2. Link ke route `/books` yang akan mengarahkan ke halaman daftar buku.

```
1 <v-btn small text to="/categories" class="blue--text">
2   All Books <v-icon>mdi-chevron-right</v-icon>
3 </v-btn>
```

3. Link ke route `/category/slug` yang akan mengarahkan ke halaman detail kategori buku.

```
1 <v-card :to="'/category/'+ category.slug">
```

4. Link ke route `/book/slug` yang akan mengarahkan ke halaman detail buku

```
1 <v-card :to="'/book/'+ book.slug">
```

Oleh karena itu selanjutnya kita akan membuat ke empat halaman tersebut.

Membuat Halaman Kategori Buku

Halaman ini menampilkan semua kategori buku dalam bentuk list, di mana pada setiap itemnya ditampilkan gambar dari kategori teks judul dan deskripsinya. Supaya tidak terlalu panjang maka akan ditampilkan menggunakan paging per 6 item kategori buku.

Halaman ini dapat diakses melalui link `All Categories` pada halaman home.

Kode Vue untuk halaman daftar kategori buku ini akan disimpan dalam file `Categories.vue` pada direktori `/src/views`

Endpoint Categories

Kembali ke projek Laravel. Kita akan membuat endpoint `Categories` yang sifatnya publik, caranya: tambahkan fungsi `index()` pada controller `Category` yang sebelumnya telah kita buat.

```
1 public function index()
2 {
3     $criteria = Category::paginate(6);
4     return new CategoryResourceCollection($criteria);
5 }
```

Fungsi ini akan mengembalikan data semua category dan jumlah yang ditampilkan perhalamannya 6 record dengan format data resource.

Langkah selanjutnya, fungsi `index` pada controller `Category` tersebut kita daftarkan pada router `api.php`.

```

1 <?php
2 use Illuminate\Http\Request;
3
4 Route::prefix('v1')->group(function () {
5     // public
6     // .. route lain
7
8     Route::get('categories/random/{count}', 'CategoryController@random');
9     Route::get('categories', 'CategoryController@index'); // ini
10
11    Route::get('books/top/{count}', 'BookController@top');
12
13    // private
14    Route::middleware(['auth:api'])->group(function () {
15        Route::post('logout', 'AuthController@logout');
16    });
17 });

```

Mari kita uji coba routing `http://larashop-api.test/v1/categories` menggunakan postman. Maka hasilnya kurang lebih sebagai berikut.

```

1 {
2     "status": "success",
3     "message": "categories data",
4     "data": [
5         {
6             "id": 1,
7             "name": "quia",
8             "slug": "quia",
9             "image": "c66e5cd9fcaff2c156e8a115317c9206.jpg",
10            "status": "PUBLISH",
11            "created_at": "2018-08-28 07:00:40",
12            "updated_at": null,
13            "deleted_at": null,
14            "created_by": null,
15            "updated_by": null,
16            "deleted_by": null
17        },
18        {
19            "id": 2,
20            "name": "reprehenderit",
21            ...
22        },
23        {
24            "id": 3,
25            "name": "molestias",
26            ...
27        },
28        {
29            "id": 4,
30            "name": "quidem",
31            ...

```

```

32 },
33 {
34     "id": 5,
35     "name": "accusamus",
36     ...
37 },
38 {
39     "id": 6,
40     "name": "iste",
41     ...
42 }
43 ],
44 "links": {
45     "first": "http://larashop-api.test/v1/categories?page=1",
46     "last": "http://larashop-api.test/v1/categories?page=2",
47     "prev": null,
48     "next": "http://larashop-api.test/v1/categories?page=2"
49 },
50 "meta": {
51     "current_page": 1,
52     "from": 1,
53     "last_page": 2,
54     "path": "http://larashop-api.test/v1/categories",
55     "per_page": 6,
56     "to": 6,
57     "total": 8
58 }
59 }

```

Yang menarik pada respons ini adalah adanya informasi tentang paging pada key meta yang bermanfaat bagi kita untuk menyusun paging di frontend.

Template & Script

Kembali ke Projek Vue. Pada halaman category ini akan kita tampilkan daftar kategori buku. Component yang akan digunakan tidak jauh beda dengan component yang digunakan pada halaman Home.

Berikut ini kode template `src/views/Categories.vue`

```

1 <template>
2   <div>
3     <v-container class="ma-0 pa-0" grid-list-sm>
4       <v-subheader>
5         All Category
6       </v-subheader>
7       <v-layout wrap>
8         <v-flex v-for="(category) in categories" :key="`category-
9           +category.id`" xs6>
10          <v-card :to="/category/'+ category.slug">
11            <v-img
12              :src="getImage('/categories/'+category.image)"
13              class="white--text"
14            >
15              <v-card-title
16                class="fill-height align-end"

```

```

17         v-text="category.name"
18     ></v-card-title>
19     </v-img>
20   </v-card>
21 </v-flex>
22 </v-layout>
23 </v-container>
24 </div>
</template>

```

Sebagai mana pada halaman home yang menampilkan data random category, maka pada halaman ini kita juga akan melakukan hal yang sama namun beda endpoint saja.

```

1 export default {
2   data: () => ({
3     categories: [],
4   }),
5   created(){
6     let url = '/categories?page=1'
7     this.axios.get(url)
8       .then((response) => {
9         let { data } = response.data
10        this.categories = data
11      })
12       .catch((error) => {
13         let { responses } = error
14         console.log(responses)
15       })
16     }
17   }
}

```

Bagaimana jika data kategori buku lebih dari enam? maka kita perlu menampilkan paging untuk berpindah ke enam data berikutnya dan seterusnya. Nah untungnya, vuetyf memiliki component untuk pagination yaitu `v-pagination`.

Component ini membutuhkan atribut `v-model` yang nilainya `page` yang sedang diakses, kemudian atribut `length` yang berisi jumlah `page`, atribut `total-visible` yang berisi jumlah maksimal button paging yang ditampilkan, serta atribut `@input` berisi method yang akan dieksekusi ketika button paging diklik.

Untungnya lagi, endpoint dari laravel mengembalikan meta yang dibutuhkan untuk component `v-pagination`.

```

1 "meta": {
2   "current_page": 1,
3   "from": 1,
4   "last_page": 2,
5   "path": "http://larashop-api.test/v1/categories",
6   "per_page": 6,
7   "to": 6,
8   "total": 8
9 }

```

Jadi, nilai atribut `v-model` sama dengan `meta.current_page`, dan atribut `length` sama dengan `meta.last_page`.

Mari kita siapkan scriptnya, di mana kita tambahkan dua properti data yaitu `page` dan `lengthPage` yang dua duanya kita berikan nilai awal nol. Kemudian proses pengaksesan data via axios kita menjadi sebuah method hal ini karena akan digunakan juga oleh component `v-pagination`. pada atribut `@input`.

```

1  export default {
2    data: () => ({
3      categories: [],
4      page: 0,
5      lengthPage: 0
6    }),
7    created(){
8      this.go()
9    },
10   methods: {
11     go(){
12       let url = '/categories?page=' + this.page
13       this.axios.get(url)
14       .then((response) => {
15         let { data } = response.data
16         let { meta } = response.data
17         this.categories = data
18         this.page = meta.current_page
19         this.lengthPage = meta.last_page
20       })
21       .catch((error) => {
22         let { responses } = error
23         console.log(responses)
24       })
25     }
26   }
27 };

```

Nah kemudian pada template, kita tambahkan component `v-pagination`.

```

1 <v-pagination
2   v-model="page"
3   @input="go"
4   :length="lengthPage"
5   :total-visible="5"
6   >
7 </v-pagination>

```

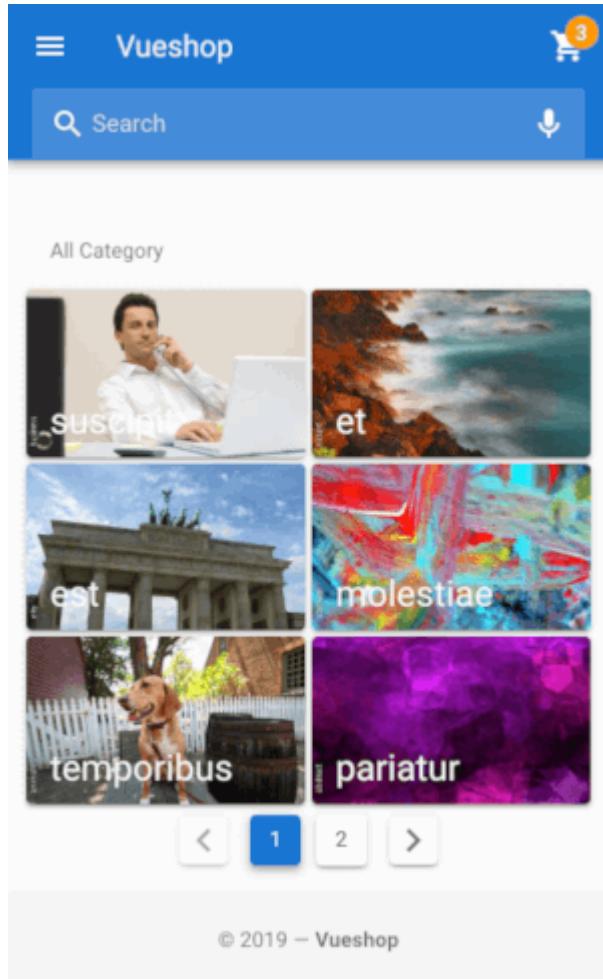
Di mana atribut `v-model` `page` menunjukkan halaman saat ini, `@input` untuk membind event click link paging yang pada contoh ini akan menjalankan method `go()`. Atribut `length` untuk menentukan jumlah total page, serta `total-visible` menunjukkan maksimal link paging yang akan ditampilkan (pada contoh ini kita set 5).

Mendaftarkan Route Categories

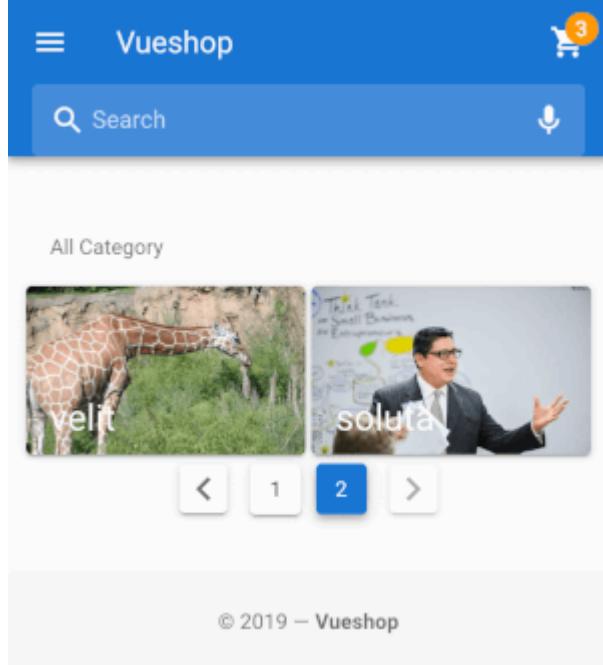
Halaman categories telah kita buat namun belum bisa di akses karena belum mempunyai route. Oleh karena itu kita perlu daftarkan pada file `'src/router.js'`. Kita bisa gunakan teknis lazy loading berikut.

```
1  {
2    path: '/categories',
3    name: 'categories',
4    component: () => import( /* webpackChunkName: "categories" */ 
5      './views/Categories.vue')
6  },
```

Jika sudah maka kita bisa ujicoba melalui browser. Berikut ini hasilnya ketika kita akses <http://localhost:8080/categories>



Ketika paging kedua kita klik maka hasilnya sebagai berikut.



Mudah sekali bukan?

Membuat Header Pada Parent vs Child

Kalau kita perhatikan bahwa halaman home dengan halaman selain home yang merupakan childnya terkesan sama, kita sebenarnya bisa menghadirkan kesan yang berbeda sehingga tidak membuat user bingung.

Salah satu yang umum digunakan adalah dengan membedakan tampilan header, di mana pada halaman lain yang merupakan anak dari halaman utama ditambahkan button back atau panah ke kiri.

Lalu bagaimana mengidentifikasi apakah suatu halaman itu home atau bukan? Karena yang akan kita ubah secara dinamis kan tampilan header yang notabene terdapat pada `src/App.vue`?

Untuk melakukannya, caranya cukup simple yaitu mendeksnya melalui route. Karena route home adalah / maka kita bisa gunakan kode ini

```
1 | (this.$route.path==='/')
```

Adapun pengecekan ini bisa kita letakkan pada properti `computed()` yaitu properti yang akan selalu dimonitor perubahannya.

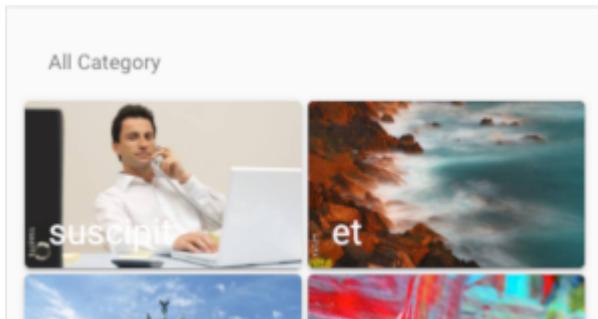
Pada file `src/App.vue` tambahkan fungsi `isHome()` pada properti `computed`. Fungsi ini akan mengembalikan nilai hasil evaluasi dari kode `(this.$route.path==='/')`

```
1 | computed: {
2 |   isHome () {
3 |     return (this.$route.path==='/')
4 |   },
5 | }
```

Kemudian pada component `v-app-bar` tambahkan atribut `v-if` yang nilainya adalah `isHome`

```
1 | <v-app-bar app color="primary" dark v-if="isHome">
```

Hasilnya pada halaman kategori buku, header aplikasi tidak ditampilkan.

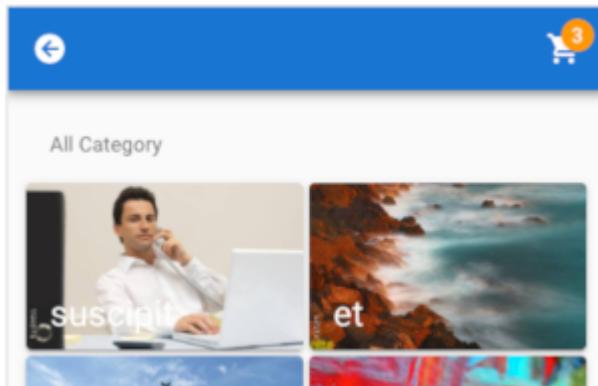


Nah, kita perlu buatkan header lagi yang berbeda dari header utama. Header ini hanya menampilkan tombol back (panah kiri) jika halaman bukan home dan apabila diklik akan menjalankan perintah `this.$router.go(-1)` yaitu kembali ke history halaman sebelumnya.

```

1 <v-app-bar app color="primary" dark v-else>
2   <v-btn icon @click.stop="$router.go(-1)">
3     <v-icon>mdi-arrow-left-circle</v-icon>
4   </v-btn>
5   <v-spacer></v-spacer>
6   <v-btn icon to="/about">
7     <v-badge color="orange" overlap>
8       <template v-slot:badge>
9         <span>3</span>
10      </template>
11      <v-icon>mdi-cart</v-icon>
12    </v-badge>
13  </v-btn>
14 </v-app-bar>
```

Jika kita jalankan maka sekarang header halaman selain home akan berbeda.



Silakan bereksplorasi.

Membuat Halaman Buku

Halaman ini menampilkan daftar semua buku dalam bentuk list, di mana pada setiap itemnya ditampilkan gambar cover dari buku, dan teks judul. Supaya tidak terlalu panjang maka juga akan ditampilkan menggunakan paging per 6 item buku.

Halaman ini dapat diakses melalui link [All Books](#) pada halaman home.

Kode Vue untuk halaman daftar kategori buku ini akan disimpan dalam file `Books.vue` pada direktori `/src/views`

Endpoint Book

Pada projek Laravel, kita akan membuat endpoint book yang sifatnya publik. Langkah yang harus kita lakukan adalah menambahkan fungsi `index()` pada controller Book yang sebelumnya telah kita buat.

```

1 public function index()
2 {
3     $criteria = Book::paginate(6);
4     return new BookResourceCollection($criteria);
5 }
```

Fungsi ini akan mengembalikan data semua book dan jumlah yang ditampilkan per halamannya 6 record dengan format data resource.

Langkah selanjutnya, fungsi index pada controller Book tersebut kita daftarkan pada router `api.php`.

```

1 <?php
2 use Illuminate\Http\Request;
3
4 Route::prefix('v1')->group(function () {
5     // public
6     Route::post('login', 'AuthController@login');
7     Route::post('register', 'AuthController@register');
8
9     Route::get('categories/random/{count}', 'CategoryController@random');
10    Route::get('categories', 'CategoryController@index');
11
12    Route::get('books/top/{count}', 'BookController@top');
13    Route::get('books', 'BookController@index'); // <== ini
14
15    // private
16    Route::middleware(['auth:api'])->group(function () {
17        Route::post('logout', 'AuthController@logout');
18    });
19});
```

Mari kita uji coba routing `http://larashop-api.test/v1/books` menggunakan postman. Maka hasilnya kurang lebih sebagai berikut.

```

1 {
2     "status": "success",
3     "message": "books data",
4     "data": [
5         {
6             "id": 1,
7             "title": "Quaerat et libero nisi",
8             "slug": "quaerat-et-libero-nisi",
9             "description": "Et officia vel unde qui enim voluptate
10            quidem. Voluptatem nemo corporis et repudiandae qui voluptas ullam. Optio
11            sit accusamus libero nostrum maiores placeat labore suscipit. Ducimus
12            velit fugit tempora ipsa temporibus.",
13            "author": "Jessie Barton",
14            "publisher": "Lesch PLC",
15            "cover": "86d8dcf0e10fa269f91eb320f8edd3c4.jpg",
```

```

16     "price": 350000,
17     "weight": 0.5,
18     "views": 0,
19     "stock": 0,
20     "status": "PUBLISH",
21     "created_at": "2018-08-28 06:59:30",
22     "updated_at": "2018-09-01 22:46:51",
23     "deleted_at": null,
24     "created_by": null,
25     "updated_by": null,
26     "deleted_by": null
27   },
28   {
29     "id": 2,
30     ...
31   },
32   {
33     "id": 3,
34     ...
35   },
36   {
37     "id": 4,
38     ...
39   },
40   {
41     "id": 5,
42     ...
43   },
44   {
45     "id": 6,
46     ...
47   }
48 ],
49 "links": {
50   "first": "http://larashop-api.test/v1/books?page=1",
51   "last": "http://larashop-api.test/v1/books?page=5",
52   "prev": null,
53   "next": "http://larashop-api.test/v1/books?page=2"
54 },
55 "meta": {
56   "current_page": 1,
57   "from": 1,
58   "last_page": 5,
59   "path": "http://larashop-api.test/v1/books",
60   "per_page": 6,
61   "to": 6,
62   "total": 25
63 }
64 }
```

Sebagaimana respon pada endpoint kategori buku maka pada endpoint ini juga mengembalikan informasi tentang paging pada key meta yang bermanfaat bagi kita untuk menyusun paging di frontend.

Template & Script

Kembali ke projek Vue, pada halaman books ini akan kita tampilkan daftar buku. Struktur template yang akan digunakan hampir sama dengan struktur pada halaman Categories.vue.

Berikut ini kode template Books.vue

```

1 <template>
2   <div>
3     <v-container class="ma-0 pa-0" grid-list-sm>
4       <v-subheader>
5         All Books
6       </v-subheader>
7       <v-layout wrap>
8         <v-flex v-for="(book) in books" :key="`book-`+book.id" xs6>
9           <v-card :to="/book/"+ book.slug">
10             <v-img
11               :src="getImage('/books/'+book.cover)"
12               class="white--text"
13             >
14             <v-card-title
15               class="fill-height align-end"
16               v-text="book.title"
17             ></v-card-title>
18             </v-img>
19           </v-card>
20         </v-flex>
21       </v-layout>
22     </v-container>
23
24   <template>
25     <div class="text-center">
26       <v-pagination
27         v-model="page"
28         @input="go"
29         :length="lengthPage"
30         :total-visible="5"
31       ></v-pagination>
32     </div>
33   </template>
34 </div>
35 </template>
```

Berikut ini kode bagian script.

```

1 <script>
2 export default {
3   data: () => ({
4     books: [],
5     page: 0,
6     lengthPage: 0
7   }),
8   created(){
9     this.go()
10 },
```

```

11  methods: {
12    go(){
13      let url = '/books?page=' + this.page
14      this.axios.get(url)
15      .then((response) => {
16        let { data } = response.data
17        let { meta } = response.data
18        this.books = data
19        // jumlah halaman di dapat dari meta endpoint books
20        this.lengthPage = meta.last_page
21        this.page = meta.current_page
22      })
23      .catch((error) => {
24        let { responses } = error
25        console.log(responses)
26      })
27    }
28  }
29 };
30 </script>

```

Secara umum kode di atas sama dengan kode pada halaman Categories, hanya berbeda endpointnya saja.

Mendaftarkan Route Books

Pada `src/router.js`, kita tambahkan aturan router untuk component Books ini.

```

1  {
2    path: '/books',
3    name: 'books',
4    component: () => import( /* webpackChunkName: "books" */ 
5      './views/Books.vue')
6  },

```

Jika sudah maka kita bisa ujicoba melalui browser. Berikut ini hasilnya

The screenshot shows a mobile application interface for a bookstore. At the top, there is a blue header bar with a back arrow icon on the left and a shopping cart icon on the right, which has a small orange circle with the number '3' indicating items in the cart.

Below the header, the text "All Books" is displayed. The main content area consists of a grid of six book category cards, arranged in two columns and three rows. Each card features a placeholder image and text describing the category.

- Category 1:** Placeholder image of a forest path in autumn. Text: "Rerum accusamus dolores totam quis quia quisquam odit". Subtitle: "nature".
- Category 2:** Placeholder image of a purple textured surface. Text: "Dolorem dolores distinctio quidem". Subtitle: "distortion".
- Category 3:** Placeholder image of a fluffy cat's face. Text: "Suscipit repellendus saepe repellat dolor beatae fugiat". Subtitle: "cats".
- Category 4:** Placeholder image of autumn leaves. Text: "Laboriosam animi quam". Subtitle: "leaves".
- Category 5:** Placeholder image of a colorful autumn scene. Text: "Exercitationem voluptatum deserunt adipisci accusantium dolorem". Subtitle: "fall".
- Category 6:** Placeholder image of a landscape with a lone tree. Text: "Enim cum laudantium accusantium voluptatem". Subtitle: "landscape".

At the bottom of the screen, there is a navigation bar with five numbered buttons (1, 2, 3, 4, 5) and arrows for navigating between pages. The number '1' is highlighted in blue, indicating the current page. Below the navigation bar, the copyright notice "© 2019 – Vueshop" is visible.

Membuat Halaman Detail Kategori

Halaman ini menampilkan detail kategori buku yaitu nama kategori, gambar dan deskripsinya. Di samping itu juga menampilkan daftar buku pada kategori tersebut dengan format paging.

Halaman ini dapat diakses melalui

- link category pada halaman home

- link category pada halaman categories

Kode vue untuk halaman detail kategori buku ini akan disimpan dalam file `Category.vue` pada direktori `/src/views`

Endpoint Detail Category

Pada projek Laravel, kita akan membuat endpoint untuk detail category yang sifatnya publik. Selain menampilkan data detail category, endpoint ini diharapkan juga dapat menampilkan data books yang memiliki category tersebut. Jika kita merujuk pada desain database maka relasi antara tabel category dan tabel book sifatnya many to many melalui tabel perantara yaitu `book_category`.

Untuk kasus relasi many to many ini, Laravel mempunyai pendekatan pivot tabel yang akan menyederhanakan prosesnya. Pertama yang harus kita lakukan adalah mendefinisikan relasi antara tabel category dengan tabel book, caranya pada model `App\Category`, definisikan relasinya dengan menambahkan fungsi `books()`.

```
1 public function books(){
2     return $this->belongsToMany("App\Book");
3 }
```

Yap hanya ini, sebab Laravel telah membuat konvensi bahwa tabel pivot atau perantara itu harusnya memiliki nama gabungan dari dua tabel yang berelasi dalam bentuk singular. Karena tabel book dan tabel category, maka pivot tabelnya adalah `book_category` (menggunakan urutan abjad sehingga bukan `category_book`). Sehingga ketika kita mendefinisikan dengan kode di atas maka Laravel otomatis tau, kemana dia harus merujuk.

Bagaimana jika penamaan tabel kita tidak sesuai konvensi tersebut? maka kita definikan tiga parameter berikutnya. Parameter kedua nama tabel pivot, parameter ketiga nama field pada tabel pivot yang menghubungkan tabel Category, dan parameter keempat nama field pada tabel pivot yang menghubungkan tabel Book. Contohnya sebagai berikut:

```
1 public function books(){
2     return $this->belongsToMany('App\Book', 'book_category',
3         'category_id', 'book_id');
4 }
```

Jika sudah, maka kita perlu buat dulu resource untuk model Category.

```
1 php artisan make:resource Category
```

Perintah ini akan menggenerate file `Category.php` pada direktori `app/Http/Resources`

Modifikasi fungsi `toArray()` menjadi sebagai berikut supaya outputnya standard (status, message, data).

```
1 public function toArray($request)
2 {
3     $parent = parent::toArray($request);
4     // ambil data books yang berelasi dengan category menggunakan pivot
5     $data['books'] = $this->books()->paginate(6);
6     $data = array_merge($parent, $data);
7     return [
8         'status'    => 'success',
9         'message'   => 'category data',
```

```

10     'data'      => $data
11 ];
12 }
```

Kemudian tambahkan fungsi `slug()` pada controller Category yang sebelumnya telah kita buat.

```

1 //...
2 use App\Http\Resources\Category as CategoryResource;
3
4 //...
5
6 public function slug($slug)
7 {
8     $criteria = Category::where('slug', $slug)->first();
9     return new CategoryResource($criteria);
10}
```

Fungsi ini akan mengembalikan data detail category berdasarkan slug-nya serta data books yang berelasi dan jumlah yang ditampilkan perhalamannya 6 record dengan format data resource.

Langkah selanjutnya, fungsi `slug` pada controller Category tersebut kita daftarkan pada router `api.php`.

```

1 <?php
2 use Illuminate\Http\Request;
3
4 Route::prefix('v1')->group(function () {
5     // public
6     Route::post('login', 'AuthController@login');
7     Route::post('register', 'AuthController@register');
8
9     Route::get('categories/random/{count}', 'CategoryController@random');
10    Route::get('categories', 'CategoryController@index');
11    Route::get('categories/slug/{slug}', 'CategoryController@slug'); //
12    <= ini
13
14    Route::get('books/top/{count}', 'BookController@top');
15    Route::get('books', 'BookController@index');
16
17    // private
18    Route::middleware(['auth:api'])->group(function () {
19        Route::post('logout', 'AuthController@logout');
20    });
});
```

Mari kita uji coba routing `http://larashop-api.test/v1/categories/slug/quia` (misal slugnya adalah `quia`) menggunakan postman. Maka hasilnya kurang lebih sebagai berikut.

```

1 {
2     "status": "success",
3     "message": "category data",
4     "data": {
5         "id": 1,
```

```

6      "name": "quia",
7      "slug": "quia",
8      "image": "c66e5cd9fcaff2c156e8a115317c9206.jpg",
9      "status": "PUBLISH",
10     "created_at": "2018-08-28 07:00:40",
11     "updated_at": null,
12     "deleted_at": null,
13     "created_by": null,
14     "updated_by": null,
15     "deleted_by": null,
16     "books": {
17         "current_page": 1,
18         "data": [
19             {
20                 "id": 1,
21                 "title": "Quaerat et libero nisi",
22                 "slug": "quaerat-et-libero-nisi",
23                 "description": "Et officia vel unde qui enim
24                 voluptate quidem. Voluptatem nemo corporis et repudiandae qui voluptas
25                 ullam. Optio sit accusamus libero nostrum maiores placeat labore
26                 suscipit. Ducimus velit fugit tempora ipsa temporibus.",
27                 "author": "Jessie Barton",
28                 "publisher": "Lesch PLC",
29                 "cover": "86d8dcf0e10fa269f91eb320f8edd3c4.jpg",
30                 "price": 350000,
31                 "weight": 0.5,
32                 "views": 0,
33                 "stock": 0,
34                 "status": "PUBLISH",
35                 "created_at": "2018-08-28 06:59:30",
36                 "updated_at": "2018-09-01 22:46:51",
37                 "deleted_at": null,
38                 "created_by": null,
39                 "updated_by": null,
40                 "deleted_by": null,
41                 "pivot": {
42                     "category_id": 1,
43                     "book_id": 1
44                 }
45             },
46             {
47                 "id": 2,
48                 "title": "Officiis assumenda unde",
49                 ...
50             }
51         ],
52         "first_page_url": "http://larashop-
53         api.test/v1/categories/slug/quia?page=1",
54         "from": 1,
55         "last_page": 1,
56         "last_page_url": "http://larashop-
57         api.test/v1/categories/slug/quia?page=1",
58         "next_page_url": null,
59         "path": "http://larashop-api.test/v1/categories/slug/quia",

```

```

60     "per_page": 6,
61     "prev_page_url": null,
62     "to": 2,
63     "total": 2
64   }
65 }
66

```

Template & Script

Kembali ke projek Vue, buka file `src/views/Category.vue`, kita akan buat mirip dengan `views/Categories` & `Books`, bedanya adalah data kategori buku berupa objek tunggal sedangkan data daftar buku berupa list/array dari objek. Dua data tersebut dihasilkan dari satu endpoint yang telah kita buat pada bagian sebelumnya (`/category/slug/nama-slugnya`).

Pertama, kita akan buat scriptnya dahulu dengan mendaftarkan empat properti data yaitu `category` yang bertipe objek untuk menyimpan data kategori buku yang terpilih, `books` untuk menyimpan data list buku yang terdapat pada kategori buku, `page` untuk menyimpan posisi halaman saat ini dan `lengthPage` untuk menyimpan jumlah halaman buku pada kategori tersebut yang ditampilkan 6 buku perhalaman.

Kita juga akan jadikan prosedur akses datanya sebagai method supaya bisa digunakan ulang ketika meminta data halaman berikutnya pada component `v-pagination`.

```

1 <script>
2 export default {
3   data: () => ({
4     category: {}, // objek category
5     books: [], // daftar buku pada category tersebut
6     page: 0,
7     lengthPage: 0
8   }),
9   created(){
10    this.go()
11  },
12   methods: {
13     go(){
14       let { slug } = this.$route.params
15       let url = '/categories/slug/'+slug
16       url = url + '?page=' + this.page
17       url = encodeURI(url)
18       this.axios.get(url)
19         .then((response) => {
20           let { data } = response.data
21           let category = data
22           this.category = category
23           this.books = category.books.data
24           this.page = category.books.current_page
25           this.lengthPage = category.books.last_page
26         })
27         .catch((error) => {
28           let { responses } = error
29           console.log(responses)
30         })
31     }
32   }

```

```

32     }
33   };
34 </script>

```

Setelah scriptnya siap maka kita petakan ke template.

```

1  <template>
2    <div>
3      <v-card :to="'/category/'+ category.slug" v-if="category.slug">
4        <v-img
5          :src="getImage('/categories/'+category.image)"
6          class="white--text"
7        >
8          <v-card-title
9            class="fill-height align-end"
10           v-text="category.name"
11         ></v-card-title>
12       </v-img>
13     </v-card>
14
15     <v-container class="ma-0 pa-0" grid-list-sm v-if="books">
16       <v-subheader>
17         All Books
18       </v-subheader>
19       <v-layout wrap>
20         <v-flex v-for="(book) in books" :key="`book-`+book.id" xs6>
21           <v-card :to="'/book/'+ book.slug">
22             <v-img
23               :src="getImage('/books/'+book.cover)"
24               class="white--text"
25             >
26               <v-card-title
27                 class="fill-height align-end"
28                 v-text="book.title"
29               ></v-card-title>
30             </v-img>
31           </v-card>
32         </v-flex>
33       </v-layout>
34     </v-container>
35     <template>
36       <div class="text-center">
37         <v-pagination
38           v-model="page"
39           @input="go"
40           :length="lengthPage"
41           :total-visible="5"
42         ></v-pagination>
43       </div>
44     </template>
45   </div>
46 </template>

```

Untuk menghindari error karena data kategori yang belum siap digunakan atau masih loading dari endpoint maka kita bisa menambahkan directive `v-if` pada wrapper yang menggunakan data kategori (component `v-card`). Pada contoh ini directive `v-if` diisi dengan `category.slug` untuk memastikan bahwa data kategori telah siap digunakan ketika component `v-card` yang dirender. Selebihnya, sama dengan kode pada bagian sebelumnya.

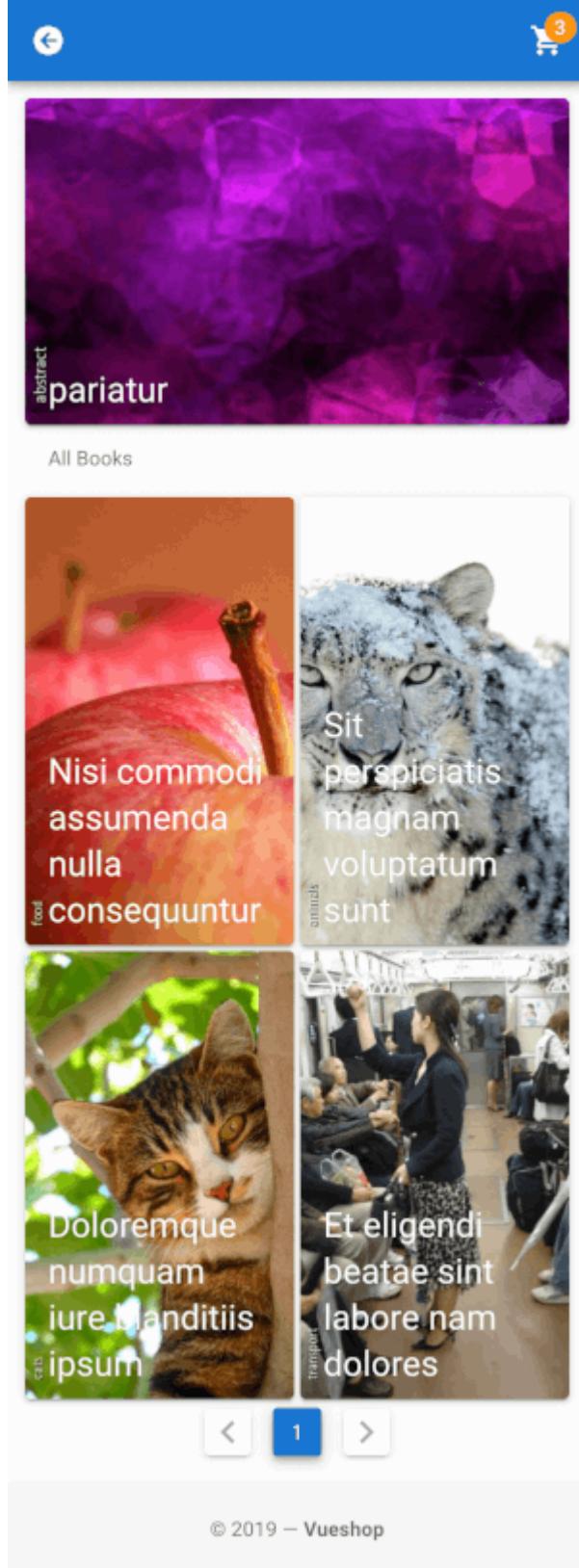
Mendaftarkan Route Detail Category

Pada `src/router.js`, kita tambahkan aturan router untuk component Category ini. Kita tambahkan parameter `slug` pada path.

```
1  {
2    path: '/category/:slug',
3    name: 'category',
4    component: () => import( /* webpackChunkName: "category" */ 
5      './views/Category.vue')
6  },
```

Jika sudah maka kita bisa ujicoba melalui browser, misalnya URL <http://localhost:8080/category/quia>.

Berikut ini hasilnya



Membuat Halaman Detail Buku

Halaman ini menampilkan data buku, yaitu gambar cover dari buku, teks judul dan harga buku. Pada bagian bawah terdapat halaman tombol **Buy** yang berfungsi menyimpan buku tersebut ke keranjang belanja serta mengarahkan user ke halaman keranjang belanja.

Halaman ini dapat diakses melalui link buku pada halaman home, daftar buku, detail kategori dan pencarian buku. Kode Vue untuk halaman detail buku ini akan disimpan dalam file `Book.vue` pada direktori `/src/views`

Endpoint Detail Book

Pada projek Laravel, kita akan membuat endpoint detail book yang sifatnya publik. Selain menampilkan data detail book, endpoint ini diharapkan juga dapat menampilkan data category dari buku tersebut. Jika kita merujuk ke desain database maka relasi antara tabel book dan tabel category sifatnya many to many melalui tabel perantara yaitu book_category.

Sebagaimana yang telah dijelaskan sebelumnya, langkah pertama yang harus kita lakukan adalah mendefinisikan relasi antara tabel category dengan tabel book, caranya pada model App\Book, definisikan relasinya dengan menambahkan fungsi categories().

```
1 public function categories()
2 {
3     return $this->belongsToMany( 'App\Category' );
4 }
```

Jika sudah, maka kita perlu buat dulu resource untuk model Book.

```
1 php artisan make:resource Book
```

Perintah ini akan menggenerate file Book.php pada direktori app/Http/Resources. Lakukan modifikasi fungsi toArray() menjadi sebagai berikut supaya outputnya standard (status, message, data).

```
1 public function toArray($request)
2 {
3     $parent = parent::toArray($request);
4     $data['categories'] = $this->categories;
5     $data = array_merge($parent, $data);
6     return [
7         'status' => 'success',
8         'message' => 'book data',
9         'data' => $data,
10    ];
11 }
```

Kemudian tambahkan fungsi slug() pada controller Book yang sebelumnya telah kita buat.

```
1 //...
2 use App\Http\Resources\Book as BookResource;
3
4 //...
5
6 public function slug($slug)
7 {
8     $criteria = Book::where('slug', $slug)->first();
9     return new BookResource($criteria);
10}
```

Fungsi ini akan mengembalikan data detail books berdasarkan slug-nya serta data categories yang berelasi dengan format data resource.

Langkah selanjutnya, fungsi slug pada controller Book tersebut kita daftarkan pada router api.php.

```

1 <?php
2 use Illuminate\Http\Request;
3
4 Route::prefix('v1')->group(function () {
5     // public
6     Route::post('login', 'AuthController@login');
7     Route::post('register', 'AuthController@register');
8
9     Route::get('categories/random/{count}', 'CategoryController@random');
10    Route::get('categories', 'CategoryController@index');
11    Route::get('categories/slug/{slug}', 'CategoryController@slug');
12
13    Route::get('books/top/{count}', 'BookController@top');
14    Route::get('books', 'BookController@index');
15    Route::get('books/slug/{slug}', 'BookController@slug'); // <== ini
16
17    // private
18    Route::middleware(['auth:api'])->group(function () {
19        Route::post('logout', 'AuthController@logout');
20    });
21 });

```

Mari kita uji coba routing `http://larashop-api.test/v1/books/slug/quaerat-et-libero-nisi` (misal slugnya adalah `quaerat-et-libero-nisi`) menggunakan postman. Maka hasilnya kurang lebih sebagai berikut.

```

1 {
2     "status": "success",
3     "message": "book data",
4     "data": {
5         "id": 1,
6         "title": "Quaerat et libero nisi",
7         "slug": "quaerat-et-libero-nisi",
8         "description": "Et officia vel unde qui enim voluptate quidem.
9 Voluptatem nemo corporis et repudiandae qui voluptas ullam. Optio sit
10 accusamus libero nostrum maiores placeat labore suscipit. Ducimus velit
11 fugit tempora ipsa temporibus.",
12         "author": "Jessie Barton",
13         "publisher": "Lesch PLC",
14         "cover": "86d8dcf0e10fa269f91eb320f8edd3c4.jpg",
15         "price": 350000,
16         "weight": 0.5,
17         "views": 0,
18         "stock": 0,
19         "status": "PUBLISH",
20         "created_at": "2018-08-28 06:59:30",
21         "updated_at": "2018-09-01 22:46:51",
22         "deleted_at": null,
23         "created_by": null,
24         "updated_by": null,
25         "deleted_by": null,
26         "categories": [

```

```

27     {
28         "id": 1,
29         "name": "quia",
30         "slug": "quia",
31         "image": "c66e5cd9fcaff2c156e8a115317c9206.jpg",
32         "status": "PUBLISH",
33         "created_at": "2018-08-28 07:00:40",
34         "updated_at": null,
35         "deleted_at": null,
36         "created_by": null,
37         "updated_by": null,
38         "deleted_by": null,
39         "pivot": {
40             "book_id": 1,
41             "category_id": 1
42         }
43     },
44     {
45         "id": 2,
46         "name": "reprehenderit",
47         ...
48     },
49     {
50         "id": 3,
51         "name": "molestias",
52         ...
53     }
]
}
}

```

Ada tambahan sedikit terkait dengan endpoint ini, dimana ketika endpoint ini dipanggil maka seharusnya jumlah views terhadap buku ini bertambah. Nah oleh karena itu, kita bisa tambahkan perintah untuk mengincrement jumlah views buku ini.

Caranya mudah saja, pada method slug update menjadi sebagai berikut.

```

1 public function slug($slug)
2 {
3     $criteria = Book::where('slug', $slug)->first();
4     $criteria->views = $criteria->views + 1;
5     $criteria->save();
6     return new BookResource($criteria);
7 }

```

Template & Script

Kembali ke projek Vue, berikut ini kode template `views/Book.vue`

```

1 <template>
2   <div>
3     <v-card v-if="book.slug">
4       <v-img
5         :src="getImage('/books/' + book.cover)"

```

```

6      class="white--text"
7      height="200px"
8    >
9      <v-card-title
10         class="fill-height align-end"
11         v-text="book.title">
12       </v-card-title>
13     </v-img>
14
15   <v-card-text>
16     <v-simple-table dense>
17       <tbody>
18         <tr>
19           <td><v-icon>mdi-account-tie</v-icon> Author</td>
20           <td>{{ book.author }}</td>
21         </tr>
22         <tr>
23           <td><v-icon>mdi-bullhorn</v-icon> Publisher</td>
24           <td>{{ book.publisher }}</td>
25         </tr>
26         <tr>
27           <td><v-icon>mdi-weight</v-icon> Weight</td>
28           <td>{{ book.weight }} kg</td>
29         </tr>
30         <tr>
31           <td><v-icon>mdi-format-list-bulleted</v-icon>
32 Stock</td>
33           <td>{{ book.stock }} item</td>
34         </tr>
35         <tr>
36           <td><v-icon>mdi-eye</v-icon> Views</td>
37           <td>{{ book.views }}</td>
38         </tr>
39         <tr>
40           <td><v-icon>mdi-cash</v-icon> Price</td>
41           <td class="orange--text">Rp {{
42 book.price.toLocaleString('id-ID')}}</td>
43         </tr>
44       </tbody>
45     </v-simple-table>
46     Description: <br>
47     {{ book.description }}
48     <br>
49     Categories:
50     <v-chip v-for="category in book.categories"
51 :key="category.id" :to="'/category/'+category.slug" small>
52       {{ category.name }}
53     </v-chip>
54   </v-card-text>
55   <v-card-actions>
56     <v-btn block color="success" @click="buy"
57 :disabled="book.stock==0">
58       <v-icon>mdi-cart-plus</v-icon> &ampnbsp
59       BUY

```

```
60 |         </v-btn>
|     </v-card-actions>
|   </v-card>
| </div>
</template>
```

Untuk harga atau price kita menggunakan fungsi `.toLocaleString('id-ID')` supaya tampilannya sesuai dengan format angka di Indonesia. Adapun category yang bentuknya array ditampilkan menggunakan v-for, dimana yang diambil hanya namanya saja kemudian supaya menarik maka dibungkus dengan component `v-chip`.

Pada kode di atas, tombol "Buy" masih belum dihubungkan dengan method apapun, nantinya akan kita gunakan untuk menambahkan data buku ini ke dalam keranjang belanja.

Berikut ini kode bagian script.

```
1  <script>
2  export default {
3    data: () => ({
4      book: {}, // objek book
5    }),
6    created(){
7      this.go()
8    },
9    methods: {
10      go(){
11        let { slug } = this.$route.params
12        let url = '/books/slug/'+slug
13        url = url + '?page=' + this.page
14        url = encodeURI(url)
15        this.axios.get(url)
16        .then((response) => {
17          let { data } = response.data
18          this.book = data
19        })
20        .catch((error) => {
21          let { responses } = error
22          console.log(responses)
23        })
24      },
25      buy(){
26        alert('buy')
27      }
28    }
29  };
</script>
```

Seperti teknik sebelumnya, pada saat hook created kita gunakan untuk mengambil data pada endpoint `book/slug`. Method `buy()` yang akan dijalankan ketika user mengklik tombol `Buy` juga belum kita fungsikan sebagai mana mestinya.

Mendaftarkan Route Book

Pada 'src/router.js', kita tambahkan aturan router untuk component Book ini. Kita tambahkan parameter slug pada path.

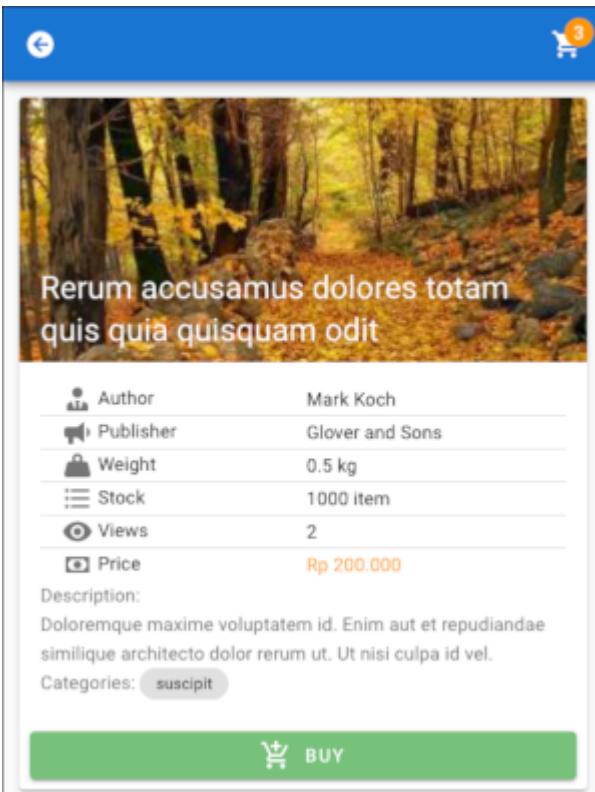
```

1  {
2    path: '/book/:slug',
3    name: 'book',
4    component: () => import( /* webpackChunkName: "book" */ 
5      './views/Book.vue')
6  },

```

Jika sudah maka kita bisa ujicoba melalui browser, misalnya slugnya `quaerat-et-libero-nisi`, maka URL detail book yang bisa diakses menjadi `http://localhost:8080/book/quaerat-et-libero-nisi`.

Hasilnya sebagai berikut.

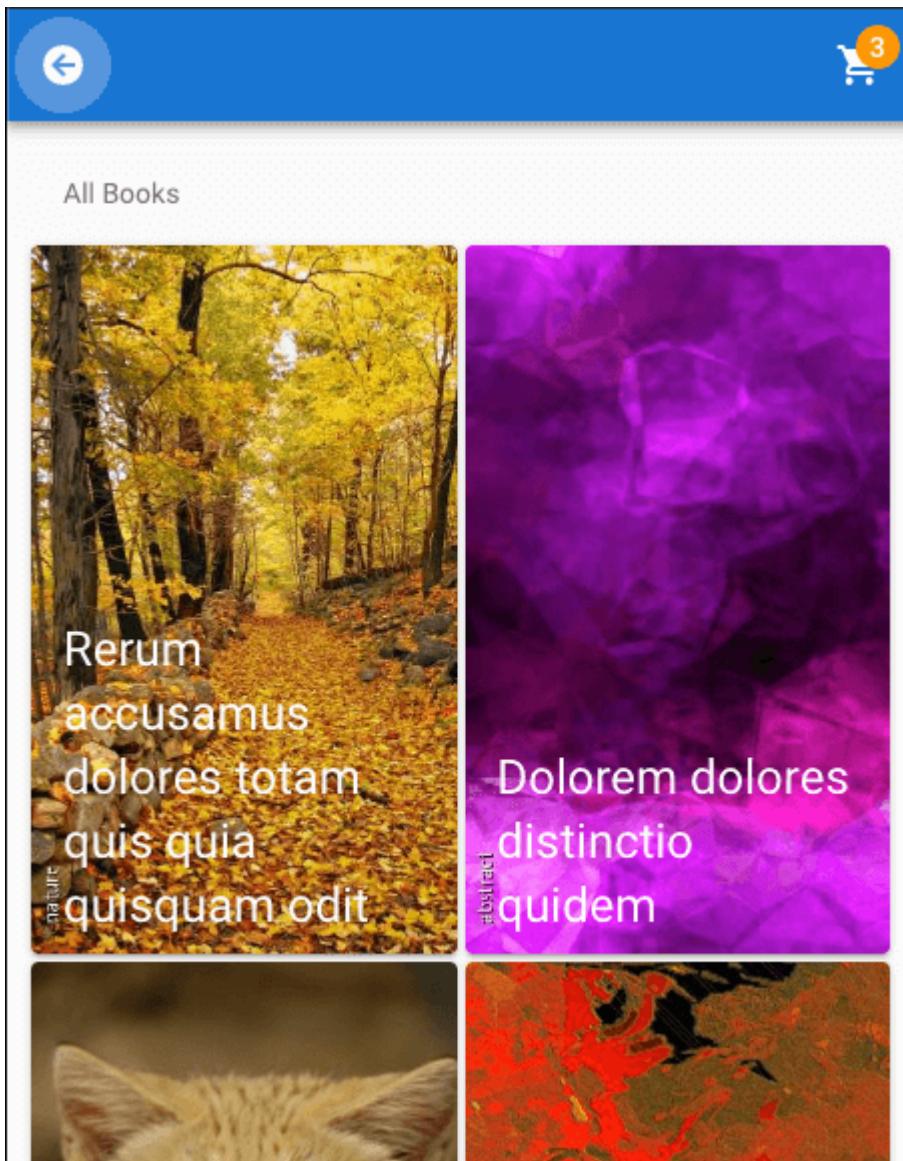


Halaman detail buku ini memang belum selesai, terutama fungsi ketika button buy diklik. Tahan dulu ya.. karena kita akan bahas yang lain dulu, jangan lupa ingetin kalau-kalau penulis lupa melanjutkan bagian ini 😊.

Membuat Reusable Component

Sebelum kita melanjutkan pembuatan halaman detail buku, penulis ingin membahas tentang bagaimana membuat sebuah reusable component atau component yang dapat digunakan secara berulang. Meskipun sebenarnya, vueify yang kita gunakan saat ini juga merupakan kumpulan reusable component 😊.

Kalau kita lihat halaman detail buku, maka ditampilkan informasi detail tentang buku tersebut mulai dari penulis, penerbit, jumlah stok hingga harga barangnya. Namun kalo kita lihat pada halaman home atau halaman books (daftar buku) di mana hanya ditampilkan cover buku dan judulnya kok penulis rasa ada yang kurang ya?



Benar saja, ketika penulis coba buka aplikasi ecommerce populer semacam Tokopedia, setidaknya mereka menampilkan harga barang, jumlah review, action like dsb.

Product	Price	Location	Rating	Action
Bape Mix X PUBG Jacket	Rp 5.000.000	Kota Batam	★★★★★ (21)	Lihat Produk Serupa
JAKET ADIDAS ZNE HOODIE WINTER RUN BLACK	Rp 160.000	Kota Bandung	★★★★★ (13)	Grosir Lihat Produk Serupa

Nah, karenanya, penulis ingin mengupdate layout bagian tersebut supaya minimal tampil harga bukunya dan jumlah viewsnya. Namun alih-alih mengedit langsung pada bagian tersebut, penulis malah berpikir mengapa

tidak sekalian dibuat component-nya? karena component ini akan dipakai dibeberapa halaman sekaligus sehingga akan lebih efisien jika dibuat component-nya.

Membuat Component BookItem

Yap, component ini akan kita berinama BookItem, pertama yang harus kita lakukan adalah membuat file component ini yaitu BookItem.vue di folder `src/components/` yang kerangkanya sebagai berikut.

```

1 <template>
2   <div>
3     <!-- layout component BookItem -->
4     BookItem
5   </div>
6 </template>
7 <script>
8   export default {
9     name: 'book-item',
10    data: () => ({
11      }),
12    }
13 </script>
```

Dengan kode seperti ini maka component lain yang ingin menggunakan tinggal mengimport file ini dan mendefinisikan component ini pada template dengan kode `<book-item />`.

Baiklah, mari kita buat layoutnya yang kurang lebih sama dengan item buku di halaman home (copy pada bagian v-card)

```

1 <template>
2   <div>
3     <v-card :to="'/book/' + book.slug">
4       <v-img
5         :src="getImage('/books/' + book.cover)"
6         class="white--text"
7       >
8         <v-card-title
9           class="fill-height align-end"
10          v-text="book.title"
11        ></v-card-title>
12       </v-img>
13     </v-card>
14   </div>
15 </template>
16 <script>
17   export default {
18     name: 'book-item',
19     props: ['book'],
20     data: () => ({
21       }),
22     }
23 </script>
```

Nah, karena component ini menggunakan data book dari component lain yang akan menjadi induk (parent) dari component ini maka kita bisa gunakan properti `props` yaitu properti yang digunakan untuk mengirimkan

data dari parent ke child componentnya.

Kenapa tidak didefinisikan via properti data saja? yap karena properti data tidak bisa diakses dari luar component dan memang kebutuhannya component ini hanya digunakan untuk merender tampilan yang datanya berasal dari parentnya.

Kemudian mari kita modifikasi sedikit layoutnya supaya lebih oke lagi. Pada cover bukunya kita batasi tinggi 200px, dan kita tambahkan dua informasi yaitu harga buku dan jumlah views. Sehingga kodenya menjadi sebagai berikut.

```

1 <template>
2   <div>
3     <v-card :to="'/book/' + book.slug">
4       <v-img
5         :src="getImage('/books/' + book.cover)"
6         class="white--text"
7         height="200px"
8       >
9       <v-card-title
10        class="fill-height align-end"
11        v-text="book.title"
12      ></v-card-title>
13    </v-img>
14
15    <v-card-actions>
16      <v-icon>mdi-cash</v-icon>
17      <span class="orange--text"> Rp {{ book.price.toLocaleString('id-ID') }}</span>
18
19      <v-spacer></v-spacer>
20      <v-icon>mdi-eye</v-icon> {{ book.views }}</v-card-actions>
21
22  </v-card>
23 </div>
24 </template>
```

Menggunakan Component BookItem

Untuk menggunakan component BookItem yang telah kita buat sebelumnya caranya cukup sederhana yaitu dengan kode `<book-item :book="book" />` atau bisa juga dengan `<BookItem :book="book" />`. Adapun book adalah nama props, :book artinya membinding atribut props book hingga variabel book ada nilainya.

Mari kita implementasikan pada halaman home (`src/views/Home.vue`).

Pada bagian templatenya saat kita mengiterasi variabel books, kita ganti v-card dengan component BookItem.

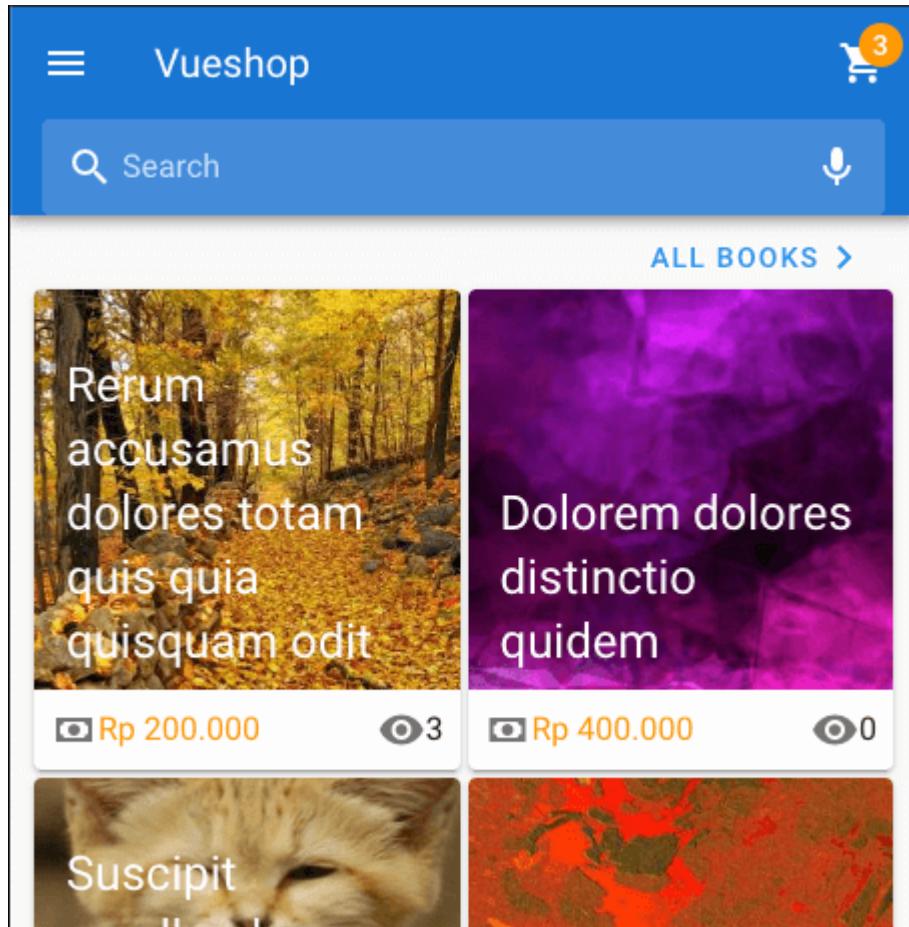
```

1 <v-layout wrap>
2   <v-flex v-for="(book) in books" :key="`book-`+book.id" xs6>
3     <book-item :book="book" />
4   </v-flex>
5 </v-layout>
```

Lalu pada script, kita definisikan dan panggil component BookItem dengan teknik lazy loading component.

```
1 export default {
2   components: {
3     BookItem: () => import(/* webpackChunkName: "book-item" */ 
4     '@/components/BookItem.vue')
5   },
6 }
```

Tidak ada perubahan pada kode selain itu, kemudian lihat hasilnya.



Oke seep.

Kamu bisa lakukan hal ini untuk halaman lain yang menggunakan seperti misalnya di halaman daftar buku dan halaman detail kategori.

Implementasi State Management

Hingga sampai disini, kita belum menerapkan state (data) management pada aplikasi kita, dan sepertinya tidak ada masalah alias jalan jalan saja tanpa state management, lalu apa masalahnya hingga kita harus menggunakananya?

Baik, apa yang telah kita buat memang telah menggunakan data, namun data yang kita gunakan tersebut hanya digunakan dalam scope component tersebut saja alias belum sharing dengan component lain. Pada bagian sebelumnya yaitu reusable component memang telah sharing namun hanya sharing satu arah dari component parent ke childnya.

Nah pada bagian ini, kita akan coba menggunakan data yang dapat digunakan secara sharing oleh beberapa component sekaligus. Dan tidak ada pilihan lain yang lebih mudah dan efisien selain kita menerapkan state management yang dalam hal ini menggunakan pustaka resmi state management yaitu Vuex.

Salah satu data yang perlu disimpan dalam Vuex adalah data keranjang belanja atau cart. Di mana data ini harus bisa di tampilkan atau diakses dari beberapa component dan dapat dimodifikasi dari beberapa component yang lain. Sebagai contoh, pada halaman detail buku, tombol `buy` seharusnya berfungsi untuk menambahkan data buku tersebut ke dalam keranjang belanja. Contoh lain, pada halaman checkout, data belanja harus dapat ditampilkan untuk mengkalkulasi total pembayarannya, serta pada halaman home, harus dapat ditampilkan data total belanja.

Mendefinisikan State

Pada bagian pertama inikita akan mendefinisikan state keranjang belanja yang akan kita berinama `carts` (bentuk jamak dari kata `cart`). State ini bertipe list atau array dari objek, kira-kira bentuknya sebagai berikut:

```

1  [
2      { id: 1, quantity: 2, title: 'abc', ... },
3      { id: 2, quantity: 1, title: 'xyz', ... },
4      //...
5  ]

```

Setiap itemnya merupakan objek buku dengan tambahan key `quantity` untuk menyimpan jumlah item yang dibeli pada produk buku tersebut.

Buka file `src/store.js`, kita tambahkan pada variabel `carts` pada bagian state, lalu tambahkan getter untuk state tersebut.

```

1  export default new Vuex.Store({
2      state: {
3          carts : [],
4      },
5      mutations: {
6
7      },
8      actions: {
9
10     },
11     getters: {
12         carts : state => state.carts
13     }
14 })

```

Dengan deklarasi ini maka, kita sudah bisa mengakses state `carts` dari component manapun melalui perintah `this.$store.state.carts` atau `this.$store.getters.carts`.

Cara lain yang bisa kita gunakan untuk mengakses state pada component adalah dengan memapping state tersebut dengan menggunakan `mapGetters` yang kita letakkan pada properti `computed`, sebagai berikut:

```

1  computed: {
2      ...mapGetters({
3          'carts': 'carts'
4      }),
5  },

```

Namun dengan sebelumnya melakukan import fungsi `mapGetters` dari vuex `import { mapGetters } from 'vuex'`. Sehingga pada component tersebut, state bisa kita akses layaknya properti data

```
this.carts.
```

Mendefinisikan Mutation & Action Pada State

Selain dapat diakses, sebuah state juga seharusnya bisa dimodifikasi datanya, entah itu ditambah, didelet maupun dihapus. Untuk melakukan modifikasi state tersebut maka Vuex menyediakan properti mutation.

Pada konteks keranjang belanja atau state carts terdapat beberapa kemungkinan perubahan state yaitu ketika user memesan buku (insert), ketika user mengubah jumlah pesan atau memesan barang yang sama (update), dan ketika user menghapus atau membatalkan pemesanan (update).

Pada mutation kita tambahkan dua fungsi tadi yaitu insert dan update.

```

1  mutations: {
2      insert: (state, payload) => {
3          state.carts.push({
4              id: payload.id,
5              title: payload.title,
6              cover: payload.cover,
7              price: payload.price,
8              weight: payload.weight,
9              quantity: 1
10         })
11     },
12     update: (state, payload) => {
13         // mendeteksi payload ada di index keberapa?
14         let idx = state.carts.indexOf(payload);
15         state.carts.splice(idx,1,{
16             id: payload.id,
17             title: payload.title,
18             cover: payload.cover,
19             price: payload.price,
20             weight: payload.weight,
21             quantity: payload.quantity
22         });
23
24         if(payload.quantity<=0){
25             // menghapus item carts jika quantity-nya nol
26             state.carts.splice(idx,1)
27         }
28     }
29 }
```

Fungsi `push` pada Js digunakan untuk menambahkan suatu data ke dalam array. Sedangkan `splice` digunakan untuk menghapus suatu item data pada array atau untuk mengupdate data pada item array tertentu. Parameter payload nantinya akan diisi dengan objek buku.

Untuk dapat mengakses mutation itu pada component maka kita perlu membuat sebuah method pada properti actions yang digunakan untuk meng-commit mutation.

```

1  actions: {
2      add: ({state, commit}, payload) => {
3          // mendeteksi apakah data yang diinput ada pada carts?
4          let cartItem = state.carts.find(item => item.id === payload.id)
5          // jika tidak ada maka mutation insert dijalankan
```

```

6   if(!cartItem){
7     commit('insert', payload)
8   }
9   // jika ada maka mutation update dijalankan
10  else{
11    cartItem.quantity++
12    commit('update', cartItem)
13  }
14 },
15 },

```

Terdapat satu actions pada yaitu add() yang akan melakukan pengecekan apakah data buku yang dimasukkan ke dalam keranjang belanja sudah ada atau belum. Melalui kode berikut:

```

1 let cartItem = state.carts.find(item => item.id === payload.id)

```

Jika data buku sudah ada pada cart maka akan meng-commit mutation update jika belum ada maka akan mengcommit mutation insert. Mutation insert akan menjalankan fungsi push (menambahkan item pada array) sedangkan mutation update akan menjalankan fungsi splice (mengupdate array pada index tertentu).

Memanggil Action State Pada Component

Untuk memanggil action add dari component maka kita bisa gunakan perintah dispatch `this.$store.dispatch('add', data_buku)` dimana `data_buku` adalah parameter objek buku.

Sebagaimana getters, kita bisa melakukan mapping action tersebut dengan menggunakan mapActions yang kita letakkan pada properti method, sebagai berikut:

```

1 methods: {
2   ...mapActions({
3     add: 'add'
4   })
}

```

Namun dengan sebelumnya melakukan import fungsi mapActions dari vuex `import { mapActions } from 'vuex'`. Sehingga pada component tersebut, action bisa kita dispatch layaknya properti methods `this.add(data_buku)`.

Oke mari kita implementasikan, pada file `src/views/Book.vue` bagian script, kita ubah menjadi sebagai berikut:

```

1 import { mapActions } from 'vuex'
2 export default {
3   data: () => ({
4     book: {}, // objek book
5   }),
6   created(){
7     this.go()
8   },
9   methods: {
10     ...mapActions({
11       add: 'add'
12     }),
13     buy(){
}

```

```

14     // alert('buy')
15     // this.$store.dispatch('add', this.book)
16     this.add(this.book)
17   }
18   go(){
19     // kode method go sebelumnya
20     // ...
21   },
22 }
23

```

Ketika memetakan menggunakan `mapActions` nama methodnya tidak harus sama dengan nama actionsnya, jadi boleh saja sebagai berikut:

```

1 ...mapActions({
2   addCart: 'add'
3 }),

```

Maka dipanggil dengan `this.addCart(this.book)`. Mari kita coba lihat hasilnya pada halaman detail buku. Lihat console browser panel Vue, klik icon switch to Vuex (mirip icon refresh)

The screenshot shows the Vue DevTools interface with the 'Vue' tab selected. At the top, there are tabs for Elements, Console, Sources, Network, Performance, Memory, and Vue. Below the tabs is a toolbar with icons for Vue DevTools, Vuex, and other tools. The main area is divided into two sections: 'Base State' and 'state'.

Base State (left column):

- Filter mutations:
- Filter inspected state:
- Time: 01:26:59

state (right column):

- state
 - carts: Array[0]

getters (right column):

- getters
 - carts: Array[0]

Perhatikan bahwa ada dua kolom di mana kolom kiri menunjukkan state awal (Base State), sedangkan kolom kanan menunjukkan isi dari state yaitu carts yang datanya masih kosong.

Sekarang coba kita klik tombol `buy`, dan perhatikan apa yang terjadi.

```

mutation {
  payload: Object
  type: "insert"
}

state {
  carts: Array[1]
  0: Object
    cover: "815511f4346a6690"
    id: 4
    price: 50000
    quantity: 1
    title: "Laboriosam animi"
    weight: 0.5
}

getters {
  carts: Array[1]
  0: Object
}

```

Nah sekarang, kolom kiri muncul pemanggilan mutation insert, dan pada kolom kanan ditampilkan detailnya yaitu jenis mutationnya insert dengan payloadnya objek, lalu pada state `carts` sekarang terdapat satu data objek buku.

Ketika buku yang sama kita klik tombol `buy`-nya lagi maka yang berubah adalah `quantity`-nya saja secara increment.

Filter mutations

Filter inspected state

Base State 01:26:59

insert 06:31:17

update 06:32:08

mutation

payload: Object
type: "update"

state

carts: Array[1]

0: Object

cover: "815511f4346a66907"
id: 4
price: 50000
quantity: 2
title: "Laboriosam animi"
weight: 0.5

getters

carts: Array[1]

0: Object

Pada kolom kiri muncul mutation update. Lalu buka buku lain dan klik tombol buy.

Maka muncul mutation insert dan state carts bertambah menjadi dua items buku.

Oke, keren kan?

Memecah State Menjadi Module Tersendiri

Pada sebuah aplikasi yang menggunakan banyak state, tentu definisi & deklarasi state, mutation, action dan getters di store-nya akan menjadi sangat kompleks. Solusi permasalahan ini adalah dengan menggunakan fitur module pada Vuex yaitu kita diizinkan untuk memecah state-state itu menjadi module-module terpisah dalam rangka mengurangi kompleksitas.

Pemisahan state-state tersebut menjadi module tersendiri bisa berdasarkan fungsinya, misal dalam konteks aplikasi studi kasus maka state untuk menangani keranjang belanja kita pisah menjadi module sendiri, state untuk menangani authentikasi dipisah, dst. Meski sebenarnya kalo dilihat dari kompleksitasnya, aplikasi studi kasus kita sebenarnya belum terlalu kompleks namun penulis ingin memberikan contoh padamu bagaimana cara melakukannya.

Pertama kita akan coba pisahkan state tentang keranjang belanja menjadi state tersendiri yang terpisah dari file state utama (`src/store.js`). Misalnya module state yang menangani keranjang belanja tersebut akan kita berinama `cart` dan akan kita simpan pada file `src/stores/cart.js`.

Untuk itu, buat file module state `cart` pada `src/stores/cart.js`. Isinya hampir sama dengan defini & deklarasi state `carts` pada file state utama, bedanya module ini hanyalah berupa objek Js biasa yang diexport.

```
1 | export default {
2 |   namespaced: true,
```

```

3   state: {
4     carts : [],
5   },
6   mutations: {
7     insert: (state, payload) => {
8       state.carts.push({
9         id: payload.id,
10        title: payload.title,
11        cover: payload.cover,
12        price: payload.price,
13        weight: payload.weight,
14        quantity: 1
15      })
16    },
17    update: (state, payload) => {
18      let idx = state.carts.indexOf(payload);
19      state.carts.splice(idx,1,{
20        id: payload.id,
21        title: payload.title,
22        cover: payload.cover,
23        price: payload.price,
24        weight: payload.weight,
25        quantity: payload.quantity
26      });
27      if(payload.quantity<=0){
28        state.carts.splice(idx,1)
29      }
30    }
31  },
32  actions: {
33    add: ({state, commit}, payload) => {
34      let cartItem = state.carts.find(item => item.id === payload.id)
35      if(!cartItem){
36        commit('insert', payload)
37      }
38      else{
39        cartItem.quantity++
40        commit('update', cartItem)
41      }
42    }
43  },
44  getters: {
45    carts : state => state.carts
46  }
47 }

```

Lalu pada file state utama atau `src/store.js` import dan definisikan state `src/stores/cart.js` sebagai module.

```

1 import Vue from 'vue'
2 import Vuex from 'vuex'
3 import cart from '@/stores/cart' // <= import ini
4 Vue.use(Vuex)

```

```

5  export default new Vuex.Store({
6    state: {
7    },
8    mutations: {
9    },
10   actions: {
11  },
12   getters: {
13  },
14   modules: {
15     cart, // <= deklarasikan pada modules
16   }
17 })

```

Apakah dengan perubahan ini akan mengubah cara memanggil state? yap tentu ada sedikit perubahan.

Jika sebelumnya `this.$store.state.carts` diubah menjadi `this.$store.state.cart.carts`

Pada getters, jika sebelumnya `this.$store.getters.carts` diubah menjadi `this.$store.getters['cart/carts']` yaitu dengan menambahkan nama modulennya di depan nama state dan formatnya array.

Pada action, jika sebelumnya `this.$store.dispatch('add')` diubah menjadi `this.$store.dispatch('cart/add')` yaitu dengan menambahkan nama modulennya di depan nama state.

Catatan: khusus untuk getters dan action, hal ini berlaku jika kita definiskan namespacing pada module `namespaced: true`. Jika tidak maka getters dan action akan diregister menjadi global fungsi sehingga untuk mengaksesnya sama dengan ketika tanpa module yaitu `this.$store.getters.carts` dan `this.$store.dispatch('add')`. Itu artinya ada kemungkinan terjadi overlapping antar fungsi getters dan actions jika menggunakan nama yang sama. Oleh karenanya disarankan kita mengeset `namespace: true` pada module supaya setiap module tidak mempengaruhi module yang lain atau bahkan store utama.

Misalnya pada halaman detail buku (`src/views/Book.vue`) yang menggunakan state carts, karena state carts sudah kita pindah menjadi module cart maka `mapAction`-nya menjadi sebagai berikut:

```

1  methods: {
2    ...mapActions({
3      addCart : 'cart/add',
4    }),
5    buy(){
6      this.addCart(this.book)
7      // Jika tanpa mapAction
8      // this.$store.dispatch('cart/add', this.book)
9    }
10   // ...
11 },

```

Mari kita coba mengklik tombol Buy.

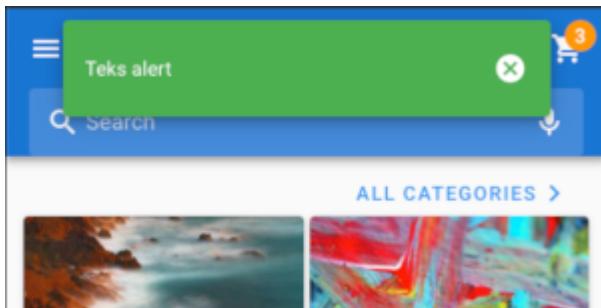
```

1 <v-snackbar
2   v-model="snackbar"
3   color="success"
4   multi-line="true"
5   top="true"
6 >
7   Teks alert
8   <v-btn dark text @click="snackbar = false">
9     <v-icon>mdi-close-circle</v-icon>
10    </v-btn>
11  </v-snackbar>

```

Kode di atas akan menampilkan alert berwarna success (hijau) dan overlay pada bagian atas aplikasi, serta terdapat tombol close.

Bentuk tampilan dari component ini sebagai berikut.



Pada template snack bar di atas, setidaknya ada 3 variabel dinamis yaitu status alert pada v-model, warna atau type alert pada color dan teks yang ditampilkan pada alert. Karenanya untuk memudahkan kita memanipulasi ketiga variabel itu dari component manapun pada aplikasi, kita bisa membuat state tersendiri untuk alert ini.

Module State Alert

Buat module alert pada state di `src/stores/alert.js` untuk memanipulasi 3 variabel pada alert yaitu status, color dan text.

```

1  export default {
2      namespaced: true,
3      state: {
4          status : false,
5          color  : 'success', // warning, error, info
6          text   : ''
7      },
8      mutations: {
9          set: (state, payload) => {
10              state.status = payload.status
11              state.text  = payload.text
12              state.color = payload.color
13          },
14      },
15      actions: {
16          set: ({commit}, payload) => {
17              commit('set', payload)
18          },
19      },
20      getters: {
21          status : state => state.status,
22          color  : state => state.color,
23          text   : state => state.text
24      }
25  }

```

Untuk memanipulasi state alert, maka cukup dengan men-dispatch action set dengan parameter objek yang berisi 3 variabel tersebut.

Jangan lupa karena state alert sebagai module vuex tersendiri maka perlu kita daftarkan pada `src/store.js`.

```

1 //...
2 import cart from '@/stores/cart'
3 import alert from '@/stores/alert' // <= tambahkan ini
4
5 Vue.use(Vuex)
6
7 export default new Vuex.Store({
8   ...
9   modules: {
10     cart,
11     alert // <= tambahkan ini
12   }
13 })

```

Component Alert

Untuk memecah kompleksitas maka kita bisa buat Alert ini sebagai sebuah component tersendiri.

Bukankah kita juga sudah menggunakan component `v-snackbar`? mengapa harus membuat component lagi?

Oke, Vue memungkinkan kita membuat component dalam sebuah component lain. Tujuan utamanya adalah untuk memecah kompleksitas. Bisa saja kita jadikan semua dalam satu file namun tentu kodingan kita akan menjadi sangat kompleks jika aplikasi kita besar.

Lho bukankah aplikasi studi kasus ini kecil?

Yap memang kecil, hanya saja penulis melakukannya (memecah menjadi component tersendiri) untuk tujuan menjelaskan padamu bagaimana cara melakukannya 😊.

Baik, langsung saja kita buat component alert pada `src/components/Alert.vue` yang berfungsi mewrap component Snackbar milik vuetify dan memetakan state alert yang telah kita buat sebelumnya.

Variabel v-model alert berbentuk computed dengan getter dan setter dari state. Di samping itu kita tambahkan method close untuk menyembunyikan alert melalui set statusnya menjadi false.

```

1 <template>
2   <v-snackbar v-model="alert" :color="color" multi-line top>
3     {{ text }}
4     <v-btn dark text @click="close">
5       <v-icon>mdi-close-circle</v-icon>
6     </v-btn>
7   </v-snackbar>
8 </template>
9 <script>
10 import { mapGetters, mapActions } from 'vuex'
11 export default {
12   name: 'alert',
13   computed: {
14     ...mapGetters({
15       status : 'alert/status',
16       color  : 'alert/color',
17       text   : 'alert/text'
18     }),
19     alert: {
20       get () {
21         return this.status

```

```

22 },
23     set (value) {
24         this.setAlert({
25             status : value,
26             type  : 'success',
27             text   : 'test',
28         })
29     }
30 },
31 },
32 methods: {
33     ...mapActions({
34         setAlert  : 'alert/set',
35     }),
36     close(){
37         this.setAlert({
38             status : false
39         })
40     }
41 }
42 }
43 </script>

```

Selanjutnya kita daftarkan component Alert ini pada file `src/App.vue`.

```

1 export default {
2     name: 'App',
3     components: {
4         Alert: () => import( /* webpackChunkName: "alert" */
5             '@/components/Alert.vue')
6     },

```

Serta untuk template-nya, tambahkan element `<alert />` dengan posisi di atas .

```

1 <alert />
2 <!-- Sizes your content based upon application components -->
3 <v-content>

```

Lalu bagaimana menampilkan alert ini?

Menampilkan Alert

Mari kembali ke `src/views/Book.vue`. Dengan menggunakan `mapAction`, petakan action `setAlert` pada methods. Kemudian pada method `buy()` kita bisa gunakan fungsi `setAlert()` untuk mengeset nilai alert dan menampilkannya.

```

1 //...
2 methods: {
3     ...mapActions({
4         addCart: 'cart/add',
5         setAlert  : 'alert/set',
6     }),
7     buy(){

```

```

8 |     this.addCart(this.book)
9 |     //this.$store.dispatch('cart/add', this.book)
10|     this.setAlert({
11|         status : true,
12|         color  : 'success',
13|         text   : 'Added to cart',
14|     })
15|     },
16|     // ...

```

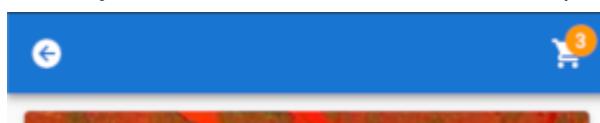
Jadi jika tombol Buy diklik maka selain akan menambahkan buku ke dalam keranjang belanja, juga akan menampilkan alert sebagai berikut.



Silakan gunakan alert ini pada component lainnya untuk menampilkan pesan informasi kepada user.

Membuat Indikator Keranjang Belanja

Pada saat ini, indikator keranjang belanja pada header yang menunjukan jumlah jenis barang yang berada di dalamnya masih manual alias hardcoded, tampak pada gambar berikut.



Pada file src/App.vue, teks pada badge masih dituliskan secara manual.

```

1 | <v-btn icon to="/about">
2 |   <v-badge color="orange" overlap>
3 |     <template v-slot:badge>
4 |       <span>3</span>
5 |     </template>
6 |     <v-icon>mdi-cart</v-icon>

```

```

7   </v-badge>
8 </v-btn>
```

Menambahkan Getters Count

Karenanya kita perlu mekanisme untuk menghitung jumlah jenis barang yang tersimpan dalam keranjang belanja. Buka state cart pada file `src/stores/cart.js`, tambahkan getters count berikut.

```

1  //...
2  getters: {
3      carts : state => state.carts,
4      count : (state) => {
5          return state.carts.length
6      },
7  }
8 }
```

Getters count ini mengembalikan panjang array dari state carts. Jika sudah maka getters ini bisa kita manfaatkan untuk memanipulasi teks badge pada header.

Menggunakan Getters Count Pada Header

Petakan getter cart count tersebut pada file `src/App.vue` bagian script computed.

```

1  computed: {
2      isHome () {
3          return (this.$route.path==='/')
4      },
5      ...mapGetters({
6          countCart : 'cart/count' // <= tambahkan ini
7      }),
8  }
```

Tentu saja sebelumnya kita harus import mapGetter dulu dari pustaka vuex `import { mapGetters } from 'vuex'`.

Lalu pada template, gunakan computed `countCart` ini untuk menggantikan hardcode teks dari badge.

```

1  <v-btn icon to="/about">
2      <v-badge color="orange" overlap>
3          <template v-slot:badge v-if="countCart>0">
4              <span>{{ countCart }}</span>
5          </template>
6          <v-icon>mdi-cart</v-icon>
7      </v-badge>
8  </v-btn>
```

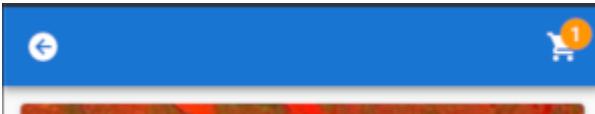
Jangan lupa bahwa apada file `src/App.vue` ada dua header yaitu header untuk parent dan child, pastika keduanya sudah kamu update badge-nya

Indikator hanya akan ditampilkan jika keranjang belanja ada isinya alias lebih besar dari nol. Mari kita lihat hasilnya.

Saat keranjang belanja kosong.



Saat keranjang belanja ada isinya.

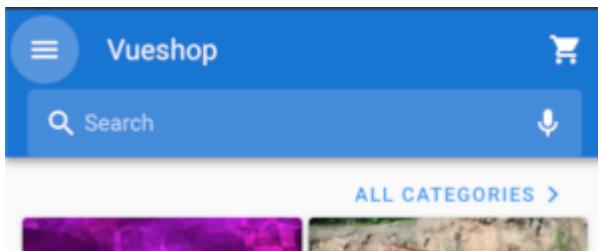


Mantap kan?

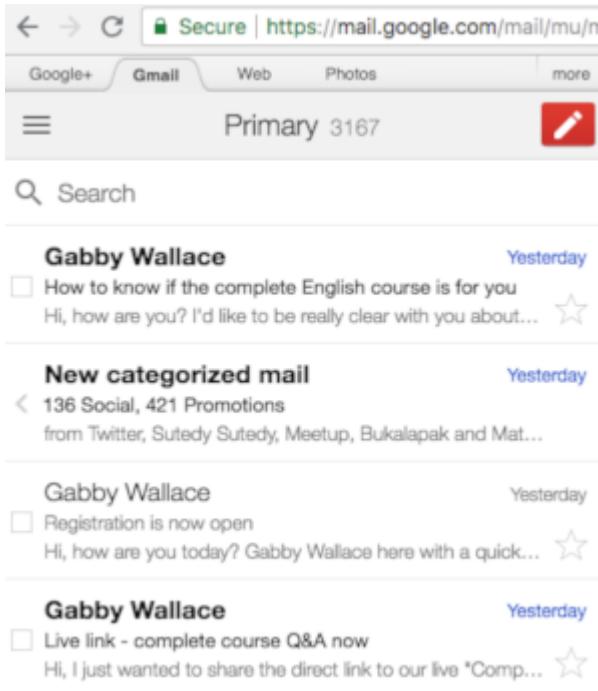
Membuat Halaman Pencarian Buku

Halaman ini menampilkan kolom pencarian buku, di mana hasil pencarian akan ditampilkan dalam bentuk list yang setiap itemnya menampilkan cover buku dan judulnya.

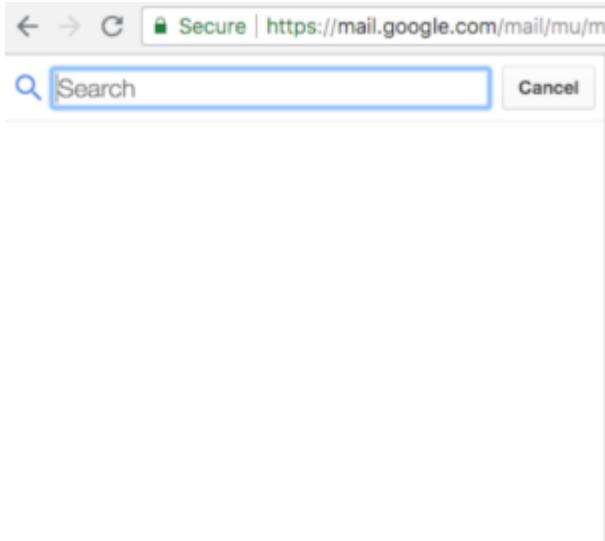
Pada desain kita sebelumnya, kolom pencarian terdapat pada halaman home tepat di bawah toolbar.



Nah, kita tidak akan mengubah desain ini, namun kita akan melakukan satu hal yang tricky yang umum dilakukan pada aplikasi berbasis mobile, ambil contoh aplikasi Gmail, berikut halaman home email-nya



ketika kolom pencarian diklik maka aplikasi akan membuka panel atau dialog baru khusus untuk pencarian.



Yaps, kita akan coba implementasikan trick ini tentunya dengan sedikit penyesuaian.

Vuetify menyediakan sebuah component yang akan membuat implementasinya menjadi cukup sederhana yaitu component v-dialog.

```

1 <v-dialog v-model="dialog" fullscreen transition="scale-transition">
2   Contoh dialog
3 </v-dialog>
```

Component v-dialog ini akan kita tampilkan secara fullscreen dan efek transisi yang akan kita gunakan adalah scale-transition yaitu efek dimana konten dialog awalnya berukuran kecil kemudian secara bertahap menjadi membesar.

Membuat Endpoint Search

Kembali ke projek Laravel. Sebelum kita membuat tampilannya, maka kita perlu buat endpoint untuk pencarian sebagai backbone data pencarian buku. Endpoint ini sifatnya publik.

Buka kembali kode pada projek laravel kita yaitu larashop-api, tambahkan fungsi `search()` pada controller Book yang sebelumnya telah kita buat. Pada fungsi search ini kita sisipkan parameter kata kunci pencarian (keyword). Lalu gunakan parameter tersebut untuk melakukan pencarian berdasarkan judul bukunya (title).

```

1 public function search($keyword)
2 {
3     $criteria = Book::select('*')
4         ->where('title', 'LIKE', "%" . $keyword . "%")
5         ->orderBy('views', 'DESC')
6         ->get();
7     return new BookResourceCollection($criteria);
8 }
```

Fungsi ini akan mengembalikan data semua buku yang judulnya berisi sebagian kata pada keyword dan diurutkan secara descending berdasarkan jumlah views. Adapun format data berupa resource collection.

Langkah selanjutnya, fungsi search pada controller Book tersebut kita daftarkan pada router api.php.

```

1 <?php
2 use Illuminate\Http\Request;
3
4 Route::prefix('v1')->group(function () {
```

```

5   // public
6   // ...
7
8   Route::get('books/top/{count}', 'BookController@top');
9   Route::get('books', 'BookController@index');
10  Route::get('books/slug/{slug}', 'BookController@slug');
11  Route::get('books/search/{keyword}', 'BookController@search'); // <=
12 ini
13
14 // private
15 Route::middleware(['auth:api'])->group(function () {
16     Route::post('logout', 'AuthController@logout');
17 });
18 });
19

```

Mari kita uji coba routing `http://larashop-api.test/v1/books/search/dolor` menggunakan postman (judul buku ada kata `dolor`-nya). Maka hasilnya kurang lebih sebagai berikut.

```

1  {
2      "status": "success",
3      "message": "books data",
4      "data": [
5          {
6              "id": 3,
7              "title": "Aut delectus dolores similique",
8              "slug": "aut-delectus-dolores-similique",
9              "description": "Ipsa blanditiis quis quas voluptates tempore.
10             Deleniti neque recusandae vel minus quam et optio illum. Et autem eos
11             maxime architecto provident ut. Eaque molestias illo corrupti aliquid.",
12              "author": "Tre Vandervort",
13              "publisher": "Kihn, Kilback and Hoeger",
14              "cover": "945e22458739b9f181c74dc7c7531105.jpg",
15              "price": 100000,
16              "weight": 0.5,
17              "views": 0,
18              "stock": 3,
19              "status": "PUBLISH",
20              "created_at": "2018-08-28 06:59:38",
21              "updated_at": "2018-09-01 23:40:18",
22              "deleted_at": null,
23              "created_by": null,
24              "updated_by": null,
25              "deleted_by": null
26          },
27          {
28              "id": 4,
29              "title": "Placeat esse rem sunt nihil dolor",
30              ...
31          },
32          {
33              "id": 10,
34              "title": "Numquam dolorem quae minima et tenetur ut",
35              ...

```

```
36     ],
  }
}
```

Perhatikan bahwa semua title (judul buku) yang ada kata `dolor`-nya akan ditampilkan.

Membuat Component Search

Component pencarian ini dibagi menjadi dua bagian yaitu toolbar yang berisi kolom pencarian dan konten yang berisi list dari buku hasil pencarian.

Component ini akan kita simpan dalam file `src/component/Search.vue`.

```
1 <template>
2   <v-card>
3     <v-toolbar dark color="primary">
4       <v-btn icon dark @click.native="close">
5         <v-icon>mdi-close</v-icon>
6       </v-btn>
7       <v-text-field
8         hide-details
9         append-icon="mdi-microphone"
10        flat
11        autofocus=true
12        label="Search"
13        prepend-inner-icon="mdi-magnify"
14        v-model="keyword"
15        @input="doSearch"
16      ></v-text-field>
17    </v-toolbar>
18    <v-card-text>
19      <v-subheader v-if="keyword.length>0">
20        Result search "{{ keyword }}"
21      </v-subheader>
22      <v-alert
23        :value="books.length==0 && keyword.length>0"
24        color="warning"
25      >
26        Sorry, but no results were found.
27      </v-alert>
28
29      <!-- Hasil pencarian ditampilkan di sini -->
30      <v-container class="ma-0 pa-0" grid-list-sm>
31        <v-layout wrap>
32          <v-flex v-for="(book) in books" :key="`book-`+book.id">
33            xs6>
34              <book-item :book="book" />
35            </v-flex>
36          </v-layout>
37        </v-container>
38      </v-card-text>
39    </v-card>
</template>
```

Pada bagian toolbar terdapat tombol (v-btn) yang akan digunakan sebagai trigger untuk keluar dari dialog search pada kasus ini akan memanggil method bernama `close`, disamping itu juga terdapat component v-text-field pada kode di atas berperan sebagai kolom pencarian dengan v-model `keyword` (nanti perlu kita definisikan variabel keyword pada properti data). Component tersebut membinding event `onInput` yang akan mentrigger method `doSearch`, method inilah yang akan merequest data buku sesuai keyword ke endpoint server.

Adapun pada bagian hasil pencarian akan menampilkan daftar buku hasil pencarian dengan menggunakan teknis list rendering seperti yang kita telah buat pada `src/views/Books.vue` yaitu dengan memanfaatkan component `BookItem` yang juga telah kita buat sebelumnya.

Supaya lebih informatif maka pada bagian atas list hasil pencarian tersebut kita tampilkan sub header yang menampilkan keyword yang sedang dicari jika buku yang dicari ada dan jika buku yang dicari tidak ada maka akan menampilkan informasi peringatan bahwa buku yang dicari tidak ada dengan menggunakan component `v-alert` bawaan vuetyf.

Berikut ini kode pada bagian script.

```

1 <script>
2 export default {
3     name: 'search',
4     components: {
5         BookItem: () => import(/* webpackChunkName: "book-item" */'
6 '@/components/BookItem.vue')
7     },
8     data () {
9         return {
10             keyword: '',
11             books: []
12         }
13     },
14     methods: {
15         doSearch(){
16             let keyword = this.keyword
17             if(keyword.length>0){
18                 this.axios.get('/books/search/'+keyword)
19                 .then((response) => {
20                     let { data } = response.data
21                     this.books = data
22                 })
23                 .catch((error) => {
24                     console.log(error)
25                 })
26             } else {
27                 this.books = []
28             }
29         },
30         close() {
31             this.$emit('closed', false)
32         }
33     },
34 }
</script>
```

Ada dua method yaitu doSearch yang berfungsi untuk melakukan request data buku berdasarkan keyword yang dimasukkan user dan method close yang melakukan emit ke directive `closed` yang akan didefinisikan di template parent (App.vue) ketika memanggil component search ini.

Menerapkan Component Search

Component search ini akan kita gunakan pada halaman utama yang penerapannya akan dibungkus dengan component v-dialog yang telah kita bahas sebelumnya. Posisi component in pada template `src/App.vue` berada diantara component alert dan v-dialog.

```

1  <alert />
2
3  <v-dialog v-model="dialog" fullscreen hide-overlay transition="scale-
4    transition">
5    <search @closed="closeDialog" />
6  </v-dialog>
7
8  <!-- Sizes your content based upon application components -->
<v-content>
```

`@closed` adalah sebuah custom directive yang dalam hal ini digunakan untuk membinding method `closeDialog`. Nah, perintah emit pada component search berkaitan dengan directive ini.

Pada bagian script `src/App.vue`, daftarkan component `Search` di bawah component alert.

```

1 components: {
2   Alert: () => import( /* webpackChunkName: "alert" */
3     '@/components/Alert.vue'),
4   Search: () => import( /* webpackChunkName: "search" */
5     '@/components/Search.vue'),
6 }
```

Tambahkan variabel dialog pada properti data serta yang digunakan sebagai v-model dari component dialog, serta tambahkan method `closeDialog` yang berfungsi mengubah nilai varaiel dialog menjadi false sehingga kotak dialog akan tertutup.

```

1 data: () => ({
2   dialog: false, // <= ini
3   // ...
4 },
5 methods: {
6   // tambahkan ini
7   closeDialog (value) {
8     this.dialog = value
9   }
10 })
```

Trigger Halaman Pencarian

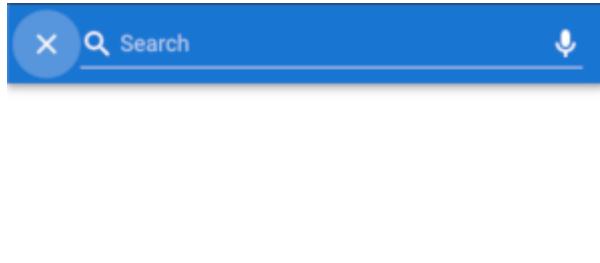
Kapan dialog search akan ditampilkan? Sebagimana yang telah dibahas sebelumnya bahwa dialog pencarian akan ditampilkan ketika user mengklik kolom pencarian pada halaman utama. Oleh karena itu kita perlu menambahkan event onclik pada kolom pencarian untuk mengubah nilai variabel dialog menjadi true sehingga memicu munculnya dialog.

Masih pada file src/App.vue tambahkan event @click pada kolom pencarian yang akan menjalankan action tersebut.

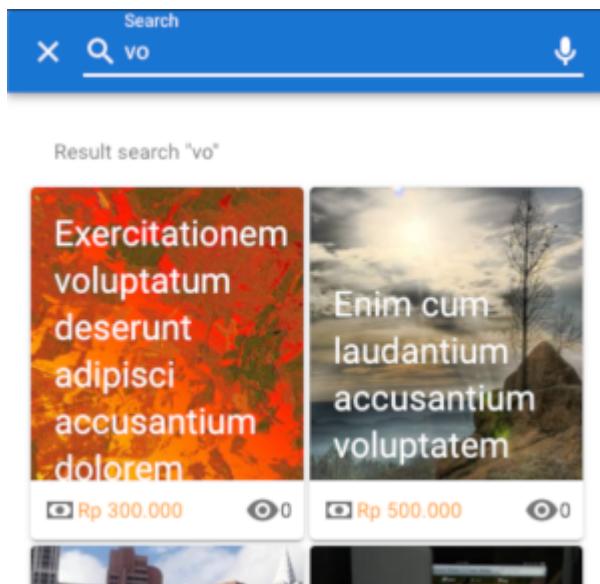
```
1 <v-text-field  
2   slot="extension"  
3   hide-details  
4   append-icon="mdi-microphone"  
5   solo-inverted flat  
6   label="Search"  
7   prepend-inner-icon="mdi-magnify"  
8   @click="dialog = true"  
9 ></v-text-field>
```

Yap cukup dengan menambahkan perintah @click="dialog = true" pada kolom pencarian.

Mari kita uji coba, pada halaman home kita klik kolom pencarian, maka popup dialog pencaria akan muncul



Ketikkan sesuatu pada kolom pencarian maka hasil pencarian akan muncul



Yess!!

Membuat Fitur Login

Halaman ini menampilkan form login email dan password serta tombol login. Halaman ini dapat diakses melalui link login pada sidebar menu (v-navigation-drawer). Halaman ini juga akan kita tampilkan sebagai dialog layaknya halaman pencarian.

Endpoint Login

Pada projek Laravel, endpoint login yang akan kita gunakan di sini masih sama dengan yang telah dibahas pada bab sebelumnya, yaitu fungsi login pada AuthController. Silakan simak kembali bagian tersebut.

Namun ada sedikit koreksi, jika sebelumnya menggunakan method firstOrFail()

```
1 | $user = User::where('email', '=', $request->email)->firstOrFail();
```

Kamu ubah menjadi method first()

```
1 | $user = User::where('email', '=', $request->email)->first();
```

Yap karena method firstOrFail() akan memicu error resource not found.

Jangan lupa pastikan route login sudah didaftarkan pada api.php

```
1 | Route::post('login', 'AuthController@login');
```

Jika kita coba mengakses endpoint login yaitu <http://larashop-api.test/v1/login> dengan method POST dan parameter email dan password yang sesuai dengan database tabel users, maka hasilnya sebagai berikut.

```
1 | {
2 |     "status": "success",
3 |     "message": "Login sukses",
4 |     "data": {
5 |         "id": 1,
6 |         "name": "Elvera Crona",
7 |         "email": "lou56@example.org",
8 |         "created_at": "2018-08-28 06:59:10",
9 |         "updated_at": "2018-09-04 00:09:45",
10 |         "roles": "[\"CUSTOMER\"]",
11 |         "address": "-",
12 |         "city": "114",
13 |         "province": "1",
14 |         "phone": "0123451",
15 |         "avatar": "d98777e02d7ca70d8a3ffef460d4acc.jpg",
16 |         "status": "ACTIVE",
17 |         "api_token":
18 |             "h8wPXhAIRvoa8fcC7jFj80KU6DeRXupAGowhLthQ9nSuwja0wA24RaYRqzdUT"
19 |     }
}
```

State Login / Auth

Data user hasil request ke endpoint login tersebut nantinya perlu kita simpan dalam sebuah state di aplikasi kita sebagai penanda apakah user sudah login atau belum. Lebih spesifik lagi, `api_token` akan kita gunakan untuk merequest resource yang membutuhkan authentication.

Oleh karena itu kita akan membuat state lagi terkait authentication ini yang akan kita buat dalam modul terpisah yaitu `src/stores/auth.js`. State ini hanya digunakan untuk menyimpan data user yang sudah login saja.

```
1 | export default {
2 |     namespaced: true,
3 |     state: {
4 |         user: {},
5 |     },
}
```

```

6   mutations: {
7     set: (state, payload) => {
8       state.user = payload
9     },
10    },
11    actions: {
12      set: ({commit}, payload) => {
13        commit('set', payload)
14      },
15    },
16    getters: {
17      user : state => state.user,
18      guest : state => Object.keys(state.user).length === 0,
19    }
20  }

```

Yup cukup simple, di mana state user bertipe object yang berisi data user, kemudian action set untuk mengeset data user yang login pada state user, getters user untuk mendapatkan data user pada state serta getters guest akan mengecek apakah ada data user atau tidak, jika tidak berarti user tersebut belum login alias tamu.

Jangan lupa daftarkan module state auth ini pada `src/store.js`

```

1 // ...
2 import auth from '@/stores/auth' // <= ini
3
4 Vue.use(Vuex)
5
6 export default new Vuex.Store({
7   // ...
8   modules: {
9     cart,
10    alert,
11    auth // <= ini
12  }

```

Memetakan State User Pada Component

State user yang telah kita buat perlu kita petaakan pada component utama untuk mengecek apakah user sudah login atau belum (guest). Pada bagian sebelumnya, kita masih melakukan hardcode yaitu dengan menggunakan properti data guest (hapus properti data guest ini), nah sekarang informasi apakah user sudah login atau belum diambil langsung dari state user.

Petakan file `src/App.vue` hook computed, tambahkan mapGetters guest dan user.

```

1 computed: {
2   isHome () {
3     return (this.$route.path==='/')
4   },
5   ...mapGetters({
6     countCart : 'cart/count',
7     guest     : 'auth/guest', // <= tambahkan ini
8     user      : 'auth/user', // <= tambahkan ini
9   })

```

```
10 |     },
| },
```

Membuat Dynamic Component

Karena halaman login ini akan kita tampilkan sebagai dialog maka supaya lebih efisien kita akan coba menerapkan teknik dynamic component yaitu component pembungkusnya satu, atau dalam hal ini v-dialog namun halaman isinya bisa dinamis yaitu search, login atau yang lain.

Untuk membuat dynamic component maka kita akan gunakan component yang disediakan Vue untuk kasus ini yaitu component. Component ini akan melakukan binding terhadap variabel nama component yang bisa kita ubah secara dinamis.

```
1 | <component :is="currentComponent"></component>
```

Supaya component sebelumnya di-cache maka kita bisa bungkus menggunakan component **keep-alive**

```
1 | <keep-alive>
2 |   <component :is="currentComponent"></component>
3 | </keep-alive>
```

Template di atas kita letakkan pada component layout utama yaitu `src/App.vue`, kita hapus saja v-dialog sebelumnya dan kita ganti dengan cara ini, jadi sekarang tidak ada lagi definisi spesifik template component search.

```
1 | <keep-alive>
2 |   <v-dialog v-model="dialog" fullscreen hide-overlay transition="dialog-
3 |   bottom-transition">
4 |     <component :is="currentComponent" @closed="closeDialog"></component>
5 |   </v-dialog>
</keep-alive>
```

Karena menyangkut banyak component, maka cara paling mudah untuk mengontrol perubahan variabel global adalah dengan menggunakan state. Pada bagian sebelumnya kita pernah properti data dialog yang intinya untuk mendefinisikan status dialog (ditampilkan atau disembunyikan), maka sekarang kita ubah status dialog tersebut menjadi sebuah state `status` serta kita juga buatkan state `component` untuk menyimpan nilai component yang sedang digunakan (current component).

State ini kita simpan pada sebuah module tersendiri yaitu modul dialog, berikut ini kode lengkapnya yang kita simpan pada file `src/stores/dialog.js`

```
1 | export default {
2 |   namespaced: true,
3 |   state: {
4 |     status      : false,
5 |     component   : 'search', // search or login or other
6 |   },
7 |   mutations: {
8 |     setStatus: (state, status) => {
9 |       state.status = status
10 |     },
11 |     setComponent: (state, component) => {
```

```

12     state.component = component
13   },
14 },
15 actions: {
16   setStatus: ({commit}, status) => {
17     commit('setStatus', status)
18   },
19   setComponent: ({commit}, component) => {
20     commit('setComponent', component)
21     commit('setStatus', true)
22   },
23 },
24 getters: {
25   status : state => state.status,
26   component : state => state.component,
27 }
28 }
```

Pada module dialog ini ada dua action yaitu setStatus() untuk mengeset status dari dialog, jika true maka tampil, jika false maka disembunyikan, serta setComponent() untuk mengeset component yang akan ditampilkan pada dialog.

Tentu saja kita harus mendaftarkannya pada store utama (src/store.js).

```

1 import dialog from '@/stores/dialog'
2 //...
3 modules: {
4   // ...
5   dialog
6 }
7 })
```

Kembali kita petakan module dialog ini pada file utama yaitu `src/App.vue`, tepatnya pada bagian script, kita import mapActions dan mapGetters karena akan digunakan untuk memetakan state module dialog. Pada properti data kita hapus variabel dialog yang nantinya akan kita ganti dengan computed dialog. Pada bagian method, kita hapus method closeDialog, yang mana kita ganti dengan setDialogStatus yang dipetakan dari module state dialog action setStatus demikian juga dengan action setComponent yang kita petakan pada method setDialogComponent.

Berikutnya pada computed kita petakan dengan menggunakan mapGetters untuk state status dan component pada module state dialog, kemudian kita buat variabel dialog yang mempunyai getter dan setter, hal ini karena variabel dialog digunakan oleh v-model, dimana v-model mensyaratkan bahwa variabel yang digunkannya harus bisa di read dan write (trik ini pernah kita bahas pada bagian sebelumnya).

Berikut ini kode lengkapnya.

```

1 import { mapActions, mapGetters } from 'vuex'
2 export default {
3   name: 'App',
4   components: {
5     Alert: () => import( /* webpackChunkName: "alert" */ 
6       '@/components/Alert.vue'),
7     Search: () => import( /* webpackChunkName: "search" */ 
8       '@/components/Search.vue'),
```

```

9 },
10 data: () => ({
11   drawer: false,
12   menus: [
13     { title: 'Home', icon: 'mdi-home', route: '/' },
14     { title: 'About', icon: 'mdi-account', route: '/about' },
15   ],
16 }), 
17 methods: {
18   ...mapActions({
19     setDialogStatus : 'dialog.setStatus',
20     setDialogComponent : 'dialog/setComponent',
21   }),
22 },
23 computed: {
24   isHome () {
25     return (this.$route.path==='/')
26   },
27   ...mapGetters({
28     countCart : 'cart/count',
29     guest : 'auth/guest',
30     user : 'auth/user',
31     dialogStatus : 'dialog/status',
32     currentComponent: 'dialog/component',
33   }),
34   dialog:{
35     get () {
36       return this.dialogStatus
37     },
38     set (value) {
39       this.setDialogStatus(value)
40     }
41   }
42 },
43 );

```

Masih pada file `src/App.vue` bagian template, kita perlu sesuaikan yaitu pada komponen pencarian event `onclick` kita ubah menjadi `setDialogComponent('search')`.

```

1 <v-text-field
2   slot="extension"
3   hide-details
4   append-icon="mdi-microphone"
5   solo-inverted flat
6   label="Search"
7   prepend-inner-icon="mdi-magnify"
8   @click="setDialogComponent('search')"
9 ></v-text-field>

```

Dari sini kita belajar bahwa jika nanti kita membuat trigger untuk komponen `login` maka perintahnya `setDialogComponent('search')`. Disamping itu pada event `@closed` komponen, kita sesuaikan, dari yang sebelumnya `closeDialog` menjadi `setDialogStatus`.

```

1 <keep-alive>
2   <v-dialog v-model="dialog" fullscreen hide-overlay transition="dialog-
3     bottom-transition">
4     <component :is="currentComponent" @closed="setDialogStatus">
5   </component>
</v-dialog>
</keep-alive>

```

Jika kode di atas benar maka ketika kita ujicoba akan muncul popup dialog yang di dalamnya terdapat tampilan dari component search.

Membuat Component Login

Component login terdiri dari dua bagian yaitu **toolbar** dan **form**. Toolbar login menggunakan component **v-toolbar** yang berisi tombol close (**v-btn**) dan judul (**v-toolbar-title**). Adapun form login menggunakan component **v-form** yang berisi dua field yaitu email dan password menggunakan component **v-text-field**, serta satu tombol login menggunakan component **v-btn**.

Untuk validasi form, kita tidak menggunakan cara imperative sebagaimana yang dicontohkan pada bab form (cara validasi ini merupakan konsep dasar validasi), namun kita akan menggunakan cara declarative sebagaimana yang dicontohkan oleh Vuetify (baca dokumentasinya). Validasi form akan dihandle oleh component **v-form** bawaan vuety, dimana rule validasi dideklarasikan melalui atribut **rules** pada **v-text-field** yang membinding property data sehingga deklarasi validasinya bisa kamu jumpai pada properti data yang diunjuk oleh **rules**.

Component ini akan kita simpan dalam file `src/views/Login.vue`. Berikut ini kode template login.

```

1 <template>
2   <v-card>
3     <v-toolbar dark color="primary">
4       <v-btn icon dark @click.native="close">
5         <v-icon>mdi-close</v-icon>
6       </v-btn>
7       <v-toolbar-title>Login and Start Shopping!</v-toolbar-title>
8     </v-toolbar>
9     <v-divider></v-divider>
10
11    <v-container fluid>
12      <v-form ref="form" v-model="valid" lazy-validation>
13        <v-text-field
14          v-model="email"
15          :rules="emailRules"
16          label="E-mail"
17          required
18          append-icon="mdi-email"
19        ></v-text-field>
20        <v-text-field
21          v-model="password"
22          :append-icon="showPassword ? 'mdi-eye' : 'mdi-eye-off'"
23          :rules="passwordRules"
24          :type="showPassword ? 'text' : 'password'"
25          label="Password"
26          hint="At least 6 characters"
27          counter

```

```

28     @click:append="showPassword = !showPassword"
29   ></v-text-field>
30
31   <div class="text-xs-center">
32     <v-btn
33       color="accent lighten-1"
34       :disabled="!valid"
35       @click="submit"
36     >
37       Login
38       <v-icon right dark>mdi-lock-open</v-icon>
39     </v-btn>
40   </div>
41 </v-form>
42 </v-container>
43 </v-card>
44 </template>

```

Pada component v-form kita gunakan atribut `lazy-validation` (bawaan Vuetify) untuk metrigger fungsi validate() (validasi form) sehingga tidak perlu dipanggil secara manual. Sedangkan pada v-text-field kita akan menjumpai directive bind pada attribut `rules` yang merujuk ke properti data.

Berikut ini script component Login.

```

1 <script>
2   import { mapGetters, mapActions } from 'vuex'
3   export default {
4     name: 'login',
5     data () {
6       return {
7         valid: true,
8         email: 'lou56@example.org',
9         emailRules: [
10           v => !!v || 'E-mail is required',
11           v => /([a-zA-Z0-9_]{1,})(@)([a-zA-Z0-9_]{2,}).([a-zA-Z0-9_]
12 {2,})+/.test(v) || 'E-mail must be valid'
13         ],
14         showPassword: false,
15         password: '',
16         passwordRules: [
17           v => !!v || 'Password required.',
18           v => (v && v.length >= 6) || 'Min 6 characters',
19         ],
20       }
21     },
22     computed: {
23       ...mapGetters({
24         user : 'auth/user',
25       }),
26     },
27     methods: {
28       ...mapActions({
29         setAlert : 'alert/set',
30         setAuth : 'auth/set',

```

```

31 },
32 submit () {
33     if (this.$refs.form.validate()) {
34         let formData = {
35             'email' : this.email,
36             'password' : this.password
37         }
38         this.axios.post('/login', formData)
39         .then((response) => {
40             let { data } = response.data
41             this.setAuth(data)
42             if(this.user.id>0){
43                 this.setAlert({
44                     status : true,
45                     color : 'success',
46                     text : 'Login success',
47                 })
48                 this.close()
49             }
50             else{
51                 this.setAlert({
52                     status : true,
53                     color : 'error',
54                     text : 'Login failed',
55                 })
56             }
57         })
58         .catch((error) => {
59             let responses = error.response
60             this.setAlert({
61                 status : true,
62                 text : responses.data.message,
63                 color : 'error',
64             })
65             })
66         }
67     },
68     close() {
69         this.$emit('closed', false)
70     }
71 },
72 }
</script>

```

Kode validasi pada v-form akan memonitor perubahan nilai yang diinputkan oleh user melalui v-text-field, jika terjadi perubahan nilai maka akan dilakukan validasi sesuai aturan pada attribut rules.

Untuk validasi email, kita menggunakan regex ini `/([a-zA-Z0-9_]{1,})@([a-zA-Z0-9_]{2,}).([a-zA-Z0-9_]{2,})+/` yaitu:

- `([a-zA-Z0-9_]{1,})` : karakter alphanumeric dan underscore minimal 1 karakter
- `@` : 1 karakter @ (a keong)
- `([a-zA-Z0-9_]{2,})` : karakter alphanumeric dan underscore minimal 2 karakter
- `.` : karakter . (titik)

- ([a-zA-Z0-9_]{2,}) : karakter alphanumeric dan underscore minimal 2 karakter

Ketika isian form valid maka akan tombol login akan aktif dan ketika tombol login tersebut diklik maka akan dijalankan method login di mana axios akan berperan mengirimkan data login ke server. Jika berhasil maka response yang diterima berupa data user yang kemudian kita simpan ke dalam state user.

Jangan lupa daftarkan component ini pada /src/App.vue.

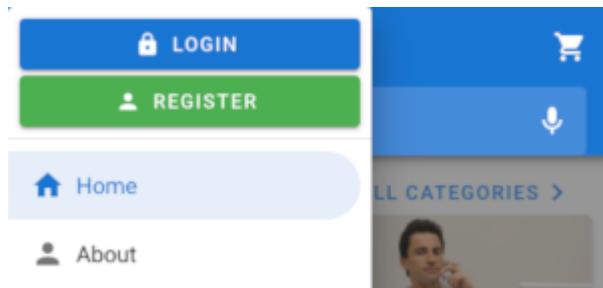
```
1  ...
2  components: {
3    Alert: () => import( /* webpackChunkName: "alert" */
4      '@/components/Alert.vue'),
5    Search: () => import( /* webpackChunkName: "search" */
6      '@/components/Search.vue'),
7    Login: () => import( /* webpackChunkName: "login" */
8      '@/components/Login.vue'),
9  },
```

Sebelum kita uji coba maka kita perlu tambahkan perintah untuk memanggil form login ini pada tombol login yang ada di sidebar alias navigation-drawer (src/App.vue)

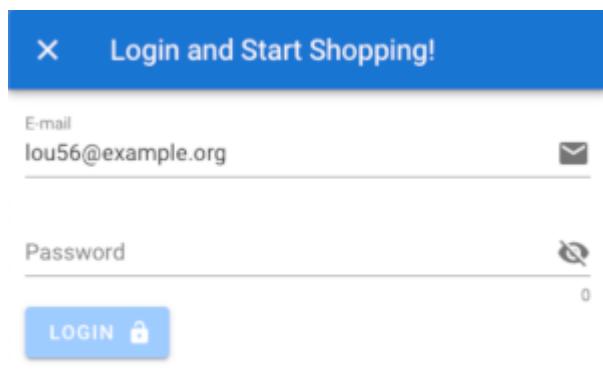
```
1 <v-btn block color="primary" class="mb-1"
2   @click="setDialogComponent('login')">
3     <v-icon left>mdi-lock</v-icon>
4     Login
5   </v-btn>
```

Mari kita uji coba, username kamu bisa ambil dari field email pada tabel users adapun password 123456 (hasil seeding pada migration sebelumnya).

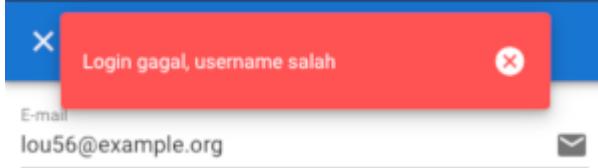
Pada sidebar klik tombol login.



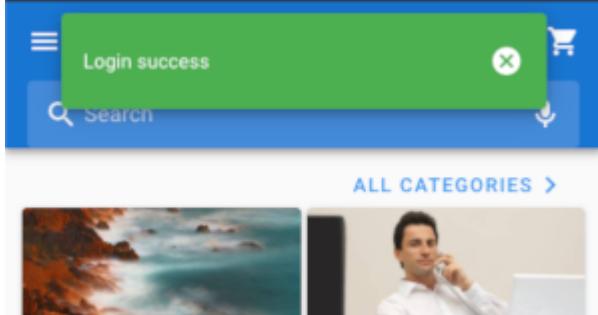
Maka akan muncul form login,



Jika password salah maka akan menampilkan pesan login gagal.



Jika login berhasil, maka dialog akan ditutup dan menampilkan pesan bahwa login berhasil



Menampilkan Data User Login

Data user sebenarnya sudah kita petakan ke component utama aplikasi kita yaitu `src/App.vue`, data itu berasal dari state user tepatnya pada modul auth. Lihat kembali pada hook computed, mapGetters.

```

1  computed: {
2    isHome () {
3      return (this.$route.path==='/')
4    },
5    ...mapGetters({
6      countCart : 'cart/count',
7      guest : 'auth/guest',
8      user : 'auth/user', // <= ini guys
9      dialogStatus : 'dialog/status',
10     currentComponent: 'dialog/component',
11   }),

```

Apa sih isi dari variabel user itu? kalo kita lihat di console browser tab vue, maka state user tampak seperti ini.

```

1 {
2   "id": 1,
3   "name": "Imogene Bradtke II", "email": "akautzer@example.net",
4   "email_verified_at": null,
5   "created_at": "2019-07-31 04:26:51", "updated_at": "2019-08-03 05:58:58",
6   "roles": "[\"CUSTOMER\"]",
7   "address": null,
8   "city_id": null,
9   "province_id": null,
10
11  "phone": null, "avatar": "27e81ac542f944ec4634b02847b37047.jpg", "status": "ACTIVE",
12  "api_token": "LYLknHEGa0EWFEBs1d2jvoxFn7w5eCFb70Py1g4eYsX5T1LdYJ2Uh
13  DlTAQ9"
14 }

```

Dengan demikian maka `user.name` adalah nama user yang sedang login, dst. Untuk avatar user, kita bisa dapatkan path lengkapnya melalui fungsi `getImage` yang telah kita gunakan sebelumnya untuk menampilkan gambar cover buku `getImage('/users/'+user.avatar)`.

Mari kita tampilkan data tersebut pada sidebar atau component navigation-drawer pada file `src/App.vue` di mana sebelumnya kita hardcode.

```

1 <v-list-item v-if="!guest">
2   <v-list-item-avatar>
3     <!-- ubah disini -->
4     <v-img :src="getImage('/users/'+user.avatar)" />

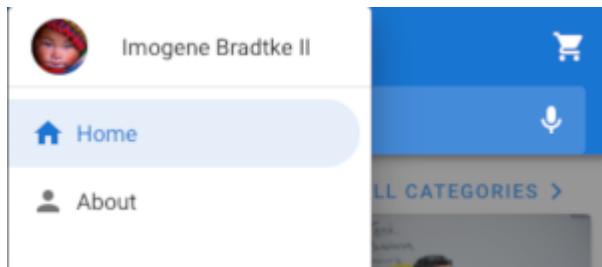
```

```

5   </v-list-item-avatar>
6
7   <v-list-item-content>
8     <v-list-item-title>
9       <!-- ubah disini -->
10      {{ user.name }}
11    </v-list-item-title>
12  </v-list-item-content>
13</v-list-item>

```

Mari kita lihat sidebar, tentunya sesudah login berhasil.



Membuat Fitur Logout

Setelah membuat fitur login, pastinya kita perlu membuat fitur logout. Cara kerja fitur ini sebenarnya simple yaitu cukup dengan menreset state user. Namun untuk kemanan, kita akan tambahkan action disisi backend yaitu mereset field api_token pada tabel users.

Link logout akan kita letakkan pada component naviagation-drawer. Kita akan gunakan v-if untuk menampilkan link logout ini yaitu hanya untuk user yang bukan guest atau yang state usernya ada isinya.

Endpoint Logout

Pada projek Laravel, endpoint logout yang akan kita gunakan di sini juga masih sama dengan yang telah dibahas pada bab sebelumnya, yaitu fungsi logout pada AuthController. Silakan simak kembali bagian tersebut.

```

1 public function logout(Request $request)
2 {
3     $user = Auth::user();
4     if ($user) {
5         $user->api_token = null;
6         $user->save();
7     }
8     return response()->json([
9         'status' => 'success',
10        'message' => 'logout berhasil',
11        'data' => []
12    ], 200);
13 }

```

Jangan lupa, pastikan route logout telah terdaftar pada api.php dengan menggunakan middleware auth:api tentunya. Karena yang bisa logout adalah user yang sudah login.

```

1 ...
2
3 // private

```

```

4 Route::middleware(['auth:api'])->group(function () {
5     Route::post('logout', 'AuthController@logout'); // <= ini
6 });
7 });

```

Jika kita coba mengakses endpoint login yaitu `http://larashop-api.test/v1/logout` dengan method POST dan menggunakan authentication bearer token, di mana tokennya berupa nilai dari field `api_token` pada database tabel users, maka hasilnya sebagai berikut.

```

1 {
2     "status": "success",
3     "message": "logout berhasil",
4     "data": []
5 }

```

Membuat Link Logout

Layaut logout sudah kita buat sebelumnya pada sidebar di file `/src/App.vue`. Tambahkan saja event onclick yang akan menjalankan method logout

```

1 <template v-slot:append v-if="!guest">
2     <div class="pa-2">
3         <v-btn block color="red" dark @click="logout">
4             <v-icon left>mdi-lock</v-icon>
5             Logout
6         </v-btn>
7     </div>
8 </template>

```

Adapun untuk method logout kita deklarasikan bagian methods yang berfungsi memanggil endpoint logout kemudian mengeset state user menjadi blank.

```

1 methods: {
2     ...mapActions({
3         setDialogStatus : 'dialog/setStatus',
4         setDialogComponent : 'dialog/setComponent',
5         setAuth : 'auth/set', // <=
6         setAlert : 'alert/set', // <=
7     }),
8     /* tambahkan fungsi logout */
9     logout(){
10         let config = {
11             headers: {
12                 'Authorization': 'Bearer ' + this.user.api_token,
13             },
14         }
15         this.axios.post('/logout', {}, config)
16         .then((response) => {
17             this.setAuth({})
18             this.setAlert({
19                 status : true,
20                 color : 'success',

```

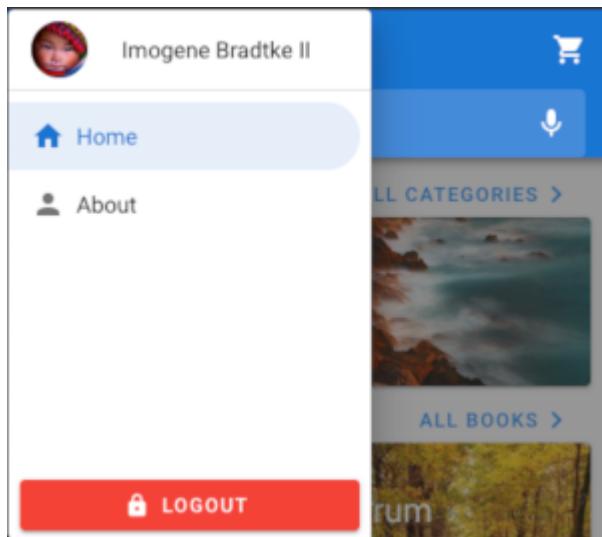
```

21     text : 'Logout successfully',
22   })
23 }
24 .catch((error) => {
25   let {data} = error.response
26   this.setAlert({
27     status : true,
28     color  : 'error',
29     text   : data.message,
30   })
31 })
32 }
33 },

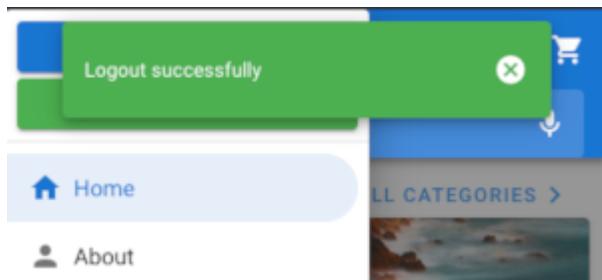
```

Mari kita uji coba.

Pada sidebar kita klik tombol logout.



Jika logout di klik makan akan menampilkan pesan bahwa logout berhasil



Halaman Register

Halaman ini menampilkan form register, nama, email, password user, serta tombol register. Halaman ini dapat diakses melalui link register pada side bar menu. Halaman ini juga akan kita tampilkan sebagai dialog layaknya halaman pencarian dan login.

Endpoint Register

Endpoint register yang akan kita gunakan di sini masih sama dengan yang telah dibahas pada bab sebelumnya, yaitu fungsi register pada AuthController. Silakan simak kembali bagian tersebut.

Ada sedikit revisi, hapus baris kode ini `Auth::login($user);`

```

1  public function register(Request $request)
2  {

```

```

3  $validator = Validator::make($request->all(), [
4      'name' => 'required|string|max:255',
5      'email' => 'required|string|email|max:255|unique:users',
6      'password' => 'required|string|min:6',
7  ]);
8
9  $status = "error";
10 $message = "";
11 $data = null;
12 $code = 400;
13 if ($validator->fails()) {
14     $errors = $validator->errors();
15     $message = $errors;
16 }
17 else{
18     $user = \App\User::create([
19         'name' => $request->name,
20         'email' => $request->email,
21         'password' => Hash::make($request->password),
22         'roles' => json_encode(['CUSTOMER']),
23     ]);
24     if($user){
25         //Auth::login($user); // hapus bari ini
26         $user->generateToken();
27         $status = "success";
28         $message = "register successfully";
29         $data = $user->toArray();
30         $code = 200;
31     }
32     else{
33         $message = 'register failed';
34     }
35 }
36
37 return response()->json([
38     'status' => $status,
39     'message' => $message,
40     'data' => $data
41 ], $code);
42 }

```

Jangan lupa daftarkan route register pada api.php

```

1 Route::prefix('v1')->group(function () {
2     // public
3     Route::post('login', 'AuthController@login');
4     Route::post('register', 'AuthController@register'); // <= ini
5     ...

```

Jika kita coba mengakses endpoint register yaitu <http://larashop-api.test/v1/register> dengan method POST dan parameter name, email dan password, maka hasilnya sebagai berikut.

```

1  {
2      "status": "success",
3      "message": "register successfully",
4      "data": {
5          "name": "hafid",
6          "email": "hafid@example.org",
7          "roles": "[\"CUSTOMER\"]",
8          "updated_at": "2018-09-07 23:03:42",
9          "created_at": "2018-09-07 23:03:42",
10         "id": 8,
11         "api_token":
12         "AiifLcehILDS6pM6MD0GFI5p5deK5bZGICgXt4aniYnWh1bAW6yT1cpmajhw"
13     }
}

```

Membuat Component Register

Pada dasarnya component register ini tidak berbeda jauh dengan component login. Hanya saja ada tambahan dua field yaitu name dan checkbox. Validasinya field nama pun juga sama sehingga tidak akan dijelaskan lagi disini.

Berikut ini kode lengkap untuk component /src/component/Register.vue.

```

1  <template>
2      <v-card>
3          <v-toolbar dark color="primary">
4              <v-btn icon dark @click.native="close">
5                  <v-icon>mdi-close</v-icon>
6              </v-btn>
7              <v-toolbar-title>Register!</v-toolbar-title>
8          </v-toolbar>
9          <v-divider></v-divider>
10
11         <v-container fluid>
12             <v-form ref="form" v-model="valid" lazy-validation>
13                 <v-text-field
14                     v-model="name"
15                     :rules="nameRules"
16                     :counter="255"
17                     label="Name"
18                     required
19                     append-icon="mdi-user"
20                 ></v-text-field>
21                 <v-text-field
22                     v-model="email"
23                     :rules="emailRules"
24                     label="E-mail"
25                     required
26                     append-icon="mdi-email"
27                 ></v-text-field>
28                 <v-text-field
29                     v-model="password"
30                     :append-icon="showPassword ? 'mdi-eye' : 'mdi-eye-off'"
31                     :rules="passwordRules"

```

```

32      :type="showPassword ? 'text' : 'password'"
33      label="Password"
34      hint="At least 6 characters"
35      counter
36      @click:append="showPassword = !showPassword"
37    ></v-text-field>
38    <v-checkbox
39      v-model="checkbox"
40      :rules="[v => !!v || 'You must agree to continue!']"
41      label="Do You agree with our Privacy Policy?"
42      required
43    ></v-checkbox>
44    <div class="text-xs-center">
45      <v-btn
46        color="accent lighten-1"
47        :disabled="!valid"
48        @click="submit"
49      >
50        Register
51        <v-icon right dark>mdi-account-plus</v-icon>
52      </v-btn>
53      <v-btn @click="clear">
54        Reset
55        <v-icon right dark>mdi-lock-reset</v-icon>
56      </v-btn>
57    </div>
58  </v-form>
59  </v-container>
60 </v-card>
61 </template>
62 <script>
63 import { mapGetters, mapActions } from 'vuex'
64 export default {
65   name: 'register',
66   data () {
67     return {
68       valid: true,
69       name: '',
70       nameRules: [
71         v => !!v || 'Name is required',
72         v => (v && v.length <= 255) || 'Name must be less than 255
73 characters'
74       ],
75       email: 'akautzer@example.net',
76       emailRules: [
77         v => !!v || 'E-mail is required',
78         v => /([a-zA-Z0-9_]{1,})(@)([a-zA-Z0-9_]{2,}).([a-zA-Z0-9_]
79 {2,})+/.test(v) || 'E-mail must be valid'
80       ],
81       showPassword: false,
82       password: '123456',
83       passwordRules: [
84         v => !!v || 'Password required.',
85         v => (v && v.length >= 6) || 'Min 6 characters',

```

```

86        ],
87        checkbox: false
88    },
89 },
90 computed: {
91     ...mapGetters({
92         user : 'auth/user',
93     }),
94 },
95 methods: {
96     ...mapActions({
97         setAlert : 'alert/set',
98         setAuth : 'auth/set',
99     }),
100    submit () {
101        if (this.$refs.form.validate()) {
102            let formData = new FormData()
103            formData.set('name', this.name)
104            formData.set('email', this.email)
105            formData.set('password', this.password)
106            this.axios.post('/register', formData)
107                .then((response) => {
108                    let { data } = response.data
109                    this.setAuth(data)
110                    this.setAlert({
111                        status : true,
112                        color : 'success',
113                        text : 'Register success',
114                    })
115                    this.close()
116                })
117                .catch((error) => {
118                    let { data } = error.response.data
119                    this.setAlert({
120                        status : true,
121                        color : 'error',
122                        text : data.message,
123                    })
124                })
125            }
126        },
127        close() {
128            this.$emit('closed', false)
129        },
130        clear () {
131            this.$refs.form.reset()
132        }
133    },
134 }
</script>

```

Setelah form di validasi maka data registrasi akan dikirim ke endpoint register menggunakan method post. Karena output dari endpoint ini adalah data user yang baru saja register, maka kita bisa gunakan data

tersebut untuk mengeser state user `this.setAuth(data)` sehingga user yang registrernya berhasil bisa otomatis diloginkan.

Menampilkan Halaman Register

Layout tombol register telah kita buat pada sidebar (`src/App.vue`), tambahkan perintah untuk menampilkan halaman register, caranya sama dengan fungsi pada tombol login yaitu menggunakan `setDialogComponent`

```

1 <v-btn block color="success" @click="setDialogComponent('register')">
2   <v-icon left>mdi-account</v-icon>
3   Register
4 </v-btn>
```

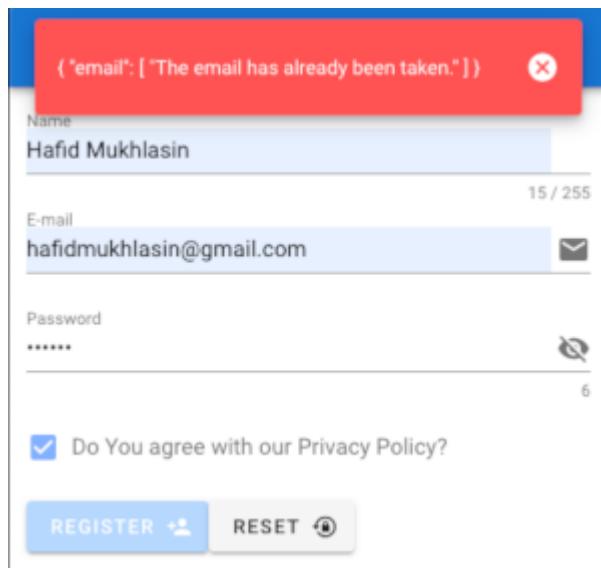
Jangan lupa daftarkan component ini pada `/src/App.vue`.

```

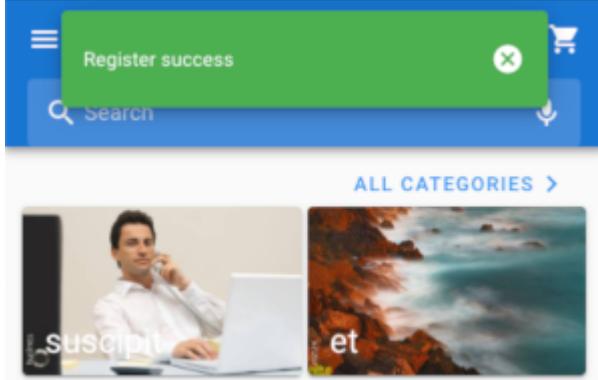
1 components: {
2   Alert: () => import( /* webpackChunkName: "alert" */ 
3     '@/components/Alert.vue'),
4   Search: () => import( /* webpackChunkName: "search" */ 
5     '@/components/Search.vue'),
6   Login: () => import( /* webpackChunkName: "login" */ 
7     '@/components/Login.vue'),
8   Register: () => import( /* webpackChunkName: "register" */ 
9     '@/components/Register.vue'), // <= ini
10 },
```

Mari kita uji coba.

Jika email sudah terdaftar maka akan menampilkan pesan bahwa email sudah digunakan.



Jika register berhasil, maka dialog akan ditutup dan menampilkan pesan bahwa register berhasil



Halaman Keranjang Belanja

Halaman ini menampilkan daftar buku yang dipesan dalam bentuk list, di mana pada setiap itemnya ditampilkan cover buku, judul buku, harga dan berat, di mana pada bagian bawah terdapat tombol Checkout. Halaman keranjang belanja ini juga akan ditampilkan dalam bentuk dialog dan dapat diakses melalui icon keranjang belanja yang terdapat pada header.

Link Keranjang Belanja

Sebelum membuat halaman keranjang belanja maka kita buat dulu link menuju halaman tersebut. Layoutnya sudah kita buat sebelumnya pada header (src/App.vue), karenanya lakukan hal yang sama dengan contoh sebelumnya, yaitu menambahkan event @click pada tombol keranjang belanja di header untuk menjalankan perintah `setDialogComponent('cart')`.

```

1  <v-btn icon @click="setDialogComponent('cart')">
2    <v-badge color="orange" overlap>
3      <template v-slot:badge v-if="countCart>0">
4        <span>{{ countCart }}</span>
5      </template>
6      <v-icon>mdi-cart</v-icon>
7    </v-badge>
8  </v-btn>

```

Pastikan kedua tombol keranjang belanja diberikan tambahan kode tersebut (di parent header dan di child header)

Mengupdate State Cart

Pada module state cart `src/stores/cart.js` yang telah kita buat sebelumnya perlu kita perbaharui dengan menambahkan beberapa fitur yaitu:

- remove (menghapus item barang pada carts),
- set cart untuk batch update,
- getters untuk menghitung total harga, total berat dan total jumlah item.

Berikut ini kode lengkapnya.

```

1  export default {
2    namespaced: true,
3    state: {
4      carts : [],
5    },
6    mutations: {

```

```

7  insert: (state, payload) => {
8      state.carts.push({
9          id: payload.id,
10         title: payload.title,
11         cover: payload.cover,
12         price: payload.price,
13         weight: payload.weight,
14         quantity: 1
15     })
16 },
17 update: (state, payload) => {
18     let idx = state.carts.indexOf(payload);
19     state.carts.splice(idx, 1, {
20         id: payload.id,
21         title: payload.title,
22         cover: payload.cover,
23         price: payload.price,
24         weight: payload.weight,
25         quantity: payload.quantity
26     });
27     if(payload.quantity<=0){
28         state.carts.splice(idx, 1)
29     }
30 },
31 // batch update carts
32 set: (state, payload) => {
33     state.carts = payload
34 },
35 },
36 actions: {
37     add: ({state, commit}, payload) => {
38         let cartItem = state.carts.find(item => item.id === payload.id)
39         if(!cartItem){
40             commit('insert', payload)
41         }
42         else{
43             cartItem.quantity++
44             commit('update', cartItem)
45         }
46     },
47     // menghapus cart pada item tertentu
48     remove: ({state, commit}, payload) => {
49         let cartItem = state.carts.find(item => item.id === payload.id)
50         if(cartItem){
51             cartItem.quantity--
52             commit('update', cartItem)
53         }
54     },
55     // batch update carts
56     set: ({commit}, payload) => {
57         commit('set', payload)
58     },
59 },
60 getters: {

```

```

61   carts : state => state.carts,
62   count : (state) => {
63     return state.carts.length
64   },
65   // menghitung total harga
66   totalPrice: (state) => {
67     let total = 0
68     state.carts.forEach(function(cart) {
69       total += cart.price * cart.quantity
70     })
71     return total
72   },
73   // total jumlah barang
74   totalQuantity: (state) => {
75     let total = 0
76     state.carts.forEach(function(cart) {
77       total += cart.quantity
78     })
79     return total
80   },
81   // total berat barang
82   totalWeight: (state) => {
83     let total = 0
84     state.carts.forEach(function(cart) {
85       total += cart.weight
86     })
87     return total
88   },
89 }
90 }
```

Membuat Component Keranjang Belanja

Component ini kita beri nama cart yang berfungsi menampilkan cart dalam bentuk tampilan list dari data state carts yang berbentuk array dari objek.

Pada bagian ini, kita akan menggunakan component v-list untuk menampilkan data keranjang belanja. Harga kita tampilkan dalam format Indonesia menggunakan fungsi `toLocaleString('id-ID')`.

File component cart akan kita buat pada file `src/components/Cart.vue`.

```

1 <template>
2   <v-card>
3     <v-toolbar dark color="primary">
4       <v-btn icon dark @click.native="close">
5         <v-icon>mdi-close</v-icon>
6       </v-btn>
7       <v-toolbar-title>Your Shopping Cart!</v-toolbar-title>
8     </v-toolbar>
9     <v-divider></v-divider>
10    <v-container fluid>
11      <div v-if="countCart === 0">
12        <v-alert
13          outlined
14          color="warning"
15        >
```

```

15           icon="mdi-cart-off"
16         >
17             Keranjang belanja kosong!
18         </v-alert>
19     </div>
20     <v-list three-line v-if="countCart>0">
21       <template v-for="(item, index) in carts" >
22         <v-list-item
23           :key="'cart'+index"
24         >
25           <v-list-item-avatar>
26             <v-img :src="getImage( '/books/' + item.cover)"></v-img>
27           </v-list-item-avatar>
28
29           <v-list-item-content>
30             <v-list-item-title v-html="item.title"></v-list-item-
31 title>
32             <v-list-item-subtitle>
33               Rp. {{ item.price.toLocaleString('id-ID') }}
34               ({{ item.weight }} kg)
35               <span style="float:right">
36                 <v-btn icon small rounded depressed
37 @click.stop="removeCart(item)">
38               <v-icon dark color="error">mdi-minus-
39 circle</v-icon>
40             </v-btn>
41             {{ item.quantity }}
42             <v-btn icon small rounded depressed
43 @click.stop="addCart(item)">
44               <v-icon dark color="success">mdi-plus-
45 circle</v-icon>
46             </v-btn>
47             </span>
48             </v-list-item-subtitle>
49           </v-list-item-content>
50         </v-list-item>
51       </template>
52     </v-list>
53     <v-card>
54       <v-card-text>
55         <v-layout wrap>
56           <v-flex pa-1 xs6>
57             Total Price ({{ totalQuantity }} items)<br>
58             <span class="title">Rp. {{ totalPrice.toLocaleString('id-
59 ID') }}</span>
60           </v-flex>
61           <v-flex pa-1 xs6 text-right>
62             <v-btn color="primary" @click="checkout"
63 :disabled="totalQuantity==0">
64               <v-icon>mdi-cart-arrow-right</v-icon> &ampnbsp
65               Checkout
66             </v-btn>
67           </v-flex>
68         </v-layout>

```

```

        </v-card-text>
    </v-card>
</v-container>
</v-card>
</template>

```

Kemudian pada script, kita petakan semua action dan getters pada state cart, serta tambahkan method checkout yang akan meredirect ke halaman checkout.

```

1  <script>
2  import { mapGetters, mapActions } from 'vuex'
3  export default {
4      name: 'cart',
5      computed: {
6          ...mapGetters({
7              carts : 'cart/carts',
8              countCart : 'cart/count',
9              totalPrice : 'cart/totalPrice',
10             totalQuantity : 'cart/totalQuantity',
11             totalWeight : 'cart/totalWeight',
12         }),
13     },
14     methods: {
15         ...mapActions({
16             setAlert : 'alert/set',
17             addCart : 'cart/add',
18             removeCart : 'cart/remove',
19             setCart : 'cart/set',
20         }),
21         checkout(){
22             this.close()
23             this.$router.push({path: "/checkout"})
24         },
25         close() {
26             this.$emit('closed', false)
27         }
28     },
29 }
30 </script>

```

Jangan lupa daftarkan component ini di `src/App.vue`

```

1 components: {
2     // ...
3     Cart: () => import( /* webpackChunkName: "cart" */ 
4     '@/components/Cart.vue'),
5 },

```

Mari kita lihat hasilnya. Pada contoh ini sudah ada dua item buku di keranjang belanja.

X Your Shopping Cart!

Suscipit repellendus saepe repellat dolor ...
Rp. 350.000 (0.5 kg) 1

Dolorem dolores distinctio quidem
Rp. 400.000 (0.5 kg) 1

Total Price (2 items)
Rp. 750.000

CHECKOUT

Jika icon plus di klik maka jumlah buku pada item buku tersebut akan bertambah, jika icon minus diklik maka jumlah buku pada item buku tersebut akan berkurang, dan jika dibawah satu maka item tersebut akan terhapus dari state cart.

X Your Shopping Cart!

Suscipit repellendus saepe repellat dolor ...
Rp. 350.000 (0.5 kg) 1

Dolorem dolores distinctio quidem
Rp. 400.000 (0.5 kg) 2

Total Price (3 items)
Rp. 1.150.000

CHECKOUT

Jika keranjang belanja kosong maka akan tampil sebagai berikut.

X Your Shopping Cart!

Keranjang belanja kosong!

Total Price (0 items)
Rp. 0

CHECKOUT

Halaman Checkout

Halaman ini hanya bisa diakses oleh user yang sudah login, jika belum login maka akan diredirect ke halaman login.

Halaman ini menampilkan tiga bagian yaitu:

1. Formulir isian alamat pengiriman buku yang dipesan.
2. Daftar item pada keranjang belanja
3. Pilihan ekspedisi yang akan digunakan, di mana aplikasi akan mengkalkulasi ongkos kirimnya.

Pada bagian bawah dari halaman ini terdapat tombol Pay yang akan mengirimkan data pemesanan ke server, jika berhasil maka halaman kemudian akan diarahkan ke halaman pembayaran.

Halaman ini dapat diakses melalui tombol **checkout** pada halaman keranjang belanja.

Endpoint Province & City

Pada projek Laravel, endpoint province dan city digunakan untuk menampilkan data dari tabel province dan city. Pertama kita akan buat dahulu resource collection-nya

```
1 | php artisan make:resource Provinces --collection
2 | php artisan make:resource Cities --collection
```

Kemudian sesuaikan format data pada fungsi `toArray()` sesuai dengan standard yang telah dijelaskan sebelumnya yaitu response dari web service kita minimal terdiri dari status, message, dan data.

Berikut ini modifikasi fungsi `toArray` pada `app\Http\Resources\Provinces.php`

```
1 | public function toArray($request)
2 |
3 |     return [
4 |         'status'  => 'success',
5 |         'message' => 'provinces data',
6 |         'data'     => parent::toArray($request),
7 |     ];
8 | }
```

Sedangkan berikut ini modifikasi fungsi `toArray` pada `app\Http\Resources\Cities.php`

```
1 | public function toArray($request)
2 |
3 |     return [
4 |         'status'  => 'success',
5 |         'message' => 'cities data',
6 |         'data'     => parent::toArray($request),
7 |     ];
8 | }
```

Kemudian kita akan membuat controller Shop untuk dua endpoint ini.

```
1 | php artisan make:controller ShopController
```

Perintah ini akan menggenerate file `ShopController.php` pada direktori `app/Http/Controllers`

Buat fungsi baru yaitu `provinces()` dan `cities()` pada controller ini yang mengembalikan objek berupa resource collection.

```
1 | public function provinces()
2 |
3 |     return new ProvinceResourceCollection(Province::get());
4 |
5 |
6 | public function cities()
7 |
8 |     return new CityResourceCollection(City::get());
9 | }
```

Tentu saja pada bagian atas kode ini, kamu harus use dulu model dan resourcenyanya.

```

1  use App\Province;
2  use App\Http\Resources\Provinces as ProvinceResourceCollection;
3  use App\City;
4  use App\Http\Resources\Cities as CityResourceCollection;
```

Jika sudah, jangan lupa daftarkan route dari dua fungsi ini pada api.php.

```

1 Route::prefix('v1')->group(function () {
2     // public
3     // ... route lain
4
5     Route::get('provinces', 'ShopController@provinces'); // <= ini
6     Route::get('cities', 'ShopController@cities'); // <= ini
7
8     // .. route lain
```

Waktunya uji coba, silakan akses endpoint <http://larashop-api.test/v1/provinces>, maka responnya:

```

1  {
2      "status": "success",
3      "message": "provinces data",
4      "data": [
5          {
6              "id": 1,
7              "province": "Bali"
8          },
9          {
10             "id": 2,
11             "province": "Bangka Belitung"
12         },
13         {
14             "id": 3,
15             "province": "Banten"
16         },
17         ...
18     ]
```

Sedangkan endpoint <http://larashop-api.test/v1/cities>, responsenya.

```

1  {
2      "status": "success",
3      "message": "cities data",
4      "data": [
5          {
6              "id": 1,
7              "province_id": 21,
8              "city_name": "Aceh Barat",
9              "type": "Kabupaten",
```

```

10         "postal_code": "23681"
11     },
12     {
13         "id": 2,
14         "province_id": 21,
15         "city_name": "Aceh Barat Daya",
16         "type": "Kabupaten",
17         "postal_code": "23764"
18     },
19     ...

```

Nah data ini nanti akan direquest oleh aplikasi Vue untuk disimpan sementara dalam state, karenanya kita perlu membuatkan state provinces dan cities untuk menyimpan dua data ini. Supaya lebih rapi maka kita jadikan state ini sebagai module sendiri yaitu `src/stores/region.js`

```

1  export default {
2      namespaced: true,
3      state: {
4          provinces: [],
5          cities: [],
6      },
7      mutations: {
8          setProvinces: (state, value) => {
9              state.provinces = value
10         },
11         setCities: (state, value) => {
12             state.cities = value
13         },
14     },
15     actions: {
16         setProvinces: ({commit}, value) => {
17             commit('setProvinces', value)
18         },
19         setCities: ({commit}, value) => {
20             commit('setCities', value)
21         },
22     },
23     getters: {
24         provinces: state => state.provinces,
25         cities: state => state.cities,
26     }
27 }

```

Jangan lupa daftarkan module ini pada `src/store.js`

```

1 // ... kode lain
2 import region from './stores/region' // <= ini
3
4 // ... kode lain
5
6 modules: {
7     cart,

```

```

8     alert,
9     dialog,
10    auth,
11    region // <= ini
12  }
13 })

```

Endpoint Update Alamat Pengiriman

Pada projek Laravel, kita akan membuat endpoint untuk mengupdate alamat pengiriman barang pada tabel users. Endpoint ini sifatnya private.

Buatlah function `shipping()` pada `ShopController` yang berfungsi menangkap 5 field dari user yaitu: name, address, phone, province_id, dan city_id. Di mana ke lima field alamat tersebut akan digunakan untuk mengupdate data alamat user pada tabel users.

```

1 public function shipping(Request $request)
2 {
3     $user = Auth::user(); // mendapatkan current user yang login
4     $status = "error";
5     $message = "";
6     $data = null;
7     $code = 200;
8     if ($user) {
9         $this->validate($request, [
10             'name' => 'required',
11             'address' => 'required',
12             'phone' => 'required',
13             'province_id' => 'required',
14             'city_id' => 'required',
15         ]);
16         $user->name = $request->name;
17         $user->address = $request->address;
18         $user->phone = $request->phone;
19         $user->province_id = $request->province_id;
20         $user->city_id = $request->city_id;
21         if($user->save()){
22             $status = "success";
23             $message = "Update shipping success";
24             $data = $user->toArray();
25         }
26     else{
27         $message = "Update shipping failed";
28     }
29 }
30 else{
31     $message = "User not found";
32 }
33
34 return response()->json([
35     'status' => $status,
36     'message' => $message,
37     'data' => $data
38

```

```
39 |     ], $code);
 }
```

Membuat Halaman Checkout Part 1

Pada bagian pertama pembuatan halaman checkout ini, kita akan fokus membuat form untuk update alamat pengiriman atau shipping address sebagaimana endpoint yang dijelaskan sebelumnya. Terdapat 5 field sesuai dengan tabel users yaitu name, address, phone, province_id, dan city_id. Nah khusus province_id dan city_id, field inputnya berupa dropdown select (v-select) yang datanya berasal dari endpoint province dan city. File componentnya akan disimpan pada file src/views/Checkout.vue.

```

1 <template>
2   <div>
3     <v-subheader>Shipping Address</v-subheader>
4     <div>
5       <v-card flat>
6         <v-container>
7           <v-form ref="form" lazy-validation>
8             <v-text-field
9               label="Name"
10              v-model="name"
11              required
12              append-icon="mdi-user"
13            ></v-text-field>
14
15             <v-textarea
16               label="Address"
17               v-model="address"
18               required
19               auto-grow
20               rows="3"
21             ></v-textarea>
22
23             <v-text-field
24               label="Phone"
25               v-model="phone"
26               required
27               append-icon="mdi-phone"
28             ></v-text-field>
29
30             <v-select
31               v-model="province_id"
32               :items="provinces"
33               item-text="province"
34               item-value="id"
35               label="Province"
36               persistent-hint
37               single-line
38             ></v-select>
39
40             <v-select
41               v-model="city_id"
42               v-if="province_id>0">
```

```

43      :items="citiesByProvince"
44      item-text="city_name"
45      item-value="id"
46      label="City"
47      persistent-hint
48      single-line
49    ></v-select>
50
51  </v-form>
52  <v-card-actions>
53    <v-btn color="success" dark @click="saveShipping">
54      <v-icon>mdi-content-save</v-icon> &ampnbsp
55      Save
56    </v-btn>
57  </v-card-actions>
58  </v-container>
59  </v-card>
60 </div>
61 </div>
62 </template>

```

Yang jelas kita perlu memetakan state user dan region pada component ini karena kita akan menggunakannya sebagai default value dari field pada form alamat pengiriman. Pada hook created kita perlu manfaatkan untuk mengambil data province dan city.

Pada v-select, city_id ini data pilihan kotanya akan bergantung (dependend) dengan data provinsi yang dipilih, karenanya kita menggunakan atribut items yang dibinding ke computed citiesByProvince supaya nilainya dinamis.

```

1 <script>
2   import { mapGetters, mapActions } from 'vuex'
3   export default {
4     data () {
5       return {
6         name: '',
7         address: '',
8         phone: '',
9         province_id: 0,
10        city_id: 0,
11      }
12    },
13    computed: {
14      ...mapGetters({
15        user: 'auth/user',
16        provinces: 'region/provinces',
17        cities: 'region/cities',
18      }),
19      citiesByProvince(){
20        let province_id = this.province_id
21        return this.cities.filter((city) => {
22          if (city.province_id==province_id) return city
23        })
24      },

```

```

25 },
26 methods: {
27   ...mapActions({
28     setAlert      : 'alert/set',
29     setAuth       : 'auth/set',
30     setProvinces : 'region/setProvinces',
31     setCities     : 'region/setCities',
32   }),
33   saveShipping(){
34     let formData = new FormData()
35     formData.set('name', this.name)
36     formData.set('address', this.address)
37     formData.set('phone', this.phone)
38     formData.set('province_id', this.province_id)
39     formData.set('city_id', this.city_id)
40
41     let config = {
42       headers: {
43         'Authorization': 'Bearer ' + this.user.api_token,
44       },
45     }
46
47     this.axios.post('/shipping', formData, config)
48       .then((response) => {
49         let { data } = response
50         this.setAuth(data.data)
51         this.setAlert({
52           status : true,
53           text   : data.message,
54           color  : 'success',
55         })
56       })
57       .catch((error) => {
58         let { data } = error
59         this.setAlert({
60           status : true,
61           text   : data.message,
62           color  : 'error',
63         })
64       })
65     },
66   },
67   created(){
68     this.name = this.user.name
69     this.address = this.user.address
70     this.phone = this.user.phone
71     this.city_id = this.user.city_id
72     this.province_id = this.user.province_id
73
74     if(this.provinces && this.provinces.length==0){
75       this.axios.get('/provinces')
76       .then((response) => {
77         let { data } = response.data
78         this.setProvinces(data)

```

```

79     })
80
81     this.axios.get('/cities')
82     .then((response) => {
83       let { data } = response.data
84       this.setCities(data)
85     })
86   }
87 }
88
89 </script>

```

Routing Checkout

Setelah component checkout kita buat maka kita perlu daftarkan pada routing. Nah yang perlu diingat bahwa component checkout ini sifatnya private, artinya hanya user yang sudah login yang boleh mengaksesnya. Karena itu, kita perlu tambahkan informasi atau flag penanda bahwa routing tersebut butuh authentication.

Buka file `src/route.js`. Tambahkan routing component checkout seperti sebelumnya, bedanya adalah pada kode ini tambahkan kode `meta: { auth: true }`

```

1  {
2    path: '/checkout',
3    name: 'checkout',
4    component: () => import( /* webpackChunkName: "checkout" */
5      './views/Checkout.vue'),
6    meta: { auth: true } // penandanya ini gans
7  },

```

Di samping itu, karena kita akan melakukan penambahan fitur navigation guard maka deklarasi routing kita ubah sedikit, dari

```

1  export default new Router({
2    mode: 'history',

```

menjadi

```

1  //export default new Router({
2  const router = new Router({
3    mode: 'history',

```

Selanjutnya kita buat navigation guard untuk mengecek routing yang private. Jika routing private dan user belum login maka tampilkan form login. Pada bagian bawah aturan routing, tambahkan kode berikut.

```

1  // tambahkan ini untuk melakukan pengecekan pada setiap routing
2  router.beforeEach((to, from, next) => {
3    // jika routing ada meta auth-nya maka
4    if (to.matched.some(record => record.meta.auth)) {
5      // jika user adalah guest
6      if(store.getters['auth/guest']){
7        // tampilkan pesan bahwa harus login dulu
8        store.dispatch('alert/set', {

```

```

Licensed to Riva Pebrian - lvhotpebrian@gmail.com - 085324454424 at 22/10/2019 20:40:19
9     status : true,
10    text : 'Login first',
11    color : 'error',
12  })
13
14  // tampilkan form login
15  store.dispatch('dialog/setComponent', 'login')
16 }
17 else{
18   next()
19 }
20 }
21 else{
22   next()
23 }
24 )
25
26 // tambah ini juga
27 export default router

```

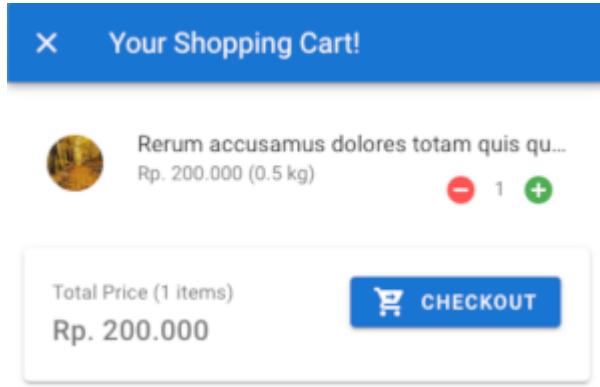
Jangan lupa, karena kita menggunakan store maka sebelum menjalankan script ini seharusnya kita import dahulu store-nya.

```

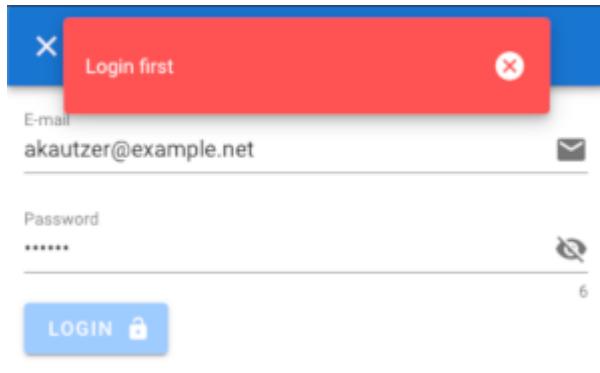
1 import store from './store'

```

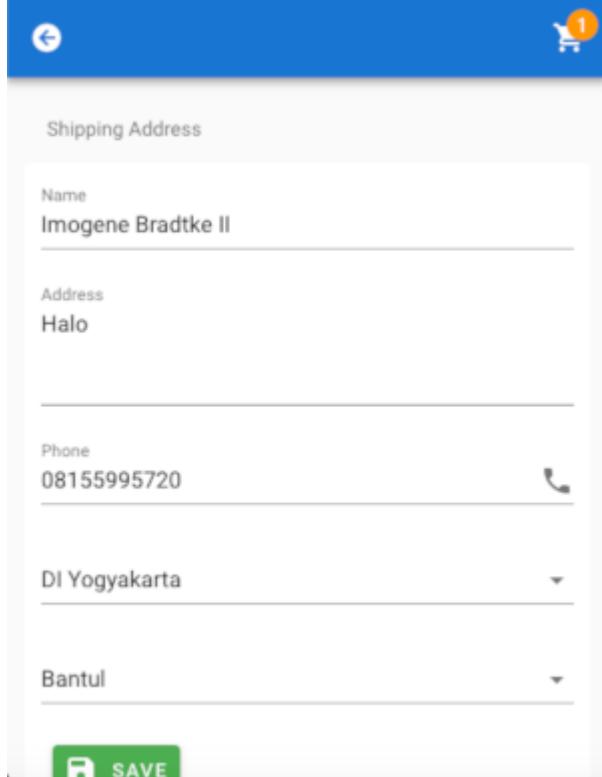
Mari kita coba. Masuk ke keranjang belanja.



Klik tombol checkout, maka dalam kondisi belum login akan muncul form login dan pesan agar login dahulu.



Kemudian, ketika login maka kita akan diarahkan ke halaman utama. Lalu kita bisa ke menu cart dan klik checkout lagi untuk melakukan checkout.



Tapi tunggu!!, setelah login seharusnya halaman dikembalikan ke halaman checkout lagi.

Untuk mencapai hal itu, caranya mudah saja. Pertama kita harus buat dahulu state untuk menyimpan URL sebelumnya.

Pada `src/store.js` tambahkan state `prevUrl`

```

1  export default new Vuex.Store({
2      state: {
3          // ...
4          prevUrl: '',
5      },
6      mutations: {
7          // ...
8          setPrevUrl: (state, value) => {
9              state.prevUrl = value
10         },
11     },
12     actions: {
13         // ...
14         setPrevUrl: ({commit}, value) => {
15             commit('setPrevUrl', value)
16         },
17     },
18     getters: {
19         // ...
20         prevUrl: state => state.prevUrl,
21     },
22 }
```

Lalu pada navigation guard di `src/router.js` kita simpan previous URL.

```

1 router.beforeEach((to, from, next) => {
2   if (to.matched.some(record => record.meta.auth)) {
3     if(store.getters['auth/guest']){
4       store.dispatch('alert/set', {
5         status : true,
6         text : 'Login first',
7         type : 'error',
8       })
9
10      store.dispatch('setPrevUrl', to.path) // tambahkan ini
11      store.dispatch('dialog/setComponent', 'login')
12      store.dispatch('dialog.setStatus', true)
13    }
14  else{
15    next()
16  }
17 }
18 else{
19   next()
20 }
21 })

```

Lalu pada method login, kita check state ini, jika ada isinya maka redirect ke URL sebelumnya sesuai dengan yang tercatat pada state prevUrl.

File `src/components/Login.vue` petakan state prevUrl.

```

1 ...mapGetters({
2   user : 'auth/user',
3   prevUrl : 'prevUrl', // <= ini
4 }),

```

Lalu pada method submit, ketika login berhasil maka lakukan pengecekan terhadap state prevUrl.

```

1 this.axios.post('/login', formData)
2 .then((response) => {
3   let data_user = response.data.data
4   this.setAuth(data_user)
5   if(this.user.id>0){
6     this.setAlert({
7       status : true,
8       text : 'Login success',
9       type : 'success',
10      })
11
12      if(this.prevUrl.length>0) this.$router.push(this.prevUrl) //
13 tambahkan ini
14      this.close()
}

```

Hasilnya adalah ketika login berhasil maka halaman akan diredirect ke prevUrl atau dalam hal ini halaman checkout.

MSilakan dicoba, ketika tombol checkout pada keranjang belanja diklik maka akan menuju halaman checkout, dan ketika tombol save diklik maka akan muncul pesan bahwa data alamat pengiriman berhasil disimpan.

Menampilkan Cart Pada Component Checkout

Pada bagian pertama pembuatan component checkout, kita telah membuat form update alamat pengiriman. Nah kini kita akan tampilkan data keranjang belanjang namun sifatnya hanya readonly.

Buka kembali file `src/views/Checkout.vue`. Petakan state cart menggunakan maps getters.

```

1 // ...
2 computed: {
3     ...mapGetters({
4         user      : 'auth/user',
5         provinces: 'region/provinces',
6         cities    : 'region/cities',
7         // tambahkan ini
8         carts     : 'cart/carts',
9         countCart : 'cart/count',
10        totalPrice: 'cart/totalPrice',
11        totalQuantity: 'cart/totalQuantity',
12        totalWeight: 'cart/totalWeight',
13    }),
14    // ...

```

Lalu pada template, tepat dibawah formulir update alamat pengiriman tambahkan kode berikut untuk menampilkan daftar belanja.

```

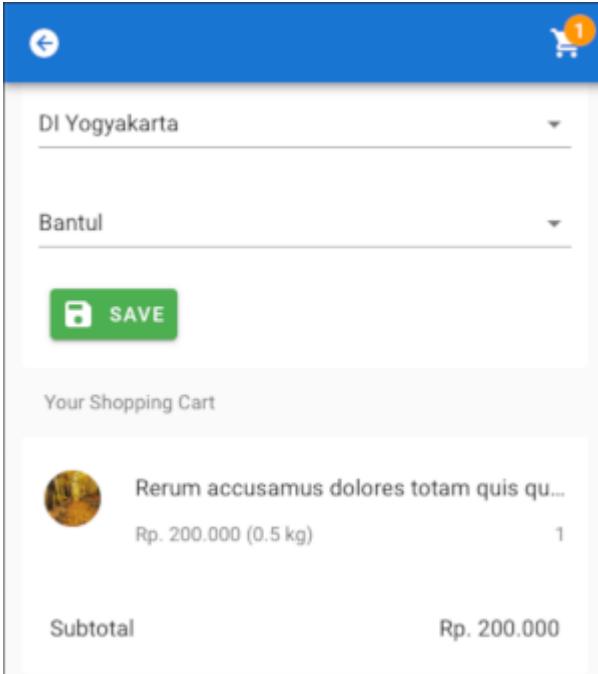
1 <v-subheader>Your Shopping Cart</v-subheader>
2 <div v-if="countCart>0">
3     <v-card flat>
4         <v-list three-line v-if="countCart>0">
5             <template v-for="(item, index) in carts" >
6                 <v-list-item
7                     :key="'cart'+index"
8                 >
9                     <v-list-item-avatar>
10                        <v-img :src="getImage('/books/'+item.cover)"></v-img>
11                    </v-list-item-avatar>
12
13                     <v-list-item-content>
14                         <v-list-item-title v-html="item.title"></v-list-item-
15 title>
16                         <v-list-item-subtitle>
17                             Rp. {{ item.price.toLocaleString('id-ID') }}
18                             ({{ item.weight }} kg)
19                             <span style="float:right">
20                                 {{ item.quantity }}
21                             </span>
22                         </v-list-item-subtitle>
23                     </v-list-item-content>
24                 </v-list-item>
25             </template>

```

```

26 </v-list>
27 <v-container>
28 <v-card-actions>
29     Subtotal
30     <v-spacer />
31     Rp. {{ totalPrice.toLocaleString('id-ID') }}
```

Berikut ini tampilannya.



Endpoint Couriers

Pada projek Laravel, endpoint couriers ini berfungsi menampilkan ekspedisi yang disediakan. Pada sistem ini hanya tiga ekspedisi yang kita sediakan yaitu pos, jne, dan tiki (karena kita menggunakan RajaOngkir versi free di mana hanya 3 ekspedisi yang didukung). Sebenarnya jika sudah pasti itemnya, kita tidak perlu menggunakan web service melainkan langsung di hardcode di kode select-nya, namun kali ini kita tetap menyediakan endpoint.

Pada ShopController tambahkan fungsi couriers() sebagai berikut.

```

1 public function couriers()
2 {
3     $couriers = [
4         ['id'=>'jne', 'text'=> 'JNE'],
5         ['id'=>'tiki', 'text'=> 'TIKI'],
6         ['id'=>'pos', 'text'=> 'POS'],
7     ];
8
9     return response()->json([
10         'status' => 'success',
11         'message' => 'couriers',
12         'data' => $couriers
13     ]);
}
```

```
14 |     ], 200);
}
```

Jangan lupa daftarkan route untuk fungsi ini.

```
1 | Route::get('couriers', 'ShopController@couriers');
```

Mari kita coba endpoint ini <http://larashop-api.test/v1/couriers>, maka responnya sebagai berikut.

```
1 | {
2 |     "status": "success",
3 |     "message": "courier",
4 |     "data": [
5 |         {
6 |             "id": "jne",
7 |             "text": "JNE"
8 |         },
9 |         {
10 |             "id": "tiki",
11 |             "text": "TIKI"
12 |         },
13 |         {
14 |             "id": "pos",
15 |             "text": "POS"
16 |         }
17 |     ]
18 | }
```

Endpoint ini akan kita panggil saat halam checkout di create.

Endpoint Courier Services

Masih pada projek Laravel, endpoint ini bertugas menampilkan daftar layanan dari ekspedisi yang dipilih beserta ongkos kirim dan estimasi waktunya. Sebagaimana yang disebutkan diawal bahwa kita menggunakan API RajaOngkir untuk mengecek ongkos kirim. Adapun endpointnya beralamat di <https://api.rajaongkir.com/starter/cost> dengan empat parameter yaitu:

- origin (id kota asal pengiriman)
- destination (id kota tujuan)
- weight (berat dalam gram)
- courier (ekspedisi: jne, tiki, pos)

Parameter origin, pada contoh ini akan kita hardcode (pada kasus nyata, kamu perlu membuat pengaturan dibackend terkait hal ini) dengan angka 153 atau Jakarta Selatan (lihat di tabel cities) artinya pada contoh ini pengiriman barang berasal dari Jakarta Selatan.

Parameter destination akan diisi dari data user field city_id.

Parameter weight adalah data berat total item pada keranjang belanja.

Parameter couriers adalah data yang ekspedisi yang dipilih.

Keempat data di atas bisa kita penuhi, namun ada satu data yang perlu kita crosscheck lagi yaitu data weight. Hal ini karena:

1. Sumber data untuk menghasilkan data weight (berat total belanja) disimpan dilokal state dan sangat rawan diotak-atik oleh user secara manual.
2. Data weight ini akan mempengaruhi ongkos kirim.

Oleh karenanya, sebaiknya perhitungan total weight dilakukan ulang di server dengan data langsung dari database. Berlaku juga untuk total harga. Sehingga dari aplikasi Vue bisa dikirim data keranjang belanja untuk dikalkulasi di server.

Pada ShopController tambahkan fungsi services(), di mana fungsi ini akan melakukan beberapa hal sebagai berikut.

```

1  public function services(Request $request){
2      // Validasi kelengkapan data
3      // 1. data belanja
4      // 2. data courier
5      // 3. data kota pengiriman dari tabel user
6
7      // Validasi data belanja
8      // 1. Cek stok barang
9      // 2. Update data belanja sesuai stok
10
11     // Request data services dari API RajaOngkir
12
13     // Response
14     // 1. Daftar services jika ada
15     // 2. Data belanja yang telah diupdate
16     // 3. Informasi jumlah belanja vs stok
17 }
```

Untuk validasi kelengkapan data, akan kita gunakan fungsi validate biasa yaitu memastikan bahwa parameter courier dan carts yang akan digunakan untuk proses selanjutnya itu ada, serta memastikan bahwa user sudah mengupdate data alamat kota pengiriman (city_id).

```

1  $this->validate($request, [
2      'courier' => 'required',
3      'carts' => 'required',
4  ]);
5
6  $user = Auth::user();
7  if($user){
8      $destination = $user->city_id;
9      if($destination>0){
```

Untuk validasi data belanja atau cart, maka kita perlu membuat fungsi tersendiri di controller Shop yaitu validateCart(). Fungsi ini akan mengecek kembali data belanja yang dikirimkan oleh user dari state carts, apakah data tersebut sesuai dengan data buku yang ada di database atau tidak. Di samping itu juga melakukan kalkulasi total item, total harga dan total berat.

Jika misalnya stok buku hanya satu sedangkan jumlah belanja (quantity) lebih dari satu maka jumlah belanja akan diupdate menjadi satu sesuai dengan jumlah maksimal stok.

Output dari fungsi ini ada dua yaitu data belanja yang sudah divalidasi dan data kalkulasi total belanja. Berikut kode lengkapnya.

```

1  protected function validateCart($carts)
2  {
3      $safe_carts = []; // persiapkan variabel untuk menampung data cart yang
4      aman
5      $total = [
6          'quantity_before' => 0,
7          'quantity' => 0,
8          'price' => 0,
9          'weight' => 0,
10     ];
11     $idx = 0;
12     // looping data state carts yang dikirim ke server untuk memastikan
13     data valid
14     foreach($carts as $cart){
15         $id = (int)$cart['id'];
16         $quantity = (int)$cart['quantity'];
17         $total['quantity_before'] += $quantity;
18         $book = Book::find($id); // ambil data buku berdasarkan id-nya
19         if($book){
20             if($book->stock>0){ // check real stock
21                 $safe_carts[$idx]['id'] = $book->id;
22                 $safe_carts[$idx]['title'] = $book->title;
23                 $safe_carts[$idx]['cover'] = $book->cover;
24                 $safe_carts[$idx]['price'] = $book->price;
25                 $safe_carts[$idx]['weight'] = $book->weight;
26                 if($book->stock < $quantity){ // jika jumlah yang dipesan
27                     melebihi stok buku maka
28                         $quantity = (int) $book->stock; // jumlah yang dipesan
29                     disamakan dengan stok
30                 }
31                 $safe_carts[$idx]['quantity'] = $quantity;
32
33                 $total['quantity'] += $quantity; // total jumlah yang dipesan di
34                 hitung kembali
35                 $total['price'] += $book->price * $quantity; //total price
36                 dihitung kembali
37                 $total['weight'] += $book->weight * $quantity; // total berat
38                 dihitung kembali
39                 $idx++;
40             }
41             else{
42                 continue;
43             }
44         }
45     }
46     return [
47         'safe_carts' => $safe_carts,
48         'total' => $total,
49     ];
50 }

```

Untuk request ke API RajaOngkir agar mendapatkan layanan ekspedisi yang tersedia beserta ongkirnya maka kita juga perlu buatkan fungsi tersendiri yaitu `getServices()`. Ada empat parameter yang perlu kita

kirimkan melalui fungsi ini sebagaimana yang telah dibahas sebelumnya yaitu origin, destination, weight, dan courier. Fungsi ini mengembalikan dua hal yaitu error jika ada dan response data layanan ekspedisi jika ada. Tambahkan fungsi ini di ShopController.

```

1 protected function getServices($data)
2 {
3     $url_cost = "https://api.rajaongkir.com/starter/cost";
4     $key="YOUR_API_RAJA_ONGKIR";
5     $postdata = http_build_query($data);
6     $curl = curl_init();
7     curl_setopt_array($curl, [
8         CURLOPT_URL => $url_cost,
9         CURLOPT_RETURNTRANSFER => true,
10        CURLOPT_ENCODING => "",
11        CURLOPT_MAXREDIRS => 10,
12        CURLOPT_TIMEOUT => 30,
13        CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
14        CURLOPT_CUSTOMREQUEST => "POST",
15        CURLOPT_POSTFIELDS => $postdata,
16        CURLOPT_HTTPHEADER => [
17            "content-type: application/x-www-form-urlencoded",
18            "key: ".$key
19        ],
20    ]);
21    $response = curl_exec($curl);
22    $error = curl_error($curl);
23    curl_close($curl);
24    return [
25        'error' => $error,
26        'response' => $response,
27    ];
28 }

```

Pada fungsi `services()` yang telah kita buat sebelumnya, bagian response, kita juga akan lakukan pengecekan apakah data hasil validasi cart berbeda antara sebelum dan sesudah divalidasi, jika berbeda maka akan memunculkan warning.

Berikut ini kode lengkap untuk fungsi `services()`.

```

1 public function services(Request $request)
2 {
3     $status = "error";
4     $message = "";
5     $data = [];
6     // validasi kelengkapan data
7     $this->validate($request, [
8         'courier' => 'required',
9         'carts' => 'required',
10    ]);
11
12     $user = Auth::user();
13     if($user){
14         $destination = $user->city_id;

```

```

15     if($destination>0){
16         // hardcode, silakan sesuaikan dengan asal pengiriman barangnya
17         $origin = 153; // Jakarta Selatan
18         $courier = $request->courier;
19         $carts = $request->carts;
20         $carts = json_decode($carts, true); // transformasi dari json
menjadi array
21
22         // validasi data belanja
23         $validCart = $this->validateCart($carts); // panggil fungsi
24         validateCart()
25             $data['safe_carts'] = $validCart['safe_carts'];
26             $data['total'] = $validCart['total'];
27             $quantity_different = $data['total']['quantity_before']
28 <>$data['total']['quantity'];
29
30             $weight = $validCart['total']['weight'] * 1000;
31             if($weight>0){
32                 // request courier service API RajaOngkir
33                 $parameter = [
34                     "origin"      => $origin,
35                     "destination" => $destination,
36                     "weight"       => $weight,
37                     "courier"      => $courier
38                 ];
39                 // check ongkos kirim ke api RajaOngkir melalui fungsi
40                 getServices()
41                     $respon_services = $this->getServices($parameter);
42                     if ($respon_services['error']==null) {
43                         $services = [];
44                         $response = json_decode($respon_services['response']); // transformasi dari json menjadi array
45                         $costs = $response->rajaongkir->results[0]->costs;
46                         foreach($costs as $cost){ // parsing ongkos kirimnya
47                             $service_name = $cost->service;
48                             $service_cost = $cost->cost[0]->value;
49                             $service_estimation = str_replace('hari', '', trim($cost-
50 >cost[0]->etd));
51                             $services[] = [
52                                 'service' => $service_name,
53                                 'cost'   => $service_cost,
54                                 'estimation' => $service_estimation,
55                                 'resume' => $service_name . ' [ Rp.
56 '.number_format($service_cost).', Etd: '.$cost->cost[0]->etd.' day(s) ]'
57                             ];
58                         }
59                     }
60
61                     // Response
62                     if(count($services)>0){
63                         $data['services'] = $services;
64                         $status = "success";
65                         $message = "getting services success";
66                     }
67                     else{
68

```

```

69         $message = "courier services unavailable";
70     }
71
72     // ketika ternyata jumlah beli berbeda dengan jumlah stok
73     maka tampilkan warninng
74     if($quantity_different){
75         $status = "warning";
76         $message = "Check cart data, ".$message;
77     }
78     } else {
79         $message = "cURL Error #:" . $respon_services['error'];
80     }
81 }
82 else{
83     $message = "weight invalid";
84 }
85 }
86 else{
87     $message = "destination not set";
88 }
89 }
90 else{
91     $message = "user not found";
92 }

return response()->json([
    'status' => $status,
    'message' => $message,
    'data' => $data
], 200);
}

```

Lalu jangan lupa daftarkan routinya pada api.php bagian auth route.

```

1 // ...
2
3 // private
4 Route::middleware(['auth:api'])->group(function () {
5     Route::post('logout', 'AuthController@logout');
6     Route::post('shipping', 'ShopController@shipping');
7     Route::post('services', 'ShopController@services'); // <= ini

```

Mari kita test endpoint ini melalui URL <http://larashop-api.test/v1/services>, dengan method POST serta authentication bearer token. Masukkan juga dua parameter yaitu courier dan carts sebagai berikut.

POST ▾	http://larashop-api.test/v1/services	Params	
✓	courier	jne	
≡ ✓	Text ▾	[{"id":2,"title":"Esse quo rem explicabo","cover":"5912d0c375d5181bb746ba495889bb0d.jpg","price":300000,"weight":0.5,"quantity":1}]	Description

Maka jika sukses hasilnya sebagai berikut.

```

1  {
2      "status": "success",
3      "message": "getting services success",
4      "data": {
5          "safe_carts": [
6              {
7                  "id": 2,
8                  "title": "Esse quo rem explicabo",
9                  "cover": "5912d0c375d5181bb746ba495889bb0d.jpg",
10                 "price": 300000,
11                 "weight": 0.5,
12                 "quantity": 1
13             }
14         ],
15         "total": {
16             "quantity_before": 1,
17             "quantity": 1,
18             "price": 300000,
19             "weight": 0.5
20         },
21         "services": [
22             {
23                 "service": "OKE",
24                 "cost": 19000,
25                 "estimation": "2-3",
26                 "resume": "OKE [ Rp. 19,000, Etd: 2-3 day(s) ]"
27             },
28             {
29                 "service": "REG",
30                 "cost": 22000,
31                 "estimation": "1-2",
32                 "resume": "REG [ Rp. 22,000, Etd: 1-2 day(s) ]"
33             },
34             {
35                 "service": "YES",
36                 "cost": 30000,
37                 "estimation": "1-1",
38                 "resume": "YES [ Rp. 30,000, Etd: 1-1 day(s) ]"
39             }
40         ]
41     }
42 }
```

Jika jumlah stock berbeda antara yang dibeli dan yang di database maka hasilnya sama kecuali pada status dan message saja.

```

1  {
2      "status": "warning",
3      "message": "Check cart data, getting services success",
```

Menampilkan Form Courier Pada Component Checkout

Pada bagian sebelumnya kita telah membuat form update alamat pengiriman dan tabel keranjang belanja. Nah kini kita akan menampilkan form untuk memilih layanan ekspedisi.

Ada dua field yaitu pilihan ekspedisi (courier) dan pilihan layanan ekspedisi (service). Sumber data field pertama berasal dari endpoint couriers, sedangkan sumber data field kedua berasal dari endpoint services.

Data services dari endpoint bisa kita request saat hook created sebagaimana data province dan city.

Ketika field pertama dipilih maka akan metrigger request ke endpoint services, dan ketika field kedua dipilih maka akan metrigger kalkulasi total biaya termasuk ongkir.

Pada projek Vue, buka kembali file `src/views/Checkout.vue`, lalu pada template tambahkan kode berikut ini di bawah tampilan keranjang belanja supaya customer bisa memilih ekspedisi dan service yang ingin dia gunakan.

```

1  <v-subheader>Courier</v-subheader>
2  <div>
3      <v-card flat>
4          <v-container>
5              <v-select
6                  v-model="courier"
7                  :items="couriers"
8                  @change="getServices"
9                  item-text="text"
10                 item-value="id"
11                 label="Courier"
12                 persistent-hint
13                 single-line
14             ></v-select>
15
16             <v-select
17                 v-model="service"
18                 v-if="courier"
19                 :items="services"
20                 @change="calculateBill"
21                 item-text="resume"
22                 item-value="service"
23                 label="Courier Service"
24                 persistent-hint
25                 single-line
26             ></v-select>
27
28             <v-card-actions>
29                 Subtotal
30                 <v-spacer />
31                 Rp. {{ shippingCost.toLocaleString('id-ID') }}
32             </v-card-actions>
33         </v-container>
34     </v-card>
35 </div>
36
37     <v-subheader>Total</v-subheader>
38     <v-card>
39         <v-container>
```

```

40 <v-layout row wrap>
41   <v-flex xs6 text-center>
42     Total Bill ({{ totalQuantity }} items)
43     <div class="title">{{ totalBill.toLocaleString('id-ID') }}</div>
44   </v-flex>
45   <v-flex xs6 text-center>
46     <v-btn color="orange">
47       <v-icon light>mdi-cash</v-icon> &ampnbsp
48       Pay
49     </v-btn>
50   </v-flex>
51 </v-layout>
52 </v-container>
53 </v-card>
```

Pada kode di atas dua method yang perlu kita buat yaitu getServices pada field pertama, dan calculateBill pada field kedua. Di samping itu, kita perlu juga tambahkan beberapa properti data yang digunakan pada template di atas, sebagai berikut.

```

1 data () {
2   return {
3     // ...
4     courier: '',
5     couriers: [],
6     service: '',
7     services: [],
8     shippingCost: 0,
9     totalBill: 0,
10    }
11  },
```

Data couriers perlu kita provide di awal melalui endpoint couriers. Oleh karena itu, pada hook created, kita bisa tambahkan kode berikut:

```

1 if(this.couriers.length==0){
2   this.axios.get('/couriers')
3   .then((response) => {
4     this.couriers = response.data.data
5   })
6 }
```

Lalu kita perlu memetakan state action `cart/set` untuk mengupdate data belanja agar sesuai dengan database.

```

1 ...mapActions({
2   // ...
3   setCart : 'cart/set'
4 }),
```

Kemudian kita buat method `getService()` yang akan merequest endpoint services

```

1  getServices(){
2      let courier = this.courier
3      let encodedCart = JSON.stringify(this.carts)
4      //console.log(encodedCart)
5      let formData = new FormData()
6      formData.set('courier', courier)
7      formData.set('carts', encodedCart);
8
9      let config = {
10         headers: {
11             'Authorization': 'Bearer ' + this.user.api_token,
12         },
13     }
14     this.axios.post('/services', formData, config)
15     .then((response) => {
16         let response_data = response.data
17         // jika tidak error maka data service dan cart akan diupdate.
18         if(response_data.status!=='error'){
19             this.services = response_data.data.services
20             this.setCart(response_data.data.safe_carts)
21         }
22
23         this.setAlert({
24             status : true,
25             text   : response_data.message,
26             color  : response_data.status,
27         })
28     })
29     .catch((error) => {
30         let responses = error.response
31         this.setAlert({
32             status : true,
33             text   : responses.data.message,
34             color  : 'error',
35         })
36     })
37 },

```

Lalu kita juga buat method calculateBill() yang akan menghitung total pembayaran.

```

1  calculateBill(){
2      let selectedService = this.services.find((service) => {
3          return (service.service==this.service)
4      })
5      this.shippingCost = selectedService.cost
6      this.totalBill = parseInt(this.totalPrice) +
7      parseInt(this.shippingCost)
8 },

```

Mari kita uji coba.

The screenshot shows a shopping cart interface. At the top, it says "Your Shopping Cart". Below that, there's a list of items:

- Esse quo rem explicabo** (image of a book) **0.5 kg** **Rp. 300.000 (1 item)**

Below the items, there are sections for shipping and payment:

- Courier**: A dropdown menu currently set to "JNE".
- REG [Rp. 22,000, Etd: 1-2 day(s)]**: A dropdown menu showing delivery options.
- Subtotal**: Rp. 300.000
- Total**: Total Bill (1 items) **322.000**

At the bottom right is a large orange button labeled **\$ PAY**.

Endpoint Payment

Pada projek Laravel, endpoint ini diakses ketika tombol Pay pada halaman checkout diklik. Prosesnya adalah memasukkan data belanja user ke dalam tabel orders. Jika berhasil maka cart akan dikosongkan dan dialihkan ke halaman pembayaran.

Masih pada controller Shop, buat fungsi baru yaitu payment.

```

1 public function payment(Request $request){
2     // validasi kelengkapan data
3
4     // buat data order
5
6     // buat data detail order
7
8     // check ongkir
9
10    // update data order
11
12    // respon
13 }
```

Untuk validasi kelengkapan data, ada tiga data yang dibutuhkan untuk fungsi ini yaitu courier, service dan carts. Seperti biasa, kita bisa gunakan fungsi validate untuk memastikan ketiga data tersebut ada.

```

1 $this->validate($request, [
2     'courier' => 'required',
3     'service' => 'required',
4     'carts' => 'required',
```

```

5   ]);
6
7   Setelah itu, untuk membuat data order, kita akan menyimpan data awal
8   pemesanan ke dalam record baru pada tabel orders, di mana total bill
9   masih kosong.
10
11  ````php
12  $order = new Order;
13  $order->user_id = $user->id;
14  $order->total_bill = 0;
15  $order->invoice_number = date('YmdHis');
16  $order->courier_service = $courier.'-'.$service;
17  $order->status = 'SUBMIT';
18  if($order->save()){
19      // ...
20  }

```

Kemudian, untuk membuat data detail order, kita akan gunakan data cart untuk disimpan di tabel book_order dan mengurangi stock dari tabel book, itupun jika stoknya mencukupi jika tidak maka kita akan tandai bahwa itu error serta tampilkan error bahwa stock kurang.

```

1 // perintah ini akan melakukan pengecekan kembali sebagaimana yang
2 dilakukan fungsi validateCarts()
3 foreach($carts as $cart){
4     $id = (int)$cart['id'];
5     $quantity = (int)$cart['quantity'];
6     $book = Book::find($id);
7     if($book){
8         if($book->stock >= $quantity){
9             $total_price += $book->price * $quantity;
10            $total_weight += $book->weight * $quantity;
11            // create book order
12            $book_order = new BookOrder;
13            $book_order->book_id = $book->id;
14            $book_order->order_id = $order->id;
15            $book_order->quantity = $quantity;
16            if($book_order->save()){
17                // kurangi stock
18                $book->stock = $book->stock - $quantity;
19                $book->save();
20            }
21        }
22        else{
23            $error++;
24            throw new \Exception('Out of stock');
25        }
26    }
27    else{
28        $error++;
29        throw new \Exception('Book is not found');
30    }
}

```

Untuk cek ongkir, kita akan gunakan fungsi `getServices()` yang telah kita buat sebelumnya.

```

1 $weight = $total_weight * 1000; // to gram
2 if($weight<=0) {
3     $error++;
4     throw new \Exception('Weight null');
5 }
6 $data = [
7     "origin"      => $origin,
8     "destination" => $destination,
9     "weight"       => $weight,
10    "courier"     => $courier
11];
12 $data_cost = $this->getServices($data);
13 if ($data_cost['error']){
14     $error++;
15     throw new \Exception('Courier service unavailable');
16 }
17
18 $response = json_decode($data_cost['response']);
19 $costs = $response->rajaongkir->results[0]->costs;
20 $service_cost = 0;
21 foreach($costs as $cost){
22     $service_name = $cost->service;
23     if($service == $service_name){
24         $service_cost = $cost->cost[0]->value;
25         break;
26     }
27 }
28 if ($service_cost<=0){
29     $error++;
30     throw new \Exception('Service cost invalid');
31 }
```

Untuk update data order kita lakukan setelah mendapatkan data ongkir. Pada proses ini kita akan mengupdate data total bayar dan mengcommit transaksi jika tidak ada error.

```

1 $total_bill = $total_price + $service_cost;
2 // update total bill order
3 $order->total_bill = $total_bill;
4 if($order->save()){
5     if($error==0){
6         DB::commit();
7         $status = 'success';
8         $message = 'Transaction success';
9         $data = [
10             'order_id' => $order->id,
11             'total_bill' => $total_bill,
12             'invoice_number' => $order->invoice_number,
13         ];
14     }
15     else{
16         $message = 'There are '.$error.' errors';
}
```

```
17     }
18 }
```

Berikut ini kode lengkap untuk fungsi payment.

```

1 public function payment(Request $request)
2 {
3     $error = 0;
4     $status = "error";
5     $message = "";
6     $data = [];
7
8     $user = Auth::user();
9     if ($user) {
10         // validasi kelengkapan data
11         $this->validate($request, [
12             'courier' => 'required',
13             'service' => 'required',
14             'carts' => 'required',
15         ]);
16
17         DB::beginTransaction();
18         try {
19             // prepare data
20             $origin = 153; // Jakarta Selatan
21             $destination = $user->city_id;
22             if($destination<=0) $error++;
23             $courier = $request->courier;
24             $service = $request->service;
25             $carts = json_decode($request->carts, true);
26
27             // create order
28             $order = new Order;
29             $order->user_id = $user->id;
30             $order->total_bill = 0;
31             $order->invoice_number = date('YmdHis');
32             $order->courier_service = $courier.'-'.$service;
33             $order->status = 'SUBMIT';
34             if($order->save()){
35                 $total_price = 0;
36                 $total_weight = 0;
37                 foreach($carts as $cart){
38                     $id = (int)$cart['id'];
39                     $quantity = (int)$cart['quantity'];
40                     $book = Book::find($id);
41                     if($book){
42                         if($book->stock>=$quantity){
43                             $total_price += $book->price * $quantity;
44                             $total_weight += $book->weight * $quantity;
45                             // create book order
46                             $book_order = new BookOrder;
47                             $book_order->book_id = $book->id;
48                             $book_order->order_id = $order->id;
49                         }
50                     }
51                 }
52             }
53         } catch (\Exception $e) {
54             $error++;
55         }
56         if($error>0) $status = "error";
57         else $status = "success";
58     }
59 }
```

```

49         $book_order->quantity = $quantity;
50         if($book_order->save()){
51             // kurangi stock
52             $book->stock = $book->stock - $quantity;
53             $book->save();
54         }
55     }
56     else{
57         $error++;
58         throw new \Exception('Out of stock');
59     }
60 }
61 else{
62     $error++;
63     throw new \Exception('Book is not found');
64 }
65 }
66
67 $totalBill = 0;
68 $weight = $total_weight * 1000; // to gram
69 if($weight<=0) {
70     $error++;
71     throw new \Exception('Weight null');
72 }
73 $data = [
74     "origin"      => $origin,
75     "destination" => $destination,
76     "weight"       => $weight,
77     "courier"      => $courier
78 ];
79 $data_cost = $this->getServices($data);
80 if ($data_cost['error']){
81     $error++;
82     throw new \Exception('Courier service unavailable');
83 }
84
85 $response = json_decode($data_cost['response']);
86 $costs = $response->rajaongkir->results[0]->costs;
87 $service_cost = 0;
88 foreach($costs as $cost){
89     $service_name = $cost->service;
90     if($service == $service_name){
91         $service_cost = $cost->cost[0]->value;
92         break;
93     }
94 }
95 if ($service_cost<=0){
96     $error++;
97     throw new \Exception('Service cost invalid');
98 }
99
100 $total_bill = $total_price + $service_cost;
101 // update total bill order
102 $order->total_bill = $total_bill;

```

```

103     if($order->save()){
104         if($error==0){
105             DB::commit();
106             $status = 'success';
107             $message = 'Transaction success';
108             $data = [
109                 'order_id' => $order->id,
110                 'total_bill' => $total_bill,
111                 'invoice_number' => $order->invoice_number,
112             ];
113         }
114     else{
115         $message = 'There are '.$error.' errors';
116     }
117 }
118 }
119 } catch (\Exception $e) {
120     $message = $e->getMessage();
121     DB::rollback();
122 }
123 }
124 else{
125     $message = "User not found";
126 }
127
128 return response()->json([
129     'status' => $status,
130     'message' => $message,
131     'data' => $data
132 ], 200);
133
134 }
```

Berikut ini hasil ujicoba

POST ▾	http://larashop-api.test/v1/payment		Params
	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	courier	jne	
<input checked="" type="checkbox"/>	service	REG	
<input checked="" type="checkbox"/>	carts	Text ▾	
	Key	[{"id":2,"title":"Esse quo rem explicabo","cover":"5912d0c375d5181bb746ba495889bb0d.jpg","price":300000,"weight":0.5,"quantity":2}]	Description

Jika stocknya kurang maka akan muncul sebagai berikut.

```

1 {
2     "status": "error",
3     "message": "Out of stock",
4     "data": []
5 }
```

Namun jika sukses akan menampilkan respon berikut.

```

1  {
2      "status": "success",
3      "message": "Transaction success",
4      "data": {
5          "order_id": 2,
6          "total_bill": 322000,
7          "invoice_number": "20180909005030"
8      }
9  }
```

Update Tombol Pay Pada Component Checkout

Setelah endpoint payment siap maka kita akan tambahkan perintah pada tombol pay untuk mengeksekusi endpoint tersebut. Namun sebelum itu karena ini menyangkut transaksi maka kita perlu memberikan dialog konfirmasi untuk memastika bahwa user benar-benar mau melanjutkan proses pemesaan.

Pada bagian total, kita modifikasi menjadi sebagai berikut.

```

1 <v-subheader>Total</v-subheader>
2 <v-card>
3 <v-container>
4 <v-layout row wrap>
5     <v-flex xs6 text-xs-center>
6         Total Bill ({{ totalQuantity }} items)
7         <div class="title">{{ totalBill.toLocaleString('id-ID') }}</div>
8     </v-flex>
9     <v-flex xs6 text-xs-center>
10        <v-btn color="orange" @click="dialogConfirm=true"
11        :disabled="totalBill==0">
12            <v-icon light>attach_money</v-icon> &ampnbsp
13            Pay
14        </v-btn>
15    </v-flex>
16 </v-layout>
17 </v-container>
18 </v-card>
19
20 <template>
21     <v-layout row justify-center>
22         <v-dialog v-model="dialogConfirm" persistent max-width="290">
23             <v-card>
24                 <v-card-title class="headline">Confirmation!</v-card-title>
25                 <v-card-text>If You continue, transaction will be
26 processed</v-card-text>
27                 <v-card-actions>
28                     <v-btn color="warning" @click="cancel">Cancel</v-btn>
29                     <v-spacer></v-spacer>
30                     <v-btn color="success" @click="pay">Continue</v-btn>
31                 </v-card-actions>
32             </v-card>
33         </v-dialog>
```

```
</v-layout>
</template>
```

Kita tambahkan dialog untuk konfirmasi, dimana ada endpoint payment akan dieksekusi jika tombol Continue pada dialog konfirmasi diklik.

Untuk itu, pada script properti data kita tambahkan data `dialogConfirm` dengan nilai default false, serta tambahkan juga dua method yaitu pay dan close.

```

1  pay(){
2      this.dialogConfirm = false
3      let courier = this.courier
4      let service = this.service
5      let safeCart = JSON.stringify(this.carts)
6      let formData = new FormData()
7      formData.set('courier', courier)
8      formData.set('service', service)
9      formData.set('carts', safeCart);
10     let config = {
11         headers: {
12             'Authorization': 'Bearer ' + this.user.api_token,
13         },
14     }
15     this.axios.post('/payment', formData, config)
16         .then((response) => {
17             let { data } = response
18             if(data && data.status=='success'){
19                 this.$router.push({path: "/payment"})
20                 this.setCart([])
21             }
22
23             this.setAlert({
24                 status : true,
25                 text   : data.message,
26                 color  : data.status,
27             })
28         })
29         .catch((error) => {
30             let { data } = error.response
31             this.setAlert({
32                 status : true,
33                 text   : data.message,
34                 color  : 'error',
35             })
36         })
37     },
38     cancel(){
39         this.dialogConfirm = false
40     }

```

Pada kode di atas, hanya jika transaksi berhasil maka halaman akan diredirect ke routing payment (halaman ini belum kita buat) dan data belanja akan dikosongkan.

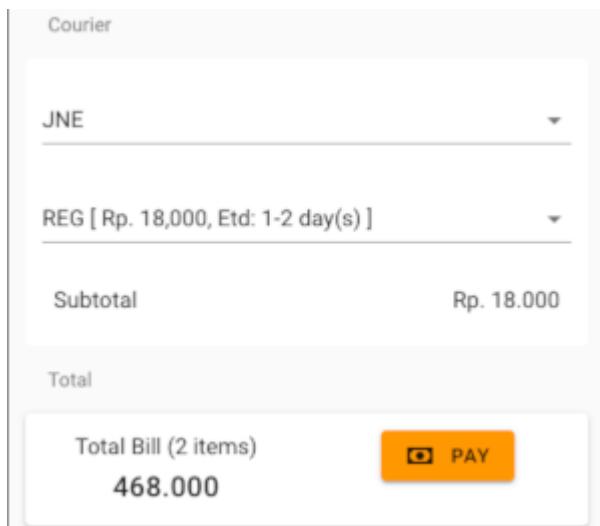
```

1 | this.$router.push({path: "/payment"})
2 | this.setCart([])

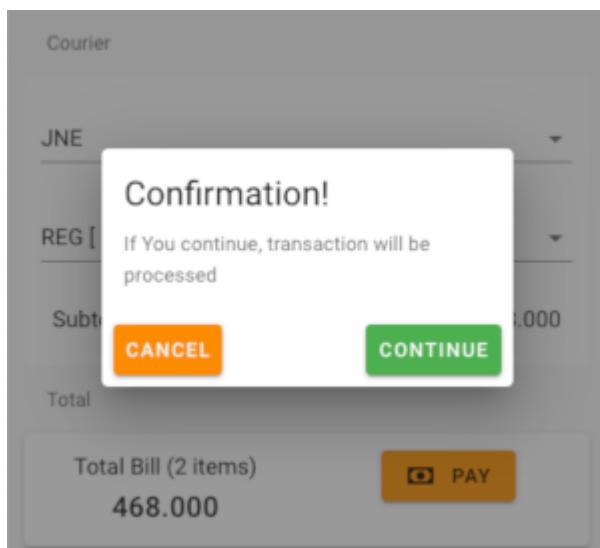
```

Mari kita coba.

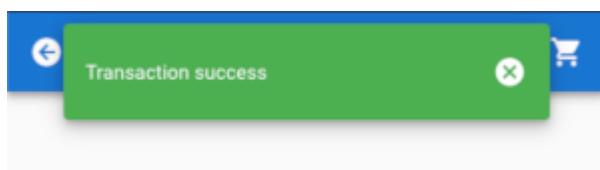
Pada halaman checkout, klik tombol Pay



Muncul dialog konfirmasi, klik Continue



Muncul pesan bahwa transaksi berhasil dan halaman diredirect ke halaman payment.



Oke, kita sebenarnya perlu state tambahan untuk menyimpan data transaksi berhasil hasil dari respon endpoint payment.

```

1 | {
2 |   "status": "success",
3 |   "message": "Transaction success",
4 |   "data": {
5 |     "order_id": 2,
6 |     "total_bill": 322000,
7 |     "invoice_number": "20180909005030"

```

```
8     }
9 }
```

Tujuannya adalah agar data ini bisa kita tampilkan pada halaman pembayaran.

Buka `src/store.js`, tambahkan state payment, termasuk mutation, action dan getternya.

```
1 state: {
2     // state lain
3     payment: []
4 },
5 mutations: {
6     // mutation lain
7     setPayment: (state, value) => {
8         state.payment = value
9     },
10 },
11 actions: {
12     // action lain
13     setPayment: ({commit}, value) => {
14         commit('setPayment', value)
15     },
16 },
17 getters: {
18     // getter lain
19     payment: state => state.payment,
20 },
```

Kemudian balik ke component Checkout, petakan action `setPayment : 'setPayment'`.

```
1 methods: {
2     ...mapActions({
3         setAlert      : 'alert/set',
4         setAuth       : 'auth/set',
5         setProvinces : 'region/setProvinces',
6         setCities     : 'region/setCities',
7         setCart       : 'cart/set',
8         setPayment    : 'setPayment' // <= ini
9     }),
10 }
```

Lalu pada method `pay()` gunakan method itu untuk menyimpan respon data jika transaksi sukses.

```
1 this.axios.post('/payment', formData, config)
2 .then((response) => {
3     let { data } = response
4     if(data && data.status=='success'){
5         this.setPayment(data.data) // <= ini
6         this.$router.push({path: "/payment"})
7         this.setCart([])
8     }
9 }
```

Halaman Pembayaran

Halaman ini sebenarnya hanya menampilkan data pembayaran dari state payment. Pada bagian bawah kita tambahkan tombol finish yang akan meredirect ke halaman utama.

Mari kita buat file src/views/Payment.vue

```

1  <template>
2      <div class="payment">
3          <v-subheader>Payment Information</v-subheader>
4          <v-card flat>
5              <v-container v-if="payment">
6                  <v-simple-table>
7                      <tr><th>Order ID</th><td>{{ payment.order_id }}</td></tr>
8                      <tr><th>Invoice Number</th><td>{{ payment.invoice_number }}</td>
9                  </tr>
10                 <tr><th>Total Bill</th><td>Rp. {{ payment.total_bill.toLocaleString('id-ID') }}</td></tr>
11             </v-simple-table>
12         </v-container>
13     </v-card>
14
15
16     <v-subheader>Transfer To</v-subheader>
17     <v-card flat>
18     <v-container>
19         <v-simple-table>
20             <tr>
21                 <td></td>
22                 <td>BCA KCP abc No Rek 123</td>
23             </tr>
24             <tr>
25                 <td></td>
26                 <td>BANK MANDIRI KCP xyz No Rek 456</td>
27             </tr>
28         </v-simple-table>
29     </v-container>
30 
```

- 15
- 16 <v-subheader>Transfer To</v-subheader>
- 17 <v-card flat>
- 18 <v-container>
- 19 <v-simple-table>
- 20 <tr>
- 21 <td></td>
- 22 <td>BCA KCP abc No Rek 123</td>
- 23 </tr>
- 24 <tr>
- 25 <td></td>
- 26 <td>BANK MANDIRI KCP xyz No Rek 456</td>
- 27 </tr>
- 28 </v-simple-table>
- 29 </v-container>
- 30 </v-card>
- 31
- 32 <v-subheader></v-subheader>
- 33 <v-card>
- 34 <v-container>
- 35 <v-layout row wrap>
- 36 <v-flex xs12 text-center>
- 37 <v-btn color="success" @click="finish">
- 38 Finish
- 39 </v-btn>
- 40 </v-flex>
- 41 </v-layout>
- 42 </v-container>
- 43 </v-card>
- 44 </div>
- 45 </template>
- 46 <script>
- 47 import { mapGetters, mapActions } from 'vuex'
- 48 export default {
- 49 computed: {

```

50     ...mapGetters({
51         payment : 'payment',
52     }),
53 },
54 created(){
55     if(this.payment==undefined){
56         this.setAlert({
57             status : true,
58             text : 'Payment undefined',
59             color : 'warning',
60         })
61         this.$router.push('/')
62     }
63 },
64 methods: {
65     ...mapActions({
66         setAlert      : 'alert/set',
67     }),
68     finish(){
69         this.setAlert({
70             status : true,
71             text : 'Transaction done',
72             color : 'success',
73         })
74         this.$router.push('/')
75     }
76 },
77
</script>

```

Daftarkan pada routing

```

1  {
2     path: '/payment',
3     name: 'payment',
4     component: () => import( /* webpackChunkName: "payment" */ 
5     './views/Payment.vue'),
6     meta: { auth: true }
7 },

```

Saatnya kita uji coba.

Payment Information

Order ID	11
Invoice Number	20190804001013
Total Bill	Rp. 218.000

Transfer To



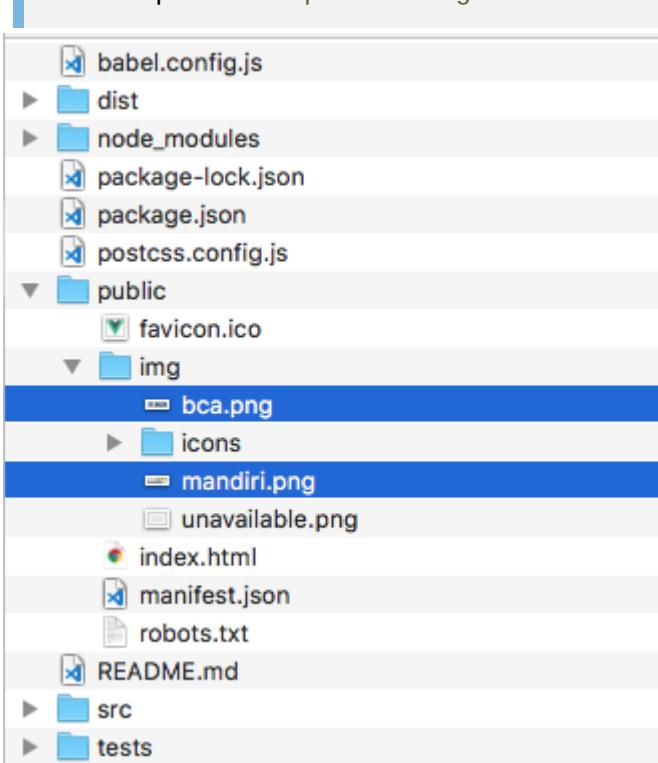
BCA KCP abc No Rek 123



BANK MANDIRI KCP xyz No Rek 456

FINISH

Catatan: data bank pada contoh ini masih hardcoded, kamu perlu siapkan image logo bank dan letakkan pada folder `public/img`



Integrasi Payment Gateway (Experimental)

Pada bagian sebelumnya kita telah membuat halaman pembayaran, di mana pada halaman itu menampilkan total tagihan dan daftar rekening bank. Ketika customer telah melakukan transfer ke salah satu rekening yang disediakan sesuai dengan nominal tagihan dan melakukan konfirmasi pembayaran maka pemilik toko memverifikasi pembayaran itu dengan cara mengecek apakah ada pembayaran atas nama customer tersebut. Jika pembayaran terverifikasi maka pemilik toko baru akan memproses pesanan.

Dari sisi pemilik toko, proses verifikasi manual pembayaran ini tentunya membutuhkan ketelitian dan waktu yang tidak sedikit yaitu mencocokkan data konfirmasi, dengan data histori transfer di rekening. Solusinya kita bisa bekerja sama dengan pihak bank untuk menyediakan web service tertentu sehingga ketika terjadi pembayaran oleh customer maka pemilik toko langsung mendapat notifikasi yang dapat mentrigger sistem untuk mengupdate secara otomatis data status pembayaran atas order terkait menjadi terverifikasi. Namun permasalahannya, bekerja sama dengan bank secara langsung bukanlah hal yang mudah.

Dari sisi customer, mereka tidak bisa secara bebas memilih metode pembayaran sesuai dengan preferensinya karena mungkin pemilik toko hanya memiliki rekening dari satu atau dua bank saja.

Solusi dari kedua permasalahan di atas adalah dengan menggunakan payment gateway.

Apa itu Payment Gateway?

Payment gateway adalah perusahaan yang menjadi perantara atau jembatan pembayaran antara customer dengan pemilik toko, di mana mereka menyediakan berbagai metode pembayaran seperti bank transfer dengan berbagai bank, kartu kredit, alfamart, indomaret, dll. Tentu mereka telah bekerjasama dengan media pembayaran tersebut sebelumnya.

Lebih dari itu, payment gateway juga menawarkan solusi integrasi sistem pembayaran dengan sistem kita, sehingga otomasi pembayaran akan lebih mudah kita wujudkan.

Lalu apa yang didapat oleh Payment Gateway ini? mereka mengenakan biaya untuk setiap transaksi yang berhasil dilakukan. Tentu kita harus berhitung jangan sampai biaya transaksi ini lebih tinggi dari keuntungan penjualan produk 😊.

Pada studi kasus kita, payment gateway ini kita gunakan untuk menerima pembayaran dari customer serta melakukan otomasi verifikasi pembayaran.

Ada beberapa perusahaan payment gateway yang ada di Indonesia, diantaranya: midtrans (veritrans), xendit, ipaymu, doku, faspay, dll. Masing-masing tentu memiliki kelebihan dan kekurangannya sendiri. Penulis sendiri telah mencoba beberapa payment gateway tersebut, namun yang sampai ke production hanya midtrans. Oleh karenanya pada bagian ini akan dibahas mengenai integrasi dengan payment gateway ini.

Persiapan Integrasi

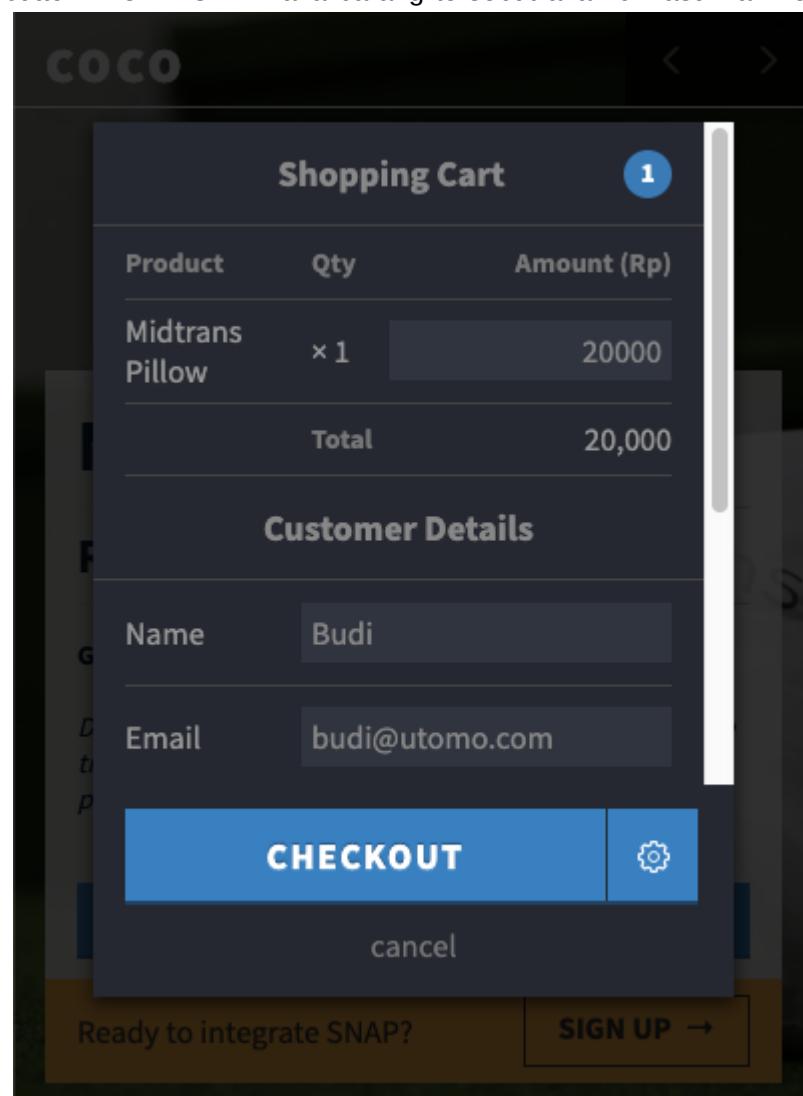
Payment gateway biasanya menyediakan dua metode integrasi yaitu integrasi penuh dan integrasi sebagian. Integrasi penuh artinya user interface dari fitur pembayaran dibuat sendiri oleh pengembang toko online sedangkan integrasi sebagian artinya user interface fitur pembayaran menggunakan user interface dari payment gateway.

Karena pada tutorial ini kita akan menggunakan midtrans, maka sebaiknya kamu coba dulu demonya di <https://demo.midtrans.com> supaya lebih faham alurnya. Pada demo ini menggunakan integrasi sebagian atau dalam bahasa midtrans disebut sebagai web snap.

Pertama customer yang akan memesan barang melihat tampilan sebagai berikut di web toko kita.



Kemudian ketika customer mengklik button "BUY NOW" maka barang tersebut akan dimasukkan ke dalam



keranjang belanja pada web toko kita.

Kemudian customer mengklik button "CHECKOUT" untuk menyelesaikan pemesanannya dan halaman akan

The screenshot shows an 'Order Summary' page from a platform named 'Coco'. At the top right, there is a yellow diagonal banner with the word 'TEST'. The page displays the following information:

- amount:** Rp 20,000
- Order ID:** sample-store-1565213849
- order details:** Midtrans Pillow
- shipping details:** (not visible in the screenshot)
- item(s) amount:** Midtrans Pillow 20,000

At the bottom, there is a large blue button labeled 'CONTINUE'.

diredirect atau dialihkan ke web midtrans

Klik button "CONTINUE" untuk memilih metode pembayaran.

The screenshot shows a 'Select Payment' screen from the 'Coco' platform. It lists several payment methods:

- Credit Card:** Pay with Visa, MasterCard, JCB, or Amex. Includes a 'promo >' button.
- ATM/Bank Transfer:** Pay from ATM Bersama, Prima or Alto.
- GO-PAY:** Pay with your GO-PAY wallet.
- KlikBCA:** Pay with your KlikBCA account.

Misalnya kita memilih ATM/Bank Transfer -> Bank Mandiri, maka berikut ini tampilannya.

Coco

Mandiri ATM
TEST

amount Rp 20,000

Order ID sample-store-1565213849

How to Pay? mandiri

ATM Mandiri Internet Banking

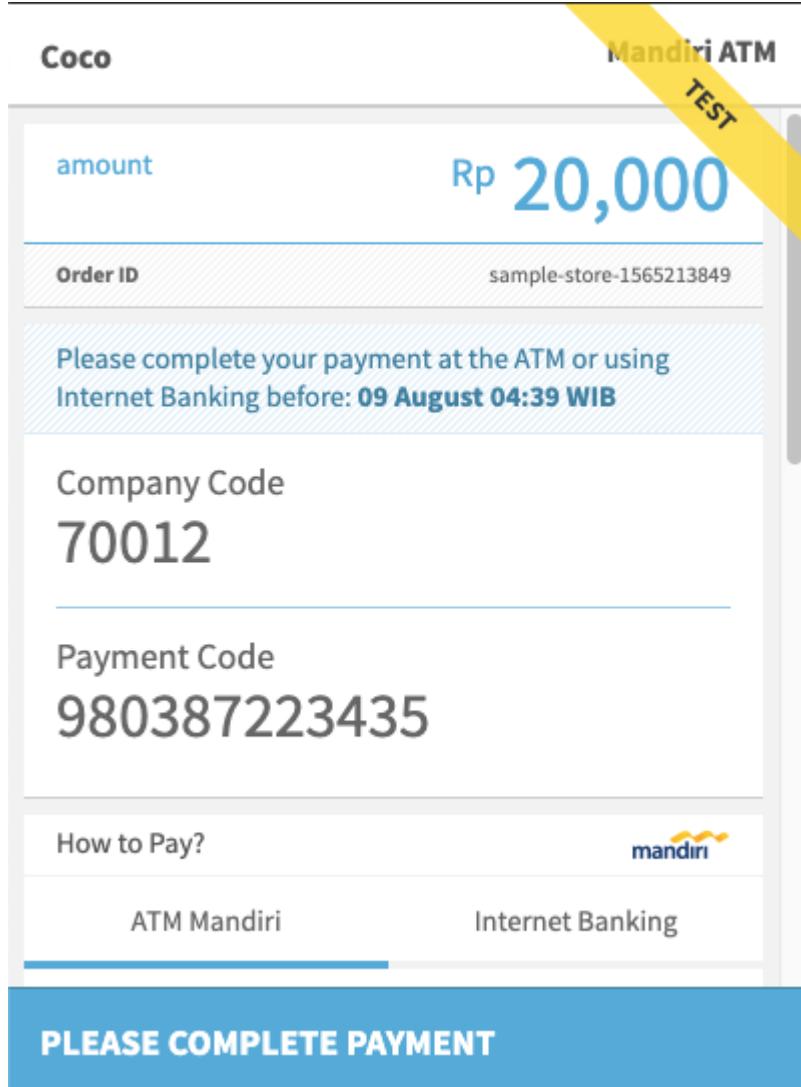
1.  On the main menu, choose Pay/Buy.

2.  Choose Others.

3.  Choose Multi Payment.

SEE ACCOUNT NUMBER >

Klik button "SEE ACCOUNT NUMBER" untuk melihat cara / panduan pembayaran (nomer rekening)



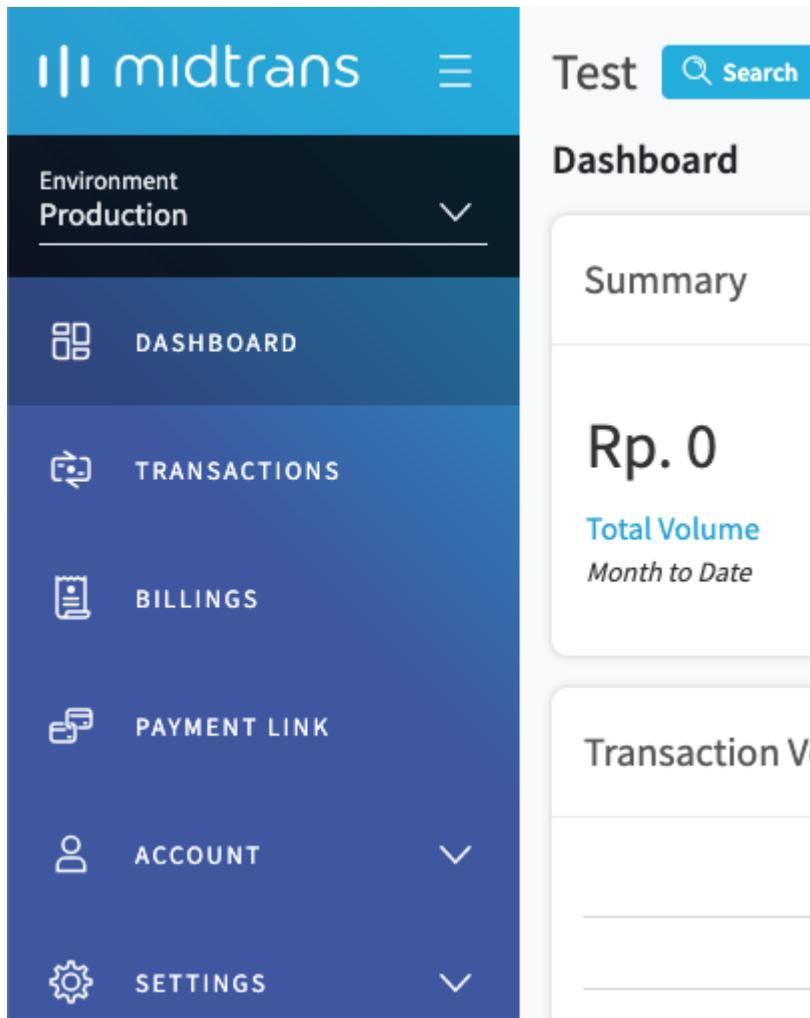
Ketika kita klik button "PLEASE COMPLETE PAYMENT" maka halaman akan diredirect kembali ke web toko kita dan proses selesai.

Demikian secara singkat alurnya jika kita menggunakan payment gateway.

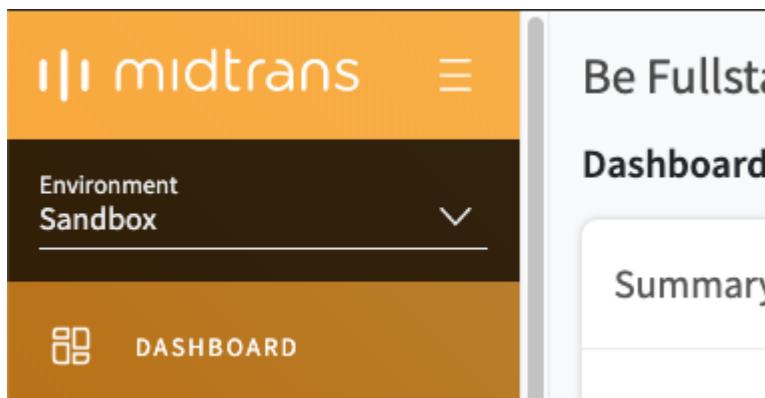
Registrasi Midtrans

Baik, silakan melakukan registrasi ke midtrans setelah mencoba demo tersebut. Masuk ke link <https://midtrans.com/tentang-passport> ada pilihan apakah bisnis kita individu atau berbadan hukum, tentu pada kasus ini kita pilih bisnis individu. Panduan registrasi dan dokumen apa yang perlu dipersiapkan telah dijelaskan pada halaman tersebut silakan diikuti saja.

Setelah registrasi berhasil, silakan login ke midtrans maka kita akan dibawa ke halaman dashboard.



Pada sidebar menu dashboard midtrans tepatnya pada bagian atas terdapat setting atau pilihan environment yaitu production dan sandbox. Mode production artinya pembayaran bisa dilakukan secara real, sedangkan sandbox untuk uji coba saja. Untuk saat ini kita pilih sandbox, dimana hal itu akan mengubah tampilan warna midtrans menjadi oranye.



Adapun menu-menu dibawahnya seperti:

- menu transaction digunakan untuk melihat histori transaksi yang terjadi beserta statusnya;
- menu billings untuk mengupdate data rekening bank kita (untuk penarikan uang);
- menu payment link digunakan untuk membuat link pembayaran, semisal kita jualan via sosmed tanpa web;
- menu account untuk mengupdate data akun kita dan mengelola akun lain yang terafiliasi dengan akun kita (misal: staf keuangan, programmer, dll). Serta yang terakhir
- menu settings yang akan banyak kita gunakan untuk melakukan pengaturan terkait aplikasi.

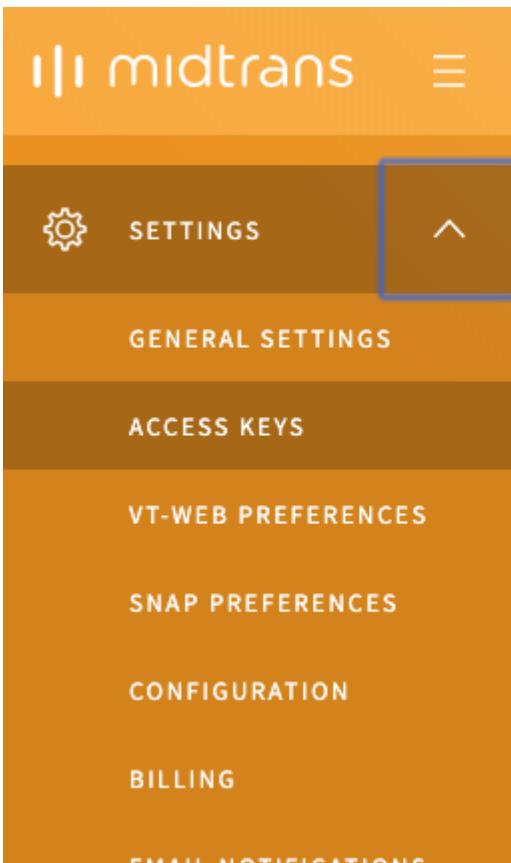
Silakan lengkapi saja data pada form-form yang tersedia pada menu-menu di atas, terutama pada menu Settings > General Settings, yang berisi informasi tentang toko kita.

The screenshot shows the midtrans merchant dashboard interface. At the top left is the midtrans logo. On the right, the text "Be Fullstack Devel" is displayed. Below the logo is a navigation bar with a gear icon labeled "SETTINGS". The main content area has a header "General Settings" and a breadcrumb navigation "Home > Settings > General Settings". The main section is titled "Business Settings" and contains the following fields:

- Merchant Name (i)
- Official Merch. Name (i)
- Director Name
- Director Phone
- NPWP

The left sidebar lists several menu items: GENERAL SETTINGS, ACCESS KEYS, VT-WEB PREFERENCES, SNAP PREFERENCES, CONFIGURATION, BILLING, EMAIL NOTIFICATIONS, PAYMENT LINK, and DAILY REPORT. The "GENERAL SETTINGS" item is highlighted with a darker background.

Informasi tentang key (Merchant ID & ServerKey) yang digunakan untuk berinteraksi antara aplikasi toko kita dengan midtrans dapat dijumpai pada menu Settings > Access Keys.



Access Keys

Home > Settings > Access Keys

Configurations

You will need to know your **Merchant ID**
Please use the Development server

API KEYS

Merchant ID	M075345
Client Key	VT-client-HH
Server Key	VT-server-D-

Catat data Server Key

Kemudian tampilan web snap untuk pembayaran (kita akan menggunakan integrasi sebagian) bisa kita kustomisasi pada menu Settings > Snap Preferences, seperti metode pembayaran yang ingin ditampilkan, urutan bank, warna layout dsb.

Snap Preferences

Set your custom preferences on Snap Payment Page

Theme and Logo Payment Channels Bank List System Settings

Display Name: Be Fullstack Developer

Logo (Max 1 MB):  Remove/Change Logo

Logo Placement Tips:

- Logo will be resized to 72 x 240 pixels
- Use PNG image instead of GIF if your logo is transparent
- Max file size is 1 MB
- Uploaded Logo will replace Display Name

Theme Color: 

PREVIEW

ORDER SUMMARY

amount: Rp 5,104,000

order ID: MID-62215736789

order details: Magic Wand x 2 shipping details: amount (Rp): 800,000

Integrasi Midtrans

Dokumentasi teknis untuk developer bisa kita jumpai pada tautan ini <https://docs.midtrans.com/en/welcome/index.html>, silakan dibaca-baca dahulu.

Midtrans menawarkan berbagai metode integrasi tergantung kebutuhan kita dan jenis aplikasi kita, yaitu Snap (integrasi sebagian), Payment link, MobileSDK (aplikasi mobile), serta COREAPI (integrasi penuh). Pada pembahasan ini kita akan fokus ke integrasi SNAP.

SNAP merupakan portal pembayaran yang memungkinkan merchant untuk menggunakan sistem pembayaran Midtrans dengan memunculkan halaman pembayaran Midtrans langsung dihalaman pembayaran Anda. Setup-nya mudah dan tidak dikenakan tagihan bulanan, cocok untuk bisnis skala kecil dan menengah.

Instalasi Pustaka Pendukung

Di PHP, terhadap pustaka pendukung yang memudahkan kita mengintegrasikan midtrans yaitu: <https://github.com/veritrans/veritrans-php> (pustaka ini disarankan penggunaannya oleh midtrans jika kita menggunakan PHP).

Pada projek Laravel, kita instalasi pustaka midtrans via composer dengan menjalankan perintah berikut:

```
1 | composer require veritrans/veritrans-php:dev-master
```

Update Endpoint Payment (Laravel)

Setelah kita instalasi pustaka midtrans, maka berikutnya kita akan modifikasi sedikit endpoint payment di mana pada bagian akhir dari endpoint ini adalah sebagai berikut:

```
1 | //..
2 | if($error==0){
3 |     DB::commit();
4 |     $status = 'success';
5 |     $message = 'Transaction success';
6 |     $data = [
7 |         'order_id' => $order->id,
8 |         'total_bill' => $total_bill,
9 |         'invoice_number' => $order->invoice_number,
10 |     ];
11 | }
```

Pada bagian ini, kita akan menambahkan fungsi untuk mendaftarkan transaksi ini ke midtrans, kemudian midtrans akan memprosesnya dan mengembalikan outputnya berupa URL atau link pembayaran.

```
1 | //..
2 | if($error==0){
3 |     DB::commit();
4 |     $status = 'success';
5 |     $message = 'Transaction success';
6 |
7 |     /* MULAI MIDTRANS */
8 |     \Veritrans_Config::$serverKey = "SERVER KEY MIDTRANS";
9 |     \Veritrans_Config::$isProduction = false;
10 |    \Veritrans_Config::$isSanitized = true;
11 |    \Veritrans_Config::$is3ds = true;
12 |    $transaction_data = [
13 |        'transaction_details' => [
14 |            'order_id' => $order->invoice_number,
15 |            'gross_amount' => $total_bill,
16 |        ]
17 |    ];
18 |    $payment_link = \Veritrans_Snap::createTransaction($transaction_data) -
```

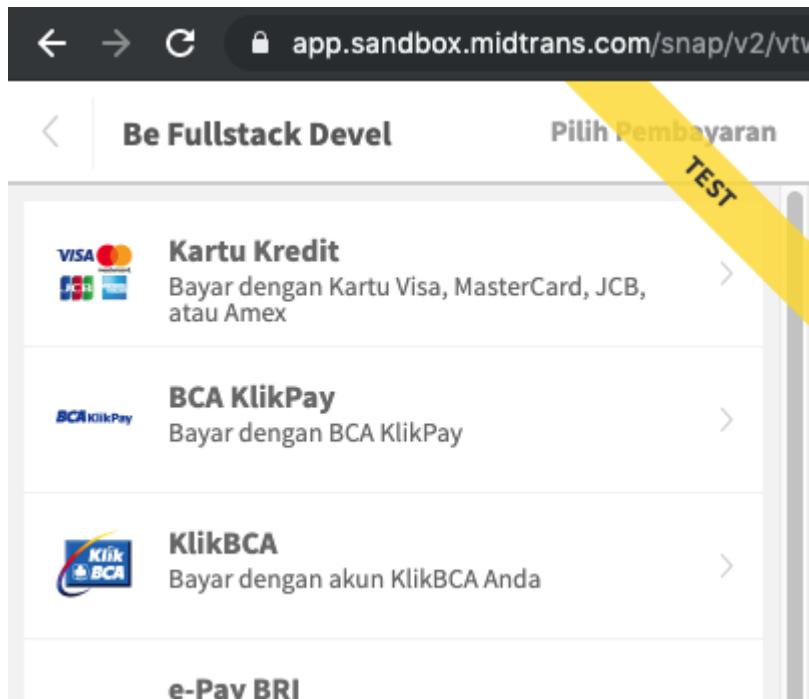
```

19 >redirect_url;
20     $data = [
21         'payment_link' => $payment_link,
22     ];
23     /* SELESAI MIDTRANS */
}

```

Apa itu payment_link? payment link adalah halaman web snap midtrans untuk pembayaran transaksi yang telah kita daftarkan sebelumnya.

Contohnya kurang lebih sebagaimana berikut: <https://app.sandbox.midtrans.com/snap/v2/vtweb/f753dd5d-403c-47b9-82e2-0136b4f7919c>



Update Halaman Checkout (Vue)

Pada projek Vue, kita akan mengupdate halaman checkout, yaitu pada method pay(), di mana method ini pada kode sebelumnya akan mengarahkan atau meredirect ke route /payment atau halaman pembayaran.

```

1 this.axios.post('/payment', formData, config)
2     .then((response) => {
3         let { data } = response
4         if(data && data.status=='success'){
5             this.setPayment(data.data) // <= ini
6             this.$router.push({path: "/payment"})
7             this.setCart([])
8         }
}

```

Maka pada kasus ini, halaman akan kita redirect sesuai data payment_link hasil dari endpoint payment.

```

1 this.axios.post('/payment', formData, config)
2     .then((response) => {
3         let { data } = response
4         if(data && data.status=='success'){
5             this.setCart([])
6             let payment_link = data.data.payment_link
}

```

```

7     window.location = payment_link
8 }
```

Silakan dicoba, checkout dan klik button pay, maka halaman akan diarahkan ke web snap (halaman pembayaran midtrans). Pada web snap tersebut, user dapat memilih metode pembayaran yang dia inginkan dan ketika user telah selesai melakukannya maka kita bisa meredirect kembali ke halaman web kita atau ke halaman lain, adapun pengaturannya dapat kita jumpai pada halaman admin midtrans menu Setting > Configurations (Redirect Settings: Finish Redirect URL).

Menangani Notifikasi Pembayaran

Midtrans juga memiliki fitur notifikasi pembayaran. Fitur ini dapat kita manfaatkan untuk melakukan action tertentu sesaat setelah user berhasil melakukan pembayaran, misal untuk mengaktifasi keanggotaan user, mengirimkan email ke user, mengirimkan notifikasi ke penjual dsb. Adapun pengaturannya dapat kamu jumpai pada halaman admin midtrans menu Setting > Configurations (Redirect Settings: Payment Notification URL*).

Implementasinya, kita bisa buat sebuah endpoint pada Laravel. Berikut ini contoh paling minimal dari endpoint untuk menangani notifikasi pembayaran dari Midtrans.

```

1 \Veritrans_Config::$isProduction = false;
2 \Veritrans_Config::$serverKey = '<your serverkey>';
3 $notif = new \Veritrans_Notification();
4 $transaction = $notif->transaction_status;
5 $type = $notif->payment_type;
6 $order_id = $notif->order_id;
7 $fraud = $notif->fraud_status;
8 if ($transaction == 'settlement'){
9     echo "Transaction order_id: " . $order_id . " successfully";
10    // implementasikan disini
11 }
```

Catatan: `settlement` adalah status pembayaran sukses.

Silakan bereksplorasi.

Membuat Halaman Profile

Halaman ini menampilkan data profile user yang sudah login yang dapat diakses melalui link profile pada sidebar menu.

Link Profile di SideBar

Buka file `src/App.vue`, pada properti data, tambahkan menu baru pada data menus.

```

1 data: () => ({
2     drawer: false,
3     menus: [
4         { title: 'Home', icon: 'mdi-home', route: '/' },
5         { title: 'Profile', icon: 'mdi-account', route: '/profile', auth:
6             true },
7         { title: 'About', icon: 'mdi-help-box', route: '/about' },
8     ],
9 }),
10
```

Flag auth: true untuk membedakan menu yang buat publik atau menu untuk user yang sudah login.

Lalu pada templatanya ubah menjadi sebagai berikut.

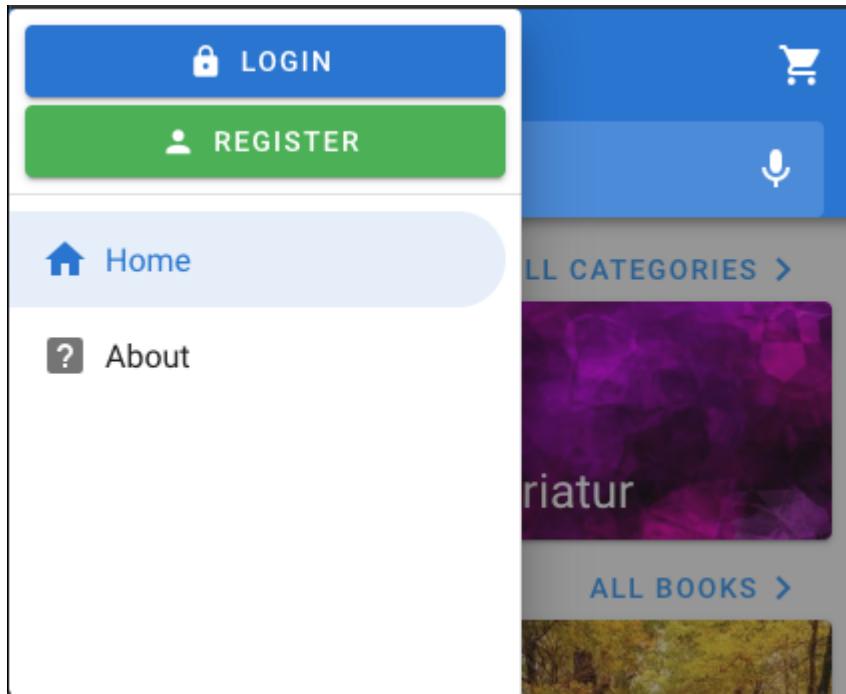
```

1 <v-list shaped>
2   <template v-for="(item, index) in menus">
3     <v-list-item
4       :key="`menu-`+index"
5       :to="item.route"
6       v-if="!item.auth || (item.auth && !guest)"
7     >
8       <v-icon left>{{ item.icon }}</v-icon>
9       <v-list-item-content>
10         <v-list-item-title>{{ item.title }}</v-list-item-title>
11       </v-list-item-content>
12     </v-list-item>
13   </template>
14 </v-list>
```

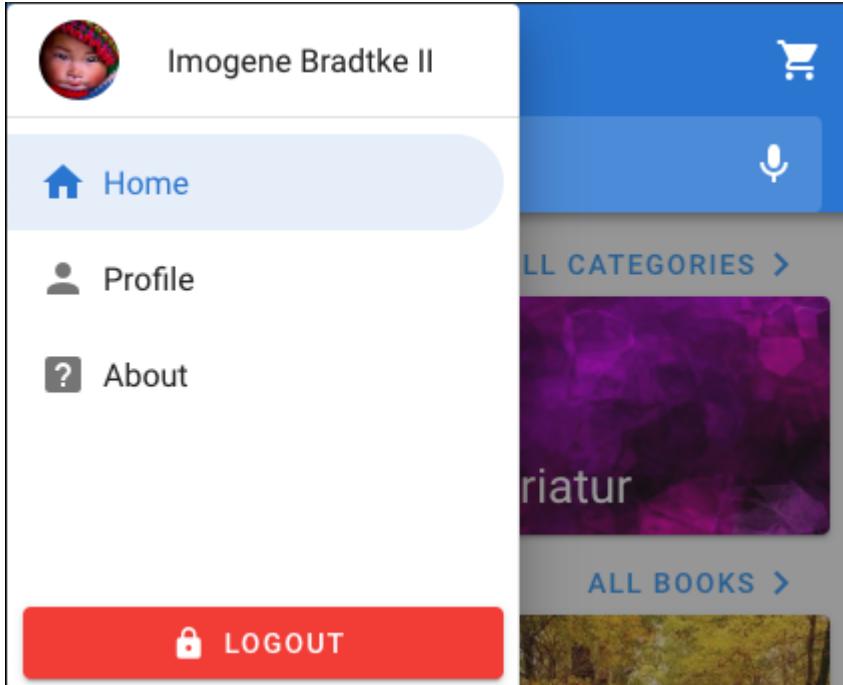
Poinnya pada bagian ini `v-if="!item.auth || (item.auth && !guest)"` yaitu menu hanya diampilkan dalam dua kondisi.

1. Tanpa ada flag auth
2. Jika ada flag auth maka harus user yang sudah login atau bukan guest

Mari kita coba, ketika belum login maka menu profile tidak muncul



Jika sudah login maka akan muncul menu tersebut.



Layout Halaman Profile

Component ini hanya akan menampilkan profile dari user yang sudah login. Simpan pada file /src/views/Profile.vue

```

1  <template>
2      <div>
3          <v-subheader>Your Profile</v-subheader>
4          <v-card flat>
5              <v-container>
6                  <v-simple-table>
7                      <tbody>
8                          <tr v-for="(value, key) in user" :key="key">
9                              <td>{{ key }}</td>
10                         <td>{{ value }}</td>
11                     </tr>
12                 </tbody>
13             </v-simple-table>
14         </v-container>
15     </v-card>
16     </div>
17 </template>
18 <script>
19 import { mapGetters } from 'vuex'
20 export default {
21     computed: {
22         ...mapGetters({
23             'user':'auth/user'
24         }),
25     }
26 }
27 </script>

```

Jangan lupa pada src/router.js tambahkan routing untuk component profile.

```

1  {
2    path: '/profile',
3    name: 'profile',
4    component: () => import( /* webpackChunkName: "profile" */
5      './views/Profile.vue'),
6    meta: { auth: true }
7  },

```

Mari kita test.

id	1
name	Imogene Bradtke II
email	akautzer@example.net
email_verified_at	
created_at	2019-07-31 04:26:51
updated_at	2019-08-04 00:27:58

Yap berhasil, silakan dikembangkan menjadi misalnya form update profile dsb.

Membuat Halaman Histori Belanja

Halaman ini menampilkan daftar histori belanja yang pernah dilakukan user dalam bentuk list. Pada tiap itemnya terdapat kode pemesanan, tanggal dan status.

Halaman ini dapat diakses melalui link my-order pada sidebar menu.

Endpoint My Order

Pada projek Laravel, controller Shop, tambahkan fungsi myOrder untuk meng-query data pemesanan user berdasarkan tabel orders.

```

1  public function myOrder(Request $request)
2  {
3    $user = Auth::user();
4    $status = "error";
5    $message = "";
6    $data = [];
7    if($user){
8      $orders = \App\Order::select('*')
9        ->where('user_id', '=', $user->id)

```

```

Licensed to Riva Pebrian - lvhotpebrian@gmail.com - 085324454424 at 22/10/2019 20:40:19
10    ->orderBy('id', 'DESC')
11    ->get();
12
13    $status = "success";
14    $message = "data my order ";
15    $data = $orders;
16 }
17 else{
18     $message = "User not found";
19 }
20
21 return response()->json([
22     'status' => $status,
23     'message' => $message,
24     'data' => $data
25 ], 200);
26 }

```

Jangan lupa daftarkan routing myOrder pada api.php

```

1 // route lain
2
3 // private
4 Route::middleware(['auth:api'])->group(function () {
5     // ... route lain
6     Route::get('my-order', 'ShopController@myOrder'); // <= ini
7     //...
8 });

```

Silakan dicoba.

Akses URL ini <http://larashop-api.test/v1/my-order> menggunakan postman method GET dan authentication bearer token, maka hasilnya.

```

1 {
2     "status": "success",
3     "message": "data my order ",
4     "data": [
5         {
6             "id": 8,
7             "user_id": 1,
8             "total_bill": 322000,
9             "invoice_number": "20180909021916",
10            "courier_service": "jne-REG",
11            "status": "SUBMIT",
12            "created_at": "2018-09-09 02:19:16",
13            "updated_at": "2018-09-09 02:19:17"
14        }
15    ]
16 }

```

Link My Order di SideBar

Buka file src/App.vue, pada properti data, tambahkan menu baru pada data menus.

```

1  data: () => ({
2    drawer: false,
3    menus: [
4      { title: 'Home', icon: 'mdi-home', route: '/' },
5      { title: 'Profile', icon: 'mdi-account', route: '/profile', auth:
6        true },
7      { title: 'My Order', icon: 'mdi-shopping', route: '/my-order', auth:
8        true }, // <= ini
9      { title: 'About', icon: 'mdi-help-box', route: '/about' },
10     ],
11   }),

```

Layout Halaman My Order

Component ini hanya akan menampilkan histori pemesanan dari user yang sudah login. Simpan pada file /src/views/MyOrder.vue

```

1  <template>
2    <div>
3      <v-subheader>Histori Belanja</v-subheader>
4      <v-card flat>
5        <v-container>
6          <v-simple-table>
7            <tbody>
8              <tr v-for="item in items" :key="item.id">
9                <td>
10                  Invoice: {{ item.invoice_number }}
11                  <div class="primary--text title">Rp. {{ item.total_bill.toLocaleString('id-ID') }}</div>
12                  <small>date: {{ item.updated_at }}. courier: {{ item.courier_service }}</small>
13                </td>
14                <td>
15                  {{ item.status }}
16                </td>
17              </tr>
18            </tbody>
19            </v-simple-table>
20        </v-container>
21      </v-card>
22    </div>
23  </template>
24  <script>
25    import { mapGetters, mapActions } from 'vuex'
26    export default {
27      data () {
28        return {
29          items: [],
30        }
31      },
32      computed: {
33
34

```

```

35     ...mapGetters({
36         user      : 'auth/user',
37     })
38 },
39 methods: {
40     ...mapActions({
41         setAlert  : 'alert/set',
42     }),
43 },
44 mounted(){
45     let config = {
46         headers: {
47             'Authorization': 'Bearer ' + this.user.api_token,
48         },
49     }
50     this.axios.get('/my-order',config)
51     .then((response) => {
52         let {data} = response.data
53         this.items = data
54     })
55     .catch((error) => {
56         let {data} = error.response
57         this.setAlert({
58             status : true,
59             text   : data.message,
60             color  : 'error',
61         })
62     })
63 }
64
}
</script>

```

Jangan lupa pada src/router.js tambahkan routing untuk component MyOrder.

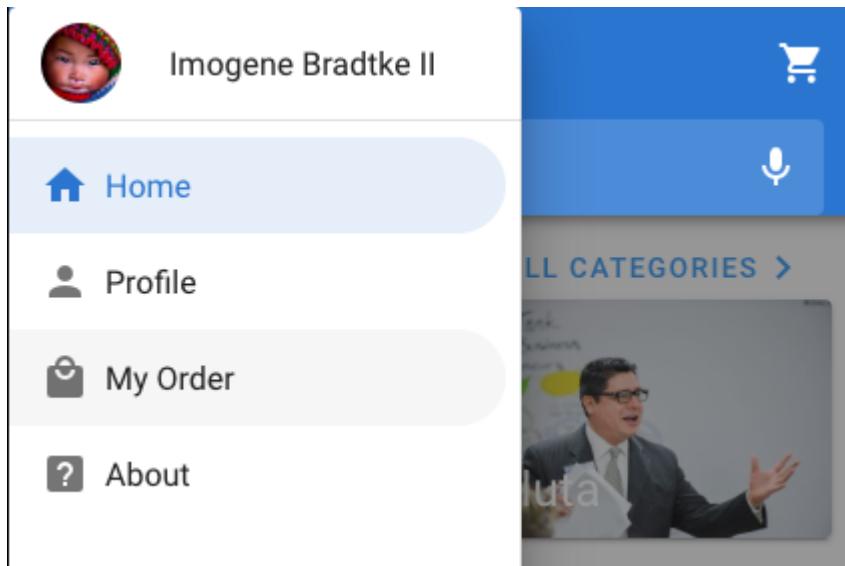
```

1  {
2     path: '/my-order',
3     name: 'my-order',
4     component: () => import( /* webpackChunkName: "my-order" */ 
5       './views/MyOrder.vue'),
6     meta: { auth: true }
7 ,

```

Mari kita test.

Pada sidebar, pilih menu my order.



Hasilnya.

Invoice	Amount	Action	Date
201908040001013	Rp. 218.000	SUBMIT	2019-08-04 00:10:13
20190804000551	Rp. 218.000	SUBMIT	2019-08-04 00:05:51
20190804000507	Rp. 218.000	SUBMIT	2019-08-04 00:05:07

Yes selesai.

Mengatasi Refresh Pada Browser

Apa yang terjadi jika browser diredirect? yap semua data state akan hilang, kenapa itu? yap karena state sebenarnya hanya disimpan pada variabel di Js dan itu artinya hanya disimpan sementara.

Bagaimana supaya tidak hilang ketika diredirect? maka kita bisa simpan data state itu di localstorage supaya persisted. Artinya ketika terjadi mutation, maka state di localstorage diperbaharui.

Nah untungnya ada pustaka yang memudahkan kita melakukannya yaitu salah satunya `vuex-persist`.

Mari kita install dengan perintah `npm install vuex-persist`

Kemudian aplikasikan pada Vuex sebagai sebuah plugins di file `src/store.js`.

```
1 import VuexPersist from 'vuex-persist'  
2  
3 const vuexPersist = new VuexPersist({  
4   key: 'my-app',  
5   storage: localStorage
```

```
6  })
7
8  Vue.use(Vuex)
9
10 export default new Vuex.Store({
11   plugins: [vuexPersist.plugin],
```

Jika sudah, silakan coba refresh, apakah state masih hilang?

Source Code

Source code lengkap dapat kamu jumpai pada tautan berikut:

- <https://github.com/laravel-vue-book/larashop-api>
- <https://github.com/laravel-vue-book/vueshop>

Kesimpulan

Akhirnya selesai juga studi kasus kita setelah melewati pembahasan yang cukup panjang. Memang tidak banyak dibahas tentang detail dari component bawaan framework Vuetify oleh karena itu untuk melengkapi pemahaman kamu maka sebaiknya detail component bisa merujuk kepada dokumentasi dari Vuetify. Di sana kamu akan dapat banyak varian penerapan dari suatu component dan hal itu tidak akan mungkin dibahas pada buku yang cukup terbatas ini.

Masih banyak yang bisa kamu kembangkan dari studi kasus ini, tapi setidaknya penulis telah menyampaikan kepadamu tentang gambaran besarnya dan fitur-fitur pentingnya.

Pada bagian berikutnya kita akan belajar tentang cara deployment ke server hosting sehingga aplikasimu bisa dinikmati oleh dunia nyata 😊.

Sudah di penghujung nih 😊

Deployment

Intro

Pada bab ini kita akan belajar tentang bagaimana langkah-langkah men-deploy / mempublikasi aplikasi kita sehingga bisa online dan diakses oleh publik. Secara umum, kita bisa mendeploy aplikasi berbasis web kita pada shared hosting, virtual private server (VPS), dan dedicated server.

Layanan hosting atau shared hosting artinya semua infrastruktur server dan tools software telah terinstalasi di server sehingga yang kita lakukan saat deploy hanya mengunggah file kode program dan database. Layanan ini digunakan secara bersama atau sharing dengan pengguna lain namun dengan hak akses yang berbeda.

Layanan VPS artinya infrastruktur server telah tersedia namun tools software belum terinstalasi sehingga yang kita lakukan saat deploy adalah menginstalasi & mengkonfigurasi tools software seperti sistem operasi, web server, PHP, dsb serta mengunggah file kode program dan database. Layanan ini dibangun dengan menggunakan tools virtualisasi sehingga dapat digunakan secara bersama atau sharing dengan pengguna lain namun dengan hak akses yang berbeda.

Dedicated server artinya server fisik yang kita gunakan sendiri (tidak sharing), kita bisa sewa atau server milik kita sendiri yang kita lakukan saat deploy sama dengan yang kita lakukan pada VPS.

Pada buku ini, kita akan fokus pada proses deployment menggunakan layanan hosting.

Catatan: Hosting provider yang penulis gunakan untuk deploy pada buku ini adalah Domainesia <https://domainesia.com> sekaligus salah satu sponsor utama dari buku ini. Oleh karena itu jika kamu menggunakan provider lain silakan menyesuaikan. Web server yang telah terinstalasi pada hosting adalah Apache sehingga sedikit berbeda dengan web server saat development yaitu Nginx, tapi itu bukan masalah.

Diskon Hosting & VPS

Khusus pembaca buku ini, kami menyediakan diskon khusus sebesar 30% dari harga reguler untuk pembelian produk Hosting & VPS di <https://domainesia.com>, berikut ketentuannya:

- Hosting, gunakan kode kupon **FullstackHosting**
- VPS, gunakan kode kupon **FullstackVPS**

Catatan: masukkan kode kupon di atas setiap transaksi. Kode ini dapat digunakan secara berulang.

Persiapan

Sebelum kita melakukan deployment aplikasi dan web service maka kita perlu melakukan beberapa persiapan. Untuk menyamakan persepsi, maka kita akan menyebut aplikasi toko buku yang kita buat dengan menggunakan Vue sebagai aplikasi web frontend, sedangkan web service toko buku yang kita buat menggunakan Laravel sebagai aplikasi web service. Adapun aplikasi yang dibuat dengan Laravel untuk manajemen toko kita sebut sebagai aplikasi web backend.

Persiapan Aplikasi Web Frontend

Sesuaikan konstanta global pada file `.env.production` dengan alamat aplikasi web service dan aplikasi web backend.

```
1 VUE_APP_NAME=Vueshop  
2 VUE_APP_BACKEND_URL=https://api.vueshop.id
```

3 | VUE_APP_API_URL=https://api.vueshop.id

Catatan: pada contoh ini, alamat URL-nya sama antara aplikasi web service dan web backend.

Untuk aplikasi web frontend karena bentuknya hanya static maka sebenarnya untuk melakukan deployment cukup simple yaitu build kode kita menggunakan perintah berikut.

```
npm run build
```

File	Size	Gzipped
dist/js/chunk-vendors.f2ff08e3.js	568.98 kb	142.07 kb
dist/js/app.d1e5b9d3.js	21.02 kb	6.86 kb
dist/js/checkout.99f8d3c5.js	7.21 kb	2.22 kb
dist/js/register.66671352.js	3.69 kb	1.49 kb
dist/js/categories.ebd9edc8.js	3.53 kb	1.04 kb
dist/js/login.a7ad912a.js	2.74 kb	1.24 kb
dist/js/cart.6580fe37.js	2.66 kb	1.12 kb
dist/js/book.5653794c.js	2.17 kb	0.96 kb
dist/js/category.f5b9ea13.js	2.08 kb	1.00 kb
dist/precache-manifest.09cb4b6028eaae2e21658e63157c5fba.js	1.99 kb	0.69 kb
dist/js/search.c12f9fea.js	1.96 kb	0.99 kb
dist/js/payment.86eea561.js	1.65 kb	0.74 kb
dist/js/my-order.910adec5.js	1.36 kb	0.75 kb
dist/js/c-alert.4583bf17.js	0.95 kb	0.54 kb
dist/service-worker.js	0.94 kb	0.53 kb
dist/js/profile.0c7d90cf.js	0.55 kb	0.38 kb
dist/css/chunk-vendors.bd48805f.css	253.87 kb	31.08 kb
dist/css/categories.943639a1.css	0.17 kb	0.14 kb
dist/css/category.7d8fc395.css	0.14 kb	0.13 kb
dist/css/app.ae414c3b.css	0.14 kb	0.13 kb

Images and other types of assets omitted.

DONE Build complete. The `dist` directory is ready to be deployed.
INFO Check out deployment instructions at <https://cli.vuejs.org/guide/deployment.html>

Maka akan tergenerate file-file yang siap dideploy pada folder `dist`. Yap, hanya file-file dalam folder `dist` itulah yang kita unggah ke server.

Persiapan Aplikasi Web Service

Untuk aplikasi web service, maka perlu kita perhatikan beberapa hal.

Konfigurasi

Kita perlu mengatur konfigurasi aplikasi sebelum di deploy. Berikut ini checklistnya.

Matikan Mode Debug

Mode ini hanya untuk development, oleh karenanya kita perlu non aktifkan menjadi mode tersebut. Pengaturannya terdapat pada file `.env` pada root folder `larashop-api`.

1 | APP_DEBUG=false

Sesuaikan Konfigurasi Database

Sesuaikan konfigurasi untuk koneksi database dengan server production. Adapun pengaturannya juga pada file .env.

```

1 DB_CONNECTION=mysql
2 DB_HOST=mysql
3 DB_PORT=3306
4 DB_DATABASE=database
5 DB_USERNAME=username
6 DB_PASSWORD=password

```

Optimization

Untuk mengoptimalkan kinerja dari aplikasi Laravel kita maka ada beberapa hal yang perlu kita lakukan yaitu:

Autoloader Optimization

Supaya composer autoloader dapat dengan cepat menemukan file class yang dirujuk, caranya pada terminal jalankan perintah berikut.

```
1 composer install --optimize-autoloader --no-dev
```

```

laradock — root@28de49479092:/var/www/larashop-api — docker ↵ docker...
[root@28de49479092:/var/www/larashop-api# composer install --optimize-autoloader
--no-dev
Do not run Composer as root/super user! See https://getcomposer.org/root for details
Loading composer repositories with package information
Installing dependencies from lock file
Package operations: 0 installs, 0 updates, 33 removals
- Removing webmozart/assert (1.3.0)
- Removing theseer/tokenizer (1.1.0)
- Removing sebastian/version (2.0.1)
- Removing sebastian/resource-operations (1.0.0)
- Removing sebastian/recursion-context (3.0.0)
- Removing sebastian/object-reflector (1.1.1)
- Removing sebastian/object-enumerator (3.0.3)
- Removing sebastian/global-state (2.0.0)
- Removing sebastian/exporter (3.1.0)
- Removing sebastian/environment (3.1.0)
- Removing phpunit/php-code-coverage (6.0.7)
- Removing phpspec/prophecy (1.8.0)
- Removing phpdocumentor/type-resolver (0.4.0)
- Removing phpdocumentor/reflection-docblock (4.3.0)
- Removing phpdocumentor/reflection-common (1.0.1)
- Removing phar-io/version (2.0.1)
- Removing phar-io/manifest (1.0.3)
- Removing nunomaduro/collision (v2.0.3)
- Removing myclabs深深-copy (1.8.1)
- Removing mockery/mockery (1.1.0)
- Removing hamcrest/hamcrest-php (v2.0.0)
- Removing fzhaninotto/faker (v1.8.0)
- Removing filp/whoops (2.2.0)
- Removing doctrine/instantiator (1.1.0)
- Removing beyondcode/laravel-dump-server (1.2.1)
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover
Discovered Package: barryvdh/laravel-cors
Discovered Package: fideloper/proxy
Discovered Package: laravel/tinker
Discovered Package: nesbot/carbon
Package manifest generated successfully.
root@28de49479092:/var/www/larashop-api#

```

Optimizing Configuration Loading

Untuk mengoptimalkan loading konfigurasi, Laravel menyarankan kita menjalankan perintah berikut.

```
1 | php artisan config:cache
```



```
[root@28de49479092:/var/www/larashop-api]# php artisan config:cache
Configuration cache cleared!
Configuration cached successfully!
```

Perintah ini akan menggabungkan semua konfigurasi pada Laravel menjadi satu file cache di mana hal ini akan mempercepat Laravel dalam membaca konfigurasi.

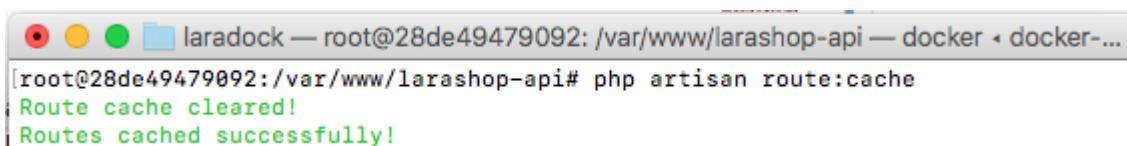
Nah hanya saja apabila ini kita lakukan di development maka akan menyebabkan error karena penggunaan absolute path pada config yang bisa jadi berbeda dengan path di server production. Jika terlanjur maka gunakan perintah berikut.

```
1 | php artisan config:clear
```

Optimizing Route Loading

Jika kita mengembangkan aplikasi dengan routing yang complex maka sebaiknya kita menjalankan perintah berikut sebelum deployment.

```
1 | php artisan route:cache
```



```
[root@28de49479092:/var/www/larashop-api]# php artisan route:cache
Route cache cleared!
Routes cached successfully!
```

Perintah ini akan meregister semua aturan routing ke dalam single method yang dipanggil dalam file cache, hal ini tentu akan meningkatkan performa sistem.

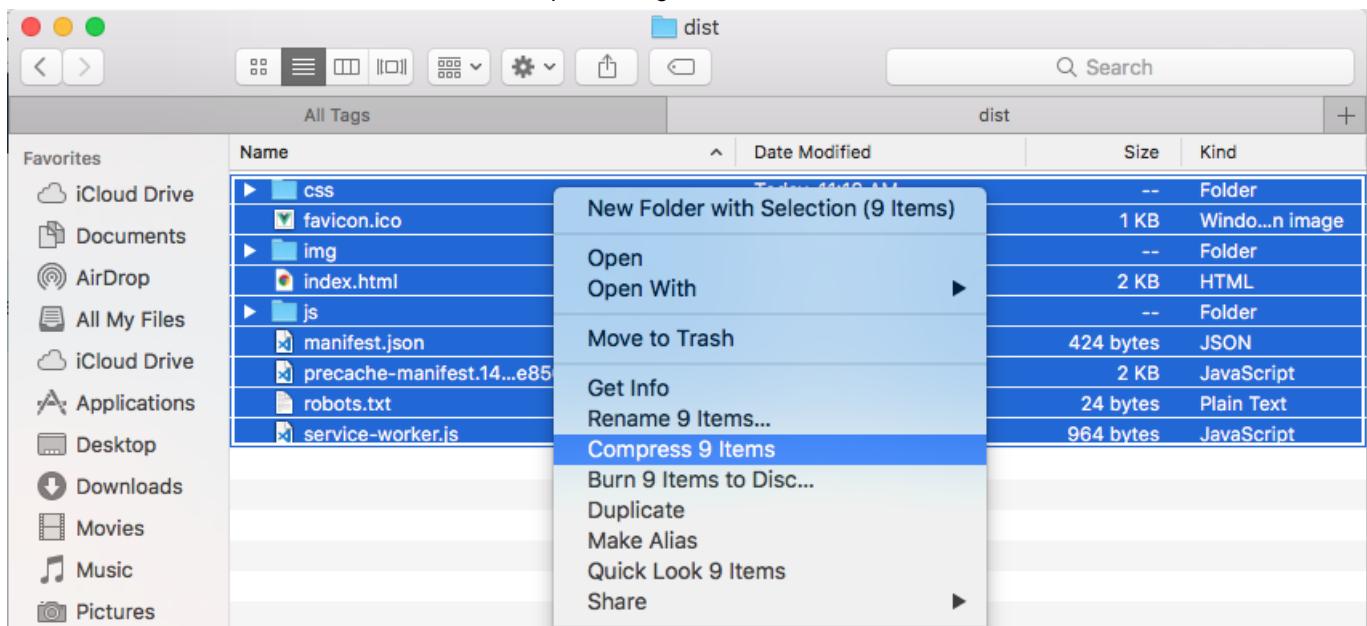
Proses Deployment

Deployment adalah proses mengunggah file-file aplikasi dan melakukan pengaturan supaya aplikasi bisa diakses oleh Publik atau user yang mengetahui alamat URL aplikasi kita. Ada dua hal yang perlu kita siapkan yaitu domain atau alamat URL aplikasi kita dan web hosting yaitu tempat kode aplikasi dan database kita diletakkan.

Pada web hosting, aplikasi web server, PHP dan databasenya umumnya sudah disediakan sehingga kita tingga mengunggah file-file kode dan database kita ke server tersebut. Di samping hosting sebenarnya ada model lain yaitu VPS atau Virtual Private Server, pada kasus ini kita harus menyiapkan sendiri (menginstalasi) sistem operasi yang digunakan, web server, PHP, dan database serta tentu segala macam extensionnya.

Deployment Aplikasi Web Frontend

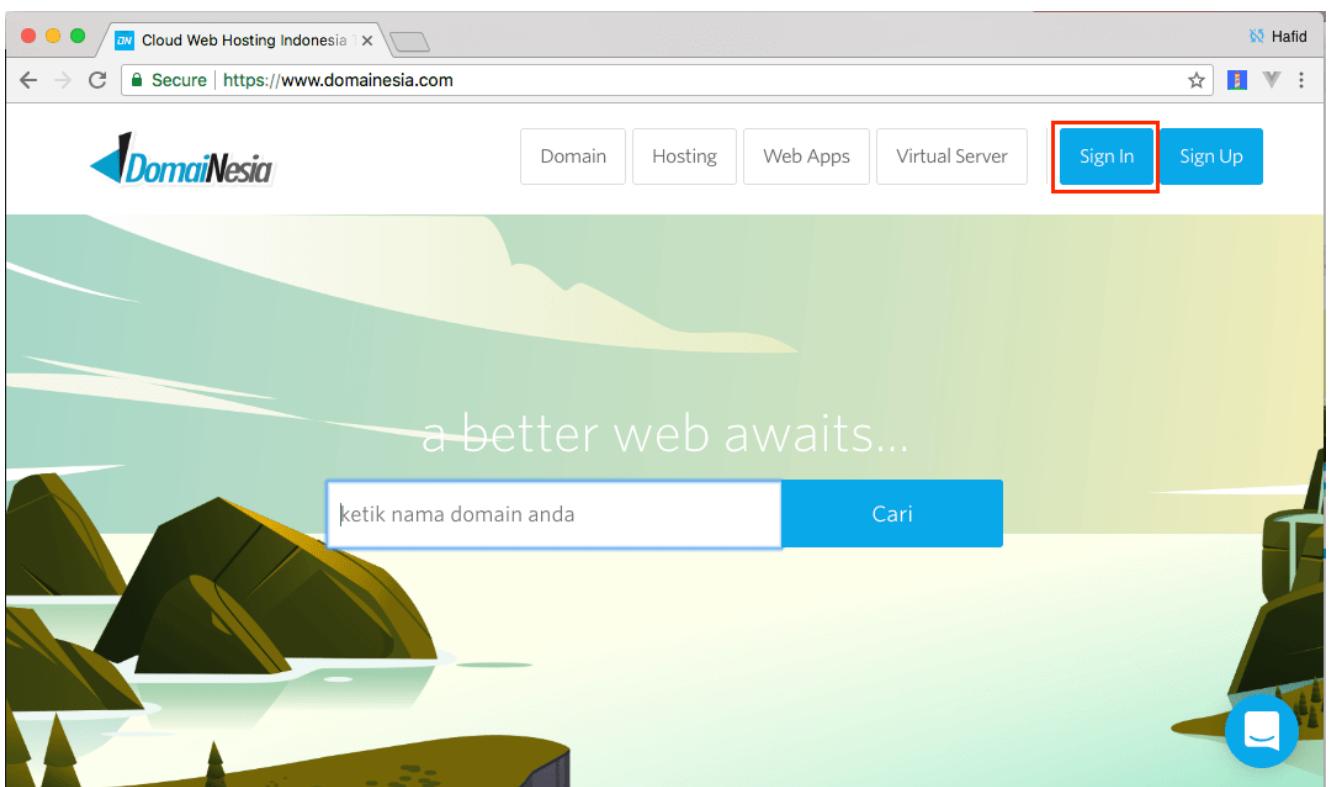
Setelah aplikasi web frontend kita siap di deploy, maka kompres semua file yang berada pada folder dist hal ini untuk memudahkan kita saat mengunggahnya ke server.



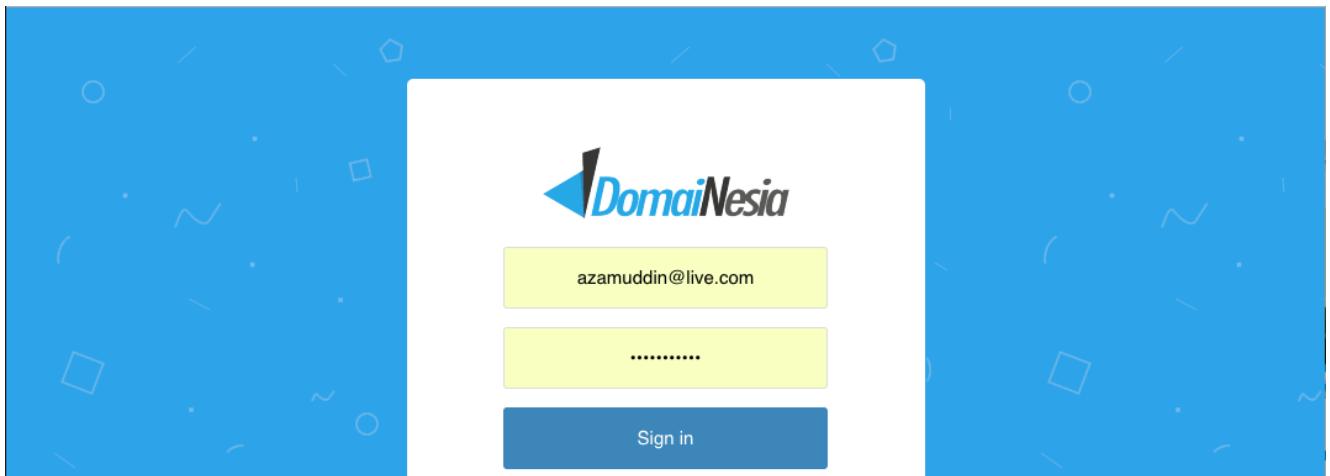
Pada contoh ini, kita menggunakan domain vueshop.id dan aplikasi web frontend akan diunggah ke root dari vueshop.id (public_html).

1. Buka

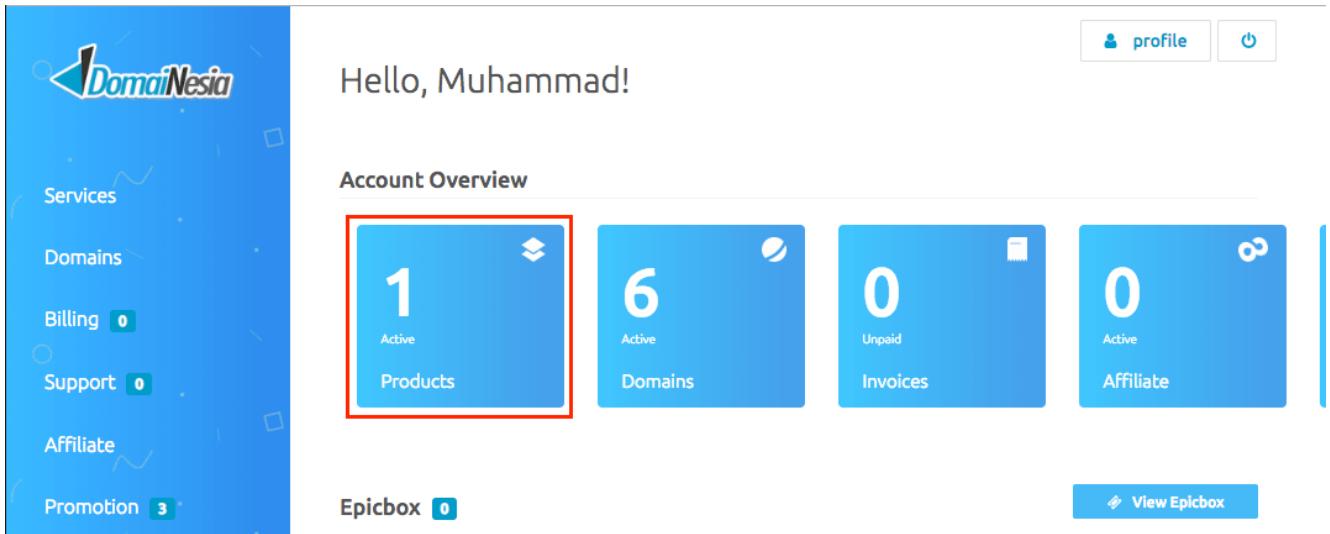
domainesia



2. Login



3. Pada halaman dashboard, pilih products



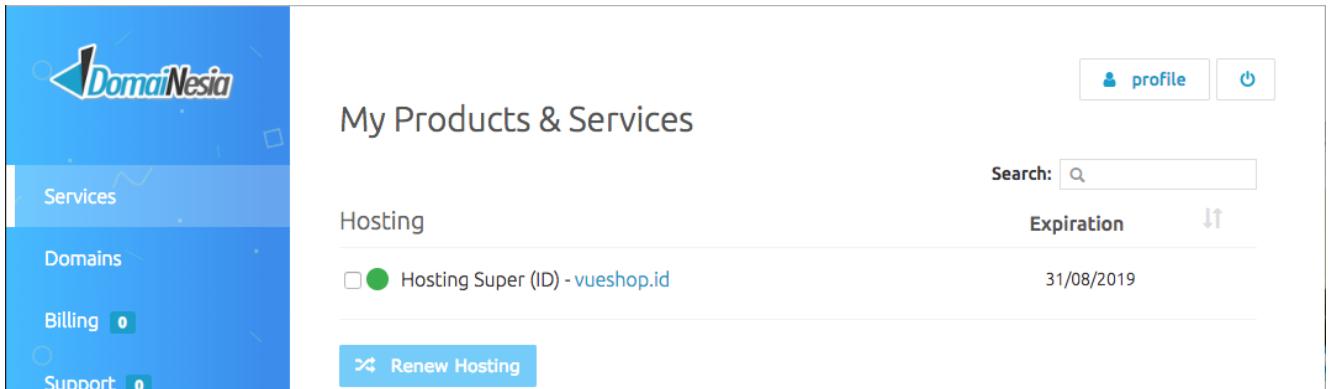
Hello, Muhammad!

Account Overview

Active Products	Active Domains	Unpaid Invoices	Active Affiliate
1	6	0	0

Epicbox 0 [View Epicbox](#)

4. Pada halaman dashboard products, pilih hostingnya yaitu pada contoh ini Vueshop



My Products & Services

Hosting

Hosting	Expiration
Hosting Super (ID) - vueshop.id	31/08/2019

[Renew Hosting](#)

5. Pada halaman dashboard Vueshop, pilih file manager

The screenshot shows the Vueshop.id control panel. On the left, there's a sidebar with 'Domains' listed under 'Services'. Under 'Domains', there are sections for 'Billing' (0), 'Support' (0), 'Affiliate', and 'Promotion' (3). Below these is a large blue button with a white plus sign. The main area has tabs for 'Overview', 'Access', 'Upgrade', and 'Addons'. Under 'Overview', there's a 'Quick Shortcuts' section with icons for Email Accounts, Forwarders, Autoresponders, File Manager, Backup, Subdomains, Addon Domains, Cron Jobs, MySQL Databases, phpMyAdmin, and Awstats. To the right is a 'Usage Statistics' section showing Disk Usage (0/2048 M) and Bandwidth Usage (0 M / Unlimited M), last updated on 09/09/2018 (01:36).

6. Maka kita akan dibawah ke halaman file manager. Pilih public_html yaitu folder root web.

The screenshot shows the cPanel File Manager. The left sidebar shows a tree view of the directory structure under '/home/vueshopi', including 'etc', 'logs', 'mail', 'public_ftp', 'public_html', 'ssl', and 'tmp'. The main pane displays the contents of the 'public_html' directory. The table headers are 'Name', 'Size', 'Last Modified', and 'Type'. The contents are:

Name	Size	Last Modified	Type
etc	4 KB	Today, 9:54 AM	httpd/unix-directory
logs	4 KB	Sep 6, 2018, 5:10 PM	httpd/unix-directory
mail	4 KB	Sep 5, 2018, 4:44 PM	mail
public_ftp	4 KB	Sep 5, 2018, 4:44 PM	publicftp
public_html	4 KB	Sep 6, 2018, 5:40 AM	publichtml
ssl	4 KB	Today, 10:21 AM	httpd/unix-directory
tmp	4 KB	Sep 6, 2018, 5:10 PM	httpd/unix-directory
access-logs	34 bytes	Sep 5, 2018, 4:57 PM	httpd/unix-directory
www	11 bytes	Sep 5, 2018, 4:44 PM	publichtml

7. Pada folder public_html ini, pilih menu Upload untuk mengunggah file aplikasi web frontend

The screenshot shows the cPanel File Manager again. The left sidebar shows the same directory structure. The main pane now shows the contents of the 'public_html' directory after an upload. The table headers are 'Name', 'Size', 'Last Modified', and 'Type'. The new files are:

Name	Size	Last Modified	Type
cgi-bin	4 KB	Sep 5, 2018, 4:44 PM	httpd/unix-directory
404.shtml	3.85 KB	Jun 22, 2017, 7:43 PM	text/html
default.html	5.75 KB	Aug 30, 2018, 8:37 PM	text/html

8. Pada form upload, unggah file hasil kompres file-file yang ada dalam folder dist.

cp File Upload

Select the file you want to upload to "/home/vueshopi/public_html".

Maximum file size allowed for upload: 1.99 GB

Overwrite existing files

Drop files here to start uploading
or
Select File

[Go Back to "/home/vueshopi/public_html"](#)

9. Jika sudah selesai, klik Go Back.

cp File Upload

Select the file you want to upload to "/home/vueshopi/public_html".

Maximum file size allowed for upload: 1.99 GB

Overwrite existing files

Drop files here to start uploading
or
Select File

Archive.zip
100%
816.26 KB complete

[Go Back to "/home/vueshopi/public_html"](#)

10. Pada file manager, pilih file archive hasil unggahan kita, kemudian extract.

cp File Manager

Search All Your Files for Go Settings

+ File + Folder Copy Move Upload Download Delete Restore Rename Edit HTML Editor

Permissions View Extract Compress

public_html Go

Home Up One Level Back Forward Reload Select All Unselect All

View Trash Empty Trash

Name	Size	Last Modified	Type
cgi-bin	4 KB	Sep 5, 2018, 4:44 PM	httpd/unix-directory
404.shtml	3.85 KB	Jun 22, 2017, 7:43 PM	text/html
Archive.zip	816.26 KB	Today, 12:12 PM	package/x-generic
default.html	5.75 KB	Aug 30, 2018, 8:37 PM	text/html

11. Tentukan target extract yaitu di folder public_html

Files to extract:
/public_html/Archive.zip

Enter the path you wish to extract the files to (if you enter a directory that does not exist it will be created, and the archive extracted in the new directory) and click Extract:

/public_html

Extract File(s) Cancel

12. Jika selesai, klik Close.

Archive: /home/vueshopi/public_html/Archive.zip

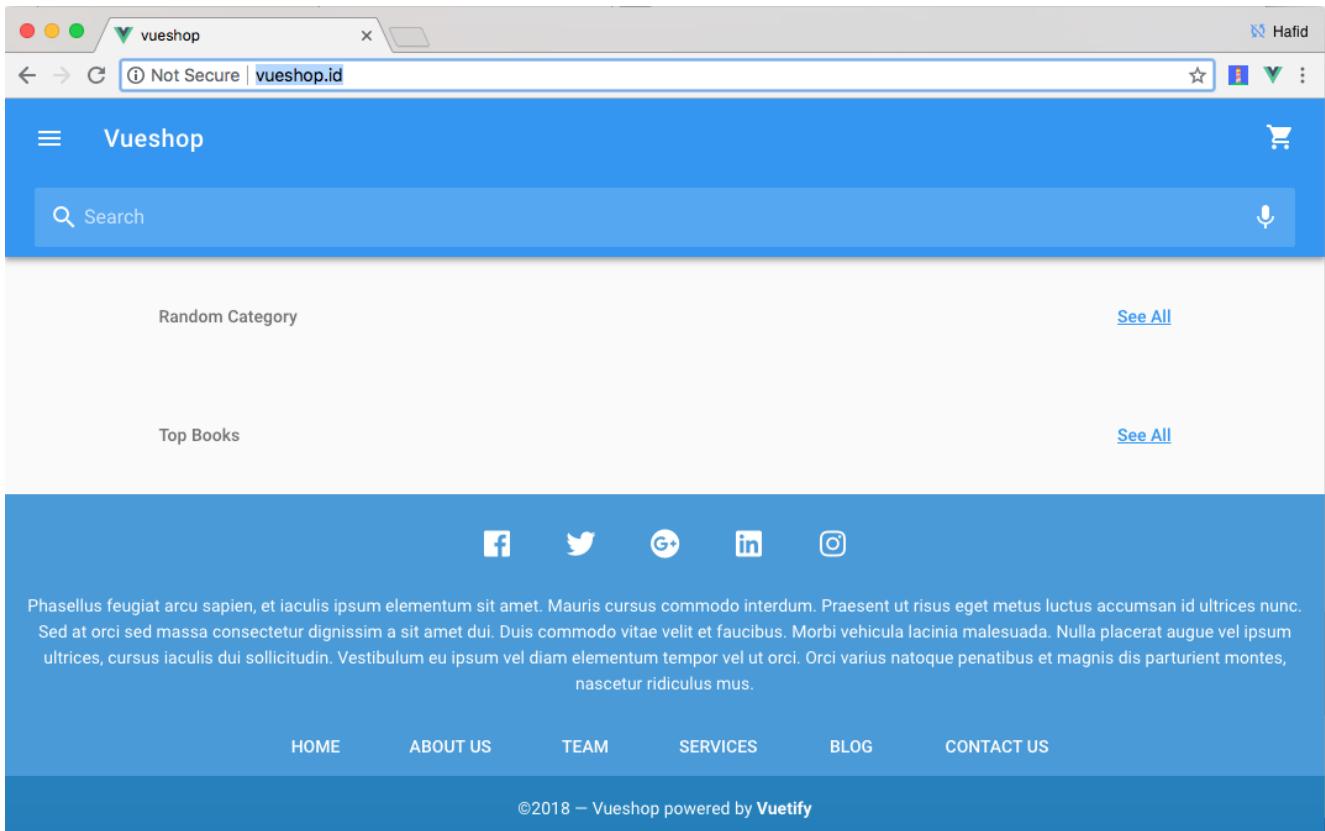
creating: css/
inflating: css/app.ae414c3b.css
inflating: css/categories.943639a1.css
inflating: css/category.7d8fc395.css
inflating: css/chunk-vendors.bd4880f.css
inflating: favicon.ico
creating: img/
inflating: img/bca.png
creating: img/icons/
inflating: img/icons/android-chrome-192x192.png
inflating: img/icons/android-chrome-512x512.png
inflating: img/icons/apple-touch-icon-120x120.png
inflating: img/icons/apple-touch-icon-152x152.png
inflating: img/icons/apple-touch-icon-180x180.png
inflating: img/icons/apple-touch-icon-60x60.png
inflating: img/icons/apple-touch-icon-76x76.png
inflating: img/icons/apple-touch-icon.png
inflating: img/icons/favicon-16x16.png

Close

13. Hapus file archive.

Name	Size	Last Modified	Type
cgi-bin	4 KB	Sep 5, 2018, 4:44 PM	httpd/unix-directory
css		day, 11:18 AM	httpd/unix-directory
img		day, 11:18 AM	httpd/unix-directory
js		day, 11:18 AM	httpd/unix-directory
404.shtml		Aug 22, 2017, 7:43 PM	text/html
Archive.zip		Aug 30, 2018, 12:12 PM	package/x-generic
default.html		Aug 30, 2018, 8:37 PM	text/html
favicon.ico		day, 11:18 AM	image/x-generic
index.html		day, 11:18 AM	text/html

14. Aplikasi telah siap diakses. Silakan diuji coba.



Mudah sekali bukan? Ya karena cuma static file. Adapun pada tampilan tersebut belum muncul data buku karena web servicenya belum kita deploy.

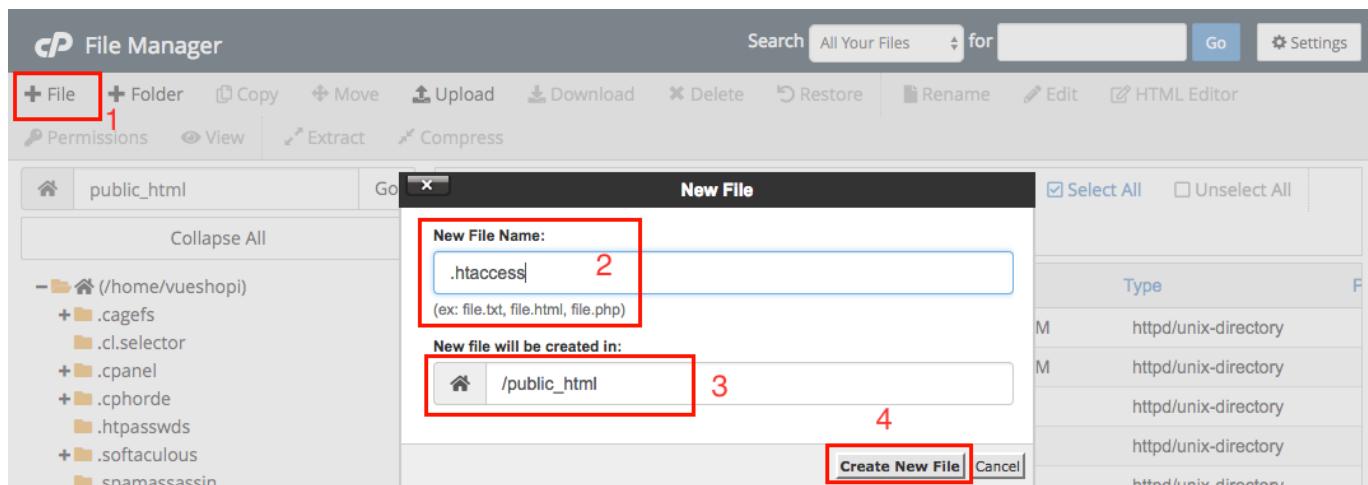
Oh Sebagai tambahan, karena domain yang kita gunakan mendukung SSL https maka kita bisa memaksa user yang mengakses ke vueshop.id agar menggunakan protocol https. Caranya gunakan file .htaccess (apache) berikut.

```

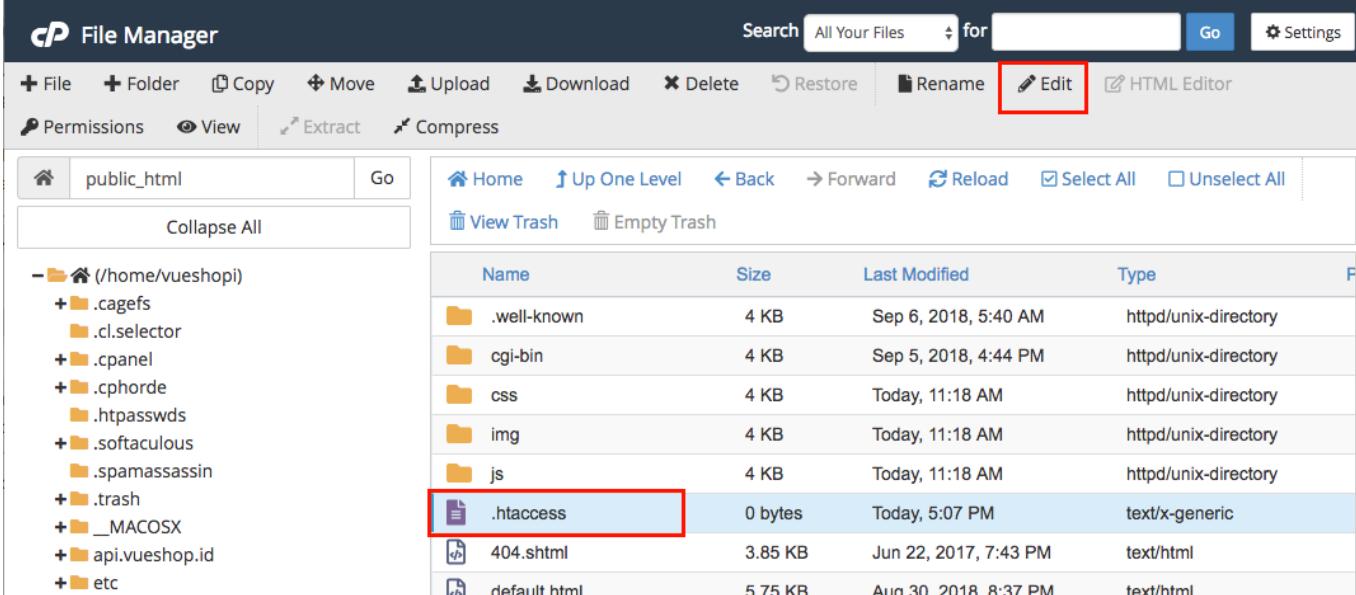
1 RewriteEngine On
2 RewriteCond %{HTTPS} !on
3 RewriteRule ^(.*)$ https:// %{HTTP_HOST}%{REQUEST_URI} [L,R=301]

```

Pada public_html, tambahkan file .htaccess



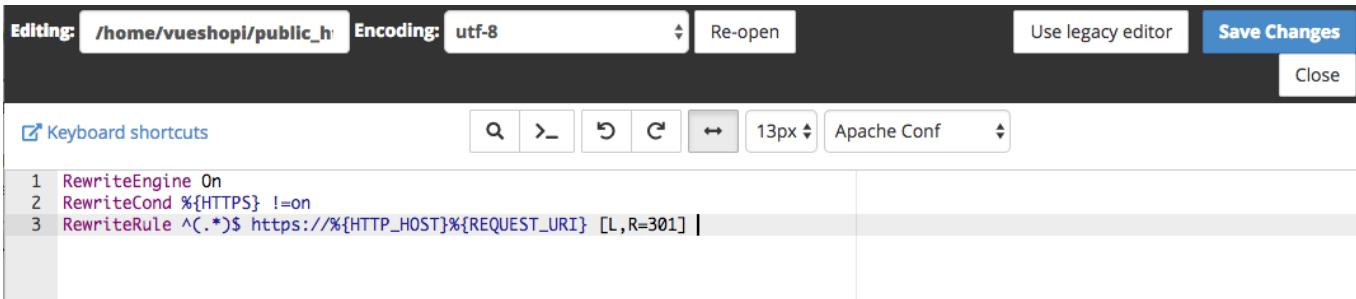
Edit file tersebut.



The screenshot shows the cPanel File Manager interface. On the left, there's a sidebar with various folder icons like .cagefs, .cl.selector, .cpanel, .cphorde, .htpasswd, .softaculous, .spamassassin, .trash, __MACOSX, api.vueshop.id, and etc. In the main area, there's a breadcrumb navigation bar with Home, Up One Level, Back, Forward, Reload, Select All, and Unselect All. Below it are buttons for View Trash and Empty Trash. A table lists files and folders in the current directory:

Name	Size	Last Modified	Type
.well-known	4 KB	Sep 6, 2018, 5:40 AM	httpd/unix-directory
cgi-bin	4 KB	Sep 5, 2018, 4:44 PM	httpd/unix-directory
css	4 KB	Today, 11:18 AM	httpd/unix-directory
img	4 KB	Today, 11:18 AM	httpd/unix-directory
js	4 KB	Today, 11:18 AM	httpd/unix-directory
.htaccess	0 bytes	Today, 5:07 PM	text/x-generic
404.shtml	3.85 KB	Jun 22, 2017, 7:43 PM	text/html
default.html	5.75 KB	Aug 30, 2018, 8:37 PM	text/html

Dengan kode htaccess di atas.



The screenshot shows the .htaccess editor interface. At the top, it says "Editing: /home/vueshopi/public_h" and "Encoding: utf-8". There are buttons for "Re-open", "Use legacy editor", and "Save Changes". Below that is a toolbar with "Keyboard shortcuts" and icons for search, refresh, and zoom. The main area contains the following code:

```

1 RewriteEngine On
2 RewriteCond %{HTTPS} !=on
3 RewriteRule ^(.*)$ https://{$HTTP_HOST}%{REQUEST_URI} [L,R=301]

```

Catatan: jika web server yang kamu gunakan nginx maka gunakan konfigurasi berikut.

```

1 server {
2     listen      80;
3     server_name vueshop.id;
4     rewrite     ^      https://$server_name$request_uri? permanent;
5 }

```

Deployment Aplikasi Web Service

Pada contoh ini, aplikasi web service akan kita deploy pada hosting yang sama dengan aplikasi web frontend namun kita tempatkan pada sebuah sub domain yaitu api.vueshop.id

Membuat Sub Domain

Pada contoh ini, kita akan menggunakan sub domain api.vueshop.id. Oleh karena itu, kita perlu membuatnya terlebih dahulu.

1. Pada dashboard vueshop, pilih Sub Domain.

The screenshot shows the DomaiNesia dashboard for the domain 'vueshop.id'. On the left sidebar, under 'Domains', there is a 'Subdomains' icon. In the main content area, there is a 'Quick Shortcuts' section with several icons: 'Email Accounts', 'Forwarders', 'Autoresponders', 'File Manager', 'Backup', 'Addon Domains', and 'Cron Jobs'. The 'Subdomains' icon is highlighted with a red box. To the right, there is a 'Usage Statistics' section showing 'Disk Usage' and 'Bandwidth Usage' with values of 0. The status bar at the bottom indicates 'Last Updated 09/09/2018 (01:36)'.

2. Pada form sub domain, masukkan api pada field sub domain, lalu klik tombol Create

The screenshot shows the 'Subdomains' creation form. The 'Subdomain' field contains the value 'api' and is highlighted with a red box. The 'Domain' dropdown is set to 'vueshop.id'. The 'Document Root' field shows '/api.vueshop.id'. At the bottom, there is a blue 'Create' button, which is also highlighted with a red box.

3. Setelah sub domain berhasil dibuat, klik link Go Back.

The screenshot shows a success message: 'Success: "api.vueshop.id" has been created.' This message is displayed in a green bar at the top of the page. Below it, there is a 'Go Back' button.

4. Balik ke cpanel melalui tombol di sisi kiri atas.

The screenshot shows the DomaiNesia cPanel interface. In the top left, there's a navigation menu with icons for Home, Domains, Email, Files, Databases, Security, and Help. A red box highlights the 'Files' icon. The main area is titled 'Modify a Subdomain'. It has tabs for 'Subdomains', 'Document Root', 'Redirection', and 'Actions'. Under 'Subdomains', there is one entry: 'api.vueshop.id' with 'Document Root' set to '/api.vueshop.id'. The 'Actions' column for this entry includes a 'Remove' button and a 'Manage Redirection' link. At the bottom, there are buttons for 'Page Size' (set to 10), navigation arrows (<<, <, >, >>), and a 'Go' button.

Ketika kita membuat sub domain api, maka sistem akan mengcreate folder baru yaitu `api.vueshop.id`. Pada folder inilah nantinya digunakan sebagai mount point aplikasi web service kita. Karena mount point Laravel ada di folder public maka artinya isi folder public seharusnya kita unggah pada folder `api.vueshop.id` di server.

Lalu di manakah kita letakkan folder selain public? Untuk keamanan, folder selain public seharusnya diletakkan diluar folder `api.vueshop.id` dan bahkan diluar `public_html`.

Membuat & Mengimport Database

Untuk membuat database, ikuti langkah-langkah berikut.

1. Pada control panel, scroll down ke panel Database. Pilih MySQL Database.

The screenshot shows the DomaiNesia cPanel interface. In the top left, there's a navigation menu with icons for Home, Domains, Email, Files, Databases, Security, and Help. A red box highlights the 'Databases' icon. The main area is titled 'Databases'. It features several links: 'phpMyAdmin' (with a database icon), 'MySQL® Databases' (with a database icon highlighted by a red box), 'MySQL® Database Wizard' (with a database icon), 'Remote MySQL®' (with a database icon), 'PostgreSQL Databases' (with a database icon), 'PostgreSQL Database Wizard' (with a database icon), and 'phpPgAdmin' (with a database icon). On the right side, there's a sidebar with metrics: 'PostgreSQL Disk Usage' (0 bytes / 1.99 GB (0%)), 'Bandwidth' (108.86 KB / ∞), 'Addon Domains' (0 / ∞), 'Subdomains' (1 / ∞), and 'Aliases' (0 / ∞).

2. Pada MySQL Database, masukkan nama databasenya yaitu `larashop` yang secara default akan ada tambahan prefix bawaan hosting. Lalu klik tombol Create Database.

DomaiNesia MySQL® Databases

Manage large amounts of information over the web easily. MySQL databases are necessary to run many web-based applications, such as bulletin boards, content management systems, and online shopping carts. For more information, read the [documentation](#).

↓ Jump to MySQL Users

Create New Database

New Database:

vueshopi_ larashop

Create Database

3. Jika berhasil maka klik link Go Back.

DomaiNesia MySQL® Databases

Added the database "vueshopi_larashop".

[Go Back](#)

4. Scrolldown, buat User database, masukkan username dan password. Lalu klik create user.

Add New User

Username: vueshopi_ app

Password:

Password (Again):

Strength: Very Strong (97/100)

Create User

5.Jika berhasil, maka klik link Go Back.

DomaiNesia MySQL® Databases

You have successfully created a MySQL user named "vueshopi_app".

[Go Back](#)

6. Scroll down, tambahkan user yang telah kita buat pada database, supaya user tersebut dapat berinteraksi dengan database. lalu klik add.

Add User To Database

User: vueshopi_app

Database: vueshopi_larashop

Add

7. Tentukan privilege atau hak akses user tersebut pada database. Sebaiknya kamu punya minimal dua jenis user, user admin bisa semua dan user aplikasi yang hanya bisa select, insert, update dan delete saja.

MySQL® Databases

Manage User Privileges

User: vueshopi_app
Database: vueshopi_larashop

ALL PRIVILEGES

<input checked="" type="checkbox"/> ALTER	<input checked="" type="checkbox"/> ALTER ROUTINE
<input checked="" type="checkbox"/> CREATE	<input checked="" type="checkbox"/> CREATE ROUTINE
<input checked="" type="checkbox"/> CREATE TEMPORARY TABLES	<input checked="" type="checkbox"/> CREATE VIEW
<input checked="" type="checkbox"/> DELETE	<input checked="" type="checkbox"/> DROP

<input checked="" type="checkbox"/> EVENT	<input checked="" type="checkbox"/> EXECUTE
<input checked="" type="checkbox"/> INDEX	<input checked="" type="checkbox"/> INSERT
<input checked="" type="checkbox"/> LOCK TABLES	<input checked="" type="checkbox"/> REFERENCES
<input checked="" type="checkbox"/> SELECT	<input checked="" type="checkbox"/> SHOW VIEW
<input checked="" type="checkbox"/> TRIGGER	<input checked="" type="checkbox"/> UPDATE

Make Changes

8. Scroll down dan klik Make Changes

Make Changes

Langkah selanjutnya adalah mengimport database lokal ke server, tentu sebelumnya kamu perlu mengeksport database larashop dilokalmu. Kamu bisa gunakan tools phpmyadmin untuk export database larashopmu.

The screenshot shows the phpMyAdmin interface for the 'larashop' database. A red box labeled '1' highlights the 'larashop' database in the left sidebar. A red box labeled '2' highlights the 'Export' tab in the top menu bar. A red box labeled '3' highlights the 'Go' button in the 'Format' section. A red box labeled '4' highlights the download link for 'larashop.sql' in the bottom left corner.

Export method:

- Quick - display only the minimal options
- Custom - display all possible options

Format:

SQL

Go

larashop.sql

Setelah database lokal berhasil kita export, kemudian akan kita unggah atau import ke server. Berikut ini langkahnya.

1. Dari control panel, scroll down ke database, pilih phpmyadmin

The screenshot shows the DomaiNesia control panel. In the 'DATABASES' section, the 'phpMyAdmin' icon is highlighted with a red box. Other options include 'MySQL® Database Wizard', 'PostgreSQL Databases', and 'phpPgAdmin'. On the right side, there are statistics for PostgreSQL Disk Usage, Bandwidth, Addon Domains, Subdomains, and Aliases.

DATABASES

phpMyAdmin

MySQL® Database Wizard

PostgreSQL Databases

phpPgAdmin

PostgreSQL Disk Usage
0 bytes / 1.99 GB (0%)

Bandwidth
108.86 KB / ∞

Addon Domains
0 / ∞

Subdomains
1 / ∞

Aliases
0 / ∞

2. Pada panel phpmyadmin, pilih database larashop.

The screenshot shows the phpMyAdmin interface with the title bar "Server: localhost:3306". The left sidebar lists databases: "information_schema" and "vueshopi_larashop". The main area has two tabs: "General settings" and "Appearance settings". Under "General settings", the "Server connection collation" is set to "utf8mb4_unicode_ci". Under "Appearance settings", the "Language" is "English", "Theme" is "pmahomme", and "Font size" is "82%". To the right, there are two panels: "Database server" listing MySQL server details like version 5.7.23-log and user cpses_vuqfavy7cz@localhost; and "Web server" listing PHP and MySQL client details like cprsvd 11.74.0.6 and PHP 5.6.30.

3. Pada panel database larashop, pilih import

The screenshot shows the phpMyAdmin interface with the title bar "Server: localhost:3306 » Database: vueshopi_larashop". The left sidebar shows the database structure. The main area has tabs: "Structure", "SQL", "Search", "Query", "Export", "Import", "Operations", and "More". A message "No tables found in database." is displayed. Below it, a "Create table" dialog is open, showing fields for "Name" and "Number of columns: 4".

4. Pada panel import, pilih file sql hasil backup database larashop yang telah kita buat untuk diimport

The screenshot shows the phpMyAdmin interface with the title bar "Server: localhost:3306 » Database: vueshopi_larashop". The left sidebar shows the database structure. The main area has tabs: "Structure", "SQL", "Search", "Query", "Export", "Import", "Operations", and "More". A message "Importing into the database \"vueshopi_larashop\" is displayed. Below it, a "File to import:" section allows users to browse their computer for an SQL file to upload. The character set is set to "utf-8".

5. Scroll down dan klik tombol Go

6. Tunggu sampai semua file sql berhasil diimport.

Selesai.

Berdasarkan pengaturan di atas, maka kita bisa mengupdate file .env pada aplikasi web service kita menjadi sebagai berikut

```

1 DB_CONNECTION=mysql
2 DB_HOST=localhost
3 DB_PORT=3306
4 DB_DATABASE=vueshopi_larashop
5 DB_USERNAME=vueshopi_app
6 DB_PASSWORD=PASSWORD_DATABASEMU

```

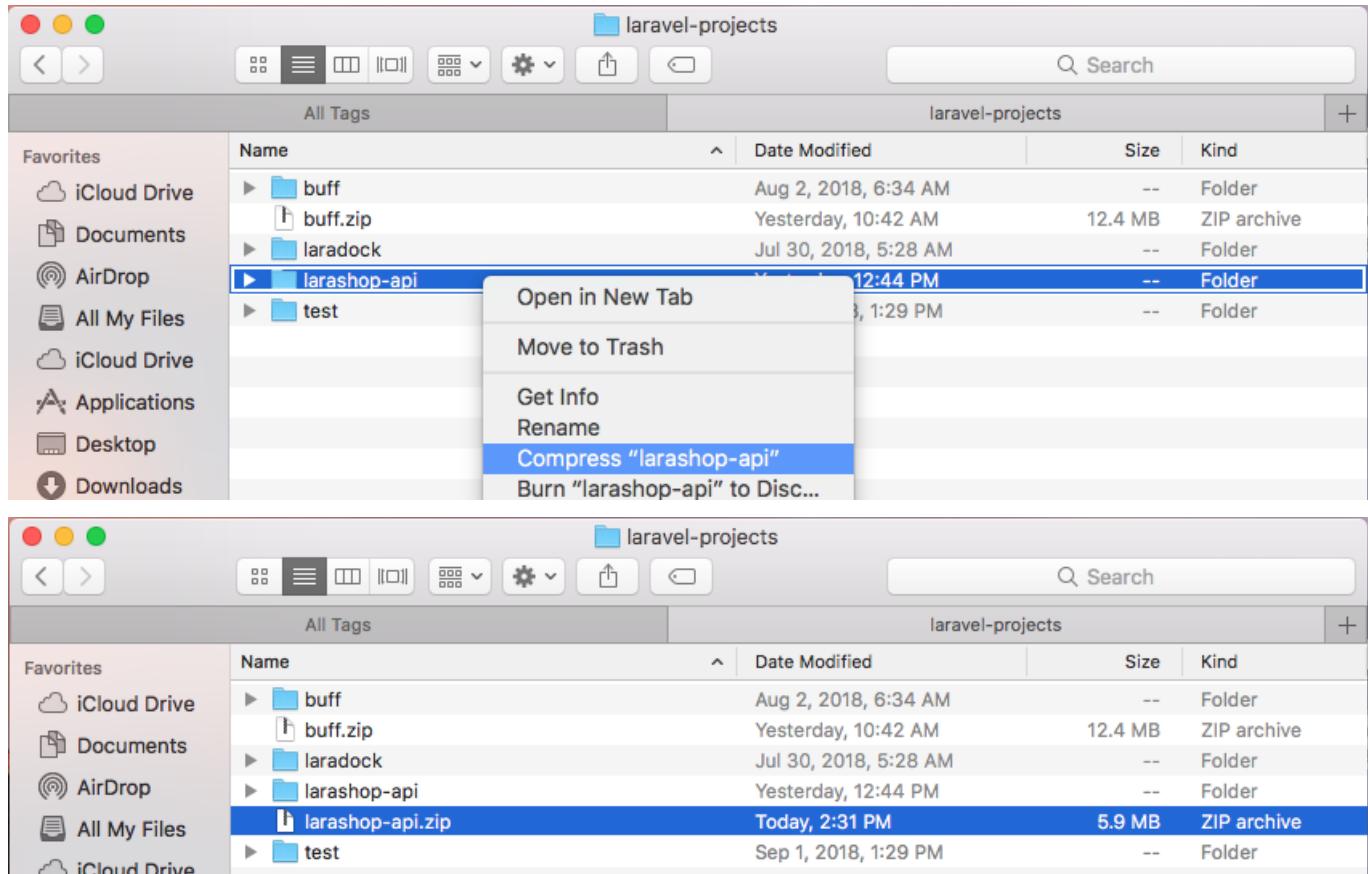
Mengunggah File Aplikasi

Setelah database siap, maka aplikasi web service kita siap di deploy.

Jalankan perintah berikut pada.

```
1 php artisan cache:clear
2 composer install --optimize-autoloader --no-dev
3 php artisan route:cache
```

Kompres folder larashop-api hal ini untuk memudahkan kita saat mengunggahnya ke server.



Berikut ini langkah demi langkah untuk mengunggah file aplikasi ke server.

1. Pada control panel, pilih file manager

2. Pada file manager pilih menu Upload

The screenshot shows the cPanel File Manager interface. At the top, there's a navigation bar with 'File Manager' and various file operations like 'Upload', 'Download', 'Delete', 'Rename', 'Edit', and 'HTML Editor'. Below the navigation bar is a toolbar with icons for 'Permissions', 'View', 'Extract', and 'Compress'. The main area has a breadcrumb path 'Home > /home/vueshopi'. On the left, there's a sidebar with a tree view of the directory structure under '/home/vueshopi', including 'api.vueshop.id', 'etc', 'logs', 'mail', 'public_ftp', 'public_html', 'ssl', and 'tmp'. On the right, there's a table listing files with columns for Name, Size, Last Modified, and Type. A red box highlights the 'Upload' button in the top navigation bar.

3. Pada form upload, klik tombol Select untuk memilih file kode aplikasi web service yang telah dikompres

The screenshot shows a 'File Upload' form. It has a message 'Select the file you want to upload to "/home/vueshopi".' Below it is a note 'Maximum file size allowed for upload: 1.99 GB'. There's a checkbox 'Overwrite existing files' and a large dashed box for dragging files. Inside the box, it says 'Drop files here to start uploading' and 'or' below it is a 'Select File' button.

4. Tunggu sampai selesai dan klik Go Back.

The screenshot shows a 'File Upload' form again. It has a message 'Select the file you want to upload to "/home/vueshopi".' Below it is a note 'Maximum file size allowed for upload: 1.99 GB'. There's a checkbox 'Overwrite existing files' and a large dashed box for dragging files. Inside the box, it says 'Drop files here to start uploading' and 'or' below it is a 'Select File' button. Below the box, a progress bar shows 'larashop-api.zip' at 100% completion with the message '5.64 MB / 5.64 MB (100%) complete'.

[Go Back to "/home/vueshopi"](#)

5. Extract

file

zip

larashop-api.zip.

The screenshot shows the cPanel File Manager interface. At the top, there are various file operations: File, Folder, Copy, Move, Upload, Download, Delete, Restore, Rename, Edit, and HTML Editor. Below these are buttons for Permissions, View, Extract (which is highlighted with a red box), Compress, Home, Up One Level, Back, Forward, Reload, Select All, and Unselect All. Underneath is a 'View Trash' and 'Empty Trash' section. The main area displays a list of files and directories under the path '/home/vueshopi'. A file named 'larashop-api.zip' is selected and highlighted with a red box. The list includes:

Name	Size	Last Modified	Type
api.vueshop.id	4 KB	Today, 1:03 PM	httpd/unix-directory
etc	4 KB	Today, 9:54 AM	httpd/unix-directory
logs	4 KB	Sep 6, 2018, 5:10 PM	httpd/unix-directory
mail	4 KB	Sep 5, 2018, 4:44 PM	mail
public_ftp	4 KB	Sep 5, 2018, 4:44 PM	publicftp
public_html	4 KB	Today, 12:14 PM	publichtml
ssl	4 KB	Today, 1:09 PM	httpd/unix-directory
tmp	4 KB	Today, 2:39 PM	httpd/unix-directory
access-logs	34 bytes	Sep 5, 2018, 4:57 PM	httpd/unix-directory
larashop-api.zip	5.64 MB	Today, 2:39 PM	package/x-generic
www	11 bytes	Sep 5, 2018, 4:44 PM	publichtml

6. Kosongkan

target

extract

dan

klik

tombol

Extract

...

The screenshot shows the 'Extract' dialog box overlaid on the cPanel File Manager. The dialog has a title bar 'Extract' and a message 'Files to extract: /larashop-api.zip'. It contains a text input field with a placeholder 'Enter the path you wish to extract the files to (if you enter a directory that does not exist it will be created, and the archive extracted in the new directory) and click Extract.' Below the input field is a dropdown menu with the current path '/'. At the bottom right are 'Extract File(s)' and 'Cancel' buttons. In the background, the file list from the previous screenshot is visible, showing the same files and their details.

7. Klik tombol close setelah proses extract selesai

Extraction Results

```

Archive: /home/vueshop/larashop-api.zip
creating: larashop-api/
inflating: larashop-api/.DS_Store
creating: __MACOSX/
inflating: __MACOSX/larashop-api/
inflating: larashop-api/.editorconfig
inflating: larashop-api/.env
inflating: larashop-api/.env.example
inflating: larashop-api/.gitattributes
inflating: larashop-api/.gitignore
creating: larashop-api/app/
inflating: larashop-api/app/.DS_Store
creating: __MACOSX/larashop-api/app/
inflating: __MACOSX/larashop-api/app/._.DS_Store
inflating: larashop-api/app/Book.php
inflating: larashop-api/app/BookCategory.php
inflating: larashop-api/app/BookOrder.php
inflating: larashop-api/app/Category.php

```

File Types

- httpd/unix-directory
- httpd/unix-directory
- httpd/unix-directory
- httpd/unix-directory
- httpd/unix-directory
- mail
- publicftp
- publichtml
- httpd/unix-directory

8. Posisi folder larashop-api hasil extract, jangan lupa hapus file zip larashop-api.zip

View Trash

Name	Size	Last Modified	Type
api.vueshop.id	4 KB	Today, 1:03 PM	httpd/unix-directory
etc	4 KB	Today, 9:54 AM	httpd/unix-directory
larashop-api	4 KB	Yesterday, 12:44 PM	httpd/unix-directory
logs	4 KB	Sep 6, 2018, 5:10 PM	httpd/unix-directory
mail	4 KB	Sep 5, 2018, 4:44 PM	mail
public_ftp	4 KB	Sep 5, 2018, 4:44 PM	publicftp

9. Pada folder larashop-api, klik folder public

View Trash

Name	Size	Last Modified	Type
css	4 KB	Yesterday, 10:48 AM	httpd/unix-directory
images	4 KB	Yesterday, 1:06 PM	httpd/unix-directory
js	4 KB	Yesterday, 10:48 AM	httpd/unix-directory
svg	4 KB	Yesterday, 10:48 AM	httpd/unix-directory
favicon.ico	0 bytes	Yesterday, 10:48 AM	image/x-generic
index.php	1.78 KB	Yesterday, 10:48 AM	application/x-httpd-php
robots.txt	24 bytes	Yesterday, 10:48 AM	text/plain
web.config	914 bytes	Yesterday, 10:48 AM	text/x-generic

10. Copy semua file dalam folder public tersebut.

Name	Size	Last Modified	Type
css	4 KB	Yesterday, 10:48 AM	httpd/unix-directory
images	4 KB	Yesterday, 1:06 PM	httpd/unix-directory
js	4 KB	Yesterday, 10:48 AM	httpd/unix-directory
svg	4 KB	Yesterday, 10:48 AM	httpd/unix-directory
favicon.ico	0 bytes	Yesterday, 10:48 AM	image/x-generic
index.php	1.78 KB	Yesterday, 10:48 AM	application/x-httpd-php
robots.txt	24 bytes	Yesterday, 10:48 AM	text/plain
web.config	914 bytes	Yesterday, 10:48 AM	text/x-generic

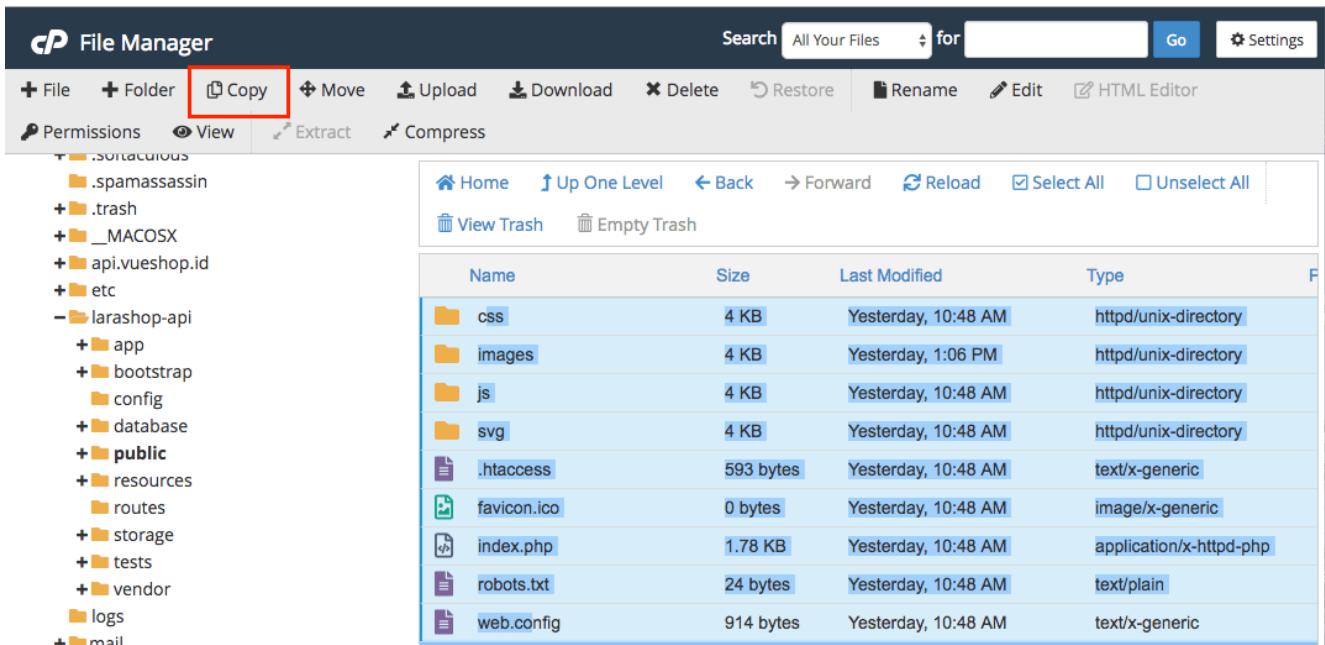
Jangan sampai ada yang ketinggalan, termasuk file .htaccess (karena web server yang kita gunakan untuk production adalah Apache). Pada cpanel umumnya secara default file ini akan disembunyikan untuk menampilkannya caranya.

Klik tombol setting pada bagian atas kanan file manager.

Kemudian pada popup yang muncul, checklist show hidden files

Type	
0 AM	httpd/unix-directory
4 PM	httpd/unix-directory
	httpd/unix-directory
	httpd/unix-directory
	text/x-generic
43 PM	text/html
27 DM	text/html

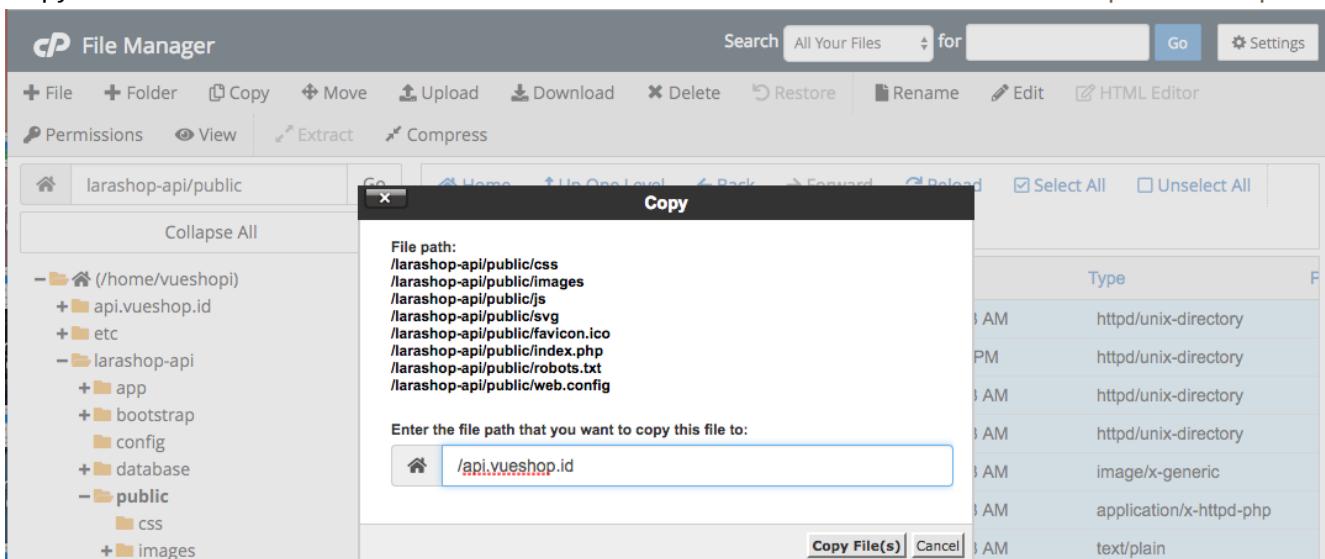
11. Copy semua file dalam folder public tersebut termasuk file .htaccess.



The screenshot shows the cPanel File Manager interface. On the left, a tree view shows the directory structure under larashop-api/public. On the right, a list view shows files and folders with their names, sizes, last modified dates, and types. A 'Copy' button is highlighted with a red box in the top navigation bar.

Name	Size	Last Modified	Type
css	4 KB	Yesterday, 10:48 AM	httpd/unix-directory
images	4 KB	Yesterday, 1:06 PM	httpd/unix-directory
js	4 KB	Yesterday, 10:48 AM	httpd/unix-directory
svg	4 KB	Yesterday, 10:48 AM	httpd/unix-directory
.htaccess	593 bytes	Yesterday, 10:48 AM	text/x-generic
favicon.ico	0 bytes	Yesterday, 10:48 AM	image/x-generic
index.php	1.78 KB	Yesterday, 10:48 AM	application/x-httpd-php
robots.txt	24 bytes	Yesterday, 10:48 AM	text/plain
web.config	914 bytes	Yesterday, 10:48 AM	text/x-generic

12. Copy semua file tersebut ke dalam folder api.vueshop.id



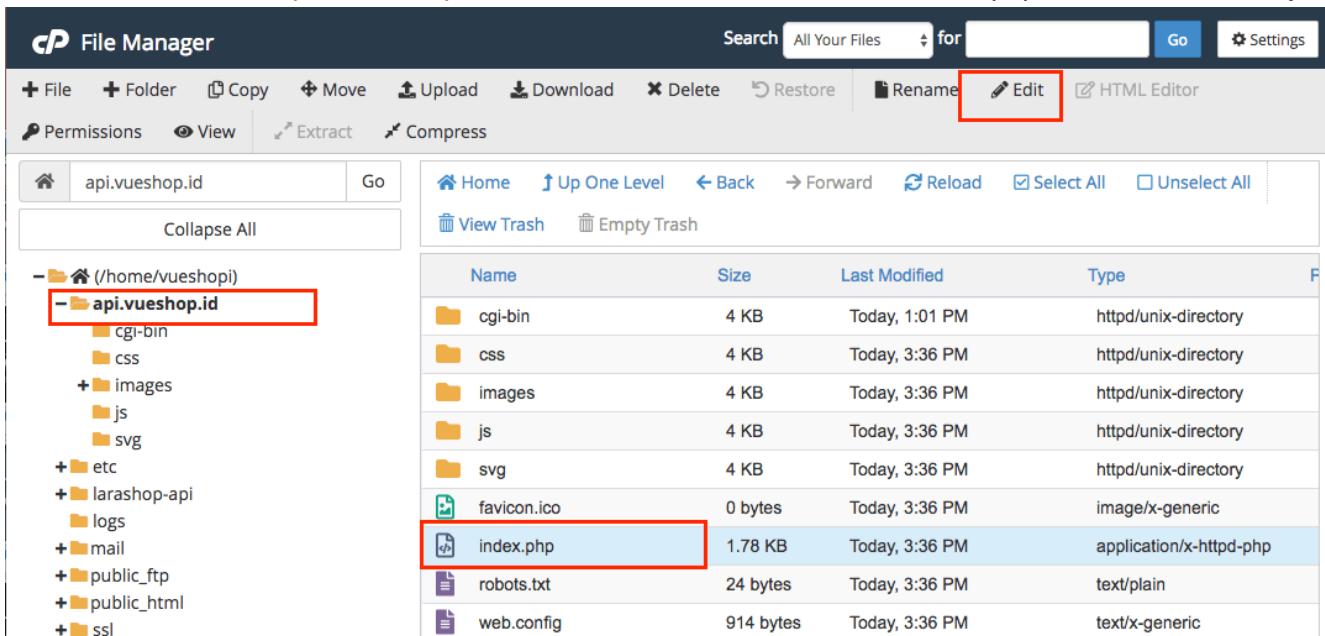
The screenshot shows the cPanel File Manager interface with a 'Copy' dialog box open. The left sidebar shows the directory structure under larashop-api/public. The right pane shows the contents of the public folder. A 'Copy' button is highlighted with a red box in the dialog box.

Copy

File path:
 /larashop-api/public/css
 /larashop-api/public/images
 /larashop-api/public/js
 /larashop-api/public/svg
 /larashop-api/public/favicon.ico
 /larashop-api/public/index.php
 /larashop-api/public/robots.txt
 /larashop-api/public/web.config

Enter the file path that you want to copy this file to:
 /api.vueshop.id

13. Buka folder api.vueshop.id, lalu edit file index.php di dalamnya.



The screenshot shows the cPanel File Manager interface. The left sidebar shows the directory structure under api.vueshop.id. The right pane shows the contents of the api.vueshop.id folder. An 'Edit' button is highlighted with a red box in the top navigation bar.

Name	Size	Last Modified	Type
cgi-bin	4 KB	Today, 1:01 PM	httpd/unix-directory
css	4 KB	Today, 3:36 PM	httpd/unix-directory
images	4 KB	Today, 3:36 PM	httpd/unix-directory
js	4 KB	Today, 3:36 PM	httpd/unix-directory
svg	4 KB	Today, 3:36 PM	httpd/unix-directory
favicon.ico	0 bytes	Today, 3:36 PM	image/x-generic
index.php	1.78 KB	Today, 3:36 PM	application/x-httpd-php
robots.txt	24 bytes	Today, 3:36 PM	text/plain
web.config	914 bytes	Today, 3:36 PM	text/x-generic

14. Buka folder `api.vueshop.id`, lalu edit file `index.php` di dalamnya.

Semula

```
1 require __DIR__.'/../vendor/autoload.php';
2 $app = require_once __DIR__.'/../bootstrap/app.php';
```

Ubah menjadi

```
1 require __DIR__.'/../larashop-api/vendor/autoload.php';
2 $app = require_once __DIR__.'/../larashop-api/bootstrap/app.php';
```

The screenshot shows a code editor interface with the following details:

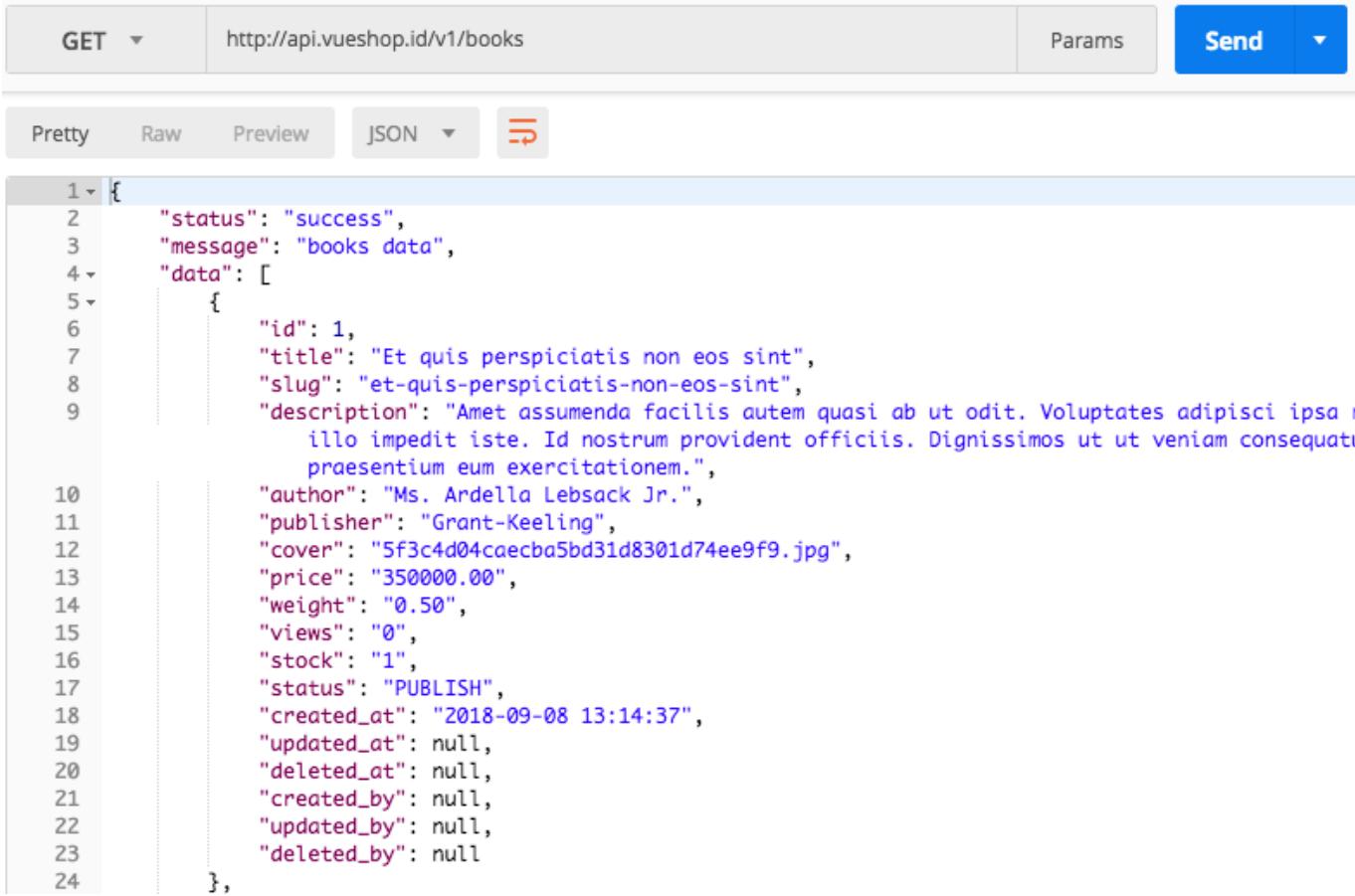
- Top bar: "Editing: /home/vueshop/api.vues" (highlighted), "Encoding: utf-8", "Re-open", "Use legacy editor", "Save Changes" (highlighted blue), and "Close".
- Toolbar: "Keyboard shortcuts", search icon, back icon, forward icon, refresh icon, "13px", "PHP", and a dropdown menu.
- Code area:

```
23
24 require __DIR__.'/../larashop-api/vendor/autoload.php';
25
26 /*
27 |-----
28 | Turn On The Lights
29 |-----
30 |
31 | We need to illuminate PHP development, so let us turn on the lights.
32 | This bootstraps the framework and gets it ready for use, then it
33 | will load up this application so that we can run it and send
34 | the responses back to the browser and delight our users.
35 |
36 */
37
38 $app = require_once __DIR__.'/../larashop-api/bootstrap/app.php';
39
40 */
```

The lines `require __DIR__.'/../larashop-api/vendor/autoload.php';` and `$app = require_once __DIR__.'/../larashop-api/bootstrap/app.php';` are highlighted with red boxes.

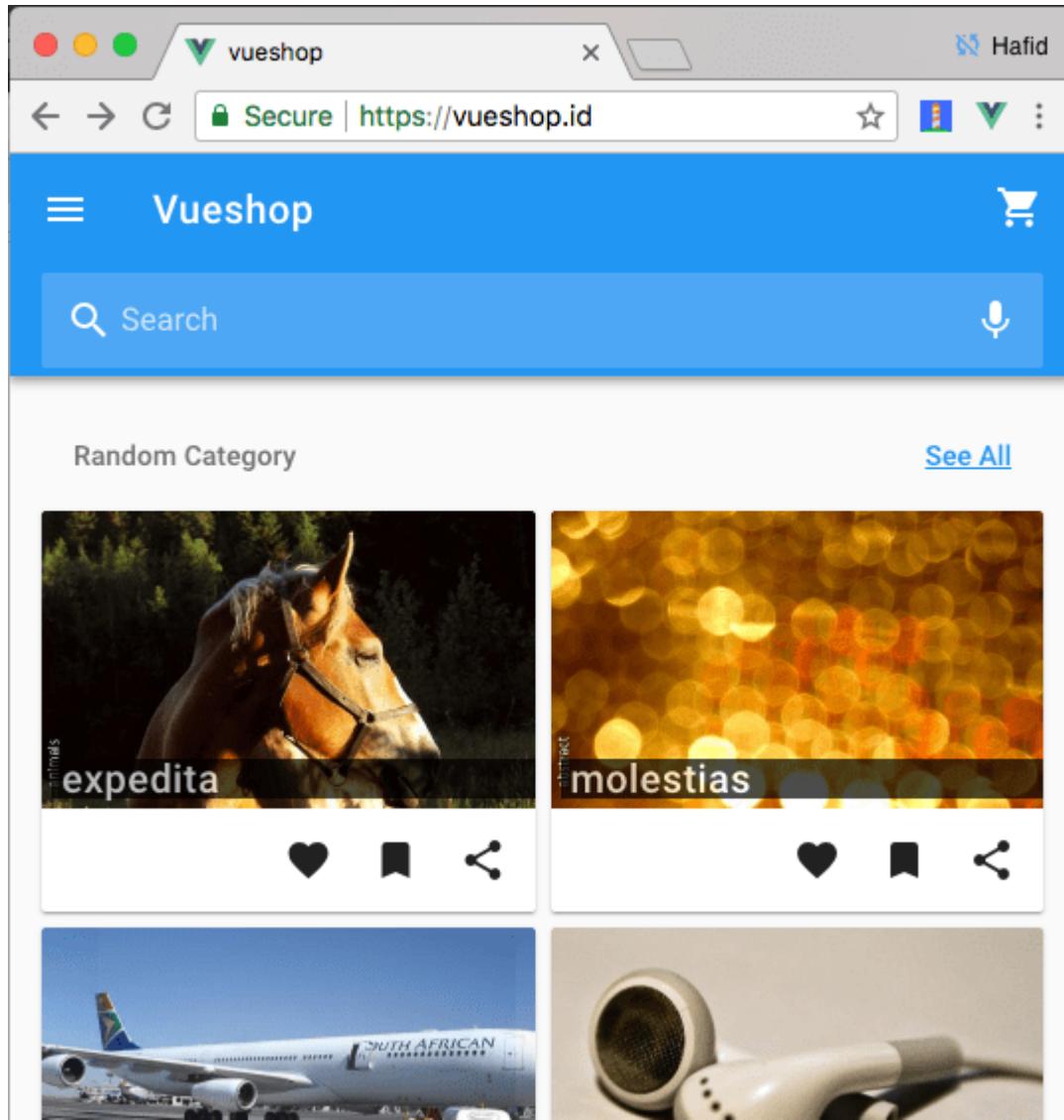
Intinya arahkan ke folder `larashop-api`.

Lakukan uji coba untuk mengakses salah satu endpoint yang sifatnya publik misalnya:
`http://api.vueshop.id/v1/books`



```
1 {
2     "status": "success",
3     "message": "books data",
4     "data": [
5         {
6             "id": 1,
7             "title": "Et quis perspiciatis non eos sint",
8             "slug": "et-quis-perspiciatis-non-eos-sint",
9             "description": "Amet assumenda facilis autem quasi ab ut odit. Voluptates adipisci ipsa illo impedit iste. Id nostrum provident officiis. Dignissimos ut ut veniam consequatur praesentium eum exercitationem.",
10            "author": "Ms. Ardella Lebsack Jr.",
11            "publisher": "Grant-Keeling",
12            "cover": "5f3c4d04caecba5bd31d8301d74ee9f9.jpg",
13            "price": "350000.00",
14            "weight": "0.50",
15            "views": "0",
16            "stock": "1",
17            "status": "PUBLISH",
18            "created_at": "2018-09-08 13:14:37",
19            "updated_at": null,
20            "deleted_at": null,
21            "created_by": null,
22            "updated_by": null,
23            "deleted_by": null
24        }
      ]
```

Sekarang kita coba untuk aplikasi web frontendnya melalui alamat <https://vueshop.id>



Yeay berhasil.

Kesimpulan

Deployment merupakan aktivitas yang dilakukan untuk mengunggah aplikasi ke server serta mengatur konfigurasinya sehingga aplikasi dapat diakses oleh publik. Sebelum melakukan proses deployment, pastikan aplikasi kita dalam mode production hal ini untuk alasan keamanan dan performa. Sesuaikan juga konfigurasinya karena bisa jadi berbeda dengan konfigurasi di server production.

Pada bab ini dijelaskan tentang langkah-langkah melakukan deploy ke share hosting dengan cara yang terbilang masih standard. Kita bisa juga menggunakan git dan atau composer yang umumnya juga telah didukung oleh share hosting provider termasuk dalam hal ini Domainesia. Untuk deployment ke virtual private server tentu cara dan tekniknya lebih beragam lagi. Silakan bereksplorasi lebih dalam terkait hal ini.

Luar biasa kamu! Sudah pantas menyandang gelar sebagai fullstack developer.



// HOSTING //

Incredibly fast and reliable hosting infrastructure with custom built features to accelerate your website. Ranked as No. 1 web hosting provider in Indonesia*.

PT Deneva

YAP Square No. C-5,
Jl C. Simanjuntak No.2 Yogyakarta 55223,
Daerah Istimewa Yogyakarta, Indonesia
info@domainesia.com | (0274) 545653
www.domainesia.com

*According to survey by thehosting.com as of September 2018

Khusus pembaca buku ini, kami menyediakan diskon khusus sebesar 30% dari harga reguler untuk pembelian produk Hosting di <https://domainesia.com>, dengan menggunakan kode kupon **FullstackHosting**

Catatan: masukkan kode kupon di atas setiap transaksi. Kode ini dapat digunakan secara berulang.



// VPS //

Instantly spin Linux virtual server instance
and get root access within seconds.
Powered by enterprise-grade server with
Native SSD storage to deliver
blazing fast performance.



PT Deneva

YAP Square No. C-5,
Jl C. Simanjuntak No.2 Yogyakarta 55223,
Daerah Istimewa Yogyakarta, Indonesia
info@domainesia.com | (0274) 545653
[www.domainesia.com](http://domainesia.com)

Khusus pembaca buku ini, kami menyediakan diskon khusus sebesar 30% dari harga reguler untuk pembelian produk VPS di <https://domainesia.com>, dengan menggunakan kode kupon **FullstackVPS**

Catatan: masukkan kode kupon di atas setiap transaksi. Kode ini dapat digunakan secara berulang.