



Hafid Mukhasin
Muhammad Azamuddin

2.5

Vue.js

The Progressive JavaScript Framework

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

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. Amiiin.

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	15
Bukan One Man Show	15
Digunakan Perusahaan Besar	15
Mudah Dipelajari	15
Mudah Diintegrasikan dengan Pustaka Lain	16
Dukungan Official untuk Pengembangan Aplikasi Enterprise	16
Fitur Utama	16
Virtual DOM	16
Component Base	17
Template	17
Modularity	17
Reactivity	17
Routing	17
State Management	17
Development Tools	18
Instalasi & Konfigurasi	18
Hello World	19
Menguji Reaktifitas	23
Kesimpulan	24
Dasar-Dasar Vue	25
Intro	25
Objek Vue	25
Inisiasi Objek Vue	25
Properti el	26
Properti Data	26
Siklus Objek Vue	28
create	30
mount	33
update	34
destroy	36
Penulisan Template	37
Data Teks	37
Data Raw HTML	38
Data Attribute	38
JavaScript Expression	39
Properti Template	40
Properti Methods, Computed, & Filters	42
Properti Methods	42
Properti Computed	42

Properti Filters	44
Argumen Pada Filters	46
Chaining Filters	47
Deklarasi Filters Secara Terpisah	48
Kesimpulan	48
Directive	50
Intro	50
Mengenal Directive	50
v-html	50
v-once	50
v-text	50
v-show	50
v-if	51
v-on	53
v-bind	60
Kesimpulan	62
List	63
Intro	63
Menampilkan Data Array	63
v-for Menggunakan Tag Template	64
v-for Menggunakan Index	64
Menampilkan Data Objek	65
Menampilkan Data Collection	67
Atribut Key	69
Membatasi v-for menggunakan v-if	70
Perubahan (mutation) Data Pada Array	71
push() & pop()	72
unshift() & shift()	72
sort() & reverse()	73
splice()	73
fungsi set pada Vue	76
Perubahan Data Pada Objek	77
Kesimpulan	78
Form	80
Intro	80
Input Binding	80
Text	82
Boolean	84
Array	85
Filtering Data List	87
Handling Submit Form & Validation	88
Validasi Data	91
Prepare Data Submit	96
Send Data To Server	97
Handling File Upload	101
Kesimpulan	103
Component	105
Intro	105
Component Dasar	105
Component Naming	106
Component Registration	106
Global Component	107
Local Component	108
Deklarasi Properti Data	110

Reusable Component	110
Component Lanjutan	111
Passing Data To Component	111
Directive Pada Component	113
Update Data Parent From Component	116
Two Way Data Binding on Component	117
Content Distribution with Slots	118
Single File Component	119
Dynamic Components	122
Transition Effect	125
Mixins	125
Plugins	127
Deklarasi Plugins	127
Menggunakan Plugin	128
Kesimpulan	128
Routing	129
Intro	129
Features	129
Installation	129
Getting Started	129
Dynamic Routing	132
Component BooksComponent	132
Component BookComponent	134
Programmatic Navigation	137
Penamaan Routes	139
Passing Props To Route Component	140
Transitions Effect	140
Navigation Guards	141
Global	142
Per Route	142
Dalam Component	142
Prevent Leave Accident	143
Authentication Route	144
Kesimpulan	145
Intro	146
Mengenal State Management	146
Pustaka State Management	147
Instalasi	148
Dev Tools	149
Getting Started	151
Mengakses Store Via Component	155
Getters	158
Mutations	159
Actions	160
Asynchronous Actions	162
Menangani Two Way Data Binding	166
Kesimpulan	170
Scaffolding Application	171
Intro	171
Briefing Projek	171
Fitur Utama Aplikasi	171
Unified Model Language	172
Use Case Diagram	172
Activity Diagram	173

Class Diagram	173
Desain Database	174
Preparing Project	175
Command Line Tools	175
Package Manager for Javascript	176
Instalasi NodeJS & NPM	176
Instalasi Vue melalui NPM	179
Apa itu Bundler	180
Instalasi Vue menggunakan Javascript Bundler	181
Browserify	181
Webpack	184
Single File Component	189
Web Server Development	193
Hot Reload	194
Vue Command Line Interface (CLI)	197
Create New Project	197
Create New Project on Web Base	203
Menambahkan Plugin Baru	214
Kesimpulan	226
Web Service	227
Intro	227
Mengenal Web Service	227
Definisi Web Service	227
Standard Web Service	227
Method Web Service	227
Cara Kerja Web Service	227
HTTP Response Code	228
Stateles pada Web Service	229
Persiapan Tools Pengembangan	229
Bahasa Pemrograman: PHP	229
Database Server: MySQL, MariaDB	230
Web Server: Nginx, Apache	230
Git	230
Package Tools	231
Docker	231
XAMPP	240
Homestead	242
Composer	242
Instalasi Composer	243
Postman	244
Penggunaan	244
Generate Dokumentasi	246
Laravel	249
Mengenal Laravel	250
Instalasi	250
Konfigurasi	252
Variabel Konfigurasi	252
Virtual Domain & Pretty URL	253
Struktur Direktori Aplikasi	255
Routing	256
Routing Web	256
Routing API (Web Service)	258
HTTP Verbs Method	258
Routing Parameter	260

Routing Name	262
Routing Group	262
Controller	265
Middleware	266
Rate Limiting	266
CORS	268
Multiple Middleware	272
Database	272
Konfigurasi	273
Migration	274
Seeding	285
Interact with Database	296
API Resources	309
Handling Error	317
Authentication	319
Konfigurasi	320
Get Authenticate User	320
Check Authenticate User	320
Protect Routing	321
Authentication Mechanism	321
Kesimpulan	331
Finishing Project	332
Intro	332
Global Constant	332
Layout Aplikasi	333
Component C-Header	334
Component C-Footer	336
Component C-Side-Bar	338
Content: Home & About	343
Update Halaman Home	346
Endpoint Random Category	347
Endpoint Top Book	349
Template & Script Home	352
Halaman Kategori Buku	359
Endpoint Book	359
Template & Script	361
Mendaftarkan Router Categories	364
Modifikasi CHeader & CFooter	365
Halaman Buku	367
Endpoint Book	367
Template & Script	369
Mendaftarkan Router Books	371
Halaman Detail Kategori Buku	372
Endpoint Detail Category	373
Template & Script	376
Mendaftarkan Router Category	378
Halaman Detail Buku	379
Endpoint Detail Book	379
Template & Script	382
Mendaftarkan Route Book	384
State Cart	385
Component Alert	388
Indicator Cart	392
Halaman Pencarian Buku	394

State Dialog	395
Membuat Endpoint Search	396
Membuat Component Search	398
Trigger Halaman Pencarian	401
Halaman Login	402
Endpoint Login	402
State Login / Auth	403
Dynamic Component	404
Membuat Component Login	406
Fitur Logout	410
Endpoint Logout	411
Membuat Link Logout	411
Halaman Register	415
Endpoint Register	415
Membuat Component Register	416
Link Halaman Register	419
Halaman Keranjang Belanja	420
Link Keranjang Belanja	420
Mengupdate State Cart	421
Membuat Component Keranjang Belanja	423
Halaman Checkout	425
Endpoint Province & City	425
Endpoint Update Alamat Pengiriman	429
Membuat Component Checkout Part 1	430
Routing Checkout	433
Menampilkan Cart Pada Component Checkout	438
Endpoint Couriers	440
Endpoint Courier Services	441
Menampilkan Form Courier Pada Component Checkout	448
Endpoint Payment	451
Update Tombol Pay Pada Component Checkout	457
Halaman Pembayaran	462
Halaman Profile	465
Link Profile di SideBar	465
Membuat Component Profile	466
Halaman Histori Belanja	467
Endpoint My Order	467
Link My Order di SideBar	468
Membuat Component My Order	469
Source Code	471
Kesimpulan	471
Deployment	473
Intro	473
Diskon Hosting & VPS	473
Persiapan	473
Persiapan Aplikasi Web Frontend	474
Persiapan Aplikasi Web Service	474
Konfigurasi	475
Matikan Mode Debug	475
Proses Deployment	477
Deployment Aplikasi Web Frontend	477
Deployment Aplikasi Web Service	484
Membuat Sub Domain	484
Membuat & Mengimport Database	486

Mengunggah File Aplikasi	491
Kesimpulan	499

Mengenal Vue

Intro

Pada bab ini, kamu akan diajak mengenal Vue, mengapa memilih Vue, tools apa saja yang diperlukan dan bagaimana cara menggunakannya secara 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 single page application (SPA).

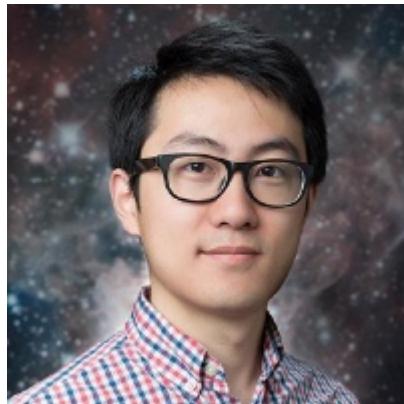


Vue sebagaimana Javascript memang awalnya didesain untuk kebutuhan web, namun seiring perkembangan teknologi yang mendukung Javascript, maka saat ini Vue juga mulai dapat digunakan untuk mengembangkan aplikasi berbasis desktop dan mobile.

Situs resmi Vue bisa kita jumpai pada alamat <http://vuejs.org>, adapun githubnya <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 Javascript & AngularJS (versi 1).



Hal inilah yang 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 ini Vue telah mencapai versi 2.0 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.

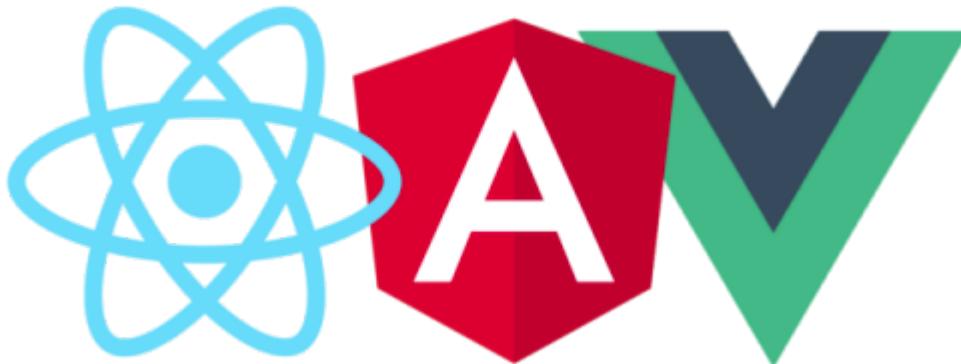
Mengapa Memilih Vue?

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 (Juni 2018), jumlah star (bintang) di akun Githubnya (<https://github.com/vuejs/vue>) mencapai lebih dari 100 ribu user dan di-fork oleh sekitar 14 ribu user, meski jumlah ini sebelas duabelas dengan perolehan React dan jauh di atas Angular (js & io)

The screenshot shows the GitHub repository page for `vuejs/vue`. The page includes a search bar, navigation links for Pull requests, Issues, Marketplace, and Explore, and a notification bell icon. The repository statistics are displayed: 2,594 commits, 22 branches, 227 releases, 189 contributors, and an MIT license. The repository has 5,051 watchers, 101,578 stars, and 14,416 forks.

A progressive, incrementally-adoptable JavaScript framework for building UI on the web. <http://vuejs.org>

vue javascript frontend framework

2,594 commits 22 branches 227 releases 189 contributors MIT

The screenshot shows the GitHub repository page for `facebook/react`. The page includes a search bar, navigation links for Pull requests, Issues, Marketplace, and Explore, and a notification bell icon. The repository statistics are displayed: 9,985 commits, 20 branches, 94 releases, 1,193 contributors, and an MIT license. The repository has 5,973 watchers, 101,543 stars, and 18,537 forks.

A declarative, efficient, and flexible JavaScript library for building user interfaces. <https://reactjs.org>

javascript react frontend declarative ui library

9,985 commits 20 branches 94 releases 1,193 contributors MIT

The screenshot shows the GitHub repository page for `angular/angular`. The page includes a search bar, navigation links for Pull requests, Issues, Marketplace, and Explore, and a notification bell icon. The repository statistics are displayed: 10,553 commits, 40 branches, 230 releases, 649 contributors, and an MIT license. The repository has 3,064 watchers, 37,041 stars, and 8,971 forks.

One framework. Mobile & desktop. <https://angular.io>

angular typescript web javascript pwa web-framework web-performance

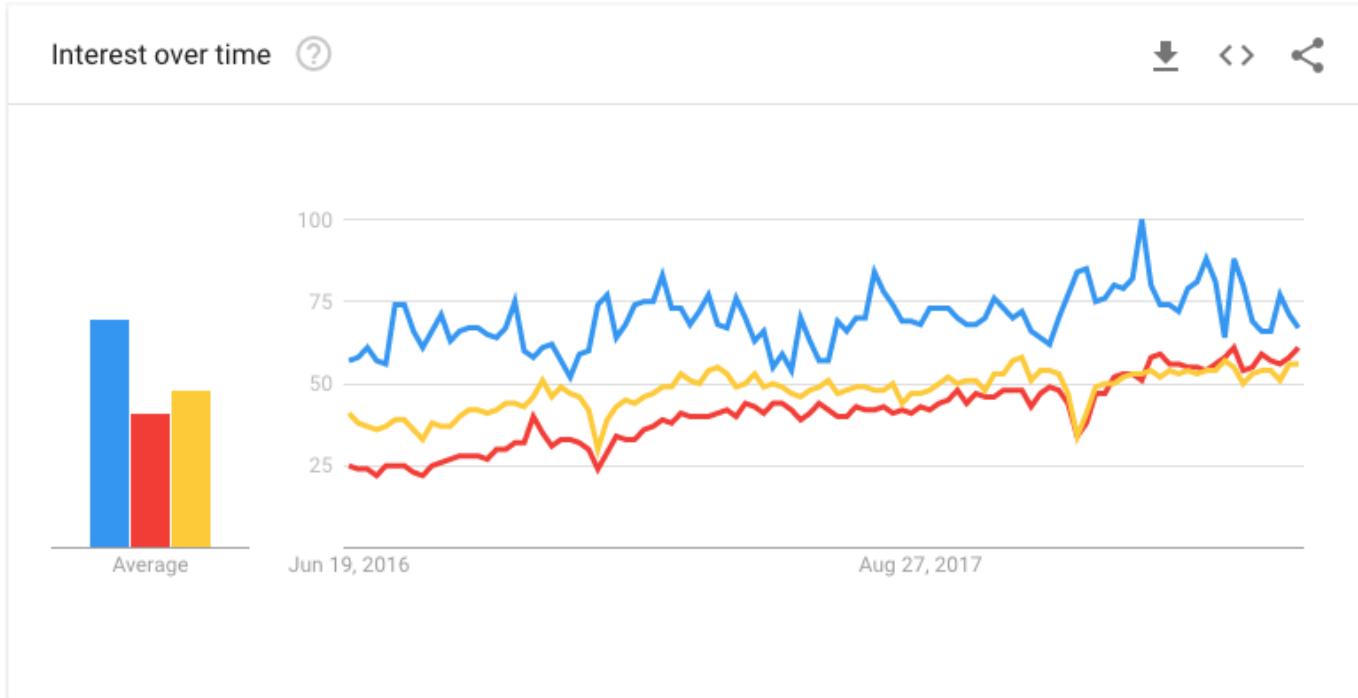
10,553 commits 40 branches 230 releases 649 contributors MIT

The screenshot shows the GitHub repository page for `angular/angular.js`. The page includes a search bar, navigation links for Pull requests, Issues, Marketplace, and Explore, and a notification bell icon. The repository statistics are displayed: 8,807 commits, 15 branches, 197 releases, 1,597 contributors, and an MIT license. The repository has 4,346 watchers, 58,604 stars, and 28,937 forks.

AngularJS - HTML enhanced for web apps! <https://angularjs.org>

8,807 commits 15 branches 197 releases 1,597 contributors MIT

Namun, berdasarkan data dari Google Trends, pencarian terkait Vue selama dua tahun terakhir jauh melampaui dua rivalnya tersebut.



Sumber: <https://trends.google.com/trends/explore?date=2016-06-16%202018-06-16&q=vue,react,angular>

Tentu saja hal ini hanyalah salah satu parameter atau tolok ukur dari kepopuleran suatu library 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 2.0, 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

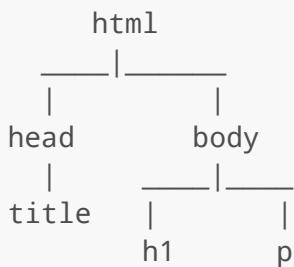
Berikut ini beberapa fitur utama yang dimiliki Vue.

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.

```
<html>
  <head>
    <title>Contoh</title>
  </head>
  <body>
    <h1> Halo </h1>
    <p> Test </p>
  </body>
</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`.



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

```
const h1s = Array.from(document.getElementsByTagName('h1'))  
  
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 7 ✓	65 7 ✓	9 7 ✓	10 10.12 ✓	16 10 ✓	8 10.10 ✓
			10 8 ✓			
			11 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 Javascript, maka kita perlu menambahkannya ke dalam halaman HTML kita sebelum kita menggunakanya. Saat buku ini ditulis, versi terbaru Vue adalah 2.5.17 (September 2018). 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.

```
<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@2.5.17/dist/vue.js> atau <https://unpkg.com/vue@2.5.17/dist/vue.js>

Catatan: sesuaikan dengan versi saat ini. Silahkan cek versi yang tersedia pada tautan ini

<https://cdn.jsdelivr.net/npm/vue>

```
<script src="https://cdn.jsdelivr.net/npm/vue@2.5.17/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.

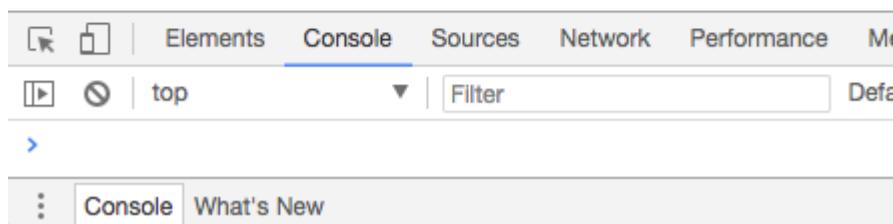
Disini, kita akan buktikan seberapa natural Vue bagi kamu yang sudah terbiasa dengan HTML, CSS & JS. Mari kita mulai dengan membuat file HTML dengan nama index.html (tentu saja kamu boleh menggunakan nama lain) sebagaimana kode HTML pada umumnya.

```
<!DOCTYPE html>
<html>
<head>
  <title>Belajar Vue</title>
</head>
<body>
  <div id="app">
    <h1>Hello world</h1>
  </div>
</body>
</html>
```

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



Hello world



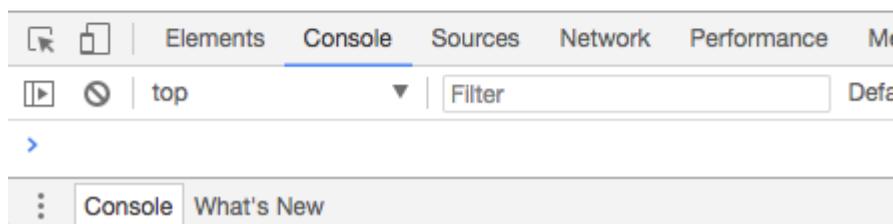
Maka pada browser akan muncul teks "Hello world". Apakah ini cukup natural? Mudah sekali bukan? Oh bukan, penulis hanya bercanda, itu bukan Vue, itu hanya HTML biasa 😊. Karenanya, mari kita ubah kode di atas (di dalam tag HTML body) menjadi sebagai berikut.

```
<!DOCTYPE html>
<html>
<head>
  <title>Belajar Vue</title>
  <script src="lib/vue.js"></script>
</head>
<body>
  <div id="app">
    <h1>{{ message }}</h1>
  </div>
  <script type="text/javascript">
var vm = new Vue({
  el: '#app',
  data: {
    message: 'Hello world!'
  }
})
</script>
</body>
</html>
```

Kode di atas akan menghasilkan tampilan sebagai berikut.



Hello world



Wah kok ribet sekali? Hanya untuk menuliskan HTML di web browser kodenya 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

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

```

<!DOCTYPE html>
<html>
  <head>
    <title>Belajar Vue</title>
    <script src="lib/vue.min.js"></script>
  </head>
  <body>
  </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.

```
<div id="app">  
  ...  
</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.

```
<div id="app">  
  <h1>{{ message }}</h1>  
</div>
```

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

```
<h1>Pesanan: {{ message }}</h1>
```

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

```
<h1>{{ 'Pesanan: ' + message }}</h1>
```

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

```
var vm = new Vue({  
  el: '#app',  
  data: {  
    message: 'Hello world!'  
  }  
)
```

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.

```

var vm = new Vue({
  data: {
    message: 'Hello world!'
  }
})

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).

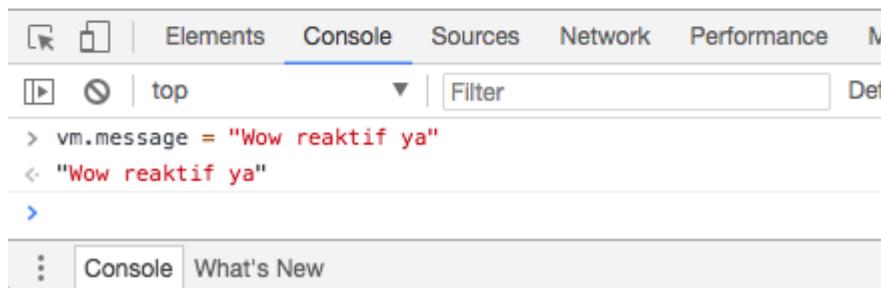
Pada console, mari kita ubah variabel `message`, dengan menggunakan perintah berikut berikut:

```
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 Javascript yang dijalankan misalnya melalui event onclick button, input dari user, dsb.

Kesimpulan

Vue merupakan pustaka Javascript 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 Javascript.

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?

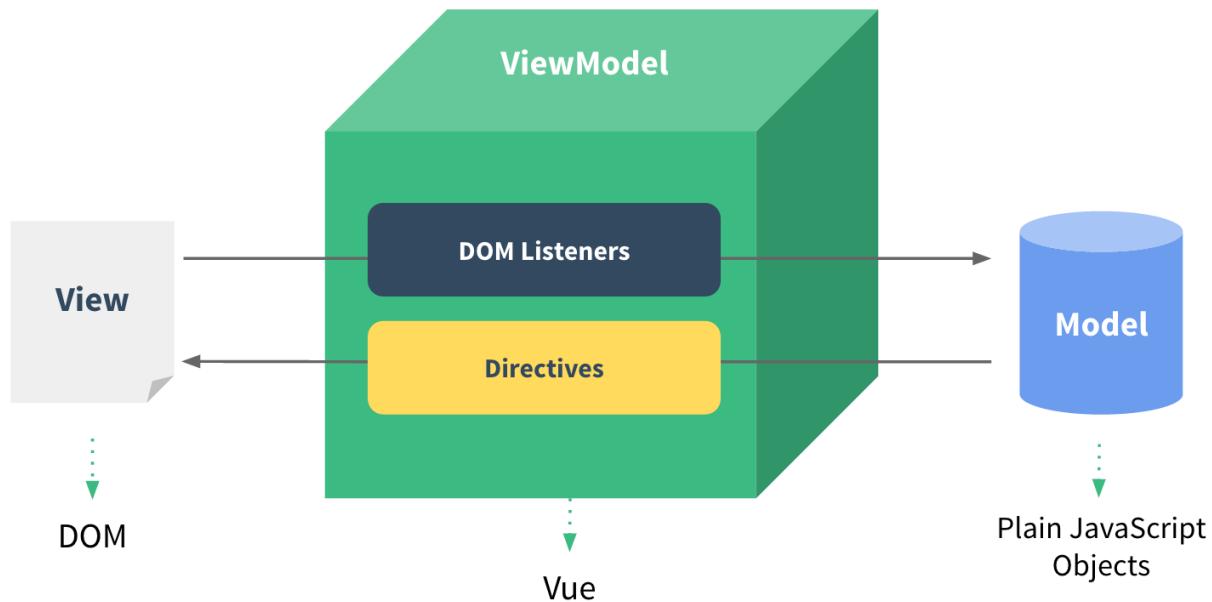
Dasar-Dasar Vue

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 di terima 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.

```
var vm = new Vue({  
  // options  
})
```

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. Dimana 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.

```
var vm = new Vue({
  el: '#app',
  ...
})
```

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.

```
var vm = new Vue({
  ...
  data: {
    message: 'Hello world!'
  }
})
```

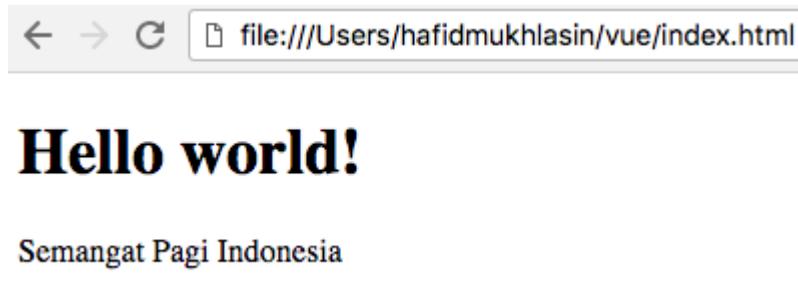
Pada kode di atas terlihat bahwa properti `data` berbentuk objek yang bertindak sebagai kontainer dari variabel `message`. Kita bisa juga menambahkan variabel lain dengan cara sebagai berikut.

```
...
data: {
  message: 'Hello world!',
  teks: 'Semangat Pagi Indonesia'
}
...
```

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.

```
<div id="app">
  <h1>{{ message }}</h1>
  <p>{{ teks }}</p>
</div>
```

Tampilan pada browser akan seperti di bawah ini.



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

```
data: {
  name: 'Hafid', // string
  age: 33, // integer
  gender: true, // boolean (Pria)
  hobby: ['coding', 'sleeping'], // array
  children: {
    1: 'Ammar',
    2: 'Faqih',
    3: 'Syamil'
  } // object
}
```

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.

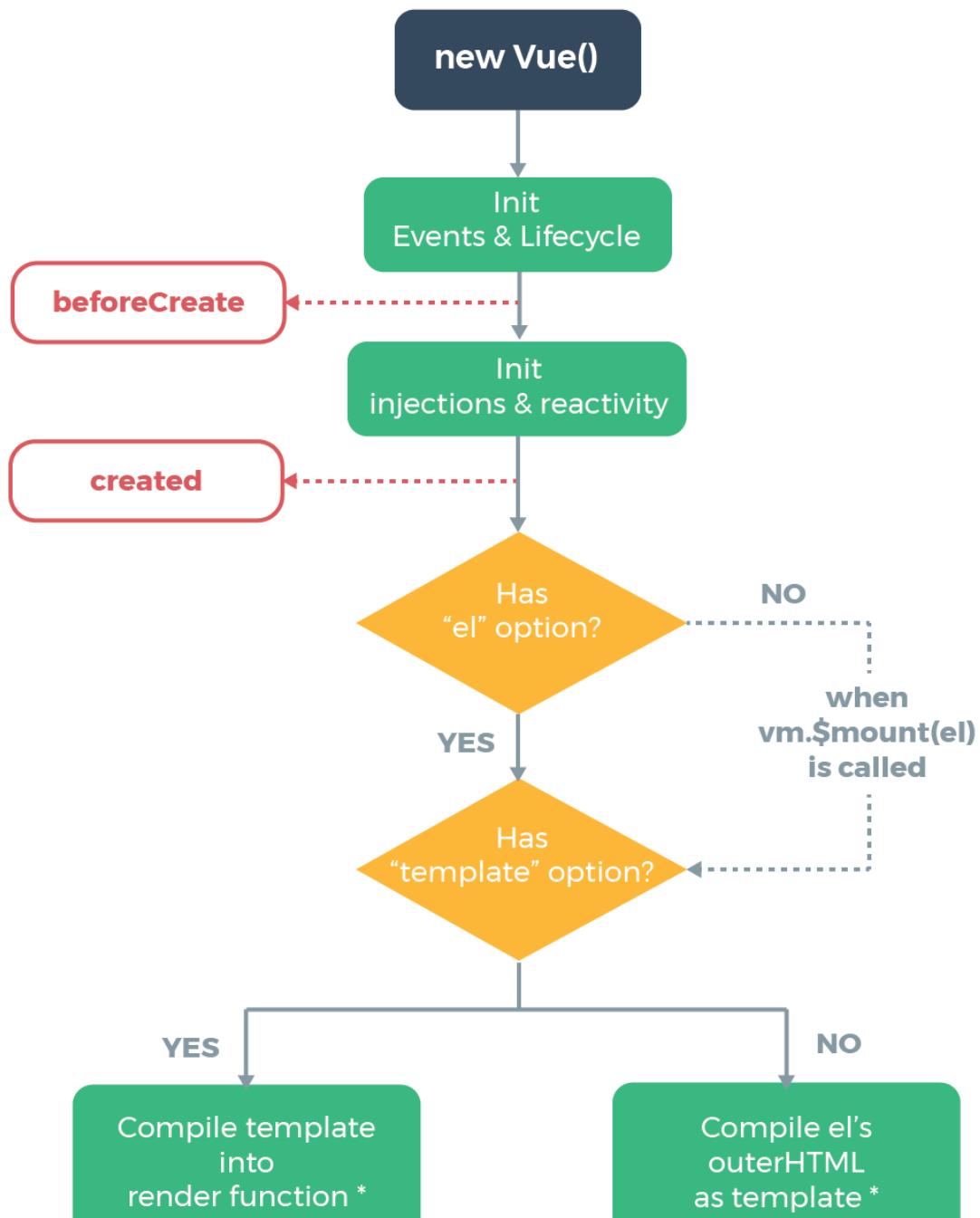
Properti data ini bisa kita akses secara langsung dari dalam objek Vue dengan menggunakan `this.$data` atau `this.message`, sedangkan jika diakses dari luar objek Vue menggunakan nama variabel objeknya `vm.$data` atau `vm.message`.

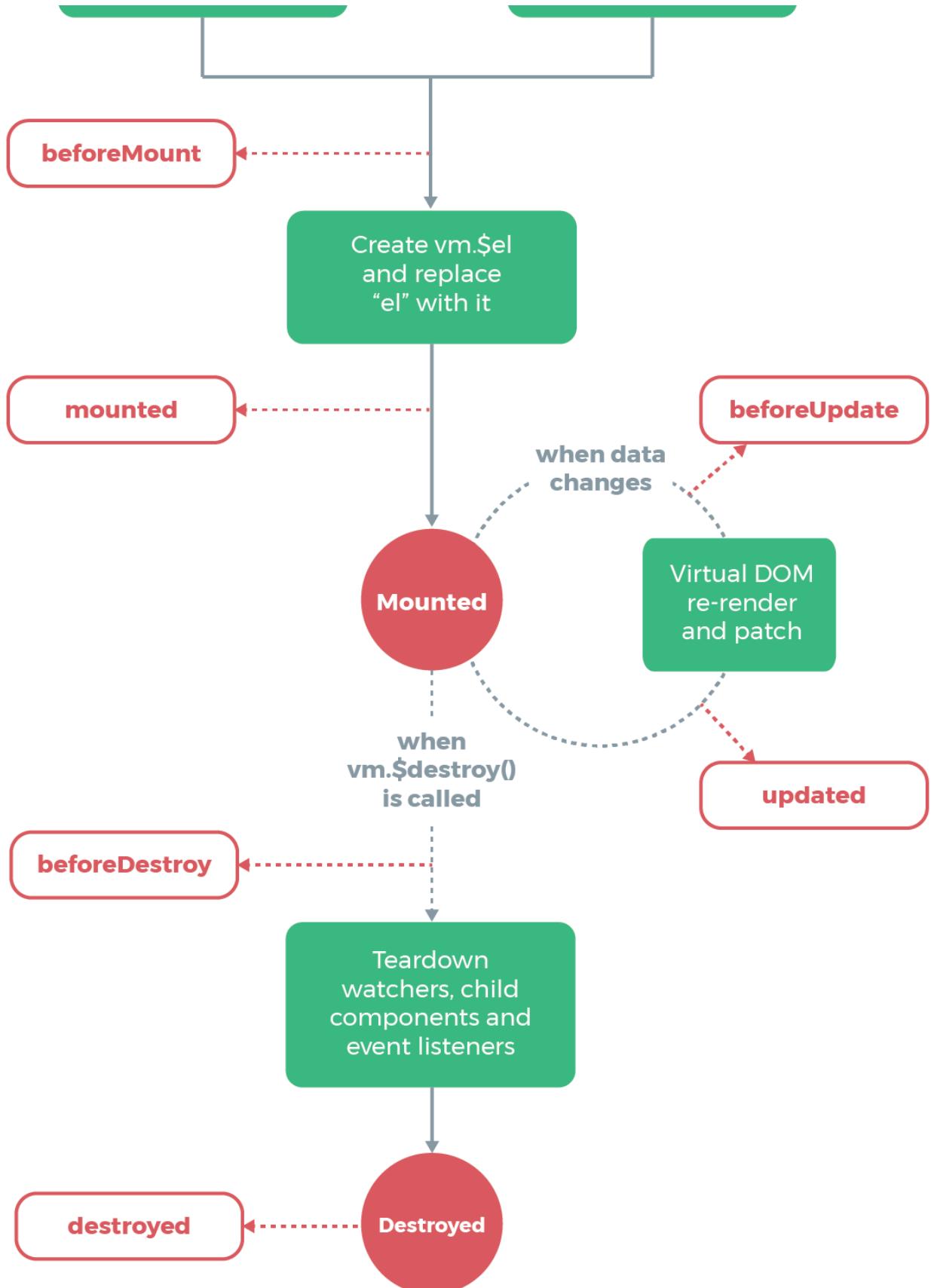
Siklus Objek Vue

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.





* 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.

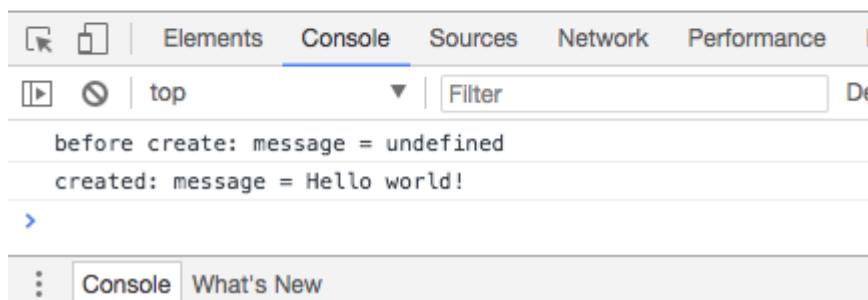
Berikut ini contoh kode untuk menggunakan kedua hook ini yaitu dengan method beforeCreate dan created.

```
var vm = new Vue({  
  el: '#app',  
  data: {  
    message: 'Hello world!',  
  },  
  beforeCreate () {  
    console.log('before create: '+  
      'message = ' + this.message)  
  },  
  created () {  
    console.log('created: '+  
      'message = ' + this.message)  
  },  
});
```

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



Hello world!

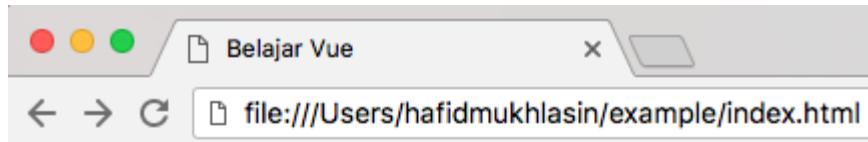


Pada gambar di atas terlihat bahwa hook created bisa mengakses variabel message sebaliknya beforeCreate tidak bisa.

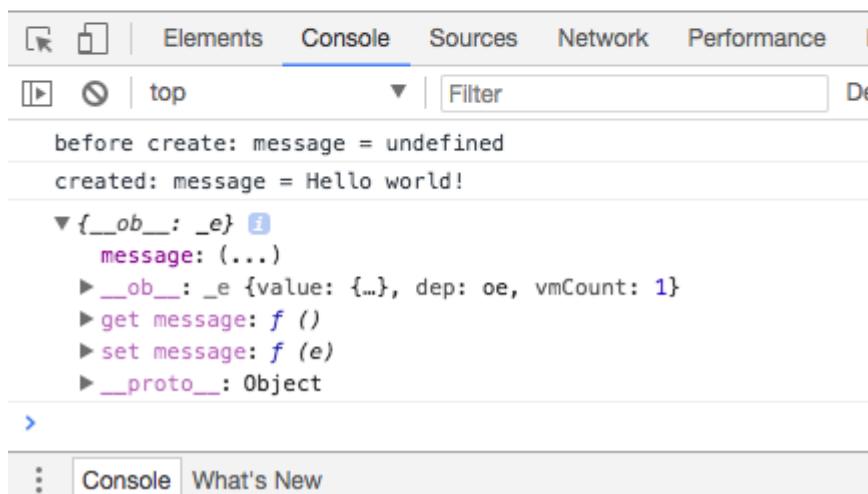
Apa yang terjadi pada properti data dapat kita lihat juga dengan cara tambahkan kode hook created menjadi sebagai berikut.

```
created () {
  console.log('created: ' +
    'message = ' + this.message)
  console.log(this.$data)
},
```

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



Hello world!



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.

```
created: function () {
    console.log('created: '+
        'message = ' + this.message)
    console.log(this.$data)
},
```

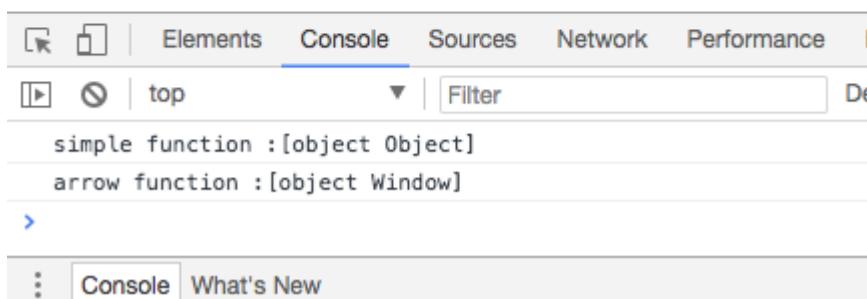
Meskipun kita dapat menggunakan format penulisan ES6, namun arrow function tidak disarankan untuk digunakan pada kasus ini, sebab keyword this di dalam arrow function tidak menunjuk ke objek Vue melainkan ke objek di atasnya (pada kasus ini objek window).

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

```
beforeCreate () {
    console.log('simple function :'+this)
},
created: () => {
    console.log('arrow function :'+this)
},
```



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).

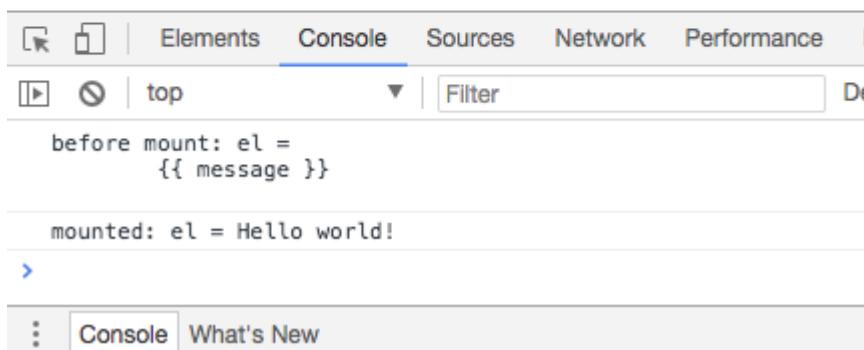
```
beforeMount() {
  console.log('before mount: ' +
  'el = ' + this.$el.textContent)
},
mounted() {
  console.log('mounted: ' +
  'el = ' + this.$el.textContent)
},
```

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!



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)

```
beforeUpdate() {
  console.log('before update: ' +
  'el = ' + this.$el.textContent)
},
updated() {
  console.log('update: ' +
  'el = ' + this.$el.textContent)
},
```

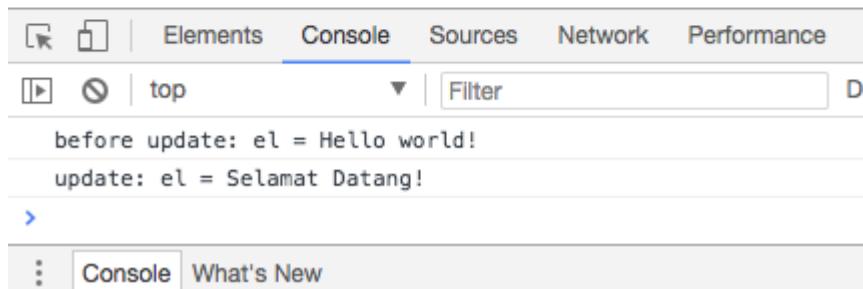
Oleh karena hook ini terjadi karena adanya manipulasi atau perubahan nilai data pada saat runtime, maka kita perlu sebuah perintah untuk mengubah variabel message. Di bawah objek Vue kita tambahkan perintah berikut.

```
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**

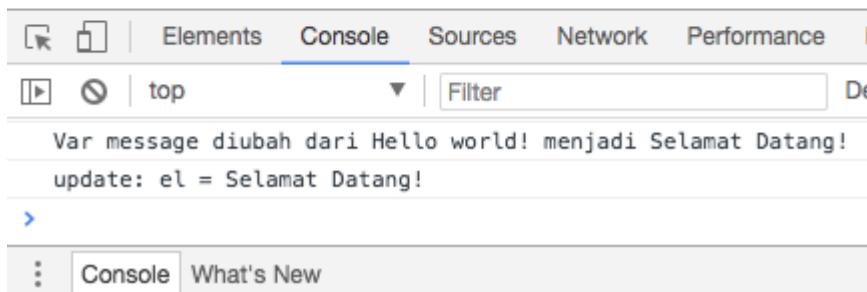
Hook beforeUpdate ini setara dengan method watch.

```
var vm = new Vue({
  el: '#app',
  data: {
    message: 'Hello world!',
  },
  beforeUpdate() {
    console.log('before update: ' +
      'el = ' + this.$el.textContent)
  },
  updated() {
    console.log('update: ' +
      'el = ' + this.$el.textContent)
  },
})
vm.$watch('message', function (newValue, oldValue) {
  console.log('Var message diubah dari '+oldValue+
    ' menjadi '+newValue)
})
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 kodennya.

```
var vm = new Vue({  
    ...  
    beforeDestroy () {  
        console.log('before destroy')  
    },  
    destroyed () {  
        console.log('destroyed')  
    },  
})  
  
vm.$destroy()
```

Hasilnya sebagai berikut.



Hello world!

Penulisan Template

Pada Vue, template merupakan kode yang menjadi dasar dari suatu tampilan yang umumnya 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.

```
<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.

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

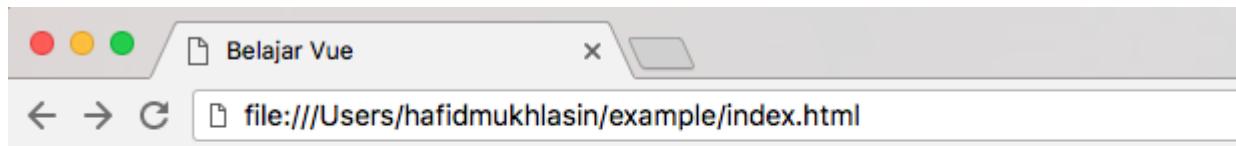
Dengan menggunakan directive ini, maka variabel message pada template seolah menjadi sebuah konstanta yang tidak dapat diubah pada saat runtime, nilainya sesuai dengan ketika pertama kali didefinisikan di objek Vue.

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

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

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 templatennya menggunakan directive v-html

```
<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.

```
<style type="text/css">
.title{
  color: green;
}
</style>

<div id="app">
  <h1 class="{{ class_h1 }}> {{ message }} </h1>
</div>

<script type="text/javascript">
Vue.config.silent = true
var vm = new Vue({
  el: '#app',
  data: {
    message: "Hello world!",
    class_h1: "title"
  },
})
</script>
```

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

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

Hasilnya.



Hello world!

JavaScript Expression

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

```
{{ 'Pesanan : ' + message }}
{{ 'Diskon : ' + total * 10% }}
{{ ok ? 'YES' : 'NO' }}
```

```
 {{ message.split(' ').reverse().join(' ') }}
```

Atau jika dalam bentuk atribut HTML maka penulisannya sebagai berikut.

```
<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.

```
<!-- ini adalah statement, bukan expression -->
{{ var a = 1 }}

<!-- gunakan ternary expressions -->
{{ if (ok) { return message } }}
```

Properti Template

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

```
<div id="app">
  <h1>{{ message }}</h1>
</div>
```

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

```
...
<div id="app"></div>

<script type="text/javascript">
var vm = new Vue({
  el: '#app',
  data: {
    message: 'Hello world!'
  },
  template: "<h1>{{ message }}</h1>"
})
</script>
...
```

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.

```
var vm = new Vue({
  el: '#app',
  data: {
    message: 'Hello world!'
  },
  render (createElement) {
    return createElement('h1', this.message)
  },
  ...
})
```

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.

```
render (createElement) {
  return createElement(Hello);
}
```

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

```
render (h){
  return h(App);
}
```

Atau jika ditulis menggunakan es6 arrow function menjadi sebagai berikut.

```
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 adakah 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.

```
var vm = new Vue({
  el: '#app',
  data: {
    counter: 0
  },
  methods: {
    increment () {
      this.counter++
    }
  }
})
```

Adapun templatanya sebagai berikut.

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

```
var vm = new Vue({
  el: '#app',
  data: {
    firstName: 'Hafid',
    lastName: 'Mukhlasin'
  },
  computed: {
    fullName: function () {
      return this.firstName + ' ' + this.lastName
    }
  }
})
```

Adapun templatanya sebagai berikut.

```
<div id="app">
{{ fullName }}
</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.

```
var vm = new Vue({
  el: '#app',
  data: {
    computed: {
      now: function () {
        return Date.now()
      }
    }
  }
})

setInterval(()=>{
  vm.now
}, 1000)
```

Adapun pada template.

```
<div id="app">
  <h1>{{ now }}</h1>
</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.

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

Fungsi `upper` dideklarasikan pada objek Vue properti filters,

```
var vm = new Vue({
  el: '#app',
  data: {
    message: 'Hello world!',
  },
  filters: {
    upper (text) {
      return text.toUpperCase()
    }
  }
})
```

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!

```
var vm = new Vue({
  el: '#app',
  data: {
    message: 'Hello world!',
  },
  filters: {
    upper (text) {
      return text.toUpperCase()
    }
  },
  methods: {
    upper (text) {
      return text.toUpperCase()
    }
  },
  computed: {
    messageUpperCase () {
      return this.message.toUpperCase()
    }
  }
})
```

Pada template, kita panggil ketiganya.

```
<!-- filters -->
<h1>Filters: {{ message | upper }}</h1>

<!-- methods -->
<h1>Methods: {{ upper(message) }}</h1>

<!-- computed -->
<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.

```
var vm = new Vue({
  el: '#app',
  data: {
    price: 500000,
  },
  filters: {
    formatCurrency (value, currency) {
      var formatter = new Intl.NumberFormat('id-ID', {
        style: 'currency',
        currency: currency,
        minimumFractionDigits: 2,
      });
      return formatter.format(value)
    }
  }
})
```

Pada template bisa kita panggil sebagai berikut:

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

Hasilnya sebagai berikut.



US\$500.000,00

Rp500.000,00

Chaining Filters ~4

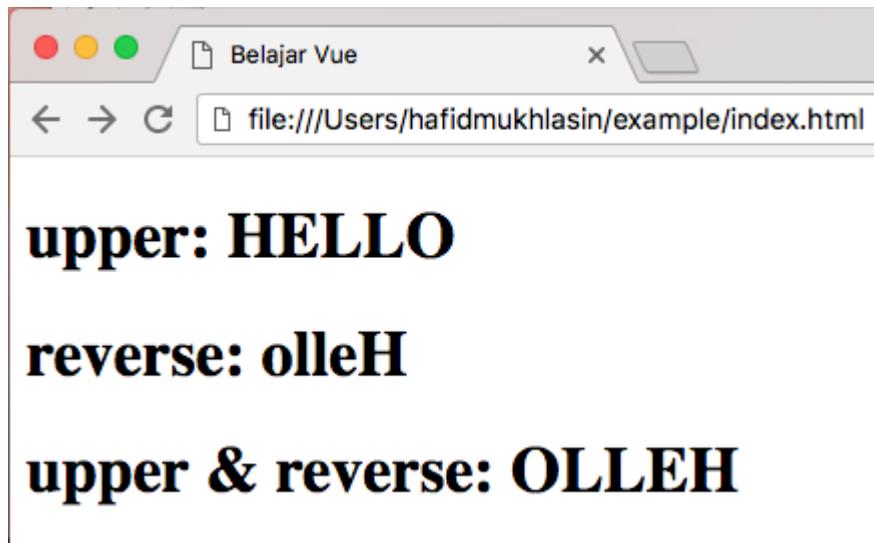
Filters juga dimungkinkan untuk digunakan secara berurutan sekaligus atau chain.

```
var vm = new Vue({
  el: '#app',
  data: {
    message: 'Hello',
  },
  filters: {
    upper (text) {
      return text.toUpperCase()
    },
    reverse (text) {
      return text.split(' ').reverse().join(' ')
    }
  }
})
```

Pada template sebagai berikut.

```
<div id="app">
  <h1> upper: {{ message | upper }} </h1>
  <h1> reverse: {{ message | reverse }} </h1>
  <h1> upper & reverse: {{ message | upper | reverse }} </h1>
</div>
```

Hasilnya:



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

```
Vue.filter('upper', function (value) {
  return value.toUpperCase()
})

var vm = new Vue({
  // ...
})
```

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

 skip

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

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

v-once

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

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

v-text

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

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

v-show

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

```
<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 di-hidden 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).

```
<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

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

atau

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

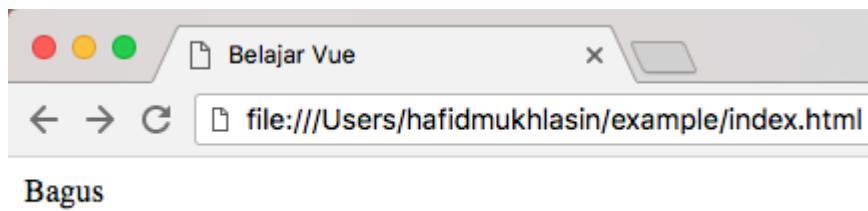
Ketika menggunakan `v-if` maka kita juga bisa menggunakan `v-else` sebagai blok pengecualian

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

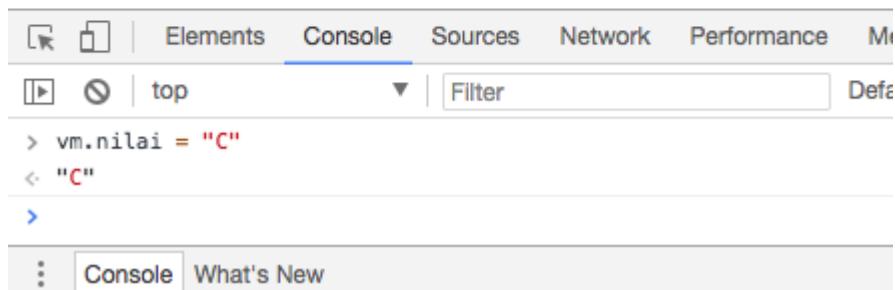
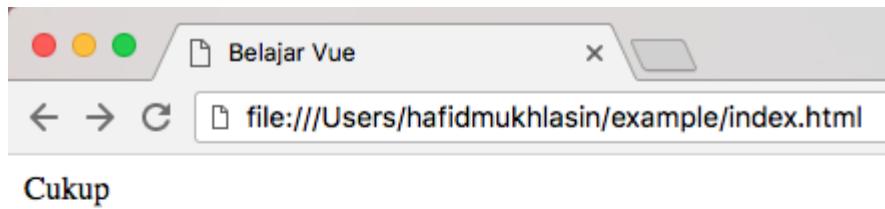
Lebih dari itu, Vue juga menyediakan `v-else-if` sebagai blok pengecualian bersyarat jika variabel yang diuji memiliki kemungkinan nilai lebih dari dua.

```
<div id="app">
  <div v-if="nilai === 'A'">
    Sempurna
  </div>
  <div v-else-if="nilai === 'B'">
    Bagus
  </div>
  <div v-else-if="nilai === 'C'">
    Cukup
  </div>
  <div v-else>
    Kurang
  </div>
</div>

<script>
var vm = new Vue({
  el: '#app',
  data: {
    nilai: "B",
  },
})
</script>
```



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.



v-on

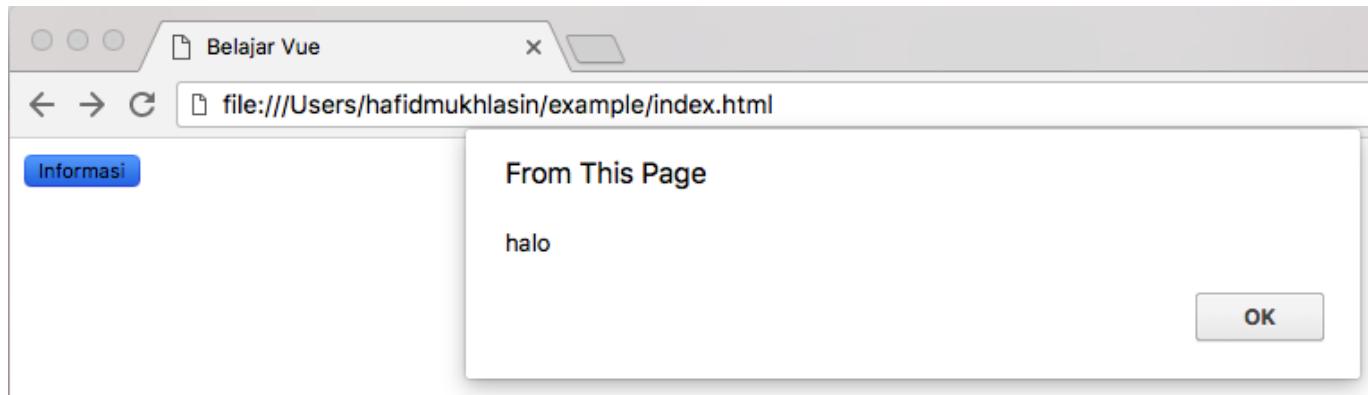
Merupakan directive yang berperan sebagai sebuah event listener pada elemen HTML/komponen Vue. Directive ini bertugas memantau aktifitas (aksi) yang dilakukan terhadap suatu elemen HTML/komponen Vue.

Contoh penggunaan.

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

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

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



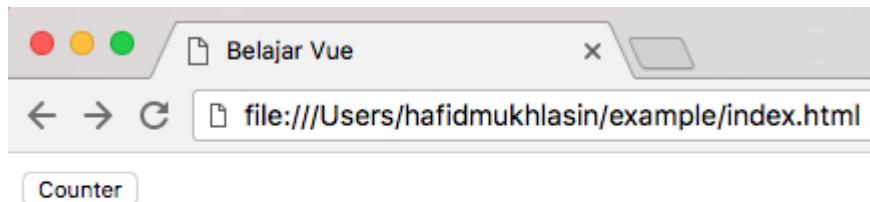
Directive **v-on:click** pada elemen HTML dikasus ini setara dengan event **onclick** native HTML biasa

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

Directive ini juga dapat kita manfaatkan untuk memanipulasi variabel data.

```
<div id="app">  
    <button v-on:click="counter += 1">  
        Counter  
    </button>  
    <p>Button di atas telah diklik sebanyak {{ counter }} kali.</p>  
</div>  
  
<script>  
var example1 = new Vue({  
    el: '#app',  
    data: {  
        counter: 0  
    }  
)  
</script>
```

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

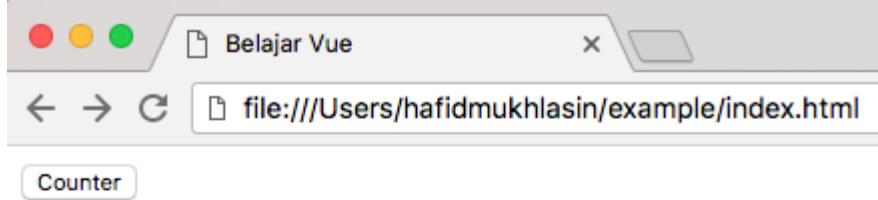


Button di atas telah diklik sebanyak 4 kali.

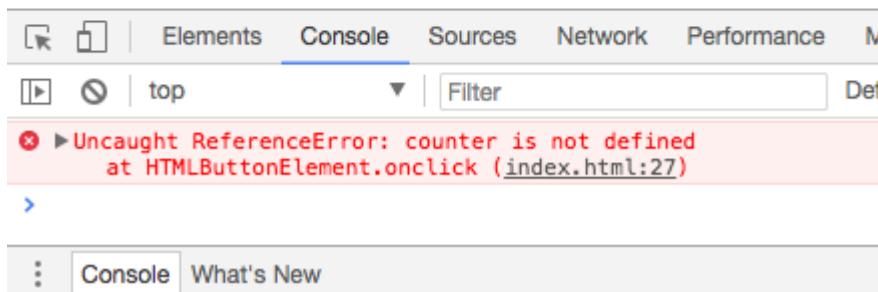
Hal ini tidak akan terjadi jika kita menggunakan event onclick native HTML.

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

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

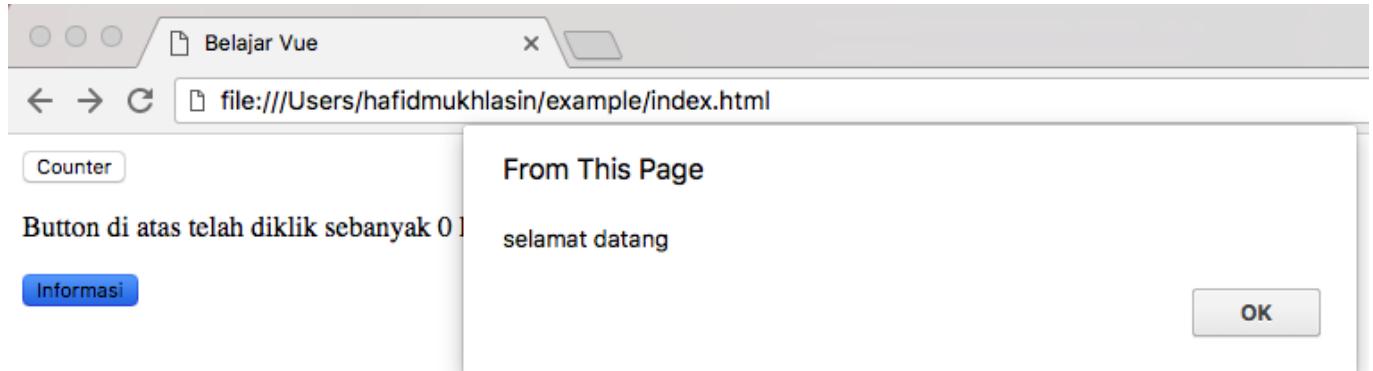
Berikut contoh pendefinisian dari methods.

```
var vm = new Vue({
  el: '#app',
  data: {
    counter: 0
  },
  methods: {
    info (text) {
      alert(text)
    }
  }
})
```

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

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

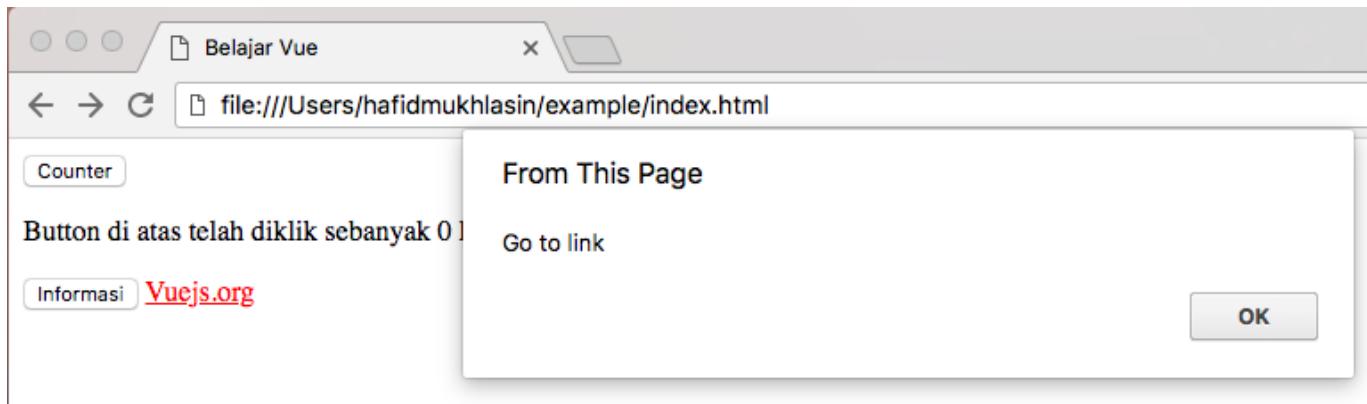
Berikut ini simulasinya.

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

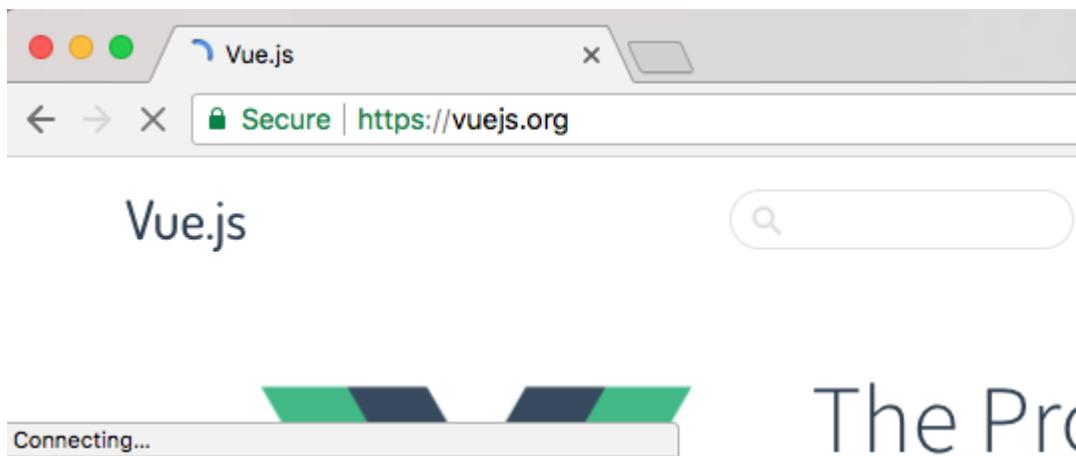
Tambahkan fungsi `link()` pada methods.

```
...  
methods: {  
    info (text) {  
        alert(text)  
    },  
    link () {  
        alert('Go to link')  
    }  
}
```

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.



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

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

Kemudian gunakan event native HTML pada fungsi link di methods

```
...  
methods: {  
  info (text) {  
    alert(text)  
  },  
  link (event) {  
    alert('Go to link')  
    event.preventDefault()  
  }  
}  
...
```

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.

```
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.

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

```
<button v-on:click.once="info('selamat datang')">  
  Informasi  
</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 beraksi ketika keyboard `Escape` ditekan.
- `.space` modifier ini akan beraksi ketika keyboard `Spasi` ditekan.
- `.native` modifier ini akan listen native event pada elemen root dari komponen.
- `.ctrl` modifier ini akan beraksi ketika keyboard `Ctrl` ditekan.
- `.alt` modifier ini akan beraksi ketika keyboard `Alt` ditekan.
- `.shift` modifier ini akan beraksi ketika keyboard `Shift` ditekan.
- dsb.

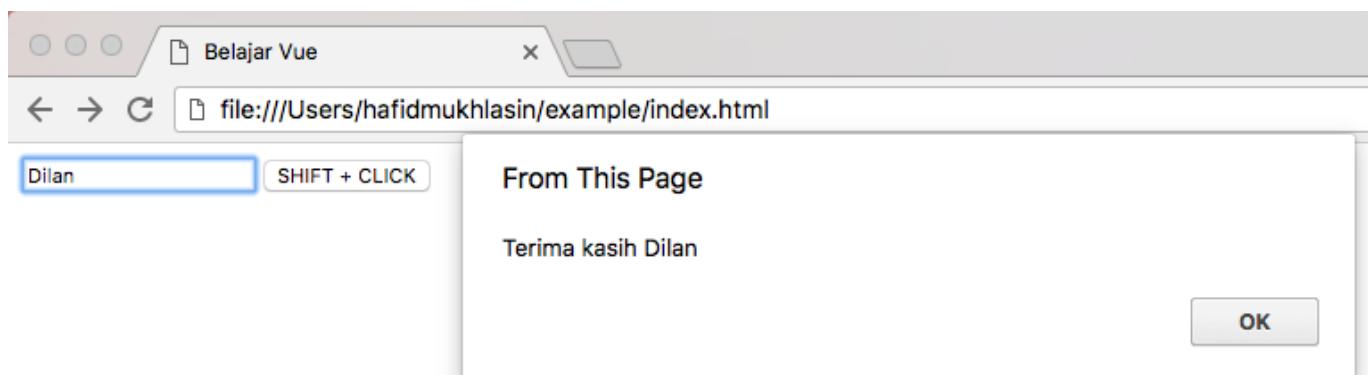
Berikut ini contoh penggunaanya.

```
<!-- mencegah reload halaman saat event submit -->
<form v-on:submit.prevent="onSubmit"></form>

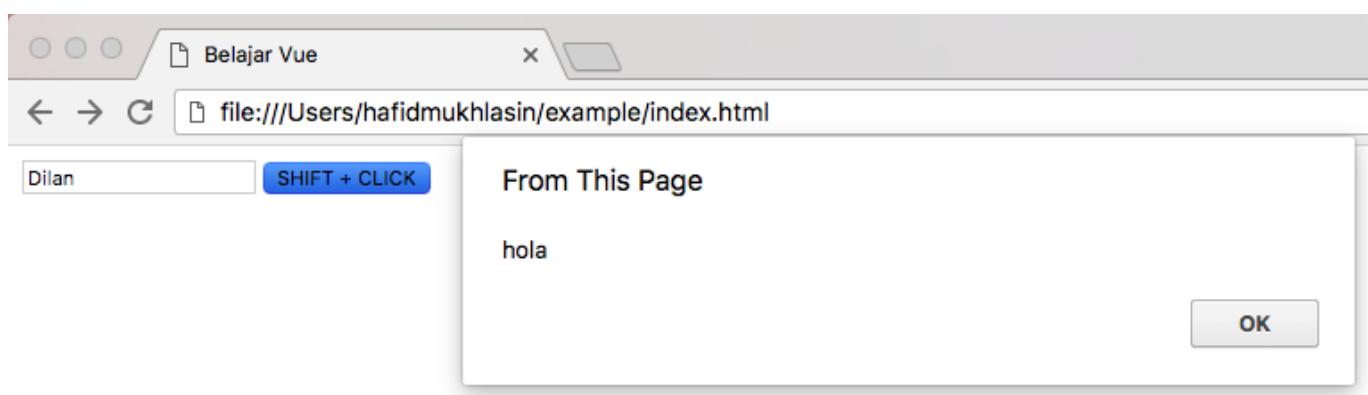
<!-- ketika ditekan enter maka akan menjalankan fungsi submit -->
<input v-on:keyup.enter="info('Terima kasih ' + $event.target.value)">

<!-- Shift + Click -->
<button v-on:click.ctrl="info('hola')">
  SHIFT + CLICK
</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.

```
<!-- object syntax (2.4.0+) -->
<button v-on="{ mousedown: doThis, mouseup: doThat }"></button>

<!-- chain modifiers -->
<button v-on:click.stop.prevent="doThis"></button>
```

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

```
<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`.

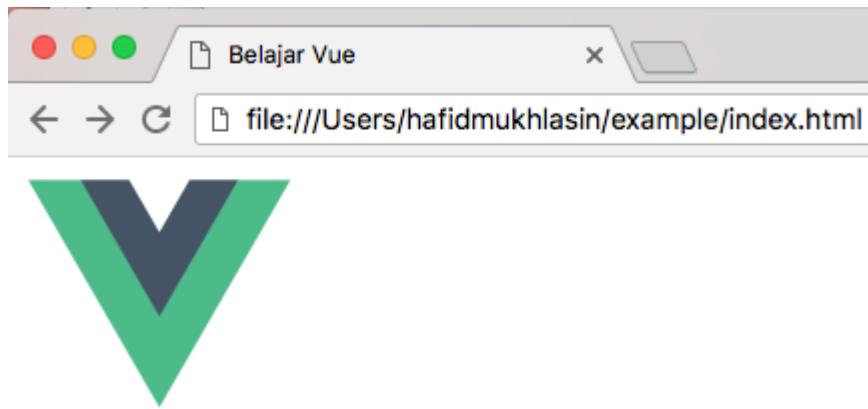
Contoh:

```
<div id="app">
  
</div>

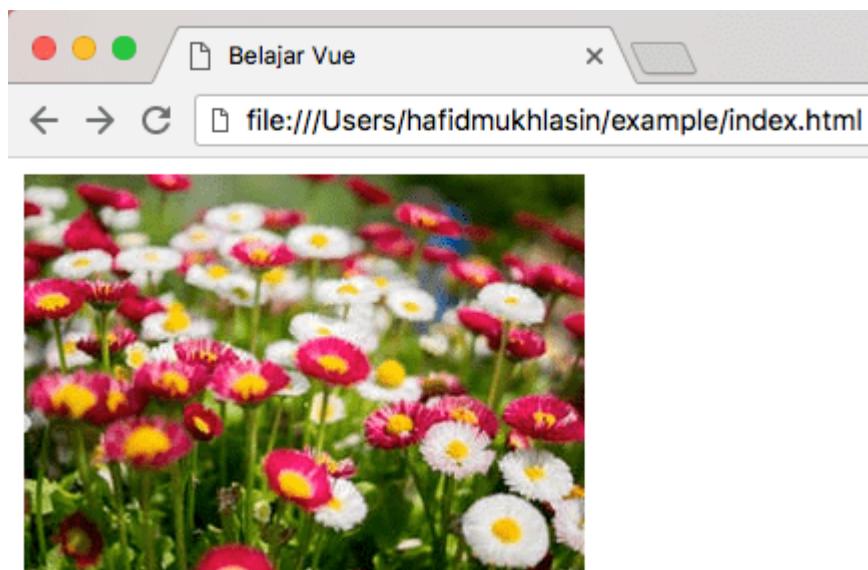
<script>
var vm = new Vue({
  el: '#app',
  data: {
    imageSrc: 'logo-vue.png',
  }
})

setTimeout(()=>{
  vm.imageSrc = 'flowers.jpg'
}, 3000);
</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 ditampilkanpun juga akan berubah karena variabel `imageSrc` diubah.



setelah tiga detik maka gambar akan berubah



Tentu saja kita bisa gabungkan dengan teks statis. Misalnya:

```

```

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

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

Boolean expression juga dimungkinkan

```
<div v-bind:class="{ red: isRed }"></div>
<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.

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

```
var vm = new Vue({  
  el: '#app',  
  data: {  
    imageID: 'image1',  
    imageSrc: 'logo-vue.png',  
  }  
})
```

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

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

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.

```
books : [
    'C++ High Performance',
    'Mastering Linux Security and Hardening', 'Python Programming
Blueprints',
    'Mastering PostgreSQL 10'
]
```

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

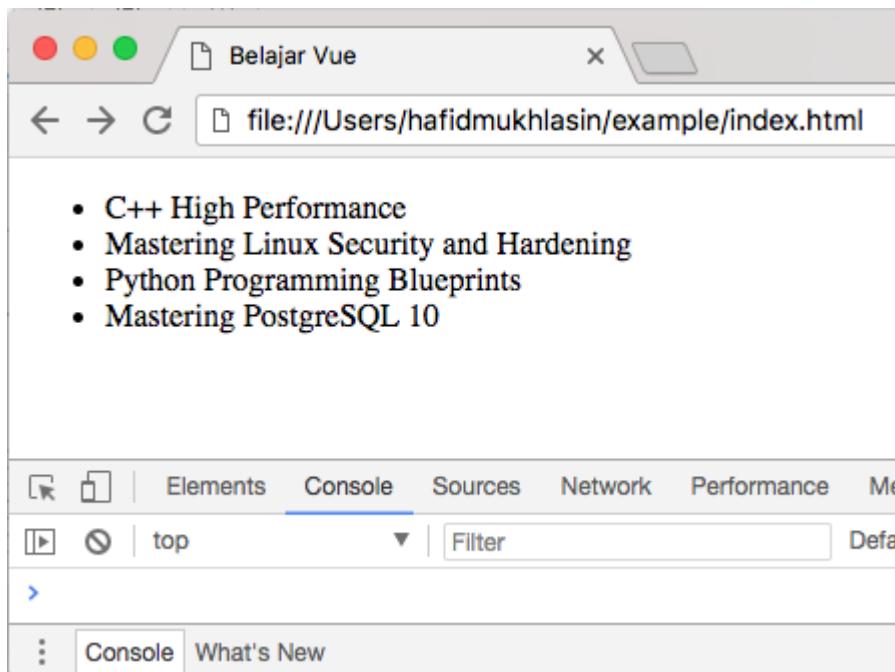
```
var vm = new Vue({
  el: '#app',
  data: {
    books : [
      'C++ High Performance',
      'Mastering Linux Security and Hardening',
      'Python Programming Blueprints',
      'Mastering PostgreSQL 10'
    ]
  }
})
```

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

```
<div id="app">
  <ul>
    <li v-for="book in books">
      {{ book }}
    </li>
  </ul>
</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.



v-for Menggunakan Tag Template

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

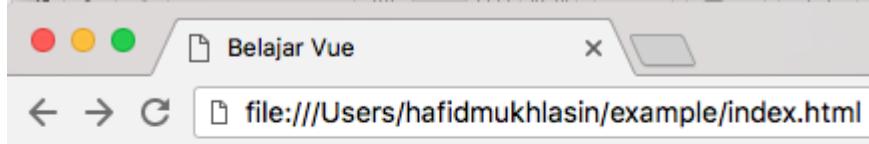
```
<div id="app">
  <ul>
    <template v-for="book in books">
      <li> {{ book }} </li>
    </template>
  </ul>
</div>
```

v-for Menggunakan Index

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

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



- 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`.

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

Menampilkan Data Objek

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

```
book: {  
    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 kode untuk templatanya sebagai berikut.

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

Berikut ini hasilnya.

A screenshot of a web browser window titled "Belajar Vue". The address bar shows "file:///Users/hafidmukhlasin/example/index.html". The content of the page 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

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

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

Berikut ini hasilnya.

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

- 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.

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

A screenshot of a web browser window titled "Belajar Vue". The address bar shows "file:///Users/hafidmukhlasin/example/index.html". The content of the page is a list of book details with indexed keys and indices:

- 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.

```
books : [
    {
        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'
    },
    {
        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'
    },
    {
        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'
    },
    {
        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'
    },
]
```

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

```
<div id="app">
  <table border=1>
    <tr v-for="book of books">
      <td>
        
      </td>
      <td>
        title: {{ book.title }} <br>
        description: {{ book.description }} <br>
        authors: {{ book.authors }} <br>
        price: {{ book.price }}
      </td>
    </tr>
  </table>
</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>`

```
<template v-for="book of books">
  <tr>
    ...
  </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 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.

```
book.title // judul buku
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.

```
<td>
  title: {{ book.title }} <br>
  description: {{ book.description }} <br>
  authors: {{ book.authors }} <br>
  price: {{ book.price }}
</td>
```

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

```
<td>
  <template v-for="(value, key) of book">
    {{ key }} : {{ value }} <br>
  </template>
</td>
```

Berikut ini hasilnya.

	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
	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
	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
	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

Atribut Key

Terkait dengan metode menampilkan list, Vue menyarankan agar sebisa 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:

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

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

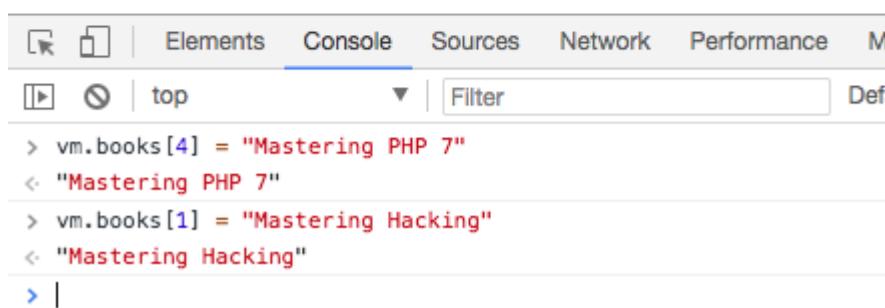
Data pada contoh sebelumnya merupakan data statis yang tidak atau belum ada perubahan. Namun sebagaimana yang telah dijelaskan diawal bahwa data pada Vue bersifat observer artinya perubahannya senantiasa dipantau dan bisa menjadi pemicu untuk perubahan yang lain. Sebagai contoh, apabila data dihubungkan 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.

```
books : [
  'C++ High Performance',
  'Mastering Linux Security and Hardening', 'Python Programming
Blueprints',
  'Mastering PostgreSQL 10'
]
```

Kita akan coba melakukan manipulasi melalui console.



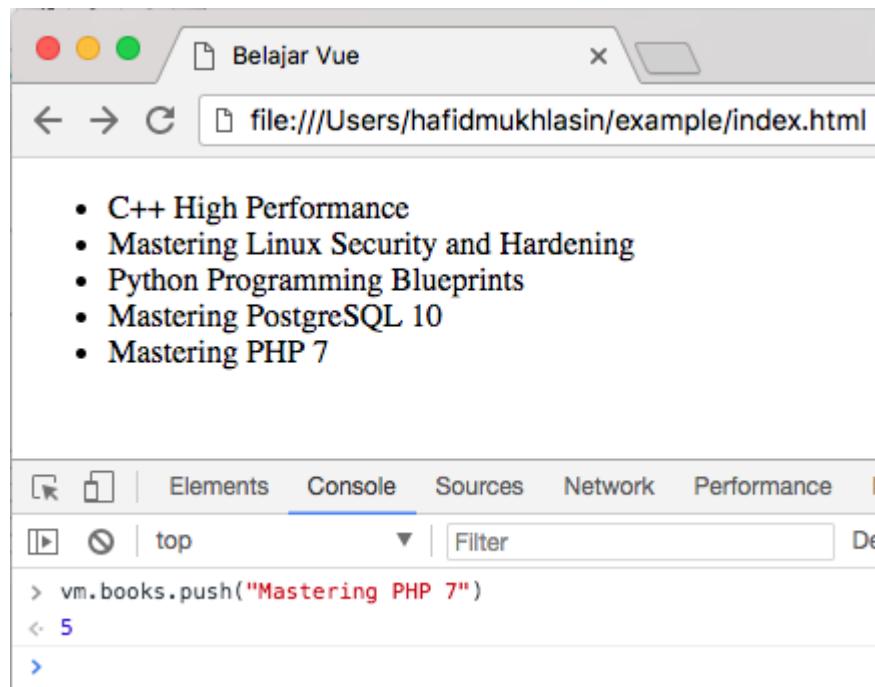
Pada contoh di atas, meski kita menambahkan data elemen baru pada variabel books (Mastering PHP 7), namun tampilan daftar buku tetap tidak berubah atau tidak dirender ulang. Demikian juga perubahan data judul buku pada index pertama dari "Mastering Linux Security and Hardening" menjadi "Masterng Hacking"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

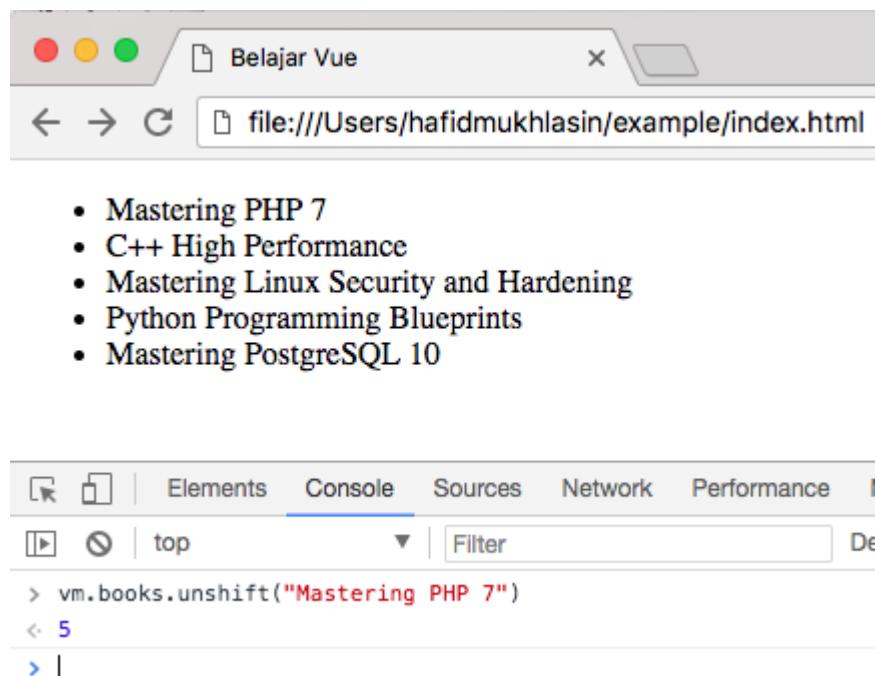
```
vm.books.push('Mastering PHP 7')
```



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')
```

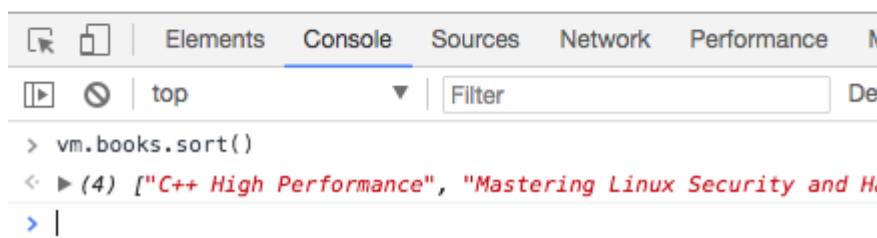


sort() & reverse()

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



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



splice()

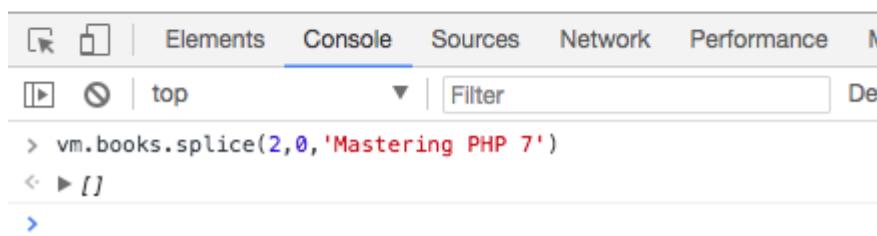
Fungsi splice ini multi fungsi, bisa digunakan untuk menambahkan data elemen baru pada suatu array. Misalnya: `vm.books.splice(2,0,'Mastering PHP 7')`

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

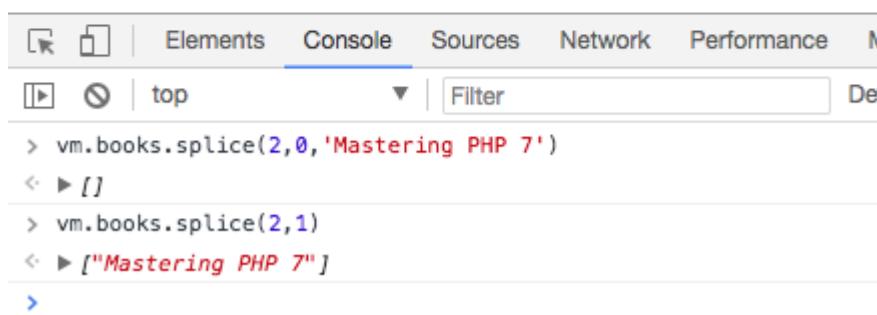


Fungsi splice juga bisa digunakan untuk menghapus elemen dari suatu array pada index tertentu. Misalnya:
`vm.books.splice(2,1)`

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

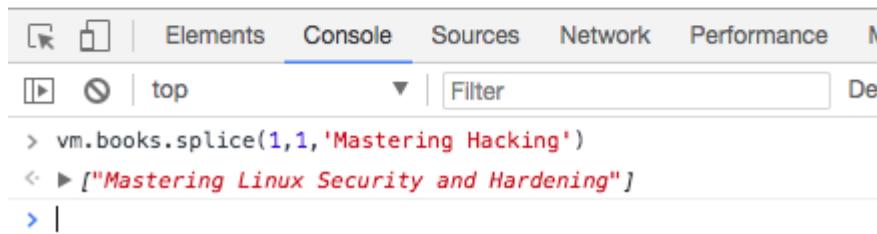


Perintah splice pun bisa digunakan untuk mengubah elemen dari suatu array pada index tertentu. Misalnya:
`vm.books.splice(1,1,'Mastering Hacking')`

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



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

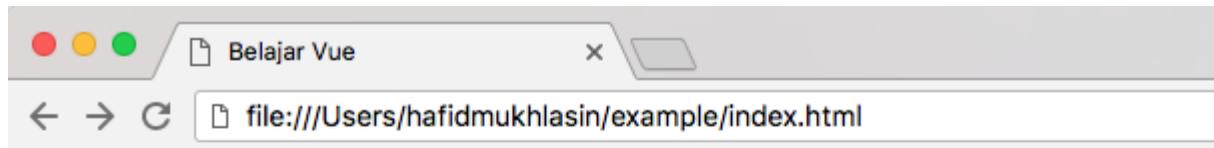


Disamping fungsi-fungsi di atas, terdapat beberapa fungsi built-in lagi terkait array yang bisa kita gunakan untuk memanipulasi elemen pada array namun sifatnya tidak melakukan perubahan langsung pada current

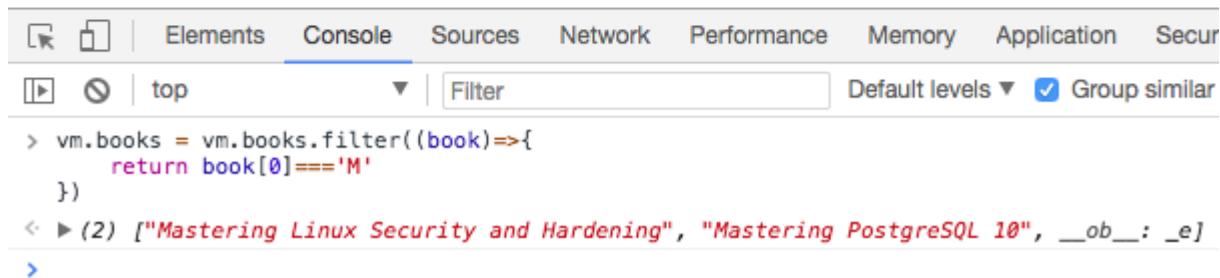
array melainkan mengembalikan data array baru. Diantaranya: filter(), concat() dan slice().

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**.

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



- 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.

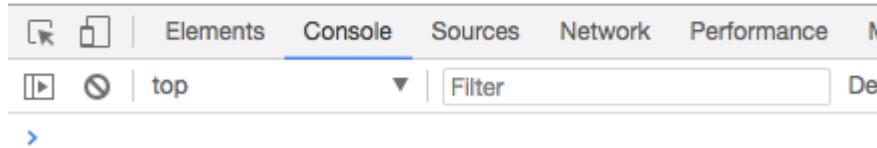
Berikut ini contoh kodennya

```
<div id="app">
  <ul>
    <li v-for="(book, index) of booksPrefixM" :key="index">
      {{ book }}
    </li>
  </ul>
</div>

<script type="text/javascript">
var vm = new Vue({
  el: '#app',
  data: {
    books : [
      'C++ High Performance',
      'Mastering Linux Security and Hardening', 'Python Programming
Blueprints',
```

```
'Mastering PostgreSQL 10'  
]  
,  
computed: {  
    booksPrefixM() {  
        return this.books.filter((book)=>{  
            return book[0]==='M'  
        })  
    }  
}  
})  
</script>
```

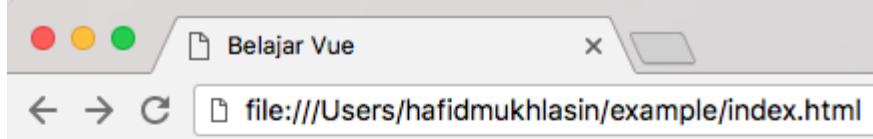
Berikut ini hasilnya



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_yang_akan_diganti, nilai_baru)
```



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

```
Elements Console Sources Network Performance M
top Filter Def
> vm.$set(vm.books, 1, "Mastering Hacking")
< "Mastering Hacking"
>
```

Perubahan Data Pada Objek

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

Asumsi, kita gunakan data objek berikut.

```
book: {
  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,
}
```

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

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

atau

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

```

• 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

```

Elements Console Sources Network Performance Memory Application Security

top Filter Default levels ▾ Group similar

```

> 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.

```

vm.book = Object.assign({}, vm.book, {
  umur: 27,
  status: 'Perjaka'
})

```

```

• 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

```

Elements Console Sources Network Performance Memory Application Security

top Filter Default levels ▾ Group similar

```

> 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.

```
<input name="username" type="text">
<input name="password" type="password">
<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 membongkar perubahan variabel data untuk diterapkan pada field input.

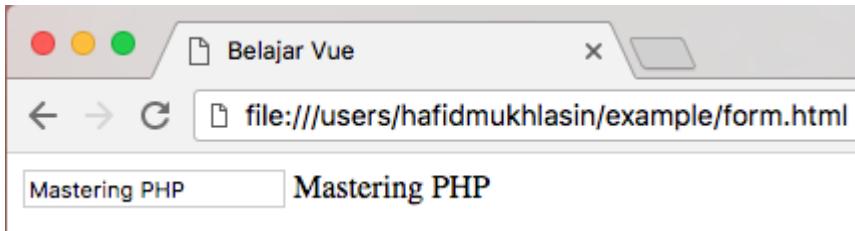
```
<div id="app">
  <form>
    <input type="text" name="title" :value="title" @input="title = $event.target.value" />

    {{ title }}
  </form>
</div>

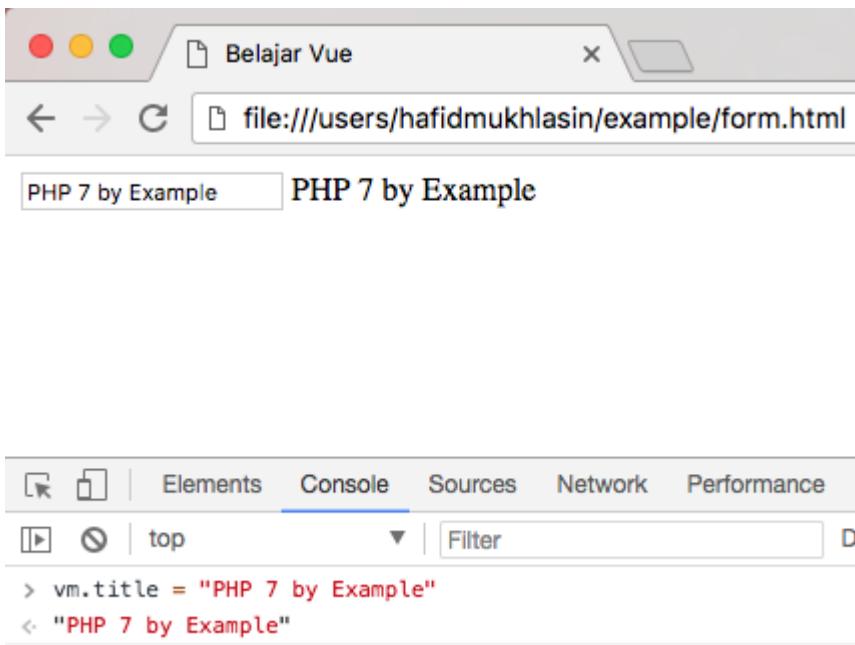
<script>
var vm = new Vue({
  el: '#app',
  data: {
    title: "Mastering "
  },
})
</script>
```

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.

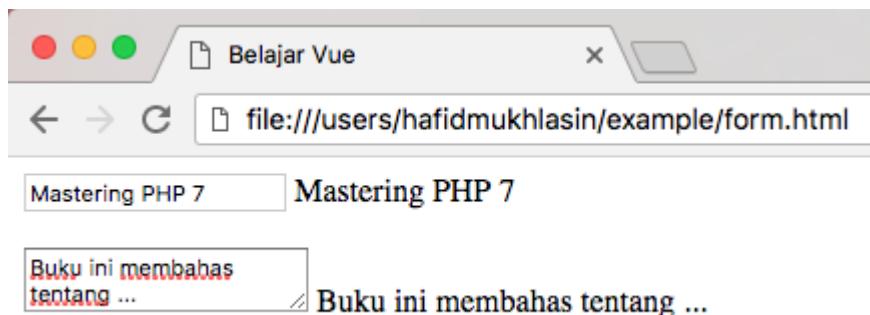
```
<form>
  <input type="text" name="title" v-model="title" placeholder="masukkan judul">
  {{ title }}
  <br><br>
  <textarea name="description" v-model="description" placeholder="masukkan deskripsi"></textarea>
  {{ description }}
</form>
```

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.

```
data: {  
    title: "",  
    description: ""  
},
```

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.

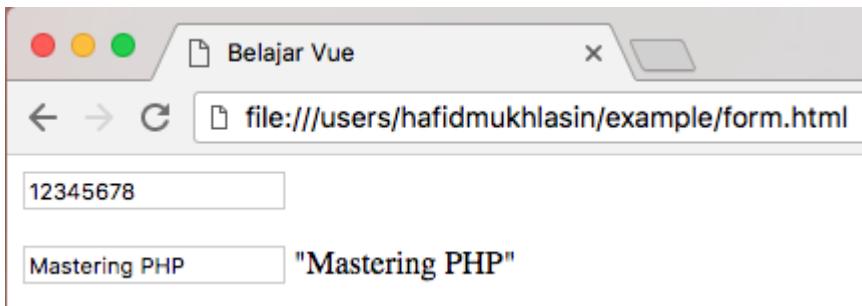
Directive v-model juga memiliki modifier, diantaranya adalah **number** dan **trim**.

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

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

Catatan: tambahkan variabel price.

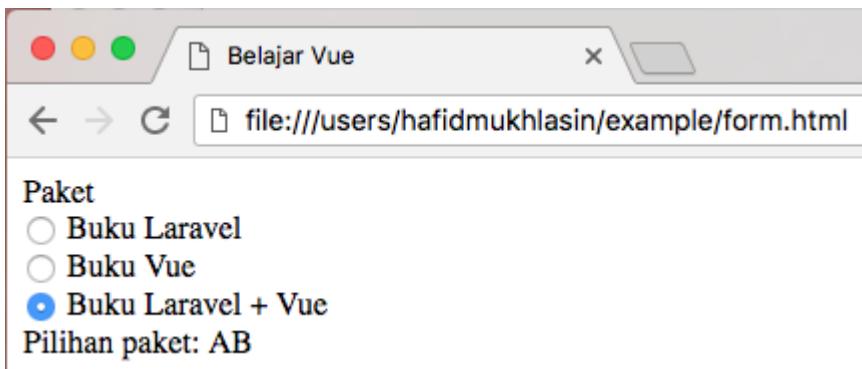
Mari kita lihat hasilnya.



Field input radion bertipe data teks.

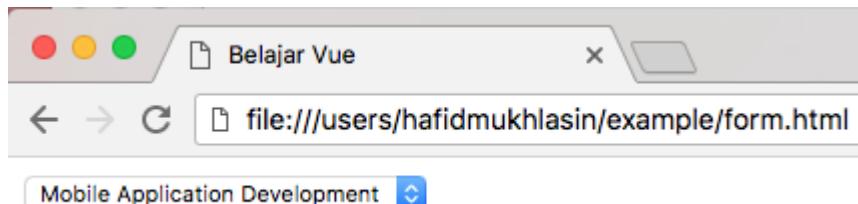
```
Paket <br>
<input type="radio" name="bukuA" value="A" v-model="paket">
<label for="bukuA">Buku Laravel</label>
<br>
<input type="radio" name="bukuB" value="B" v-model="paket">
<label for="bukuB">Buku Vue</label>
<br>
<input type="radio" name="bukuAB" value="AB" v-model="paket">
<label for="bukuAB">Buku Laravel + Vue</label>
<br>
<span>Pilihan paket: {{ paket }}</span>
```

```
data: {
  paket: ''
}
```



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

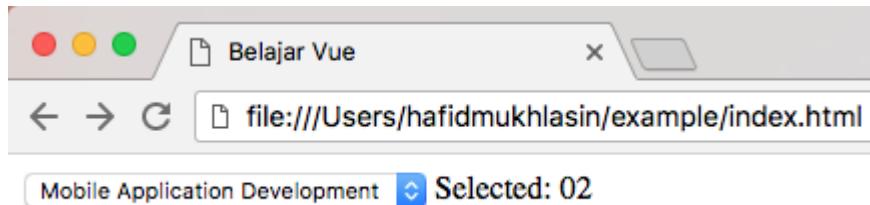
```
<select name="category" v-model="category">
  <option disabled value="">Please select one</option>
  <option>Graphics Programming</option>
  <option>Mobile Application Development</option>
  <option>Virtual and Augmented Reality</option>
</select>
```



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.

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

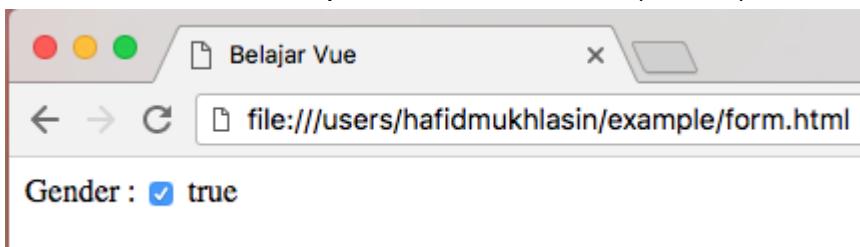


Boolean

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

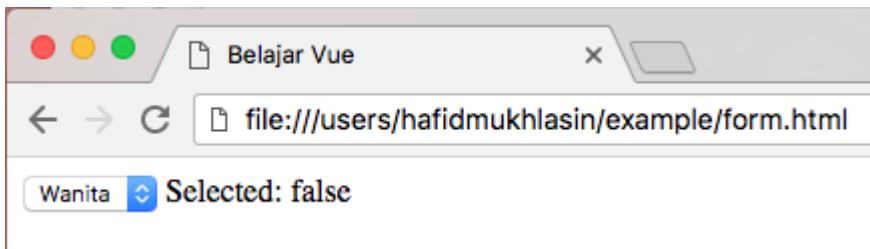
```
Gender :
<input type="checkbox" name="checkbox" v-model="gender">
<label for="checkbox">{{ gender }}</label>
```

```
data: {
    gender: true
}
```



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

```
<select name="gender" v-model="gender">
    <option value="false">Wanita</option>
    <option value="true">Pria</option>
</select>
<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.

```
Hobi <br>
<input type="checkbox" name="hobby1" value="nonton" v-model="hobbies">
<label for="hobby1">Nonton</label>
<input type="checkbox" name="hobby2" value="jalan" v-model="hobbies">
<label for="hobby2">Jalan</label>
<input type="checkbox" name="hobby3" value="makan" v-model="hobbies">
<label for="hobby3">Makan</label>
<br>
<span>Pilihan hobi: {{ hobbies }}</span>

<hr>

<select v-model="categories" multiple>
    <option value="01">Graphics Programming</option>
    <option value="02">Mobile Application Development</option>
    <option value="03">Virtual and Augmented Reality</option>
</select>
<span>Selected: {{ categories }}</span>
```

Adapun definisi dari variabel data harus sebagai array.

```
var vm = new Vue({
  el: '#app',
  data: {
    hobbies: [],
    categories: []
  }
})
```



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.

```
data: {
  categories: [],
  options: [
    { text: 'Graphics Programming', value: '01' },
    { text: 'Mobile Application Development', value: '02' },
    { text: 'Virtual and Augmented Reality', value: '03' }
  ]
}
```

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

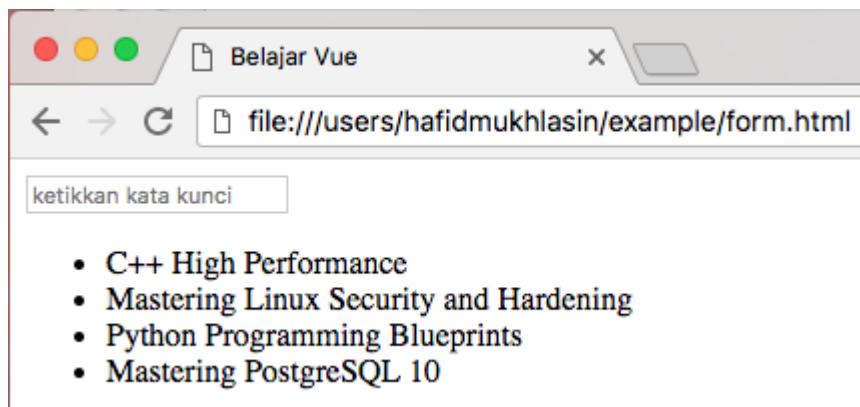
```
<div id="app">
  <input type="text" v-model="keyword" placeholder="ketikkan kata kunci">
  <ul>
    <li v-for="(book, index) of filterBooks" :key="index">
      {{ book }}
    </li>
  </ul>
</div>
```

Adapun kode Javascript-nya sebagai berikut.

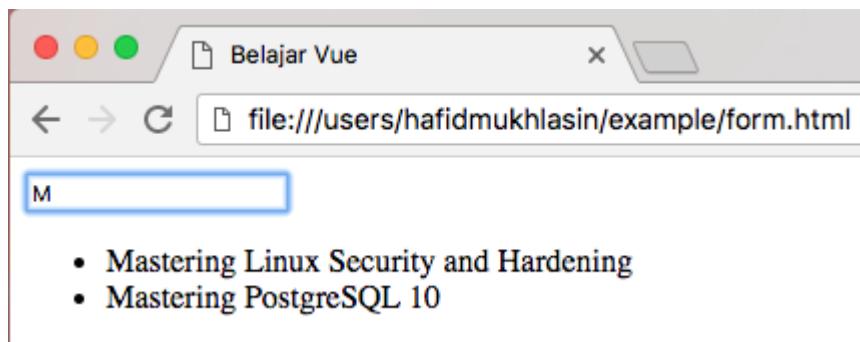
```
var vm = new Vue({
  el: '#app',
  data: {
    keyword: '',
    books : [
      'C++ High Performance',
      'Mastering Linux Security and Hardening', 'Python Programming Blueprints',
      'Mastering PostgreSQL 10'
    ],
    computed: {
      filterBooks() {
        return this.books.filter((book)=>{
          return book.includes(this.keyword)
        })
      }
    }
})
```

Ada dua variabel 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.

```
<form @submit="submitForm($event)" action="http://example.com/add-product"
method="post">
    <label>Title:</label>
    <input type="text" v-model="title" />

    <label>Description:</label>
    <textarea v-model="description"></textarea>
```

```

<label>Authors:</label>
<input type="text" v-model="authors">

<label>Price:</label>
<input type="number" v-model.number="price">

<label>Categories:</label>
<select v-model="categories" multiple>
    <option v-for="option in options" :value="option.value">
        {{ option.text }}
    </option>
</select>

<label></label>
<input type="submit" value="Submit">
</form>

```

Pada elemen form `<form @submit="submitForm" action="http://example.com/add-product" method="post">`, kita menggunakan 3 atribut yaitu 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.

```

var vm = new Vue({
    el: '#app',
    data: {
        title: '',
        description: '',
        authors: '',
        price: 0,
        categories: [],
        options: [
            { text: 'Graphics Programming', value: '01' },
            { text: 'Mobile Application Development', value: '02' },
            { text: 'Virtual and Augmented Reality', value: '03' }
        ]
    },
    methods: {
        submitForm(event){
            console.log(event)

            // kode validasi
        }
    }
})

```

```
// kode kirim ke server

// kode status informasi
alert('Terima kasih')

// block redirect ke action
event.preventDefault()
}

}

})
```

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`

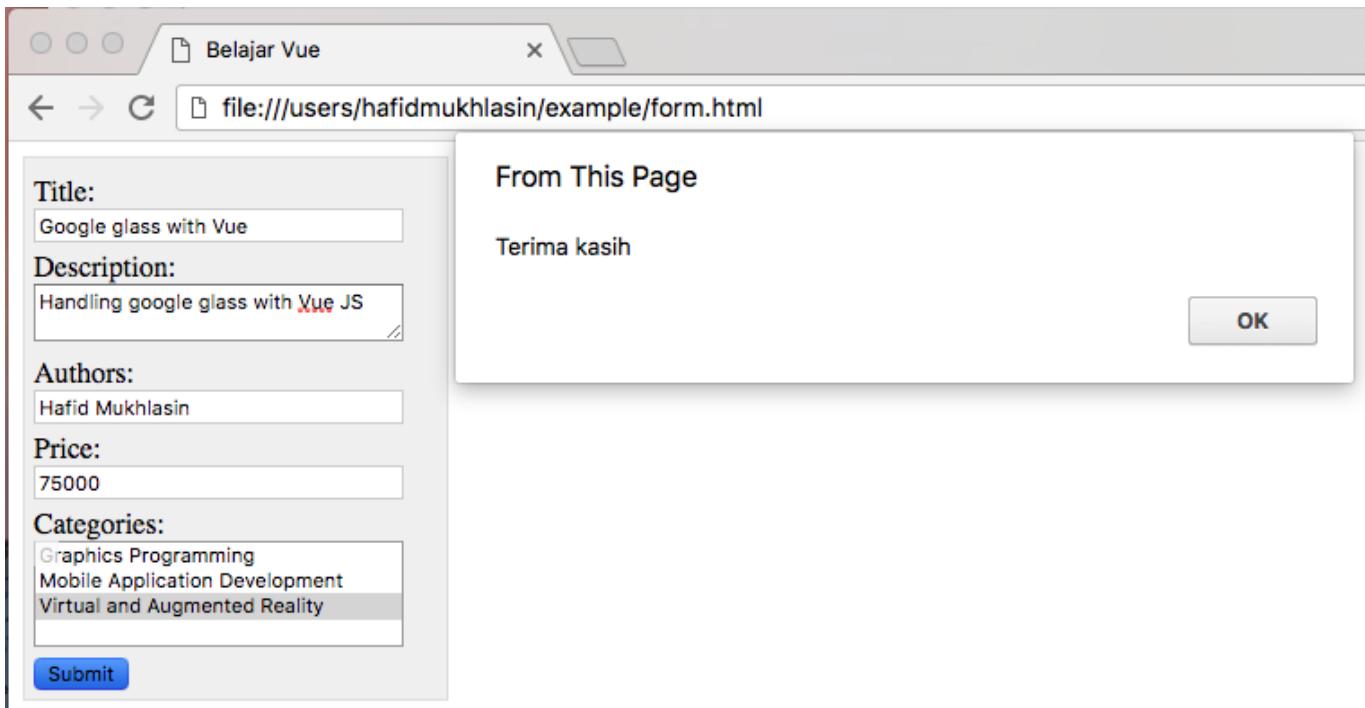
```
<form @submit.prevent="submit"...
```

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

```
<style>
form {
    border: 1px solid #ddd;
    padding: 5px;
    width: 225px;
    background: #efefef;
}
label {
    display: block;
    margin-top: 5px;
}

input, textarea, select, option {
    min-width: 200px;
}
</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 akan tidak (error = 0), jika tidak ada error maka ditampilkan pesan terima kasih dan kode untuk mengirim data tersebut ke server.

```
submitForm(event){
    let error = 0

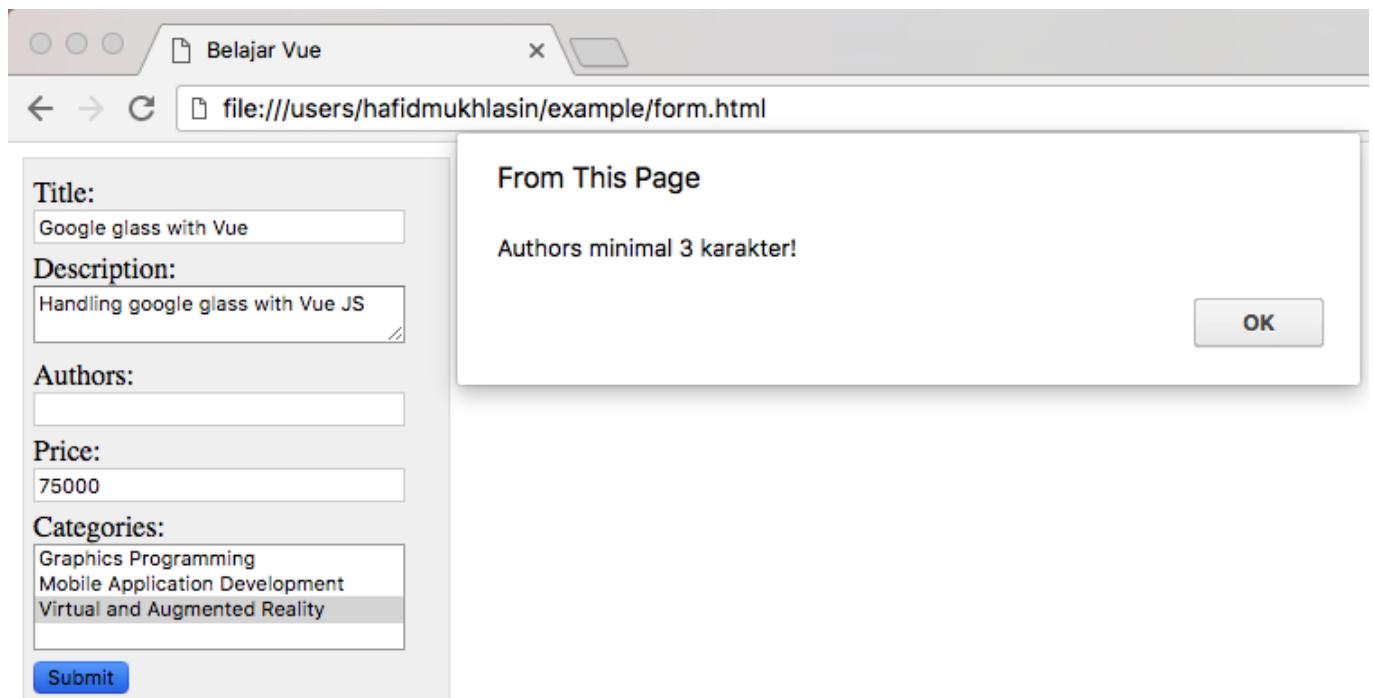
    if(this.title.length < 3){
        error++
        alert('Title minimal 3 karakter!')
    }
    else if(this.description.length > 500){
        error++
        alert('Description maximal 500 karakter!')
    }
    else if(this.authors.length < 3){
        error++
        alert('Authors minimal 3 karakter!')
    }
    else if(this.price < 0){
        error++
        alert('Price tidak boleh minus!')
    }
    else if(this.categories.length === 0){
```

```
        error++
        alert('Pilih minimal 1 category!')
    }

    if( error === 0 ){
        alert('Terima kasih telah mengisi data dengan benar!')
        // kirim data ke server
    }

    event.preventDefault()
}
```

Berikut hasilnya jika ada field yang tidak memenuhi minimal persyaratan



Dan berikut ini jika semua field telah memenuhi persyaratan.

The screenshot shows a browser window with a title bar 'Belajar Vue'. Below it is a URL bar with 'file:///users/hafidmukhlasin/example/form.html'. The main content is a form with the following fields:

- 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 (selected)

Below the form is a blue 'Submit' button. To the right of the form is a modal dialog box with the title 'From This Page' and the message 'Terima kasih telah mengisi data dengan benar!'. There is an 'OK' button in the bottom right corner of the modal.

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.

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

Melalui directive tersebut kemudian bisa kita akses dengan kode berikut.

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

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 konstruktur Vue menjadi sebagai berikut.

```
var vm = new Vue({
  el: '#app',
  data: {
```

```

        title: 'Google Glass with VueJS',
        description: 'Control Google Glass with VueJS',
        authors: 'Hafid Mukhasin',
        price: 75000,
        categories: [],
        options: [
            { text: 'Graphics Programming', value: '01' },
            { text: 'Mobile Application Development', value: '02' },
            { text: 'Virtual and Augmented Reality', value: '03' }
        ],
        errors: []
    },
    methods: {
        submitForm(event){
            this.errors = []
            if(this.title.length < 3){
                this.errors.push('Title minimal 3 karakter!')
                this.$refs.title.select()
            }
            if(this.description.length > 500){
                this.errors.push('Description maximal 500 karakter!')
                this.$refs.description.select()
            }
            if(this.authors.length < 3){
                this.errors.push('Authors minimal 3 karakter!')
                this.$refs.authors.select()
            }
            if(this.price < 0){
                this.errors.push('Price tidak boleh minus!')
                this.$refs.price.select()
            }
            if(this.categories.length === 0){
                this.errors.push('Pilih minimal 1 category!')
                this.$refs.categories.focus()
            }

            if( this.errors.length === 0 ){
                alert('Terima kasih telah mengisi data dengan benar!')
                // kirim data ke server
            }
        }
    }
})

```

Kemudian pada template, kita ubah menjadi sebagai berikut.

```

<form ref="formBook" @submit.prevent="submitForm($event)"
action="http://example.com/add-product" method="post">

<p v-if="errors.length">
    <b>Please correct the following error(s):</b>
    <ul>

```

```
<li v-for="error in errors">{{ error }}</li>
</ul>
</p>

<label>Title:</label>
<input name="title" ref="title" type="text" v-model="title">

<label>Description:</label>
<textarea name="description" ref="description" v-model="description">
</textarea>

<label>Authors:</label>
<input name="authors" ref="authors" type="text" v-model="authors">

<label>Price:</label>
<input name="price" ref="price" type="number" v-model.number="price">

<label>Categories:</label>
<select name="categories" ref="categories" v-model="categories"
multiple>
    <option v-for="option in options" :value="option.value">
        {{ option.text }}
    </option>
</select>

<label></label>
<input type="submit" value="Submit">
</form>
```

Kode di atas akan menghasilkan tampilan berikut.

The screenshot shows a browser window titled "Belajar Vue" displaying a local file "form.html". The page contains a form with validation errors. The errors are:

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

The form fields and their current values are:

- Title:** Google glass with Vue
- Description:** Handling google glass with Vue JS
- Authors:** (empty input field)
- Price:** 75000
- Categories:** Graphics Programming, Mobile Application Development, Virtual and Augmented Reality

A "Submit" button is visible at the bottom of the form.

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.

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

Berikut ini implementasi pada kasus kita.

```
if( this.errors.length === 0 ){
    alert('Terima kasih telah mengisi data dengan benar!')
    // persiapkan data
    let formData = new FormData()
    formData.append('title', this.title)
    formData.append('description', this.description)
    formData.append('authors', this.authors)
    formData.append('price', this.price)
    formData.append('categories', this.categories)

    // kirim data ke server
}
```

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.

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

Dengan sedikit modifikasi maka implementasi pada kasus kita menjadi sebagai berikut

```
// persiapkan data
let formBook = this.$refs.formBook
formData = new FormData(formBook);
// 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

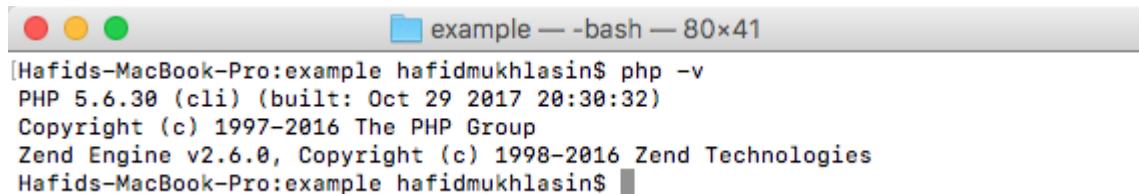
Setelah data di-*bundle* dalam satu objek formData, maka data siap untuk dikirim ke server. Pada bagian ini, kita akan mensimulasikan pengiriman data form ke server menggunakan PHP native. Oleh karenanya siapkan web server (nginx atau apache) serta PHP untuk dapat mencoba simulasi ini, atau bisa juga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

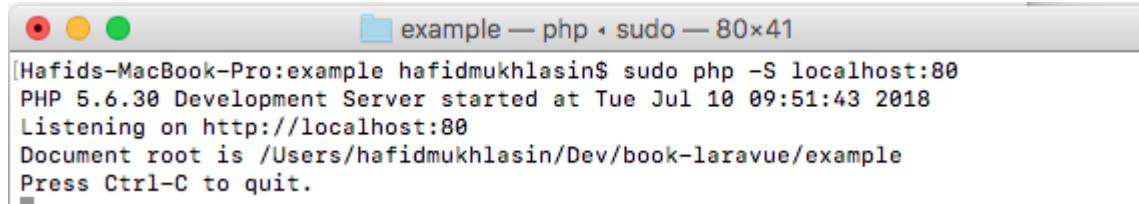
```
php -v
```



```
[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-laravue/example
Press Ctrl-C to quit.
[Hafids-MacBook-Pro:example hafidmukhlasin$ ]
```

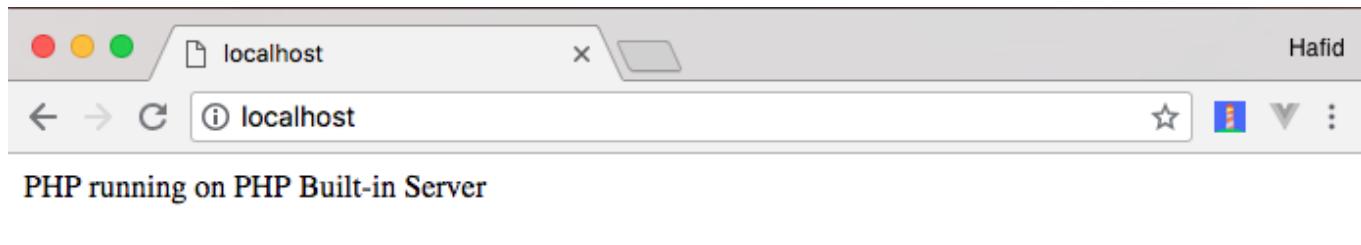
Catatan: 80 adalah nomer port dari webserver, kita bebas mengubahnya dengan nomer lain yang sedang tidak digunakan.

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.

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

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



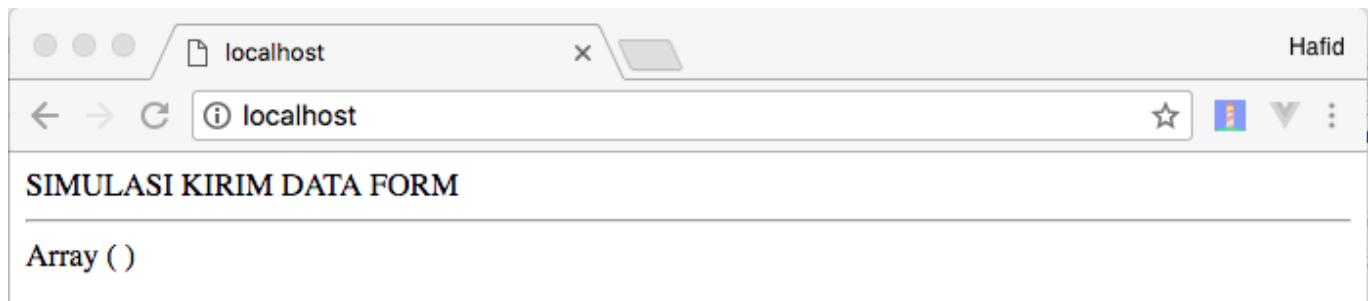
Jika berhasil, maka selanjutnya kita akan membuat kode PHP yang bertugas sebagai *endpoint* penerima data kirim dari form dan mengembalikan data tersebut dalam bentuk array. Berikut ini kodennya.

```
<?php
// untuk mencegah error akibat CORS
header("Access-Control-Allow-Origin: *");
header('Access-Control-Allow-Credentials: true');
header("Access-Control-Allow-Methods: GET, POST, OPTIONS");

echo "SIMULASI KIRIM DATA FORM <hr>";

// menampilkan data yang dikirimkan dengan method post
print_r($_POST);
```

Kemudian coba akses melalui browser.



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.

```
if( this.errors.length === 0 ){
    //alert('Terima kasih telah mengisi data dengan benar!')

    // persiapkan data
    let formBook = this.$refs.formBook
    formData = new FormData(formBook);

    // kirim data ke server
    let xhttp = new XMLHttpRequest() // create objek XMLHttpRequest

    // definisikan fungsi ketika terjadi perubahan state
```

```
xhttp.onreadystatechange = function() {
    // state ini menunjukkan data terkirim dan diterima server dengan
    baik
    if (this.readyState == 4 && this.status == 200) {
        // respon text dari server
        console.log(this.responseText)
    }
}
// sesuaikan dengan lokasi file index.php di lokasi komputer kamu
xhttp.open("POST", "http://localhost/index.php", true)

// bisa juga langsung nama filenya jika berada dalam satu folder yang
sama
// xhttp.open("POST", "index.php", true)

// kirim objek formData
xhttp.send(formData)
}
```

Mari kita ujicoba kode di atas pada browser.

The screenshot shows a browser window titled 'Belajar Vue' with the URL 'localhost/form-send-server.html'. The page displays a form with fields for Title, Description, Authors, Price, and Categories. The 'Categories' field contains three options: 'Graphics Programming', 'Mobile Application Development' (which is selected), and 'Virtual and Augmented Reality'. Below the form is a 'Submit' button. At the bottom of the browser, the developer tools' Console tab is active, showing the simulated data being sent:

```
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
)
```

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.

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

Kemudian pada kode Javascript-nya tambahkan kode berikut

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

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

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

Hasilnya.

The screenshot shows a web browser window titled "Belajar Vue". The address bar displays "localhost/form.html". The page content is a form for adding a book entry:

- Title:** Google glass with Vue
- Description:** Handling google glass with Vue JS
- Authors:** Hafid Mukhasin
- Price:** 75000
- Categories:** A dropdown menu with three options: Graphics Programming, Mobile Application Development (which is currently 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.

```

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.

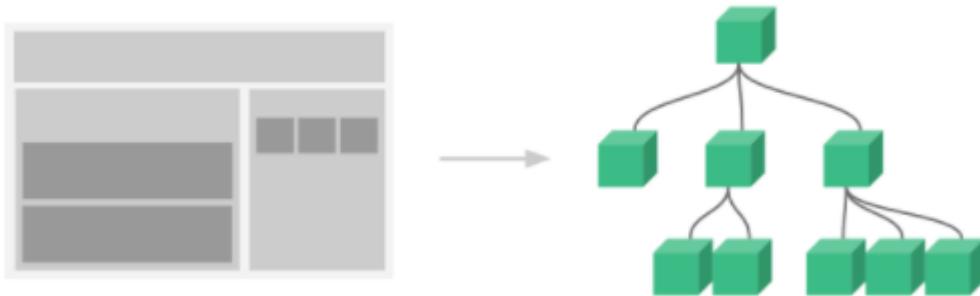
Ayoo lebih semangat lagi!

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 */ }).
```

```
Vue.component('hello-world', {
  data () {
    return {
      message: 'Hello world!'
    }
  },
  template: '<h1> {{ message }}</h1>'
})

var vm = new Vue({
  el: '#app',
})
```

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.

```
<div id="app">
  <hello-world></hello-world>
</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.

Component Naming

Terdapat dua format penamaan component pada Vue

1. kebab-case

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

```
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.

```
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

```
Vue.component('my-component-name', /* ... */)
```

Contoh kita memiliki tiga component Global dan dua objek Vue.

```
Vue.component('component-a', {
  template: `<p>Component A</p>`
})
Vue.component('component-b', {
  template: `<p>Component B</p>`
})
Vue.component('component-c', {
  template: `<p>Component C</p>`
})

new Vue({ el: '#app1' })
new Vue({ el: '#app2' })
```

Implementasi pada template

```
<div id="app1">
  <h1>Objek Vue 1</h1>
  <component-a></component-a>
  <component-b></component-b>
</div>

<div id="app2">
  <h1>Objek Vue 2</h1>
  <component-a></component-a>
  <component-c></component-c>
</div>
```

Hasilnya

Belajar Vue

file:///users/hafidmukhlasin/example/component.html

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

```
var ComponentA = {
  template: `<p>Component A</p>`
}

var ComponentB = {
  template: `<p>Component B</p>`
}

var ComponentC = {
  template: `<p>Component C</p>`
}
```

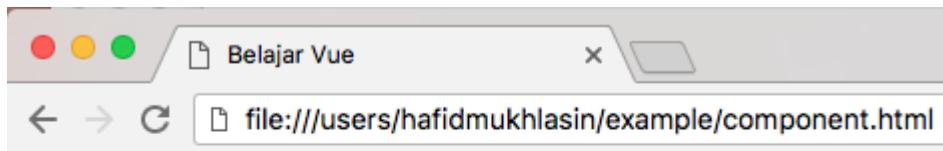
Adapun untuk meregister pada objek vue menggunakan properti **components**.

```
new Vue({
  el: '#app',
  components: {
    'component-a': ComponentA,
    'component-b': ComponentB,
    'component-c': ComponentC
  }
})
```

Pada template

```
<div id="app">
  <component-a></component-a>
  <component-b></component-b>
  <component-c></component-c>
</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.

```
var ComponentA = {
  template: `<p>Component A</p>`
}

new Vue({
  el: '#app1',
  components: {
    'component-a': ComponentA,
  }
})

new Vue({
  el: '#app2'
})
```

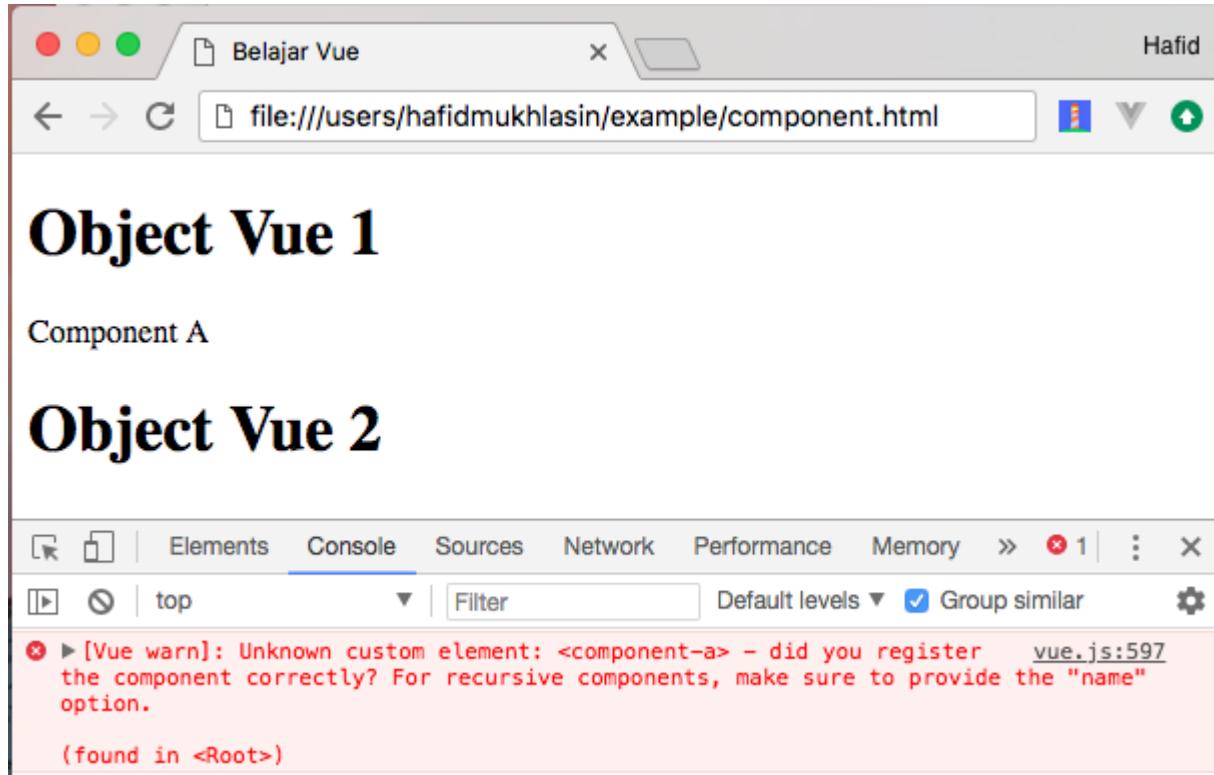
Sedangkan pada template objek Vue kedua juga menggunakan component tersebut

```
<div id="app1">
  <h1>Object Vue 1</h1>
  <component-a></component-a>
</div>

<div id="app2">
  <h1>Object Vue 2</h1>
```

```
<component-a></component-a>  
</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.

```
data: {  
    message: 'Hello world'  
}  
  
// menjadi  
data () {  
    return {  
        message: 'Hello world!'  
    }  
}
```

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.

```
<div id="app">
  <hello-world></hello-world>
  <hello-world></hello-world>
  <hello-world></hello-world>
</div>
```

Component Lanjutan

Passing Data To Component

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

Contoh, kita mempunyai component book

```
var BookComponent = {
  data () {
    return {
      classCard: 'card'
    }
  },
  props: [ 'title', 'description', 'image' ],
  template : `
    <div :class="classCard">
      <h3>{{ title }}</h3>
      
      <p v-html="description"></p>
    </div>
  `
}

new Vue({
  el: '#app',
  components: {
    'book': BookComponent,
  }
})
```

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

Pada template bisa kita definisikan sebagai berikut.

```
<div id="app">
  <book
    title="C++ High Performance"
    description="Write code that scales across CPU registers, multi-
    core, and machine clusters">
```

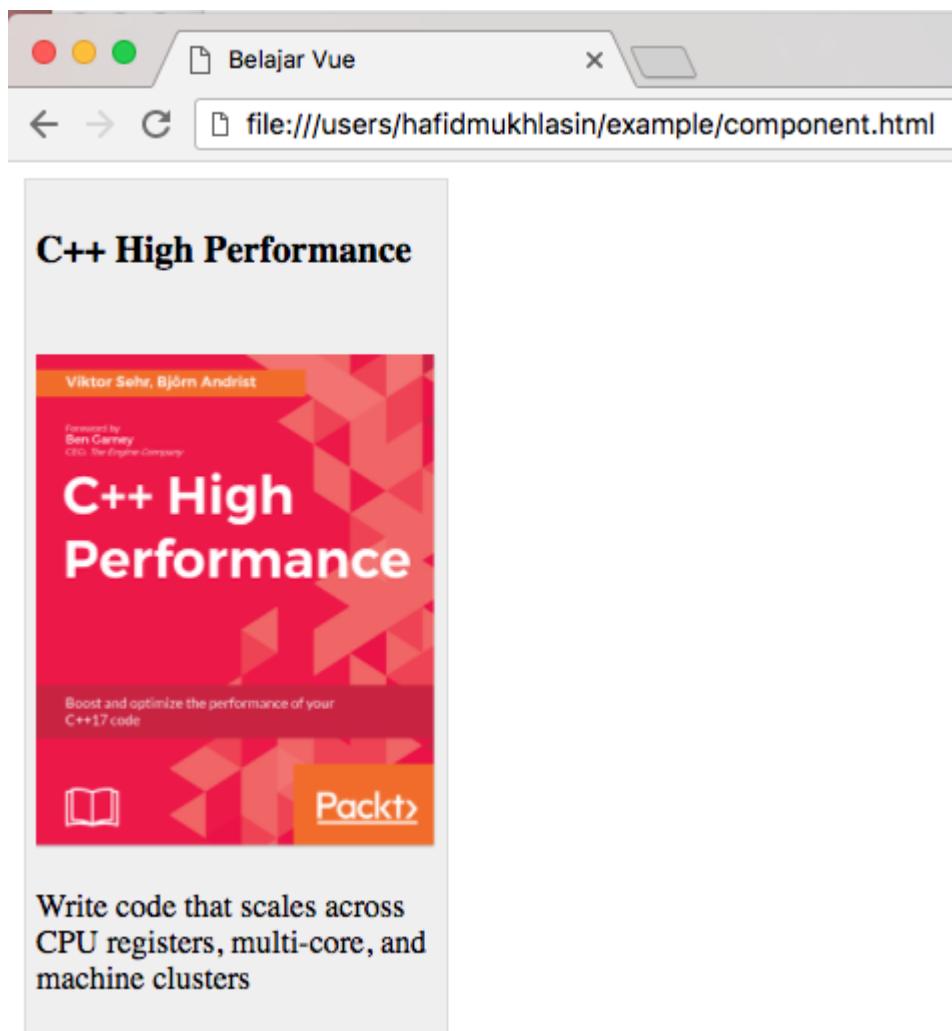
```
    image = "c++-high-performance.png"
    >
</book>
</div>
```

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

```
.card {
    background: #efefef;
    border: 1px solid #ddd;
    margin-right: 5px;
    padding: 5px;
    width: 200px;
    float: left;
}

h3{
    min-height: 45px;
}
```

Hasilnya.



Nilai dari variable title, description, dan image dipassing dari luar component dan digunakan di dalam template component seperti layaknya variabel 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 contoh ini kita masih akan menggunakan component book,

```
// deklarasi component book, lihat contoh sebelumnya
// ...

// deklarasi object vue dengan data books
var vm = new Vue({
  el: '#app',
  components: {
    'book': BookComponent,
  },
  data: {
    books : [
      {
        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'
      },
      {
        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'
      },
      {
        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'
      },
    ]
  }
})
```

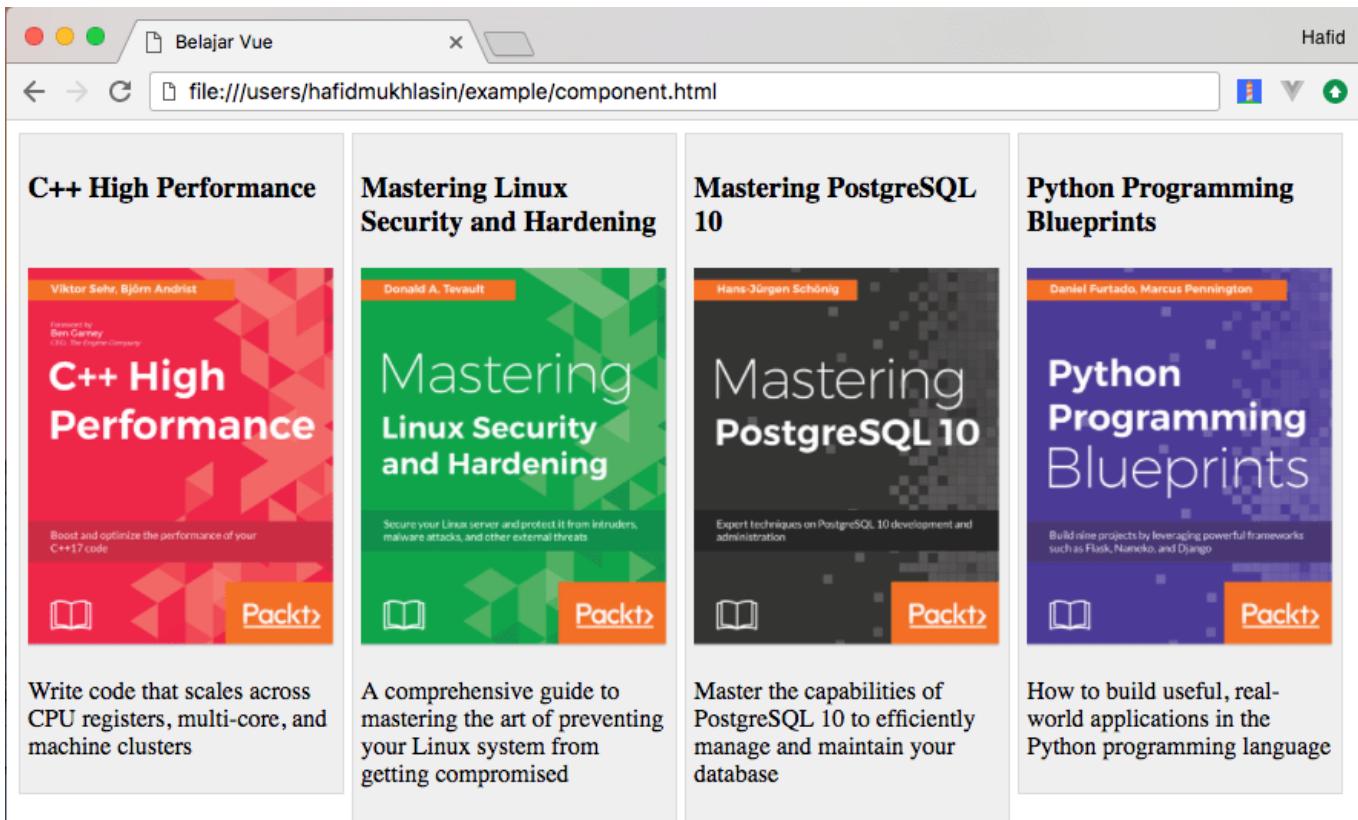
```
{  
    id: 102,  
    title: 'Python Programming Blueprints',  
    description: 'How to build useful, real-world applications  
in the Python programming language',  
    authors: 'Daniel Furtado, Marcus Pennington',  
    publish_year: 2017,  
    price: 75000,  
    image: 'python-programming-blueprints.png'  
},  
]  
}  
})
```

Pada template kita gunakan v-for untuk merender data books di atas.

```
<div id="app">  
  <book  
    v-for="book in books"  
    :key="book.id"  
    :title="book.title"  
    :description="book.description"  
    :image = "book.image"  
  >  
  </book>  
</div>
```

Semua atribut untuk props menggunakan 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 ubah tipe data yang kita kirimkan ke component melalui props, di mana sebelumnya teks biasa menjadi objek buku.

Deklarasi component book kita ubah menjadi seperti berikut.

```
var BookComponent = {
  data () {
    return {
      classCard: 'card'
    }
  },
  props: [ 'book' ],
  template : `
    <div :class="classCard">
      <h3>{{ book.title }}</h3>
      
      <p v-html="book.description"></p>
    </div>
  `
}
```

Tampilan template-nya cukup seperti berikut.

```
<div id="app">
  <book
    v-for="book in books"
    :key="book.id"
    :book="book"
  >
```

```
>
</book>
</div>
```

Update Data Parent From Component

Kebalikan dari *Passing Data To Component*, kita juga diizinkan untuk mengupdate data pada objek parent melalui component.

Contoh, pada objek Vue (parent) kita juga memiliki variabel *selectedBook*

```
var vm = new Vue({
  el: '#app',
  data: {
    books: [ /* ... */ ],
    selectedBook: '',
  },
  components: {
    'book': BookComponent,
  },
})
```

Pada template kita tambahkan custom event yang berfungsi mengubah variabel *selectedBook*, serta kita tampilkan variabel tersebut pada header sebagai berikut

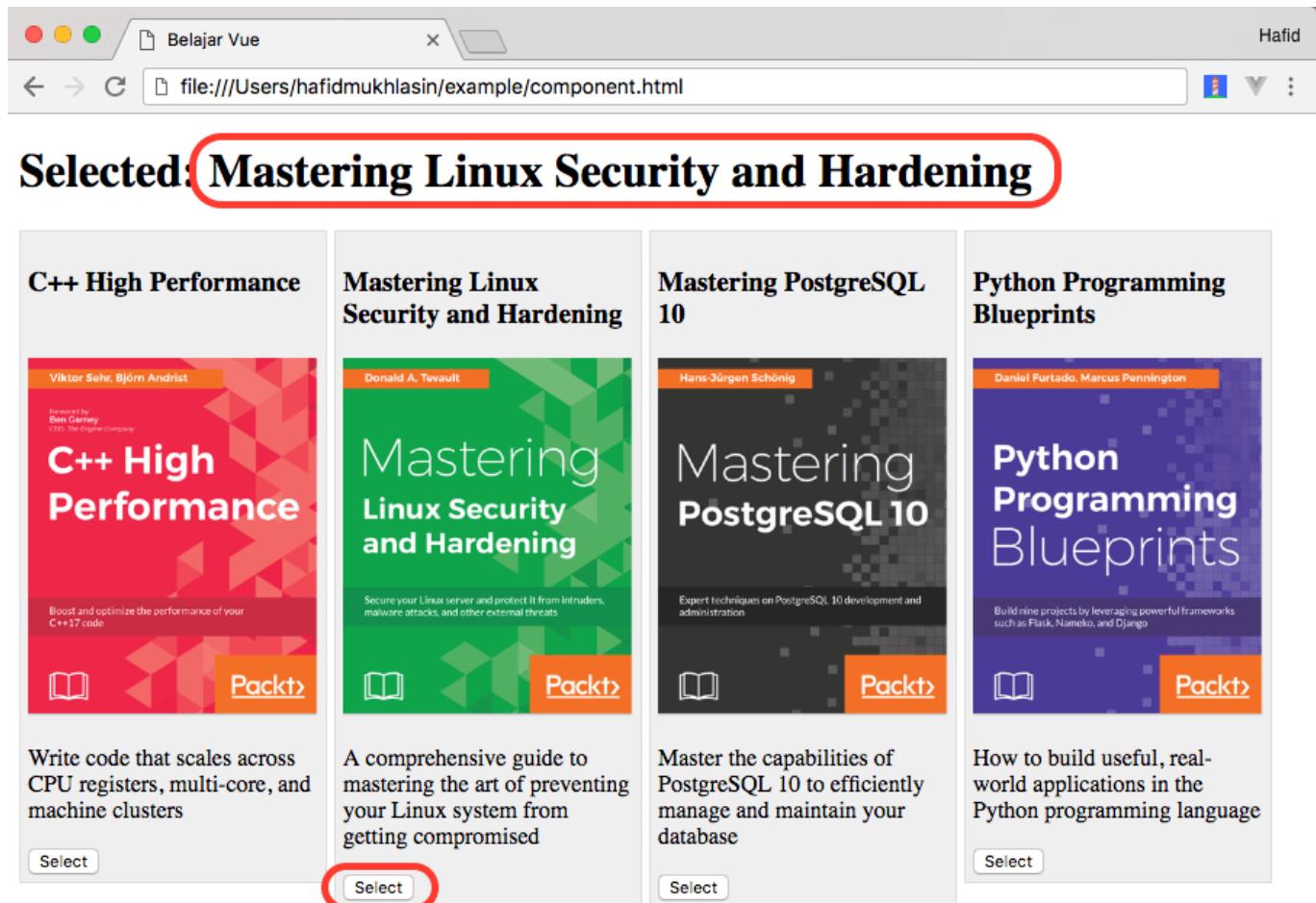
```
<div id="app">
  <h1>Selected: {{ selectedBook }}</h1>
  <book
    v-for="book in books"
    :key="book.id"
    :book="book"
    @selected="selectedBook = $event"
  >
  </book>
</div>
```

Kemudian pada template component, kita tambahkan button yang berfungsi mentrigger event *selected* yang kita definisikan di atas.

```
var BookComponent = {
  data () {
    return {
      classCard: 'card'
    }
  },
  props: [ 'book' ],
  template :
```

```
<div :class="classCard">
    <h3>{{ book.title }}</h3>
    
    <p v-html="book.description"></p>
    <button @click="$emit('selected', book.title)"> Select
</button>
</div>
.
.
}
```

Hasilnya sebagai berikut.



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.

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

setara dengan kode berikut.

```
<input
  :value="searchText"
  @input="searchText = $event.target.value"
>
```

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

```
<custom-input v-model="searchText"></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.

```
Vue.component('custom-input', {
  props: ['value'],
  template: `
    <input
      :value="value"
      @input="$emit('input', $event.target.value)"
    >
  `
})
```

Content Distribution with Slots

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

Untuk mengirimkan konten kepada component, kita bisa menggunakan elemen `slot`. Misalnya kita memiliki component `information`.

```
Vue.component('information', {
  template: `
    <div class="card">
      <strong>Informasi</strong>
      <hr>
      <slot></slot>
    </div>
  `
```

```
})
}

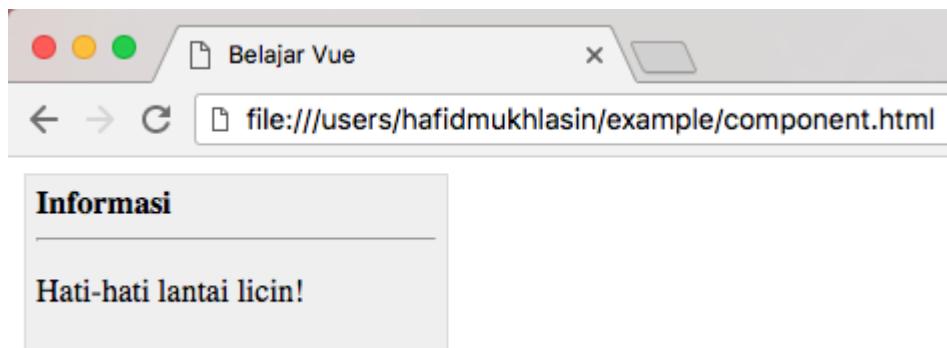
var vm = new Vue({
  el: '#app'
})
```

Kemudian component tersebut dapat kita panggil sebagai berikut.

```
<div id="app">
  <information>
    <p>Hati-hati lantai licin!</p>
  </information>
</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.

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.

```
export const BookComponent = {
  data () {
    return {
      classCard: 'card'
    }
  },
  props: [ 'book' ],
```

```
template :`  
    <div :class="classCard">  
        <h3>{{ book.title }}</h3>  
          
        <p v-html="book.description"></p>  
        <button @click="$emit('selected', book.title)"> Select  
</button>  
    </div>  
,  
`  
}
```

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

```
<script type="module">  
  
import { BookComponent } from './BookComponent.js'  
  
var vm = new Vue({  
    el: '#app',  
    components: {  
        'book': BookComponent,  
    },  
    data: {  
        selectedBook: '',  
        books: [  
            /* ... */  
        ]  
    }  
})  
</script>
```

Ada dua bagian penting pada kode di atas yaitu

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

Mari kita lihat hasilnya.

The screenshot shows a web browser window titled "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 displayed: "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 web browser window titled "Belajar Vue". The address bar contains the URL "localhost/component.html". The page displays four book components in a grid:

- C++ High Performance** by Viktor Sehr, Björn Andrist. Cover image shows a red geometric pattern. Description: "Boost and optimize the performance of your C++17 code". Call-to-action button: "Select".
- Mastering Linux Security and Hardening** by Donald A. Tevault. Cover image shows a green geometric pattern. Description: "Secure your Linux server and protect it from intruders, malware attacks, and other external threats". Call-to-action button: "Select".
- Mastering PostgreSQL 10** by Hans-Jürgen Schönig. Cover image shows a dark gray geometric pattern. Description: "Expert techniques on PostgreSQL 10 development and administration". Call-to-action button: "Select".
- Python Programming Blueprints** by Daniel Furtado, Marcus Pennington. Cover image shows a purple geometric pattern. Description: "Build nine projects by leveraging powerful frameworks such as Flask, Nameko, and Django". Call-to-action button: "Select".

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, there are four book covers displayed in a grid:

- C++ High Performance** by Viktor Sehr, Björn Andrist. Description: Write code that scales across CPU registers, multi-core, and machine clusters. Call-to-action button: 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 button: 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 button: 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 button: Select.

Selected: Mastering Linux Security and Hardening

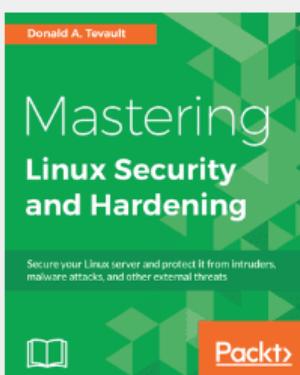
C++ High Performance



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

Select

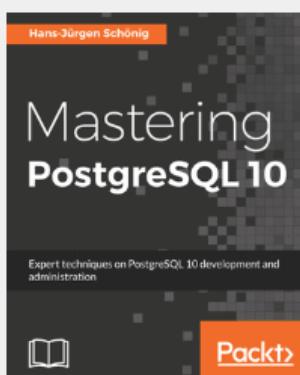
Mastering Linux Security and Hardening



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

Select

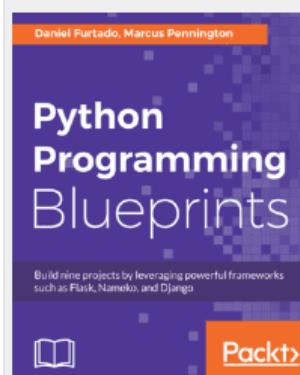
Mastering PostgreSQL 10



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

Kita bisa memuat suatu component yang sudah diregister secara dinamis.

Misalnya kita memiliki dua component yaitu list dan detail.

```
var list = {
  template: `
    <div class="card">
      <strong>Bahasa Pemrograman</strong>
      <ul>
        <li>Javascript</li>
        <li>PHP</li>
        <li>Java</li>
      </ul>
    </div>
  `

  var detail = {
    template: `
```

```
<div class="card">
    <strong>PHP</strong>
    <p>
        PHP adalah singkatan dari PHP Hypertext Preprocessor.
    </p>
</div>
` 

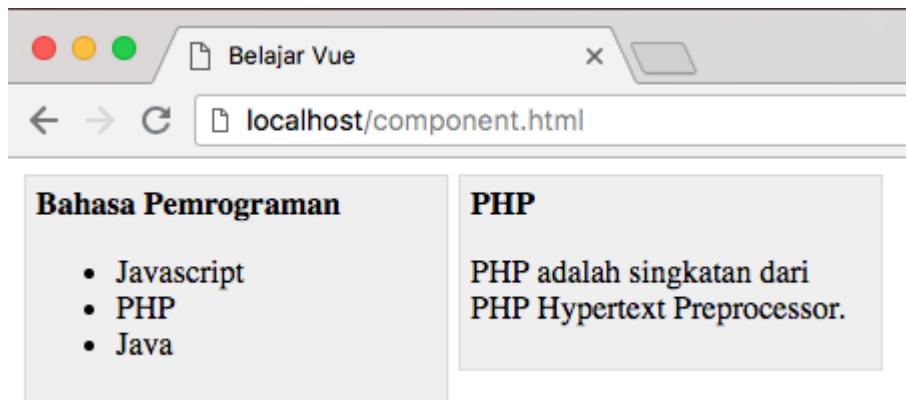
}

var vm = new Vue({
    el: '#app',
    components: {
        'list': list,
        'detail': detail,
    },
})
```

Lalu pada template kita tampilkan semua.

```
<div id="app">
    <list></list>
    <detail></detail>
</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.

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

Baik, pada objek Vue, tambahkan variabel `currentComponent`, seperti berikut ini.

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

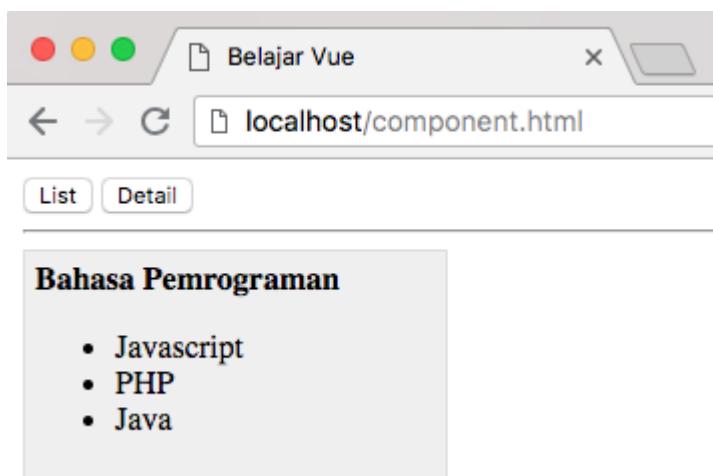
```
data: {
    currentComponent: 'list'
},
components: {
    'list': list,
    'detail': detail,
},
})
```

Lalu pada template ubah menjadi.

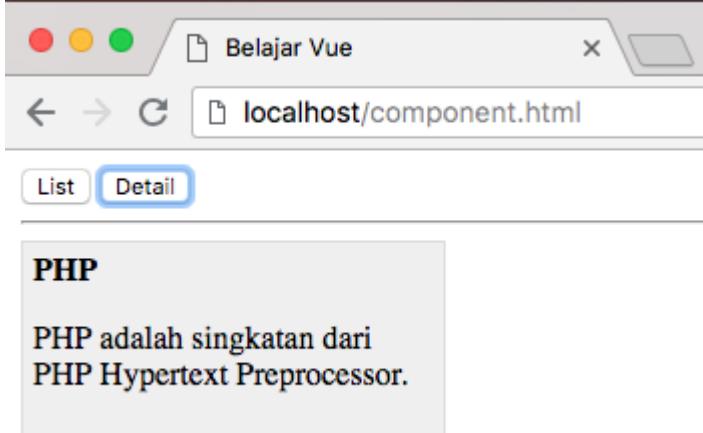
```
<div id="app">
<button @click="currentComponent = 'list'">List</button>
<button @click="currentComponent = 'detail'">Detail</button>
<hr>
<component :is="currentComponent"></component>
</div>
```

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 pergantianya terasa lembut dan menarik.

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

```
<div id="app">
  <button @click="currentComponent = 'list'">List</button>
  <button @click="currentComponent = 'detail'">Detail</button>
  <hr>
  <transition name="fade" mode="out-in">
    <component :is="currentComponent"></component>
  </transition>
</div>
```

Kemudian tambahkan style css berikut.

```
.fade-enter-active, .fade-leave-active {
  transition: opacity .5s;
}
.fade-enter, .fade-leave-to /* .fade-leave-active below version 2.1.8 */ {
  opacity: 0;
}
```

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 menggunakaninya tersebut.

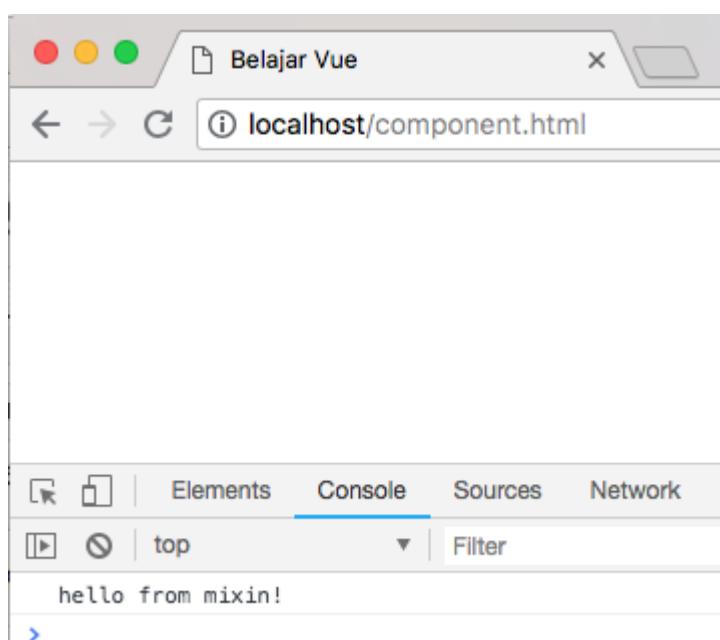
Contoh definisi mixins.

```
var MixinHello = {
  created: function () {
    this.hello()
  },
  methods: {
    hello: function () {
      console.log('hello from mixin!')
    }
  }
}
```

Defini tersebut jika digunakan pada Vue objek seperti berikut.

```
var vm = new Vue({
  el: '#app',
  mixins: [
    MixinHello
  ]
})
```

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.

```
// inject a handler for `text` custom option
Vue.mixin({
  created: function () {
    let text = this.$options.text
    if (text) {
      console.log(text)
    }
  }
})

new Vue({
  text: 'hello!'
})
// => "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.

```
MyPlugin.install = function (Vue, options) {
  // 1. add global method or property
  Vue.myGlobalMethod = function () {
    // something logic ...
  }

  // 2. add a global asset
  Vue.directive('my-directive', {
    bind (el, binding, vnode, oldVnode) {
      // something logic ...
    }
  })
}
```

```
}

...
})

// 3. inject some component options
Vue.mixin({
  created: function () {
    // something logic ...
  }
  ...
})

// 4. add an instance method
Vue.prototype.$myMethod = function (methodOptions) {
  // something logic ...
}
}
```

Menggunakan Plugin

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

```
// calls `MyPlugin.install(Vue)`
Vue.use(MyPlugin)

// atau jika ada options
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@3.0.1/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.

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

Pada template, kita gunakan dua elemen atau component routing yaitu **router-link** yang berfungsi menggenerate menu link, dan **router-view** yang berfungsi sebagai tempat penampung component yang ditampilkan. Ketika suatu link diklik dan menghasilkan URL route maka Vue Router akan mengecek apakah ada route yang sesuai, jika ada maka component akan dimuat didalam **router-view** (mirip konsep dynamic component). Berikut ini kode pada template.

```
<div id="app">
  <p>
    <router-link to="/">Home</router-link>
    <router-link to="/about">About</router-link>
  </p>

  <router-view></router-view>
</div>
```

Adapun kode Javascriptnya sebagai berikut.

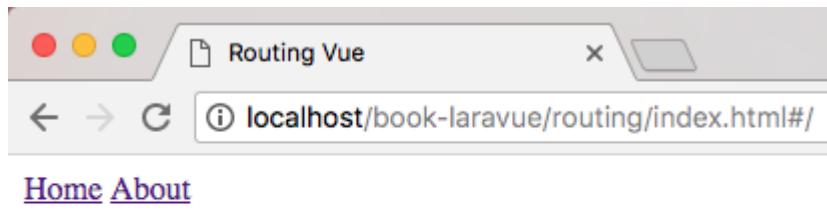
```
// definisikan konfigurasi component
const Home = { template: '<div>Halaman Home</div>' }
const About = { template: '<div>Halaman About</div>' }

// mapping route path dengan componentnya
const routes = [
  { path: '/', component: Home, alias: '/home' },
  { path: '/about', component: About }
]

// register routes pada objek router
const router = new VueRouter({
  routes // short `routes: routes`
})

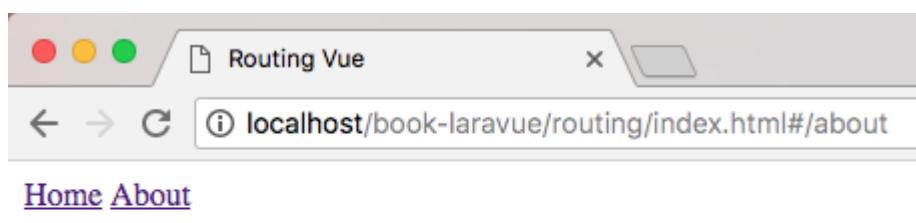
// register objek router pada objek Vue
var vm = new Vue({
  el: '#app',
  router,
})
```

Mari kita ujicoba pada browser.



Halaman Home

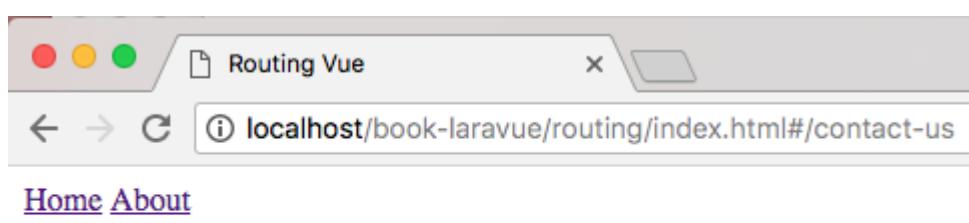
Pada saat pertama kali diakses maka routing pada URL address akan mengarah ke `/` yang telah kita mapping dengan component Home. Oleh karenanya, ketika pertama kali dibuka maka component Home akan tampil di router-view. Adapun kemudian, ketika link About diklik maka router-view akan menampilkan component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



Halaman About

Mudah sekali bukan?

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



Maka akan tidak 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.

```
const routes = [
  { path: '/', component: Home, alias: '/home' },
  { path: '/about', component: About },
  { path: '*', redirect: '/' }
]
```

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. Jika jumlah data buku sangat banyak maka tentu cukup melelahkan jika kita harus definisikan satu persatu.

Oleh karena itu, Vue Router menyediakan cara untuk mengatasi hal ini.

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

Kemudian pada objek Vue atau pada component Book dapat kita akses id tersebut dengan kode `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.

Component BooksComponent

Component ini berfungsi menampilkan data 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`

```
export const BooksComponent = {
  data () {
    return {
      books: [
        {
          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'
        },
        {
          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'
    },
    {
        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'
    },
    {
        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'
    },
]
}
},
template: `
<div>
    <h1>Daftar Buku</h1>
    <ul>
        <li v-for="book of books">
            <router-link :to="'/book/' + book.id">
                {{ book.title }}
            </router-link>
        </li>
    </ul>
</div>
` ,
}

```

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/B00K_ID`

Pada file utama `index.html`, import component BooksComponent,

```

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

```

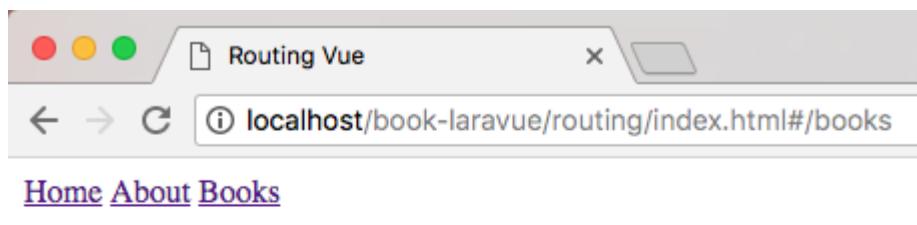
Daftarkan component ini pada mapping routing.

```
const routes = [
  { path: '/', component: Home, alias: '/home' },
  { path: '/about', component: About },
  { path: '/books', component: BooksComponent },
  { path: '*', redirect: '/' }
]
```

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

```
<div id="app">
  <router-link to="/">Home</router-link>
  <router-link to="/about">About</router-link>
  <router-link to="/books">Books</router-link>
  <hr>
  <router-view></router-view>
</div>
```

Lihat hasilnya. Maka akan tampil link daftar judul buku



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-laravue/routing/index.html#/book/101> maka akan diredirect ke component Home, karena path book tidak sesuai dengan salah satu dari daftar mapping routes.

Component BookComponent

Component BooksComponent menampilkan link daftar judul buku yang apabila diklik link tersebut maka kita inginnya akan memuat component BookComponent yang menampilkan data detail buku sesuai dengan link yang diklik. Misalnya jika diklik link buku **Mastering PostgreSQL 10** maka akan menampilkan detail dari buku tersebut.

Buat file BookComponent.js.

```

export const BookComponent = {
  data () {
    return {
      books: [
        {
          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'
        },
        {
          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'
        },
        {
          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'
        },
        {
          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'
        },
      ],
    }
  },
  computed: {
    book() {
      return this.books.filter((book)=>{
        return book.id === parseInt(this.$route.params.id)
      })
    }
  }
}

```

```

        })[0]
    }
},
template: `
<div v-if="book">
    <h1>Buku {{ book.title }}</h1>
    <ul>
        <li v-for="(num, value) of book">
            {{ num + ' : ' + value }} <br>
        </li>
    </ul>
</div>
` 
}

```

Pada kode di atas, data books yang berformat list di filter pada properti computed dengan cara membandingkan `book.id` dan `this.$route.params.id`. Dari mana varaiel ini `this.$route.params.id`?

Pada file utama `index.html`, import component BookComponent,

```

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

```

Daftarkan component ini pada mapping routing.

```

const routes = [
    { path: '/', component: Home, alias: '/home' },
    { path: '/about', component: About },
    { path: '/books', component: BooksComponent },
    { path: '/book/:id', component: BookComponent },
    { path: '*', redirect: '/' }
]

```

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.

The screenshot shows a browser window with the title "Routing Vue". The address bar contains the URL "localhost/book-laravel/routing/index.html#/book/101". Below the address bar, there is a navigation menu with links for "Home", "About", and "Books". The main content area displays the title "Buku Mastering PostgreSQL 10" in a large, bold font.

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()

```
router.push( /* location */ )

// jika di dalam objek Vue atau component, tambahkan this.$
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.

```
// literal string path
router.push('/home')

// object
router.push({ path: '/home' })

// named route /user/123
router.push({ name: 'user', params: { userId: 123 } })

// with query, resulting in /register?plan=private
router.push({ path: 'register', query: { plan: 'private' } })
```

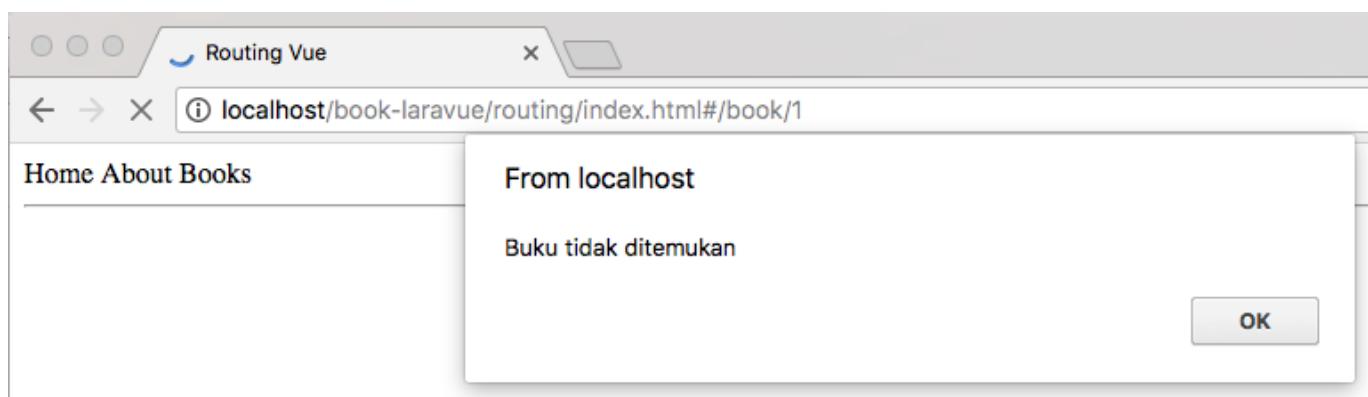
Catatan: jangan bingung dengan \$route vs \$router. \$route mengembalikan routing saat ini, sedangkan \$router adalah objek router yang bisa kita gunakan untuk menjalankan fungsi push() dan go().

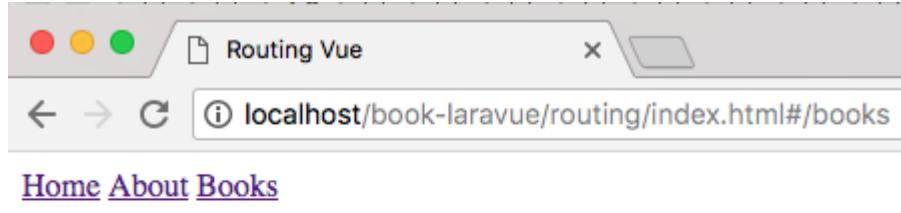
Untuk mencoba method ini, kita akan gunakan contoh sebelumnya, di mana jika pada detail buku atau component BookComponent ternyata buku dengan id yang dikirimkan tidak ditemukan maka halaman akan diredirect ke path books atau component BooksComponent.

```
computed: {
    book() {
        let book = this.books.filter((book)=>{
            return book.id === parseInt(this.$route.params.id)
        })

        // jika buku tidak ditemukan
        if (book.length==0){
            // redirect ke path books
            alert("Buku tidak ditemukan")
            this.$router.push("/books")
        }
        else{
            return book[0]
        }
    }
},
```

Mari kita uji coba, akses URL ini <http://localhost/book-laravel/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`.

```
// go forward by one record, the same as history.forward()
router.go(1)

// go back by one record, the same as history.back()
router.go(-1)

// go forward by 3 records
router.go(3)
```

Penamaan Routes

Untuk mengidentifikasi suatu route kita bisa menggunakan nama, dibanding menggunakan path-nya. Tambahkan key `name` pada mapping route

```
const routes = [
  { path: '/', component: Home, alias: '/home' },
  { path: '/about', component: About },
  { path: '/books', component: BooksComponent },
  { path: '/book/:id', name: 'book', component: BookComponent },
  { path: '*', redirect: '/' }
]
```

Untuk membuat link ke route tersebut kita bisa gunakan kode berikut:

```
<router-link :to="{ name: 'book', params: { bookId: 123 } }>Mastering
Vue</router-link>
```

Atau jika menggunakan method push kita bisa gunakan kode berikut.

```
router.push({ name: 'book', params: { bookId: 123 } })
```

Kedua contoh diatas akan mengarahkan halaman ke path /book/123.

Passing Props To Route Component

Kita bisa mengirimkan props kepada route component, 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 menganti `this.$route.params.id` menjadi `this.id`

```
export const BookComponent = {
  data () {
    return {
      books: [ /* */ ],
    }
  },
  props: ['id'],
  computed: {
    book() {
      let book = this.books.filter((book)=>{
        return book.id === parseInt(this.id)
      })
      ...
    }
  },
  ...
}
```

Kemudian pada mapping route, tambahkan key props yang bernilai true

```
const routes = [
  { path: '/', component: Home, alias: '/home' },
  { path: '/about', component: About },
  { path: '/books', component: BooksComponent },
  { path: '/book/:id', name: 'book', component: BookComponent, props: true },
  { path: '*', redirect: '/' }
]
```

Transitions Effect

Sebagaimana bahasan pada component dynamic, 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

```
<transition name="slide" mode="out-in">
  <router-view></router-view>
</transition>
```

Tambahkan sedikit css berikut.

```
.fade-enter-active, .fade-leave-active {
  transition: opacity .5s;
}
.fade-enter, .fade-leave-to /* .fade-leave-active below version 2.1.8 */ {
  opacity: 0;
}
```

Dan lihatlah keajaibannya 😊.

Bisa juga kita definisikan efek transisi ini per component. Caranya, bungkus template pada component dengan transition.

```
const Home = {
  template: `
    <transition name="slide">
      <div> Halaman Home </div>
    </transition>
  `
}
```

Navigation Guards

Vue Router memungkinkan kita menolak atau mengizinkan akses ke suatu route. 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('/'): 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

```
const router = new VueRouter({ ... })

// akan dijalankan sebelum route dituju
router.beforeEach((to, from, next) => {
    // ...
})

// akan dijalankan setelah route dituju
router.afterEach((to, from) => {
    // ...
})
```

Per Route

```
routes: [
    {
        path: '/home',
        component: Home,
        beforeEnter: (to, from, next) => {
            // ...
        }
    },
    // ...
]
```

Dalam Component

```
const Home = {
    template: `...`,
    beforeRouteEnter (to, from, next) {
        // dipanggil sebelum route dituju & sebelum component yang dituju itu dibuat
        // sehingga kita tidak bisa mengakses `this` component
    },
    beforeRouteUpdate (to, from, next) {
        // dipanggil ketika route yang merender component diubah, `this` bisa diakses
    },
    beforeRouteLeave (to, from, next) {
```

```
// dipanggil ketika akan meninggalkan current route
}
}
```

Prevent Leave Accident

Contoh implementasi penggunaan navigation guard ini adalah untuk mencegah pengguna keluar dari suatu halaman hanya karena salah klik (tidak sengaja), maka kita bisa menampilkan pesan konfirmasi "Apakah Anda yakin akan keluar?".

Kita akan mencoba melalui deklarasi dalam component menggunakan hook `beforeRouteLeave`.

Pada component BookComponent, tambahkan properti ini.

```
export const BookComponent = {
  data () {
    return {
      books: [ /* */ ],
    }
  },
  props: ['id'],
  computed: {
    /* */
  },
  template: `
    ...
  `,
  beforeRouteLeave (to, from , next) {
    const answer = window.confirm('Apakah Anda yakin akan keluar?')
    if (answer) {
      next()
    } else {
      next(false)
    }
  }
}
```

Kita bisa menampilkan dialog confirm, jika user setuju atau mengkonfirmasi dialog tersebut maka halaman akan diredirect keluar, jika tidak setuju maka akan tetap pada halaman tersebut.

Mari kita coba, akses component BookComponent, kemudian klik tombol back pada browser atau menu link lain (home misalnya)

From localhost

Apakah Anda yakin akan keluar?

Buku C++ High Performance

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

Authentication Route

Dengan menggunakan navigation guards, kita bisa mencegah user yang tidak memiliki hak akses untuk mengakses suatu route tertentu. Namun sebelumnya kita perlu definisikan pada mapping route, mana saja route yang hanya boleh diakses oleh misalnya user yang sudah login. Untuk melakukannya, Vue Router telah menyediakan key `meta`.

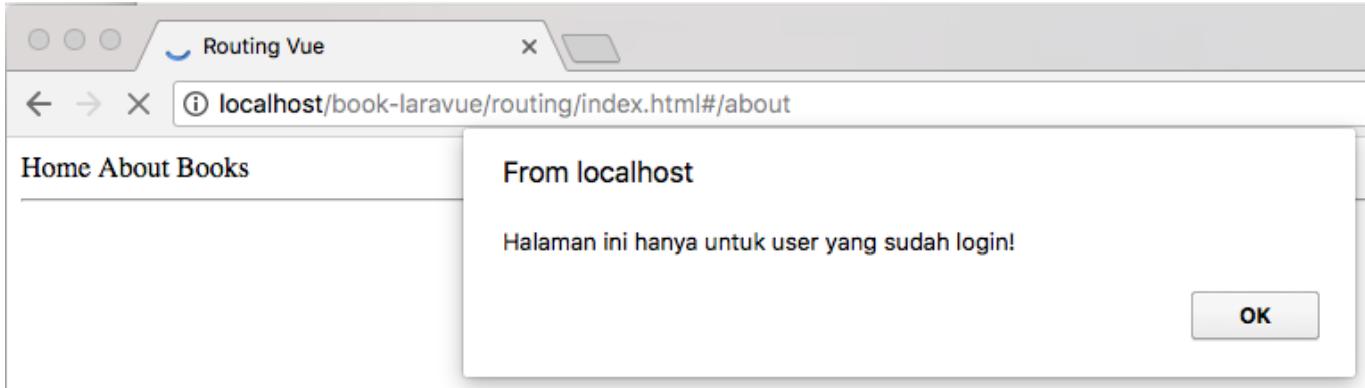
Misalnya, route atau halaman about hanya boleh diakses oleh user yang sudah login, maka..

```
const routes = [
  { path: '/', component: Home, alias: '/home' },
  { path: '/about', component: About, meta: { login: true } },
  { path: '/books', component: BooksComponent },
  { path: '/book/:id', name: 'book', component: BookComponent, props:
    true },
  { path: '*', redirect: '/' }
]
```

Kemudian navigation guard bisa kita definisikan secara global

```
router.beforeEach((to, from, next) => {
  if (to.matched.some(record => record.meta.login)) {
    alert('Halaman ini hanya untuk user yang sudah login!')
    next(false)
  }
  else{
    next()
  }
})
```

Mari kita coba dengan mengakses link about, hasilnya



Tentu saja kita bisa buat skenario, misalnya ketika user sudah login maka boleh mengakses, jika belum login maka route akan diredirect ke halaman login.

```
if (!auth.loggedIn()) {
    next('/login')
} else {
    next()
}
```

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

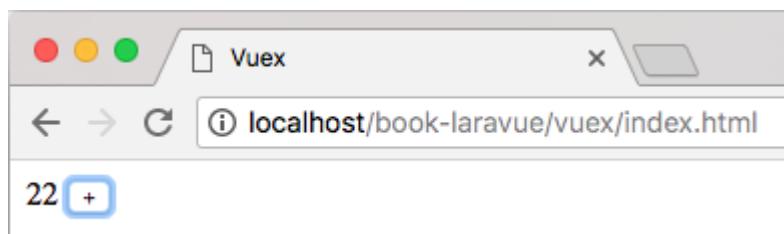
Pada bab ini kita akan belajar tentang state management menggunakan pustaka official Vuex.

Mengenal State Management

State management itu 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.

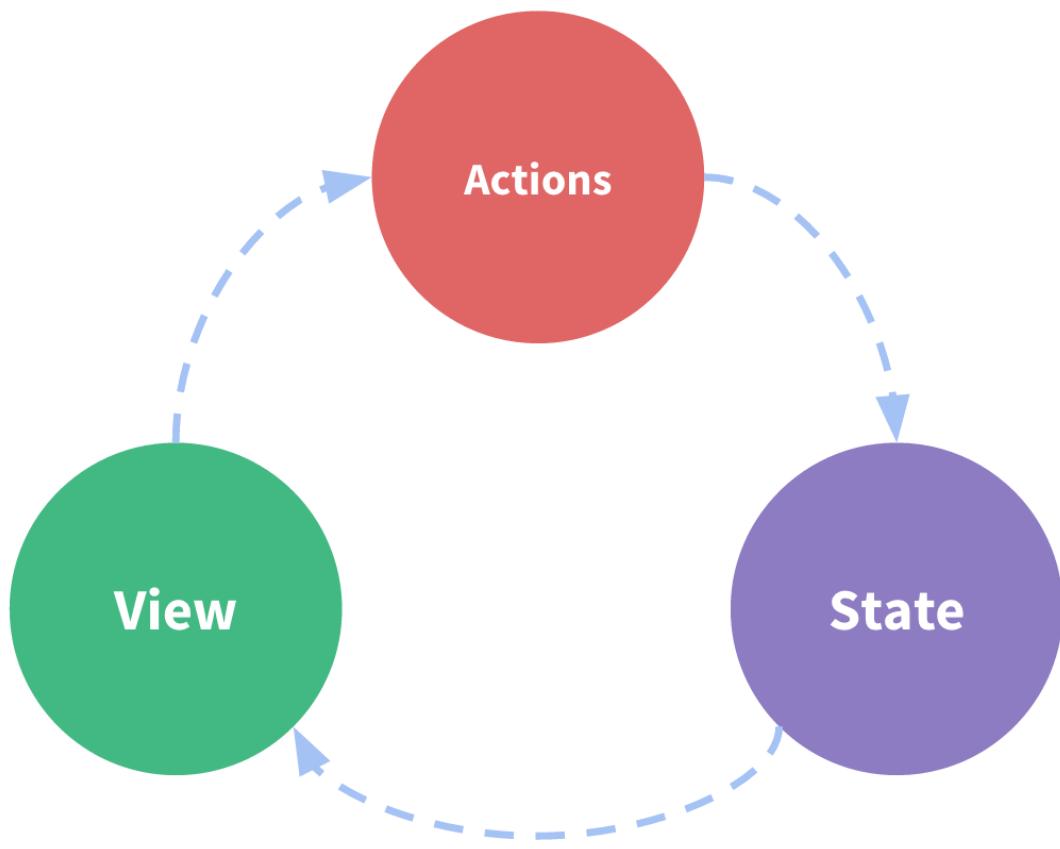
```
new Vue({
  el: '#app',
  // state
  data: {
    counter: 0
  },
  // view
  template: `
    <div>
      {{ counter }}
      <button @click="increment()"> + </button>
    </div>
  `,
  // actions
  methods: {
    increment () {
      this.counter++
    }
  }
})
```



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).



Terlihat sederhana, namun kalo kita sudah bermain dengan banyak component di mana setiap component menggunakan state yang sama sehingga mungkin beberapa view menggunakan state yang sama dan action dari view yang berbeda melakukan perubahan pada state yang sama.

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.

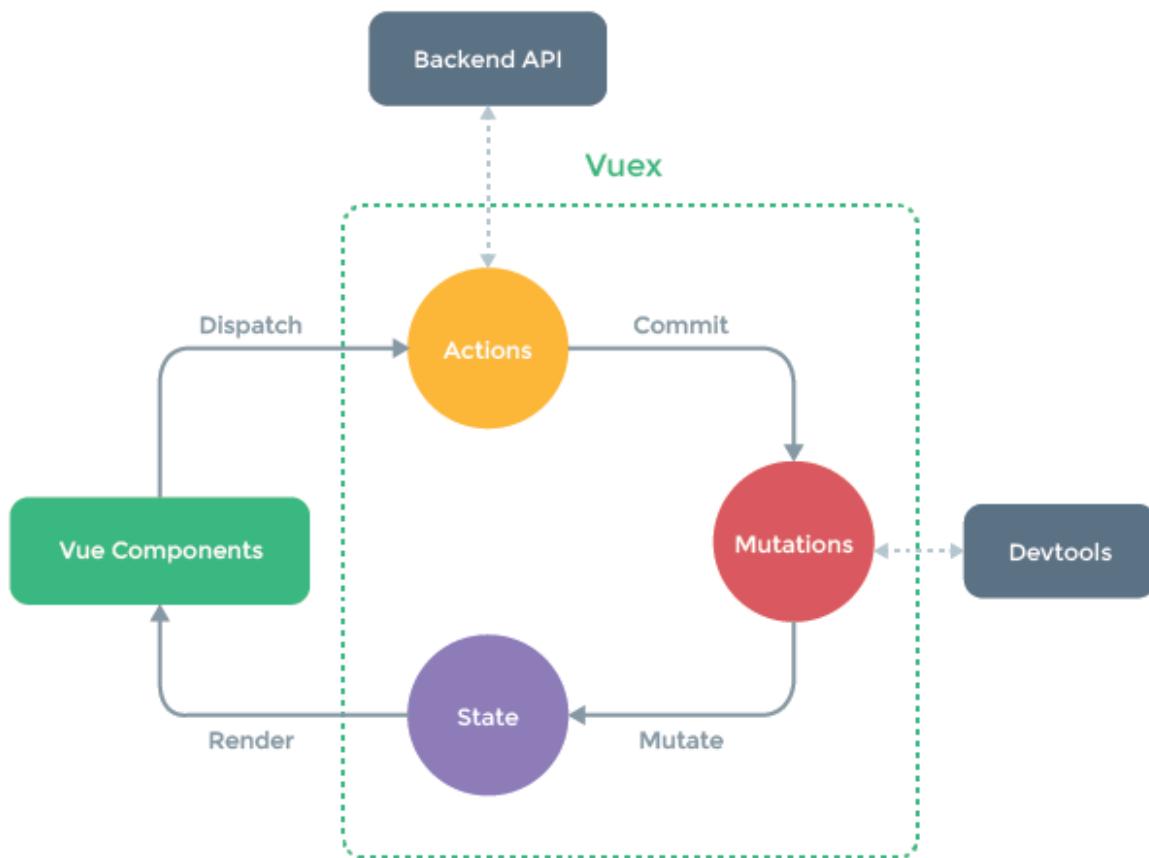
Oleh karena itu mengapa kita tidak menggunakan state terpusat yang bisa diakses dan dimanipulasi oleh semua component dalam aplikasi, sehingga perubahan state dapat lebih mudah dimonitoring dan dicontrol.

Jadi, jika aplikasi yang kita bangun menggunakan banyak component maka sebaiknya kita menerapkan konsep state management.

Pustaka State Management

Implementasi state management itu out of the box, pada dasarnya kita bisa pake pustaka apa saja misalnya redux, mobx, dll. Namun Vue telah membuat versi officialnya 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 official devtools extension.



Pustaka Vuex menangani dan terdiri dari 3 hal 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 bisa memicu rendering component.

Instalasi

Sebagai sebuah pustaka Javascript 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 ditarik 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@3.0.1/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`.

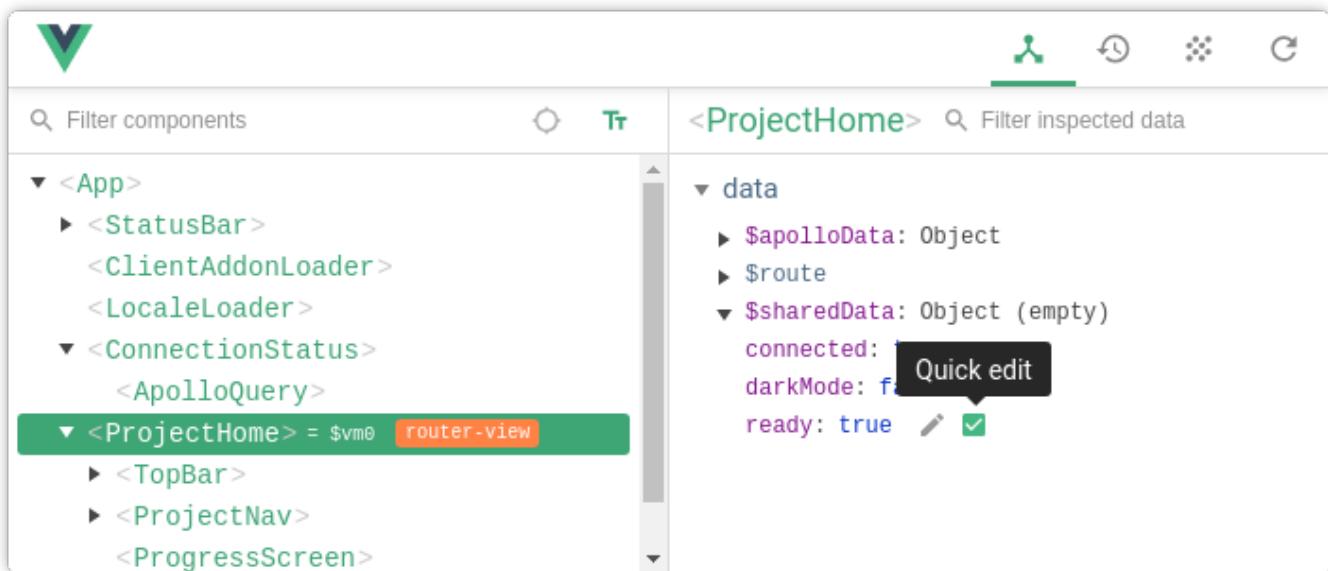
Pustaka Vuex ditambahkan setelah pustaka Vue.

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

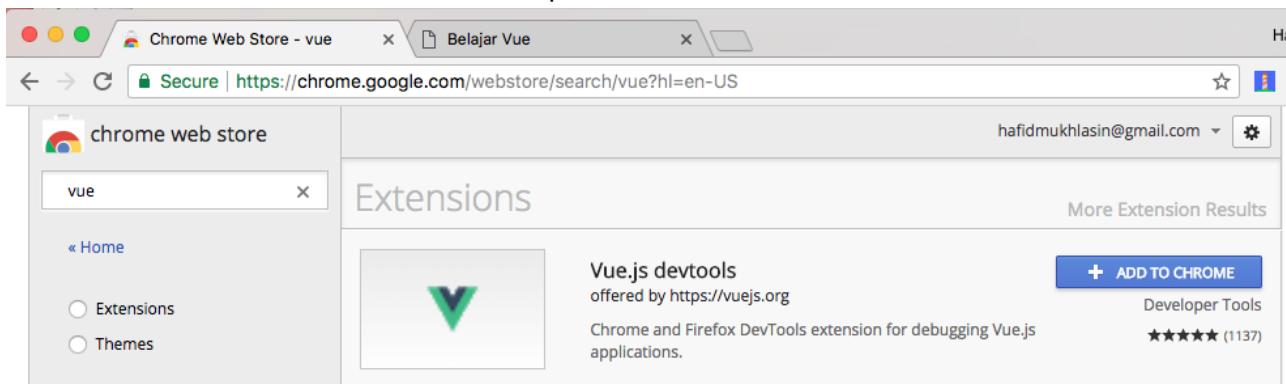
Dev Tools

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.

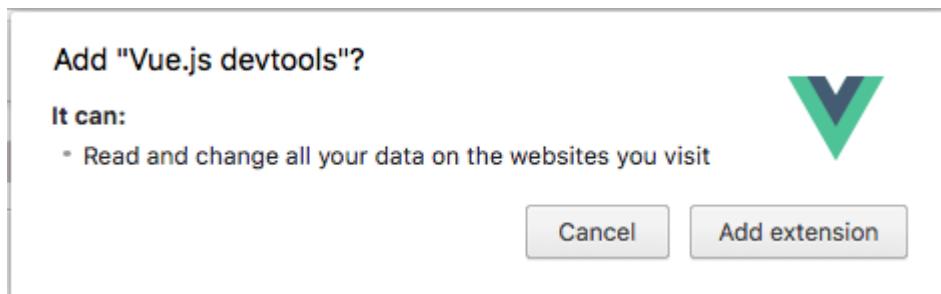


Untuk instalasi selengkapnya silakan cek tautan ini <https://github.com/vuejs/vue-devtools>, pada tutorial ini, penulis menggunakan browser chrome.

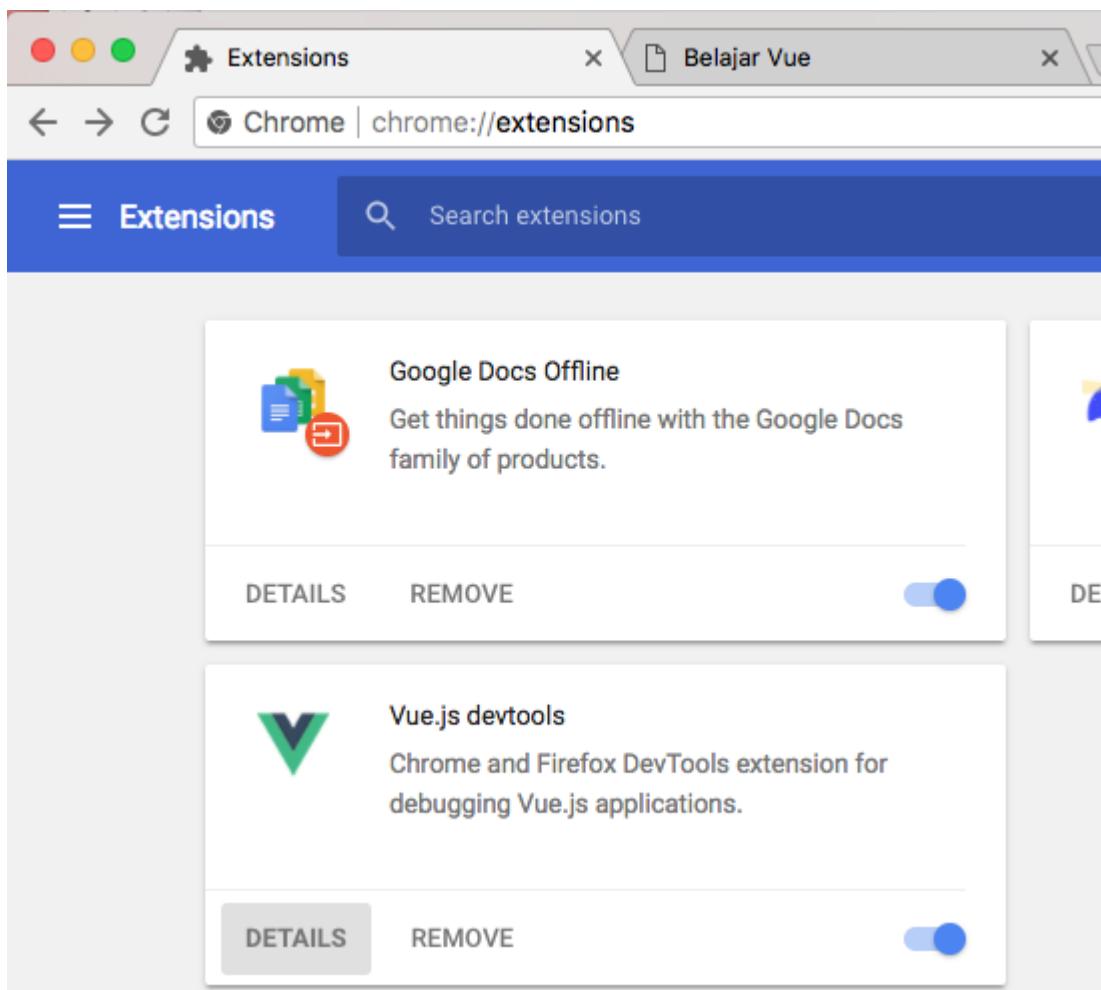
- Buka chrome web store, search Vue, maka pilih devtools



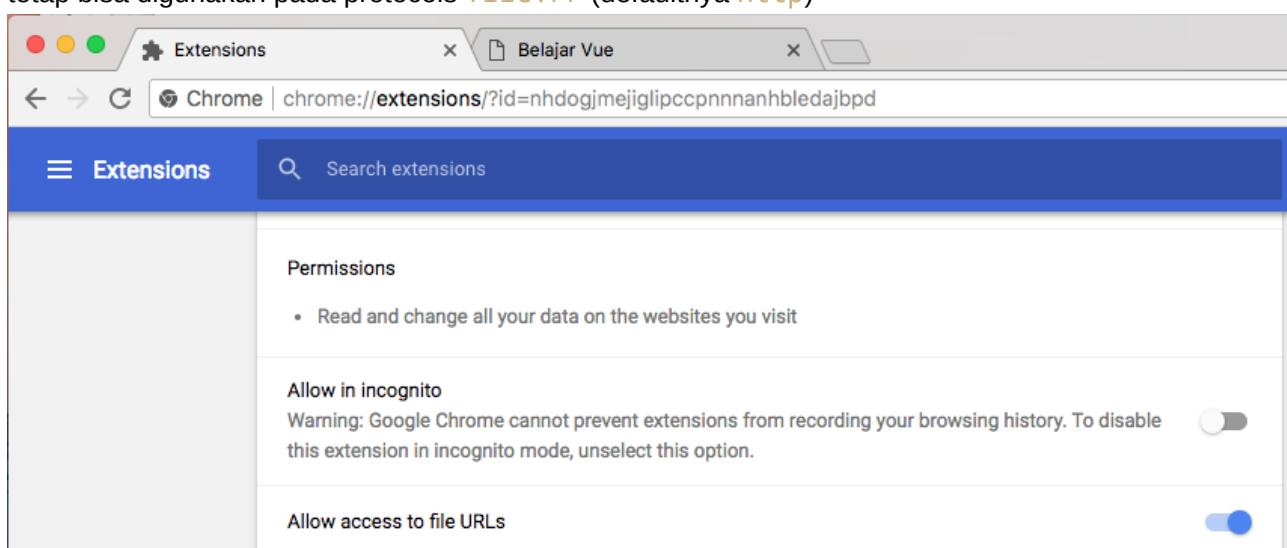
- Klik tombol Add to Chrome, lalu Add extension



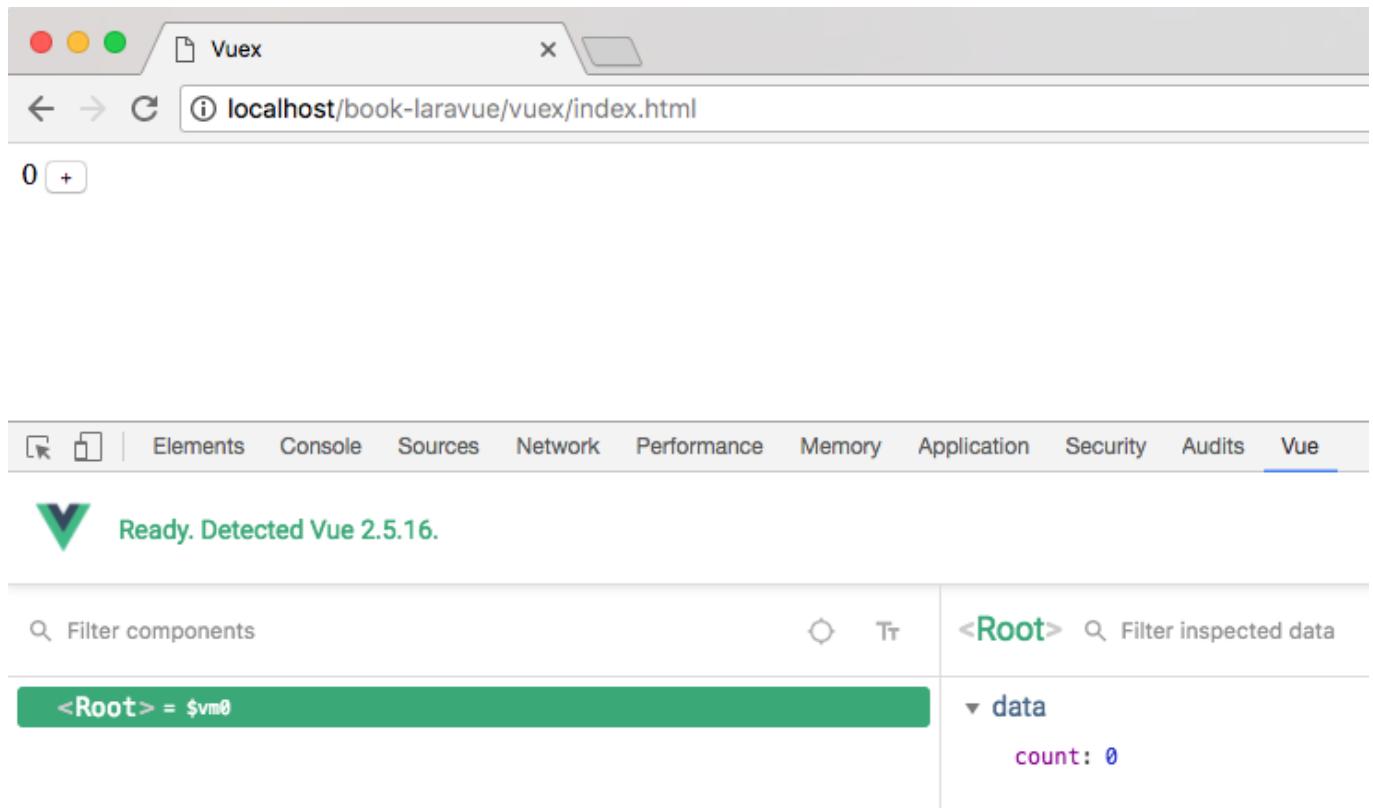
- Akses chrome://extensions untuk memastikan bahwa extension telah terinstalasi



- Pilih tombol Detail pada Vue.js devtools, kemudian aktifkan Allow access to file URLs tetapi bisa digunakan pada protocols file:// (defaultnya http)



Cara menggunakannya, pada Developer Tools - Browser, pilih menu Vue yaitu menu paling kanan



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 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 **counter** dan perubahannya adalah **increment** dari state.

Kedua, kita buat objek Vuex store berdasarkan poin pertama.

```
const store = new Vuex.Store({
  state: {
    counter: 0
  },
  mutations: {
    increment: state => state.counter++
  }
})
```

Berdasarkan kode store di atas, kita bisa menjalankan fungsi increment dengan meng-commit mutation increment `store.commit('increment')`, dan untuk mengaksesnya gunakan perintah `store.state.counter`.

Ketiga, kita implementasikan perubahan state pada objek Vue menggunakan store. Gunakan 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.

```
new Vue({
  el: '#app',
  // local state as computed
  computed: {
    counter(){
      return store.state.counter
    }
  }
  // view
  template: `
    <div>
      {{ counter }}
      <button @click="increment()"> + </button>
    </div>
  `,
  // actions
  methods: {
    increment () {
      store.commit('increment')
    },
  },
})
```

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-laravue/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 displayed, showing the 'state' node with a value of 'counter: 3' and the 'mutation' node with '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.

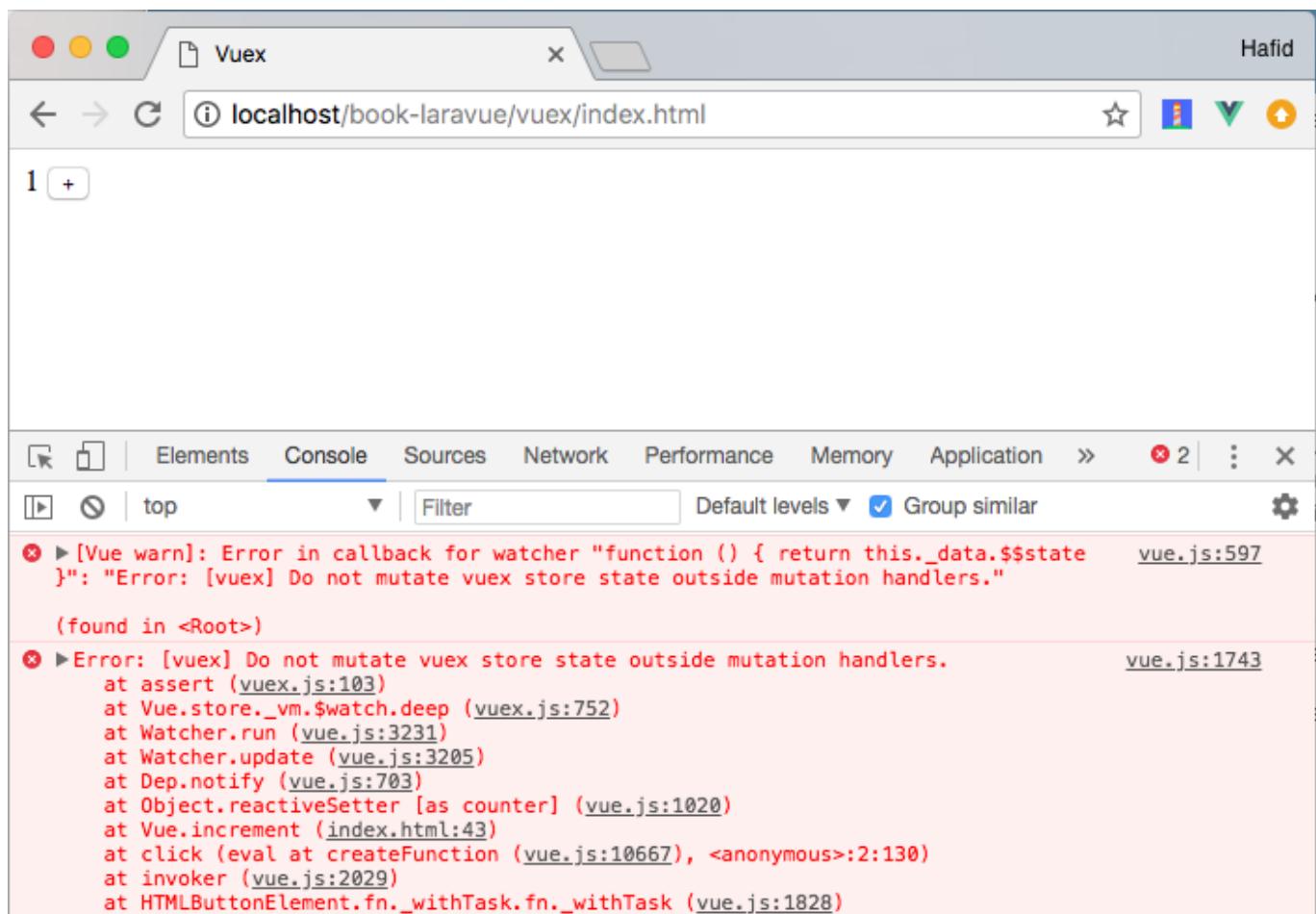
Jika kita aktifkan mode strict pada Vuex, maka perubahan state diluar mutation akan menimbulkan pesan warning pada console browser, tambahkan properti `strict: true` pada store.

```
const store = new Vuex.Store({
  strict: true,
  state: {
    counter: 0
  },
  ...
})
```

Kemudian ubah method increment menjadi sebagai berikut.

```
new Vue({
  ...
  // actions
  methods: {
    increment () {
      //store.commit('increment')
      store.state.counter++
    },
  },
})
```

Mari kita lihat hasilnya.



Disamping itu perubahan state tidak tercatat pada devtools Vue

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 di deklarasikan 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`)

```
export const Hello = {
  template: `
    <p>
      State counter pada hello :
      {{ counter }}
    </p>
  `,
  computed: {
    counter(){
      return store.state.counter
    }
  }
}
```

Kemudian pada file utama, kita import component ini.

```
<script type="module">
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.

```
new Vue({
  el: '#app',
  components: {
    'hello': Hello
  },
  ...
  // view
  template: `
    <div>
      {{ counter }}
      <button @click="increment()"> + </button>
      <hello></hello>
    </div>
  `,
  ...
})
</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 two errors:

- [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

Filter mutations

Base State 01:50:00

increment 02:53:43

increment 02:53:43

Filter inspected state

state

counter: 2

mutation

payload: undefined

type: "increment"

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.

```
new Vue({
  el: '#app',
  store, // tambahkan ini atau store: store
  components: {
    'hello': Hello
  },
  ...
})
```

Kemudian pada component, kita bisa gunakan perintah `this.$store` untuk mengakses store.

```
this.$store.state.counter
```

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.

```
var store = new Vuex.Store({
  ...
  getters: {
    counter: state => state.counter

    /*
    counter(state){
      return state.counter
    }
    */
  }
})
```

Untuk memanggilnya pada computed menggunakan perintah `store.getters.counter`, Oleh karenanya kita bisa ubah cara pemanggilannya pada properti computed

```
computed: {
  counter(){
    return store.getters.counter
  }
},
```

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.

```
increment (state) {
  state.counter++
}

// atau versi es6 arrow function
increment: state => state.counter++

// untuk mengakses
store.commit('increment')
```

Kita juga dizinkan menambahkan argumen (payload) pada fungsi ini.

```
increment (state, n) {
  state.counter += n
}
```

```
// untuk mengaksesnya  
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.

```
mutations: {  
    //increment: state => state.counter++  
    increment(state) {  
        setTimeout(()=>{  
            state.counter++  
        }, 1000)  
    }  
},
```

Hasilnya akan muncul error dan state tidak berubah.

The screenshot shows a browser window titled "Vuex". The address bar shows "localhost/book-laravel/vuex/index.html". The page content displays "State counter pada hello : 1". Below the browser window is the Chrome DevTools console tab. The console output shows two errors related to Vuex mutations:

```
[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  
(found in <Root>)  
Error: [vuex] Do not mutate vuex store state outside mutation handlers. vue.js:1743  
at assert (vuex.js:103)  
at Vue.store._vm.$watch.deep (vuex.js:752)  
at Watcher.run (vuex.js:3231)  
at Watcher.update (vuex.js:3205)  
at Dep.notify (vuex.js:703)  
at Object.reactiveSetter [as counter] (vuex.js:1020)  
at setTimeout (index.html:27)
```

Artinya, perubahan state pada mutation harus dilakukan pada saat itu juga, tidak boleh menunggu barang satu detik pun

Perhatian: 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.

```
var store = new Vuex.Store({
  strict: true,
  state: {
    counter: 0
  },
  mutations: {
    increment: state => state.counter++
  },
  actions: {
    increment: (context) => {
      context.commit('increment')
    }
    // atau
    /*
    increment: ({commit}) => {
      commit('increment')
    }
    */
  },
  getters: {
    counter: state => state.counter
  }
})
```

Lalu cara memanggilnya pada objek Vue atau component

```
new Vue({
  el: '#app',
  store,
  components: {
    'hello': Hello
  },
  computed: {
    counter(){
      return store.getters.counter
    }
  },
  // view
  template: `
    <div>
      {{ counter }}
      <button @click="increment()"> + </button>
      <hello></hello>
    </div>
  `
})
```

```

        ,
methods: {
    increment () {
        // store.commit('increment')
        store.dispatch('increment')
    },
},
})

```

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.

Kerangka Promise.

```

return new Promise((resolve, reject) => {
    //if success
    resolve()
    // if error
    reject()
})

```

Untuk mensimulasikan asynchronous action maka kita bisa gunakan data JSON buku (dummy) sebagaimana yang telah digunakan pada bab terdahulu.

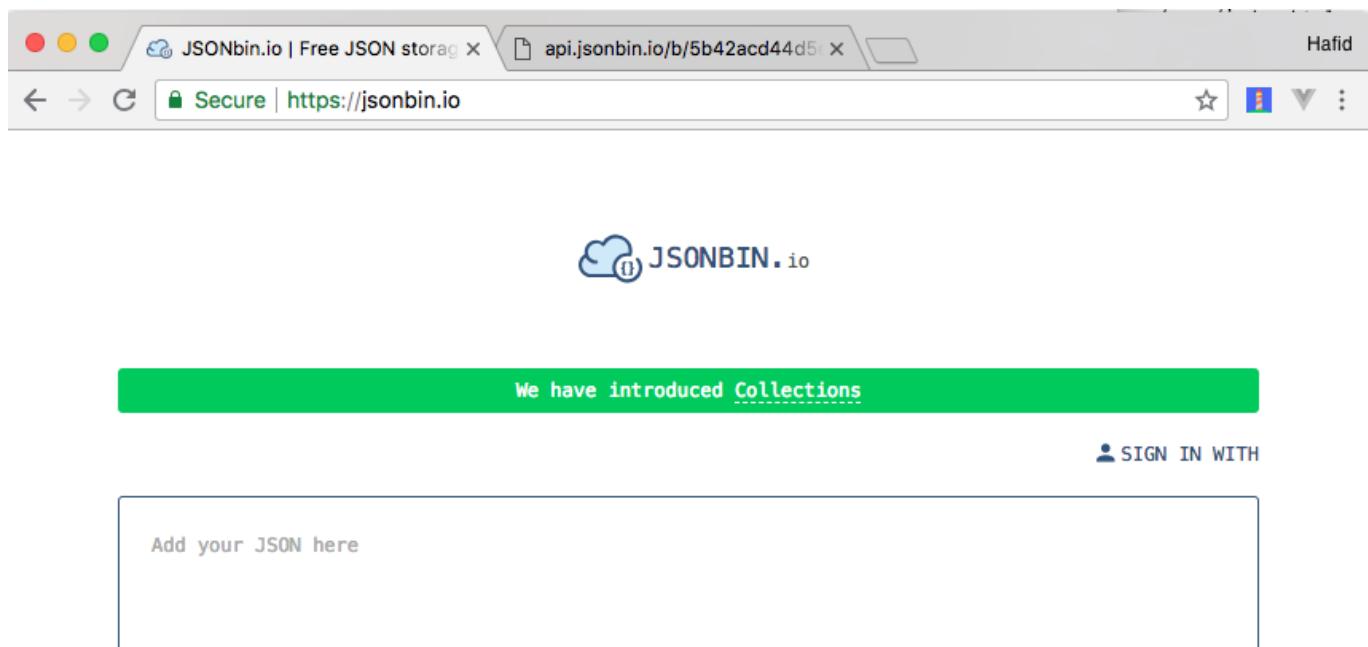
```

[
{
    "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"
},
{
    "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"
},
{

```

```
        "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"
    },
{
    "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"
}
]
```

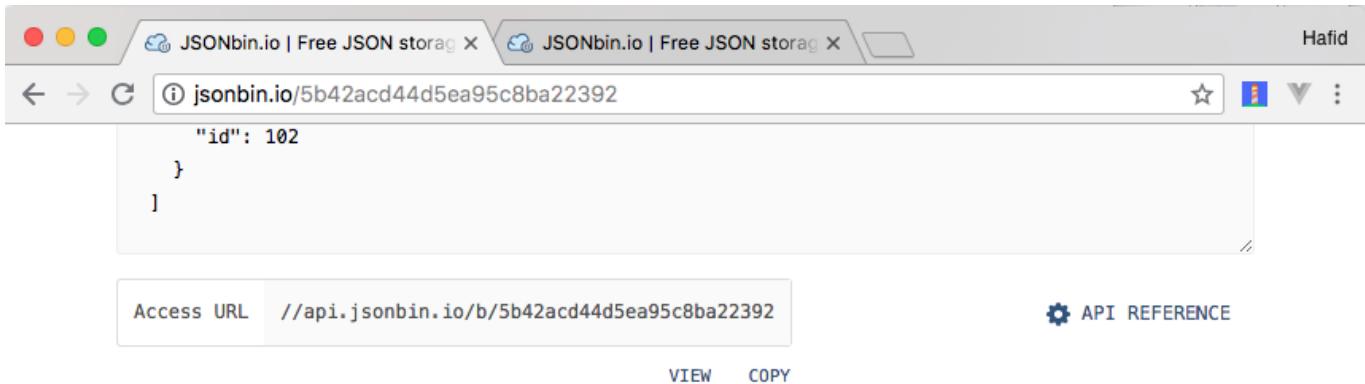
Data JSON ini kemudian bisa kita hosting ke (misalnya) <https://jsonbin.io> untuk sekedar menguji coba.



The screenshot shows a web browser window with the following details:

- Address Bar:** Shows "Secure | https://jsonbin.io".
- Header:** "JSONbin.io | Free JSON storage" and "api.jsonbin.io/b/5b42acd44d5ea95c8ba22392".
- User:** "Hafid".
- Content Area:** Displays the JSON data from the code block above. It includes a green banner at the top that says "We have introduced Collections".
- Input Field:** A large text area with the placeholder "Add your JSON here".
- Buttons:** "SIGN IN WITH" and "SIGN UP".

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>



The screenshot shows a browser window with two tabs, both titled "JSONbin.io | Free JSON storage". The active tab's URL is "jsonbin.io/5b42acd44d5ea95c8ba22392". The content of the page is a JSON array with one item:

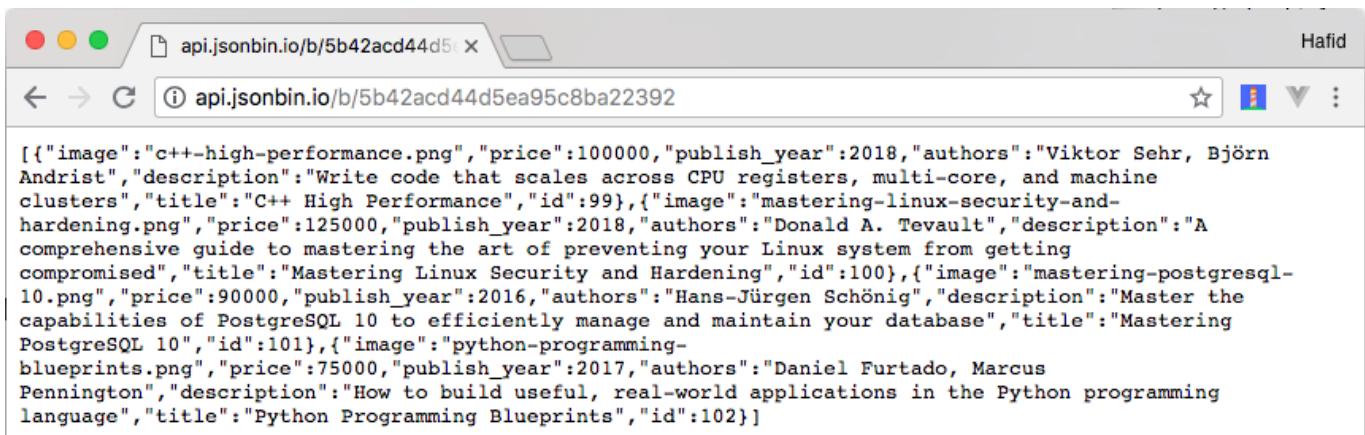
```

{
  "id": 102
}

```

Below the JSON, there are buttons for "Access URL" (with the value //api.jsonbin.io/b/5b42acd44d5ea95c8ba22392), "VIEW", and "COPY".

Testing API tersebut.



The screenshot shows a browser window with one tab, titled "api.jsonbin.io/b/5b42acd44d5ea95c8ba22392". The content of the page is a JSON array containing multiple book objects:

```

[{"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}]

```

Kemudian kita siapkan kerangka store menjadi sebagai berikut.

```

var store = new Vuex.Store({
  strict: true,
  state: {
    books: []
  },
  mutations: {
    setBooks(state, books){
      state.books = books
    }
  },
  actions: {
    getBooks ({ commit }) {
      return new Promise((resolve, reject) => {
        //if success
        resolve()
        // if error
        reject()
      })
    }
  },
  getters: {
    books: state => state.books
  }
})

```

Kita siapkan state books, mutations setBooks, action getBooks dan getters books.

Kita masih akan gunakan pustaka XMLHttpRequest untuk merequest data JSON buku yang telah kita siapkan. Kodenya kita bungkus menggunakan Promise pada actions getBooks() agar mendukung asynchronous.

```
getBooks ({ commit }) {
    return new Promise((resolve, reject) => {
        var xhr = new XMLHttpRequest();
        xhr.open("GET",
"http://api.jsonbin.io/b/5b42acd44d5ea95c8ba22392");
        xhr.onload = function () {
            if (this.status >= 200 && this.status < 300) {
                commit('setBooks', JSON.parse(xhr.response) )
                resolve(xhr.response);
            } else {
                reject({
                    status: this.status,
                    statusText: xhr.statusText
                });
            }
        };
        xhr.onerror = function () {
            reject({
                status: this.status,
                statusText: xhr.statusText
            });
        };
        xhr.send();
    })
}
```

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`

```
commit('setBooks', JSON.parse(xhr.response) )
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.

```
new Vue({
  el: '#app',
```

```

store,
computed: {
    books(){
        return store.getters.books
    }
},
created() {
    store.dispatch('getBooks')
        .then((response) => {
            console.log('result: ', response)
        })
        .catch((error) => {
            console.log('error: ', error)
        })
}
)
}
)

```

Selanjutnya pada template, kita gunakan teknis list rendering.

```

<div id="app">
    <ul v-for="book in books">
        <li>{{ book.title }}</li>
    </ul>
</div>

```

Waktunya ujicoba.

The screenshot shows a browser window with the title "Vuex" and the URL "localhost/book-laravel/vuex/index.html". The page content is a simple list of books:

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

Below the browser window is the "Vue DevTools" developer tools panel. The "Elements" tab is selected, showing the DOM structure. The "Console" tab displays the following JavaScript object:

```

result: [{"image": "c++-high-performance.png", "price": 10000, "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": 12500, "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": 9000, "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": 7500, "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}]

```

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

```
var store = new Vuex.Store({
    strict: true, // supaya warning muncul
    state: {
        name: 'Hafid'
    },
    mutations: {
        setName: (state, name) => {
            state.name = name
        }
    },
    /*
    // boleh juga fungsi biasa
    setName (state, name) {
        state.name = name
    }
    */
},
getters: {
    name: state => state.name
}
)

new Vue({
    el: '#app',
    store,
    computed: {
        name(){
            return store.getters.name
        }
    },
    template: `
        <div>
            <input v-model='name'>
        </div>
    `,
})

```

Mari kita lihat hasilnya..

The screenshot shows a browser window with a title bar 'Vuex' and a URL 'localhost/book-laravue/vuex/index.html'. Below the browser is the Chrome DevTools interface. The 'Console' tab is active. In the input field above the console, the text 'Hafid' is typed. The console output shows the following message:

```
Vue warn]: Computed property "name" was assigned to but it has no setter.
```

Wah berhasil!

Tapi!! ketika field input kita ketikkan suatu teks ternyata muncul error pada console.

The screenshot shows a browser window with a title bar 'Vuex' and a URL 'localhost/book-laravue/vuex/index.html'. Below the browser is the Chrome DevTools interface. The 'Console' tab is active. In the input field above the console, the text 'Hafid Mu' is typed. The console output shows the following message:

```
Vue warn]: Computed property "name" was assigned to but it has no setter.
```

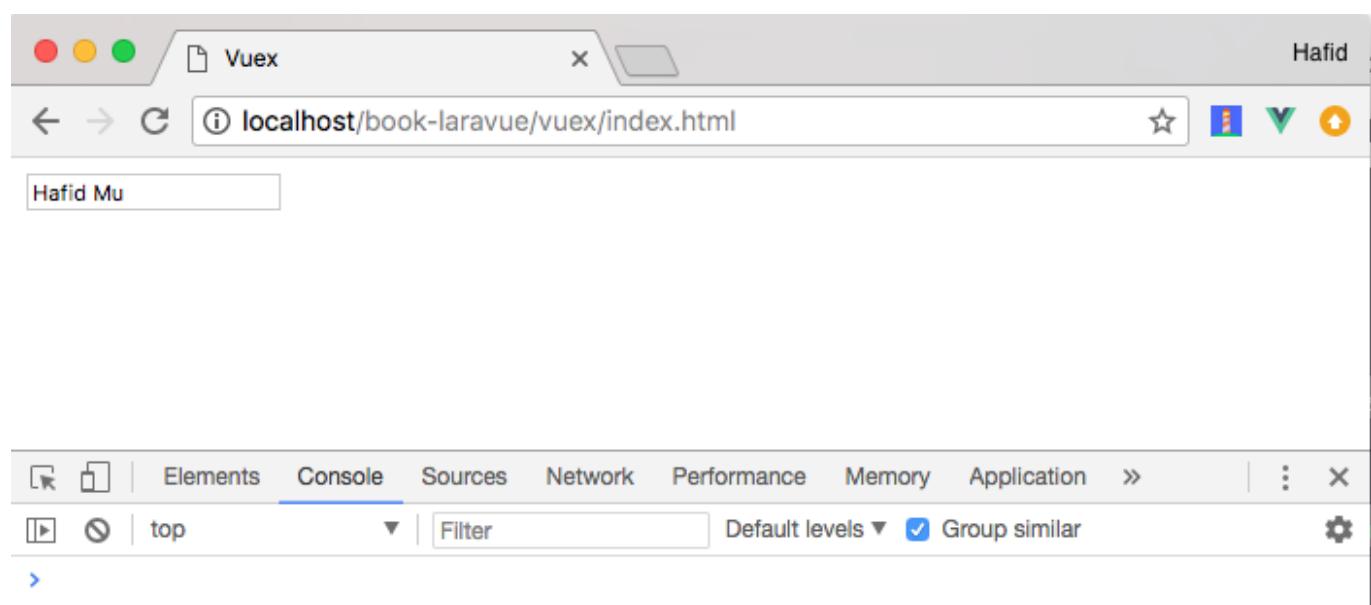
Intinya properti name tidak memiliki setter. Yap, karena properti computed itu secara default sifatnya hanya getter saja.

Oleh karena itu, kita perlu buat atau definisikan getter dan setter pada computed `name`, seperti ini bentuknya.

```
// ...
new Vue({
  el: '#app',
  computed: {
    name: {
      get () {
        return store.getters.name
      }
    }
  }
})
```

```
        },
        set (value) {
            store.commit('setName', value)
        }
    },
    template: `
        <div>
            <input v-model='name'>
        </div>
    `,
})
)
```

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 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.

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 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:

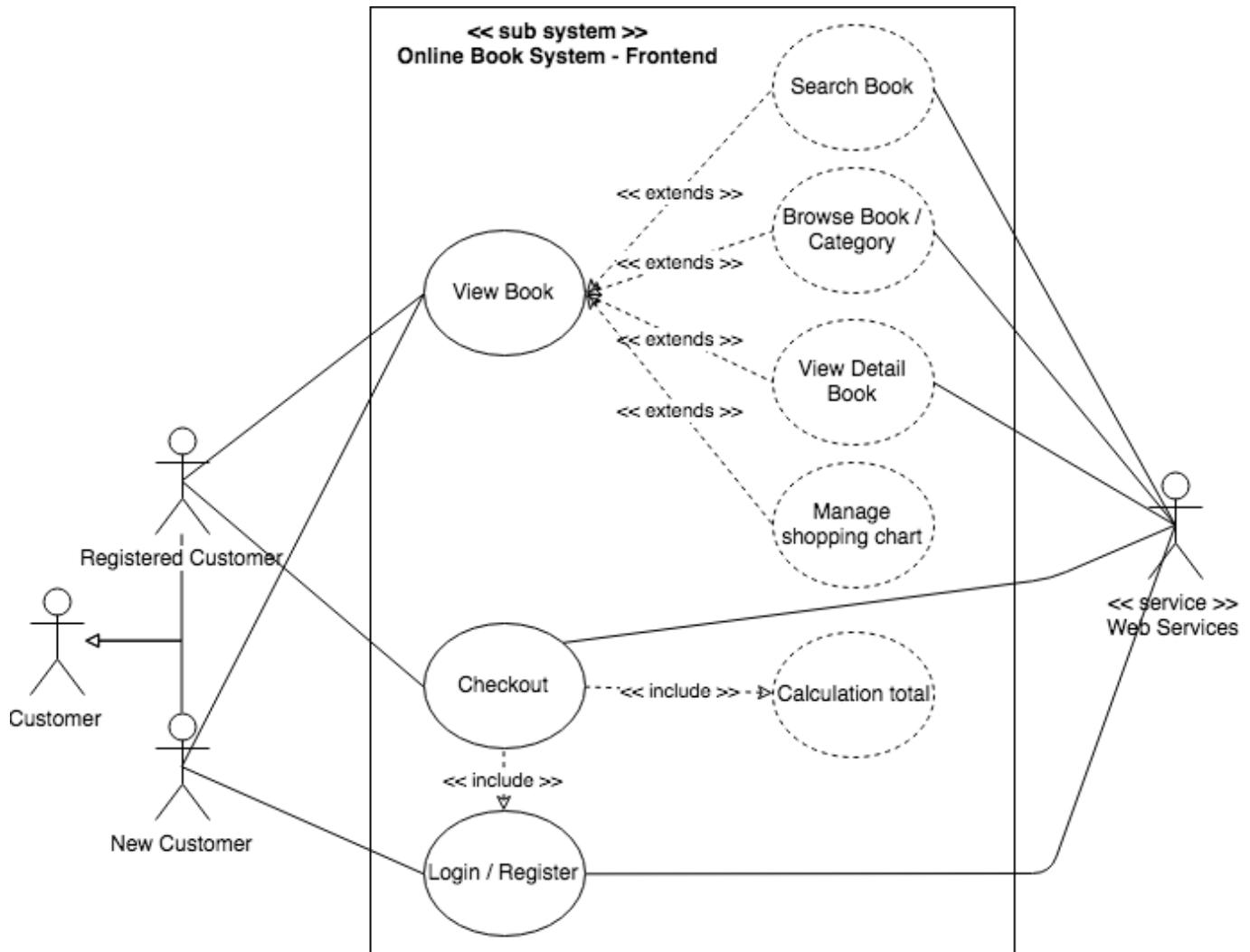
- Tampilan daftar buku (list & detail)
- Keranjang belanja
- Keanggotaan
- Checkout Pemesanan
- Single Page Application (SPA) berkaitan dengan kenyamanan user interface di mobile
- Progressive Web Application (PWA) untuk mengatasi performa di mobile

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.

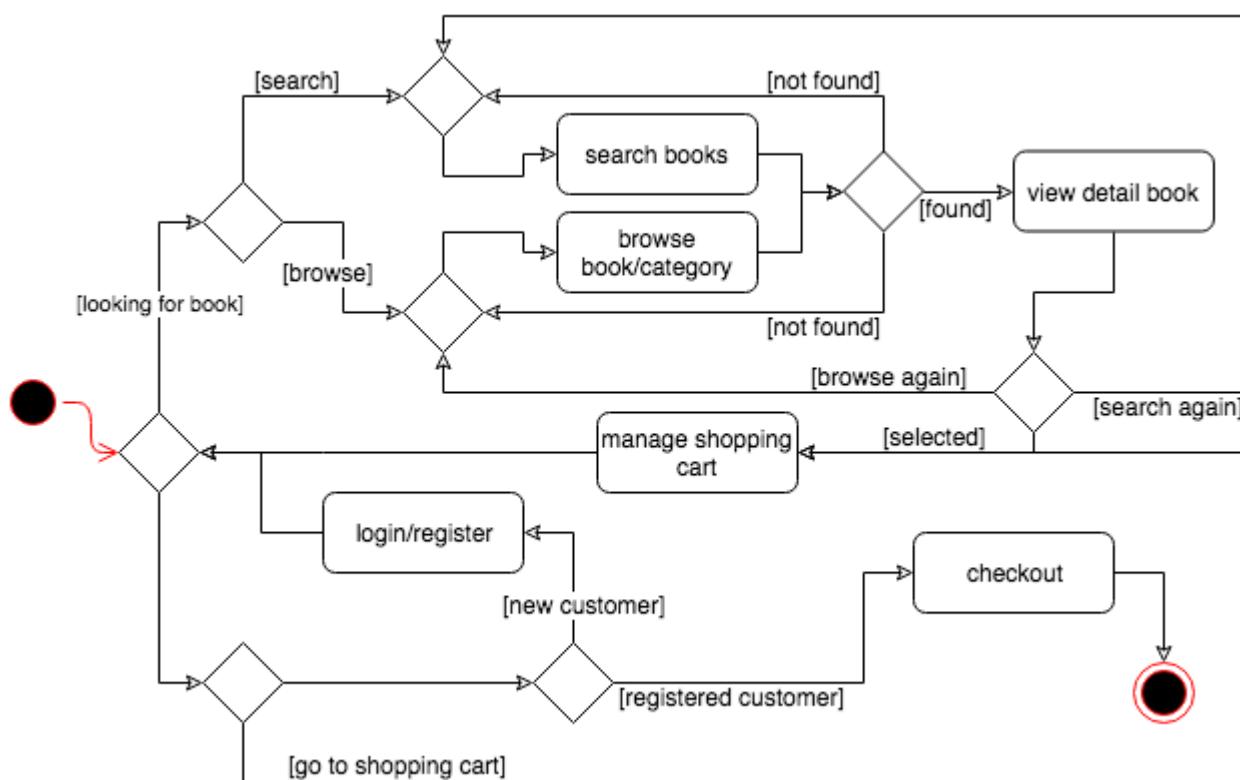
Action **Checkout** memiliki sub action yang sudah include yaitu **Calculation Total** (perhitungan total bayar). Action ini juga terhubung ke **Web Service** yang menjembatani pengiriman data pemesanan buku ke server.

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: Apa itu web service? aplikasi di sisi backend yang akan menyuplai dan memproses data (buku, pemesanan, dsb) yang kita gunakan sistem ini.

Activity Diagram

Diagram ini menggambarkan alur dari setiap aktivitas yang dilakukan pada sistem ini.



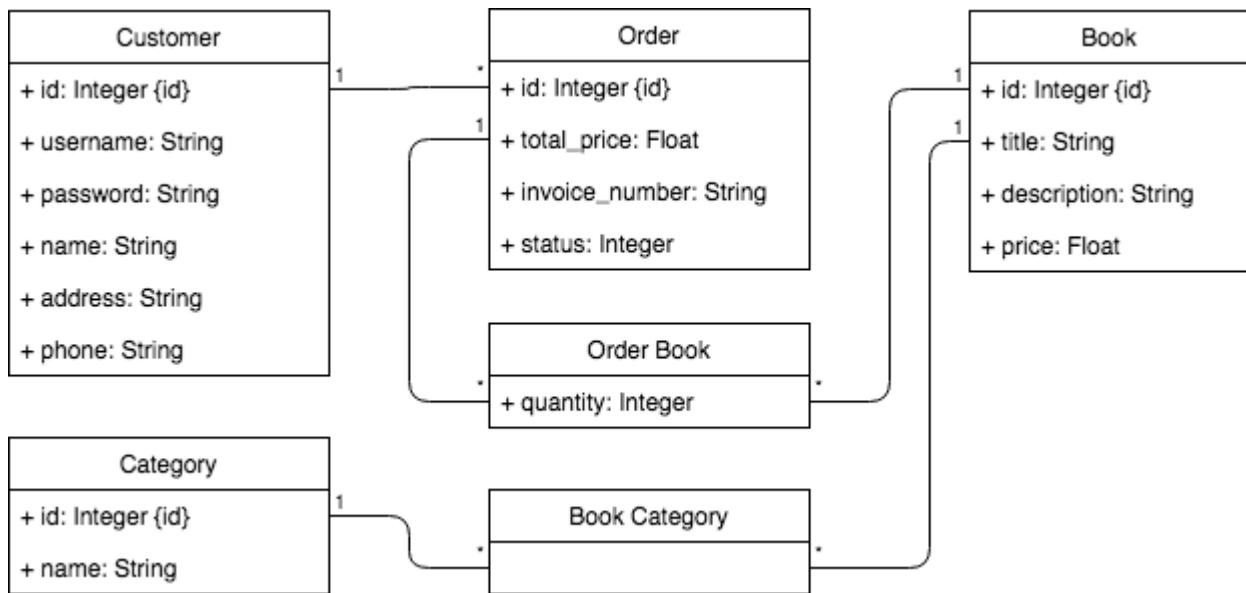
Mula-mula customer dapat melakukan beberapa pilihan aktivitas yaitu **search books**, **browse book/category**, **manage shopping cart** dan **login/register**. Setelah melakukan aktivitas **search books** dan **browse book/category**, jika buku ditemukan maka customer akan melakukan aktivitas **view 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 aktifitas 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

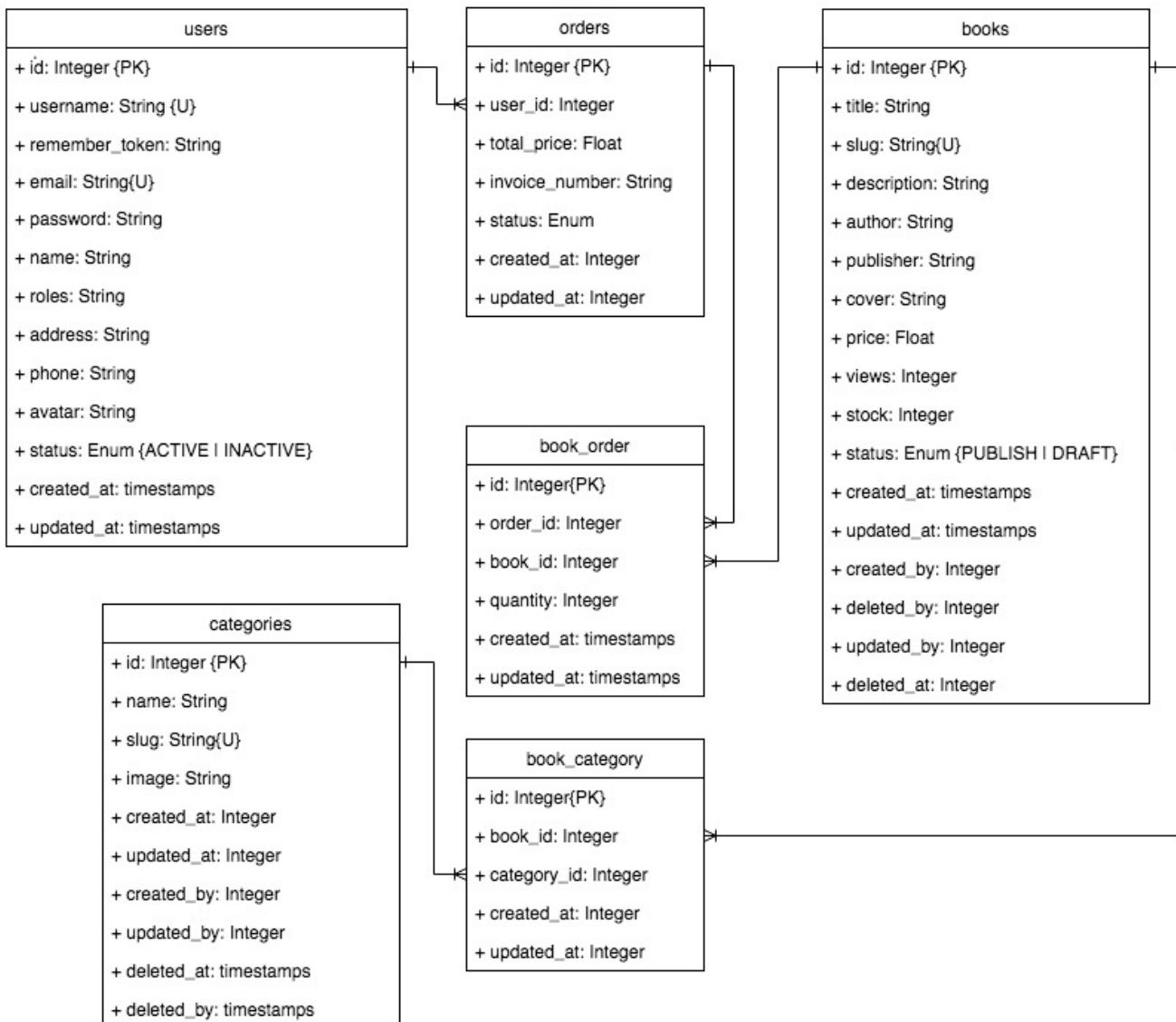
Diagram ini menggambarkan objek-objek & properti utama yang terlibat di dalam sistem beserta hubungannya satu sama lain.



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.

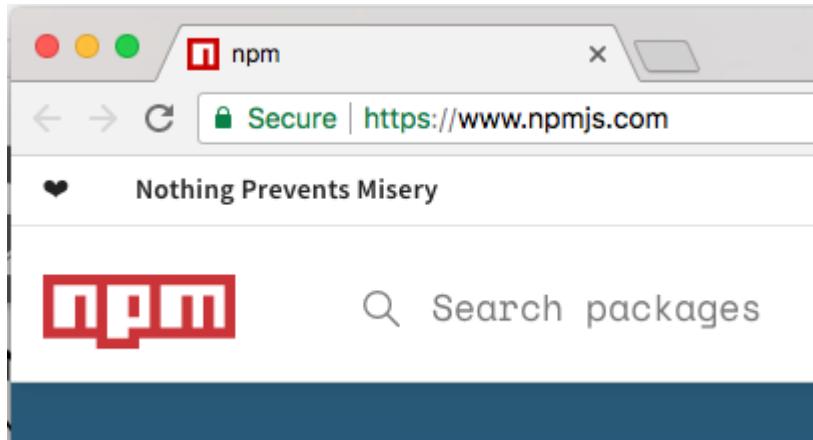
Command Line Tools

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 Javascript

Package manager merupakan tools berbasis command line (terminal) untuk memudahkan kita dalam mengelola (baca: instalasi, pengecekan dependensi, updating, sharing) pustaka Javascript yang digunakan pada aplikasi kita. Saat ini, terdapat dua package manager populer dalam dunia Javascript 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>.

A screenshot of a web browser window titled "Node.js". The address bar shows a secure connection to "https://nodejs.org/en/". The page features the Node.js logo at the top center. Below the logo is a navigation bar with links: HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, NEWS, and FOUNDATION. A green button labeled "FOUNDATION" is highlighted. The main content area contains text about Node.js and a call to action for the "JS Interactive" event.

Node.js® is a JavaScript runtime built on [Chrome's V8 JavaScript engine](#).



August 2018 security releases available, upgrade now

Download for macOS (x64)

8.11.4 LTS

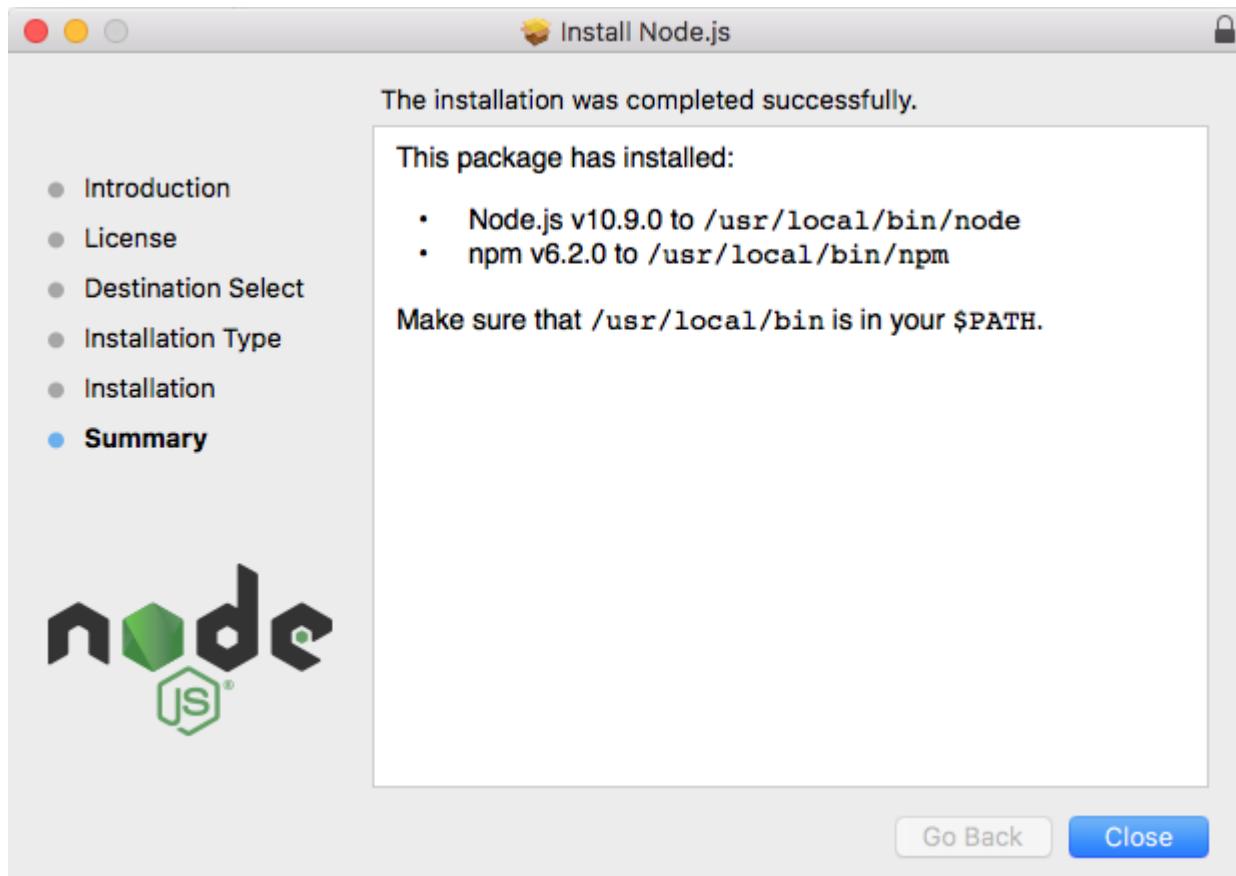
Recommended For Most Users

10.9.0 Current

Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

[Other Downloads](#) | [Changelog](#) | [API Docs](#)



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
```

```
vue-projects — root@8a48a85542c9: /var/www/larashop-api — docker + docker-comp
[Hafids-MacBook-Pro:vue-projects hafidmukhlasin$ node -v
v10.9.0
[Hafids-MacBook-Pro:vue-projects hafidmukhlasin$ npm -v
6.2.0
```

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
```

```
vue-projects — root@8a48a85542c9: /var/www/la
[Hafids-MacBook-Pro:vue-projects hafidmukhlasin$ sudo npm install npm@latest -g
Password:
/usr/local/bin/npm -> /usr/local/lib/node_modules/npm/bin/npm-cli.js
/usr/local/bin/npx -> /usr/local/lib/node_modules/npm/bin/npx-cli.js
+ npm@6.4.0
updated 1 package in 7.832s
Hafids-MacBook-Pro:vue-projects hafidmukhlasin$
```

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 di awal bahwa pustaka Vue bisa juga diinstalasi dengan mudah melalui tools NPM.

```
npm install vue
```

```
[Hafids-MacBook-Pro:example hafidmukhlasi$ mkdir npm-vue
[Hafids-MacBook-Pro:example hafidmukhlasi$ cd npm-vue/
[Hafids-MacBook-Pro:npm-vue hafidmukhlasi$ npm install vue
[npm WARN saveError ENOENT: no such file or directory, open '/Users/hafidmukhlasi/n/Dev/book-laravue/example/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/hafidmukhlasi/D/ev/book-laravue/example/npm-vue/package.json'
[npm WARN vue No description
[npm WARN vue No repository field.
[npm WARN vue No README data
[npm WARN vue No license field.

+ vue@2.5.16
added 1 package from 1 contributor and audited 1 package in 2.304s
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
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.

```
<!DOCTYPE html>
<html>
<head>
  <title>Belajar Vue</title>
  <script src=".//node_modules/vue/dist/vue.js"></script>
</head>
<body>

  <div id="app">
    {{ msg }}
  </div>

  <script>
var vm = new Vue({
  el: '#app',
  data: {
    msg: 'Vue on npm'
  }
})
</script>
</body>
</html>
```

Kemudian, jalankan pada browser.



Apa itu Bundler

Pada bagian sebelumnya, sedikit disinggung mengenai Javascript bundler, tools apa lagi itu?

Kalau kita membuat aplikasi menggunakan Javascript maka akan banyak file pustaka yang kita butuhkan atau *include*-kan pada file utama kita misal: `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.

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

Javascript bundler mengatasi hal ini dengan meletakkan file pustaka dan dependensinya dalam satu file Javascript, sehingga kita cukup meng-*include*-kan file tersebut saja dan semua pustaka bisa kita gunakan dalam kode kita.

Ada banyak pustaka Javascript bundler yang ada saat ini, diantaranya: webpack, browserify, rollup, parcel, dll. Konon webpack adalah yang paling populer sedangkan parcel adalah yang paling cepat.

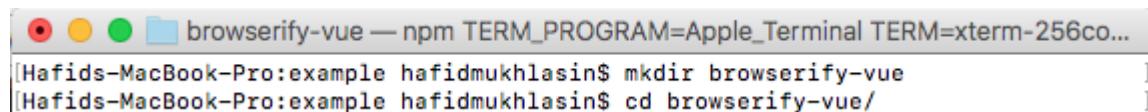
Instalasi Vue menggunakan Javascript Bundler

Jika pada bagian sebelumnya kita menginstalasi Vue dengan meng-*include*-kan secara langsung pustakanya melalui elemen `<script>` maka pada bagian ini kita akan mencoba menggunakan Javascript bundler yang direkomendasikan untuk pengembangan projek aplikasi berbasis Javascript skala besar.

Sebagai bahan latihan, kita akan mencoba menggunakan dua pustaka Javascript bundler yaitu browserify dan webpack.

Browserify

Kita siapkan dulu folder untuk latihan ini.



```
[Hafids-MacBook-Pro:example hafidmukhlasin$ mkdir browserify-vue
[Hafids-MacBook-Pro:example hafidmukhlasin$ cd browserify-vue/]
```

Lalu kita masuk ke dalam folder tersebut, Kita inisialisasi projek ini dengan menjalankan perintah `npm init`, perintah ini akan menggenerate file `package.json` yang berisi profil dari projek kita

```
[Hafids-MacBook-Pro:browserify-vue hafidmukhlasin$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See 'npm help json' for definitive documentation on these fields
and exactly what they do.

Use 'npm install <pkg>' afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
[package name: (browserify-vue) browserify-vue
[version: (1.0.0) 1.0.0
[description: Vue on Browserify
[entry point: (index.js) main.js
[test command:
[git repository:
[keywords:
[author: Hafid Mukhlasin
[license: (ISC)
About to write to /Users/hafidmukhlasin/Dev/book-laravue/example/browserify-vue/
package.json:

{
  "name": "browserify-vue",
  "version": "1.0.0",
  "description": "Vue on Browserify",
  "main": "main.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "author": "Hafid Mukhlasin",
  "license": "ISC"
}

[Is this OK? (yes) yes
Hafids-MacBook-Pro:browserify-vue hafidmukhlasin$ ]]
```

Kemudian kita instalasi Vue melalui terminal dengan menggunakan perintah `npm install --save vue`

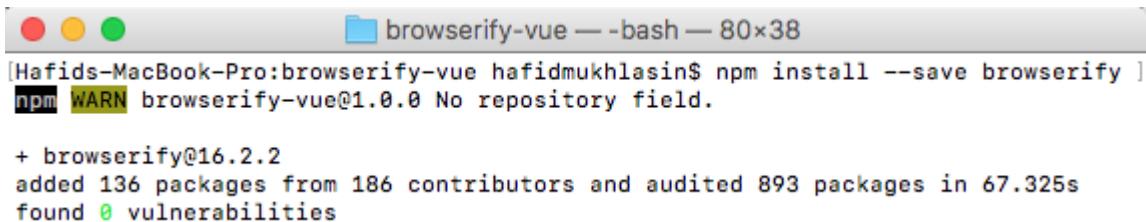
```
[Hafids-MacBook-Pro:browserify-vue hafidmukhlasin$ npm install --save vue
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN browserify-vue@1.0.0 No repository field.

+ vue@2.5.16
added 1 package from 1 contributor in 13.572s ]
```

Jika kita lihat pada file package.json maka akan muncul dependencies dari pustaka yang baru kita instalasi yaitu Vue.

```
...
"dependencies": {
  "vue": "^2.5.17"
}
```

Setelah itu kita instalasi browserify, gunakan perintah `npm install --save browserify`



```
[Hafids-MacBook-Pro:browserify-vue hafidmukhlasin$ npm install --save browserify ]
npm WARN browserify-vue@1.0.0 No repository field.

+ browserify@16.2.2
added 136 packages from 186 contributors and audited 893 packages in 67.325s
found 0 vulnerabilities
```

Maka dependecies pada package.json akan bertambah

```
"dependencies": {
  "browserify": "^16.2.2",
  "vue": "^2.5.17"
}
```

Buat file app.js

```
const Vue = require('vue/dist/vue.common')
new Vue({
  el: "#app",
  data: {
    message: "Vue on Browserify",
  }
})
```

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 `main.js`

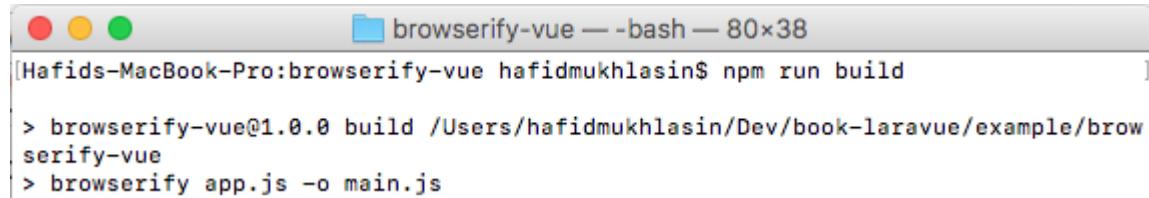
```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "build": "browserify app.js -o main.js"
},
```

Buat file index.html, *include*-kan file main.js menggunakan elemen `<script>`.

```
<!DOCTYPE html>
<html>
<head>
  <title>Belajar Vue</title>
</head>
<body>
  <div id="app">
    {{ message }}
  </div>
```

```
<script src="main.js"></script>
</body>
</html>
```

Pada terminal, jalankan perintah `npm run build`



```
[Hafids-MacBook-Pro:browserify-vue hafidmukhlasin$ npm run build
> browserify-vue@1.0.0 build /Users/hafidmukhlasin/Dev/book-laravue/example/browserify-vue
> browserify app.js -o main.js
```

Kemudian, jalankan file `index.html` tersebut via browser.



Demikianlah cara yang perlu kita lakukan jika menggunakan browserify sebagai bundler.

Webpack

Webpack merupakan Javascript 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 dan menjalankan perintah `npm init` untuk menginisialisasi projek kita via `package.json`.

```

Hafids-MacBook-Pro:example hafidmukhlasin$ mkdir webpack-vue
[Hafids-MacBook-Pro:example hafidmukhlasin$ cd webpack-vue/
[Hafids-MacBook-Pro:webpack-vue hafidmukhlasin$ npm init
[This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (webpack-vue) webpack-vue
[version: (1.0.0) 1.0.0
[description: Vue on Webpack
[entry point: (index.js) ./dist/build.js
[test command:
[git repository:
[keywords:
[author: Hafid Mukhlasin
[license: (ISC)
[About to write to /Users/hafidmukhlasin/Dev/book-laravue/example/webpack-vue/pac
kage.json:

{
  "name": "webpack-vue",
  "version": "1.0.0",
  "description": "Vue on Webpack",
  "main": "./dist/build.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "author": "Hafid Mukhlasin",
  "license": "ISC"
}

Is this OK? (yes) yes
[Hafids-MacBook-Pro:webpack-vue hafidmukhlasin$ ]

```

Pada kode di atas, file Javascript utama diletakkan pada lokasi `./dist/build.js`, di mana file ini nantinya akan di-generate oleh webpack.

Buat file `index.html` pada root directory yang meng-*include* file `build.js`.

```

<!DOCTYPE html>
<html>
  <head>
    <title>Belajar Vue Webpack</title>
  </head>
  <body>
    <div id="app">
      {{ message }}
    </div>
    <script src="dist/build.js"></script>
  </body>
</html>

```

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.

```
import Vue from 'vue'

new Vue({
  el: '#app',
  data: {
    message: "Vue on Webpack"
  }
})
```

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:

```
const path = require('path')

module.exports = {
  mode: 'development',
  entry: './src/app.js',
  output: {
    path: path.resolve(__dirname, 'dist'),
    publicPath: '/dist/',
    filename: 'build.js'
  },
  resolve: {
    alias: {
      'vue$': 'vue/dist/vue.esm.js',
    }
  },
  module: {
    rules: [
      {
        test: /\.js$/,
        exclude: /node_modules/,
        use: {
          loader: "babel-loader"
        }
      }
    ]
  }
}
```

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 Javascript yang didukung oleh browser dengan menggunakan pustaka babel.

Apa itu babel? pustaka loader untuk file Javascript. <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.

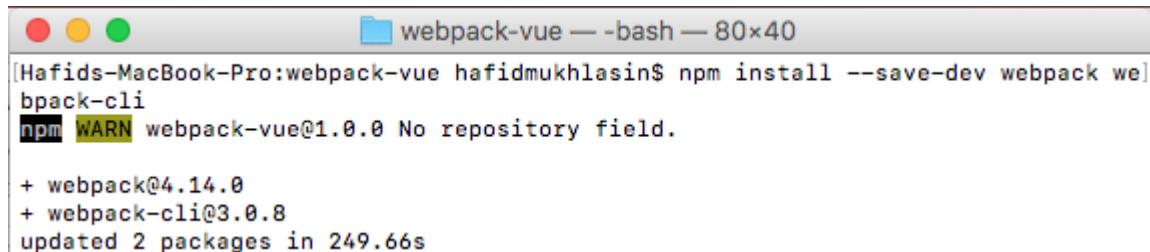
```
{  
  "presets": ["env"]  
}
```

Lakukan instalasi pustaka utama yaitu Vue dengan menggunakan perintah npm install --save vue. Maka dependecies pada package.json akan bertambah

```
"vue": "^2.5.17"  
}  
}
```

Selanjutnya, kita instalasi pustaka pendukung untuk development yaitu webpack, dan babel.

```
npm install --save-dev webpack webpack-cli  
npm install --save-dev babel-core babel-loader babel-preset-env
```



```
[Hafids-MacBook-Pro:webpack-vue hafidmukhlasin$ npm install --save-dev webpack we]  
bpack-cli  
npm WARN webpack-vue@1.0.0 No repository field.  
+ webpack@4.14.0  
+ webpack-cli@3.0.8  
updated 2 packages in 249.66s
```

```
[Hafids-MacBook-Pro:webpack-vue hafidmukhlasin$ npm install --save-dev babel-core  
babel-loader babel-preset-env  
npm WARN webpack-vue@1.0.0 No repository field.  
+ babel-core@6.26.3  
+ babel-loader@7.1.4  
+ babel-preset-env@1.7.0  
added 1 package from 1 contributor, updated 2 packages and audited 6181 packages  
in 167.388s  
found 0 vulnerabilities]
```

Hasil dari instalasi ini akan mengupdate secara otomatis file package.json

```
...  
"devDependencies": {  
  "babel-cli": "^6.26.0",  
  "babel-core": "^6.26.3",  
  "babel-loader": "^7.1.4",  
  "babel-preset-env": "^1.7.0",  
  "webpack": "^4.14.0",  
  "webpack-cli": "^3.0.8"  
}  
}
```

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`

```
"scripts": {  
  "test": "echo \\\"Error: no test specified\\\" && exit 1",  
  "dev": "webpack --config webpack.config.dev.js",  
},
```

Jalankan perintah `npm run dev` pada terminal untuk megeksekusi webpack dengan konfigurasi sebagaimana file `webpack.config.dev.js`.

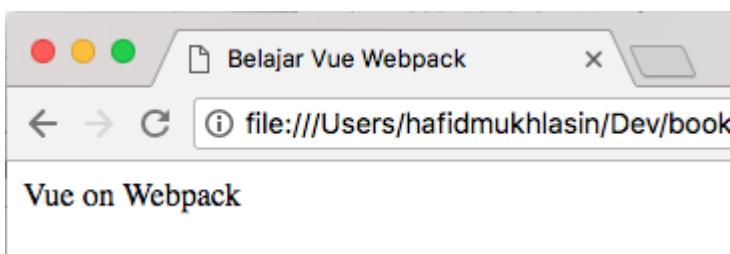
```

Hafids-MacBook-Pro:webpack-vue hafidmukhlasin$ npm run dev
> webpack-vue@1.0.0 dev /Users/hafidmukhlasin/Dev/book-laravel/example/webpack-vue
> webpack --config webpack.config.dev.js

Hash: f6e23d2b90a3a297ddaf
Version: webpack 4.14.0
Time: 869ms
Built at: 06/30/2018 12:18:49 PM
    Asset      Size  Chunks             Chunk Names
build.js   317 KiB  main  [emitted]  main
[./node_modules/webpack/buildin/global.js] (webpack)/buildin/global.js 489 bytes
  {main} [built]
[./src/app.js] 155 bytes {main} [built]
+ 4 hidden modules

```

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 Javascript 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
```

```

Hafids-MacBook-Pro:webpack-vue hafidmukhlasin$ npm install --save-dev vue-loader
npm WARN vue-loader@15.2.4 requires a peer of css-loader@* but none is installed
+ You must install peer dependencies yourself.
npm WARN vue-loader@15.2.4 requires a peer of vue-template-compiler@^2.0.0 but none is installed. You must install peer dependencies yourself.
npm WARN webpack-vue@1.0.0 No repository field.

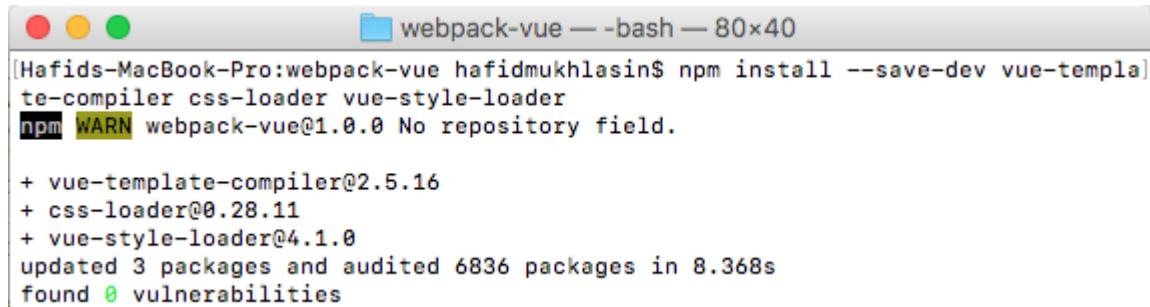
+ vue-loader@15.2.4
added 20 packages from 46 contributors and audited 6222 packages in 183.894s
found 0 vulnerabilities

```

Karena SFC pada vue tidak hanya berisi kode Javascript, namun juga template dan css maka kita perlu menginstalasi pustaka yang bisa menangani template dan css yaitu `vue-template-compiler`, `css-loader`, dan `css-loader vue-style-loader`.

Gunakan perintah perintah berikut pada terminal.

```
npm install --save-dev vue-template-compiler css-loader vue-style-loader
```



```
[Hafids-MacBook-Pro:webpack-vue hafidmukhlasin$ npm install --save-dev vue-template-compiler css-loader vue-style-loader
npm WARN webpack-vue@1.0.0 No repository field.

+ vue-template-compiler@2.5.16
+ css-loader@0.28.11
+ vue-style-loader@4.1.0
updated 3 packages and audited 6836 packages in 8.368s
found 0 vulnerabilities
```

Kemudian update file `webpack-config.dev.js` dengan menambahkan konfigurasi vue-loader.

```
const path = require('path')
const VueLoaderPlugin = require('vue-loader/lib/plugin')

module.exports = {
  mode: 'development',
  entry: './src/app.js',
  output: {
    path: path.resolve(__dirname, 'dist'),
    publicPath: '/dist/',
    filename: 'build.js'
  },
  resolve: {
    alias: {
      'vue$': 'vue/dist/vue.esm.js',
    }
  },
  module: {
    rules: [
      {
        test: /\.js$/,
        exclude: /node_modules/,
        use: {
          loader: "babel-loader"
        },
      },
      {
        test: /\.vue$/,
        loader: 'vue-loader'
      },
      {
        test: /\.css$/,
        use: [
          'vue-style-loader',
        ]
      }
    ]
  }
}
```

```

        'css-loader'
    ]
}
],
plugins: [
    new VueLoaderPlugin()
]
}
}

```

Kemudian kita coba buat sebuah component pada file terpisah (SFC). Buat file `Hello.vue` pada direktori `src`.

```

<template>
  <div class="hello">{{ text }}</div>
</template>

<script>
export default {
  data () {
    return {
      text: 'Hello dari vue-loader!'
    }
  }
}
</script>

<style>
.hello {
  color: blue;
}
</style>

```

Kode `Hello.vue` di atas hanya memberikan contoh bahwa kita dapat menggunakan tiga jenis kode sekaligus yaitu template Vue (HTML), Javascript dan CSS.

Ubah file `src/app.js` dengan mendaftarkan component Hello.

```

import Vue from 'vue'
import Hello from './Hello.vue'

new Vue({
  el: '#app',
  data: {
    message: "Vue on Webpack"
  },
  components: { Hello }
})

```

Kemudian tambahkan elemen component Hello pada index.html

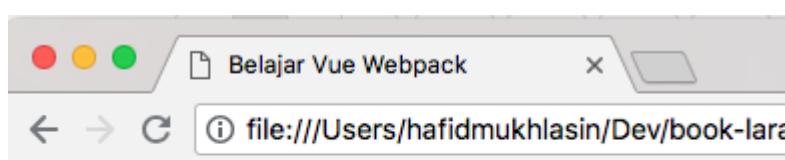
```
<div id="app">
  {{ message }}
  <hello></hello>
</div>
```

Jalankan perintah `npm run dev` pada terminal untuk mengeksekusi webpack.

```
[Hafids-MacBook-Pro:webpack-vue hafidmukhlasin$ npm run dev
> webpack-vue@1.0.0 dev /Users/hafidmukhlasin/Dev/book-laravue/example/webpack-vue
> webpack --config webpack.config.dev.js

Hash: f08ef0a2aa19b06085a3
Version: webpack 4.14.0
Time: 1235ms
Built at: 06/30/2018 12:27:15 PM
    Asset      Size  Chunks             Chunk Names
build.js  345 KiB  main  [emitted]  main
[./node_modules/babel-loader/lib/index.js!./node_modules/vue-loader/lib/index.js??vue-loader-options!./src/Hello.vue?vue&type=script&lang=js] ./node_modules/bab el-loader/lib!./node_modules/vue-loader/lib??vue-loader-options!./src/Hello.vue?vue&type=script&lang=js 104 bytes {main} [built]
[./node_modules/css-loader/index.js!./node_modules/vue-loader/lib/loaders/stylePostLoader.js!./node_modules/vue-loader/lib/index.js??vue-loader-options!./src/He llo.vue?vue&type=style&index=0&lang=css] ./node_modules/css-loader!./node_modules/vue-loader/lib/loaders/stylePostLoader.js!./node_modules/vue-loader/lib??vue- loader-options!./src/Hello.vue?vue&type=style&index=0&lang=css 190 bytes {main} [built]
[./node_modules/vue-loader/lib/loaders/templateLoader.js??vue-loader-options!./node_modules/vue-loader/lib/index.js??vue-loader-options!./src/Hello.vue?vue&type=template&id=184cbba9] ./node_modules/vue-loader/lib/loaders/templateLoader.js??vue-loader-options!./node_modules/vue-loader/lib??vue-loader-options!./src/Hello .vue?vue&type=template&id=184cbba9 272 bytes {main} [built]
[./node_modules/vue-style-loader/index.js!./node_modules/css-loader/index.js!./node_modules/vue-loader/lib/loaders/stylePostLoader.js!./node_modules/vue-loader/ lib/index.js??vue-loader-options!./src/Hello.vue?vue&type=style&index=0&lang=css] ./node_modules/vue-style-loader!./node_modules/css-loader!./node_modules/vue-1 oader/lib/loaders/stylePostLoader.js!./node_modules/vue-loader/lib??vue-loader-o ptions!./src/Hello.vue?vue&type=style&index=0&lang=css 1.38 KiB {main} [built]
[./node_modules/webpack/buildin/global.js] (webpack)/buildin/global.js 489 bytes {main} [built]
[./src/Hello.vue] 1.12 KiB {main} [built]
[./src/Hello.vue?vue&type=script&lang=js] 334 bytes {main} [built]
[./src/Hello.vue?vue&type=style&index=0&lang=css] 538 bytes {main} [built]
[./src/Hello.vue?vue&type=template&id=184cbba9] 196 bytes {main} [built]
[./src/app.js] 153 bytes {main} [built]
```

Kemudian jalankan file index.html pada browser. Lihat hasilnya..



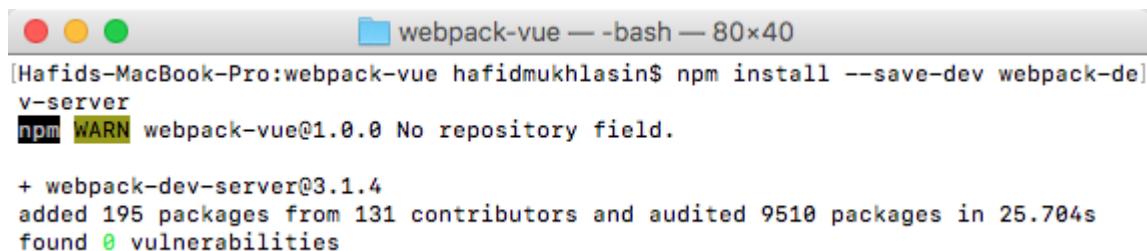
Vue on Webpack
Hello dari vue-loader!

Yes!

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
```



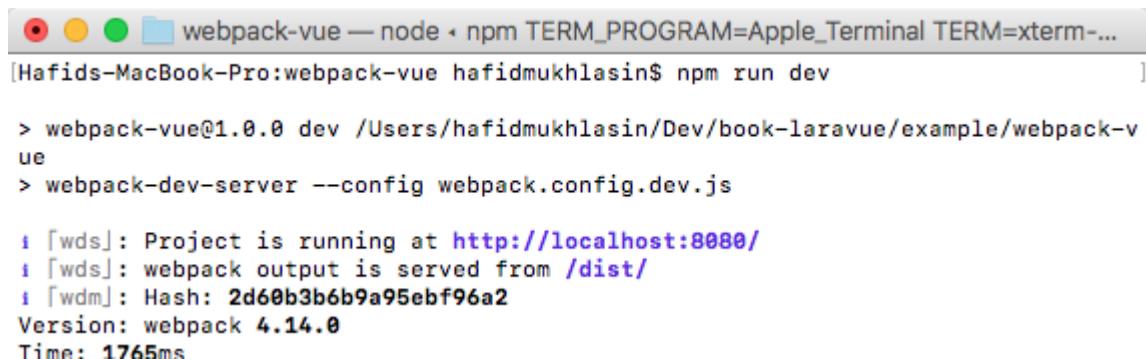
```
[Hafids-MacBook-Pro:webpack-vue hafidmukhlasin$ npm install --save-dev webpack-de] v-server
npm WARN webpack-vue@1.0.0 No repository field.

+ 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`.

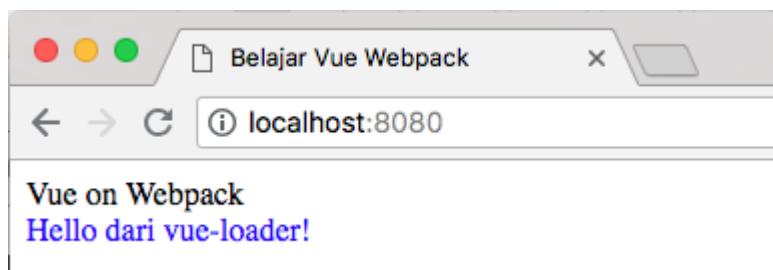
```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "dev": "webpack-dev-server --config webpack.config.dev.js"
},
```

Kemudian jalankan lagi perintah `npm run dev`, perintah ini disamping mem-bundle file juga menjalankan web server development.



```
[Hafids-MacBook-Pro:webpack-vue hafidmukhlasin$ npm run dev
> webpack-vue@1.0.0 dev /Users/hafidmukhlasin/Dev/book-laravue/example/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 [wdm]: Hash: 2d60b3b6b9a95ebf96a2
Version: webpack 4.14.0
Time: 1765ms
```

Selanjutnya kita bisa mengakses melalui URL : `http://localhost:8080`



Awesome!

Untuk mengubah port default 8080 kita bisa tambahkan pengaturanya pada file `webpack.config.dev.js` key `devServer` sebagai berikut.

```
module.exports = {
  entry: './src/app.js',
  output: {
    ...
  },
  module: {
    ...
  },
  ...
  devServer: {
    port: 9000
  }
}
```

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`

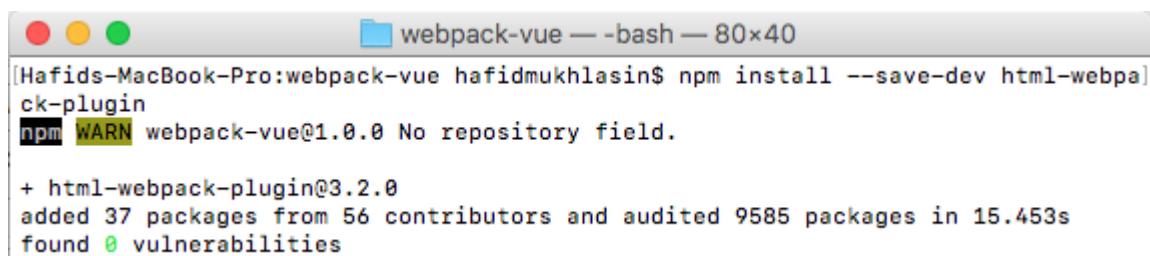
Keren kan?

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>). Untuk menginstalasinya jalankan perintah berikut.

```
npm install vue-hot-reload-api --save-dev
```



```
[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.

```
const path = require('path')
const VueLoaderPlugin = require('vue-loader/lib/plugin')
const webpack = require('webpack')
```

```
module.exports = {
  mode: 'development',
  entry: './src/app.js',
  output: {
    path: path.resolve(__dirname, 'dist'),
    publicPath: '/dist/',
    filename: 'build.js'
  },
  resolve: {
    alias: {
      'vue$': 'vue/dist/vue.esm.js',
    }
  },
  module: {
    rules: [
      {
        test: /\.js$/,
        exclude: /node_modules/,
        use: {
          loader: "babel-loader"
        },
      },
      {
        test: /\.vue$/,
        loader: 'vue-loader'
      },
      {
        test: /\.css$/,
        use: [
          'vue-style-loader',
          'css-loader'
        ]
      }
    ]
  },
  plugins: [
    new VueLoaderPlugin(),
    new webpack.HotModuleReplacementPlugin(),
  ],
  devServer: {
    port: 9000,
    hot: true
  }
}
```

Jalankan pada `npm run dev` dan lihat hasilnya pada browser.



Vue on Webpack

Hello dari vue-loader! Otomatis berubah ya?

```

[ ] Elements | Console | Sources | Network | Performance | Memory | Application | > | : X
[ ] top | Filter | Default levels ▾ | Group similar | Settings

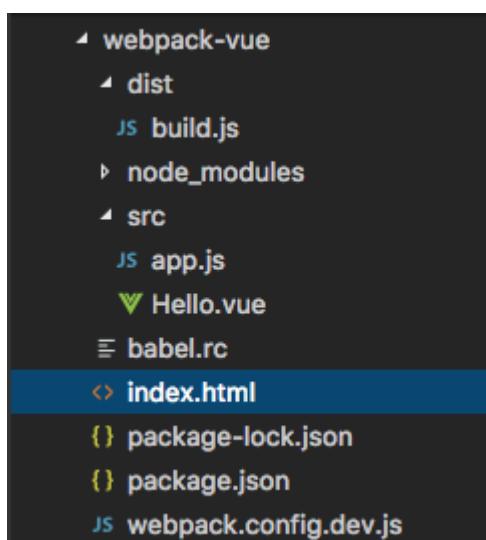
[HMR] Waiting for update signal from WDS... log.js:24
You are running Vue in development mode.
Make sure to turn on production mode when deploying for production.
See more tips at https://vuejs.org/guide/deployment.html vue.esm.js:8554
[WDS] Hot Module Replacement enabled. client:71
② [WDS] App updated. Recompiling... client:74
[WDS] App hot update... client:217
[HMR] Checking for updates on the server... log.js:24
[HMR] Updated modules: log.js:24
▶ [HMR] - ./src/Hello.vue?vue&type=script&lang=js log.js:16
[HMR] - ./src/Hello.vue?vue&type=script&lang=js log.js:24
[HMR] - ./src/Hello.vue log.js:24
[HMR] App is up to date. log.js:24

```

Ketika kita melakukan perubahan pada component Hello.vue maka otomatis akan dilakukan kompilasi ulang dan perubahannya seketika bisa dilihat pada browser.

Gokil kan?

Dari hasil coding 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 Javascript 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.

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.

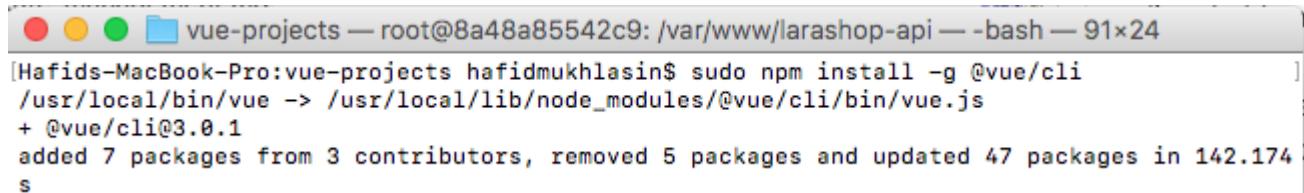
Saat ini Vue CLI sudah mencapai versi 3 dengan berbagai perubahan dan fitur baru salah satunya adalah menyediakan user interface berbasis web untuk menyiapkan projek kita. Vue CLI versi ini menggunakan plugins arsitektur yang bisa ditambahkan dan dikurangi, plugin tersebut telah diprekonfigurasi menurut 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

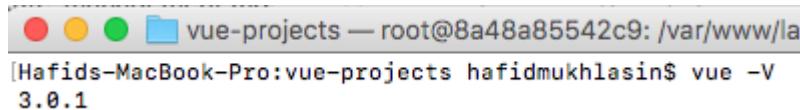
```
yarn global add @vue/cli
```



```
vue-projects — root@8a48a85542c9: /var/www/larashop-api — bash — 91x24
[Hafids-MacBook-Pro:vue-projects hafidmukhlasin$ sudo npm install -g @vue/cli
/usr/local/bin/vue -> /usr/local/lib/node_modules/@vue/cli/bin/vue.js
+ @vue/cli@3.0.1
added 7 packages from 3 contributors, removed 5 packages and updated 47 packages in 142.174
s
```

Untuk memastikan bahwa vue-cli berhasil terinstalasi dengan baik maka jalankan perintah berikut yang seharusnya akan menampilkan versi dari vue-cli.

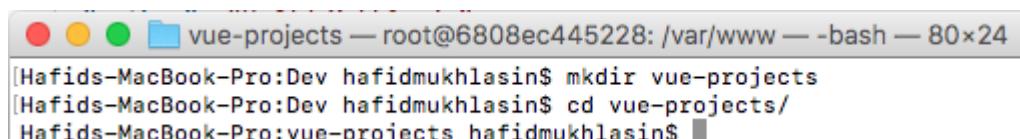
```
vue -V
```



```
vue-projects — root@8a48a85542c9: /var/www/la
[Hafids-MacBook-Pro:vue-projects hafidmukhlasin$ vue -V
3.0.1
```

Create New Project

Sebelum kita membuat projek baru menggunakan Vue CLI, 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.



```
vue-projects — root@6808ec445228: /var/www — bash — 80x24
[Hafids-MacBook-Pro:Dev hafidmukhlasin$ mkdir vue-projects
[Hafids-MacBook-Pro:Dev hafidmukhlasin$ cd vue-projects/
[Hafids-MacBook-Pro:vue-projects hafidmukhlasin$
```

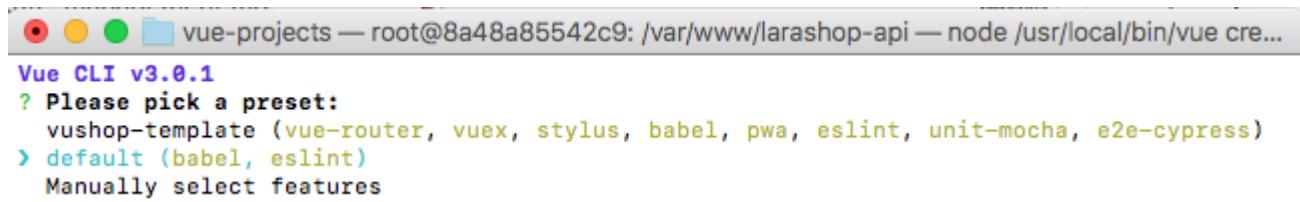
Untuk membuat projek baru dengan menggunakan vue-cli ini maka cukup jalankan perintah dengan format berikut.

```
vue create <project-name>
```

Contoh:

```
vue create myproject
```

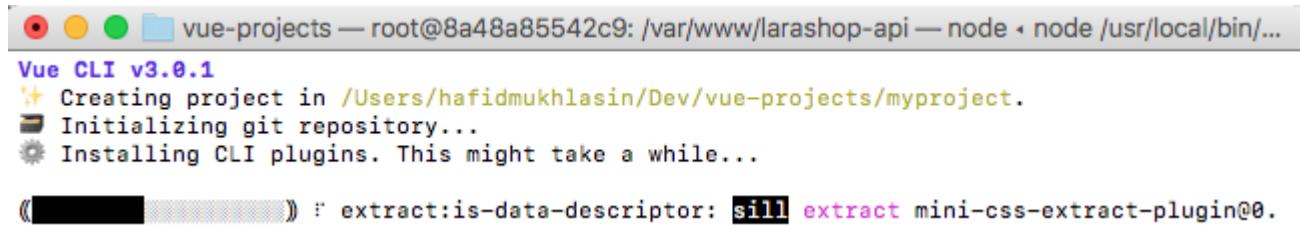
Perintah di atas akan menampilkan panduan interaktif (wizard) untuk menginstalasi dan mempersiapkan tools-tools yang diperlukan untuk pengembangan aplikasi berbasis Vue. Hasilnya akan diletakkan pada folder **myproject** sebagaimana parameter yang dituliskan pada perintah **vue create**.



```
vue-projects — root@8a48a85542c9: /var/www/larashop-api — node /usr/local/bin/vue cre...
Vue CLI v3.0.1
? Please pick a preset:
  vushop-template (vue-router, vuex, stylus, babel, pwa, eslint, unit-mocha, e2e-cypress)
> default (babel, eslint)
  Manually select features
```

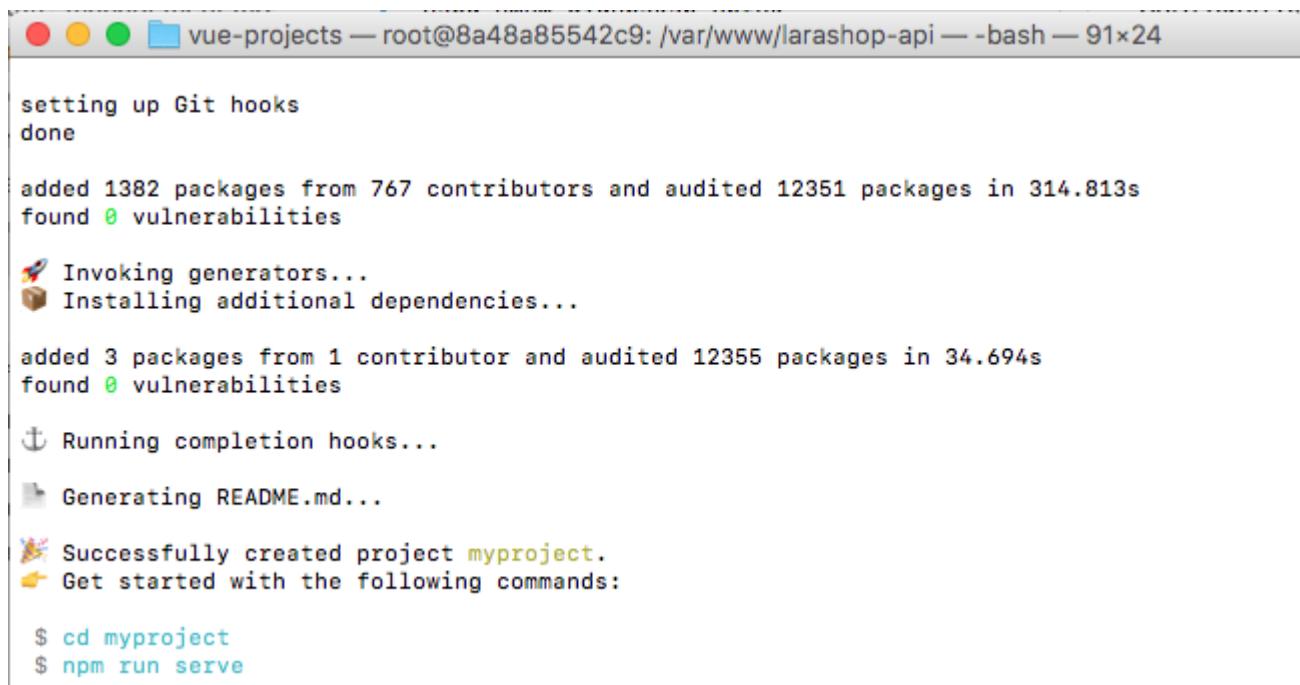
Minimal ada dua pilihan preset (pre konfigurasi), apakah default atau pilih secara manual fiturnya. Jika kita pilih default preset maka sebuah projek baru akan dibuat dan pustaka yang umum digunakan untuk pengembangan aplikasi berbasis Vue akan diinstalasi.

Pertama, kita coba dulu yang default preset.



```
vue-projects — root@8a48a85542c9: /var/www/larashop-api — node /usr/local/bin/vue create myproject
Vue CLI v3.0.1
✖ Creating project in /Users/hafidmukhlasin/Dev/vue-projects/myproject.
⠼ Initializing git repository...
⚙️ Installing CLI plugins. This might take a while...
[██████████] ⚡ extract:is-data-descriptor: sill extract mini-css-extract-plugin@0.
```

tldr;



```
vue-projects — root@8a48a85542c9: /var/www/larashop-api — bash — 91x24

setting up Git hooks
done

added 1382 packages from 767 contributors and audited 12351 packages in 314.813s
found 0 vulnerabilities

⚡ Invoking generators...
📦 Installing additional dependencies...

added 3 packages from 1 contributor and audited 12355 packages in 34.694s
found 0 vulnerabilities

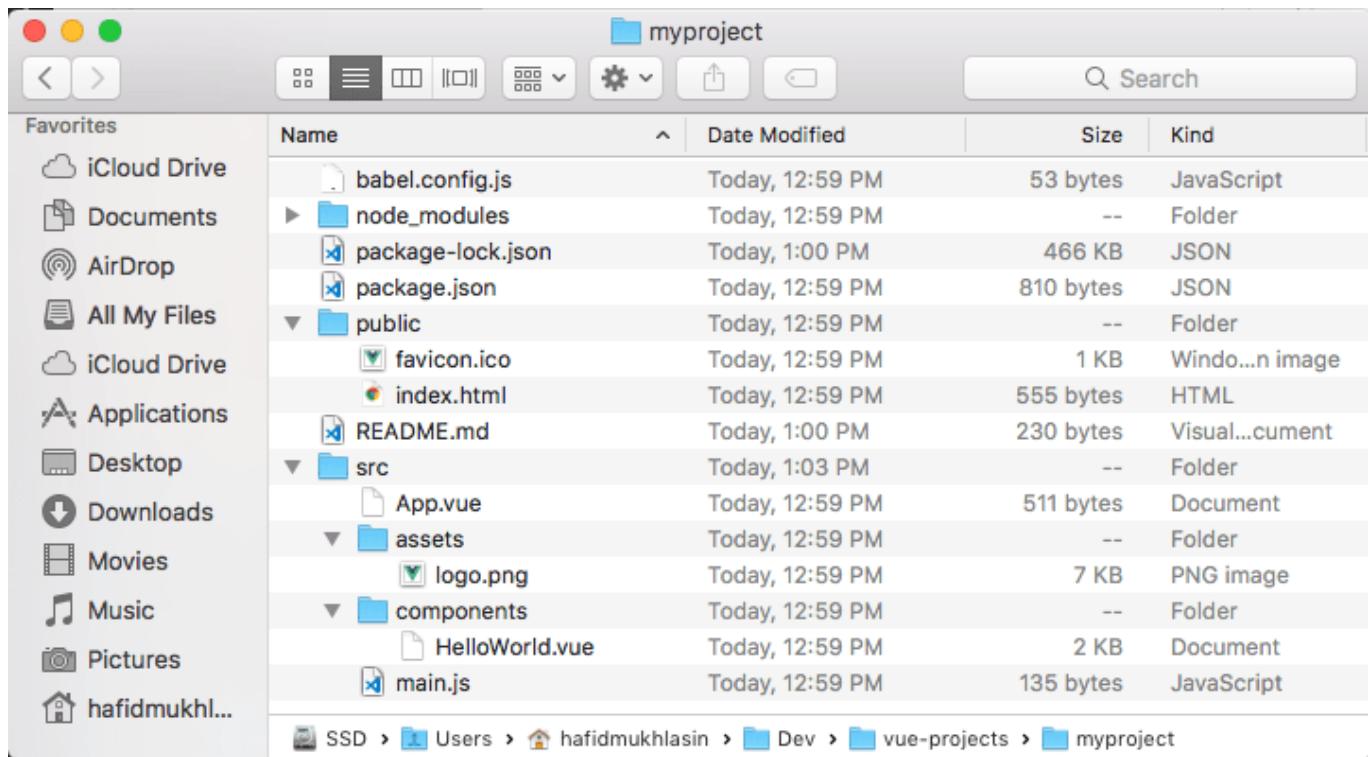
⚓ Running completion hooks...

📄 Generating README.md...

🎉 Successfully created project myproject.
👉 Get started with the following commands:

$ cd myproject
$ npm run serve
```

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.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width,initial-scale=1.0">
```

```
<link rel="icon" href="<%= BASE_URL %>favicon.ico">
<title>myproject</title>
</head>
<body>
<noscript>
<strong>We're sorry but myproject doesn't work properly without
JavaScript enabled. Please enable it to continue.</strong>
</noscript>
<div id="app"></div>
<!-- built files will be auto injected --&gt;
&lt;/body&gt;
&lt;/html&gt;</pre>
```

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 Javascript tersebut nantinya akan diinject ke dalam file index.html ini ketika projek di jalankan.

File berikutnya adalah main.js pada folder `src`.

```
import Vue from 'vue'
import App from './App.vue'

Vue.config.productionTip = false

new Vue({
  render: h => h(App)
}).$mount('#app')
```

Tentu kamu tidak asing dengan kode di atas? itu hanya sedikit modifikasi dari bentuk ini.

```
new Vue({
  el: '#app',
  components: { App },
  template: '<App/>'
})
```

Terkait hal ini, juga telah dibahas pada bab awal. Silakan dibaca lagi ya 😊.

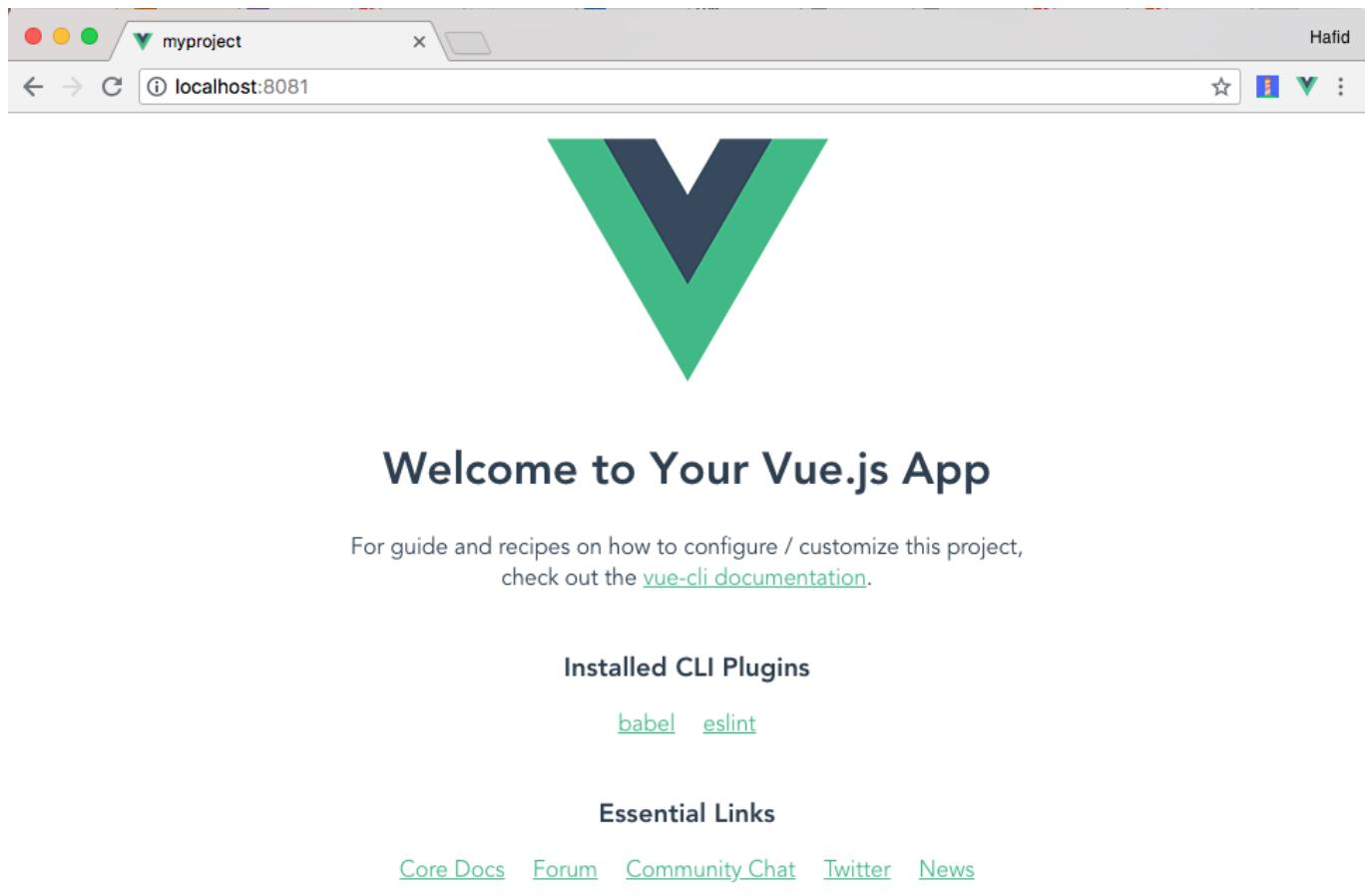
File berikutnya masih dalam folder `src` adalah `App.vue` yang merupakan component utama atau wrapper dari semua component yang nantinya digunakan pada projek aplikasi kita. Pada contoh ini component `HelloWorld` (`src/components/HelloWorld.vue`) diimport pada component `App.vue`, demikian juga nanti 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`

```
myproject — root@8a48a85542c9: /var/www/larashop-api — node -e npm TERM_PROGRAM=HyperTerm  
[Hafids-MacBook-Pro:vue-projects hafidmukhlasin$ cd myproject/  
[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 3721ms  
1:06:41 PM  
  
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.
```

Akses pada browser `http://localhost:8081`



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)

```
myproject — root@6808ec445228: /var/www — bash — 80x37
[Hafids-MacBook-Pro:myproject hafidmukhlasin$ npm run build
> myproject@0.1.0 build /Users/hafidmukhlasin/Dev/vue-projects/myproject
> vue-cli-service build

.. Building for production...

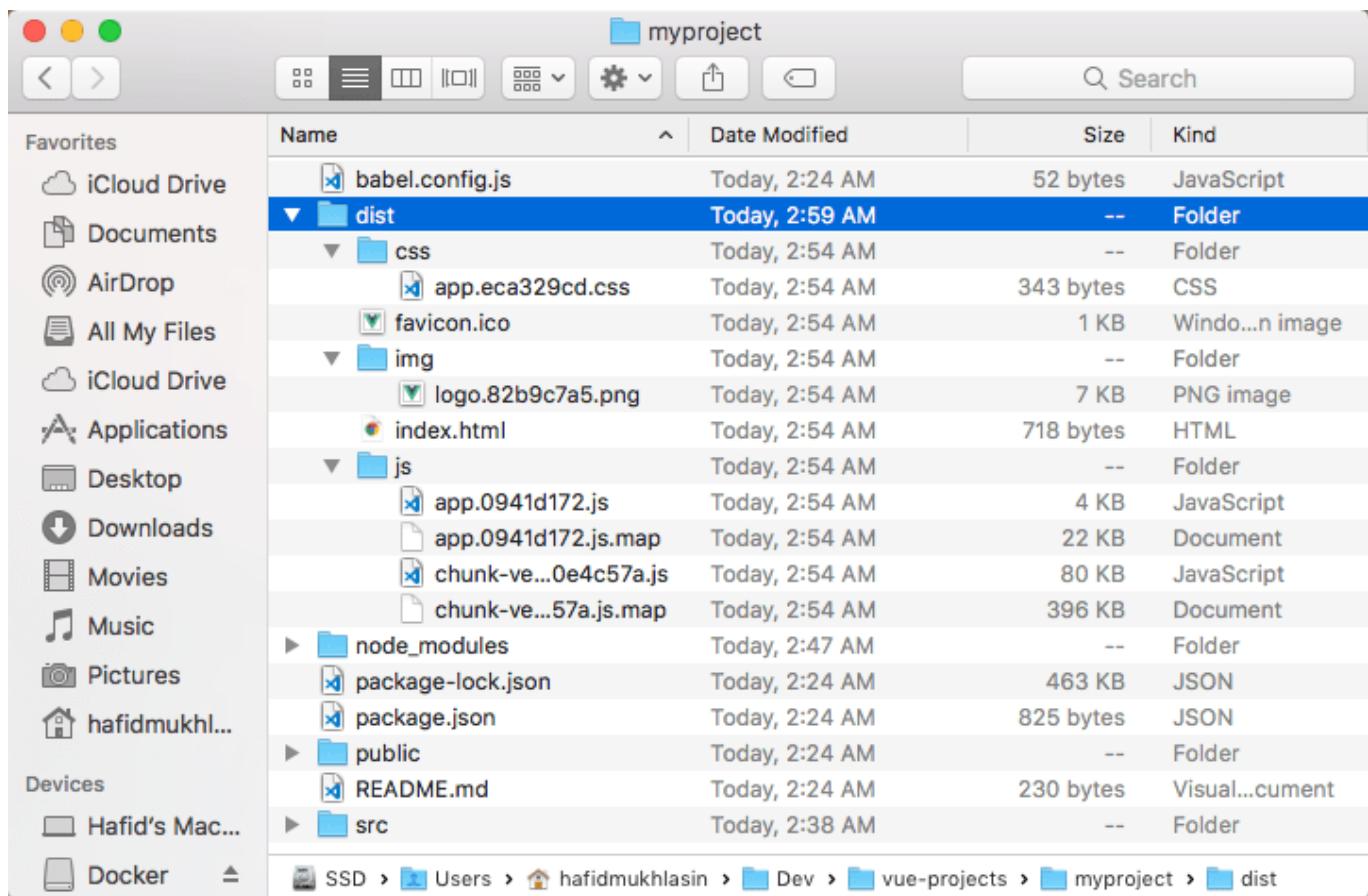
[DONE] Compiled successfully in 7184ms 2:54:40 AM

File Size Gzipped
dist/js/chunk-vendors.90e4c57a.js 78.23 kb 28.24 kb
dist/js/app.0941d172.js 4.37 kb 1.62 kb
dist/css/app.eca329cd.css 0.33 kb 0.23 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 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
```

```
myproject — root@6808ec445228: /var/www — node /usr/local/bin/serve -s d...
[Hafids-MacBook-Pro:myproject hafidmukhlasin$ npm install -g serve
/usr/local/bin/serve -> /usr/local/lib/node_modules/serve/bin/serve.js
+ serve@9.4.0
added 73 packages from 39 contributors in 7.749s
[Hafids-MacBook-Pro:myproject hafidmukhlasin$ serve -s dist
ERROR: DNS lookup failed: getaddrinfo ENOTFOUND Hafids-MacBook-Pro.local
```

```
Serving!
Local: http://localhost: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 projek 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>

```
vue-projects — root@8a48a85542c9: /var/www/larashop-api — node /usr/local/bin/vue ui...
[Hafids-MacBook-Pro:vue-projects hafidmukhlasin$ vue ui
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/project/select>

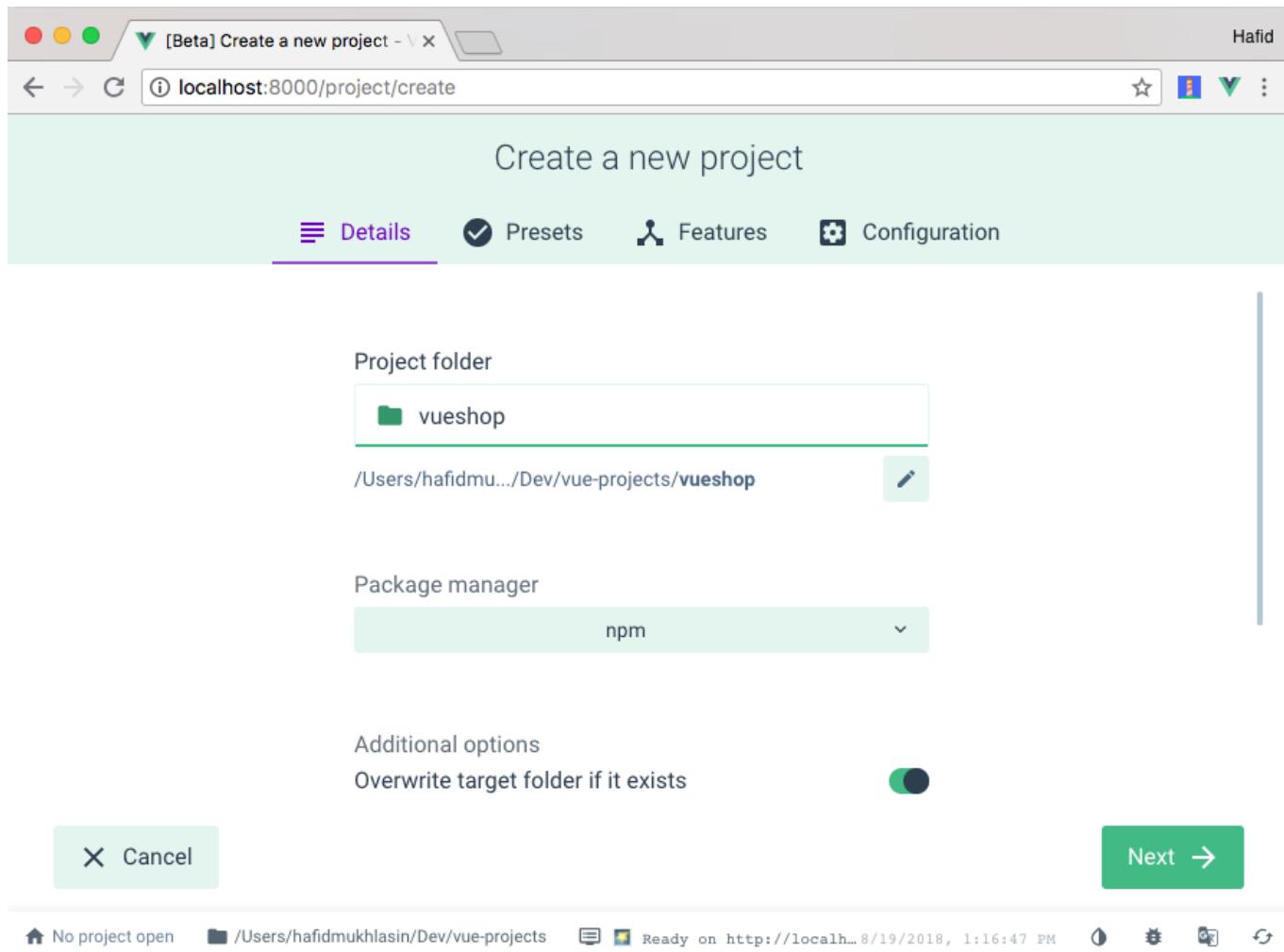
The screenshot shows the Vue Project Manager application interface. At the top, there's a header bar with the title '[Beta] Vue Project Manager' and a user name 'Hafid'. Below the header is a navigation bar with three tabs: 'Projects' (selected), 'Create', and 'Import'. The main content area features a large paperclip icon and the text 'No existing projects'. At the bottom, there's a footer bar with various icons and the URL 'localhost:8000/project/select'.

Projek yang pernah kita buat sebelumnya yaitu myproject akan muncul karena kebetulan lokasi direktori yang kita gunakan untuk menjalankan perintah `vue ui` berada pada folder `vue-projects`.

Pertama kita akan membuat projek baru. Pilih tab create. Tentukan lokasi direktori dimana projek kita akan disimpan.

The screenshot shows the Vue Project Manager application interface with the 'Create' tab selected. The main content area features a green button with the text '+ Create a new project here'. Below it is a file browser interface with a sidebar showing 'Users' and 'hafidmukhlasin'. The current directory is 'Dev/vue-projects'. At the bottom, there's a footer bar with various icons and the URL 'localhost:8000/project/select'.

Tekan tombol `Create a new project here`. Maka akan muncul wizard untuk membuat prjek baru. Masukkan nama folder projek kita `vueshop`, pilih package manager `npm`



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

A preset is an association of plugins and configurations. After you've selected features, you can optionally save it as a preset so that you can reuse it for future projects, without having to reconfigure everything again.

Select a preset

Default preset
babel, eslint

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 http://localhost:8100 8/19/2018, 1:16:47 PM

Klik **Next**, enable semua fitur kecuali **TypeScript** dan E2E Testing.

Catatan: **TypeScript** merupakan penulisan kode Javascript 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

i You will be able to add features after the project is created by installing plugins.

Enable features

Babel
Transpile modern JavaScript to older versions (for compatibility)

TypeScript
Add support for the TypeScript language More Info

Progressive Web App (PWA) Support
Improve performances with features like Web manifest and Service workers More Info

Router
Structure the app with dynamic pages More Info

Vuex

← Previous Next →

No project open /Users/hafidmukhlasin/Dev/vue-projects Ready... 7/30/2018, 3:17:22 AM

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

Vuex
Manage the app state with a centralized store [More Info](#)

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 →

No project open /Users/hafidmukhsin/Dev/vue-projects Ready... 7/30/2018, 3:17:22 AM

Klik **Next**,

Pada tab configuration

- 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**,

The screenshot shows a web-based configuration interface for creating a new project. At the top, there are tabs for 'Details', 'Presets' (which is selected), 'Features', and 'Configuration'. Below the tabs, a note says 'Use history mode for router? (Requires proper server setup for index fallback in production)' with a toggle switch set to 'on'. Under 'Pick a CSS pre-processor:', it says 'PostCSS, Autoprefixer and CSS Modules are supported by default.' with a dropdown menu set to 'Stylus'. Under 'Pick a linter / formatter config:', it says 'Checking code errors and enforcing an homogeneous code style is recommended.' with a dropdown menu set to 'ESLint with error prevention only'. Under 'Pick additional lint features:', there are two options: 'Lint on save' (selected) and 'Lint and fix on commit'. Under 'Pick a unit testing solution:', it says 'Mocha + Chai' with a dropdown menu. At the bottom left is a 'Previous' button, and at the bottom right is a green 'Create Project' button.

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.

The screenshot shows a modal dialog box titled 'Save as a new preset'. It has a 'Preset name' input field containing 'vueshop-template'. Below the input field is a note: 'Save the features and configuration into a new preset'. At the bottom, there are three buttons: 'Cancel', 'Continue without saving', and a green 'Create a new preset' button.

Tekan tombol **Create a new preset**.

ing an homogeneous code style is
ESLint w

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

```
vue-projects — root@8a48a85542c9: /var/www/larashop-api — npm - node /usr/local/bin/...
[Hafids-MacBook-Pro:vue-projects hafidmukhlasin$ vue ui
  Starting GUI...
  Ready on http://localhost:8000
Auto cleaned 1 projects (folder not found).

Vue CLI v3.0.1
  Creating project in /Users/hafidmukhlasin/Dev/vue-projects/vueshop.
  Initializing git repository...
  Installing CLI plugins. This might take a while...

((.....)) : fetchMetadata: sill pacote range manifest for regjsparser@^0.1.4 fe
```

tldr;

```
vue-projects — root@8a48a85542c9: /var/www/larashop-api — node /usr/local/bin/vue ui...
setting up Git hooks
done

added 1456 packages from 1203 contributors and audited 14258 packages in 45.711s
found 0 vulnerabilities

  Invoking generators...
  Installing additional dependencies...

added 23 packages from 70 contributors and audited 14305 packages in 41.07s
found 0 vulnerabilities

  Running completion hooks...

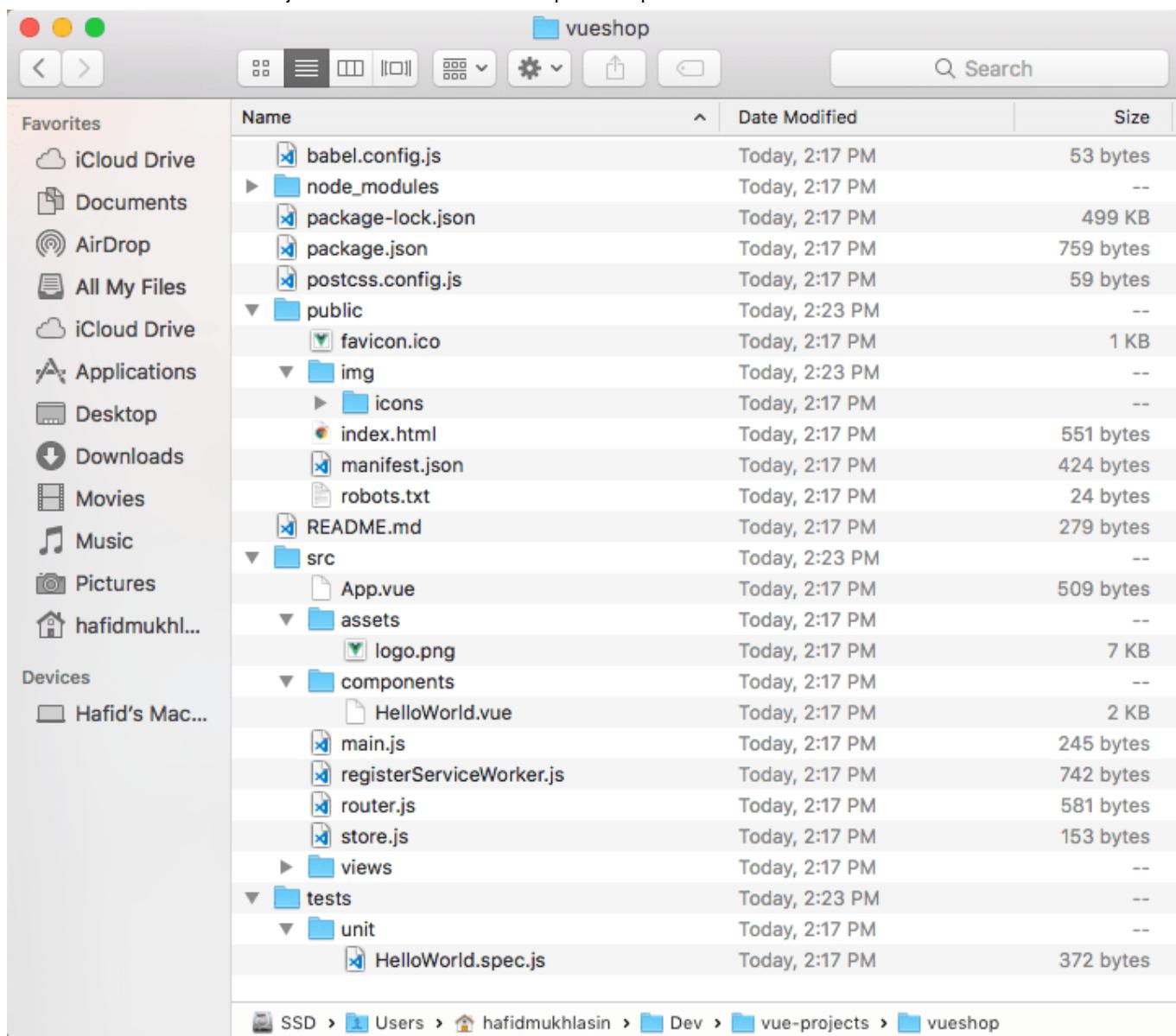
  Generating README.md...

  Successfully created project vueshop.
  Get started with the following commands:

  $ cd vueshop
  $ npm run serve
```

dan selesailah semua.

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`.

```
import Vue from 'vue'
import Router from 'vue-router'
import Home from './views/Home.vue'

Vue.use(Router)

export default new Router({
  mode: 'history',
  base: process.env.BASE_URL,
  routes: [
    {
      path: '/',
      name: 'home',
      component: Home
    },
  ],
})
```

```

{
  path: '/about',
  name: 'about',
  // route level code-splitting
  // this generates a separate chunk (about.[hash].js) for this route
  // which is lazy-loaded when the route is visited.
  component: () => import(/* webpackChunkName: "about" */ './views/About.vue')
}
]
})

```

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.

```

<template>
  <div id="app">
    <div id="nav">
      <router-link to="/">Home</router-link> |
      <router-link to="/about">About</router-link>
    </div>
    <router-view/>
  </div>
</template>
<style lang="stylus">
#app
  font-family 'Avenir', Helvetica, Arial, sans-serif
  -webkit-font-smoothing antialiased
  -moz-osx-font-smoothing grayscale
  text-align center
  color #2c3e50

#nav
  padding 30px
  a
    font-weight bold
    color #2c3e50
    &.router-link-exact-active
      color #42b983
</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

Licensed to M Najamudin Ridha - admin@komputerkampus.com - 087814493571 at 01/12/2018 19:54:30
bahasa stylus yaitu style penulisan CSS dengan menggunakan indentasi.

Pada views/Home.vue, dimuat components/HelloWorld.vue yang berisi link.

```
<template>
  <div class="home">
    
    <HelloWorld msg="Welcome to Your Vue.js App"/>
  </div>
</template>

<script>
// @ is an alias to /src
import HelloWorld from '@/components/HelloWorld.vue'

export default {
  name: 'home',
  components: {
    HelloWorld
  }
}
</script>
```

Sekarang waktunya menjalankan projek kita. Caranya: keluar dulu dari vue-ui dengan mengklik CTRL+C, masuk ke direktori **vueshop** dan jalankan perintah **npm run serve**

```
vueshop — root@8a48a85542c9: /var/www/larashop-api — node • npm TERM_PROGRAM=...
[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 4234ms                                         2:40:05 PM
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.
```

Hasilnya bisa kita lihat di browser.



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



Welcome to Your Vue.js App

For guide and recipes on how to configure / customize this project,
check out the [vue-cli documentation](#).

Klik halaman About maka akan muncul tampilan berikut.



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

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 vuetyf.

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 CSRF. 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 7 ✓	67 7 ✓	9 7 ✗	17 10 ✓	9 10.11 ✓
		10 8 ✓		
		11 8.1 ✓		

Untuk menginstalasi pustaka Axios sebagai plugins Vue CLI pada projek kita maka bisa melalui vue ui <http://localhost:8000/plugins>,

The screenshot shows the 'Project plugins' page of the Vue UI. On the left, there's a sidebar with icons for Home, Plugins, Components, Directives, Filters, Mixins, and Script. The main area has a title 'Project plugins' and a search bar. A green button labeled '+ Add plugin' is visible. The main content area is titled 'Installed plugins' and lists the following packages:

- @vue/cli-service (version 3.0.1, latest 3.0.1) - Official, Installed, local service for vue-cli projects. Status: ✓
- @vue/cli-plugin-babel (version 3.0.1, latest 3.0.1) - Official, Installed, babel plugin for vue-cli. Status: ✓
- @vue/cli-plugin-eslint (version 3.0.1, latest 3.0.1) - Official, Installed, eslint plugin for vue-cli. Status: ✓
- @vue/cli-plugin-pwa (version 3.0.1, latest 3.0.1) - Official, Installed, pwa plugin for vue-cli. Status: ✓
- @vue/cli-plugin-unit-mocha (version 3.0.1, latest 3.0.1) - Official, Installed, mocha unit testing plugin for vue-cli. Status: ✓

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 17.7K canhkieu

1

Cancel Search by algolia Install vue-cli-plugin-axios

17.7K canhkieu

Installing vue-cli-plugin-axios...

Pada tab configuration, klik **Finish installation**

Add a plugin

Search Configuration Files changed

Installation of vue-cli-plugin-axios

No configuration

Cancel Finish installation

Di akhir instalasi akan muncul konfirmasi bahwa file package.json akan dimodifikasi melalui penambahan dua pustaka yaitu Axios & vue-cli-plugin-axios.

Add a plugin

Search Configuration Files changed

Files changed 4

package-lock.json

package.json

Commit changes 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.

```
vueshop — root@8a48a85542c9: /var/www/larashop-api — node /usr/local/bin/vue ui — 91...
^C
Hafids-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.

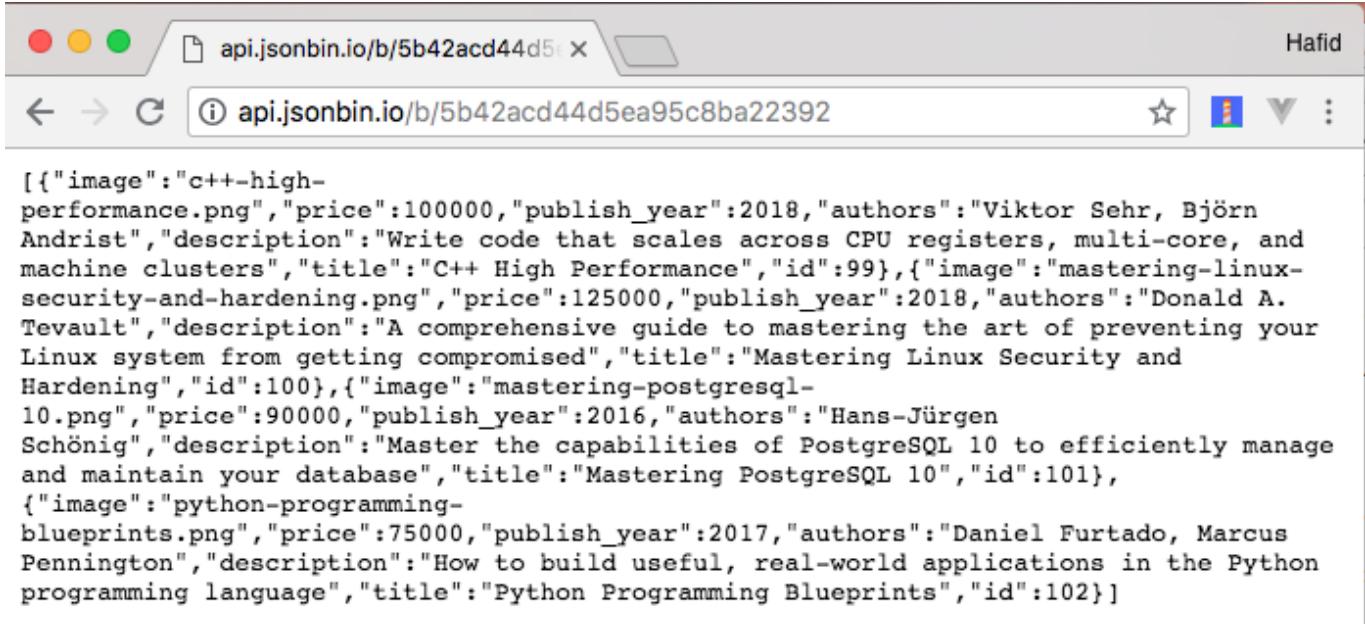
Plugin Axios bisa kita jumpai pada lokasi `src/plugins/axios.js`, di mana kita nantinya bisa melakukan konfigurasi jika diperlukan. Sedangkan pada file `src/main.js` yang merupakan file di mana objek Vue diciptakan telah di-*import* plugin tersebut.

Oleh karena itu, untuk menggunakannya maka kita bisa gunakan perintah `this.axios`

```
this.axios.get(URL_API_WEB_SERVICE)
  .then(function (response) {
    // handle success
  })
  .catch(function (error) {
    // handle error
  })
```

Sebagai contoh, kita akan gunakan data JSON buku sebagaimana yang telah digunakan pada bab terdahulu.

<http://api.jsonbin.io/b/5b42acd44d5ea95c8ba22392>



The screenshot shows a web browser window with the URL api.jsonbin.io/b/5b42acd44d5ea95c8ba22392. The page displays a JSON array of book objects. Each book has fields like image URL, price, publish year, authors, and description. The books listed are:

- C++ High Performance (id: 99)
- Mastering Linux Security and Hardening (id: 100)
- Mastering PostgreSQL 10 (id: 101)
- Python Programming Blueprints (id: 102)

Berikut contoh penggunaanya.

```
this.axios.get('http://api.jsonbin.io/b/5b42acd44d5ea95c8ba22392')
  .then((response) => {
    console.log(response.data)
  })
  .catch((error) => {
    console.log(error)
  })
}
```

Lebih sederhana dibandingkan menggunakan XMLHttpRequest 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.

The screenshot shows the 'Add a plugin' interface of the Vue CLI 3. At the top, there's a search bar with 'vuetify' typed in. Below it, two plugin entries are listed:

- 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

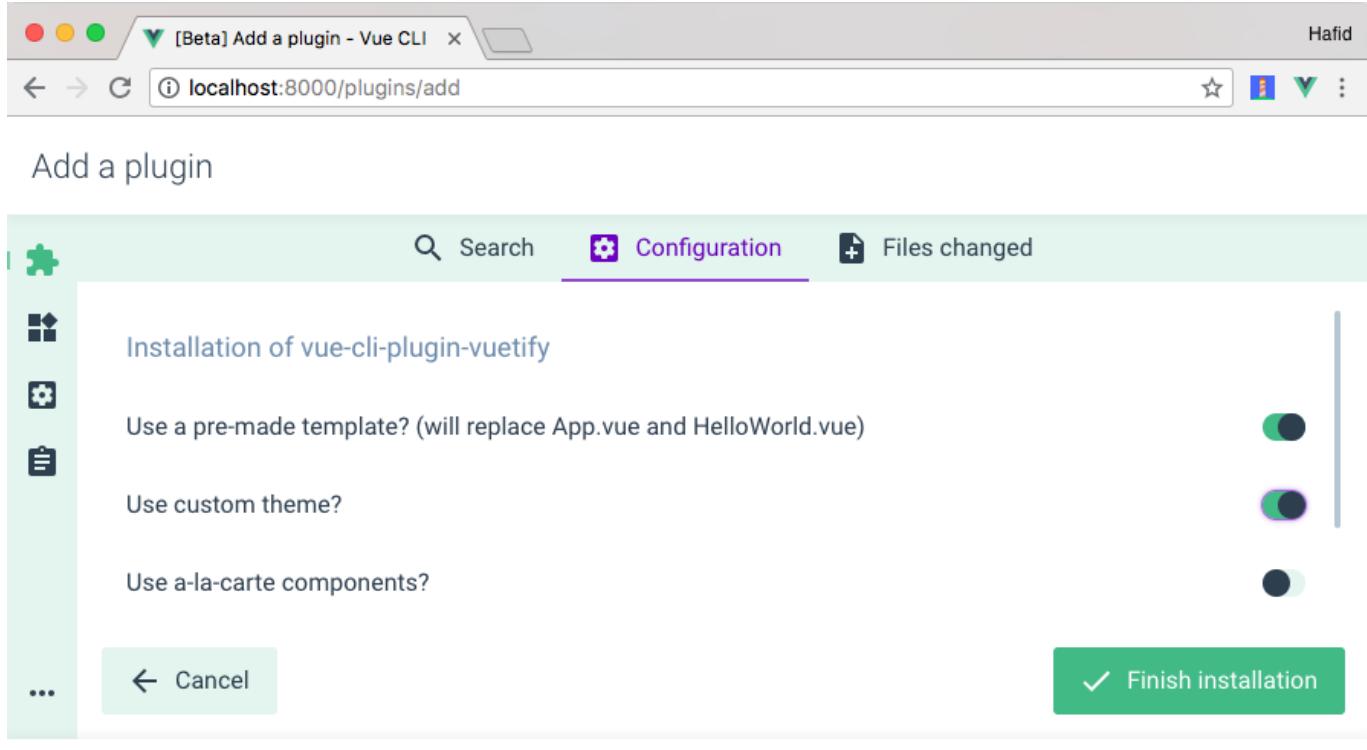
At the bottom right of the interface, there's a green button labeled 'Install vue-cli-plugin-vuetify' with a download icon. On the left side, there's a sidebar with icons for project management. A status bar at the bottom shows the path '/Users/hafidmukhlasin/Dev/vue-projects/vueshop' and a message 'You should review these changes... 8/19/2018, 2:45:13 PM'.

Catatan: jika menggunakan terminal, kita bisa menginstalasi dengan menjalankan perintah berikut `vue add vuetify`

A terminal window is shown with the command `vue add vuetify` being typed. The output below shows the progress of the installation:

```
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-based interface for managing code changes. At the top, there's a header with a search bar, configuration settings, and a 'Files changed' tab which is currently active. On the left, there's a sidebar with icons for file operations. The main area shows a list of files under 'Files changed' (9 items). One file, 'babel.config.js', is highlighted. 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.

This screenshot shows a modal dialog titled 'Commit changes'. Inside, there's a text input field labeled 'Enter a commit message' containing the text 'install vuetyfy'. Below the input field is a button labeled 'Record changes to the repository'. At the bottom of the dialog are two buttons: 'Cancel' on the left and a green 'Commit' button on the right.

Waktunya uji coba di browser, pastikan berada pada direktori `vueshop` kemudian jalankan `npm run serve`.

```
vueshop — root@6808ec445228: /var/www — node • npm TERM_PROGRAM...
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

[WARNING] Compiled with 1 warnings
4:45:32 AM

Module Warning (from ./node_modules/eslint-loader/index.js):
error: 'options' is defined but never used (no-unused-vars) at src/plugins/axios
.js:42:32:
40 | );
41 |
> 42 | Plugin.install = function(Vue, options) {
        ^
43 |     Vue.axios = _axios;
44 |     window.axios = _axios;
45 |     Object.defineProperties(Vue.prototype, {

1 error found.

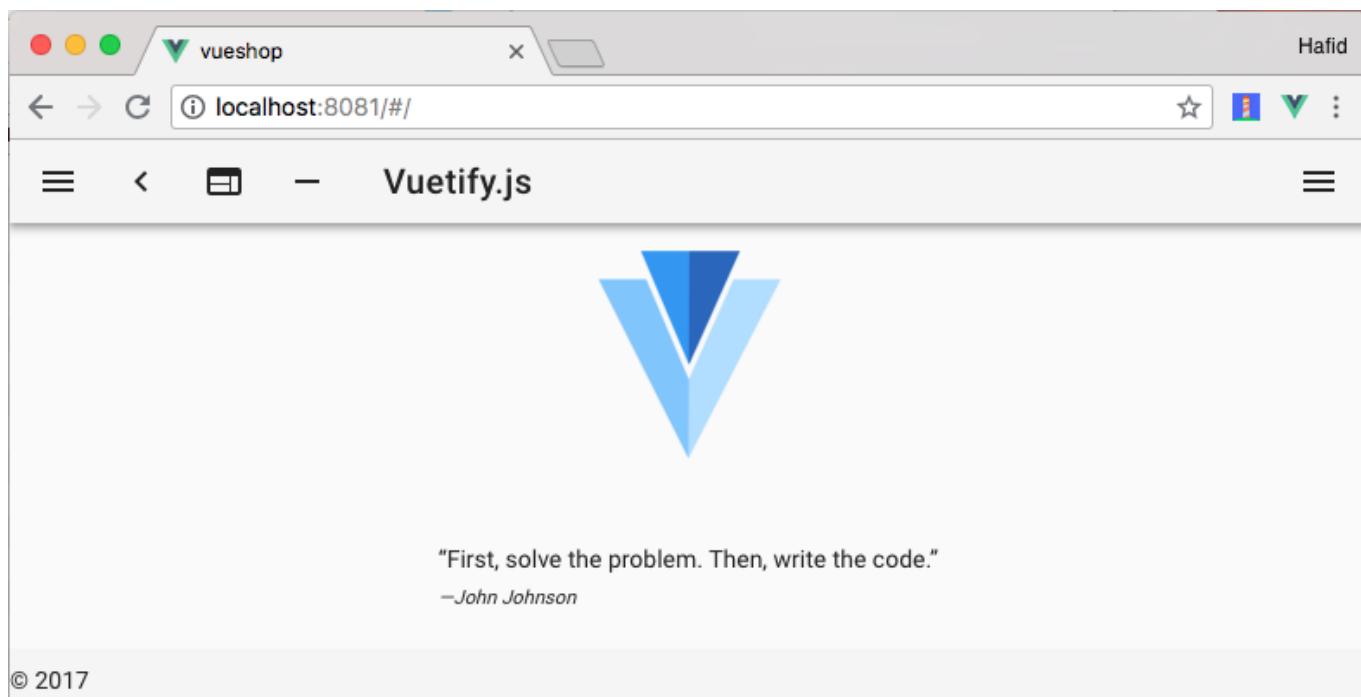
You may use special comments to disable some warnings.
Use // eslint-disable-next-line to ignore the next line.
Use /* eslint-disable */ to ignore all warnings in a file.

App running at:
- Local: http://localhost:8081/
- Network: http://192.168.43.127:8081/

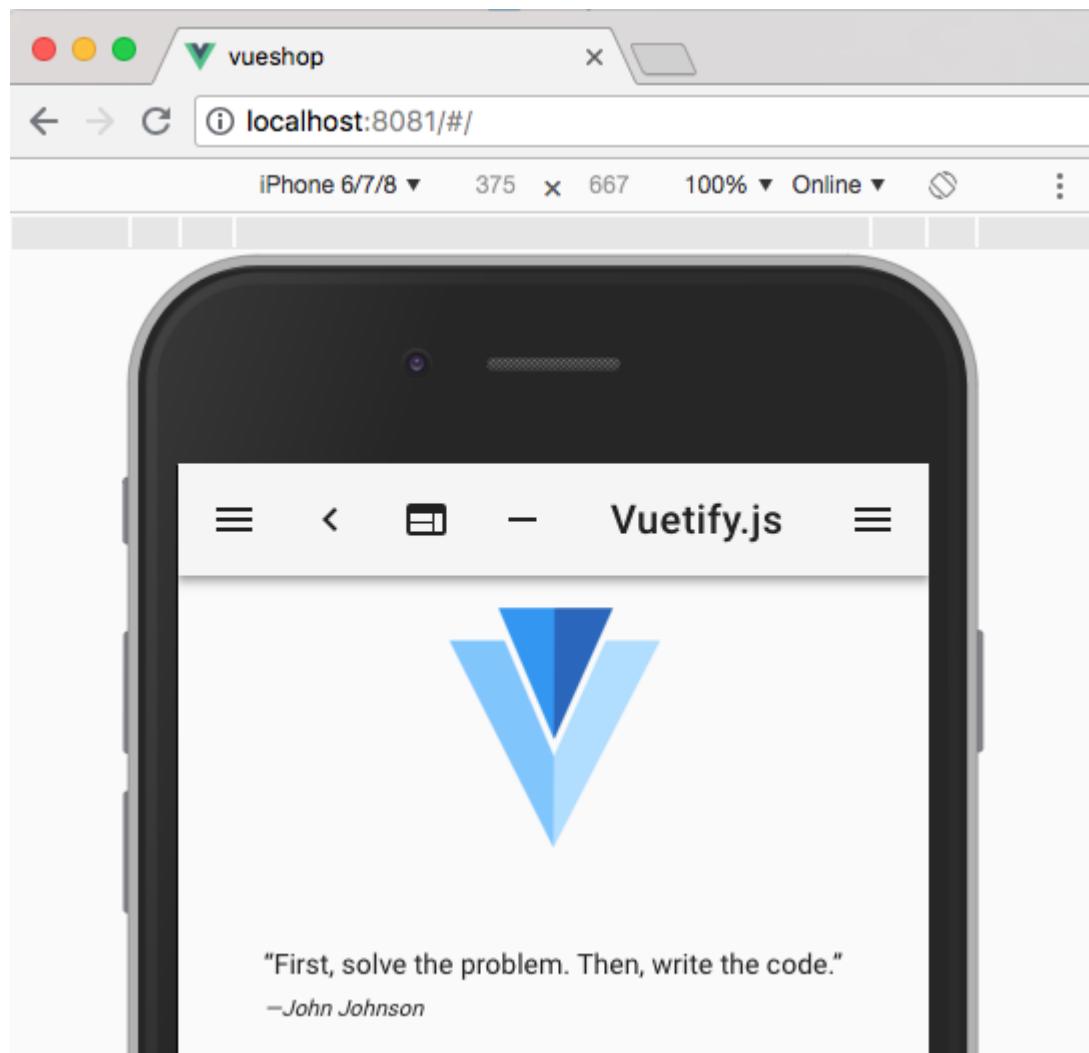
Note that the development build is not optimized.
To create a production build, run npm run build.
```

Ops, ada error `error: 'options' is defined but never used (no-unused-vars) at src/plugins/axios.js:42:32:`, intinya varibel options pada axios.js didefinisikan tapi tidak digunakan. Linter-lah yang melakukan pengecekan ini. Kita abaikan dulu error warning ini atau kamu boleh saja menghapus options pada file tersebut dan semua akan aman.

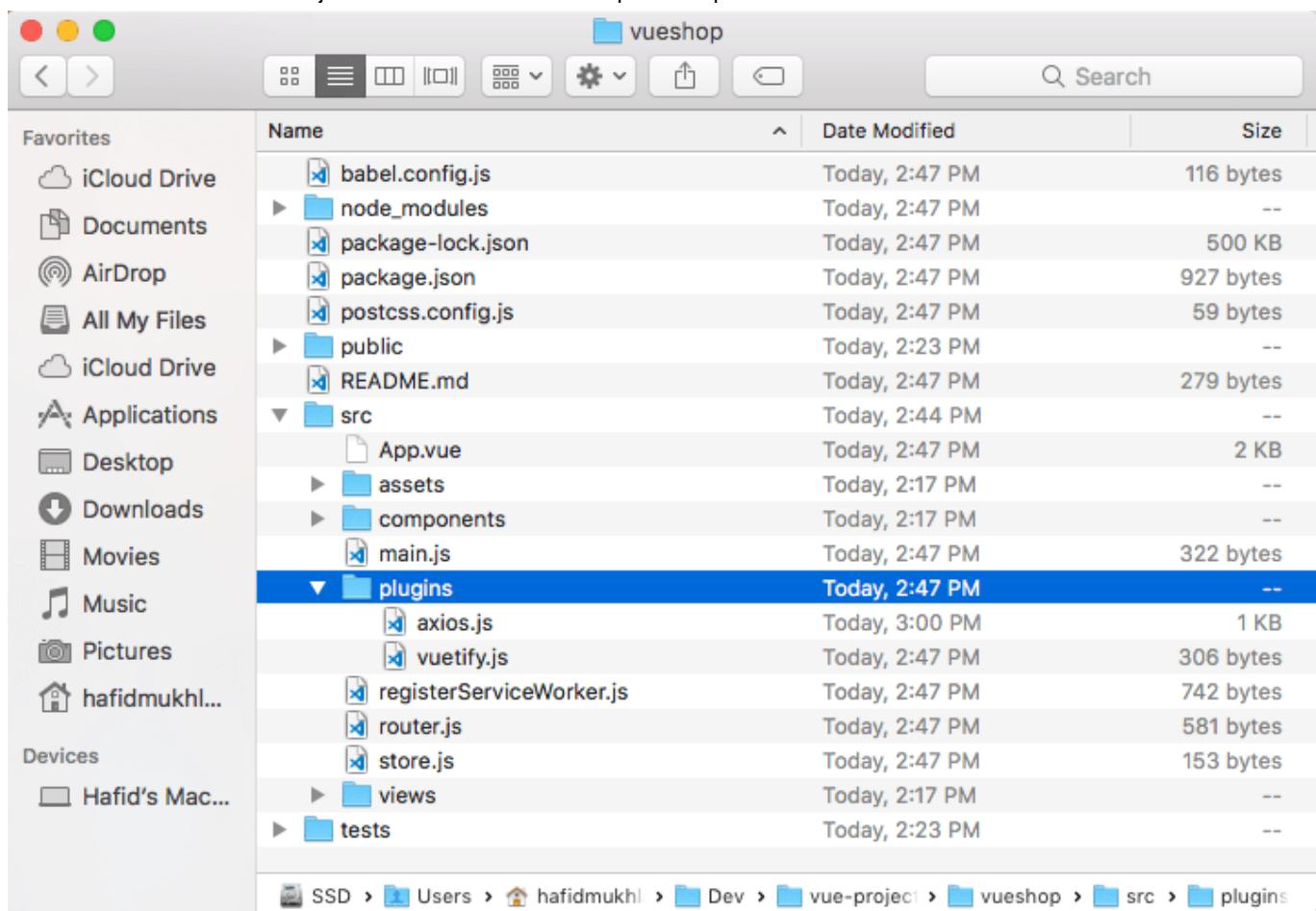
Mari kita coba akses melalui web browser.



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`

```
import Vue from 'vue'
import Vuetify from 'vuetify'
import 'vuetify/dist/vuetify.min.css'

Vue.use(Vuetify, {
  theme: {
    primary: '#ee44aa',
    secondary: '#424242',
    accent: '#82B1FF',
    error: '#FF5252',
    info: '#2196F3',
    success: '#4CAF50',
    warning: '#FFC107'
  }
})
```

Kita bisa menggunakan tools theme generator pada tautan ini <https://vuetifyjs.com/en/theme-generator>, untuk memilih warna yang kita sukai.

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>

Code	Status	Keterangan
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

Stateless 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

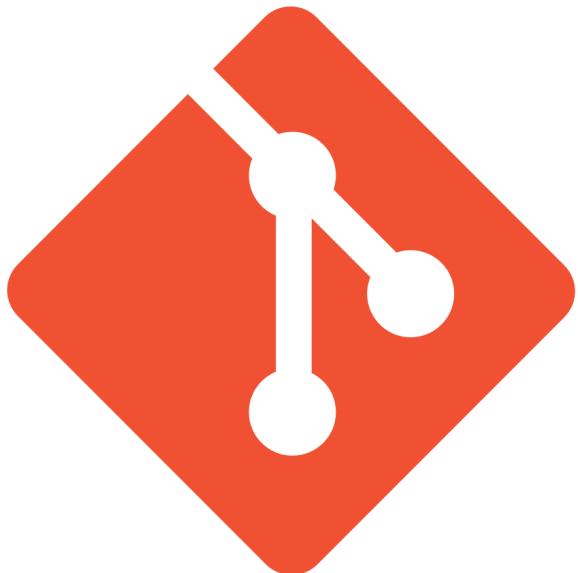
Web server merupakan software yang bertugas melayani request dari client yang dalam hal ini request dilakukan menggunakan browser dengan protokol http/s. PHP sendiri sebenarnya mempunyai built-in web server yang pada bab sebelumnya telah kita gunakan.

`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.

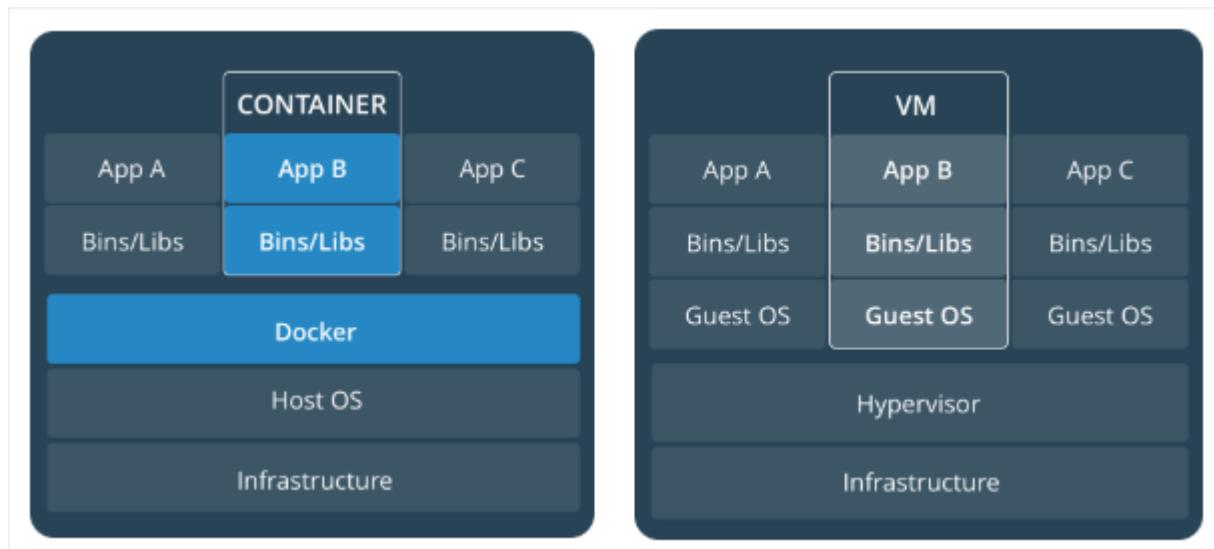
Ada banyak paket yang bisa kita gunakan, misalnya: Docker (Laradock), XAMPP, dan Homestead. Tentu saja jangan kamu install semua, melainkan pilih salah satu saja, adapun penulis menggunakan Docker.

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.

```
[Hafids-MacBook-Pro:book hafidmukhlasin$ docker -v
Docker version 18.03.1-ce, build 9ee9f40
Hafids-MacBook-Pro:book hafidmukhlasin$ ]
```

A screenshot of a Mac OS terminal window. The title bar says "book — root@c5ac5cd7e2ae: /var/www — -bash — 74x32". The command entered is "docker -v". The output shows "Docker version 18.03.1-ce, build 9ee9f40". The prompt "Hafids-MacBook-Pro:book hafidmukhlasin\$" is visible at the end of the line.

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$ ]]
```

A screenshot of a Mac OS terminal window. The title bar says "laravel-projects — root@6808ec445228: /var/www — -bash — 80x37". The commands entered are "mkdir laravel-projects" and "cd laravel-projects". The prompt "Hafids-MacBook-Pro:laravel-projects hafidmukhlasin\$" is visible at the end of the line.

Kemudian, pada terminal jalankan perintah berikut untuk mengcloning projek laradock.

```
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.

```
# Point to the path of your applications code on your host
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.

```
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).

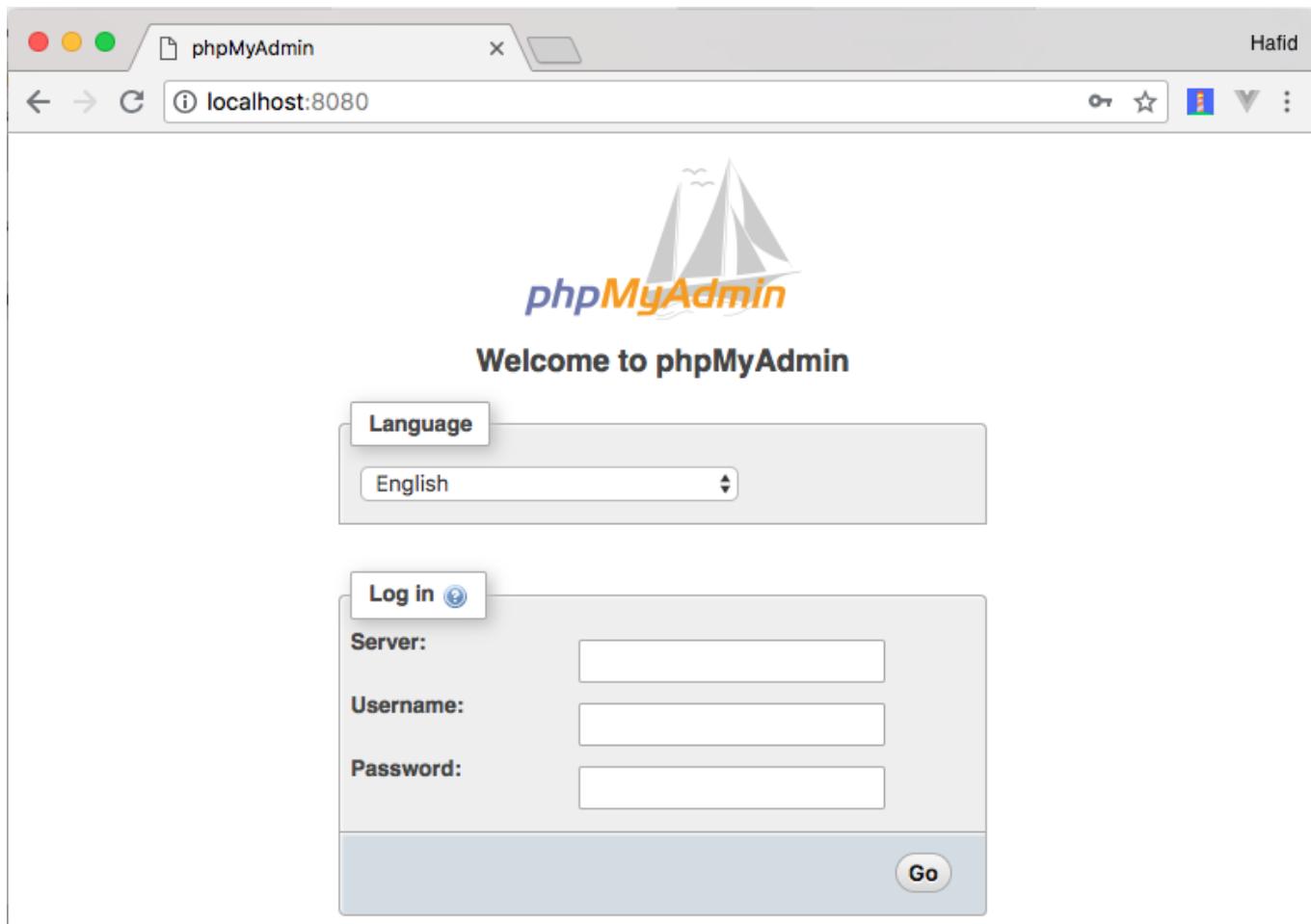
Setiap container merujuk ke image dari aplikasi yang dibungkusnya, misalnya container mysql berisi image aplikasi mysql yang oleh docker dihosting pada repositori docker image di <https://hub.docker.com> (tepatnya di https://hub.docker.com/_/mysql/). Oleh karena itu, ketika pertama kali kita menjalankan `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`

```
MYSQL_VERSION=latest
MYSQL_DATABASE=default
MYSQL_USER=default
MYSQL_PASSWORD=secret
MYSQL_PORT=3306
MYSQL_ROOT_PASSWORD=root
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`.

```
# MYSQL_VERSION=latest
MYSQL_VERSION=5.7
```

Kemudian, masih pada file `.env`, ubah lokasi di mana data mysql disimpan, dari `~/.laradock/data` menjadi misalnya: `~/.laradock/data2`.

```
# DATA_PATH_HOST=~/.laradock/data  
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 hafidmukhlasin$ 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 hafidmukhlasin$ ]
```

Lalu build ulang container mysql dengan menjalankan perintah berikut.

```
docker-compose build mysql
```

```
[Hafids-MacBook-Pro:laradock hafidmukhlasin$ 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

```
docker exec -it mysql bash
```

Kemudian login ke mysqlnya, lalu jalankan perintah berikut.

```
mysql -u root -p  
mysql> SET GLOBAL innodb_fast_shutdown = 1;
```

Ketika **exit** atau tekan CTRL+C untuk keluar dari container mysql, kemudian jalankan perintah mysql upgrade berikut.

```
mysql_upgrade -u root -p
```

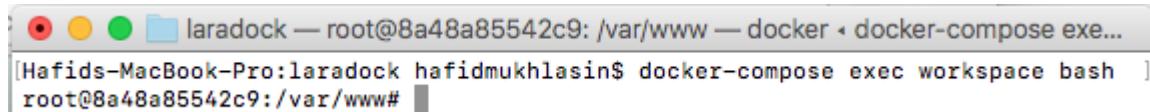
Dan silakan dicoba login lagi.

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.

```
docker-compose exec workspace bash
```



```
[Hafids-MacBook-Pro:laradock hafidmukhlasin$ docker-compose exec workspace bash
root@8a48a85542c9:/var/www# ]
```

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**

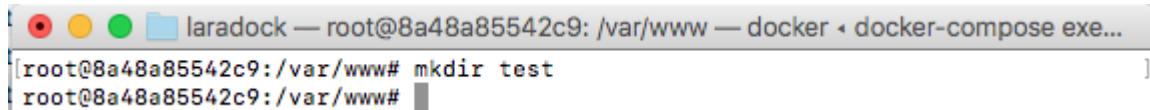
Catatan: **webroot** pada komputer penulis terletak di **~/Dev/laravel-projects/**.

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**.

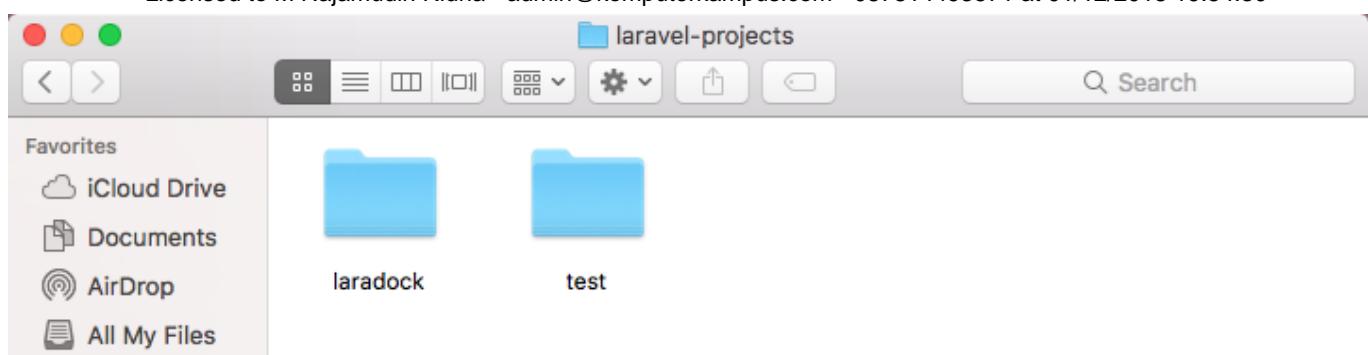
Uji Coba Membuat Projek Baru

Pada terminal workspace, buat folder **test** menggunakan perintah **mkdir**.



```
[root@8a48a85542c9:/var/www# mkdir test
root@8a48a85542c9:/var/www# ]
```

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.

```
<?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`

```
# root /var/www/public;
root /var/www;
```

Karena kita telah melakukan perubahan pada konfigurasi nginx, maka kita perlu merestart container nginx supaya perubahan tersebut diterapkan.

```
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' '--enable-fpm' '--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.

```
listen 80;
listen [::]:80;

server_name coba.test;
# root /var/www/public;
root /var/www/test;
```

```
⑥ 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`.

```
127.0.0.1    coba.test
```

```
##  
# 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/lib/php' '--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 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>.

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 pada `/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.

```
<?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/lamp/etc/httpd.conf`, cari kata "Virtual Host" dan pastikan konfigurasi virtual host tidak di-comment.

- Pada Windows

```
# Virtual hosts
Include C:\xampp\apache\conf\extra\httpd-vhosts.conf
```

- Pada MacOS

```
# Virtual hosts
Include /Applications/XAMPP/etc/extra/httpd-vhosts.conf
```

- Pada Ubuntu

```
# Virtual hosts
Include etc/extra/httpd-vhosts.conf
```

Kemudian tambahkan konfigurasi berikut pada file **httpd-vhosts.conf**.

```
<VirtualHost *:80>
    DocumentRoot C:\xampp\htdocs\test
    ServerName coba.test
    <Directory "C:\xampp\htdocs\test">
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

Catatan: sesuaikan lokasi directory **test**-nya.

Jangan lupa daftarkan virtual domain tersebut pada file **hosts** (pada bagian sebelumnya telah dibahas mengenai file ini).

```
127.0.0.1   coba.test
```

Restart Apache (XAMPP) melalui control panel,

Modules		Service	Module	PID(s)	Port(s)	Actions		
<input type="checkbox"/>	Apache			5764 6352	80, 443	Stop	Admin	Config
<input type="checkbox"/>	MySQL			4916	3306	Stop	Admin	Config
<input type="checkbox"/>	FileZilla					Start	Admin	Config

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/5.7/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.

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('SHA384', 'composer-setup.php') ===
'544e09ee996cdf60ece3804abc52599c22b1f40f4323403c44d44fdfdd586475ca9813a858
088ffbc1f233e9b180f061') { echo 'Installer verified'; } else { echo
'Installer corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"
```

```
php composer-setup.php  
php -r "unlink('composer-setup.php');"
```

atau bisa juga menggunakan CURL

```
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

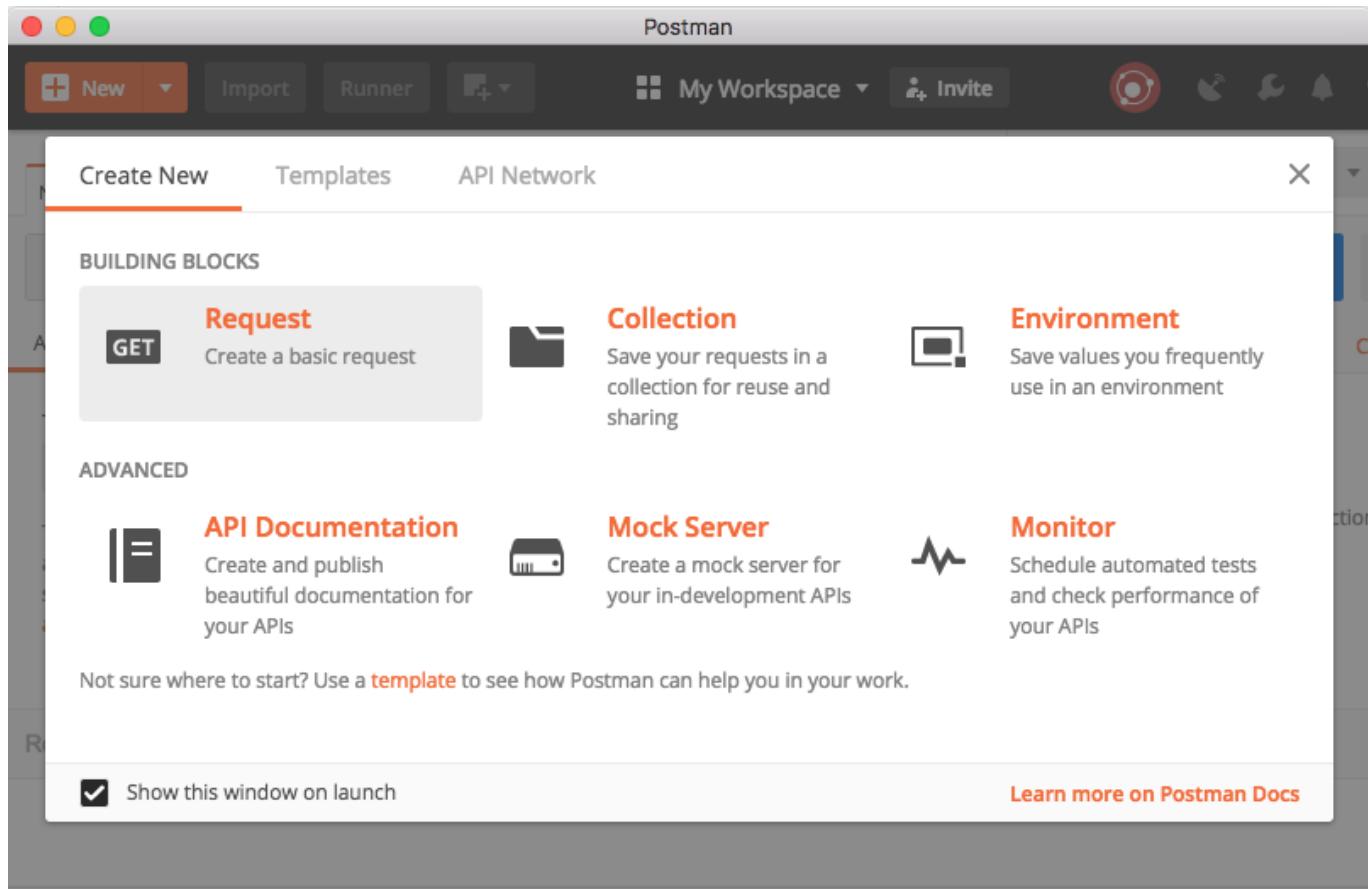
Postman adalah tools berbasis user interface yang digunakan untuk menguji coba suatu web service. Sehingga untuk sekedar mencoba apakah web service yang kita buat berjalan dengan baik atau tidak maka kita cukup menggunakan tools ini tanpa perlu membuat aplikasi clientnya dulu.



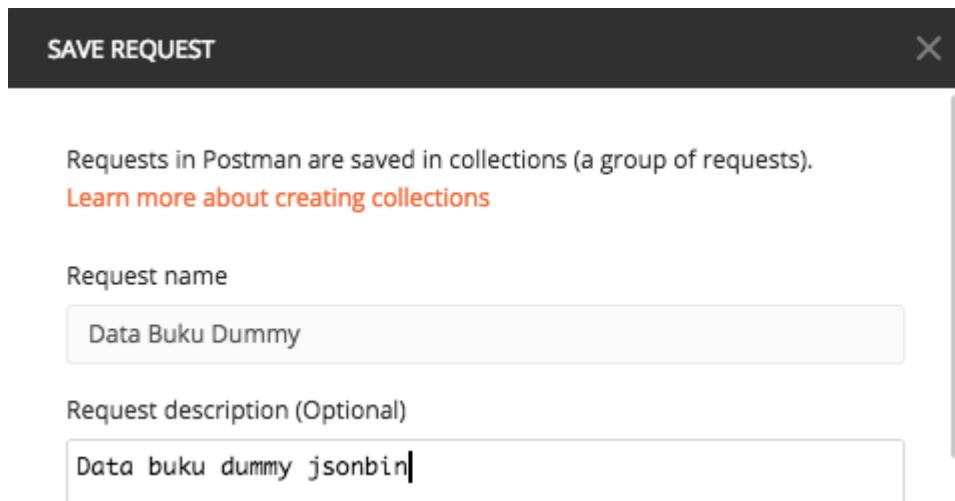
Postman tersedia untuk semua platform OS: Mac, Windows dan Linux. Untuk instalasinya silakan unduh file installernya pada tautan berikut: <https://www.getpostman.com/apps>

Penggunaan

Setelah menginstalasi Postman, jalankan Postman lalu pada halaman login, kamu bisa skip saja melalui menu bagian bawah. Maka akan muncul popup **Create New**.

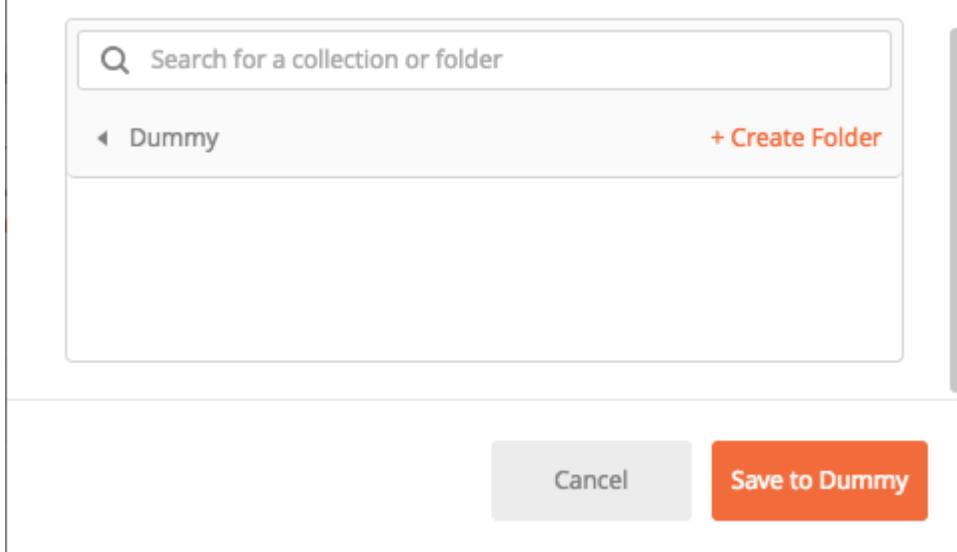


Pilih **Request** dan masukkan deskripsi dari URL web service yang ingin kita daftarkan. Tuliskan nama dan deskripsinya.



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   {
3     "image": "c++-high-performance.png",
4     "price": 100000,
5     "publish_year": 2018,
6     "authors": "Viktor Sehr, Björn Andrist",
7     "description": "Write code that scales across CPU registers, multi-core, and machine clusters",
8     "title": "C++ High Performance",
9     "id": 99
10   },
11   {
12     "image": "mastering-linux-security-and-hardening.png",
13     "price": 125000,

```

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**

The screenshot shows a software interface with a sidebar on the left containing several collections: 'Posts' (57 requests), 'Digit' (2 requests), 'Dump' (1 request), and 'Semantik' (6 requests). A context menu is open over the 'Posts' collection, listing options: Share Collection, Rename (⌘E), Edit, Add Request, Add Folder, Duplicate (⌘D), Export, Monitor Collection, Mock Collection, and Publish Docs. The 'Publish Docs' option is highlighted with a gray background. The main workspace on the right displays a table titled 'cd44d5ea95c8ba22392' with columns 'Request Script' and 'Tests'. Below this is a section titled 'Result' with a table header 'Value'. At the bottom of the workspace, there is some blue-highlighted text: 'resql-10.png', 'chönig', 'e capabilities of PreSQL 10', and 'Security and Harde'.

Pilih **Publish Collection**.

The screenshot shows the 'Publish Collection' dialog box from the Postman application. At the top, it says 'DUMMY ➔ PUBLISH COLLECTION'. Below that is the title 'Publish Collection' and a section labeled 'Dummy'. A horizontal line separates this from the configuration options. The first option is 'Choose an environment', which has a dropdown menu set to 'No Environment'. To the right of the dropdown, a note states: 'The shared environment will be made available with the collection's public documentation'. The second option is 'Select a custom domain', with a dropdown menu set to 'No custom domain'. To the right, a note states: 'The published collection will be made available at the selected domain'. The third option is 'Allow collection discovery', with an unchecked checkbox next to the text 'Allow other Postman users to discover this collection through the [API network](#)'. Below these options is a red link 'Show Custom Styling Options ▾'. At the bottom of the dialog are two buttons: 'Cancel' and a large orange 'Publish Collection' button. To the right of the 'Publish Collection' button is a note: '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 title bar says 'Edit Published Collection'. The main area displays the message 'Your collection is published! ✓' and 'Dummy, No Environment'. A 'Published URL' section shows the link <https://documenter.getpostman.com/view/255653/RWTfzMtG>. Below it are two buttons: 'Edit published collection' and 'Unpublish this collection'.

Lalu akses URL yang dihasilkan.

The screenshot shows the published collection 'DUMMY' as it appears in a web browser. The title is 'Dummy'. It contains a single endpoint: 'GET Data Buku Dummy' with the URL <http://api.jsonbin.io/b/5b42acd44d5ea95c8ba22392> and the response 'Data buku dummy jsonbin'. On the right, there's an 'Example Request' section with the 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 5.7.

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 5.7 sebagai berikut.

- PHP >= 7.1.3
- 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`

Ingin: 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 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.

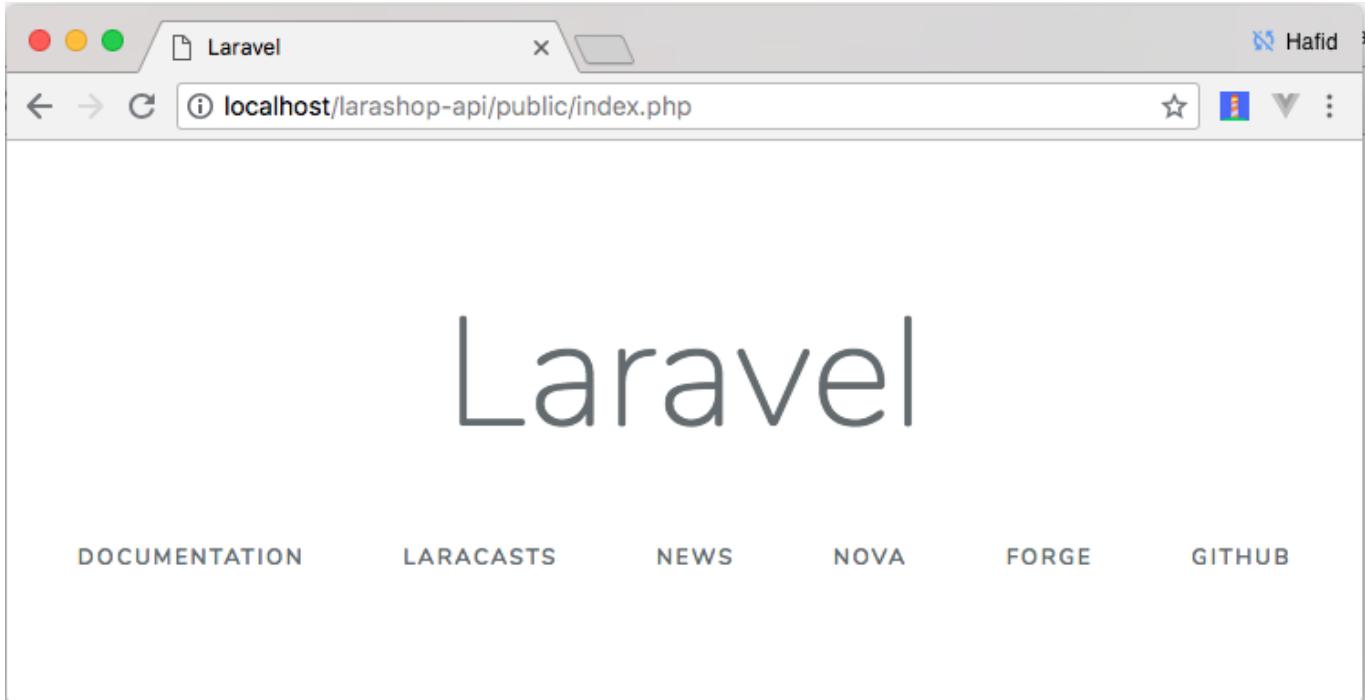
```
[Hafids-MacBook-Pro:laradock hafidmukhlasin$ docker-compose exec workspace bash      ]
[root@28de49479092:/var/www# composer create-project --prefer-dist laravel/laravel ]
larashop-api
Do not run Composer as root/super user! See https://getcomposer.org/root for details
ls
Installing laravel/laravel (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-code-coverage suggests installing ext-xdebug (^2.6.0)
phpunit/phpunit suggests installing phpunit/php-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.

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:K9MFukfG8flpHEtwWRdzzkuGG/P5nThKDqXRNFiTCI4=
APP_DEBUG=true
APP_URL=http://localhost

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=homestead
DB_USERNAME=homestead
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     listen 80;  
3     listen [::]:80;  
4  
5     server_name larashop-api.test;  
6     root /var/www/larashop-api/public;  
7     index index.php index.html index.htm;  
8  
9     location / {  
10        try_files $uri $uri/ /index.php$is_args$args;  
11    }  
12}
```

```
server {  
  
    listen 80;  
    listen [::]:80;  
  
    server_name larashop-api.test;  
    root /var/www/larashop-api/public;  
    index index.php index.html index.htm;  
  
    location / {  
        try_files $uri $uri/ /index.php$is_args$args;  
    }  
  
    ...
```

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>
    DocumentRoot C:\xampp\htdocs\larashop-api\public
    ServerName larashop-api.test
    <Directory "C:\xampp\htdocs\larashop-api\public">
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

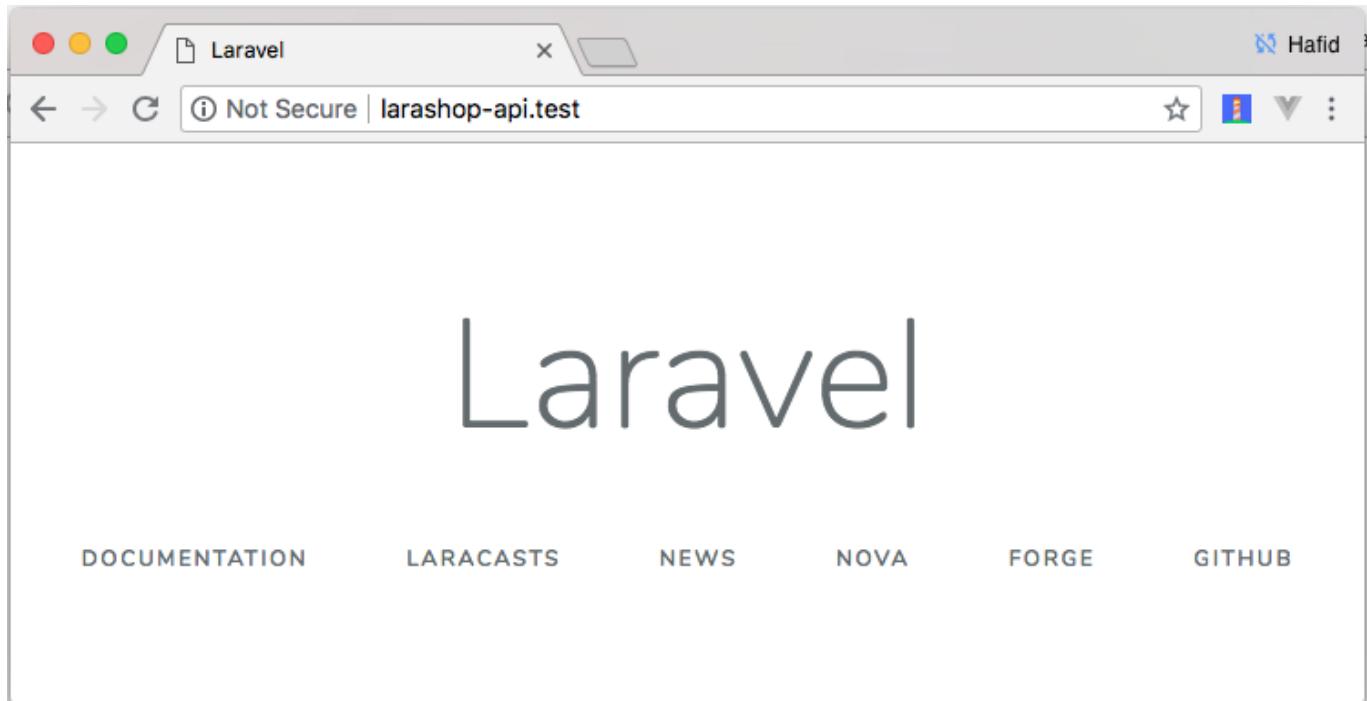
Supaya aplikasi kita mendukung pretty URL (URL yang SEO friendly) maka jika menggunakan XAMPP atau web server Apache, pastikan kita mengaktifkan module `mod_rewrite` pada PHP sehingga file `.htaccess` yang sudah disertakan oleh Laravel dalam folder `public` akan bekerja.

Kemudian, daftarkan virtual domain larashop-api.test pada file hosts.

```
laradock — root@8a48a85542c9: /var/www/laradock/nginx/sites — vim + sud...
## 
# 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.

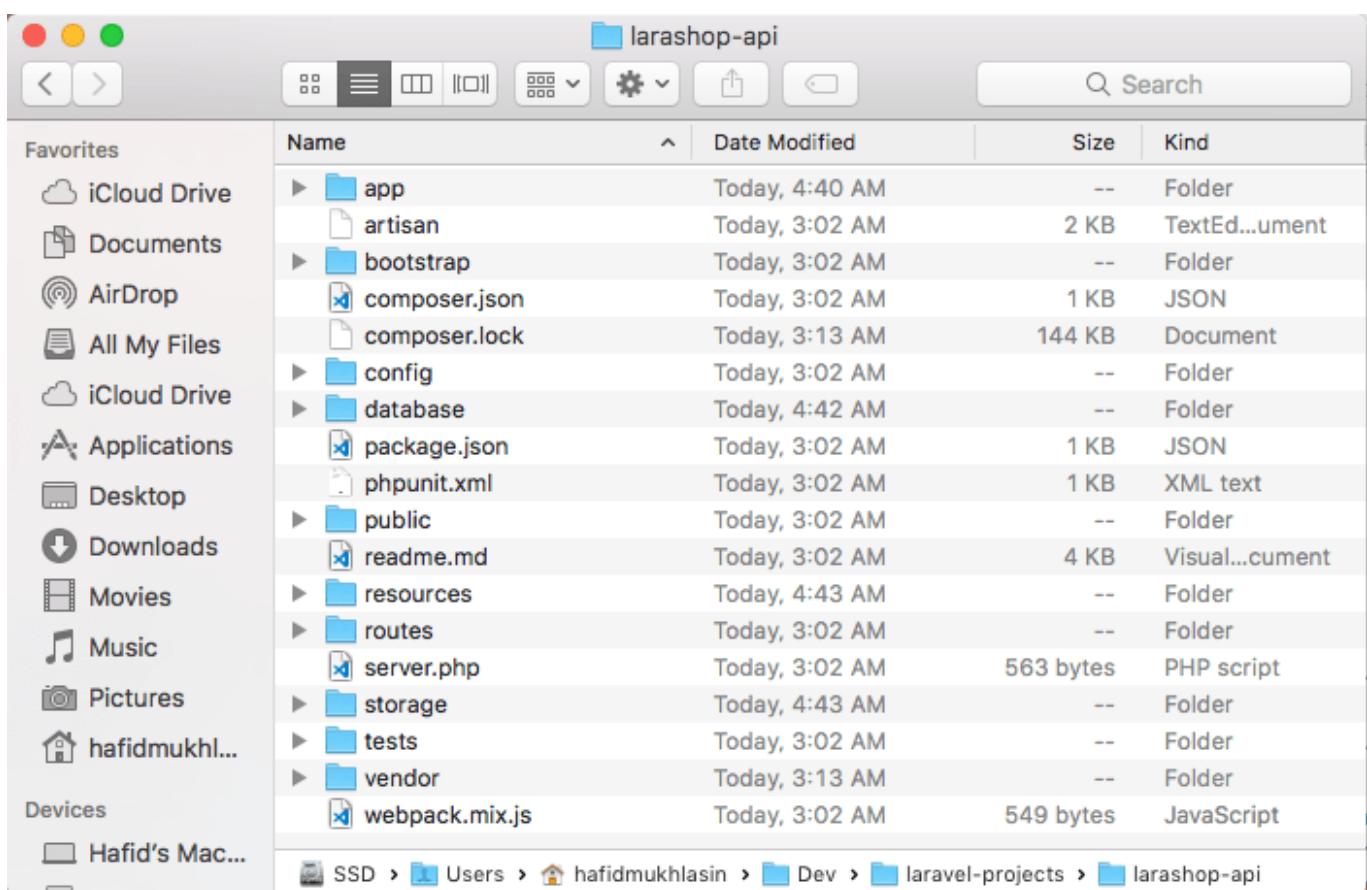
Pada laradock jalankan perintah `docker-compose restart nginx` atau pada XAMPP control panel stop dan start lagi service Apache. Lalu jika semua sudah kita lakukan maka pada browser, silakan akses <http://larashop-api.test>



horee!

Struktur Direktori Aplikasi

Struktur direktori projek aplikasi yang digenerate oleh Laravel adalah sebagai berikut.



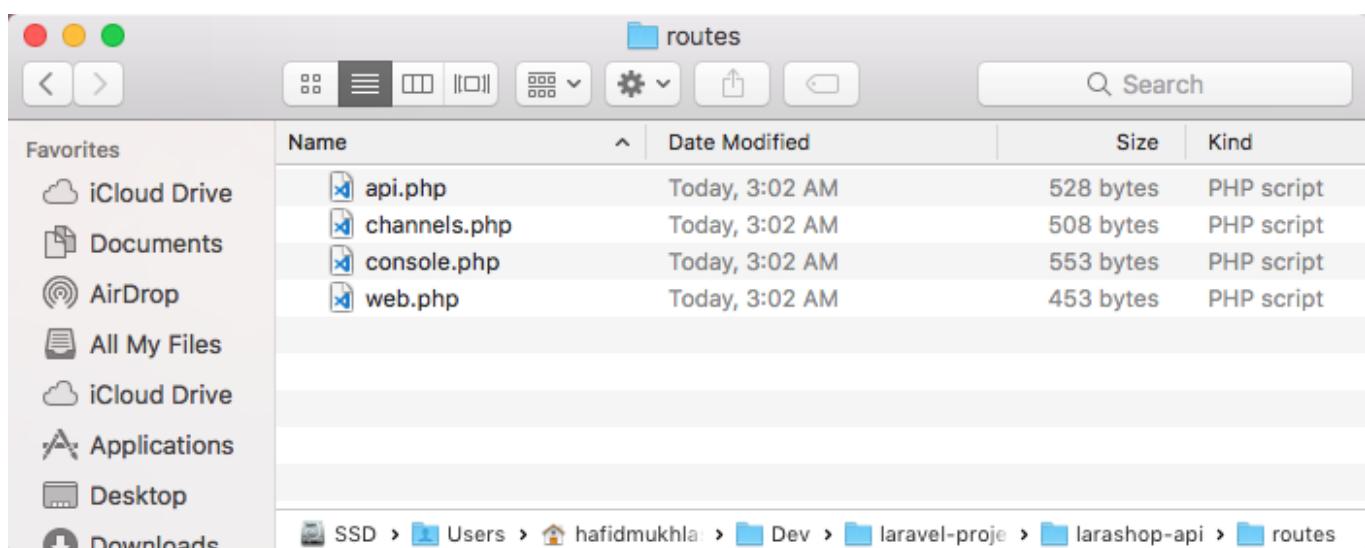
Struktur tersebut dapat dijelaskan sebagai berikut.

- **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.

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

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.

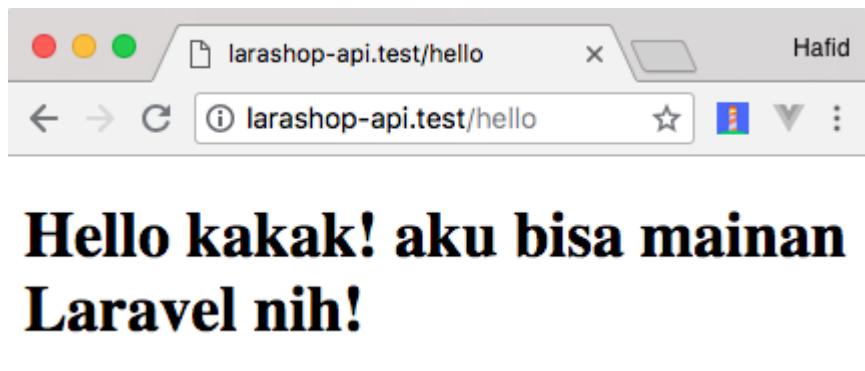
Merujuk hal tersebut maka kita bisa menambahkan kode routing berikut.

```
Route::get('hello', function () {
    return view('hello');
});
```

Kemudian buat file `hello.blade.php` pada direktori `resources/views/`

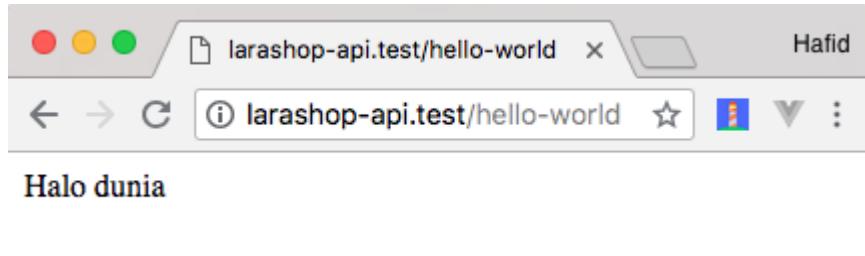
```
<!doctype html>
<html>
    <body>
        <h1>Hello kakak! aku bisa mainan Laravel nih!</h1>
    </body>
</html>
```

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.

```
Route::get('hello-world', function () {
    return 'Halo dunia';
});
```



Routing web ini terhubung dengan group middleware (mekanisme untuk filtering http request) yang menyediakan fitur seperti session dan proteksi CSRF.

Routing API (Web Service)

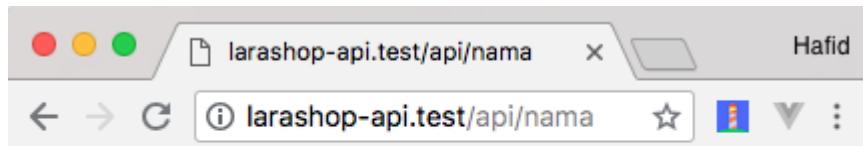
Perbedaan utama dengan routing web dengan routing api ini adalah sifatnya yang stateless (tanpa session). Sebagaimana yang telah dijelaskan sebelumnya bahwa kode routing untuk web service disimpan dalam file `api.php`, yang jika kita buka file tersebut maka terdapat kode berikut.

```
Route::middleware('auth:api')->get('/user', function (Request $request) {
    return $request->user();
});
```

Nah sebelum kita memahami maksud dari kode di atas, mari kita membuat aturan routing baru pada file tersebut. Misalnya:

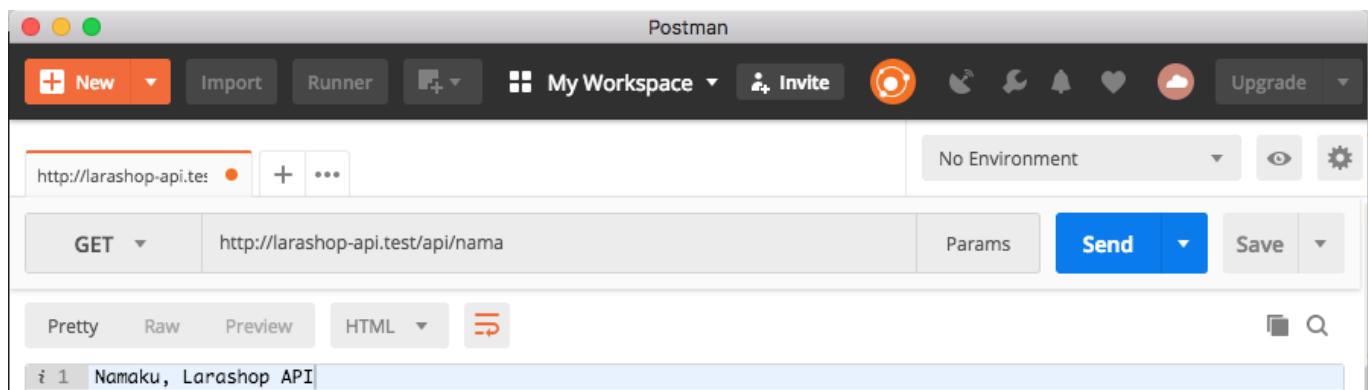
```
Route::get('nama', function () {
    return 'Namaku, Larashop API';
});
```

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`



Namaku, Larashop API

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.

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

Ketika suatu route kita deklarasikan menggunakan method POST maka kita tidak akan bisa mengaksesnya menggunakan method lain. Contoh:

```
Route::post('umur', function () {
    return 17;
});
```

Ketika kita akses menggunakan browser yang notabene menggunakan GET maka akan menampilkan error.

The screenshot shows a browser window with the following details:

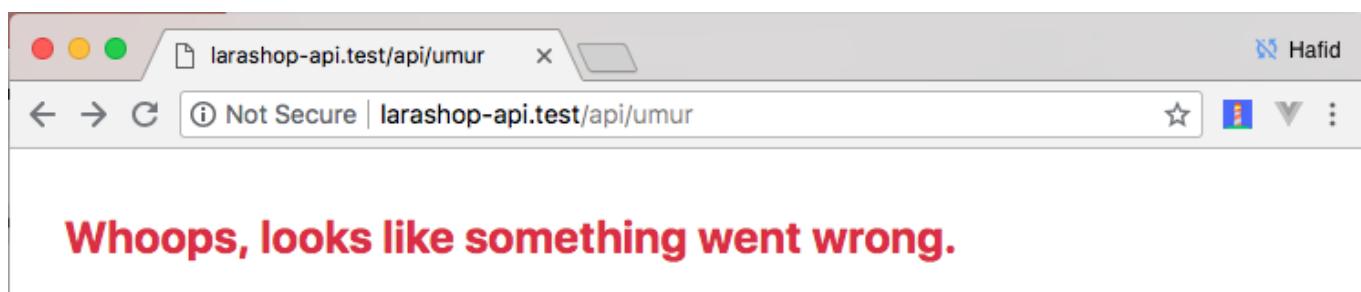
- Header:** Whoops! There was an error.
- URL:** Not Secure | larashop-api.test/api/umur
- Error Message:** Symfony \ Component \ HttpKernel \ Exception \ MethodNotAllowedHttpException
- No message**
- Buttons:** G, ⌂, ⌂
- Stack Trace:** Application frames (1) → All frames (27) → Symfony\Component\HttpKernel\Exception\MethodNotAllowedHttpException at .../vendor/laravel/framework/src/Illuminate/Routing/RouteCollection.php:255
- Code Snippet (from RouteCollection.php:255):**

```
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 $others)
254.     {
255.         throw new MethodNotAllowedHttpException($others);
256.     }
257.     /**
258.      * Get routes from the collection by method.
259.      *
260.      * @param string|null $method
261.      * @return array
262.      */
263. 
```

Tampilan error **menyeramkan** ini muncul karena setting APP_DEBUG pada file `.env` kita set bernilai true, namun jika nilainya false.

```
APP_DEBUG=false
```

Maka akan muncul tampilan error yang lebih ramah sebagai berikut.



Catatan: saat pengembangan aplikasi, sebaiknya kita menggunakan pengaturan `APP_DEBUG=true` untuk memudahkan kita mengetahui detail kesalahan yang terjadi.

Untuk mengakses routing `umur` di atas kita bisa menggunakan Postman, dengan method POST.

POST http://larashop-api.test/api/umur

Params

Pretty Raw Preview

17

Adakalanya kita ingin membuat routing yang bisa diakses oleh beberapa method sekaligus, maka kita bisa gunakan fungsi `match`.

```
Route::match(['get', 'post'], 'test', function () {  
    //  
});
```

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`.

```
Route::any('foo', function () {  
    //  
});
```

Routing Parameter

Kita juga dapat menyisipkan parameter pada routing, sehingga bisa membuat respon yang dinamis sesuai dengan parameter yang dikirimkan oleh user. Contoh:

```
Route::get('category/{id}', function ($id) {
    $categories = [
        1 => 'Programming',
        2 => 'Desain Grafis',
        3 => 'Jaringan Komputer',
    ];
    $id = (int) $id;
    if($id==0) return 'Silakan pilih kategori';
    else return 'Anda memilih kategori <b>' . $categories[$id] . '</b>';
});
```

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.

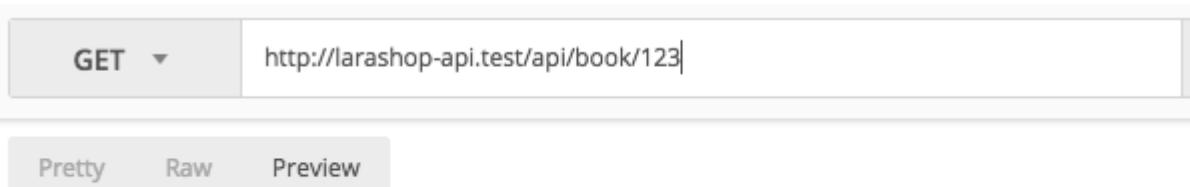
Solusinya, kita bisa membuat parameter id pada routing tersebut optional. Caranya, tambahkan tanda tanya ? pada parameter dan berikan nilai default pada fungsi.

```
Route::get('category/{id?}', function ($id=null) {
    // ...
});
```

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*.

```
Route::get('book/{id}', function () {
    return 'buku angka';
})->where('id', '[0-9]+');
```

Pada contoh di atas, routing akan match jika parameter id bernilai numerik atau angka.



buku angka

```
Route::get('book/{title}', function ($title) {
    return 'buku abjad';
})->where('title', '[A-Za-z]+');
```

Pada contoh di atas, routing akan match jika parameter id bernilai abjad.

The screenshot shows a browser's developer tools Network tab. A GET request is listed with the URL `http://larashop-api.test/api/book/abjad`. The response body is displayed as `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.

```
Route::get('book/{id}', function ($id) {
    //
})->name('book');
```

Sehingga untuk menggunakannya kita cukup memanggil namanya saja, misal

```
// Generating URLs...
$url = route('book', ['id' => 1]);

// Generating Redirects...
return redirect()->route('book', [
    'id' => 1
]);
```

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.

```
Route::prefix('v1')->group(function () {
    Route::get('books', function () {
        // Match dengan "/v1/books"
    });

    Route::get('categories', function () {
        // Match dengan "/v1/categories"
    });
});
```

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()`

```
protected function mapApiRoutes()
{
    Route::prefix('api')
        ->middleware('api')
        ->namespace($this->namespace)
        ->group(base_path('routes/api.php'));
}
```

Jika method `prefix('api')` di atas kita hapus, menjadi sebagai berikut.

```
protected function mapApiRoutes()
{
    Route::middleware('api')
        ->namespace($this->namespace)
        ->group(base_path('routes/api.php'));
}
```

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.

```
Route::domain('{category}.larashop.id')->group(function () {
    Route::get('book/{id}', function ($category, $id) {
        //
    });
});
```

Setelah kita banyak melakukan konfigurasi routing, kita bisa jalankan perintah berikut untuk melihat routing yang tersedia pada aplikasi kita.

```
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\(\)](#)

```
public function map()
{
    $this->mapApiRoutes();
    // $this->mapWebRoutes(); // <= ini
}
```

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.

```
php artisan make:controller BookController
```

```
root@28de49479092:/var/www/larashop-api# php artisan make:controller BookController
r
Controller created successfully.
```

Maka akan digenerate file BookController.php pada direktori [app/Http/Controllers](#)

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class BookController extends Controller
{
```

```
//  
}
```

Lalu pada class BookController tersebut kita tambahkan fungsi

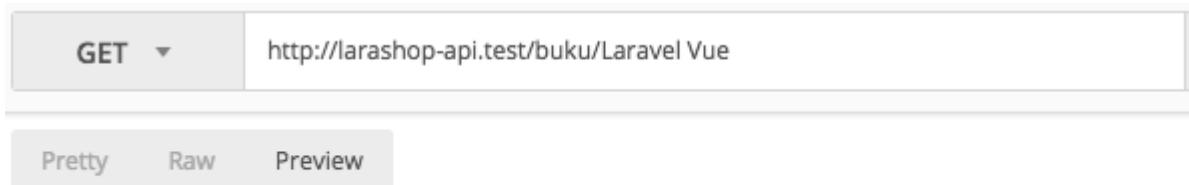
```
public function cetak($judul)  
{  
    return $judul;  
}
```

Catatan: merupakan *best practice* memberikan nama class controller dengan akhiran Controller.
Apakah harus demikian? jawabannya tidak harus.

Kemudian pada konfigurasi routing `api.php`, kita tambahkan.

```
Route::get('buku/{judul}', 'BookController@cetak');
```

Mari kita coba melalui postman dengan mengakses URL `http://larashop-api.test/buku/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

Licensed to M Najamudin Ridha - admin@komputerkampus.com - 087814493571 at 01/12/2018 19:54:30
dalam setiap menitnya. Sebagai contoh routing buku maksimal hanya boleh diakses sebanyak 10 kali dalam setiap menitnya:

```
Route::middleware('throttle:10,1')->group(function () {  
    Route::get('buku/{judul}', 'BookController@cetak');  
});
```

Ketika diakses lebih dari itu maka akan muncul error Too many request.

The screenshot shows a Laravel application's error page for a rate limit exceeded. At the top, there is a navigation bar with 'GET' selected, a URL 'http://larashop-api.test/buku/Laravel Vue', and a 'Params' button. Below the navigation are three tabs: 'Pretty' (selected), 'Raw', and 'Preview'. The main content features a large '429' error code. Below it, a horizontal line separates the error from the explanatory text: 'Sorry, you are making too many requests to our servers.' At the bottom, there is a button labeled 'GO HOME'.

Secara default, throttle telah didefinisikan pada App\Http\Kernel.php

```
protected $middlewareGroups = [  
    'web' => [  
        ...  
    ],  
    'api' => [  
        'throttle:60,1',  
        'bindings',  
    ],  
];
```

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 menyenggung 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.

```
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# cd larashop-api/
[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.

```
<?php
namespace App\Http\Middleware;
use Closure;
class Cors
{
    public function handle($request, Closure $next)
    {
        return $next($request)
            ->header('Access-Control-Allow-Origin', '*')
            ->header('Access-Control-Allow-Credentials', true)
            ->header('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE,
OPTIONS')
            ->header('Access-Control-Allow-Headers', 'X-Requested-With,
Content-Type, X-Token-Auth, Authorization')
            ;
    }
}
```

Setelah itu kita bisa daftarkan pada file [/app/Http/Kernel.php](#) tepatnya di properti [routeMiddleware](#)

```
protected $routeMiddleware = [
    ...
    'cors' => \App\Http\Middleware\Cors::class
];
```

Selanjutnya kita bisa gunakan pada routing.

```
Route::middleware(['cors'])->group(function () {
    Route::get('buku/{judul}', 'BookController@cetak');
});
```

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.

```
<script>
export default {
    created () {
        window.axios.get('http://larashop-api.test/buku/PHP MySQL')
            .then((response) => {
                console.log(response)
            })
            .catch((error) => {
                console.log(error)
            })
    }
}
</script>
```

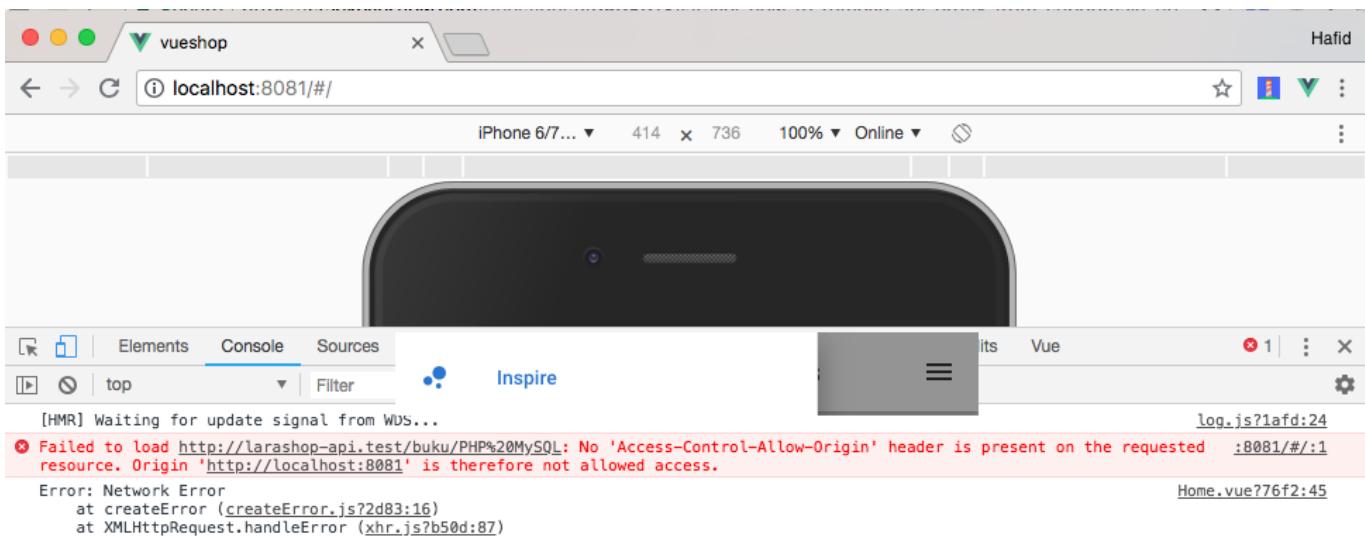
Kode di atas akan menampilkan hasil request pada console log.

Lalu pada terminal, jalankan perintah **npm run serve**

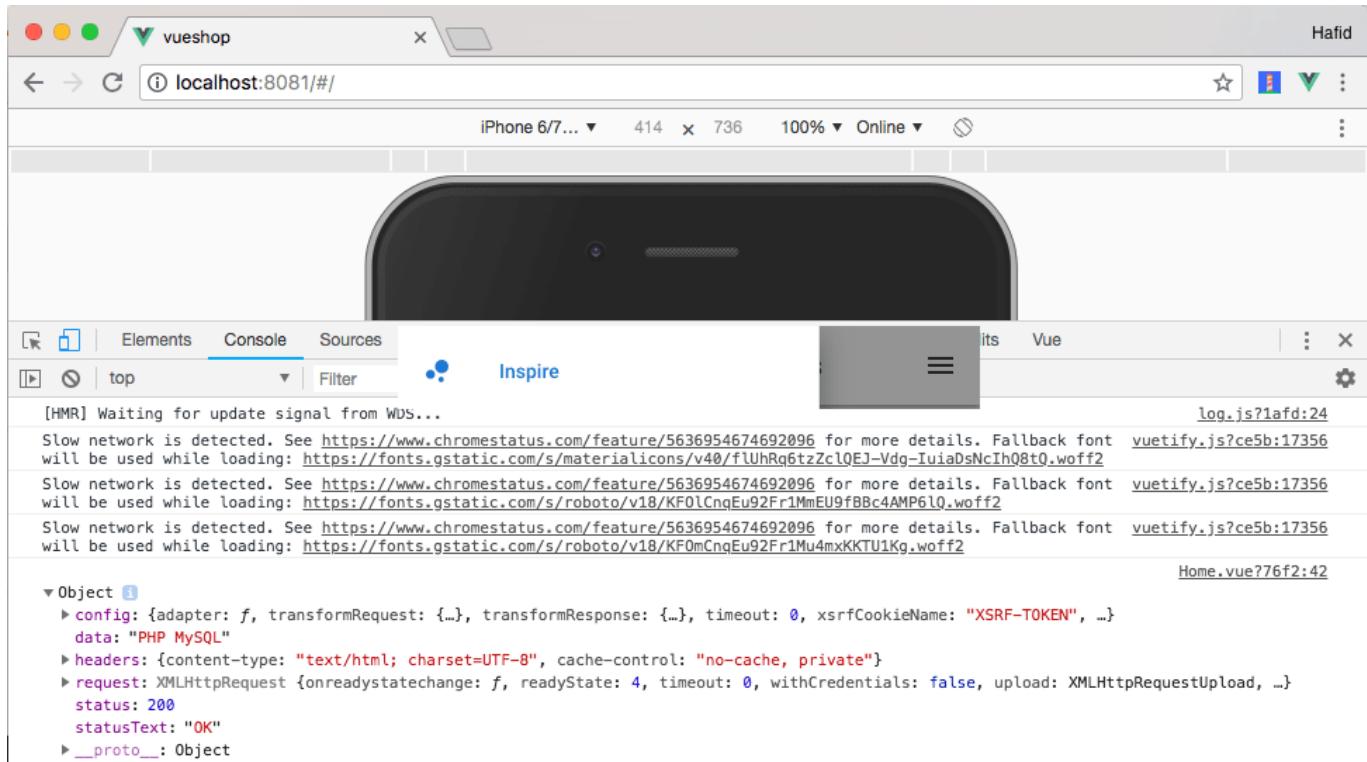
```
vueshop — root@8a48a85542c9: /var/www/larashop-api — node < npm TERM_PROGRAM=A  
[Hafids-MacBook-Pro:Dev hafidmukhlasin$ cd vue-projects/  
[Hafids-MacBook-Pro:vue-projects hafidmukhlasin$ cd vueshop/  
[Hafids-MacBook-Pro:vueshop hafidmukhlasin$ npm run serve  
  
> vueshop@0.1.0 serve /Users/hafidmukhlasin/Dev/vue-projects/vueshop  
> 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) (<https://github.com/barryvdh/laravel-cors>).

```
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

```
protected $routeMiddleware = [
    // ... middleware lain
    'cors' => \Barryvdh\Cors\HandleCors::class,
];
```

Supaya otomatis tanpa mendeklarasikan secara manual di routing maka cors bisa kita tambahkan pada Kernel yaitu di bagian middlewareGroups.

```
protected $middlewareGroups = [
    'api' => [
        'throttle:60,1',
        'bindings',
        'cors', // <= ini
    ],
];
```

Multiple Middleware

Kita dapat mengimplementasikan multiple middleware sekaligus dengan menggunakan array. Contohnya sebagai berikut.

```
Route::middleware(['throttle:10,1', 'cors'])->group(function () {
    Route::get('buku/{judul}', 'BookController@cetak');
});
```

atau bisa juga nested.

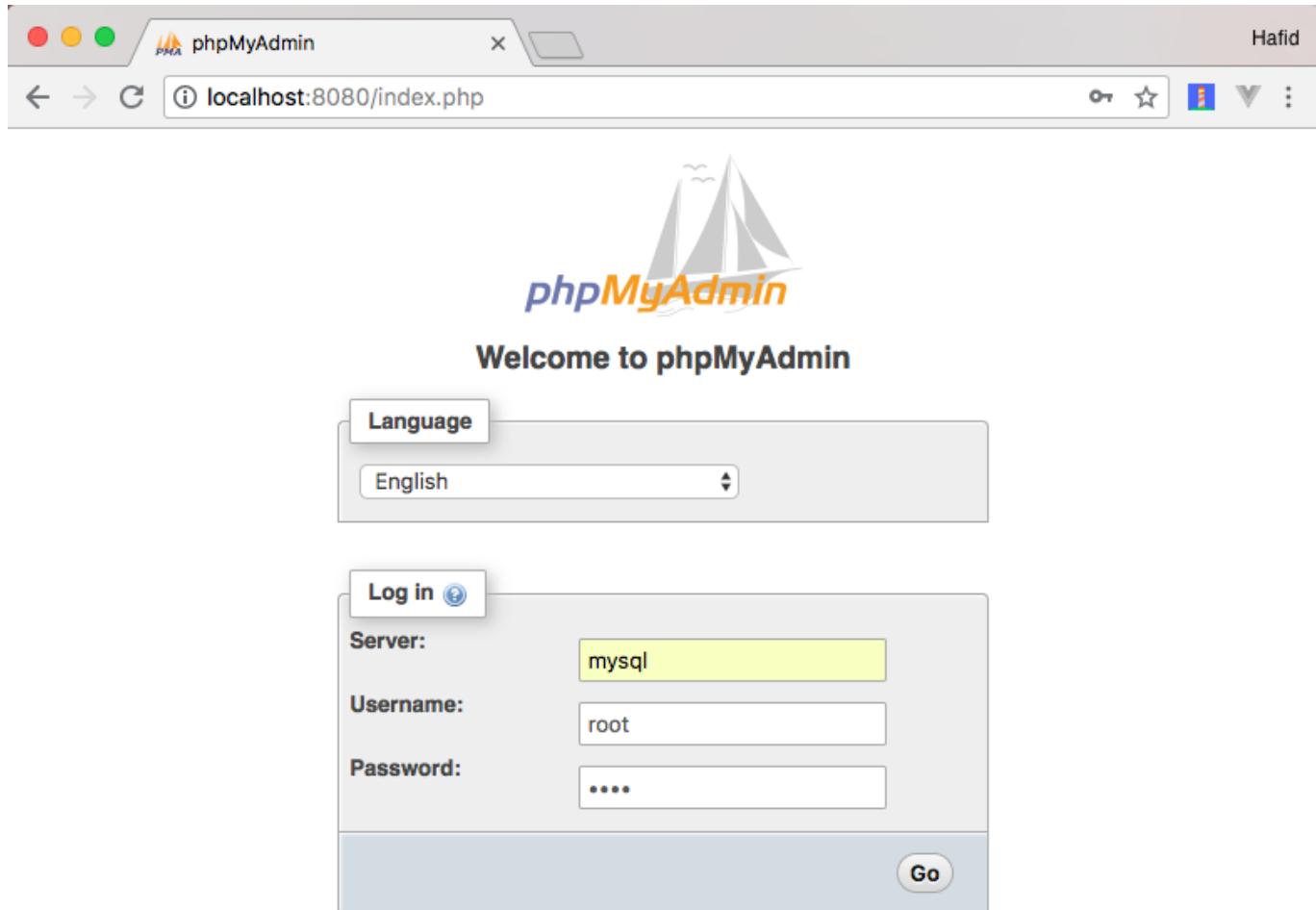
```
Route::middleware(['throttle:10,1'])->group(function () {
    Route::middleware(['cors'])->group(function () {
        Route::get('buku/{judul}', 'BookController@cetak');
    });
});
```

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 phpMyAdmin. On the left, a sidebar lists databases: New, default, information_schema, mysql, performance_schema, and sys. In the main area, a 'Create database' button is visible with the name 'larashop' entered into its input field. A dropdown menu next to it shows 'utf8_general_ci' selected. A 'Create' button is located to the right of the input field.

Konfigurasi

Sesuaikan konfigurasi database pada file .env laravel

```
DB_CONNECTION=mysql  
DB_HOST=mysql  
DB_PORT=3306  
DB_DATABASE=larashop
```

```
DB_USERNAME=root  
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.

```
php artisan make:migration create_books_table
```

```
[● ● ● laradock — root@8a48a85542c9: /var/www/larashop-api — docker < docker-...  
[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`

```
<?php  
use Illuminate\Support\Facades\Schema;  
use Illuminate\Database\Schema\Blueprint;  
use Illuminate\Database\Migrations\Migration;  
  
class CreateBooksTable extends Migration  
{  
    public function up()  
    {  
        Schema::create('books', function (Blueprint $table) {
```

```

        $table->increments('id');
        $table->timestamps();
    });
}

public function down() {
    Schema::dropIfExists('books');
}
}

```

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.

```

public function up()
{
    Schema::create('books', function (Blueprint $table) {
        $table->increments('id');
        $table->string('title');
        $table->string('slug')->unique();
        $table->text('description')->nullable();
        $table->string('author')->nullable();
        $table->string('publisher')->nullable();
        $table->string('cover')->nullable();
        $table->float('price')->unsigned()->default(0);
        $table->float('weight')->unsigned()->default(0);
        $table->integer('views')->unsigned()->default(0);
        $table->integer('stock')->unsigned()->default(0);
        $table->enum('status', ['PUBLISH', 'DRAFT'])->default('PUBLISH');
        $table->timestamps(); // created_at, updated_at
        $table->softDeletes(); // deleted_at
        $table->integer('created_by')->nullable();
        $table->integer('updated_by')->nullable();
        $table->integer('deleted_by')->nullable();
    });
}

```

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.

```
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

```
<?php
use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class AlterUsersTable extends Migration
{
    public function up()
    {
        Schema::table('users', function (Blueprint $table) {
            $table->text('roles')->nullable();
            $table->text('address')->nullable();
            $table->integer('city_id')->nullable();
            $table->integer('province_id')->nullable();
            $table->string('phone')->nullable();
            $table->string('avatar')->nullable();
            $table->enum('status', ['ACTIVE', 'INACTIVE'])->default('ACTIVE');
        });
    }

    public function down()
    {
```

```

Schema::table('users', function (Blueprint $table) {
    $table->dropColumn("roles");
    $table->dropColumn("address");
    $table->dropColumn("city_id");
    $table->dropColumn("province_id");
    $table->dropColumn("phone");
    $table->dropColumn("avatar");
    $table->dropColumn("status");
})];
}
}

```

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

```
php artisan make:migration create_categories_table
```

Maka kode migration fungsi up-nya akan menjadi sebagai berikut.

```

public function up()
{
    Schema::create('categories', function (Blueprint $table) {
        $table->increments('id');
        $table->string('name');
        $table->string('slug')->unique();
        $table->string('image')->nullable();
        $table->enum('status', ['PUBLISH', 'DRAFT'])->default('PUBLISH');
    });
}

```

```
$table->timestamps(); // created_at, updated_at
$table->softDeletes(); // deleted_at
$table->integer('created_by')->nullable();
$table->integer('updated_by')->nullable();
$table->integer('deleted_by')->nullable();
});
}
```

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

```
php artisan make:migration create_book_category_table
```

Maka kode migration fungsi up-nya akan menjadi sebagai berikut.

```
public function up()
{
    Schema::create('book_category', function (Blueprint $table) {
        $table->increments('id');
        $table->integer('book_id');
        $table->integer('category_id');
        $table->timestamps();
    });
}
```

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

```
php artisan make:migration create_orders_table
```

Maka kode migration fungsi up-nya akan menjadi sebagai berikut.

```
public function up()
{
    Schema::create('orders', function (Blueprint $table) {
        $table->increments('id');
        $table->integer('user_id');
        $table->float('total_price');
        $table->string('invoice_number');
        $table->string('courier_service')->nullable();
        $table->enum('status', ['SUBMIT', 'PROCESS', 'FINISH', 'CANCEL'])-
>default('SUBMIT');
        $table->timestamps();
    });
}
```

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

```
php artisan make:migration create_book_order_table
```

Maka kode migration fungsi up-nya akan menjadi sebagai berikut.

```
public function up()
{
    Schema::create('book_order', function (Blueprint $table) {
        $table->increments('id');
        $table->integer('book_id');
        $table->integer('order_id');
        $table->integer('quantity');
        $table->timestamps();
    });
}
```

Tabel Provinces

Tabel ini untuk menyimpan data provinces yang didapat dari API RajaOngkir.com, berikut ini skemanya.

provinces
+ id: Integer{PK}
+ province: String

```
php artisan make:migration create_provinces_table
```

Maka kode migration fungsi up-nya akan menjadi sebagai berikut.

```
public function up()
{
    Schema::create('provinces', function (Blueprint $table) {
        $table->increments('id');
        $table->string('province');
    });
}
```

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

```
php artisan make:migration create_cities_table
```

Maka kode migration fungsi up-nya akan menjadi sebagai berikut.

```
public function up()
{
    Schema::create('cities', function (Blueprint $table) {
        $table->increments('id');
        $table->integer('province_id');
        $table->string('city');
        $table->string('type');
        $table->string('postal_code');
    });
}
```

Execute Migration

Untuk menjalankan migration pada terminal jalankan perintah berikut.

```
php artisan migrate
```

```
laradock — root@28de49479092: /var/www/larashop-api — docker < docker-co...
[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.

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** mysql » **Database:** larashop
- Structure:** The main tab selected.
- SQL:** SQL tab is present but not selected.
- Search:** Search tab is present but not selected.
- Query:** Query tab is present but not selected.
- Export:** Export tab is present but not selected.
- More:** More tab is present but not selected.
- Filters:** A search bar labeled "Containing the word:" is present.
- Table:** A table listing 10 tables in the larashop database:

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
- Console:** Console tab is present at the bottom.

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](#)

[Table structure](#) [Relation view](#)

#	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.

```
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.

```
php artisan migrate:refresh
```

Atau gunakan perintah

```
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

Misalnya kita akan membuat seeder untuk tabel `users` maka gunakan perintah berikut.

```
php artisan make:seeder UsersTableSeeder
```

```
laradock — root@8a48a85542c9: /var/www/larashop-api — docker + docker-...
root@8a48a85542c9:/var/www/larashop-api# php artisan make:seeder UsersTableSeede
r
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.

```
<?php
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;
```

```

class UsersTableSeeder extends Seeder
{
    public function run() {
        DB::table('users')->insert([
            'name' => 'Anwar',
            'email' => 'anwar@gmail.com',
            'password' => bcrypt('123456'),
            'roles' => json_encode(['CUSTOMER']),
            'status' => 'ACTIVE',
        ],
        [
            'name' => 'Budi',
            'email' => 'budi@gmail.com',
            'password' => bcrypt('123456'),
            'roles' => json_encode(['CUSTOMER']),
            'status' => 'ACTIVE',
        ]);
        // dst
    }
}

```

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.

```

public function run()
{
    $users = [];
    $faker = Faker\Factory::create();
    for($i=0;$i<5;$i++){
        $avatar_path = '/var/www/larashop-api/public/images/users';
        $avatar_fullpath = $faker->image( $avatar_path, 200, 250, 'people',
true, true, 'people');
        $avatar = str_replace($avatar_path . '/' , '', $avatar_fullpath);
        $users[$i] = [
            'name'      => $faker->name,
            'email'     => $faker->unique()->safeEmail,
            'password'  => bcrypt('123456'),
            'roles'     => json_encode(['CUSTOMER']),
            'avatar'    => $avatar,
            'status'    => 'ACTIVE',
            'created_at'=> Carbon\Carbon::now(),
        ];
    }
    DB::table('users')->insert($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`

```
mkdir public/images
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.

```
php artisan make:seeder BooksTableSeeder
```

Lalu tambahkan data dummy pada seeder tabel Books atau `BooksTableSeeder.php`

```
<?php

use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;

class BooksTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        DB::table('books')->insert([
            'title' => 'C++ High Performance',
            'slug' => 'c++-high-performance',
            'description' => 'Write code that scales across CPU registers,
multi-core, and machine clusters',
        ]);
    }
}
```

```

        'author' => 'Viktor Sehr, Björn Andrist',
        'publisher' => 'Packtpub',
        'cover' => 'c++-high-performance.png',
        'price' => 100000,
        'weight' => 0.5,
        'status' => 'PUBLISH',
    ], [
        'title' => 'Mastering Linux Security and Hardening',
        'slug' => 'mastering-linux-security-and-hardening',
        'description' => 'A comprehensive guide to mastering the art of preventing your Linux system from getting compromised',
        'author' => 'Donald A. Tevault',
        'publisher' => 'Packtpub',
        'cover' => 'mastering-linux-security-and-hardening.png',
        'price' => 125000,
        'weight' => 0.5,
        'status' => 'PUBLISH',
    ], [
        'title' => 'Mastering PostgreSQL 10',
        'slug' => 'mastering-postgresql-10',
        'description' => 'Master the capabilities of PostgreSQL 10 to efficiently manage and maintain your database',
        'author' => 'Hans-Jürgen Schönig',
        'publisher' => 'Packtpub',
        'cover' => 'mastering-postgresql-10.png',
        'price' => 90000,
        'weight' => 0.5,
        'status' => 'PUBLISH',
    ], [
        'title' => 'Python Programming Blueprints',
        'slug' => 'c++-high-performance',
        'description' => 'How to build useful, real-world applications in the Python programming language',
        'author' => 'Daniel Furtado, Marcus Pennington',
        'publisher' => 'Packtpub',
        'cover' => 'python-programming-blueprints.png',
        'price' => 75000,
        'weight' => 0.5,
        'status' => 'PUBLISH',
    ]);
}
}

```

Tentu, kita juga bisa menggunakan faker.

```

public function run()
{
    $books = [];
    $faker = Faker\Factory::create();
    $image_categories = ['abstract', 'animals', 'business', 'cats',
    'city', 'food',
    'nature', 'technics', 'transport'];

```

```
for($i=0;$i<25;$i++){
    $title = $faker->sentence(mt_rand(3, 6));
    $title = str_replace('.', '-', $title);
    $slug = str_replace(' ', '-', strtolower($title));
    $category = $image_categories[mt_rand(0, 8)];
    $cover_path = '/var/www/larashop-api/public/images/books';
    $cover_fullpath = $faker->image( $cover_path, 300, 500,
$category, true, true, $category);
    $cover = str_replace($cover_path . '/' , '', $cover_fullpath);
    $books[$i] = [
        'title' => $title,
        'slug' => $slug,
        'description' => $faker->text(255),
        'author' => $faker->name,
        'publisher' => $faker->company,
        'cover' => $cover,
        'price' => mt_rand(1, 10) * 50000,
        'weight' => 0.5,
        'status' => 'PUBLISH',
        'created_at' => Carbon\Carbon::now(),
    ];
}
DB::table('books')->insert($books);
}
```

Jangan lupa buat dulu folder `books` pada direktori `public/image`

```
mkdir public/images/books
```

Catatan: `str_slug` adalah helpers Laravel untuk mengkonversi string menjadi format slug, selengkapnya: <https://laravel.com/docs/5.7/helpers>

Tabel Categories

Buat seeder untuk tabel Categories.

```
php artisan make:seeder CategoriesTableSeeder
```

Lalu tambahkan data dummy pada seeder tabel Categories atau `CategoriesTableSeeder.php` di fungsi `run()` dengan menggunakan faker.

```
public function run()
{
    $categories = [];
    $faker = Faker\Factory::create();
    $image_categories = ['abstract', 'animals', 'business', 'cats', 'city',
```

```

'food',
'nature', 'technics', 'transport'];
for($i=0;$i<8;$i++){
    $name = $faker->unique()->word();
    $name = str_replace('.', ' ', $name);
    $slug = str_replace(' ', '-', strtolower($name));
    $category = $image_categories[mt_rand(0, 8)];
    $image_path = '/var/www/larashop-api/public/images/categories';
    $image_fullpath = $faker->image( $image_path, 500, 300, $category,
true, true, $category);
    $image = str_replace($image_path . '/' , '', $image_fullpath);
    $categories[$i] = [
        'name' => $name,
        'slug' => $slug,
        'image' => $image,
        'status' => 'PUBLISH',
        'created_at' => Carbon\Carbon::now(),
    ];
}
DB::table('categories')->insert($categories);
}

```

Jangan lupa buat folder **categories** pada direktori **public/image**

```
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.

RajaOngkir

Rajanya Ongkos Kirim Terpadu

Beranda Integrasi Download Bantuan

Hafid

API Key

Gunakan API Key ini untuk menggunakan API RajaOngkir. Untuk informasi lebih lanjut mengenai cara menggunakan API RajaOngkir, silakan baca [dokumentasi](#).

a23e...7c80a...55405c

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

```

GET https://api.rajaongkir.com/starter/province?key=a23e...51...
Params Send Save
Body Cookies Headers (7) Test Results Status: 200 OK Time: 284 ms Size: 1.93 KB
Pretty Raw Preview JSON
1 [
2   "rajaongkir": {
3     "query": {
4       "key": "a23e...55405c"
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  ]

```

Buat seeder untuk tabel provinces.

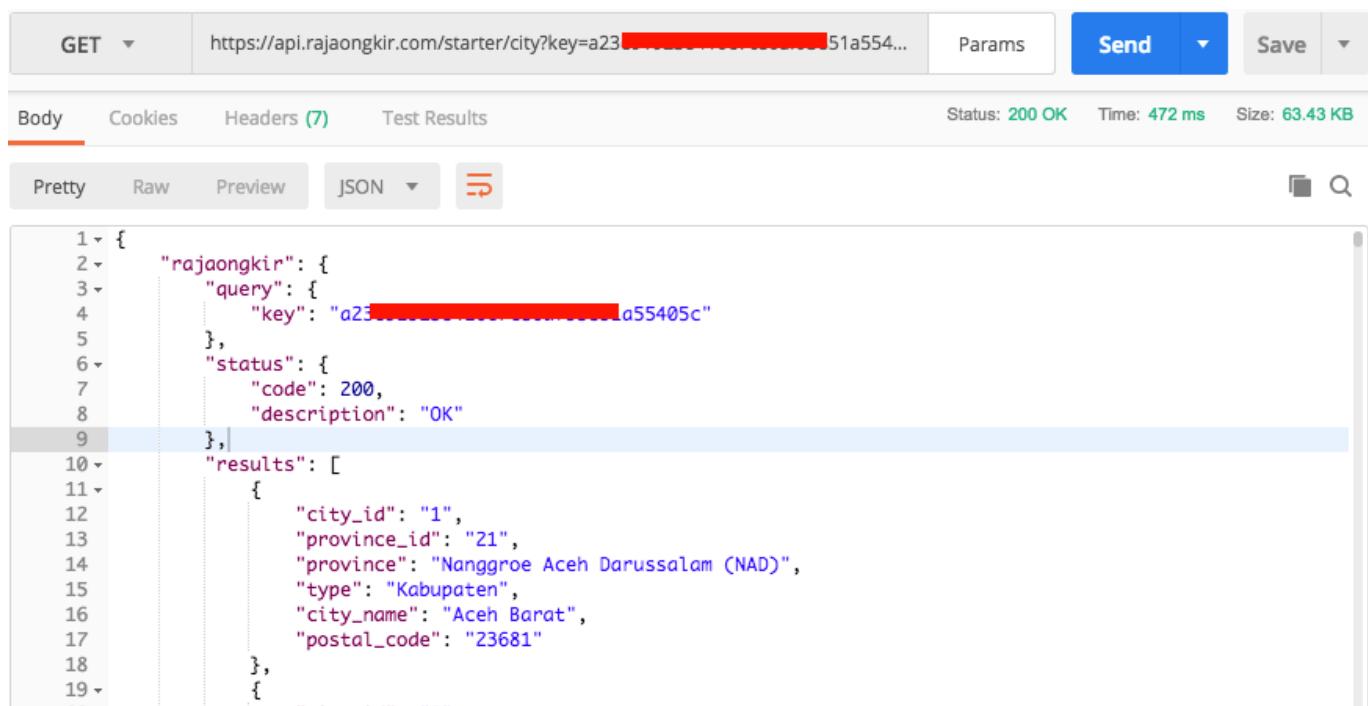
```
php artisan make:seeder ProvincesTableSeeder
```

Lalu tambahkan data province dari API RajaOngkir pada seeder tabel Provinces atau file ProvincesTableSeeder.php di fungsi run().

```
public function run()
{
    $url_province = "https://api.rajaongkir.com/starter/province?
key=API_KEY";
    $json_str = file_get_contents($url_province);
    $json_obj = json_decode($json_str);
    $provinces = [];
    foreach($json_obj->rajaongkir->results as $province){
        $provinces[] = [
            'id' => $province->province_id,
            'province' => $province->province
        ];
    }
    DB::table('provinces')->insert($provinces);
}
```

Untuk mendapatkan data kota kita bisa gunakan URL ini

https://api.rajaongkir.com/starter/city?key=API_KEY



Body Cookies Headers (7) Test Results Status: 200 OK Time: 472 ms Size: 63.43 KB

Pretty Raw Preview JSON ↗

```
1 {
2   "rajaongkir": {
3     "query": {
4       "key": "a23...51a55405c"
5     },
6     "status": {
7       "code": 200,
8       "description": "OK"
9     },
10    "results": [
11      {
12        "city_id": "1",
13        "province_id": "21",
14        "province": "Nanggroe Aceh Darussalam (NAD)",
15        "type": "Kabupaten",
16        "city_name": "Aceh Barat",
17        "postal_code": "23681"
18      },
19      {
20        "city_id": "2",
21        "province_id": "21",
22        "province": "Nanggroe Aceh Darussalam (NAD)",
23        "type": "Kota",
24        "city_name": "Banda Aceh",
25        "postal_code": "23111"
26      }
27    ]
28  }
29 }
```

Buat seeder untuk tabel cities.

```
php artisan make:seeder CitiesTableSeeder
```

Lalu tambahkan data kota dari API RajaOngkir pada seeder tabel cities atau file CitiesTableSeeder.php di fungsi run().

```
public function run()
{
    $url_city = "https://api.rajaongkir.com/starter/city?key=API_KEY";
    $json_str = file_get_contents($url_city);
    $json_obj = json_decode($json_str);
    $cities = [];
    foreach($json_obj->rajaongkir->results as $city){
        $cities[] = [
            'id'          => $city->city_id,
            'province_id' => $city->province_id,
            'city'         => $city->city_name,
            'type'         => $city->type,
            'postal_code'  => $city->postal_code,
        ];
    }
    DB::table('cities')->insert($cities);
}
```

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.

```
<?php
use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    public function run()
    {
        $this->call(UsersTableSeeder::class);
        $this->call(BooksTableSeeder::class);
        $this->call(CategoriesTableSeeder::class);
        $this->call(ProvincesTableSeeder::class);
        $this->call(CitiesTableSeeder::class);
    }
}
```

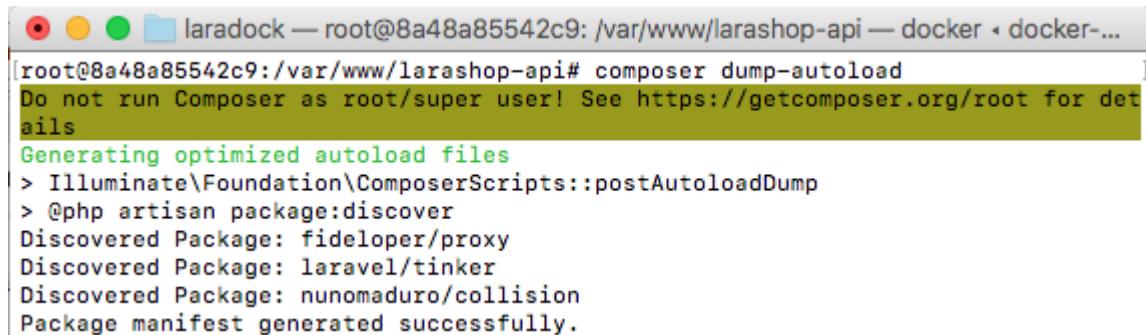
Execute Seeder

Setiap kali kita selesai menulis seeder maka kita perlu meng-generate kembali autoloader dari Composer menggunakan perintah `dump-autoload`.

```
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`

```
php artisan db:seed  
php artisan db:seed --class=UsersTableSeeder
```



The screenshot shows a terminal window titled "laradock — root@8a48a85542c9: /var/www/larashop-api — docker < docker-...". The command run is "composer dump-autoload". The output shows the process of generating optimized autoload files, discovering packages, and generating a package manifest successfully.

```
[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.

```
php artisan migrate:refresh --seed
```

Perintah di atas akan melakukan rollback migration, kemudian mengeksekusi migration dan yang terakhir melakukan seeding data.

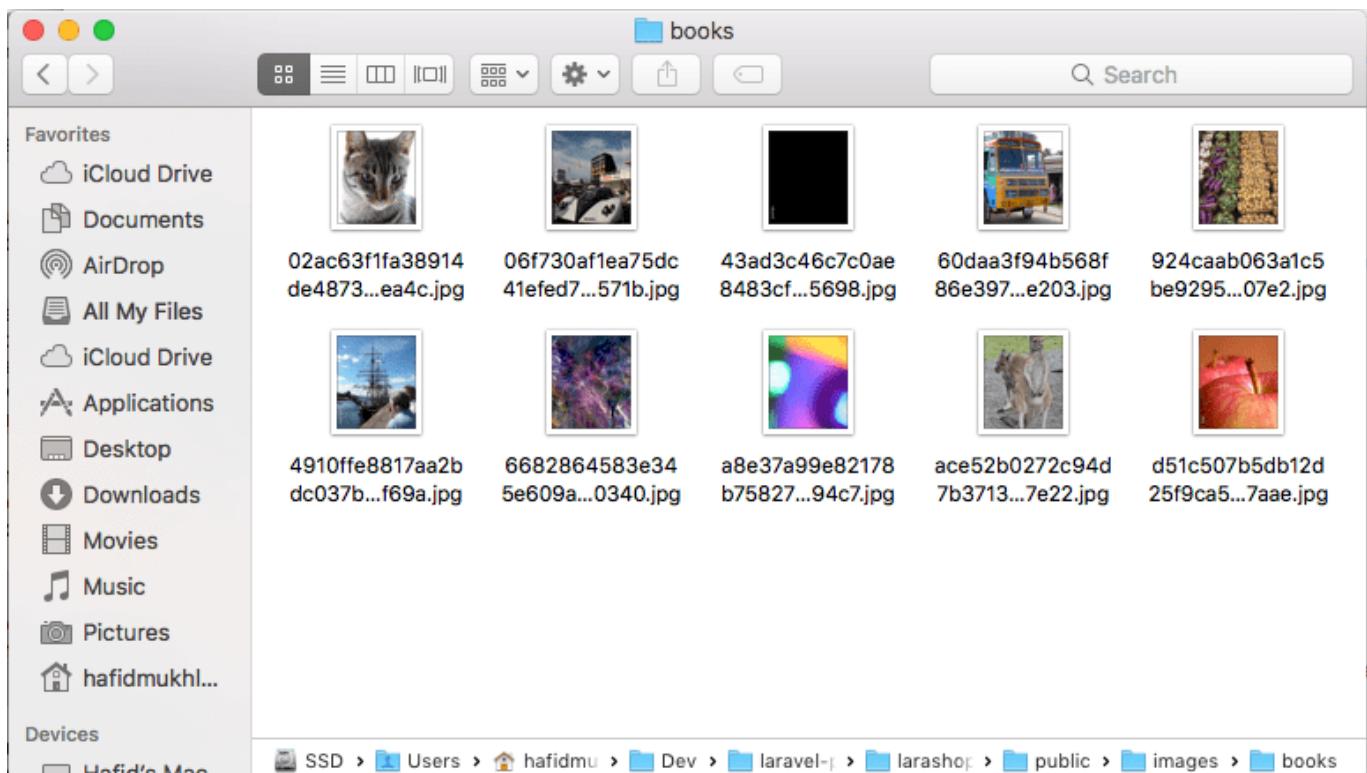
```
[root@8a48a85542c9:/var/www/larashop-api# php artisan migrate:refresh --seed
Rolling back: 2018_08_06_213147_create_cities_table
Rolled back: 2018_08_06_213147_create_cities_table
Rolling back: 2018_08_06_213134_create_provinces_table
Rolled back: 2018_08_06_213134_create_provinces_table
Rolling 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.

Screenshot of a MySQL database table named 'books' in a browser window. The table has columns: id, title, slug, description, author, publisher, cover, price, weight, views, stock, and status.

	Browse	Structure	SQL	Search	Insert	Export	Import	Privileges	Operations	Triggers	
id	title	slug	description	author	publisher	cover	price	weight	views	stock	status
1	Ut tempore doloremque	ut-tempore-doloremque	Sint et ut unde suscipit. Sed officia delectus nes...	Forest Grimes	Funk, Aufderhar and Kuvalis	ace52b0272c94d7b371355f743087e22.jpg	500000.00	0.50	0	1	P
2	Eligendi praesentium aut voluptatem molestias ut	eligendi-praesentium-aut-voluptatem-molestias-ut	Consequuntur velit iure est omnis dolores quo... Do...	Seth Tromp	Yost PLC	d51c507b5db12d25f9ca590e19c67aae.jpg	200000.00	0.50	0	1	P
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	P
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	P
5	Fugit voluptatibus delectus in	fugit-voluptatibus-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	P

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().

```
public function index()
{
    $books = DB::select('select * from books');
    return $books;
}
```

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.

```
Route::prefix('v1')->group(function () {
    Route::get('books', 'BookController@index');
});
```

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, showing a single book entry with ID 28 and various fields like title, slug, description, etc.

```

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().

```
public function view($id)
{
    $book = DB::select('select * from books where id = ?', [ $id ]);
    return $book;
}
```

Di samping menggunakan tanda ?, kita juga bisa membinding nama variabel.

```
$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.

```
Route::prefix('v1')->group(function () {
    Route::get('books', 'BookController@index');
    Route::get('book/{id}', 'BookController@view')->where('id', '[0-9]+');
    // <== tambahkan ini
});
```

Akses menggunakan URL v1/book/2, maka hasilnya:

```

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": 500000,
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  } 
22 ]

```

Untuk fungsi query yang lain seperti insert, update, delete, dan statement bisa dilihat pada contoh berikut.

```

DB::insert('insert into books (title, slug) values (?, ?)', ['Learn VueJS', 'learn-vuejs']);

$affected = DB::update('update books set price = ? where id = ?', [125000, 3]);

$deleted = DB::delete('delete from books where id=?', [5]);

DB::statement('drop table books');

```

Database transaction juga didukung dengan menggunakan kode berikut.

```

DB::transaction(function () {
    DB::insert('insert into books (title) values (?)', ['Learn VueJS']);

    $affected = DB::update('update books set price = ? where id = ?', [125000, 3]);

    $deleted = DB::delete('delete from books where id=?', [5]);
});

```

Atau bisa juga menggunakan cara manual `DB::beginTransaction()`; untuk rollback transaction `DB::rollBack()`; dan untuk commit transaction menggunakan `DB::commit()`;

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)

```
// select * from books
$books = DB::table('books')->get();

// select * from books where id = 3
$books = DB::table('books')->where('id', 3)->first();

// select title from books where id = 3
$title = DB::table('books')->where('id', 3)->value('title');

// mereturn collection data
// select id, title from books
$books = DB::table('books')->pluck('id', 'title');
foreach ($books as $id => $title) {
    echo $title;
}

// select count(*) from books
$count_books = DB::table('books')->count();

// select max(price) from books
$max_book_price = DB::table('books')->max('price');

// select average(price) from books
$avg_book_price = DB::table('books')->avg('price');

// insert into books (title) values ('Learn VueJS')
DB::table('books')->insert(
    ['title' => 'Learn VueJS', 'slug' => 'learn-vuejs']
);

// update books set price = 125000 where id =3
DB::table('books')
    ->where('id', 3)
    ->update(['price' => 125000]);

// delete from books where id=5
DB::table('books')->where('id', '=', 5)->delete();
```

Selengkapnya mengenai query builder dapat kamu baca melalui tautan ini
<https://laravel.com/docs/5.7/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:

```
php artisan make:model Book
```

Catatan: tambahkan parameter `-m` atau `--migration` jika kita ingin sekaligus membuat kode untuk database migrationnya

```
php artisan make:model Book --migration  
php artisan make:model Book -m
```

Hanya saja pada bagian sebelumnya kan kita sudah membuat database migration.

```
[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`

```
<?php  
namespace App;  
use Illuminate\Database\Eloquent\Model;  
class Book extends Model  
{  
    //  
}
```

Buat juga model untuk tabel yang lain.

 laradock — root@8a48a85542c9: /var/www/larashop-api — docker • docker

```
[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'

```
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.

```
protected primaryKey = 'code'; // string
public $incrementing = false;
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.

```
const CREATED_AT = 'creation_date';
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.

```
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

```
class Book extends Model
{
    protected $fillable = [
        'title', 'slug', 'description', 'author', 'publisher',
        'cover', 'price', 'weight', 'stock', 'status'
    ];
}
```

- Model Category

```
class Category extends Model
{
    protected $fillable = [
        'name', 'slug', 'image', 'status'
    ];
}
```

- Model BookCategory

```
class BookCategory extends Model
{
    protected $table = 'book_category';
    protected $fillable = [
        'book_id', 'category_id', 'invoice_number', 'status'
    ];
}
```

- Model Order

```
class Order extends Model
{
    protected $fillable = [
        'user_id', 'total_price', 'invoice_number', 'status'
    ];
}
```

- Model BookOrder

```
class BookOrder extends Model
{
    protected $table = 'book_order';

    protected $fillable = [
        'book_id', 'order_id', 'quantity'
    ];
}
```

- Model User

Model ini masih sama dengan defaultnya, hanya saja kita tambahkan beberapa field pada properti \$fillable sesuai dengan desain database yang kita buat.

```
protected $fillable = [
    'name', 'email', 'password', 'username', 'roles',
    'address', 'city_id', 'province_id', 'phone', 'avatar', 'status'
];
```

- Model Province

```
class Province extends Model
{
    public $timestamps = false;
}
```

- Model City

```
class City extends Model
{
    public $timestamps = false;
}
```

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.

```
// select * from books
$books = \App\Book::all();
foreach ($books as $book) {
    echo $book->title;
}

// select * from books where status = 'PUBLISH' limit 10 order by title desc
$books = \App\Book::where('status', 'PUBLISH')
    ->orderBy('title', 'asc')
    ->limit(10)
    ->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.

```
// jangan ambil buku yang statusnya draft
$published_books = $books->reject(function ($book) {
    return $book->status=='DRAFT';
});

// ambil buku yang statusnya publish
$published_books = $books->filter(function ($book) {
    return $book->status=='PUBLISH';
});

// ambil 2 items buku secara random
$books->random(2)->all();
```

Catatan: fungsi keren lain pada *collection* yang bisa kita gunakan bisa dilihat pada tautan ini
<https://laravel.com/docs/5.7/eloquent-collections#available-methods>

Fungsi-fungsi pada *query builder* juga bisa kita gunakan di sini.

```
$books = \App\Book::all();
// ambil record pertama
$first_book = $books->first();

// ambil nilai dari atribut title pada record pertama
$title = $books->first()->value('title');
```

Selain menggunakan fungsi `first()` untuk mendapatkan *record* pertama saja, kita juga bisa gunakan fungsi `find()` sebagai berikut.

```
// select * from books where id = 1
$book = \App\Book::find(1);
echo $book->title;
```

Di samping itu, fungsi `find` juga bisa digunakan untuk mengambil beberapa *record* berdasarkan id atau primary key-nya.

```
$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.

```
$book = \App\Book::findOrFail(1);

$book = \App\Book::where('status', '=', 'DRAFT')->firstOrFail();
```

Berikut ini contoh penggunaan fungsi *aggregat* seperti: `count`, `max`, `avg`

```
$count = \App\Book::count();
$max_price = \App\Book::max('price'); // nilai maksimal
$avg_price = \App\Book::avg('price'); // rata-rata
```

Query lain seperti `insert`, `update`, `delete` pun dengan mudah bisa kita lakukan.

```
// insert into books (title, slug) values ('Learn VueJS', 'learn-vuejs')
$book = new \App\Book;
$book->title = 'Learn VueJS';
$book->slug = str_slug($book->title);
$book->save();
```

Catatan: `str_slug` adalah fungsi bawaan Laravel untuk merubah teks menjadi format URL slug.

```
// update books set price = 125000 where id =3
$book = \App\Book::find(3);
$book->price = 125000;
$book->save();
```

```
// attau bisa juga
\App\Book::where('id', 3)
->update([
    'price' => 125000
]);

// delete from books where id = 2
$book = \App\Book::find(2);
$book->delete()

// delete from books where id = 2
\App\Book::destroy(2);

// delete from books where id = 2 or 5
\App\Book::destroy([2,5]);

// delete from books where id = 2
\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*.

```
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.

```
protected $guarded = [];
```

Maka kemudian kita bisa gunakan

```
$book = \App\Book::create(
    ['title' => 'Judul 01', 'slug' => 'judul-01']
);

$book = \App\Book::create(
    ['title' => 'Judul 02', 'slug' => 'judul-02'],
    ['title' => 'Judul 03', 'slug' => 'judul-03'],
);

// update judul buku dengan id 26
```

```
$book = \App\Book::find(26);
$book->fill(
    ['title' => 'Judul 01 - updated']
);
```

Fitur lain yang menarik adalah fitur `updateOrCreate` yaitu update data namun jika data tidak ditemukan maka create/insert.

```
$book = \App\Book::updateOrCreate(
    ['id' => '28', 'title' => 'Test updateOrCreate', 'slug' => 'test-
updateorcreate'],
    ['price' => 99]
);
```

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`.

```
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;

class Book extends Model
{
    use SoftDeletes;
    protected $dates = ['deleted_at'];

    protected $fillable = [
        'title', 'slug', 'description', 'author', 'publisher',
        'cover', 'price', 'weight', 'stock', 'status'
    ];
}
```

Tambahkan untuk semua model yang menggunakan softdelete.

```
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;

class Category extends Model
{
    use SoftDeletes;
```

```
protected $dates = ['deleted_at'];

protected $fillable = [
    'name', 'slug', 'image', 'status'
];

}
```

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()`.

```
$book = \App\Book::find(1);
$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()`.

```
// semua record buku
$books = \App\Book::withTrashed()->get();

// semua record buku yang sudah dihapus
$books = \App\Book::onlyTrashed()->get();
```

Untuk me-*restore* data yang sudah terhapus, kita bisa gunakan fungsi `restore()`.

```
\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.

```
php artisan make:resource Book
```

Maka file Book.php akan terlihat sebagai berikut.

```
<?php
namespace App\Http\Resources;
use Illuminate\Http\Resources\Json\JsonResource;
class Book extends JsonResource
{
    public function toArray($request)
    {
        return parent::toArray($request);
    }
}
```

Fungsi `toArray()` di atas akan menampilkan semua field dari tabel. Namun kita bisa mendefinisikan secara spesifik, misalnya:

```
public function toArray($request)
{
    return [
        'id' => $this->id,
        'title' => $this->title,
        'created_at' => $this->created_at,
        'updated_at' => $this->updated_at,
    ];
}
```

Kemudian pada route/api.php, tambahkan routing ke resource tersebut, misalnya.

```
use App\Book;
use App\Http\Resources\Book as BookResource;

Route::get('/book', function () {
    return new BookResource(Book::find(1));
});
```

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.

```
public function view($id)
{
    //$book = DB::select('select * from books where id = ? ', [ $id ]);
    $book = new BookResource(Book::find($id));
    return $book;
}
```

Lalu akses dengan route sesuai dengan routing api.php sebelumnya.

```
Route::prefix('v1')->group(function () {
    Route::get('books', 'BookController@index');
    Route::get('book/{id}', 'BookController@view')->where('id', '[0-9]+');
});
```

Gunakan format URL endpoint ini `/v1/book/1`

GET ▾	http://larashop-api.test/v1/book/1	Params
Pretty	Raw	Preview
JSON ▾		

```

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 [{"data": {
2   "id": 1,
3   "title": "Reprehenderit ut molestias et deleniti molestiae ullam",
4   "created_at": {
5     "date": "2018-08-07 19:51:39.000000",
6     "timezone_type": 3,
7     "timezone": "UTC"
8   },
9   "updated_at": {
10    "date": "2018-08-07 22:33:13.000000",
11    "timezone_type": 3,
12    "timezone": "UTC"
13  }
14 }
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:

```

php artisan make:resource Books --collection
php artisan make:resource BookCollection

```

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.

```

<?php
namespace App\Http\Resources;
use Illuminate\Http\Resources\Json\ResourceCollection;
class Books extends ResourceCollection
{
    public function toArray($request)
    {
        return parent::toArray($request);
    }
}

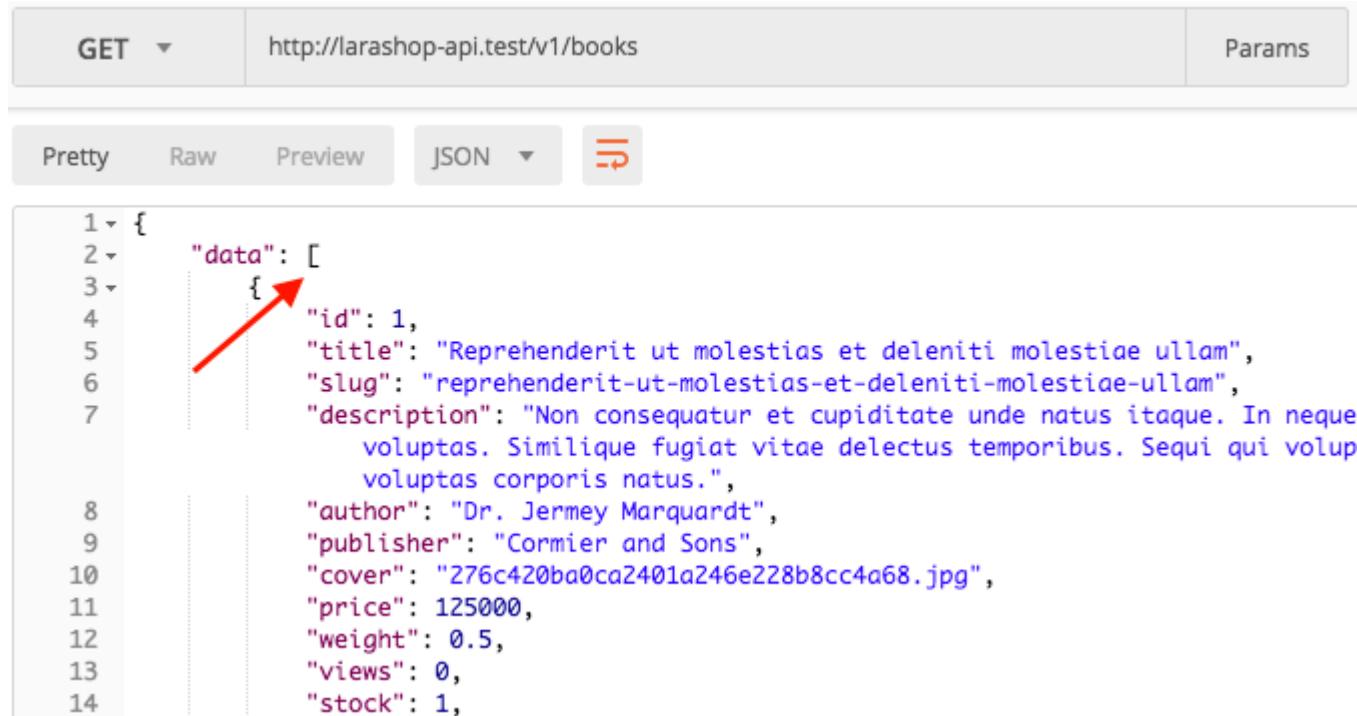
```

Buka BookController, pada fungsi `index()`, ubah kodennya menjadi sebagai berikut.

```
public function index(){
    $books = new BookCollectionResource(Book::get());
    return $books;
}
```

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:



GET ▾ http://larashop-api.test/v1/books Params

Pretty Raw Preview JSON ▾ ↲

```
1 - {
2 -   "data": [
3 -     {
4 -       "id": 1,
5 -       "title": "Reprehenderit ut molestias et deleniti molestiae ullam",
6 -       "slug": "reprehenderit-ut-molestias-et-deleniti-molestiae-ullam",
7 -       "description": "Non consequatur et cupiditate unde natus itaque. In neque voluptas. Similique fugiat vitae delectus temporibus. Sequi qui 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.

```
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

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

Atau bisa juga kita definisikan spesifik modelnya.

```
php artisan make:controller CategoryController --resource --model=Category
```

Hasilnya sebagai berikut. (komentar dan whitespace sudah penulis dihapus):

```
<?php
namespace App\Http\Controllers;
use App\Category;
use Illuminate\Http\Request;
class CategoryController extends Controller
{
    public function index()
    {
        //
    }

    public function create()
    {
        //
    }

    public function store(Request $request)
    {
        //
    }

    public function show(Category $category)
    {
        //
    }

    public function edit(Category $category)
    {
        //
    }

    public function update(Request $request, Category $category)
    {
        //
    }

    public function destroy(Category $category)
    {
        //
    }
}
```

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.

```
Route::resource('categories', 'CategoryController');
```

Atau untuk mendaftarkan lebih dari satu controller resource, kita bisa gunakan format array.

```
Route::resources([
    'categories' => 'CategoryController',
    'books' => 'BookController',
]);
```

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`.

```
Route::resource('categories', 'CategoryController')->only([
    'index', 'show'
]);

// atau

Route::resource('categories', 'CategoryController')->except([
    'create', 'store', 'update', 'destroy'
]);
```

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.

```
Route::apiResources([
    'categories' => 'CategoryController'
    'books' => 'BookController',
])
```

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.

```
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 sering kali 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.

```
public function render($request, Exception $exception)
{
    // baca konfigurasi apakah aplikasi menggunakan mode production atau
    // development
    $debug = config('app.debug');
    $message = '';
    $status_code = 500;
    // cek jika eksepsinya dikarenakan model tidak ditemukan
    if ($exception instanceof ModelNotFoundException) {
        $message = 'Resource is not found';
        $status_code = 404;
    }
    // cek jika eksepsinya dikarenakan resource tidak ditemukan
```

```

elseif ($exception instanceof NotFoundHttpException) {
    $message = 'Endpoint is not found';
    $status_code = 404;
}
// cek jika eksepsinya dikarenakan method tidak diizinkan
elseif ($exception instanceof MethodNotAllowedHttpException) {
    $message = 'Method is not allowed';
    $status_code = 405;
}
// cek jika eksepsinya dikarenakan kegagalan validasi
else if ($exception instanceof ValidationException) {
    $validationErrors = $exception->validator->errors()->getMessages();
    $validationErrors = array_map(function($error) {
        return array_map(function($message) {
            return $message;
        }, $error);
    }, $validationErrors);
    $message = $validationErrors;
    $status_code = 405;
}
// cek jika eksepsinya dikarenakan kegagalan query
else if ($exception instanceof QueryException) {
    if ($debug) {
        $message = $exception->getMessage();
    } else {
        $message = 'Query failed to execute';
    }
    $status_code = 500;
}
$rendered = parent::render($request, $exception);
$status_code = $rendered->getStatusCode();
if (empty($message)) {
    $message = $exception->getMessage();
}
$errors = [];
if ($debug) {
    $errors['exception'] = get_class($exception);
    $errors['trace'] = explode("\n", $exception->getTraceAsString());
}
return response()->json([
    'status'      => 'error',
    'message'    => $message,
    'data'        => null,
    'errors'      => $errors,
], $status_code);
}

```

Jangan lupa, sebelumnya kita harus **use** terlebih dahulu lima class tersebut.

```

use Illuminate\Database\Eloquent\ModelNotFoundException;
use Symfony\Component\HttpKernel\Exception\NotFoundHttpException;
use Symfony\Component\HttpKernel\Exception\MethodNotAllowedHttpException;

```

```
use Illuminate\Validation\ValidationException;
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.

The screenshot shows a Postman interface. The method is set to 'GET'. The URL is 'http://larashop-api.test/v1/url-palsu'. The 'Pretty' tab is selected, showing the JSON response:

```
{"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`.

```
protected function unauthenticated($request, AuthenticationException $exception)
{
    return response()->json([
        'status'      => 'error',
        'message'     => 'Unauthenticate',
        'data'        => null
    ], 401);
}
```

Hasilnya.

The screenshot shows a Postman interface. The method is set to 'GET'. The URL is 'http://larashop-api.test/user'. The 'Pretty' tab is selected, showing the JSON response:

```
{"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.

```
'defaults' => [
    'guard' => 'api', // <== update ini
    'passwords' => 'users',
],
'guards' => [
    'web' => [
        'driver' => 'session',
        'provider' => 'users',
    ],
    'api' => [
        'driver' => 'token', // <== pastikan ini
        'provider' => 'users',
    ],
],
```

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.

```
$user = Auth::user();
// Get user id
$id = Auth::id();
```

Check Authenticate User

Untuk mengecek apakah user sudah login atau belum maka bisa gunakan kode berikut.

```
if (Auth::check()) {
    // The user is logged in...
}
```

Protect Routing

Untuk memproteksi route tertentu yang hanya boleh diakses oleh user yang sudah login, maka kita bisa gunakan middleware berikut.

```
Route::get('profile', function () {
    // Hanya untuk user yang sudah login
})->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.

```
php artisan make:migration --table=users adds_api_token_to_users_table
```

Kemudian pada file migration kita tambahkan kode berikut.

```
public function up()
{
    Schema::table('users', function (Blueprint $table) {
        $table->string('api_token', 60)->unique()->nullable();
    });
}

public function down()
{
```

```
Schema::table('users', function (Blueprint $table) {
    $table->dropColumn(['api_token']);
});
```

Jalankan migration

```
php artisan migrate
```

```
laradock — root@8a48a85542c9: /var/www/larashop-api — docker ✧ docker-compose exec workspace bash
root@8a48a85542c9: /var/www/larashop-api — docker ✧ docker-compose exec...
~/Dev — root@8a48a85542c9: /var/www/larashop-api — docker ✧ docker-compose exec...
root@8a48a85542c9:/var/www# 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.

```
public function generateToken()
{
    $this->api_token = str_random(60);
    $this->save();
    return $this->api_token;
}
```

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.

```
php artisan make:controller AuthController
```

Pada `AuthController` ini kita akan buat fungsi login, register dan logout.

```
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class AuthController extends Controller
{
    public function login(Request $request)
    {
```

```

    }

    public function register(Request $request)
    {

    }

    public function logout(Request $request)
    {

    }
}

```

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`.

```

use Hash;
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()`.

```

public function login(Request $request)
{
    $user = User::where('email', '=', $request->email)->firstOrFail();
    $status = "error";
    $message = "";
    $data = null;
    $code = 401;
    if($user){
        // jika hasil hash dari password yang diinput user sama dengan
        // password di database user maka
        if (Hash::check($request->password, $user->password)) {
            // generate token
            $user->generateToken();
            $status = 'success';
            $message = 'Login sukses';
            // tampilkan data user menggunakan method toArray
            $data = $user->toArray();
            $code = 200;
        }
    }
}

```

```

    }
    else{
        $message = "Login gagal, password salah";
    }
}
else{
    $message = "Login gagal, username salah";
}

return response()->json([
    'status' => $status,
    'message' => $message,
    'data' => $data
], $code);
}

```

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.

```

Route::prefix('v1')->group(function () {
    // ...
    Route::post('login', 'AuthController@login');

    // tambahkan sekalian untuk register dan logout :
    Route::post('register', 'AuthController@register');
    Route::post('logout', 'AuthController@logout');
});

```

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.

```

1  {
2    "data": {
3      "id": 3,
4      "name": "Timmothy Howell",
5      "email": "mcglynny.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": "mM0nBNctiyFKMrzAoBr4pZu0MaTML6pYRCjdCkqqkEZgxip82y3PSFQw8gJ"
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 .

```

$this->validate($request, [
    'email' => 'required',
    'password' => 'required',
]);

```

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.

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> email	mcglynny.rey@example.org	
Key	Value	Description

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 penggunaannya Validatornya.

```
public function register(Request $request)
{
    $validator = Validator::make($request->all(), [
        'name' => 'required|string|max:255', // name harus diisi teks dengan panjang maksimal 255
        'email' => 'required|string|email|max:255|unique:users', // email harus unik pada tabel users
        'password' => 'required|string|min:6', // password minimal 6 karakter
    ]);
    if ($validator->fails()) { // fungsi untuk ngecek apakah validasi gagal
        // validasi gagal
    }
    else{
        // validasi sukses
    }
}
```

Ketika validasi gagal maka kita bisa tampilkan errornya dengan cara berikut.

```
if ($validator->fails()) {
    $errors = $validator->errors();
    return response()->json([
        'data' => [
```

```

        'message' => $errors,
    ]
], 400);
}

```

Mari kita lihat hasilnya.

POST <http://larashop-api.test/v1/register>

Params

Authorization Headers Body ● Pre-request Script Tests

form-data x-www-form-urlencoded raw binary

KEY	VALUE	DESCRIPTION
email	hafid@gmail.org	
password	1234	
name	Hafid	
Key	Value	Description

Body Cookies (2) Headers (13) Test Results Status: 400 Bad Request

Pretty Raw Preview

```
{"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).

```

$validator = Validator::make($request->all(), [
    'name' => 'required|string|max:255',
    'email' => 'required|string|email|max:255|unique:users',
    'password' => 'required|string|min:6',
]);

$status = "error";
$message = "";
$data = null;
$code = 400;
if ($validator->fails()) {
    $errors = $validator->errors();
    $message = $errors;
}
else{
    $user = \App\User::create([
        'name' => $request->name,

```

```

'email' => $request->email,
'password' => Hash::make($request->password),
'roles'      => json_encode(['CUSTOMER']),
]);
if($user){
    // Auth::login($user);
    $user->generateToken();
    $status = "success";
    $message = "register successfully";
    $data = $user->toArray();
    $code = 200;
}
else{
    $message = 'register failed';
}
}

return response()->json([
    'status' => $status,
    'message' => $message,
    'data' => $data
], $code);

```

Berikut ini tampilan jika, register berhasil.

The screenshot shows a Postman request for a POST to `http://larashop-api.test/v1/register`. The request body contains three fields: `email` (hafid@gmail.org), `password` (123456), and `name` (Hafid). The response status is `200 OK` with a time of `140 ms`. The response body is a JSON object:

```
{"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"}}
```

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

```
Route::middleware('auth:api')->get('/user', function (Request $request) {
    return $request->user();
});
```

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 screenshot shows a POSTMAN interface with a GET request to `http://larashop-api.test/user`. The `Authorization` tab is active, with the `TYPE` dropdown set to `Bearer Token` and the `Token` input field containing a long string of characters: `um4PikXSve4IH5Qi5HuTGX4YVJOU1W6wlVGtUQ7KvDMUmexxRbMGD...`.

Jika token benar maka akan menampilkan data user sebagaimana definisi routing di atas.

The screenshot shows a POSTMAN interface with a GET request to `http://larashop-api.test/user`. The `Headers` tab is active, showing a single header: `Authorization: Bearer um4PikXSve4IH5Qi5HuTGX4YVJOU1W6wlVGtUQ7KvDMUmexxRbMGD...`. The response status is `200 OK`.

```
{"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":"um4PikXSve4IH5Qi5HuTGX4YVJOU1W6wlVGtUQ7KvDMUmexxRbMGD..."}  
{"data":{"message":"Unauthenticated","status_code":401}}
```

Jika gagal maka akan menampilkan json message 'Unauthentication'.

The screenshot shows a POSTMAN interface with a GET request to `http://larashop-api.test/user`. The `Headers` tab is active, showing a single header: `Authorization: Bearer um4PikXSve4IH5Qi5HuTGX4YVJOU1W6wlVGtUQ7KvDMUmexxRbMGD...`. The response status is `401 Unauthorized`.

```
{"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.

```
public function logout(Request $request)
{
    $user = Auth::user();
    if ($user) {
        $user->api_token = null;
        $user->save();
    }
    return response()->json([
        'status' => 'success',
        'message' => 'logout berhasil',
        'data' => null
    ], 200);
}
```

Untuk routing, sekalian kita rapikan. Di dalam routing dengan prefix v1, kita bagi menjadi dua yaitu routing yang sifatnya public artinya siapapun boleh mengakses resourcenyanya dan routing yang sifatnya private atau hanya user yang berwenang saja yang boleh mengakses resourcenyanya. Pada routing private ini kita kelompokkan menggunakan middleware `auth:api`.

```
Route::prefix('v1')->group(function () {
    // public
    Route::post('login', 'AuthController@login');
    Route::post('register', 'AuthController@register');
    // ...

    // private
    Route::middleware('auth:api')->group(function () {
        Route::post('logout', 'AuthController@logout');
        //...
    });
});
```

Catatan: silakan hapus dulu definisi `routing-routing` terdahulu hasil latihan kita.

Waktunya mencoba, silakan akses routing `/logout` dengan header bearer token.

The screenshot shows a Postman interface with a successful API call. The request method is POST, the URL is <http://larashop-api.test/v1/logout>, and the response status is 200 OK. The Headers tab is selected, showing one header: Authorization with the value Bearer um4PikXSve4IH5Qi5HuTGX4YVJOU1W6wlVGtUQ7KvDMUmexxRbMGD... . The Body tab shows the JSON response: {"data": {"message": "logout berhasil", "status_code": 200}} . The Cookies, Headers, and Test Results tabs are also visible.

KEY	VALUE	DESCRIPTION
Authorization	Bearer um4PikXSve4IH5Qi5HuTGX4YVJOU1W6wlVGtUQ7KvDMUmexxRbMGD...	
Key	Value	Description

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 ini dulu sebelum merujuk ke source code lengkapnya supaya pemahaman kamu lebih baik lagi.

Global Constant

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).

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, misalnya sebagai berikut.

```
VUE_APP_NAME=Vueshop  
VUE_APP_BACKEND_URL=http://larashop-api.test  
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.

```
mounted(){  
    console.log(process.env)  
}
```

Lalu jalankan perintah `npm run serve` maka hasilnya akan sebagai berikut.

```
[HMR] Waiting for update signal from log.js?1afd: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 environtment production, kita bisa tambahkan file `env.production` dengan nilai yang berbeda, misalnya sebagai berikut.

```
VUE_APP_NAME=Vueshop
VUE_APP_BACKEND_URL=http://api.larashop.id
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 tersembunyi di sisi kiri yang bisa ditampilkan.

Pada konsep scaffolding Vue CLI, kode untuk layout aplikasi bisa kita jumpai pada file `App.vue`, yaitu sebuah component yang akan dimuat pertama kali. Merujuk pada layout di atas maka mari kita ubah kode `App.vue` menjadi sebagai berikut.

```
<template>
<v-app>
  <!-- component header -->
  <c-header />

  <!-- component sidebar -->
  <c-side-bar />

  <!-- konten utama -->
  <v-content>
```

```

<v-slide-y-transition mode="out-in">
  <router-view></router-view>
</v-slide-y-transition>
</v-content>

<!-- component footer -->
<c-footer />

</v-app>
</template>

<script>
export default {
  name: 'App'
}
</script>

```

Yap, kita akan menggunakan konsep component untuk mengatur layout supaya lebih mudah dalam pengelolaan kode sumbernya. Component yang namanya diawali dengan `v-` merupakan component bawaan Vuetify. Di bagian tengah jika kita perhatikan menggunakan component `router-view` yaitu component Vue Router untuk menampilkan page atau view berdasarkan suatu routing, di mana component tersebut kita bungkus dengan `v-slide-y-transition` yang merupakan component Vuetify yang berperan menambahkan efek transisi.

Catatan: untuk component-component yang berasal dari Vuetify maka kamu sebaiknya langsung merujuk ke dokumentasi Vuetify karena pada bab ini tidak akan dijelaskan secara detail terutama terkait dengan varian dan cara modifikasinya.

Adapun component yang diawali dengan `c-` merupakan component yang akan kita buat sendiri dan kita simpan dalam direktori `src/components` yaitu `c-header`, `c-side-bar`, dan `c-footer`. Oleh karena itu, berikut ini akan kita buat satu persatu component tersebut.

Component C-Header

Pada component C-Header ini, kita akan menggunakan component standard Vuetify untuk membuat header yaitu `v-toolbar`. Header ini dibagi menjadi tiga bagian yaitu bagian kiri berisi tombol untuk menampilkan sidebar `v-toolbar-side-icon`, bagian tengah untuk menampilkan judul aplikasi `v-toolbar-title` serta bagian kanan untuk menampilkan tombol keranjang belanja yang umum ada pada aplikasi toko online `v-btn`, `v-icon` dan `v-badge`. Supaya lebih lengkap maka pada bagian bawahnya kita tambahkan kolom pencarian `v-text-field`.

Adapun kode lengkapnya kira-kira sebagai berikut.

```

<template>
  <!-- toolbar vuetify dengan warna primary -->
  <v-toolbar dark color="primary">

    <!-- header bagian kiri -->
    <v-toolbar-side-icon></v-toolbar-side-icon>

```

```
<!-- header bagian tengah -->
<v-toolbar-title class="white--text">Vueshop</v-toolbar-title>

<!-- separator supaya header bagaian kanan posisinya rata kanan -->
<v-spacer></v-spacer>

<!-- header bagian kanan -->
<v-btn icon>
  <v-badge left overlap color="orange">
    <span slot="badge">1</span>
    <v-icon>shopping_cart</v-icon>
  </v-badge>
</v-btn>

<!-- kolom pencarian di bawah header -->
<v-text-field
  slot="extension"
  hide-details
  append-icon="mic"
  flat
  label="Search"
  prepend-inner-icon="search"
  solo-inverted
></v-text-field>

</v-toolbar>
</template>
```

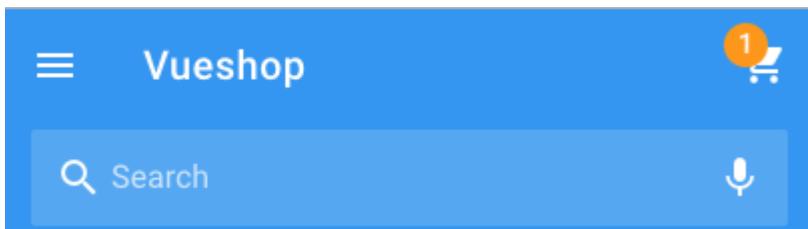
Catatan: sekali lagi silakan merujuk ke dokumentasi Vuetify untuk memahami pengaturan componentnya.

Kode ini kita simpan dengan nama `CHeader.vue` pada direktori `src/components`. Lalu kembali ke `App.vue`, kita akan muat atau import component ini ke layout utama.

```
<script>
import CHeader from '@/components/CHeader.vue'

export default {
  name: 'App',
  components: {
    CHeader
  },
}
</script>
```

Jika kita lihat hasilnya, kira-kira sebagai berikut.



Component C-Footer

Pada component ini, kita akan menggunakan contoh footer pada dokumentasi Vuetify. Pada intinya footer dibagi menjadi empat bagian yaitu menu media sosial, deskripsi aplikasi, menu halaman dan teks copyright.

```
<template>
<v-footer dark height="auto">
  <v-flex xs12>
    <v-card
      flat tile
      class="secondary darken-1 white--text text-xs-center">
      >

      <!-- menu icon media sosial media -->
      <v-card-text>
        <v-btn
          v-for="icon in icons"
          :key="icon"
          class="mx-3 white--text"
          icon
        >
          <v-icon size="24px">{{ icon }}</v-icon>
        </v-btn>
      </v-card-text>

      <!-- deskripsi aplikasi -->
      <v-card-text v-if="isHome" class="white--text pt-0">
        Lorem ipsum sit dolor amet ...
      </v-card-text>

      <v-layout justify-center row wrap>
        <!-- link menu halaman aplikasi -->
        <v-btn
          v-for="link in links"
          :key="link"
          color="white"
          flat
          round
        >
          {{ link }}
        </v-btn>

        <!-- teks copyright -->
        <v-flex
          secondary

```

```

        darken-2
        py-3
        text-xs-center
        white--text
        xs12
    >
        &copy;2018 – Vueshop powered by <strong>Vuetify</strong>
    </v-flex>
    </v-layout>
</v-card>
</v-flex>
</v-footer>
</template>

<script>
export default {
  data: () => ({
    // variabel icon sosial media yang ingin ditampilkan
    icons: [
      'fab fa-facebook',
      'fab fa-twitter',
      'fab fa-google-plus',
      'fab fa-linkedin',
      'fab fa-instagram'
    ],
    // variabel link halaman yang ingin ditampilkan
    links: [
      'Home',
      'About Us',
      'Team',
      'Services',
      'Blog',
      'Contact Us'
    ]
  })
}
</script>

```

Icons, links dan deskripsi aplikasi pada kode di atas hanyalah dummy, karena memang secara umum, footer pada study kasus ini hanya sebagai kosmetik saja. Silakan bereksplorasi.

Kode di atas kita simpan dengan nama **CFooter.vue** pada direktori **src/components**. Lalu pada **App.vue**, kita akan muat atau import component ini ke layout utama sebagaimana component CHeader.

```

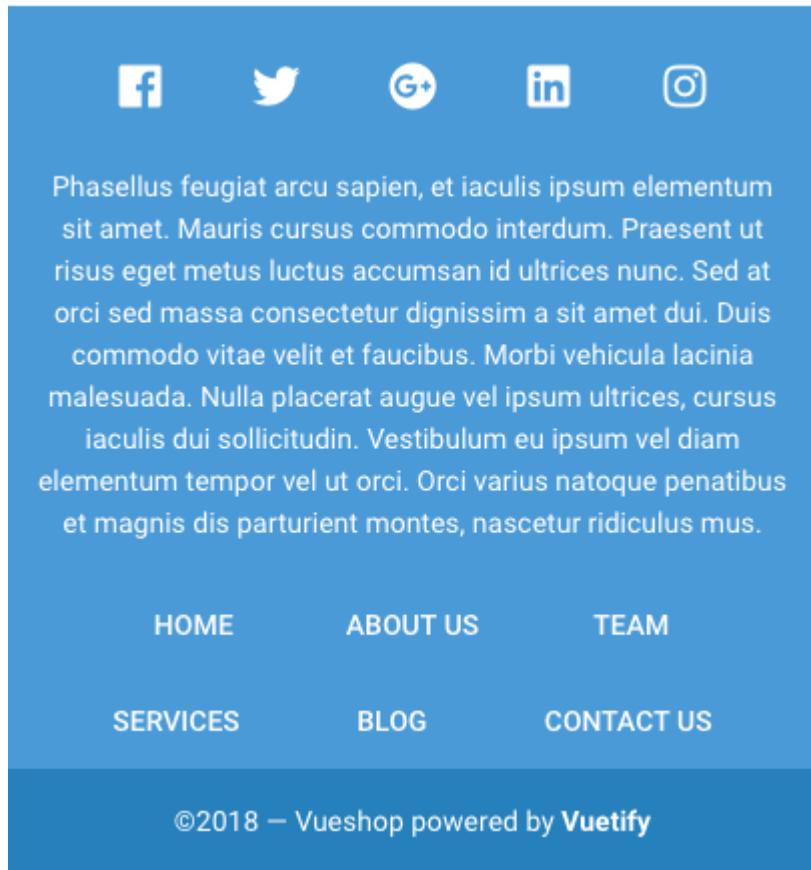
<script>
import CHeader from '@/components/CHeader.vue'
import CFooter from '@/components/CFooter.vue' // <= ini

export default {
  name: 'App',
  components: {

```

```
CHeader ,  
CFooter // <= ini  
,  
}  
</script>
```

Jika kita lihat hasilnya, kira-kira sebagai berikut.



Catatan: pada file `public/index.html` bagian `<head>` tambahkan pustaka fontawesome karena logo sosial media di atas menggunakan icon dari fontawesome.

```
<link href="https://use.fontawesome.com/releases/v5.0.8/css/all.css"  
rel="stylesheet">
```

Bagaimana? make sense?

Component C-Side-Bar

Component sidebar merupakan component yang ditampilkan secara overlay di sisi kiri layout aplikasi. Triggernya melalui tombol pada bagian kiri component header. Pada vuetify, untuk membuat sidebar cukup mudah yaitu melalui component `v-navigation-drawer`. Component ini akan kita pecah menjadi tiga bagian yaitu toolbar, tombol login/register, dan menu navigasi.

```
<template>  
<v-navigation-drawer v-model="drawer" absolute fixed clipped>
```

```
<!-- header toolbar pada sidebar supaya lebih cantik -->
<v-toolbar dark color="primary">
  <v-btn icon dark>
    <v-icon>close</v-icon>
  </v-btn>
  <v-toolbar-title>Vueshop</v-toolbar-title>
</v-toolbar>

<v-list>
  <v-list-tile>
    <!-- tombol register -->
    <v-btn
      depressed
      block
      round
      color="secondary"
      class="white--text"
    >
      Register
      <v-icon right dark>person_add</v-icon>
    </v-btn>
  </v-list-tile>
  <v-list-tile>
    <!-- tombol login -->
    <v-btn
      block
      round
      depressed
      color="accent lighten-1"
      class="white--text"
    >
      Login
      <v-icon right dark>lock_open</v-icon>
    </v-btn>
  </v-list-tile>
</v-list>

<v-list class="pt-0" dense>
  <v-divider></v-divider>
  <!-- menu navigasi pada properti data items -->
  <v-list-tile
    v-for="(item,index) in items"
    :key="index"
    :href="item.route"
    :to="{name: item.route}"
  >
    <v-list-tile-action>
      <v-icon>{{ item.icon }}</v-icon>
    </v-list-tile-action>

    <v-list-tile-content>
      <v-list-tile-title>{{ item.title }}</v-list-tile-title>
    </v-list-tile-content>
  </v-list-tile>
```

```

        </v-list>

    </v-navigation-drawer>
</template>
<script>
export default {
    name: 'c-side-bar',
    data: () => ({
        // variabel untuk mengontrol visibilitas dari sidebar
        drawer: true,
        // variabel berisi daftar menu navigasi aplikasi
        items: [
            { title: 'Home', icon: 'dashboard', route: 'home' },
            { title: 'About', icon: 'question_answer', route: 'about' },
        ]
    }),
}
</script>

```

Pada kode di atas, component `v-navigation-drawer` di control visibilitasnya melalui v-model `drawer`, sehingga terlihat atau tersembunyinya sidebar tergantung nilai dari properti data `drawer` tersebut. Saklar pemicu (trigger) terhadap on atau offnya sidebar ini nantinya akan kita letakkan pada component `CHeader` di mana kita akan tambahkan mekanisme komunikasi antar component untuk mengubah nilai dari v-model `drawer` tersebut. Pada kode di atas, v-model belum akan terhubung dengan triggernya sehingga defaultnya kita set bernilai true supaya kita bisa melihat tampilannya.

Kode ini kita simpan dengan nama `CSideBar.vue` pada direktori `src/components`, tentu saja kemudian pada `App.vue`, kita akan muat component ini ke layout utama.

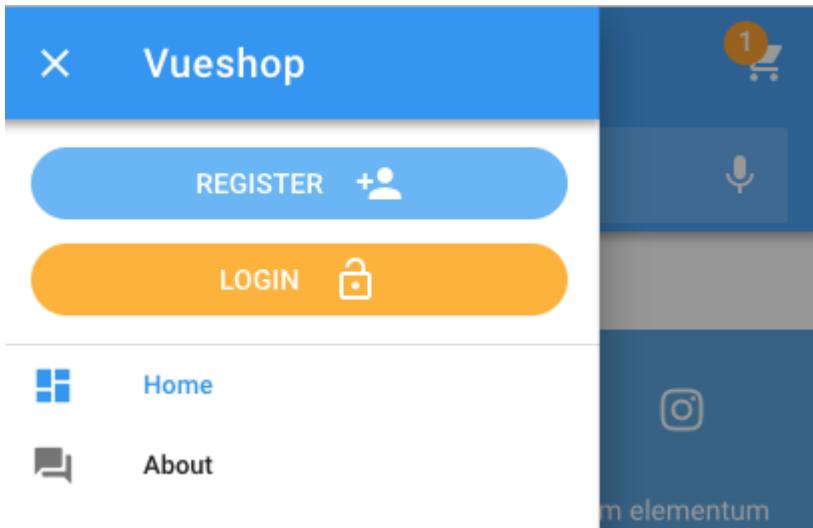
```

<script>
import CHeader from '@/components/CHeader.vue'
import CFooter from '@/components/CFooter.vue'
import CSideBar from '@/components/CSideBar.vue'

export default {
    name: 'App',
    components: {
        CHeader,
        CFooter,
        CSideBar
    },
}
</script>

```

Jika kita lihat hasilnya, kira-kira sebagai berikut.



Nah, sekarang kita akan mencoba untuk menghubungkan antara v-model drawer ini dengan triggernya pada component **CHeader** melalui state (vuex) meski sebenarnya bisa saja kita menggunakan pendekatan props namun tentu akan lebih mudah dan terkontrol jika menggunakan state.

Kode state disimpan dalam file `src/store.js`. Mari kita buat state, mutation, action serta getters untuk menangani visibilitas dari navigation drawer.

```
import Vue from 'vue'
import Vuex from 'vuex'

Vue.use(Vuex)

export default new Vuex.Store({
  state: {
    sideBar: false,
  },
  mutations: {
    setSideBar: (state, value) => {
      state.sideBar = value
    },
  },
  actions: {
    setSideBar: ({commit}, value) => {
      commit('setSideBar', value)
    },
  },
  getters: {
    sideBar: state => state.sideBar,
  },
})
```

Intinya sih hanya mengeset apakah nilai state variabel `sideBar`-nya true atau false. Kita tau bahwa actions dan getters sebenarnya opsional, namun pada studi kasus ini akan penulis contohkan cara yang baik supaya alur perubahan state menjadi jelas dan terstruktur yaitu action dan getters lah yang akan diakses secara langsung dari component sedangkan state dan mutation tidak dapat diakses secara langsung.

Mari kita kembali pada component `CSideBar`. Pada bagian script, kita modifikasi menjadi sebagai berikut.

```
// import map getter dan map action dari vuex
import { mapGetters, mapActions } from 'vuex'
export default {
  name: 'c-side-bar',
  data: () => ({
    items: [
      { title: 'Home', icon: 'dashboard', route: 'home' },
      { title: 'About', icon: 'question_answer', route: 'about' },
    ]
  }),
  computed: {
    // mapping state sideBar menggunakan map getter
    ...mapGetters({
      sideBar: 'sideBar',
    }),
    // ubah properti data drawer menjadi computed dimana nilainya membaca
    // dari state sideBar
    drawer: {
      get () {
        return this.sideBar
      },
      set (value) {
        this.setSideBar(value)
      }
    },
  },
  methods: {
    // mapping action setSideBar pada store menggunakan map action
    ...mapActions({
      setSideBar: 'setSideBar',
    }),
  },
}
```

Pertama kita menggunakan `mapGetters` dan `mapActions` pada vuex untuk memudahkan kita dalam memetakan fungsi getter dan action pada store. Nah kemudian variabel `drawer` kita pindahkan dari properti data ke `computed` yang kita definisikan `getter` dan `setter`nya. Getter `drawer` akan membaca data `sideBar` dari `getters` store, sedangkan `setter` akan menjalankan (`dispatch`) `methods/actions` `setSideBar()` pada store yang telah dipetakan ke component.

Langkah berikutnya masih pada component `CSideBar`, pada tombol close (x) kita akan tambahkan prosedur untuk mengeset nilai dari variabel `drawer` menjadi `false` (menyembunyikan sidebar), perintahnya adalah `@click="drawer=false"`. Lebih lengkapnya sebagai berikut.

```
...
<v-toolbar dark color="primary">
  <v-btn icon dark @click="drawer=false">
```

```
<v-icon>close</v-icon>
</v-btn>
<v-toolbar-title>Vueshop</v-toolbar-title>
</v-toolbar>
...

```

Sekarang kita akan berpindah ke component **CHeader** untuk menambahkan trigger yang akan men-dispatch (memanggil) action `setSideBar`. Sebagaimana pada bagian sebelumnya, kita akan petakan dahulu variabel `state sideBar` dan action `setSideBar` store pada component **CHeader**.

```
<script>
import { mapGetters, mapActions } from 'vuex'
export default {
  name: 'c-header',
  methods: {
    ...mapActions({
      setSideBar : 'setSideBar',
    }),
  },
  computed: {
    ...mapGetters({
      sideBar : 'sideBar',
    }),
  }
}
</script>
```

Kemudian pada tombol on/off sidebar di **CHeader**, kita tambahkan trigger tersebut dengan menggunakan event `onclick="@click='setSideBar(!sideBar)"`.

```
<v-toolbar-side-icon @click="setSideBar(!sideBar)"></v-toolbar-side-
icon>
```

Tanda ! pada javascript artinya menegaskan, sehingga perintah `setSideBar(!sideBar)` akan menjalankan fungsi sebaliknya dari kondisi sidebar, jadi jika sidebar sedang tampil maka akan disembunyikan, namun jika sidebar sedang disembunyikan maka akan ditampilkan.

Silakan dicoba, ketika tombol **v-toolbar-side-icon** diklik maka akan muncul sidebar ketika tombol close pada sidebar diklik lagi maka sidebar akan disembunyikan.

Content: Home & About

Pada bagian ini kita akan membuat component pada content. Component pada bagian ini berupa halaman yang akan kita simpan pada direktori **src/views**. Ada dua component yang akan kita buat yaitu Home dan About untuk sekedar menguji apakah navigasi berjalan lancar.

Berikut ini kode untuk file **Home.vue**

```
<template>
<div>
  <v-container>
    <h1>This is an home page</h1>
  </v-container>
</div>
</template>
```

Sedangkan untuk file `About.vue`, kodenya seperti berikut ini.

```
<template>
<div>
  <v-container>
    <h1>This is an about page</h1>
  </v-container>
</div>
</template>
```

Kali ini component Home dan About ini tidak kita daftarkan pada `App.vue` sebagaimana component CHeader dan CSideBar melainkan akan kita daftarkan pada routing. Hal ini karena output dari dua component ini nantinya akan kita tampilkan pada component `<router-view>`.

Mari kita buka file `src/router.js`

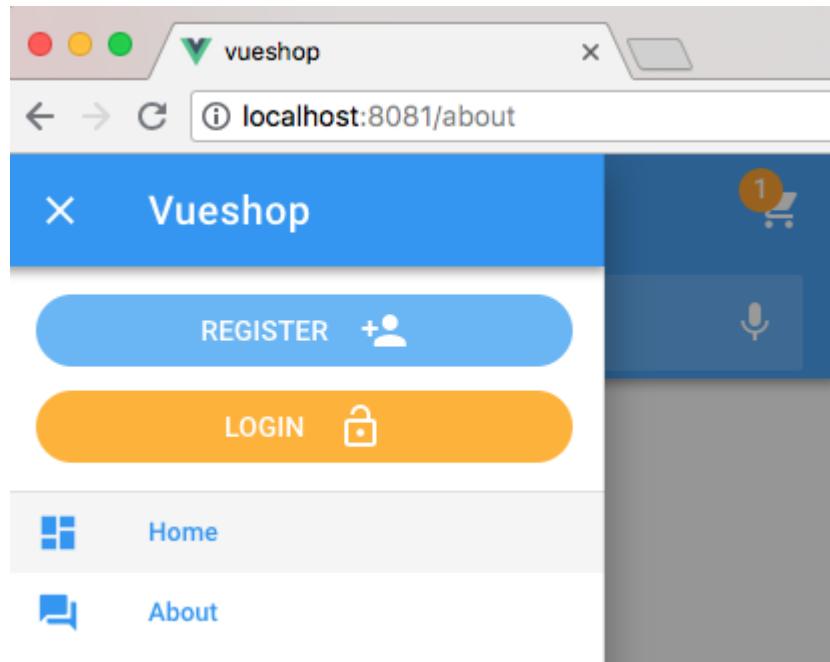
```
import Vue from 'vue'
import Router from 'vue-router'
// import component Home dan About
import Home from './views/Home.vue' // boleh juga -> import CHeader from
'@/views/Home.vue'
import About from './views/About.vue'

Vue.use(Router)

const router = new Router({
  mode: 'history',
  base: process.env.BASE_URL,
  routes: [
    // jika routenya / maka component yang akan ditampilkan pada router-
    view adalah Home
    {
      path: '/',
      name: 'home',
      component: Home
    },
    {
      path: '/about',
      name: 'about',
      component: About
    }
  ]
})
```

```
},
// jika routenya apapun selain definisi di atas maka component yang
akan ditampilkan pada router-view
// adalah Home, route default untuk mencegah error
{
  path: '*',
  redirect: {
    name: 'home'
  }
},
],
})
export default router
```

Mari kita coba kode di atas dengan mengakses route home dan about melalui menu pada sidebar.



Setelah menu Home di klik maka akan muncul sebagai berikut.

This is an home page

Phasellus feugiat arcu sapien, et iaculis ipsum elementum sit amet. Mauris cursus commodo interdum. Praesent ut risus eget metus luctus accumsan id ultrices nunc. Sed at orci sed massa consectetur dignissim a sit amet dui. Duis commodo vitae velit et faucibus. Morbi vehicula lacinia malesuada. Nulla placerat augue vel ipsum ultrices, cursus iaculis dui sollicitudin. Vestibulum eu ipsum vel diam elementum tempor vel ut orci. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

HOME ABOUT US TEAM
SERVICES BLOG CONTACT US

©2018 – Vueshop powered by Vuetify

Gimana? make sense kan?

Update 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 empat kategori buku yang diambil secara random dalam bentuk list grid atau card, di mana setiap itemnya akan ditampilkan gambar dari kategori dan teks judulnya. Pada bagian atas sebelah kanan heading, kita akan tampilkan teks link [See All](#) yang akan mengarahkan user ke halaman daftar lengkap dari kategori buku .

Pada bagian bawahnya, akan menampilkan delapan 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 [See All](#) yang akan mengarahkan user ke halaman daftar seluruh buku yang tersedia.

Kode halaman home ini tetap diletakkan dalam file `Home.vue` pada direktori `/src/views`.

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 Random Category

Endpoint ini sifatnya publik artinya bebas diakses oleh siapapun tanpa perlu login dahulu. Sebelumnya, kita perlu buat dulu resource collection untuk model Category.

```
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.

```
public function toArray($request)
{
    return [
        'status'  => 'success',
        'message' => 'categories data',
        'data'     => parent::toArray($request),
    ];
}
```

Kemudian kita buat controller Category.

```
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()`.

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
```

```
use App\Category;
use App\Http\Resources\Categories as CategoryResourceCollection;

class CategoryController extends Controller
{
    public function random($count)
    {
        $criteria = Category::select('*')
            ->inRandomOrder()
            ->limit($count)
            ->get();
        return new CategoryResourceCollection($criteria);
    }
}
```

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**.

```
<?php
use Illuminate\Http\Request;

Route::prefix('v1')->group(function () {
    // public
    Route::post('login', 'AuthController@login');
    Route::post('register', 'AuthController@register');

    Route::get('categories/random/{count}', 'CategoryController@random');
    // <== ini ya gaes

    // private
    Route::middleware(['auth:api'])->group(function () {
        Route::post('logout', 'AuthController@logout');
    });
});
```

Mari kita uji coba routing <http://larashop-api.test/v1/categories/random/4> menggunakan postman. Maka hasilnya kurang lebih sebagai berikut.

```
{
    "status": "success",
    "message": "categories data",
    "data": [
        {
            "id": 3,
```

```

        "name": "molestias",
        "slug": "molestias",
        "image": "3134bd94e92e26ed96ea95b65a7fed5f.jpg",
        "status": "PUBLISH",
        "created_at": "2018-08-28 07:00:46",
        "updated_at": null,
        "deleted_at": null,
        "created_by": null,
        "updated_by": null,
        "deleted_by": null
    },
    {
        "id": 5,
        "name": "accusamus",
        ...
    },
    {
        "id": 1,
        "name": "quia",
        ...
    },
    {
        "id": 4,
        "name": "quidem",
        ...
    }
]
}

```

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.

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.

```
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.

```

public function toArray($request)
{
    return [
        'status' => 'success',
        'message' => 'books data',
        'data' => parent::toArray($request),
    ];
}

```

```
];
}
```

Kemudian kita buat controller Book dengan menggunakan perintah berikut.

```
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.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Book;
use App\Http\Resources\Books as BookResourceCollection;

class BookController extends Controller
{
    public function top($count)
    {
        $criteria = Book::select('*')
            ->orderBy('views', 'DESC')
            ->limit($count)
            ->get();
        return new BookResourceCollection($criteria);
    }
}
```

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`.

```
<?php
use Illuminate\Http\Request;

Route::prefix('v1')->group(function () {
    // public
    Route::post('login', 'AuthController@login');
    Route::post('register', 'AuthController@register');

    Route::get('categories/random/{count}', 'CategoryController@random');
```

Licensed to M Najamudin Ridha - admin@komputerkampus.com - 087814493571 at 01/12/2018 19:54:30
Route::get('books/top/{count}', 'BookController@top'); // <= ini ya

```
// private
Route::middleware(['auth:api'])->group(function () {
    Route::post('logout', 'AuthController@logout');
});
});
```

Mari kita uji coba routing <http://larashop-api.test/v1/books/top/6> menggunakan postman. Maka hasilnya kurang lebih sebagai berikut.

```
{
    "status": "success",
    "message": "books data",
    "data": [
        {
            "id": 1,
            "title": "Quaerat et libero nisi",
            "slug": "quaerat-et-libero-nisi",
            "description": "Et officia vel unde qui enim voluptate quidem. Voluptatem nemo corporis et repudiandae qui voluptas ullam. Optio sit accusamus libero nostrum maiores placeat labore suscipit. Ducimus velit fugit tempora ipsa temporibus.",
            "author": "Jessie Barton",
            "publisher": "Lesch PLC",
            "cover": "86d8dcf0e10fa269f91eb320f8edd3c4.jpg",
            "price": 350000,
            "weight": 0.5,
            "views": 0,
            "stock": 0,
            "status": "PUBLISH",
            "created_at": "2018-08-28 06:59:30",
            "updated_at": "2018-09-01 22:46:51",
            "deleted_at": null,
            "created_by": null,
            "updated_by": null,
            "deleted_by": null
        },
        {
            "id": 2,
            "title": "Officiis assumenda unde",
            ...
        },
        {
            "id": 3,
            "title": "Aut delectus dolores similique",
            ...
        },
        {
            "id": 4,
            "title": "Placeat esse rem sunt nihil dolor",
            ...
        }
    ]
}
```

```

    ...
},
{
  "id": 5,
  "title": "Praesentium et rerum beatae",
  ...
},
{
  "id": 6,
  "title": "Ut quo omnis ratione velit",
  ...
}
]
}

```

Template & Script Home

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.

Pertama kita susun dahulu templatennya.

Sebagaimana konsep awal bahwa halaman Home ini akan kita bagi menjadi dua bagian yaitu daftar category dan daftar book yang masing-masing mempunyai subheader dan link ke daftar lengkapnya (router-link). Di sini kita akan menggunakan component standard vuetyf seperti v-flex untuk membuat tampilan kolom yang dikontrol oleh v-layout. Di dalamnya kita bisa gunakan v-card supaya lebih ciamik saja. Tentu untuk mendapatkan gambaran lebih detailnya silakan merujuk ke dokumentasi vuetyf.

Berikut ini kode template `Home.vue`

```

<template>
  <div>
    <!-- Bagian pertama yaitu Category -->
    <v-container grid-list-md>
      <v-subheader>
        Random Category
        <v-spacer></v-spacer>
        <!-- link ke route categories yang nantinya akan kita definisikan
        routing dan componentnya -->
        <router-link to="/categories">See All</router-link>
      </v-subheader>

      <v-layout row wrap>
        <!-- variabel categories ini nanti akan kita isi dengan data dari
        endpoint category -->
        <v-flex v-for="category in categories" xs6 :key="category.id">
          <v-card :to="'/category/'+ category.slug">
            <!-- untuk load image supaya lebih rapi akan kita buatkan method
            getImage -->
            <v-card-media
              :src="getImage('/categories/'+category.image)">
        </v-flex>
      </v-layout>
    </v-container>
  </div>
</template>
<script>
  export default {
    data() {
      return {
        categories: [
          {
            id: 1,
            title: 'Praesentium et rerum beatae',
            image: 'category1.jpg'
          },
          {
            id: 2,
            title: 'Ut quo omnis ratione velit',
            image: 'category2.jpg'
          },
          {
            id: 3,
            title: 'Quod est enim ratione velit',
            image: 'category3.jpg'
          }
        ]
      }
    }
  }
</script>
<style>
</style>

```

```

        height="150px"
    >
    <v-container fill-height fluid pa-2>
        <v-layout fill-height>
            <v-flex xs12 align-end flexbox>
                <!-- nama category-nya akan ditampilkan di sini --&gt;
                &lt;span class="title white--text text-block" v-
text="category.name"&gt;&lt;/span&gt;
            &lt;/v-flex&gt;
        &lt;/v-layout&gt;
    &lt;/v-container&gt;
&lt;/v-card-media&gt;

<!-- icon dummy saja, nantinya kamu bisa sesuaikan --&gt;
&lt;v-card-actions&gt;
    &lt;v-spacer&gt;&lt;/v-spacer&gt;
    &lt;v-btn icon&gt;
        &lt;v-icon&gt;favorite&lt;/v-icon&gt;
    &lt;/v-btn&gt;
    &lt;v-btn icon&gt;
        &lt;v-icon&gt;bookmark&lt;/v-icon&gt;
    &lt;/v-btn&gt;
    &lt;v-btn icon&gt;
        &lt;v-icon&gt;share&lt;/v-icon&gt;
    &lt;/v-btn&gt;
&lt;/v-card-actions&gt;

&lt;/v-card&gt;
&lt;/v-flex&gt;
&lt;/v-layout&gt;
&lt;/v-container&gt;

<!-- Bagian kedua yaitu Book --&gt;
&lt;v-container grid-list-md&gt;
    &lt;v-subheader&gt;
        Top Books
        &lt;v-spacer&gt;&lt;/v-spacer&gt;
        &lt;!-- link ke route books yang nantinya akan kita definisikan routing
        dan componentnya --&gt;
        &lt;router-link to="/books"&gt;See All&lt;/router-link&gt;
    &lt;/v-subheader&gt;
    &lt;v-layout row wrap&gt;
        &lt;!-- data buku kita tampilkan dalam dua kolom (xs6) --&gt;
        &lt;v-flex
            v-for="(book, index) in books" xs6 :key="index"&gt;
            &lt;v-card :to="'/book/'+ book.slug"&gt;
                &lt;v-card-media
                    :src="getImage('/books/'+book.cover)"
                    height="150px"
                &gt;
                &lt;v-container fill-height fluid pa-2&gt;
                    &lt;v-layout fill-height&gt;
                        &lt;v-flex xs12 align-end flexbox&gt;
                            &lt;span class="title white--text text-block" v-
</pre>

```

```

text="book.title"></span>
        </v-flex>
    </v-layout>
</v-container>
</v-card-media>

<v-card-actions>
    <v-spacer></v-spacer>
    <v-btn icon>
        <v-icon>favorite</v-icon>
    </v-btn>
    <v-btn icon>
        <v-icon>bookmark</v-icon>
    </v-btn>
    <v-btn icon>
        <v-icon>share</v-icon>
    </v-btn>
</v-card-actions>
</v-card>
</v-flex>
</v-layout>
</v-container>
</div>
</template>
<style scoped>
/* mengatur posisi judul */
.text-block {
    position: absolute;
    bottom: 5px;
    left: 5px;
    background-color: black;
    padding-left: 5px;
    padding-right: 5px;
    opacity: 0.7;
    width: 100%;
}
</style>

```

Component `v-card-actions` yang digunakan di atas hanya tambahan saja dan belum dimaksimalkan pada buku ini, silakan di eksplorasi sendiri. CSS class `.text-block` kita gunakan untuk mengatur posisi judul dari kategori atau buku pada gambarnya.

Untuk gambar dari book dan category, karena yang disimpan dalam database hanya nama filenya, maka kita perlu tambahkan alamat lengkapnya, karenanya kita gunakan fungsi `getImage()` supaya lebih ringkas kodennya. File gambar category diletakkan di server pada direktori `images/categories` sedangkan book pada direktori `images/books`.

Pada contoh ini, kita akan menggunakan axios untuk merequest data category dan book ke endpoint server, namun sebelumnya pastikan base URL web service-nya telah kita set pada file `src/plugins/axios.js`

```
let config = {
  baseURL: 'http://larashop-api.test/v1',
};
```

Atau kita bisa menggunakan konstanta global `.env` supaya lebih dinamis lagi.

```
let config = {
  baseURL: process.env.VUE_APP_API_URL + '/v1',
};
```

Berikut ini kode scriptnya `Home.vue`

```
<script>
export default {
  data: () => ({
    categories: [], // kita definisikan sebagai array kosong
    books: []
  }),
  methods: {
    getImage (image){
      if(image!=null && image.length>0){
        return "http://larashop-api.test/images"+ image
      }
      // default image jika tidak ditemukan,
      // letakkan image ini pada folder /public/img di project Vue
      return "/img/unavailable.png"
    },
  },
  created(){
    let count = 4
    // request ke endpoint category random dengan parameter count = 4
    this.axios.get('/categories/random/'+count)
      .then((response) => {
        let categories = response.data.data
        // ketika dapat datanya maka nilainya dimasukkan ke dalam
        properti data categories
        this.categories = categories
      })
      .catch((error) => {
        let responses = error.response
        console.log(responses)
      })
    count = 8
    // request ke endpoint top book dengan parameter count = 8
    this.axios.get('/books/top/'+count)
      .then((response) => {
        let books = response.data.data
        this.books = books
      })
  }
}
```

```
        })
        .catch((error) => {
            let responses = error.response
            console.log(responses)
        })
    }
</script>
```

Pada hook created, kita gunakan axios untuk request data category dan book.

Fungsi getImage() pada properti methods di atas masih menggunakan hardcoded untuk alamat backendnya, sebaiknya kita ubah menggunakan env.

```
getImage (image){
    if(image!=null && image.length>0){
        return process.env.VUE_APP_BACKEND_URL+ "/images" + image
    }
    // default image jika tidak ditemukan
    return "/img/unavailable.png"
},
```

Ehm tunggu, fungsi getImage ini pasti akan banyak digunakan pada component lain yang berkaitan dengan gambar buku atau category, karena itu kita bisa saja mendaftarkannya sebagai global method menggunakan mixins atau yang lainnya. Namun, kali ini penulis akan mencontohkan kepadamu melalui pendekatan plugins.

Buat file `helper.js` pada direktori `src/plugins`.

```
"use strict"
import Vue from 'vue'
const Helper = {
    install(Vue) {
        Vue.prototype.appName = process.env.VUE_APP_NAME

        Vue.prototype.getImage = function (image){
            if(image!=null && image.length>0){
                return process.env.VUE_APP_BACKEND_URL + "/images" + image
            }
            return "/img/unavailable.png"
        }
    }
}

Vue.use(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.

Kemudian jangan lupa import file helper.js ini pada `src/main.js`

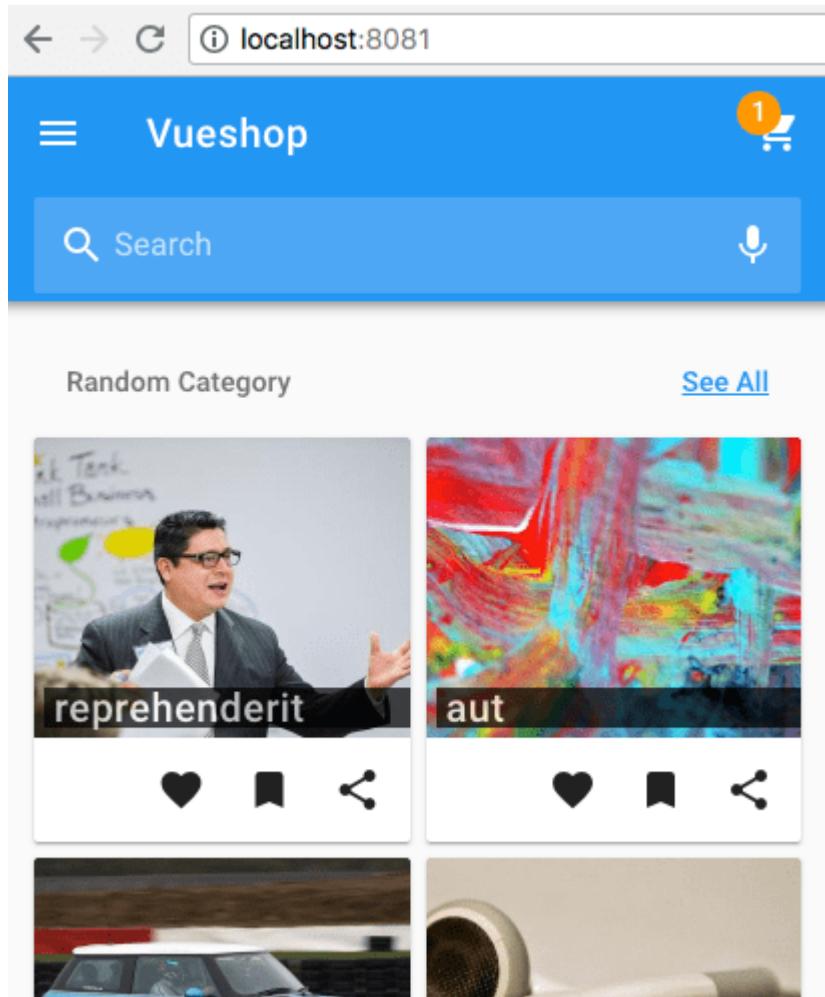
```
import './plugins/helper'
```

Pada CHeader.vue kita juga bisa gunakan properti `appName`.

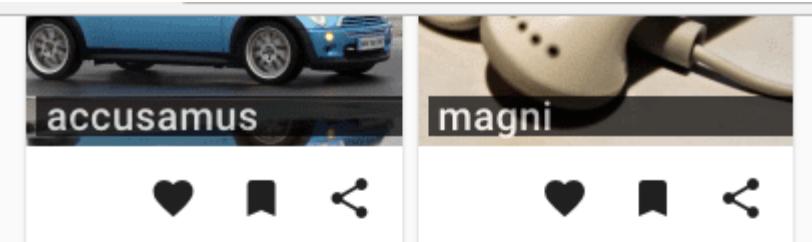
```
<v-toolbar-title class="white--text">{{ appName }}</v-toolbar-title>
```

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.

Berikut ini hasilnya

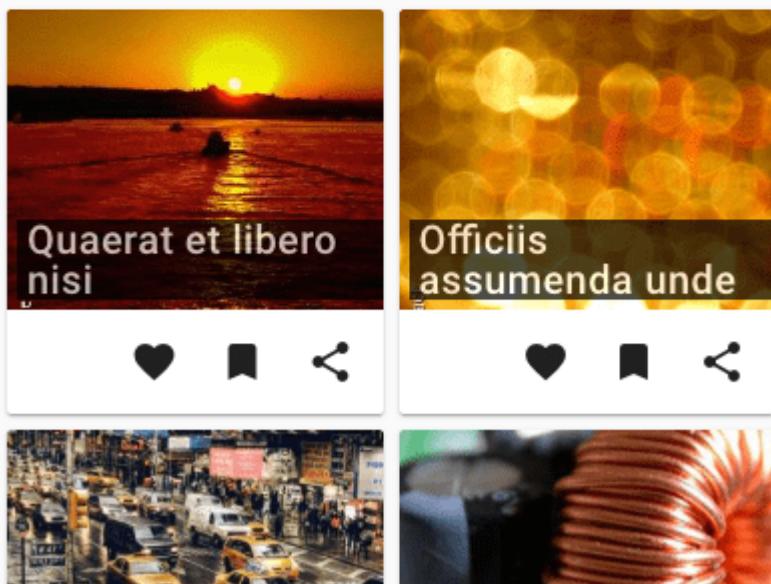


Ketika di-scroll down.



Top Books

[See All](#)



Terdapat 4 jenis link pada halaman Home ini.

1. Link ke route `/categories` pada sub header **Random Category** yang akan mengarahkan ke halaman daftar kategori buku.

```
<v-subheader>
  Random Category
  <v-spacer></v-spacer>
  <router-link to="/categories">See All</router-link>
</v-subheader>
```

2. Link ke route `/books` pada sub header **Top Books** yang akan mengarahkan ke halaman daftar buku.

```
<v-subheader>
  Top Books
  <v-spacer></v-spacer>
  <router-link to="/books">See All</router-link>
</v-subheader>
```

3. Link ke route `/category/slug` yang akan mengarahkan ke halaman detail kategori buku.

```
<v-card :to="'/category/'+ category.slug">
```

4. Link ke route `/book/slug` yang akan mengarahkan ke halaman detail buku

```
<v-card :to="'/book/'+ book.slug">
```

Oleh karena itu selanjutnya kita akan membuat ke empat halaman tersebut.

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 `See All` pada halaman home.

Kode Vue untuk halaman daftar kategori buku ini akan disimpan dalam file `Categories.vue` pada direktori `/src/views`

Endpoint Book

Kembali ke projek Laravel. Kita akan membuat endpoint Book yang sifatnya publik, caranya: tambahkan fungsi `index()` pada controller Category yang sebelumnya telah kita buat.

```
public function index()
{
    $criteria = Category::paginate(6);
    return new CategoryResourceCollection($criteria);
}
```

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`.

```
<?php
use Illuminate\Http\Request;

Route::prefix('v1')->group(function () {
    // public
    // .. route lain

    Route::get('categories/random/{count}', 'CategoryController@random');
    Route::get('categories', 'CategoryController@index'); // ini

    Route::get('books/top/{count}', 'BookController@top');
```

```
// private
Route::middleware(['auth:api'])->group(function () {
    Route::post('logout', 'AuthController@logout');
});
});
```

Mari kita uji coba routing <http://larashop-api.test/v1/categories> menggunakan postman. Maka hasilnya kurang lebih sebagai berikut.

```
{
    "status": "success",
    "message": "categories data",
    "data": [
        {
            "id": 1,
            "name": "quia",
            "slug": "quia",
            "image": "c66e5cd9fcaff2c156e8a115317c9206.jpg",
            "status": "PUBLISH",
            "created_at": "2018-08-28 07:00:40",
            "updated_at": null,
            "deleted_at": null,
            "created_by": null,
            "updated_by": null,
            "deleted_by": null
        },
        {
            "id": 2,
            "name": "reprehenderit",
            ...
        },
        {
            "id": 3,
            "name": "molestias",
            ...
        },
        {
            "id": 4,
            "name": "quidem",
            ...
        },
        {
            "id": 5,
            "name": "accusamus",
            ...
        },
        {
            "id": 6,
            "name": "iste",
            ...
        }
    ]
}
```

```

        }
    ],
    "links": {
        "first": "http://larashop-api.test/v1/categories?page=1",
        "last": "http://larashop-api.test/v1/categories?page=2",
        "prev": null,
        "next": "http://larashop-api.test/v1/categories?page=2"
    },
    "meta": {
        "current_page": 1,
        "from": 1,
        "last_page": 2,
        "path": "http://larashop-api.test/v1/categories",
        "per_page": 6,
        "to": 6,
        "total": 8
    }
}
}

```

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 namun ada tambahan yaitu component untuk pagination **v-pagination**.

Berikut ini kode template **Categories.vue**

```

<template>
  <div>
    <v-container grid-list-md>
      <v-subheader>
        All Category
      </v-subheader>
      <v-layout row wrap>
        <!-- lakukan perulangan pada properti categories -->
        <v-flex v-for="category in categories" xs6 :key="category.id">
          <v-card :to="'/category/' + category.slug">
            <v-card-media
              v-if="category.image"
              :src="getImage('/categories/' + category.image)"
              height="150px"
            >
              <v-container fill-height fluid pa-2>
                <v-layout fill-height>
                  <v-flex xs12 align-end flexbox>
                    <span class="title white--text text-block" v-
text="category.name"></span>
                  </v-flex>
                </v-layout>
              </v-container>
            </v-card-media>
          </v-card>
        </v-flex>
      </v-layout>
    </v-container>
  </div>

```

```
</v-layout>
</v-container>
</v-card-media>

<v-card-actions>
  <v-spacer></v-spacer>
  <v-btn icon>
    <v-icon>favorite</v-icon>
  </v-btn>
  <v-btn icon>
    <v-icon>bookmark</v-icon>
  </v-btn>
  <v-btn icon>
    <v-icon>share</v-icon>
  </v-btn>
</v-card-actions>
</v-card>
</v-flex>
</v-layout>
</v-container>

<template>
  <div class="text-xs-center">
    <!-- kode untuk link paging halaman -->
    <v-pagination
      v-model="page"
      @input="go"
      :length="lengthPage"
      :total-visible="5"
    ></v-pagination>
  </div>
</template>

</div>
</template>
<style scoped>
.text-block {
  position: absolute;
  bottom: 5px;
  left: 5px;
  background-color: black;
  padding-left: 5px;
  padding-right: 5px;
  opacity: 0.7;
  width:100%;
}
</style>
```

Tidak ada yang unik pada kode template di atas alias masih sama strukturnya dengan kode template pada Home, kecuali pada bagian **v-pagination**.

```
<v-pagination
v-model="page"
@input="go"
:length="lengthPage"
:total-visible="5"
></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 (informasinya bisa kita dapat dari web service), serta `total-visible` menujukkan maksimal link paging yang akan ditampilkan.

Berikut ini kode bagian script.

```
<script>
export default {
  data () {
    return {
      categories: [],
      page: 0,
      lengthPage: 0
    }
  },
  methods: {
    go(){
      let url = '/categories'
      if(this.page>0) url = '/categories?page=' +this.page
      this.axios.get(url)
        .then((response) => {
          let response_data = response.data
          let categories = response_data.data
          this.lengthPage = response_data.meta.last_page // jumlah
          halaman di dapat dari meta endpoint categories
          this.categories = categories // daftar category dari endpoint
          categories
        })
        .catch((error) => {
          console.log(error.response)
        })
      },
      created(){
        this.go()
      }
    }
</script>
```

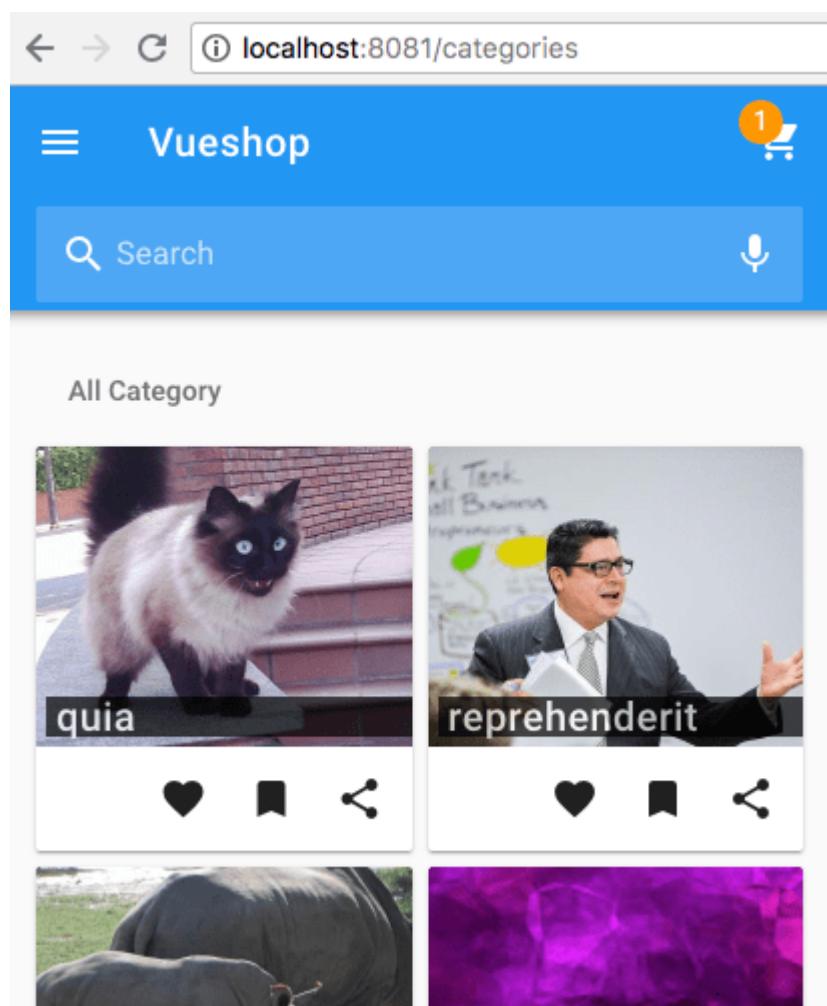
Pada hook `created`, kita gunakan untuk mengakses data endpoint `/categories` melalui methods `go()`.

Mendaftarkan Router Categories

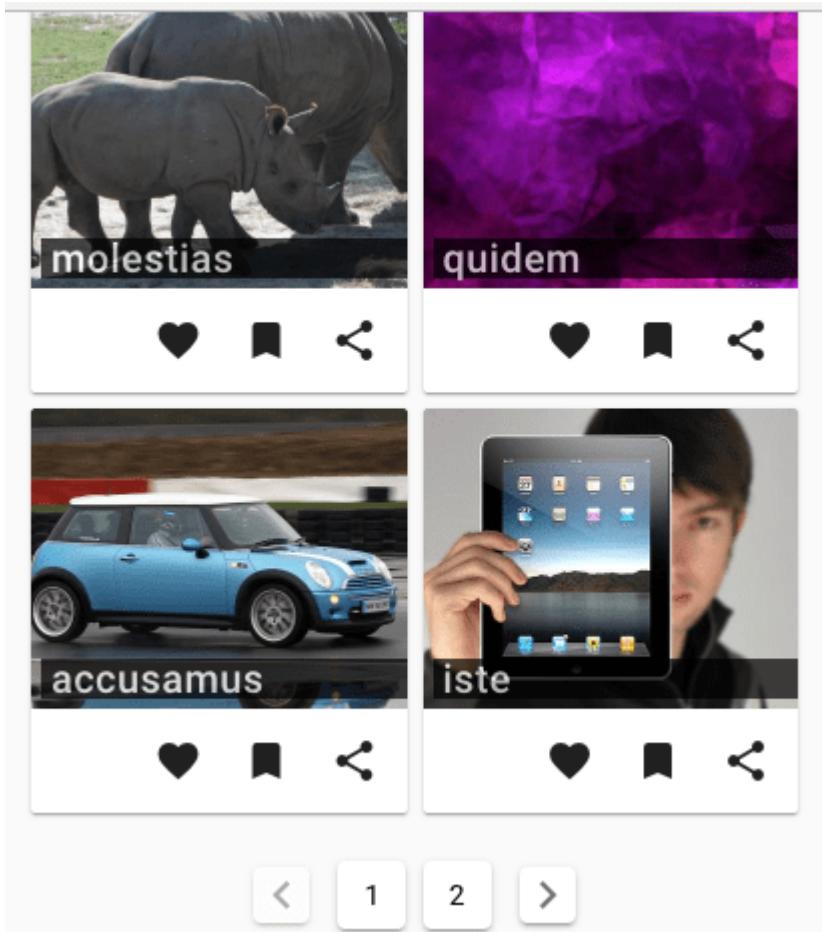
Pada 'src/router.js`, kita tambahkan aturan router untuk component Categories ini. Kita bisa gunakan teknis lazy loading berikut.

```
{  
  path: '/categories',  
  name: 'categories',  
  component: () => import( /* webpackChunkName: "categories" */  
    './views/Categories.vue')  
},
```

Jika sudah maka kita bisa ujicoba melalui browser. Berikut ini hasilnya



Ketika di-scroll down maka kita jumpai paging.



Modifikasi CHeader & CFooter

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. Pada bagian ini kita akan mengimplementasikannya pada component CHeader dan CFooter di mana tampilan pada home dengan selain home akan kita bedakan. Untuk itu kita perlu mengecek apakah halaman saat ini merupakan halaman home atau bukan, caranya simple yaitu mendeteksinya melalui route. Karena route home adalah `/` maka kita bisa gunakan kode ini (`this.$route.path==='/'`). Adapun pengecekan ini bisa kita letakkan pada properti `computed()` yaitu properti yang akan selalu dimonitor perubahannya.

Pada CHeader tambahkan fungsi `isHome()` pada properti `computed`.

```
computed: {
  ...mapGetters({
    sideBar : 'sideBar',
  }),
  isHome () {
    return (this.$route.path==='/')
  },
}
```

Misalnya pada toolbar, kita akan tambahkan tombol back jika halaman bukan home yang menjalankan perintah `this.$router.go(-1)` yaitu kembali ke history page sebelumnya. Tombol ini menggantikan

```
<v-toolbar-side-icon v-if="isHome" @click="setSideBar(!sideBar)"></v-
toolbar-side-icon>
<v-btn v-if="!isHome" icon @click="$router.go(-1)">
    <v-icon>arrow_back</v-icon>
</v-btn>
```

Demikian juga dengan kolom pencarian hanya akan ditampilkan pada halaman home saja. Adapun kode lengkap CHeader.vue menjadi sebagai berikut.

```
<template>
    <v-toolbar dark color="primary">
        <v-toolbar-side-icon v-if="isHome" @click="setSideBar(!sideBar)"></v-
toolbar-side-icon>
        <v-btn v-if="!isHome" icon @click="this.$router.go(-1)">
            <v-icon>arrow_back</v-icon>
        </v-btn>
        <v-toolbar-title class="white--text">{{ appName }}</v-toolbar-title>
        <v-spacer></v-spacer>
        <v-btn icon>
            <v-badge left overlap color="orange">
                <span slot="badge">1</span>
                <v-icon>shopping_cart</v-icon>
            </v-badge>
        </v-btn>
        <v-text-field
            v-if="isHome"
            slot="extension"
            hide-details
            append-icon="mic"
            flat
            label="Search"
            prepend-inner-icon="search"
            solo-inverted
        ></v-text-field>
    </v-toolbar>
</template>
<script>
import { mapGetters, mapActions } from 'vuex'
export default {
    name: 'c-header',
    methods: {
        ...mapActions({
            setSideBar : 'setSideBar',
        }),
    },
    computed: {
        ...mapGetters({
            sideBar : 'sideBar',
        })
    }
}</script>
```

```

    },
    isHome () {
      return (this.$route.path==='/')
    },
}
</script>

```

Demikian juga pada component CFooter bisa kita mainkan juga. Misalnya link sosmed dan deskripsi aplikasi hanya akan ditampilkan pada halaman home saja. Tentu saja penulis yakin kamu bisa melakukannya.

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 [See All](#) 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.

```

public function index()
{
  $criteria = Book::paginate(6);
  return new BookResourceCollection($criteria);
}

```

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`.

```

<?php
use Illuminate\Http\Request;

Route::prefix('v1')->group(function () {
  // public
  Route::post('login', 'AuthController@login');
  Route::post('register', 'AuthController@register');

  Route::get('categories/random/{count}', 'CategoryController@random');
  Route::get('categories', 'CategoryController@index');
});

```

```

Route::get('books/top/{count}', 'BookController@top');
Route::get('books', 'BookController@index'); // <== ini

// private
Route::middleware(['auth:api'])->group(function () {
    Route::post('logout', 'AuthController@logout');
});

```

Mari kita uji coba routing <http://larashop-api.test/v1/books> menggunakan postman. Maka hasilnya kurang lebih sebagai berikut.

```
{
    "status": "success",
    "message": "books data",
    "data": [
        {
            "id": 1,
            "title": "Quaerat et libero nisi",
            "slug": "quaerat-et-libero-nisi",
            "description": "Et officia vel unde qui enim voluptate quidem. Voluptatem nemo corporis et repudiandae qui voluptas ullam. Optio sit accusamus libero nostrum maiores placeat labore suscipit. Ducimus velit fugit tempora ipsa temporibus.",
            "author": "Jessie Barton",
            "publisher": "Lesch PLC",
            "cover": "86d8dcf0e10fa269f91eb320f8edd3c4.jpg",
            "price": 350000,
            "weight": 0.5,
            "views": 0,
            "stock": 0,
            "status": "PUBLISH",
            "created_at": "2018-08-28 06:59:30",
            "updated_at": "2018-09-01 22:46:51",
            "deleted_at": null,
            "created_by": null,
            "updated_by": null,
            "deleted_by": null
        },
        {
            "id": 2,
            ...
        },
        {
            "id": 3,
            ...
        },
        {
            "id": 4,
            ...
        },
        ...
    ],
}
```

```

{
    "id": 5,
    ...
},
{
    "id": 6,
    ...
}
],
"links": {
    "first": "http://larashop-api.test/v1/books?page=1",
    "last": "http://larashop-api.test/v1/books?page=5",
    "prev": null,
    "next": "http://larashop-api.test/v1/books?page=2"
},
"meta": {
    "current_page": 1,
    "from": 1,
    "last_page": 5,
    "path": "http://larashop-api.test/v1/books",
    "per_page": 6,
    "to": 6,
    "total": 25
}
}
}

```

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

```

<template>
<div>
    <v-container grid-list-md>
        <v-subheader>
            All Books
        </v-subheader>
        <v-layout row wrap>
            <v-flex v-for="(book, index) in books" xs6 :key="index">
                <v-card :to="/book/'+ book.slug">
                    <v-card-media
                        :src="getImage('/books/'+book.cover)"
                        height="150px"
                    >
                        <v-container fill-height fluid pa-2>
                            <v-layout fill-height>

```

```
<v-flex xs12 align-end flexbox>
    <span class="title white--text text-block" v-
text="book.title"></span>
    </v-flex>
    </v-layout>
    </v-container>
</v-card-media>

<v-card-actions>
    <v-spacer></v-spacer>
    <v-btn icon>
        <v-icon>favorite</v-icon>
    </v-btn>
    <v-btn icon>
        <v-icon>bookmark</v-icon>
    </v-btn>
    <v-btn icon>
        <v-icon>share</v-icon>
    </v-btn>
</v-card-actions>
</v-card>
</v-flex>
</v-layout>
</v-container>

<template>
    <div class="text-xs-center">
        <v-pagination
            v-model="page"
            @input="go"
            :length="lengthPage"
            :total-visible="5"
        ></v-pagination>
    </div>
</template>
</div>
</template>
<style scoped>
.text-block {
    position: absolute;
    bottom: 5px;
    left: 5px;
    background-color: black;
    padding-left: 5px;
    padding-right: 5px;
    opacity: 0.7;
    width: 100%;
}
</style>
```

Berikut ini kode bagian script.

```
<script>
export default {
  data () {
    return {
      books: [],
      page: 0,
      lengthPage: 0
    }
  },
  methods: {
    go(){
      let url = '/books'
      if(this.page>0) url = '/books?page=' +this.page
      this.axios.get(url)
        .then((response) => {
          let response_data = response.data
          let books = response_data.data
          this.lengthPage = response_data.meta.last_page
          this.books = books
        })
        .catch((error) => {
          console.log(error.response)
        })
    },
    created(){
      this.go()
    }
  }
}
</script>
```

Secara umum kode di atas sama dengan kode pada halaman Categories, hanya berbeda endpointnya saja.

Mendaftarkan Router Books

Pada 'src/router.js', kita tambahkan aturan router untuk component Books ini.

```
{
  path: '/books',
  name: 'books',
  component: () => import( /* webpackChunkName: "books" */ 
  './views/Books.vue')
},
```

Jika sudah maka kita bisa ujicoba melalui browser. Berikut ini hasilnya

← → ⌂ ⓘ localhost:8081/books

Vueshop

1

All Category

quia

Like Share

reprehenderit

Like Share

molestias

quidem

Like Share

accusamus

iste

Like Share

< 1 2 >

Halaman Detail Kategori Buku

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()**.

```
public function books(){
    return $this->belongsToMany("App\Book");
}
```

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:

```
public function books(){
    return $this->belongsToMany('App\Book', 'book_category', 'category_id',
        'book_id');
}
```

Jika sudah, maka kita perlu buat dulu resource untuk model Category.

```
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).

```
public function toArray($request)
{
    $parent = parent::toArray($request);
    // ambil data books yang berelasi dengan category menggunakan pivot
    $data['books'] = $this->books()->paginate(6);
    $data = array_merge($parent, $data);
    return [
        'status'      => 'success',
        'message'     => 'category data',
        'data'         => $data
    ];
}
```

Kemudian tambahkan fungsi **slug()** pada controller Category yang sebelumnya telah kita buat.

```
//...
use App\Http\Resources\Category as CategoryResource;

//...

public function slug($slug)
{
    $criteria = Category::where('slug', $slug)->first();
    return new CategoryResource($criteria);
}
```

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**.

```
<?php
use Illuminate\Http\Request;

Route::prefix('v1')->group(function () {
    // public
    Route::post('login', 'AuthController@login');
    Route::post('register', 'AuthController@register');

    Route::get('categories/random/{count}', 'CategoryController@random');
    Route::get('categories', 'CategoryController@index');
    Route::get('categories/slug/{slug}', 'CategoryController@slug'); // <=
ini
```

```
Route::get('books/top/{count}', 'BookController@top');
Route::get('books', 'BookController@index');

// private
Route::middleware(['auth:api'])->group(function () {
    Route::post('logout', 'AuthController@logout');
});
});
```

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.

```
{
    "status": "success",
    "message": "category data",
    "data": {
        "id": 1,
        "name": "quia",
        "slug": "quia",
        "image": "c66e5cd9fcaff2c156e8a115317c9206.jpg",
        "status": "PUBLISH",
        "created_at": "2018-08-28 07:00:40",
        "updated_at": null,
        "deleted_at": null,
        "created_by": null,
        "updated_by": null,
        "deleted_by": null,
        "books": {
            "current_page": 1,
            "data": [
                {
                    "id": 1,
                    "title": "Quaerat et libero nisi",
                    "slug": "quaerat-et-libero-nisi",
                    "description": "Et officia vel unde qui enim voluptate quidem. Voluptatem nemo corporis et repudiandae qui voluptas ullam. Optio sit accusamus libero nostrum maiores placeat labore suscipit. Ducimus velit fugit tempora ipsa temporibus.",
                    "author": "Jessie Barton",
                    "publisher": "Lesch PLC",
                    "cover": "86d8dcf0e10fa269f91eb320f8edd3c4.jpg",
                    "price": 350000,
                    "weight": 0.5,
                    "views": 0,
                    "stock": 0,
                    "status": "PUBLISH",
                    "created_at": "2018-08-28 06:59:30",
                    "updated_at": "2018-09-01 22:46:51",
                    "deleted_at": null,
                    "created_by": null,
                    "updated_by": null,
                    "deleted_by": null
                }
            ]
        }
    }
}
```

```
        "updated_by": null,
        "deleted_by": null,
        "pivot": {
            "category_id": 1,
            "book_id": 1
        }
    },
{
    "id": 2,
    "title": "Officiis assumenda unde",
    ...
},
],
"first_page_url": "http://larashop-
api.test/v1/categories/slug/quia?page=1",
"from": 1,
"last_page": 1,
"last_page_url": "http://larashop-
api.test/v1/categories/slug/quia?page=1",
"next_page_url": null,
"path": "http://larashop-api.test/v1/categories/slug/quia",
"per_page": 6,
"prev_page_url": null,
"to": 2,
"total": 2
}
}
}
```

Template & Script

Kembali ke projek Vue, buka kode template `views/Category.vue` dan update menjadi sebagai berikut.

```
<template>
<div>
    <v-container grid-list-md>
        <v-subheader> {{ category.name }} </v-subheader>
        <v-card-media v-if="category.image" :src="getImage('/categories/' +
category.image)" height="150px"></v-card-media>
        <v-subheader> Books by "{{ category.name }}" category </v-subheader>
        <v-layout row wrap>
            <!-- data books yang berelasi akan ditampilkan menggunakan looping
-->
            <v-flex v-for="book in books" xs6 :key="book.id">
                <v-card :to="/book/'+ book.slug">
                    <v-card-media v-if="book.cover"
                        :src="getImage('/books/' +book.cover)"
                        height="150px"
                    >
                        <v-container fill-height fluid pa-2>
                            <v-layout fill-height>
```

```
<v-flex xs12 align-end flexbox>
    <span class="title white--text text-block" v-
text="book.title"></span>
    </v-flex>
    </v-layout>
    </v-container>
</v-card-media>

<v-card-actions>
    <v-spacer></v-spacer>
    <v-btn icon>
        <v-icon>favorite</v-icon>
    </v-btn>
    <v-btn icon>
        <v-icon>bookmark</v-icon>
    </v-btn>
    <v-btn icon>
        <v-icon>share</v-icon>
    </v-btn>
</v-card-actions>
</v-card>
</v-flex>
</v-layout>

<template>
    <div class="text-xs-center">
        <v-pagination
            v-model="page"
            @input="go"
            :length="lengthPage"
            :total-visible="5"
        ></v-pagination>
    </div>
    </template>
</v-container>
</div>
</template>

<style scoped>
.text-block {
    position: absolute;
    bottom: 5px;
    left: 5px;
    background-color: black;
    padding-left: 5px;
    padding-right: 5px;
    opacity: 0.7;
    width: 100%;
}
</style>
```

Berikut ini kode bagian script.

```

<script>
export default {
  data () {
    return {
      category: {}, // penampung satu objek category
      books: [], // penampung daftar buku pada category tersebut
      page: 0,
      lengthPage: 0
    }
  },
  methods: {
    go(){
      let slug = this.$route.params.slug
      let url = '/categories/slug/'+slug
      if(this.page>0) url = url + '?page=' + this.page
      url = encodeURI(url)
      this.axios.get(url)
        .then((response) => {
          let response_data = response.data
          let category = response_data.data
          this.category = category
          this.books = category.books.data
          this.lengthPage = category.books.last_page
        })
        .catch((error) => {
          console.log(error.response)
        })
    }
  },
  created(){
    this.go()
  }
}
</script>

```

Mendaftarkan Router Category

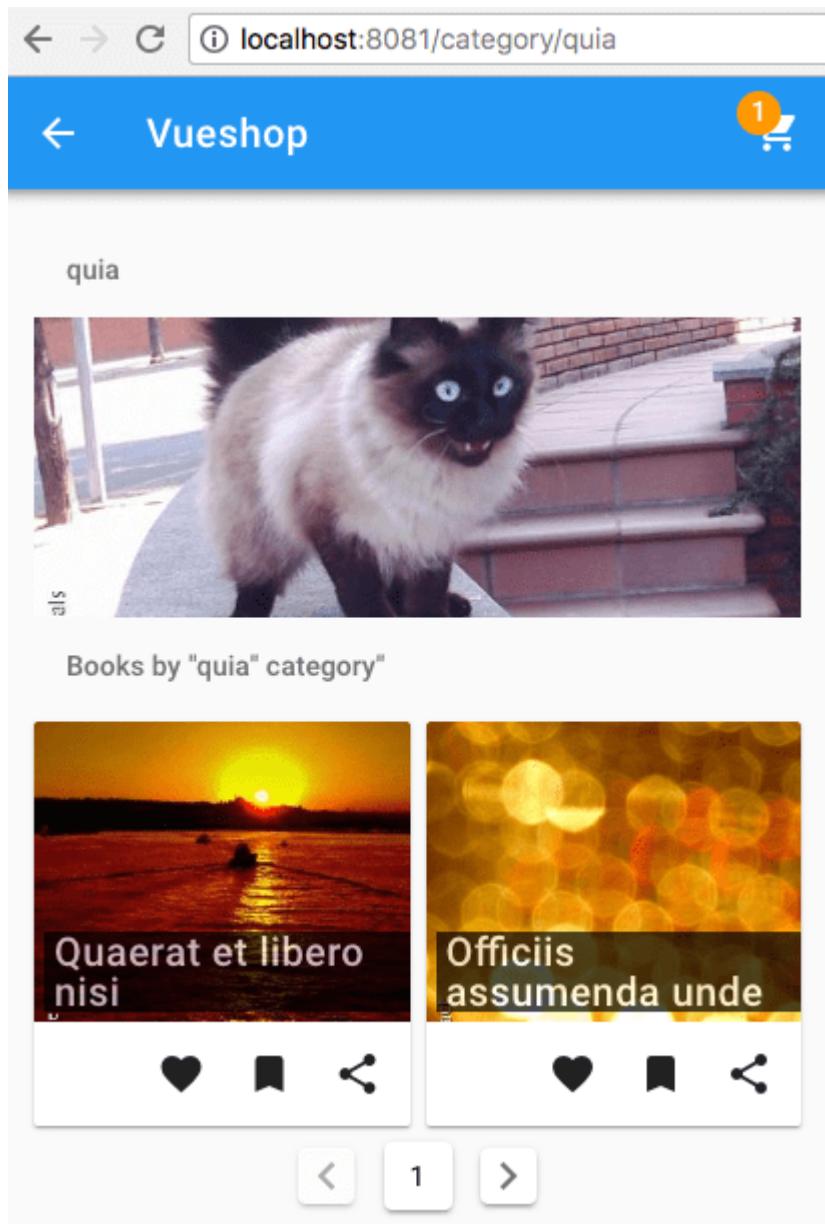
Pada 'src/router.js', kita tambahkan aturan router untuk component Category ini. Kita tambahkan parameter slug pada path.

```
{
  path: '/category/:slug',
  name: 'category',
  component: () => import( /* webpackChunkName: "category" */
  './views/Category.vue')
},
```

Jika sudah maka kita bisa ujicoba melalui browser, misalnya URL

<http://localhost:8081/category/quia>.

Berikut ini hasilnya



Halaman Detail Buku

Halaman ini menampilkan daftar buku dalam bentuk list, di mana pada setiap itemnya ditampilkan 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 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()`.

```
public function categories()
{
    return $this->belongsToMany('App\Category');
}
```

Jika sudah, maka kita perlu buat dulu resource untuk model Book.

```
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).

```
public function toArray($request)
{
    $parent = parent::toArray($request);
    $data['categories'] = $this->categories;
    $data = array_merge($parent, $data);
    return [
        'status' => 'success',
        'message' => 'book data',
        'data' => $data,
    ];
}
```

Kemudian tambahkan fungsi `slug()` pada controller Book yang sebelumnya telah kita buat.

```
//...
use App\Http\Resources\Book as BookResource;

//...

public function slug($slug)
{
    $criteria = Book::where('slug', $slug)->first();
    return new BookResource($criteria);
}
```

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`.

```
<?php
use Illuminate\Http\Request;

Route::prefix('v1')->group(function () {
    // public
    Route::post('login', 'AuthController@login');
    Route::post('register', 'AuthController@register');

    Route::get('categories/random/{count}', 'CategoryController@random');
    Route::get('categories', 'CategoryController@index');
    Route::get('categories/slug/{slug}', 'CategoryController@slug');

    Route::get('books/top/{count}', 'BookController@top');
    Route::get('books', 'BookController@index');
    Route::get('books/slug/{slug}', 'BookController@slug'); // <== ini

    // private
    Route::middleware(['auth:api'])->group(function () {
        Route::post('logout', 'AuthController@logout');
    });
});
```

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.

```
{
    "status": "success",
    "message": "book data",
    "data": {
        "id": 1,
        "title": "Quaerat et libero nisi",
        "slug": "quaerat-et-libero-nisi",
        "description": "Et officia vel unde qui enim voluptate quidem. Voluptatem nemo corporis et repudiandae qui voluptas ullam. Optio sit accusamus libero nostrum maiores placeat labore suscipit. Ducimus velit fugit tempora ipsa temporibus.",
        "author": "Jessie Barton",
        "publisher": "Lesch PLC",
        "cover": "86d8dcf0e10fa269f91eb320f8edd3c4.jpg",
        "price": 350000,
        "weight": 0.5,
        "views": 0,
        "stock": 0,
        "status": "PUBLISH",
        "created_at": "2018-08-28 06:59:30",
        "updated_at": "2018-09-01 22:46:51",
        "deleted_at": null,
```

```
"created_by": null,
"updated_by": null,
"deleted_by": null,
"categories": [
    {
        "id": 1,
        "name": "quia",
        "slug": "quia",
        "image": "c66e5cd9fcaff2c156e8a115317c9206.jpg",
        "status": "PUBLISH",
        "created_at": "2018-08-28 07:00:40",
        "updated_at": null,
        "deleted_at": null,
        "created_by": null,
        "updated_by": null,
        "deleted_by": null,
        "pivot": {
            "book_id": 1,
            "category_id": 1
        }
    },
    {
        "id": 2,
        "name": "rehenderit",
        ...
    },
    {
        "id": 3,
        "name": "molestias",
        ...
    }
]
```

Template & Script

Kembali ke projek Vue, berikut ini kode template `views/Book.vue`

```
<template>
<div class="about">
<v-container>

    <v-subheader class="title"> {{ book.title }} </v-subheader>
    <v-card-media v-if="book.cover" :src="getImage('/books/' + book.cover)" height="200px"></v-card-media>

    <v-subheader> Information </v-subheader>
    <table class="v-table">
        <tbody>
            <tr><th class="text-xs-left">Author</th><td>{{ book.author }}</td>
```

```

</tr>
    <tr><th class="text-xs-left">Publisher</th><td>{{ book.publisher }}</td></tr>
    <tr><th class="text-xs-left">Price</th><td v-if="book.price">Rp. {{ book.price.toLocaleString('id-ID') }}</td></tr>
    <tr><th class="text-xs-left">Weight</th><td>{{ book.weight }} kg</td>
</tr>
    <tr><th class="text-xs-left">Stock</th><td>{{ book.stock }}</td></tr>
    <tr><th class="text-xs-left">Categories</th>
        <td>
            <template v-for="category in book.categories" v-key="category.id">
                {{ category.name }},
            </template>
        </td>
    </tr>
</tbody>
</table>

<v-subheader> Description </v-subheader>
<p class="body-2"> {{ book.description }} </p>

<div style="position:relative">
<v-btn block color="success" @click="buy" :disabled="book.stock==0">
    <v-icon>save_alt</v-icon> &ampnbsp
    BUY
</v-btn>
</div>
</v-container>
</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.

Berikut ini kode bagian script.

```

<script>
export default {
  data () {
    return {
      book: {},
    }
  },
  methods: {
    buy(){
      alert('Buy')
    }
  },
  created(){
    let slug = this.$route.params.slug
  }
}

```

```
this.axios.get('/books/slug/'+slug)
.then((response) => {
  let book = response.data.data
  this.book = book
})
.catch((error) => {
  let responses = error.response
  console.log(responses)
})
}
}

</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.

Mendaftarkan Route Book

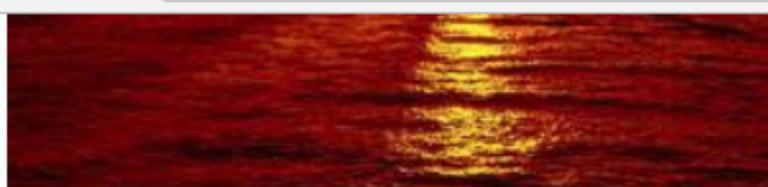
Pada 'src/router.js', kita tambahkan aturan router untuk component Book ini. Kita tambahkan parameter slug pada path.

```
{
  path: '/book/:slug',
  name: 'book',
  component: () => import( /* webpackChunkName: "book" */ 
  './views/Book.vue')
},
```

Jika sudah maka kita bisa ujicoba melalui browser, misalnya slugnya `quaerat-et-libero-nisi`, maka URL detail book yang bisa diakses menjadi <http://localhost:8081/book/quaerat-et-libero-nisi>.

Hasilnya sebagai berikut.

← → ⌂ ⓘ localhost:8081/book/quaerat-et-libero-nisi



Information

Author Jessie Barton

Publisher Lesch PLC

Price Rp. 350.000

Weight 0.5 kg

Stock 1

Categories quia, reprehenderit, molestias,

Description

Et officia vel unde qui enim voluptate quidem. Voluptatem nemo corporis et repudiandae qui voluptas ullam. Optio sit accusamus libero nostrum maiores placeat labore suscipit. Ducimus velit fugit tempora ipsa temporibus.

BUY

State Cart

Sebelum kita membuat tombol **Buy** berfungsi dengan baik yaitu untuk memasukkan data buku ke dalam keranjang belanja, maka kita perlu membuat state keranjang belanja dulu. State ini akan kita beri nama **cart**, bentuknya list atau array dari objek, sebagai berikut

```
[  
  { id: 1, quantity: 2, title: 'abc', ... },  
  { id: 2, quantity: 1, title: 'xyz', ... },  
  //....  
]
```

Setiap itemnya merupakan objek buku dengan tambahan key **quantity** untuk menyimpan jumlah item yang dibeli pada produk buku tersebut.

Supaya lebih rapi maka state ini akan kita jadikan module terpisah pada **src/stores/cart.js**.

```

export default {
  namespaced: true,
  state: {
    carts: [],
  },
  mutations: {
    insert: (state, payload) => {
      state.carts.push({
        id: payload.id,
        title: payload.title,
        cover: payload.cover,
        price: payload.price,
        weight: payload.weight,
        quantity: 1
      })
    },
    update: (state, payload) => {
      let idx = state.carts.indexOf(payload);
      state.carts.splice(idx, 1, {
        id: payload.id,
        title: payload.title,
        cover: payload.cover,
        price: payload.price,
        weight: payload.weight,
        quantity: ++payload.quantity
      });
    },
  },
  actions: {
    add: ({state, commit}, payload) => {
      let cartItem = state.carts.find(item => item.id === payload.id)
      if(!cartItem){
        commit('insert', payload)
      }
      else{
        commit('update', cartItem)
      }
    },
  },
  getters: {
    carts: state => state.carts,
  }
}

```

Terdapat satu actions pada state cart ini yaitu add() yang akan melakukan pengecekan apakah data buku yang dimasukkan ke dalam keranjang belanja atau cart sudah ada atau belum.

```
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 maka akan mengcommit mutation insert. Mutation insert akan menjalankan fungsi push (menambahkan item pada array)

Lalu pada `src/store.js` tambahkan state `cart.js` sebagai module

```
import Vue from 'vue'
import Vuex from 'vuex'
import cart from './stores/cart'

Vue.use(Vuex)

export default new Vuex.Store({
  state: {
    sideBar: false,
  },
  mutations: {
    setSideBar: (state, value) => {
      state.sideBar = value
    },
  },
  actions: {
    setSideBar: ({commit}, value) => {
      commit('setSideBar', value)
    },
  },
  getters: {
    sideBar: state => state.sideBar,
  },
  modules: {
    cart, // <= tambahkan ini
  }
})
```

Update methods `buy()` pada `Book.vue` menjadi sebagai berikut.

```
<script>
import { mapGetters, mapActions } from 'vuex'
export default {
  data () {
    return {
      book: {}
    }
  },
  methods: {
    ...mapActions({
      addCart : 'cart/add',
    }),
    buy(){
      this.addCart(this.book)
    }
  },
}
```

```
created(){
    // ...
}
</script>
```

Mari kita coba mengklik tombol **Buy**. Pada contoh ini, tombol Buy telah diklik dua kali.

Publisher	Lesch PLC
Price	Rp. 350.000
Weight	0.5 kg
Stock	1
Categories	quia, reprehenderit, molestias, ...
Description	Et officia vel unde qui enim voluptate quidem. Voluptatem nemo corporis et repudiandae qui voluptas ullam. Optio sit accusamus libero nostrum maiores placeat labore suscipit. Ducimus velit fugit tempora ipsa temporibus.
 BUY	

cart/insert 16:10:26
cart/update 16:10:35

```

    ▾ auth: object
      ▶ user: Object (empty)
    ▾ cart: Object
      ▾ carts: Array[1]
        ▾ 0: Object
          cover: "86d8dcf0e10fa269f91eb320f8edd3c4.jpg"
          id: 1
          price: 35000
          quantity: 2
          title: "Quaerat et libero nisi"
          weight: 0.5
    
```

Perhatikan pada console Vue. Terdapat state carts bertipe array dan pada itemnya terdapat key quantity dengan jumlah 2 (karena diklik dua kali).

Component Alert

Ada yang kurang, ketika user mengklik tombol Buy. Memang sih data buku sudah berhasil tersimpan ke dalam keranjang belanja, namun dari perpektif user tidak ada informasi yang menunjukkan keberhasilan itu. Tidak mungkin kan user melihat panel Vue di console 😊.

Karena itu kita akan membuat semacam alert yang akan kita tampilkan salah satunya untuk menginfokan bahwa buku telah berhasil dimasukkan ke dalam keranjang belanja.

Pada vuety ada component alert yang bagus yaitu v-snackbar. Kode dasar dari component ini sebagai berikut.

```
<v-snackbar v-model="alert"
            color="primary" top>
  Ini alert Snackbar
  <v-btn dark flat @click="alert=false">
    Close
  </v-btn>
</v-snackbar>
```

```
</v-btn>
</v-snackbar>
```

Kode di atas akan menampilkan alert berwarna primary dan overly pada bagian atas aplikasi, serta terdapat tombol close. Nah karena alert ini pastinya akan digunakan dibanyak tempat di aplikasi maka kita perlu membuatnya menjadi global.

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 alert. Karenanya untuk memudahkan kita memanipulasi ketiga variabel itu dari component manapun pada aplikasi, kita bisa membuat state tersendiri untuk alert ini.

Berikut ini [src/stores/alert.js](#) untuk memanipulasi 3 variabel/state yaitu status, text, dan type (color).

```
export default {
  namespaced: true,
  state: {
    status: false,
    text: '',
    type: 'success', // warning, error
  },
  mutations: {
    set: (state, payload) => {
      state.status = payload.status
      state.text = payload.text
      state.type = payload.type
    },
  },
  actions: {
    set: ({commit}, payload) => {
      commit('set', payload)
    },
  },
  getters: {
    status: state => state.status,
    text: state => state.text,
    type: state => state.type,
  }
}
```

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](#).

```
...
import cart from './stores/cart'
import alert from './stores/alert' // <== tambahkan ini
```

```
Vue.use(Vuex)
```

```
export default new Vuex.Store({
  ...
  modules: {
    cart,
    alert // <== tambahkan ini
  }
})
```

Kemudian kita buat component alert pada `src/components/CAlert.vue` yang fungsi mewrap component snackbar vuety dan memetakan state alert. 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.

```
<template>
  <v-snackbar v-model="alert" :color="type" top>
    {{ text }}
    <v-btn dark flat @click="close">
      Close
    </v-btn>
  </v-snackbar>
</template>
<script>
import { mapGetters, mapActions } from 'vuex'
export default {
  name: 'c-alert',
  computed: {
    ...mapGetters({
      status : 'alert/status',
      text   : 'alert/text',
      type   : 'alert/type',
    }),
    alert: {
      get () {
        return this.status
      },
      set (value) {
        this.setAlert({
          status : value,
          text   : 'test',
          type   : 'error',
        })
      }
    },
    methods: {
      ...mapActions({
        setAlert : 'alert/set',
      }),
      close(){
        this.alert = false
      }
    }
  }
</script>
```

```
this.setAlert({
    status : false
})
}
}
}
</script>
```

Selanjutnya kita daftarkan component CAlert ini pada file `src/App.vue`.

```
export default {
  name: 'App',
  components: {
    CHeader, CSideBar, CFooter,
    CAlert: () => import('@/components/CAlert.vue')
  ...
}
```

Serta untuk template-nya, tambahkan element `<c-alert />`.

```
<c-side-bar />
<c-footer />

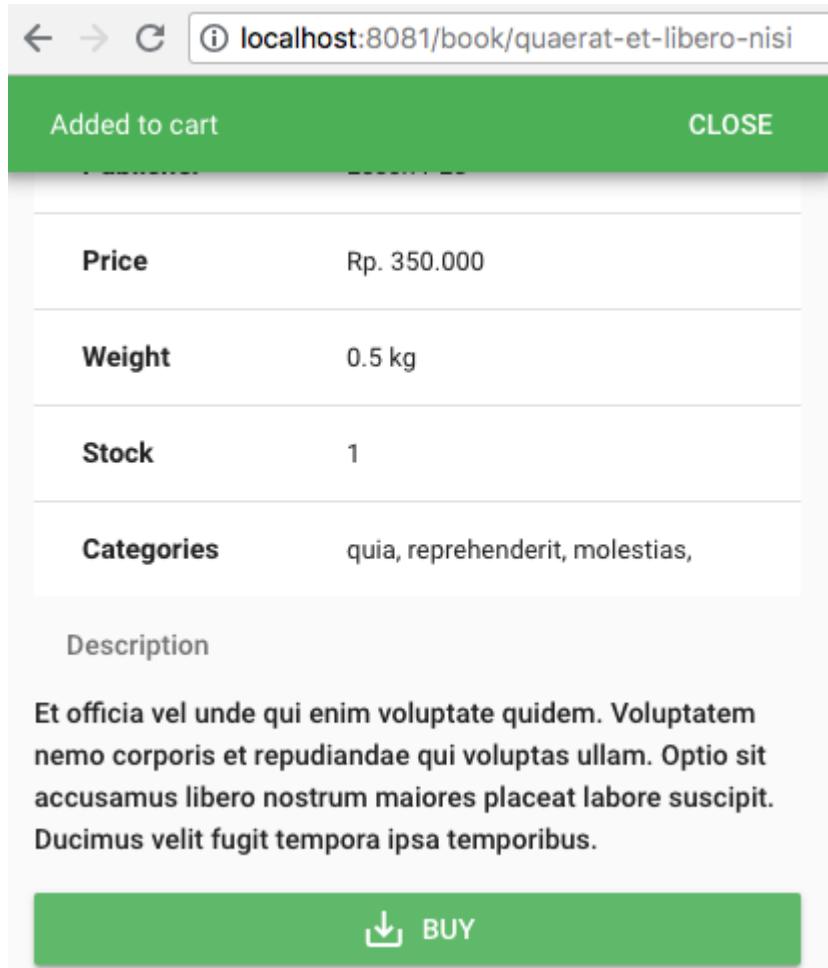
<c-alert />
```

Lalu bagaimana menampilkan alert ini?

Mari kembali ke `src/views/Book.Vue`. Dengan menggunakan `mapActions`, petakan action `setAlert` pada methods. Kemudian pada method `buy()` kita bisa gunakan fungsi `setAlert()` untuk mengeset nilai alert dan menampilkannya.

```
...
methods: {
  ...mapActions({
    addCart : 'cart/add',
    setAlert : 'alert/set',
  }),
  buy(){
    this.addCart(this.book)
    this.setAlert({
      status : true,
      text : 'Added to cart',
      type : 'success',
    })
  }
},
```

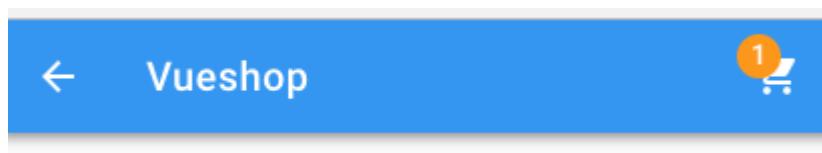
Jadi jika tombol Buy diklik maka selain akan menambahkan buku ke dalam keranjang belanja, juga akan menampilkan alert sebagai berikut.



Silakan gunakan alert ini pada component lainnya untuk menampilkan pesan informasi kepada user.

Indicator Cart

Pada saat ini, indikator keranjang belanja pada header yang menunjukkan jumlah jenis barang yang berada di dalamnya masih manual alias hardcoded, tampak pada gambar berikut.



Pada file `src/components/CHeader.vue`, teks pada badge masih dituliskan secara manual.

```
<v-btn icon>
  <v-badge left overlap color="orange">
    <span slot="badge">1</span>
    <v-icon>shopping_cart</v-icon>
  </v-badge>
</v-btn>
```

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.

```
...
getters: {
    carts : state => state.carts,
    count : (state) => {
        return state.carts.length
    },
}
}
```

Getters count ini mengembalikan panjang array dari state carts. Jika sudah maka getters ini bisa kita manfaatkan untuk memanipulasi teks badge pada component CHeader. Petaikan getter cart count tersebut pada component `src/components/CHeader.vue` bagian script computed.

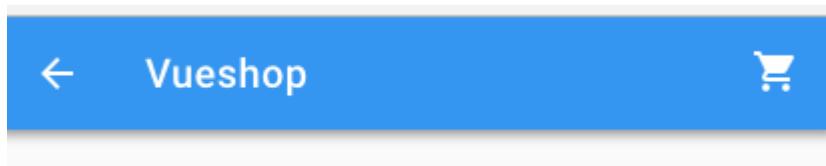
```
computed: {
    ...mapGetters({
        sideBar : 'sideBar',
        countCart : 'cart/count' // <= tambahkan ini
    }),
    ...
}
}
```

Lalu pada template CHeader, gunakan computed countCart ini untuk menggantikan hardcode teks dari badge.

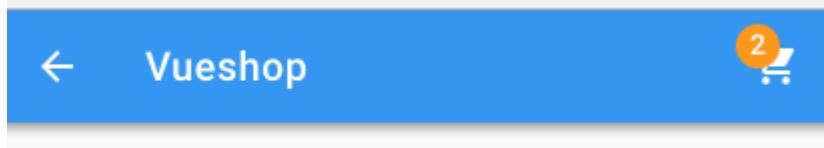
```
<v-btn icon>
<v-badge left overlap color="orange">
    <span slot="badge" v-if="countCart>0"> {{ countCart }} </span>
    <v-icon>shopping_cart</v-icon>
</v-badge>
</v-btn>
```

Indikator hanya akan ditampilkan jika keranjang belanja ada isinya alias lebih besar dari nol. Mari kita lihat hasilnya.

Saat keranjang belanja kosong.



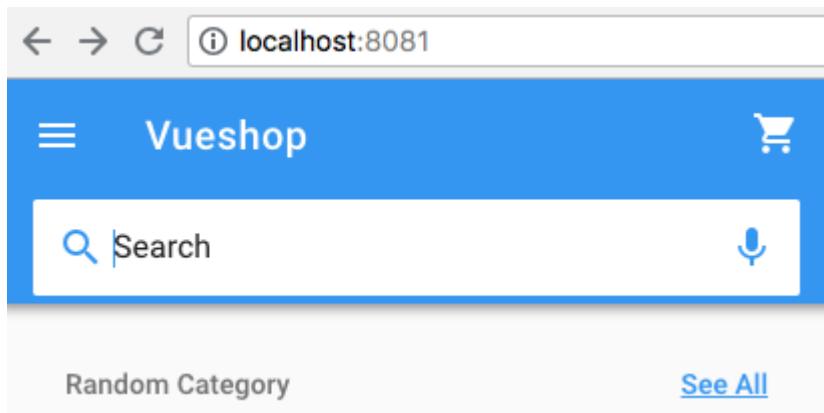
Saat keranjang belanja ada isinya.



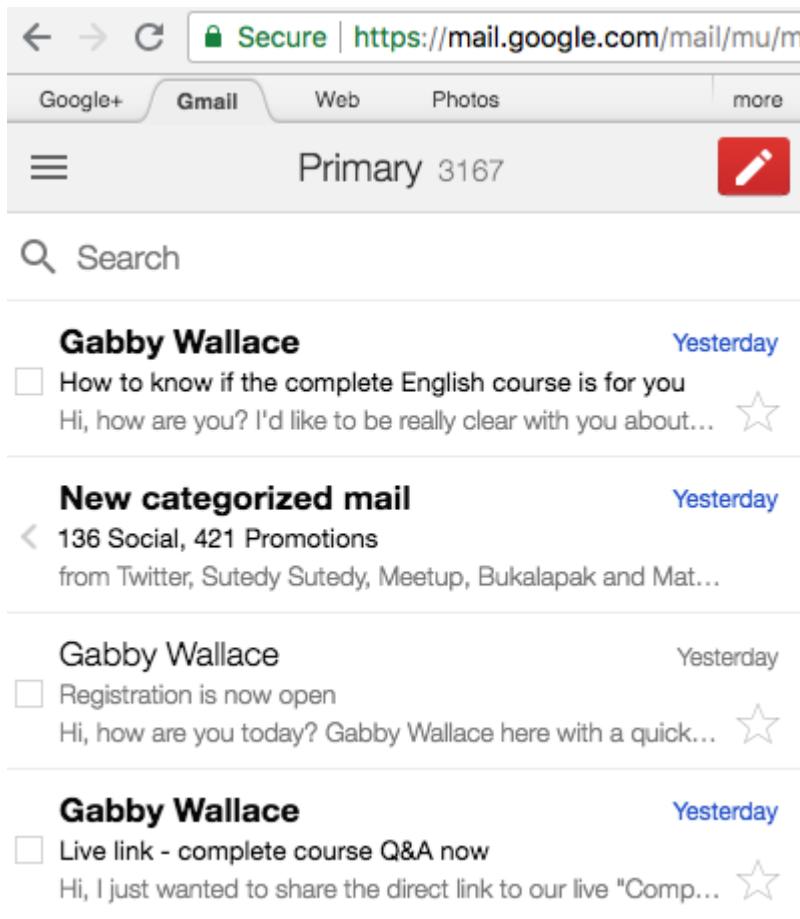
Halaman Pencarian Buku

Halaman ini menampilkan kolom pencarian buku, di mana hasil pencarian akan ditampilkan dalam bentuk list yang setiap itemnya menampilkan cover buku dan judulnya.

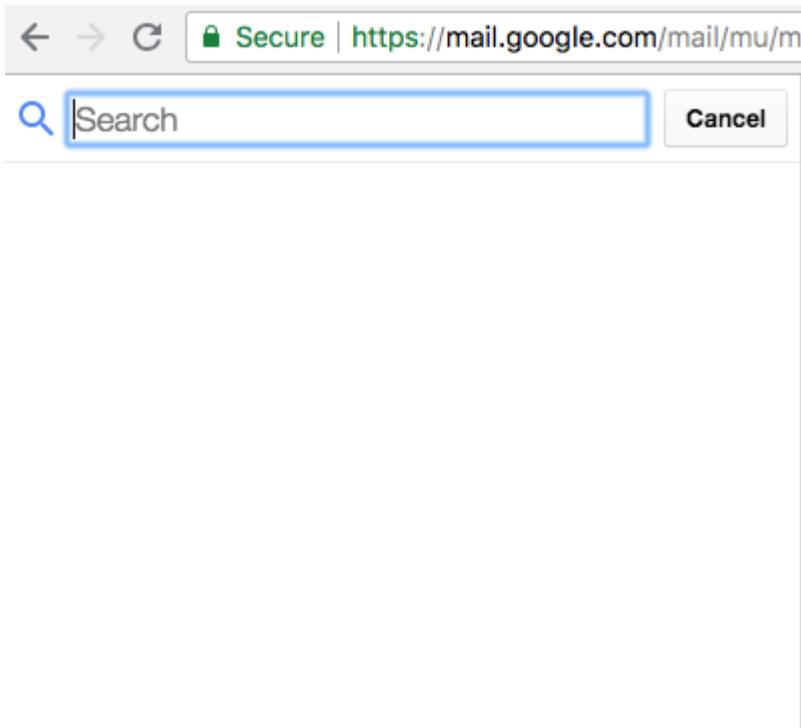
Pada desain kita sebelumnya, kolom pencarian terdapat pada halaman home tepat di bawah toolbar.



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. Component v-dialog ini akan kita tampilkan secara fullscreen dengan toolbar berupa kolom pencarian.

State Dialog

Dari simulasi di atas, dapat kita pahami bahwa kita membutuhkan variabel yang dapat dimanipulasi dari beberapa component sekaligus yaitu variabel status dialog, di mana trigger ketika dialog akan ditampilkan dan ketika dialog disembunyikan berada pada component yang berbeda. Oleh karena itu agar lebih mudah maka kita bisa menggunakan state.

Supaya lebih rapi, kita akan menjadikan state pada kasus ini sebagai sebuah modul tersendiri. Buat file `src/stores/dialog.js` di mana di dalamnya hanya ada satu state yaitu state status yang bertipe boolean untuk menentukan status visibilitas dari dialog.

```
export default {
  namespaced: true,
  state: {
    status : false,
  },
  mutations: {
    setStatus: (state, status) => {
      state.status = status
    },
  },
  actions: {
    setStatus: ({commit}, status) => {
      commit('setStatus', status)
    },
  }
}
```

```
},
getters: {
    status : state => state.status,
}
}
```

Jangan lupa daftarkan pada `src/store.js` sebagai module.

```
...
import alert from './stores/alert'
import dialog from './stores/dialog' // <== ini
...
export default new Vuex.Store({
    ...
    modules: {
        cart,
        alert,
        dialog // <== ini
    }
})
```

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).

```
public function search($keyword)
{
    $criteria = Book::select('*')
        ->where('title', 'LIKE', "%".$keyword."%")
        ->orderBy('views', 'DESC')
        ->get();
    return new BookResourceCollection($criteria);
}
```

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`.

```
<?php
use Illuminate\Http\Request;
```

```

Route::prefix('v1')->group(function () {
    // public
    // ...

    Route::get('books/top/{count}', 'BookController@top');
    Route::get('books', 'BookController@index');
    Route::get('books/slug/{slug}', 'BookController@slug');
    Route::get('books/search/{keyword}', 'BookController@search'); // <=
ini

    // private
    Route::middleware(['auth:api'])->group(function () {
        Route::post('logout', 'AuthController@logout');
    });
});

```

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.

```
{
    "status": "success",
    "message": "books data",
    "data": [
        {
            "id": 3,
            "title": "Aut delectus dolores similiique",
            "slug": "aut-delectus-dolores-similiique",
            "description": "Ipsa blanditiis quis quas voluptates tempore.
Deleniti neque recusandae vel minus quam et optio illum. Et autem eos
maxime architecto provident ut. Eaque molestias illo corrupti aliquid.",
            "author": "Tre Vandervort",
            "publisher": "Kihn, Kilback and Hoeger",
            "cover": "945e22458739b9f181c74dc7c7531105.jpg",
            "price": 100000,
            "weight": 0.5,
            "views": 0,
            "stock": 3,
            "status": "PUBLISH",
            "created_at": "2018-08-28 06:59:38",
            "updated_at": "2018-09-01 23:40:18",
            "deleted_at": null,
            "created_by": null,
            "updated_by": null,
            "deleted_by": null
        },
        {
            "id": 4,
            "title": "Placeat esse rem sunt nihil dolor",
            ...
        },
        {

```

```

        "id": 10,
        "title": "Numquam dolorem quae minima et tenetur ut",
        ...
    },
]
}

```

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. Pada projek Vue, component ini akan kita simpan dalam file `src/views/Search.vue`.

```

<template>
  <v-card>
    <v-toolbar dark color="primary">
      <v-btn icon dark @click.native="close">
        <v-icon>close</v-icon>
      </v-btn>
      <v-text-field
        v-model="keyword"
        @input="doSearch"
        hide-details
        append-icon="mic"
        flat
        label="Search"
        prepend-inner-icon="search"
        solo-inverted
        ref="txtSearch"
      ></v-text-field>
    </v-toolbar>

    <v-divider></v-divider>

    <v-subheader v-if="keyword.length>0">
      Result search "{{ keyword }}"
    </v-subheader>

    <v-alert
      :value="items.length==0 && keyword.length>0"
      color="warning">
      Sorry, but no results were found.
    </v-alert>

    <!-- Hasil pencarian ditampilkan di sini -->
    <v-list two-line v-if="items.length>0">
      <template v-for="(item) in items">
        <v-list-item
          :key="item.id" avatar @click="close"

```

```

        :to="'/book/'+ item.slug">
      <v-list-tile-avatar>
        
      </v-list-tile-avatar>
      <v-list-tile-content>
        <v-list-tile-title v-html="item.title">
        </v-list-tile-title>
        </v-list-tile-content>
      </v-list-tile>
    </template>
  </v-list>
</v-card>
</template>

```

Component v-text-field pada kode di atas berperan sebagai kolom pencarian dengan v-model keyword. Component tersebut membinding event onInput yang akan mentrigger method `doSearch`, method inilah yang akan merequest data buku sesuai keyword ke endpoint server. Adapun component v-list kita gunakan untuk menampung daftar buku hasil pencarian dengan teknik list yang masih sama dengan cara sebelumnya.

Berikut ini kode pada bagian script.

```

<script>
import { mapActions } from 'vuex'
export default {
  name: 'search',
  data () {
    return {
      keyword: '',
      items: []
    }
  },
  methods: {
    ...mapActions({
      setStatusDialog : 'dialog/setStatus',
    }),
    close(){
      this.setStatusDialog(false)
    },
    doSearch(){
      let keyword = this.keyword
      if(keyword.length>0){
        this.axios.get('/books/search/'+keyword)
        .then((response) => {
          let books = response.data.data
          this.items = books
        })
        .catch((error) => {
          console.log(error)
        })
      }
    }
  }
}

```

```

mounted(){
    if(this.$refs.txtSearch != undefined){
        this.$nextTick(() => this.$refs.txtSearch.focus())
    }
},
}
</script>

```

Terdapat pemetaan action dialog setStatus untuk menangani perubahan state status dialog melalui method close, karena halaman ini akan ditampilkan dalam sebuah dialog.

Daftarkan component ini pada template `src/App.vue`, letakkan di bawah c-alert dan bungkus dengan v-model dialog dengan atribut fullscreen plus dengan transition bawaan Vuetify.

```

<c-alert />

<v-dialog v-model="dialog" fullscreen hide-overlay transition="dialog-bottom-transition">
    <search />
</v-dialog>

```

Pada bagian script `src/App.vue`, daftarkan component `Search` serta petakan `actions` dan getters dari state dialog.

```

import { mapGetters, mapActions } from 'vuex'

export default {
    name: 'App',
    components: {
        CHeader, CSideBar, CFooter,
        CAlert: () => import(/* webpackChunkName: "c-alert" */ '@/components/CAlert.vue'),
        Search: () => import(/* webpackChunkName: "search" */ '@/views/Search.vue')
    },
    methods: {
        ...mapActions({
            setStatusDialog : 'dialog/setStatus',
        }),
    },
    computed: {
        ...mapGetters({
            statusDialog : 'dialog/status',
        }),
        dialog: {
            get () {
                return this.statusDialog
            },

```

```
    set (value) {
      this.setStatusDialog(value)
    }
  },
}
```

Trigger Halaman Pencarian

Kita kembali ke component CHeader untuk menambahkan trigger yang akan menampilkan dialog halaman pencarian melalui event onclick pada kolom pencarian.

Pada file `src/components/CHeader.vue` petakan action setStatus dan tambahkan event @click pada kolom pencarian yang akan menjalankan action tersebut.

```
<v-text-field
  v-if="isHome"
  @click="setStatusDialog(true)"
  slot="extension"
  hide-details
  append-icon="mic"
  flat
  label="Search"
  prepend-inner-icon="search"
  solo-inverted
></v-text-field>
</v-toolbar>
</template>
<script>
import { mapGetters, mapActions } from 'vuex'
export default {
  name: 'c-header',
  methods: {
    ...mapActions({
      setSideBar : 'setSideBar',
      setStatusDialog : 'dialog/setStatus', // <= ini
    }),
  },

```

Mari kita uji coba, pada halaman home kita klik kolom pencarian.

The screenshot shows a search interface with a blue header bar containing a back arrow, forward arrow, refresh icon, and the text 'localhost:8081'. Below the header is a search bar with a magnifying glass icon and the word 'dolor'. A microphone icon is also present in the search bar. The main content area is titled 'Result search "dolor"'. It displays three search results, each with a small circular thumbnail image:

- Aut delectus dolores similiique
- Placeat esse rem sunt nihil dolor
- Numerum dolorem quae minima et tenetii

Halaman Login

Halaman ini menampilkan form login email dan password serta tombol login. Halaman ini dapat diakses melalui link login pada component CSideBar. 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. Jangan lupa pastikan route login sudah didaftarkan pada api.php

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.

```
{  
    "status": "success",  
    "message": "Login sukses",  
    "data": {  
        "id": 1,  
        "name": "Elvera Crona",  
        "email": "lou56@example.org",  
        "created_at": "2018-08-28 06:59:10",  
        "updated_at": "2018-09-04 00:09:45",  
        "roles": "[\"CUSTOMER\"]",  
        "address": "-",  
        "city": "114",  
        "province": "1",  
        "phone": "0123451",  
        "avatar": "d98777e02d7ca70d8a3fffeef460d4acc.jpg",  
        "status": "ACTIVE",  
        "api_token": "  
    }  
}
```

```
"h8wPXhAIrvoa8fc7jFj80KU6DeRXupAGoWhLtHQ9nSuwja0wA24RaYRqzdUT"
```

```
}
```

```
}
```

State Login / Auth

Data user hasil request ke endpoint login tersebut nantinya perlu kita simpan dalam sebuah state di frontend 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.

```
export default {
  namespaced: true,
  state: {
    user: {},
  },
  mutations: {
    set: (state, payload) => {
      state.user = payload
    },
  },
  actions: {
    set: ({commit}, payload) => {
      commit('set', payload)
    },
  },
  getters: {
    user: state => state.user,
    guest: state => Object.keys(state.user).length === 0,
  }
}
```

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`

```
...
import dialog from './stores/dialog'
import auth from './stores/auth'

...
```

```
modules: {
  cart,
  alert,
  dialog,
  auth
}
})
```

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.

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

Supaya component sebelumnya di-cache maka kita bisa bungkus menggunakan component **keep-alive**

```
<keep-alive>
<component :is="currentComponent"></component>
</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.

```
<c-alert />

<keep-alive>
  <v-dialog v-model="dialog" fullscreen hide-overlay transition="dialog-bottom-transition">
    <component :is="currentComponent"></component>
  </v-dialog>
</keep-alive>
```

Karena menyangkut banyak component, maka cara paling mudah untuk mengontrol perubahan variabel global adalah dengan menggunakan state. Jika sebelumnya kita pernah membuat state dialog yang intinya untuk mendefinisikan status dialog (ditampilkan atau disembunyikan), maka sekarang kita tambahkan satu state lagi yaitu **component** untuk menyimpan nilai component yang sedang digunakan (current component).

Berikut ini kode lengkap untuk file `src/stores/dialog.js`

```
export default {
  namespaced: true,
  state: {
    status: false,
    component: '', // search or login or other
  },
  mutations: {
    setStatus: (state, status) => {
      state.status = status
    },
    setComponent: (state, component) => {
      state.component = component
    },
  },
  actions: {
    setStatus: ({commit}, status) => {
      commit('setStatus', status)
    },
    setComponent: ({commit}, component) => {
      commit('setComponent', component)
    },
  },
  getters: {
    status: state => state.status,
    component: state => state.component,
  }
}
```

Kembali ke `src/App.vue`, kita tambahkan getters `dialog/component` pada scriptnya.

```
computed: {
  ...mapGetters({
    statusDialog: 'dialog/status',
    currentComponent: 'dialog/component' // <= ini
  }),
}
```

Sebagai ujicoba apakah langkah ini tepat, mari kita implementasikan untuk component search. Buka component `src/components/CHeader.vue`, tambahkan action dialog/setComponent, tujuannya ketika kolom pencarian di-click maka current component akan diset menjadi bernilai `search` dan status dialognya true.

```
export default {
  name: 'c-header',
  methods: {
    ...mapActions({
      setSideBar: 'setSideBar',
    })
  }
}
```

```
setStatusDialog : 'dialog/setStatus',
setComponent   : 'dialog/setComponent', // <= ini
},
search(){
  this.setStatusDialog(true)
  this.setComponent('search') // <= ini
  this.setSideBar(false)
}
},
...
}
```

Jika kode di atas benar maka ketika kita ujicoba akan muncul popup dialog yang di dalamnya terdapat tampilan dari component search.

Lakukan hal yang sama untuk login. Buka `src/components/SideBar.vue`, tambahkan methods login yang nantinya akan kita bind ke menu login (tentu saja kamu perlu melakukan mapping action `dialog/setComponent` terlebih dahulu)

```
login(){
  this.setStatusDialog(true)
  this.setComponent('login')
  this.setSideBar(false)
}
```

Pada template tombol login, kita tambahkan fungsi login pada event onclick.

```
<v-btn
  @click="login()"
  block round depressed
  color="accent lighten-1"
  class="white--text">
  Login
  <v-icon right dark>lock_open</v-icon>
</v-btn>
```

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 vuetify, dimana rule validasi dideklarasikan melalui atribut `rules` pada `v-text-field`.

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.

```
<template>
  <v-card>
    <v-toolbar dark color="primary">
      <v-btn icon dark @click="close">
        <v-icon>close</v-icon>
      </v-btn>
      <v-toolbar-title>Login and Start Shopping!</v-toolbar-title>
    </v-toolbar>
    <v-divider></v-divider>

    <v-container fluid>
      <v-form ref="form" v-model="valid" lazy-validation>
        <v-text-field
          v-model="email"
          :rules="emailRules"
          label="E-mail"
          required
          append-icon="email"
        ></v-text-field>
        <v-text-field
          v-model="password"
          :append-icon="showPassword ? 'visibility' : 'visibility_off'"
          :rules="passwordRules"
          :type="showPassword ? 'text' : 'password'"
          label="Password"
          hint="At least 6 characters"
          counter
          @click:append="showPassword = !showPassword"
        ></v-text-field>

        <div class="text-xs-center">
          <v-btn
            color="accent lighten-1"
            :disabled="!valid"
            @click="submit"
          >
            Login
            <v-icon right dark>lock_open</v-icon>
          </v-btn>
        </div>
      </v-form>
    </v-container>
  </v-card>
</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.

```
<script>
  import { mapGetters, mapActions } from 'vuex'
  export default {
    name: 'login',
    data () {
      return {
        valid: true,
        email: 'lou56@example.org',
        emailRules: [
          v => !!v || 'E-mail is required',
          v => /([a-zA-Z0-9_]{1,})(@)([a-zA-Z0-9_]{2,}).([a-zA-Z0-9_]{2,})+/.test(v) || 'E-mail must be valid'
        ],
        showPassword: false,
        password: '',
        passwordRules: [
          v => !!v || 'Password required.',
          v => (v && v.length >= 6) || 'Min 6 characters',
        ],
      }
    },
    computed: {
      ...mapGetters({
        user : 'auth/user',
      }),
    },
    methods: {
      ...mapActions({
        setAlert : 'alert/set',
        setStatusDialog : 'dialog/setStatus',
        setAuth : 'auth/set',
      }),
      close(){
        this.setStatusDialog(false)
      },
      submit () {
        if (this.$refs.form.validate()) {
          let formData = {
            'email' : this.email,
            'password' : this.password
          }
          this.axios.post('/login', formData)
            .then((response) => {
              let data_user = response.data.data
              this.setAuth(data_user)
              if(this.user.id>0){

```

```

        this.setAlert({
            status : true,
            text   : 'Login success',
            type   : 'success',
        })
        this.setStatusDialog(false)
    }
    else{
        this.setAlert({
            status : true,
            text   : 'Login error',
            type   : 'error',
        })
    }
})
.catch((error) => {
    let responses = error.response
    this.setAlert({
        status : true,
        text   : responses.data.message,
        type   : 'error',
    })
})
}
},
},
}
</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 apphanumeric dan underscore minimal 1 karakter
- `@` : 1 karakter @ (a keong)
- `([a-zA-Z0-9_]{2,})` : karakter apphanumeric dan underscore minimal 2 karakter
- `.` : karakter . (titik)
- `([a-zA-Z0-9_]{2,})` : karakter apphanumeric dan underscore minimal 2 karakter

Ketika isian form valid maka akan aktif tombol login 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**.

```

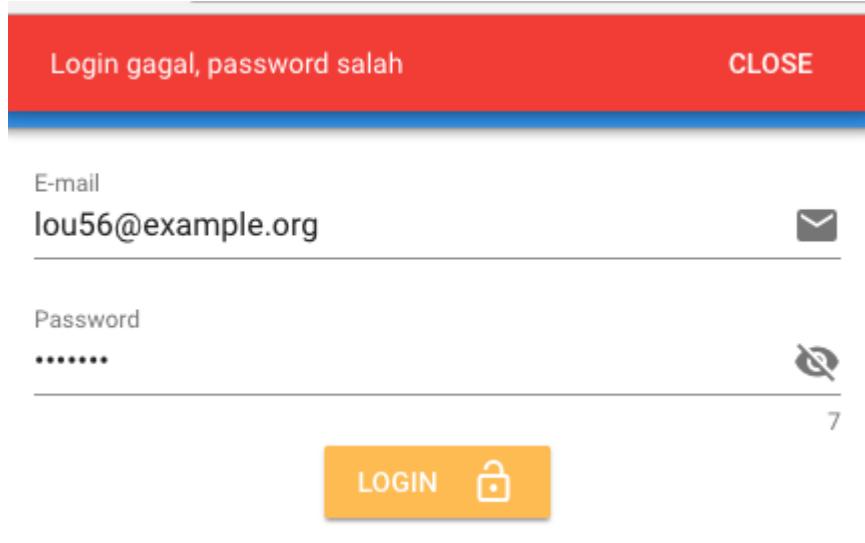
...
export default {
    name: 'App',
    components: {

```

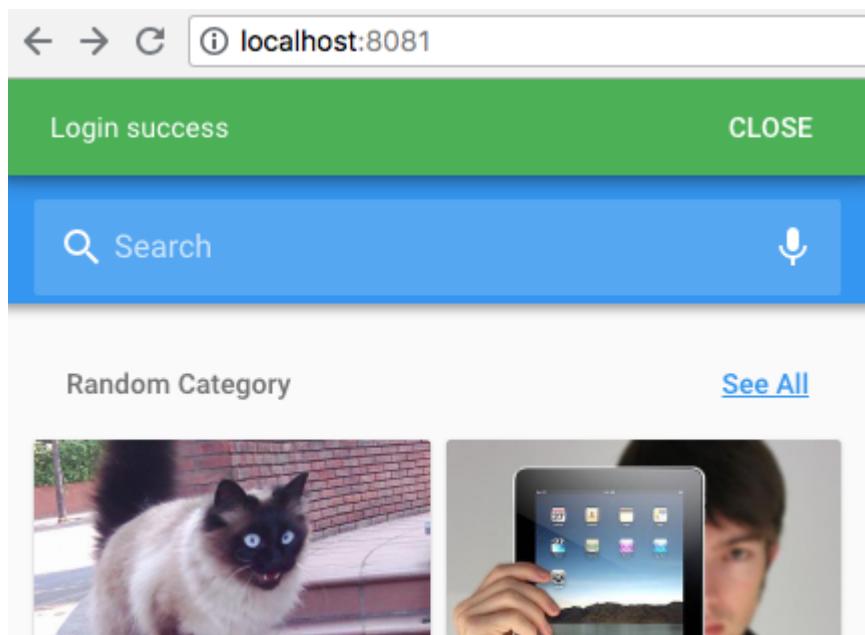
```
...
CAlert: () => import(/* webpackChunkName: "c-alert" */  
'@/components/CAlert.vue'),  
Search: () => import(/* webpackChunkName: "search" */  
'@/views/Search.vue'),  
Login: () => import(/* webpackChunkName: "login" */  
'@/views/Login.vue'), // <= ini
```

Mari kita uji coba.

Jika password salah maka akan menampilkan pesan login gagal.



Jika login berhasil, maka dialog akan ditutup dan menampilkan pesan bahwa login berhasil



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 CSideBar. 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.

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.

```
...
// private
Route::middleware(['auth:api'])->group(function () {
    Route::post('logout', 'AuthController@logout'); // <= ini
});
});
```

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.

```
{
    "status": "success",
    "message": "logout berhasil",
    "data": []
}
```

Membuat Link Logout

Sekarang kita masuk ke projek Vue, buka file `/src/components/CSideBar.vue`. Tambahkan component v-list dan tampilkan avatar serta nama usernya, kemudian tambahkan juga tombol logout yang akan mentrigger method logout. Kita bisa gunakan `v-if="!guest"` untuk menampilkan link logout hanya untuk user yang sudah login. Adapun pada tombol login dan register dapat juga kita tambahkan `v-if="guest"` sehingga hanya akan tampil jika user belum login.

Berikut kode templatanya.

```
<v-list v-if="guest">
    <v-list-tile>
        <v-btn
            @click="register()"
            depressed
            block
            round>
```

```

        color="secondary"
        class="white--text"
      >
    Register
    <v-icon right dark>person_add</v-icon>
    </v-btn>
  </v-list-tile>
  <v-list-tile>
    <v-btn
      @click="login()"
      block
      round
      depressed
      color="accent lighten-1"
      class="white--text"
    >
    Login
    <v-icon right dark>lock_open</v-icon>
    </v-btn>
  </v-list-tile>
</v-list>

<v-list v-if="!guest">
  <v-list-tile>
    <v-list-tile-avatar>
      
    </v-list-tile-avatar>

    <v-list-tile-content>
      <v-list-tile-title>
        {{ user.name }}
      </v-list-tile-title>
    </v-list-tile-content>
  </v-list-tile>
  <v-list-tile>
    <v-btn
      block
      small
      round
      depressed
      color="error lighten-1"
      class="white--text"
      @click.stop="logout();"
    >
    Logout
    <v-icon small right dark>settings_power</v-icon>
    </v-btn>
  </v-list-tile>
</v-list>

```

Adapun untuk kode scriptnya, pada intinya kita tambahkan fungsi logout yang memanggil endpoint logout kemudian kita set state user menjadi blank.

```

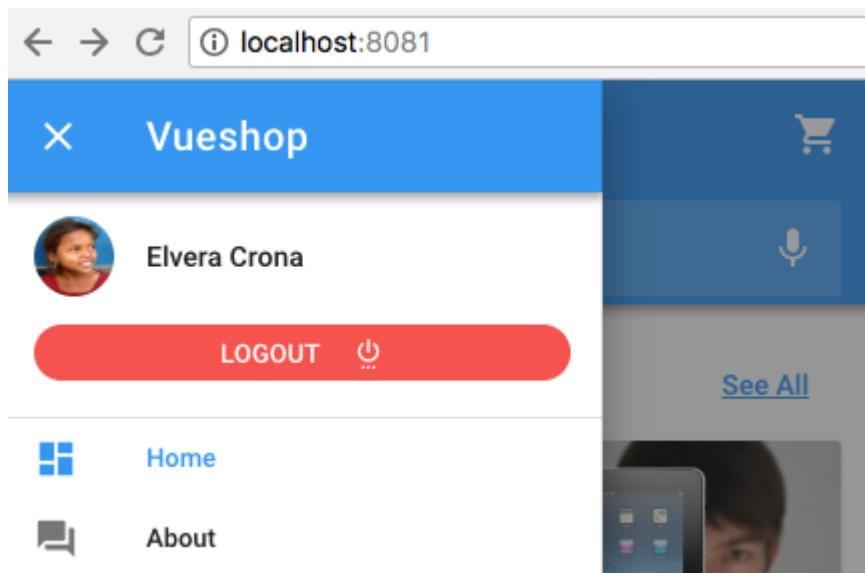
<script>
import { mapGetters, mapActions } from 'vuex'
export default {
  name: 'c-side-bar',
  data: () => ({
    items: [
      { title: 'Home', icon: 'dashboard', route: 'home' },
      { title: 'About', icon: 'question_answer', route: 'about' },
    ]
  }),
  computed: {
    ...mapGetters({
      sideBar: 'sideBar',
      user: 'auth/user',
      guest: 'auth/guest',
    }),
    drawer: {
      get () {
        return this.sideBar
      },
      set (value) {
        this.setSideBar(value)
      }
    },
  },
  methods: {
    ...mapActions({
      setSideBar: 'setSideBar',
      setStatusDialog: 'dialog/setStatus',
      setComponent: 'dialog/setComponent',
      setAuth: 'auth/set',
      setAlert: 'alert/set',
    }),
    login(){
      this.setStatusDialog(true)
      this.setComponent('login')
      this.setSideBar(false)
    },
    register(){
      this.setStatusDialog(true)
      this.setComponent('register')
      this.setSideBar(false)
    },
    logout(){
      let config = {
        headers: {
          'Authorization': 'Bearer ' + this.user.api_token,
        },
      }
      this.axios.post('/logout', {}, config)
        .then((response) => {
          this.setAuth({}) // kosongkan auth ketika logout
          this.setAlert({
        })
    }
  }
}

```

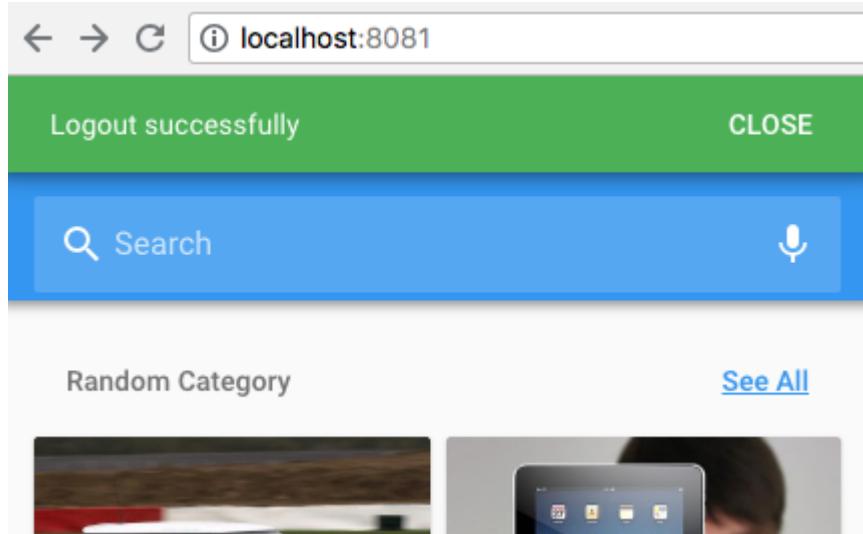
```
        status : true,
        text   : 'Logout successfully',
        type   : 'success',
    })
    this.setSideBar(false)
})
.catch((error) => {
    let responses = error.response
    this.setAlert({
        status : true,
        text   : responses.data.message,
        type   : 'error',
    })
})
},
}
</script>
```

Mari kita uji coba.

Pada sidebar akan muncul tampilan berikut.



Jika logout di klik maka 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 component CSideBar. 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.

Jangan lupa daftarkan route register pada api.php

```
Route::prefix('v1')->group(function () {
    // public
    Route::post('login', 'AuthController@login');
    Route::post('register', 'AuthController@register'); // <= ini
    ...
});
```

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.

```
{
    "status": "success",
    "message": "register successfully",
    "data": {
        "name": "hafid",
        "email": "hafid@example.org",
        "roles": "[\"CUSTOMER\"]",
        "updated_at": "2018-09-07 23:03:42",
        "created_at": "2018-09-07 23:03:42",
        "id": 8,
        "api_token": "AiifLcehILDS6pM6MD0GFI5p5deK5bZGICgXt4aniYnWh1bAw6yT1cpmajhW"
}
```

```

    }
}

```

Membuat Component Register

Pada dasarnya component register ini tidak berbeda jauh dengan component login. Hanya saja ada tambahan satu field yaitu nama. Validasinya pun juga sama sehingga tidak dijelaskan lagi disini.

Berikut ini kode lengkap untuk component `/src/views/Register.vue`.

```

<template>
  <v-card>
    <v-toolbar dark color="primary">
      <v-btn icon dark @click.native="close">
        <v-icon>close</v-icon>
      </v-btn>
      <v-toolbar-title>Register and Start Shopping!</v-toolbar-title>
    </v-toolbar>
    <v-divider></v-divider>

    <v-container fluid>
      <v-form ref="form" v-model="valid" lazy-validation>
        <v-text-field
          v-model="name"
          :rules="nameRules"
          :counter="255"
          label="Name"
          required
          append-icon="person"
        ></v-text-field>
        <v-text-field
          v-model="email"
          :rules="emailRules"
          label="E-mail"
          required
          append-icon="email"
        ></v-text-field>
        <v-text-field
          v-model="password"
          :append-icon="showPassword ? 'visibility' : 'visibility_off'"
          :rules="passwordRules"
          :type="showPassword ? 'text' : 'password'"
          label="Password"
          hint="At least 6 characters"
          counter
          @click:append="showPassword = !showPassword"
        ></v-text-field>
        <v-checkbox
          v-model="checkbox"
          :rules="[v => !!v || 'You must agree to continue!']"
          label="Do You agree with our Privacy Policy?"
        ></v-checkbox>
      </v-form>
    </v-container>
  </v-card>

```

```

    required
  ></v-checkbox>

  <div class="text-xs-center">
    <v-btn
      color="primary"
      :disabled="!valid"
      @click="submit"
    >
      submit
      <v-icon right dark>person_add</v-icon>
    </v-btn>
    <v-btn @click="clear">clear</v-btn>
  </div>

  </v-form>
</v-container>

</v-card>
</template>
<script>
  import { mapGetters, mapActions } from 'vuex'
  export default {
    name: 'register',
    data () {
      return {
        valid: true,
        name: '',
        nameRules: [
          v => !!v || 'Name is required',
          v => (v && v.length <= 255) || 'Name must be less than 255
characters'
        ],
        showPassword: false,
        password: '',
        passwordRules: [
          v => !!v || 'Password required.',
          v => (v && v.length >= 6) || 'Min 6 characters',
        ],
        email: '',
        emailRules: [
          v => !!v || 'E-mail is required',
          v => /([a-zA-Z0-9_]{1,})(@)([a-zA-Z0-9_]{2,}).([a-zA-Z0-9_]
{2,})+/.test(v) || 'E-mail must be valid'
        ],
        checkbox: false
      }
    },
    computed: {
      ...mapGetters({
        user : 'auth/user',
      }),
    },
    methods: {

```

```

...mapActions({
    setAlert : 'alert/set',
    setStatusDialog : 'dialog/setStatus',
    setAuth : 'auth/set',
}),
close(){
    this.setStatusDialog(false)
},
submit () {
    if (this.$refs.form.validate()) {
        let formData = new FormData()
        formData.set('name', this.name)
        formData.set('email', this.email)
        formData.set('password', this.password)

        this.axios.post('/register', formData)
            .then((response) => {
                let data_user = response.data.data
                this.setAuth(data_user)
                this.setAlert({
                    status : true,
                    text : 'register success',
                    type : 'success',
                })
                this.close()
            })
            .catch((error) => {
                console.log(error)
                let responses = error.response
                this.setAlert({
                    status : true,
                    text : responses.data.message,
                    type : 'error',
                })
            })
    }
},
clear () {
    this.$refs.form.reset()
}
},
</script>

```

Setelah form di validasi maka data registrasi akan dikirim ke endpoint register menggunakan method post.

Jangan lupa daftarkan component ini pada [/src/App.vue](#).

```

...
export default {
    name: 'App',
    components: {

```

```
...
Login: () => import(/* webpackChunkName: "login" */  
'@/views/Login.vue'),  
Register: () => import(/* webpackChunkName: "register" */  
'@/views/Register.vue'), // <= ini
```

Link Halaman Register

Untuk membuka halaman register melalui menu register pada sidebar. Tambahkan fungsi register pada component `/src/components/CSidebar.Vue`

```
methods: {  
    ...  
    register(){  
        this.setStatusDialog(true)  
        this.setComponent('register')  
        this.setSideBar(false)  
    }  
},
```

Lalu pada template tambahkan fungsi register pada event onclick tombol register

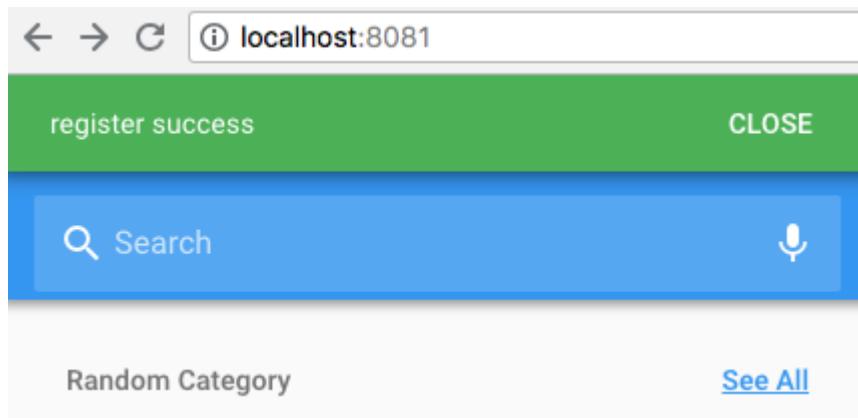
```
<v-btn  
@click="register()"  
depressed block round  
color="secondary"  
class="white--text">  
    Register  
    <v-icon right dark>person_add</v-icon>  
</v-btn>
```

Mari kita uji coba.

Jika email sudah terdaftar maka akan menampilkan pesan bahwa email sudah digunakan.

The screenshot shows a registration dialog box. At the top, a red bar displays the error message: { "email": ["The email has already been taken."] }. To the right is a 'CLOSE' button. Below the error message, there are input fields for 'Name' (Hafid Mukhlasin), 'E-mail' (hafidmukhlasin@gmail.com), and 'Password' (*****). A password strength meter indicates a length of 6. Below these fields is a checkbox labeled 'Do You agree with our Privacy Policy?' which is checked. At the bottom are two buttons: 'SUBMIT' with a user icon and 'CLEAR'.

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

Link keranjang belanja terdapat pada component CHeader yaitu di `src/components/CHeader.vue`. Tambahkan methods cart.

```
cart(){
    this.setStatusDialog(true)
    this.setComponent('cart')
    this.setSideBar(false)
}
```

Lalu pada templatennya, tambahkan event onclick yang membinding method cart().

```
<v-btn icon @click="cart()">
    <v-badge left overlap color="orange">
        <span slot="badge" v-if="countCart>0"> {{ countCart }} </span>
        <v-icon>shopping_cart</v-icon>
    </v-badge>
</v-btn>
```

Mengupdate State Cart

State cart [src/stores/cart.js](#) yang telah kita buat sebelumnya perlu kita perbaharui dengan menambahkan beberapa fitur yaitu kurangi item barang pada cart, set cart untuk batch update, serta getters untuk menghitung total harga, total berat dan total jumlah item.

```
export default {
    namespaced: true,
    state: {
        carts: []
    },
    mutations: {
        insert: (state, payload) => {
            state.carts.push({
                id: payload.id,
                title: payload.title,
                cover: payload.cover,
                price: payload.price,
                weight: payload.weight,
                quantity: 1
            })
        },
        update: (state, payload) => {
            let idx = state.carts.indexOf(payload);
            state.carts.splice(idx, 1, {
                id: payload.id,
                title: payload.title,
                cover: payload.cover,
                price: payload.price,
                weight: payload.weight,
                quantity: payload.quantity
            });
            if(payload.quantity<=0){
```

```

        state.carts.splice(idx, 1)
    }
},
set: (state, payload) => {
    state.carts = payload
},
},
actions: {
    add: ({state, commit}, payload) => {
        let cartItem = state.carts.find(item => item.id === payload.id)
        if(!cartItem){
            commit('insert', payload)
        }
        else{
            cartItem.quantity++
            commit('update', cartItem)
        }
    },
    remove: ({state, commit}, payload) => {
        let cartItem = state.carts.find(item => item.id === payload.id)
        if(cartItem){
            cartItem.quantity--
            commit('update', cartItem)
        }
    },
    set: ({commit}, payload) => {
        commit('set', payload)
    },
},
getters: {
    carts : state => state.carts,
    count : (state) => {
        return state.carts.length
    },
    totalPrice: (state) => {
        let total = 0
        state.carts.forEach(function(cart) {
            total += cart.price * cart.quantity
        })
        return total
    },
    totalQuantity: (state) => {
        let total = 0
        state.carts.forEach(function(cart) {
            total += cart.quantity
        })
        return total
    },
    totalWeight: (state) => {
        let total = 0
        state.carts.forEach(function(cart) {
            total += cart.weight
        })
        return total
    }
}

```

```

        },
    }
}

```

Membuat Component Keranjang Belanja

Component ini kita beri nama cart yang berfungsi menampilkan cart dalam bentuk tampilan list dari data state cart yang berbentuk array dari objek.

Seperti sebelumnya, kita menggunakan component v-list untuk menampilkan data keranjang belanja. Harga kita tampilkan dalam format Indonesia menggunakan fungsi `toLocaleString('id-ID')`.

File `src/views/Cart.vue`.

```

<template>
<v-card>
  <v-toolbar dark color="primary">
    <v-btn icon dark @click="close">
      <v-icon>close</v-icon>
    </v-btn>
    <v-toolbar-title>Your Shopping Cart</v-toolbar-title>
    <v-spacer />
  </v-toolbar>
  <v-divider></v-divider>
  <div v-if="countCart === 0">
    <v-alert
      value="true"
      color="success"
      icon="new_releases"
    >
      Keranjang belanja kosong!
    </v-alert>
  </div>
  <div v-else>
    <v-list two-line>
      <template v-for="item in carts">
        <v-list-item
          :key="item.id"
          avatar
        >
          <v-list-item-avatar>
            
          </v-list-item-avatar>

          <v-list-item-content>
            <v-list-item-title v-html="item.title"></v-list-item-title>
            <v-list-item-subtitle>
              Rp. {{ item.price.toLocaleString('id-ID') }}
              <span style="float:right">
                <v-btn icon small rounded depressed
                  @click.stop="removeCart(item)">

```

```

        <v-icon dark color="error">remove_circle</v-icon>
      </v-btn>
      {{ item.quantity }}
      <v-btn icon small rounded depressed
@click.stop="insertCart(item)">
        <v-icon dark color="success">add_circle</v-icon>
      </v-btn>
      </span>
    </v-list-tile-sub-title>
    <v-list-tile-sub-title>
      <v-divider/>
    </v-list-tile-sub-title>
  </v-list-tile-content>
  <v-list-tile-action>
    <v-list-tile-action-text>{{ item.weight }} kg</v-list-tile-
action-text>
  </v-list-tile-action>
</v-list-tile>

</template>
</v-list>
<v-card>
  <v-layout row wrap>
    <v-flex xs6 text-xs-center>
      Total Price ({{ totalQuantity }} items)
      <div class="title">{{ totalPrice.toLocaleString('id-ID') }}</div>
</div>
  </v-flex>
  <v-flex xs6 text-xs-center>
    <v-btn color="orange" dark @click="checkout">
      <v-icon>check_circle</v-icon> &ampnbsp
      Checkout
    </v-btn>
  </v-flex>
</v-layout>
</v-card>
</div>
</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.

```

<script>
  import { mapGetters, mapActions } from 'vuex'
  export default {
    name: 'cart',
    computed: {
      ...mapGetters({
        carts : 'cart/carts',
        countCart : 'cart/count',

```

```
        totalPrice    : 'cart/totalPrice',
        totalQuantity : 'cart/totalQuantity',
        totalWeight   : 'cart/totalWeight',
      },
    },
  methods: {
    ...mapActions({
      setStatusDialog  : 'dialog.setStatus',
      setAlert         : 'alert/set',
      addCart          : 'cart/add',
      removeCart       : 'cart/remove',
      setCart          : 'cart/set',
    }),
    close(){
      this.setStatusDialog(false)
    },
    checkout(){
      this.close()
      this.$router.push({path: "/checkout"})
    }
  },
</script>
```

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

```
php artisan make:resource Provinces --collection
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`

```
public function toArray($request)
{
    return [
        'status'  => 'success',
        'message' => 'provinces data',
        'data'     => parent::toArray($request),
    ];
}
```

Sedangkan berikut ini modifikasi fungsi toArray pada `app\Http\Resources\Cities.php`

```
public function toArray($request)
{
    return [
        'status'  => 'success',
        'message' => 'cities data',
        'data'     => parent::toArray($request),
    ];
}
```

Kemudian kita akan membuat controller Shop untuk dua endpoint ini.

```
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.

```
public function provinces()
{
    return new ProvinceResourceCollection(Province::get());
}

public function cities()
{
    return new CityResourceCollection(City::get());
}
```

Tentu saja pada bagian atas kode ini, kamu harus use dulu model dan resourcennya.

```
use App\Province;
use App\Http\Resources\Provinces as ProvinceResourceCollection;
use App\City;
use App\Http\Resources\Cities as CityResourceCollection;
```

Jika sudah, jangan lupa daftarkan route dari dua fungsi ini pada api.php.

```
Route::prefix('v1')->group(function () {
    // public
    // ... route lain

    Route::get('provinces', 'ShopController@provinces'); // <= ini
    Route::get('cities', 'ShopController@cities'); // <= ini

    // .. route lain
```

Waktunya uji coba, silakan akses endpoint <http://larashop-api.test/v1/provinces>, maka responnya:

```
{
    "status": "success",
    "message": "provinces data",
    "data": [
        {
            "id": 1,
            "province": "Bali"
        },
        {
            "id": 2,
            "province": "Bangka Belitung"
        },
        {
            "id": 3,
            "province": "Banten"
        },
        ...
    ]}
```

Sedangkan endpoint <http://larashop-api.test/v1/cities>, responsenya.

```
{
    "status": "success",
    "message": "cities data",
    "data": [
        {
            "id": 1,
            "province_id": 21,
```

```

    "city_name": "Aceh Barat",
    "type": "Kabupaten",
    "postal_code": "23681"
},
{
    "id": 2,
    "province_id": 21,
    "city_name": "Aceh Barat Daya",
    "type": "Kabupaten",
    "postal_code": "23764"
},
...

```

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](#)

```

export default {
  namespaced: true,
  state: {
    provinces: [],
    cities: [],
  },
  mutations: {
    setProvinces: (state, value) => {
      state.provinces = value
    },
    setCities: (state, value) => {
      state.cities = value
    },
  },
  actions: {
    setProvinces: ({commit}, value) => {
      commit('setProvinces', value)
    },
    setCities: ({commit}, value) => {
      commit('setCities', value)
    },
  },
  getters: {
    provinces: state => state.provinces,
    cities: state => state.cities,
  }
}

```

Jangan lupa daftarkan module ini pada [src/store.js](#)

```
// ... kode lain
import region from './stores/region' // <= ini
```

```
// ... kode lain

modules: {
    cart,
    alert,
    dialog,
    auth,
    region // <= ini
}
})
```

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.

```
public function shipping(Request $request)
{
    $user = Auth::user(); // mendapatkan current user yang login
    $status = "error";
    $message = "";
    $data = null;
    $code = 200;
    if ($user) {
        $this->validate($request, [
            'name' => 'required',
            'address' => 'required',
            'phone' => 'required',
            'province_id' => 'required',
            'city_id' => 'required',
        ]);
        $user->name = $request->name;
        $user->address = $request->address;
        $user->phone = $request->phone;
        $user->province_id = $request->province_id;
        $user->city_id = $request->city_id;
        if($user->save()){
            $status = "success";
            $message = "Update shipping success";
            $data = $user->toArray();
        }
        else{
            $message = "Update shipping failed";
        }
    }
    else{
```

```

        $message = "User not found";
    }

    return response()->json([
        'status' => $status,
        'message' => $message,
        'data' => $data
    ], $code);
}

```

Membuat Component Checkout Part 1

Pada bagian pertama pembuatan component 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 di simpan pada `src/views/Checkout.vue`.

```

<template>
<div>
    <v-subheader>Shipping Address</v-subheader>
    <div>
        <v-card flat>
        <v-container>
            <v-form ref="form" lazy-validation>
                <v-text-field
                    label="Name"
                    v-model="name"
                    required
                    append-icon="person"
                ></v-text-field>

                <v-textarea
                    label="Address"
                    v-model="address"
                    required
                    auto-grow
                    rows="3"
                ></v-textarea>

                <v-text-field
                    label="Phone"
                    v-model="phone"
                    required
                    append-icon="phone"
                ></v-text-field>

                <v-select
                    v-model="province_id"

```

```

        :items="provinces"
        item-text="province"
        item-value="id"
        label="Province"
        persistent-hint
        single-line
    ></v-select>

    <v-select
        v-model="city_id"
        v-if="province_id>0"
        :items="citiesByProvince"
        item-text="city_name"
        item-value="id"
        label="City"
        persistent-hint
        single-line
    ></v-select>

</v-form>
<v-card-actions>
    <v-btn color="success" dark @click="saveShipping">
        <v-icon>save</v-icon> &ampnbsp
        Save
    </v-btn>
</v-card-actions>
</v-container>
</v-card>
</div>
</div>
</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.

```

<script>
    import { mapGetters, mapActions } from 'vuex'
    export default {
        data () {
            return {
                name: '',
                address: '',
                phone: '',
                province_id: 0,
                city_id: 0,
            }
        }
    }

```

```

},
computed: {
    ...mapGetters({
        user      : 'auth/user',
        provinces: 'region/provinces',
        cities    : 'region/cities',
    }),
    citiesByProvince(){
        let province_id = this.province_id
        return this.cities.filter(function(city){
            if (city.province_id==province_id) return city
        })
    },
},
methods: {
    ...mapActions({
        setAlert      : 'alert/set',
        setAuth       : 'auth/set',
        setProvinces : 'region/setProvinces',
        setCities     : 'region/setCities',
    }),
    saveShipping(){
        let formData = new FormData()
        formData.set('name', this.name)
        formData.set('address', this.address)
        formData.set('phone', this.phone)
        formData.set('province_id', this.province_id)
        formData.set('city_id', this.city_id)

        let config = {
            headers: {
                'Authorization': 'Bearer ' + this.user.api_token,
            },
        }

        this.axios.post('/shipping', formData, config)
            .then((response) => {
                this.setAuth(response.data.data)
                this.setAlert({
                    status : true,
                    text   : response.data.message,
                    type   : 'success',
                })
            })
            .catch((error) => {
                let responses = error.response
                this.setAlert({
                    status : true,
                    text   : responses.data.message,
                    type   : 'error',
                })
            })
        },
    },
},
}
,
```

```

created(){
    this.name = this.user.name
    this.address = this.user.address
    this.phone = this.user.phone
    this.city_id = this.user.city_id
    this.province_id = this.user.province_id

    if(this.provinces && this.provinces.length==0){
        this.axios.get('/provinces')
        .then((response) => {
            this.setProvinces(response.data.data)
        })

        this.axios.get('/cities')
        .then((response) => {
            this.setCities(response.data.data)
        })
    }
}

</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 }`

```

const router = new Router({
  mode: 'history',
  base: process.env.BASE_URL,
  routes: [
    // ...
    {
      path: '/checkout',
      name: 'checkout',
      component: () => import(/* webpackChunkName: "checkout" */ './views/Checkout.vue'),
      meta: { auth: true } // penandanya ini gans
    },
  ]
}

```

Kemudian 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.

```

router.beforeEach((to, from, next) => {
  // jika routing ada meta auth-nya maka

```

```
if (to.matched.some(record => record.meta.auth)) {
  // jika user adalah guest
  if(store.getters['auth/guest']){
    // tampilkan pesan bahwa harus login dulu
    store.dispatch('alert/set', {
      status : true,
      text   : 'Login first',
      type   : 'error',
    })
  }

  // tampilkan form login
  store.dispatch('dialog/setComponent', 'login')
  store.dispatch('dialog.setStatus', true)
}
else{
  next()
}
}
else{
  next()
}
}

export default router
```

Jangan lupa, karena kita menggunakan store maka sebelum menjalankan script ini seharusnya kita import dahulu store-nya.

```
import store from './store'
```

Mari kita coba. Masuk ke keranjang belanja.

The screenshot shows a web browser window with the following details:

- URL:** `localhost:8081/book/esse-quo-rem-explicabo`
- Title Bar:** Your Shopping Cart
- Cart Item:** Esse quo rem explicabo (Rp. 300.000, 1 unit, 0.5 kg)
- Total Price:** Total Price (1 items) 300.000
- Checkout Button:** A large orange button with a checkmark icon labeled "CHECKOUT".

Klik tombol checkout, maka dalam kondisi belum login akan muncul form login dan pesan agar login dahulu.

← → ⌛ ⓘ localhost:8081/book/esse-quo-rem-explicabo

Login first CLOSE

E-mail
mayer.glennie@example.com ✉

Password
***** 👁 6

LOGIN

Kemudian, ketika login maka kita akan diarahkan ke halaman utama.

← → ⌛ ⓘ localhost:8081

Login success CLOSE

Search Mic icon

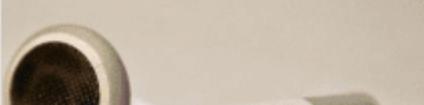
Random Category [See All](#)



quisquam



et





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`

```
export default new Vuex.Store({
  state: {
    // ...
    prevUrl: '',
  },
  mutations: {
    // ...
    setPrevUrl: (state, value) => {
      state.prevUrl = value
    },
  },
  actions: {
    // ...
    setPrevUrl: ({commit}, value) => {
      commit('setPrevUrl', value)
    },
  },
  getters: {
    // ...
    prevUrl: state => state.prevUrl,
  },
})
```

Lalu pada navigation guard di `src/router.js` kita simpan previous URL.

```
router.beforeEach((to, from, next) => {
  if (to.matched.some(record => record.meta.auth)) {
    if(store.getters['auth/guest']){
      store.dispatch('alert/set', {
        status : true,
        text : 'Login first',
        type : 'error',
      })
      store.dispatch('setPrevUrl', to.path) // tambahkan ini
      store.dispatch('dialog/setComponent', 'login')
      store.dispatch('dialog.setStatus', true)
    }
    else{
      next()
    }
  }
  else{
    next()
  }
})
```

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/views/Login.vue` petakan state prevUrl.

```
...mapGetters({
  user : 'auth/user',
  prevUrl : 'prevUrl', // <= ini
}),
```

Lalu pada method submit, ketika login berhasil maka lakukan pengecekan terhadap state prevUrl.

```
this.axios.post('/login', formData)
.then((response) => {
  let data_user = response.data.data
  this.setAuth(data_user)
  if(this.user.id>0){
    this.setAlert({
      status : true,
      text : 'Login success',
      type : 'success',
    })
    if(this.prevUrl.length>0) this.$router.push(this.prevUrl) // tambahkan ini
    this.close()
  }
}
```

Hasilnya adalah ketika login berhasil maka halaman akan diredirect ke prevUrl atau dalam hal ini halaman checkout.

Mari kita coba, 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.

← → ⌂ ⓘ localhost:8081/checkout

Update shipping success CLOSE

Shipping Address

Name
Rosie Cremin 

Address
Jalan Bintaro Raya

Phone
08121621212 

Banten ▼

Tangerang Selatan ▼

 SAVE

hore..

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.

```
// ...  
computed: {  
  ...mapGetters({  
    user: 'auth/user',  
    provinces: 'region/provinces',  
    cities: 'region/cities',  
    // tambahkan ini  
    carts: 'cart/carts',  
    countCart: 'cart/count',  
    totalPrice: 'cart/totalPrice',  
  })  
}
```

```
totalQuantity : 'cart/totalQuantity',
totalWeight   : 'cart/totalWeight',
}),
// ...
```

Lalu pada template, tepat dibawah formulir update alamat pengiriman tambahkan kode berikut untuk menampilkan daftar belanja.

```
<v-subheader>Your Shopping Cart</v-subheader>
<div v-if="countCart>0">
  <v-card flat>
    <v-list two-line>
      <template v-for="item in carts">
        <v-list-tile :key="item.id" avatar>
          <v-list-tile-avatar>
            
          </v-list-tile-avatar>
          <v-list-tile-content>
            <v-list-tile-title v-html="item.title"></v-list-tile-title>
            <v-list-tile-sub-title>
              Rp. {{ item.price.toLocaleString('id-ID') }}  

              ({{ item.quantity }}) item
            </v-list-tile-sub-title>
            <v-list-tile-sub-title>
              <v-divider/>
            </v-list-tile-sub-title>
            </v-list-tile-content>
            <v-list-tile-action>
              <v-list-tile-action-text>{{ item.weight }} kg</v-list-tile-action-
text>
            </v-list-tile-action>
          </v-list-tile>
        </template>
      </v-list>
      <v-container>
        <v-card-actions>
          Subtotal
          <v-spacer />
          Rp. {{ totalPrice.toLocaleString('id-ID') }}
        </v-card-actions>
      </v-container>
    </v-card>
  </div>
```

Berikut ini tampilannya.

← → ⌂ ⓘ localhost:8081/checkout

08121621212		
Province		
City		
SAVE		
Your Shopping Cart		
	Esse quo rem explicabo Rp. 300.000 (1 item)	0.5 kg
Subtotal	Rp. 300.000	

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.

```
public function couriers()
{
    $couriers = [
        ['id'=>'jne', 'text'=> 'JNE'],
        ['id'=>'tiki', 'text'=> 'TIKI'],
        ['id'=>'pos', 'text'=> 'POS'],
    ];

    return response()->json([
        'status' => 'success',
        'message' => 'couriers',
        'data' => $couriers
    ], 200);
}
```

Jangan lupa daftarkan route untuk fungsi ini.

```
Route::get('couriers', 'ShopController@couriers');
```

Mari kita coba endpoint ini <http://larashop-api.test/v1/couriers>, maka responnya sebagai berikut.

```
{
    "status": "success",
    "message": "courier",
    "data": [
        {
            "id": "jne",
            "text": "JNE"
        },
        {
            "id": "tiki",
            "text": "TIKI"
        },
        {
            "id": "pos",
            "text": "POS"
        }
    ]
}
```

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.

```
public function services(Request $request){  
    // Validasi kelengkapan data  
    // 1. data belanja  
    // 2. data courier  
    // 3. data kota pengiriman dari tabel user  
  
    // Validasi data belanja  
    // 1. Cek stok barang  
    // 2. Update data belanja sesuai stok  
  
    // Request data services dari API RajaOngkir  
  
    // Response  
    // 1. Daftar services jika ada  
    // 2. Data belanja yang telah diupdate  
    // 3. Informasi jumlah belanja vs stok  
}
```

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).

```
$this->validate($request, [  
    'courier' => 'required',  
    'carts' => 'required',  
]);  
  
$user = Auth::user();  
if($user){  
    $destination = $user->city_id;  
    if($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.

```
protected function validateCart($carts)
{
    $safe_carts = []; // persiapkan variabel untuk menampung data cart yang aman
    $total = [
        'quantity_before' => 0,
        'quantity' => 0,
        'price' => 0,
        'weight' => 0,
    ];
    $idx = 0;
    // looping data state carts yang dikirim ke server untuk memastikan data valid
    foreach($carts as $cart){
        $id = (int)$cart['id'];
        $quantity = (int)$cart['quantity'];
        $total['quantity_before'] += $quantity;
        $book = Book::find($id); // ambil data buku berdasarkan id-nya
        if($book){
            if($book->stock>0){ // check real stock
                $safe_carts[$idx]['id'] = $book->id;
                $safe_carts[$idx]['title'] = $book->title;
                $safe_carts[$idx]['cover'] = $book->cover;
                $safe_carts[$idx]['price'] = $book->price;
                $safe_carts[$idx]['weight'] = $book->weight;
                if($book->stock < $quantity){ // jika jumlah yang dipesan melebihi stok buku maka
                    $quantity = (int) $book->stock; // jumlah yang dipesan disamakan dengan stok
                }
                $safe_carts[$idx]['quantity'] = $quantity;

                $total['quantity'] += $quantity; // total jumlah yang dipesan dihitung kembali
                $total['price'] += $book->price * $quantity; //total price dihitung kembali
                $total['weight'] += $book->weight * $quantity; // total berat dihitung kembali
                $idx++;
            }
            else{
                continue;
            }
        }
    }
}
```

```

        }
    }

    return [
        'safe_carts' => $safe_carts,
        'total' => $total,
    ];
}

```

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.

```

protected function getServices($data)
{
    $url_cost = "https://api.rajaongkir.com/starter/cost";
    $key="YOUR_API_RAJA_ONGKIR";
    $postdata = http_build_query($data);
    $curl = curl_init();
    curl_setopt_array($curl, [
        CURLOPT_URL => $url_cost,
        CURLOPT_RETURNTRANSFER => true,
        CURLOPT_ENCODING => "",
        CURLOPT_MAXREDIRS => 10,
        CURLOPT_TIMEOUT => 30,
        CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
        CURLOPT_CUSTOMREQUEST => "POST",
        CURLOPT_POSTFIELDS => $postdata,
        CURLOPT_HTTPHEADER => [
            "content-type: application/x-www-form-urlencoded",
            "key: ".$key
        ],
    ]);
    $response = curl_exec($curl);
    $error = curl_error($curl);
    curl_close($curl);
    return [
        'error' => $error,
        'response' => $response,
    ];
}

```

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 diverifikasi, jika berbeda maka akan memunculkan `warning`.

Berikut ini kode lengkap untuk fungsi `services()`.

```

public function services(Request $request)
{
    $status = "error";
    $message = "";
    $data = [];
    // validasi kelengkapan data
    $this->validate($request, [
        'courier' => 'required',
        'carts' => 'required',
    ]);

    $user = Auth::user();
    if($user){
        $destination = $user->city_id;
        if($destination>0){
            // hardcode, silakan sesuaikan dengan asal pengiriman barangnya
            $origin = 153; // Jakarta Selatan
            $courier = $request->courier;
            $carts = $request->carts;
            $carts = json_decode($carts, true); // transformasi dari json
            menjadi array

            // validasi data belanja
            $validCart = $this->validateCart($carts); // panggil fungsi
            validateCart()
            $data['safe_carts'] = $validCart['safe_carts'];
            $data['total'] = $validCart['total'];
            $quantity_different = $data['total']['quantity_before']
            <>$data['total']['quantity'];

            $weight = $validCart['total']['weight'] * 1000;
            if($weight>0){
                // request courier service API RajaOngkir
                $parameter = [
                    "origin"      => $origin,
                    "destination" => $destination,
                    "weight"       => $weight,
                    "courier"     => $courier
                ];
                // check ongkos kirim ke api RajaOngkir melalui fungsi
                getServices()
                $respon_services = $this->getServices($parameter);
                if ($respon_services['error']==null) {
                    $services = [];
                    $response = json_decode($respon_services['response']); // transformasi dari json menjadi array
                    $costs = $response->rajaongkir->results[0]->costs;
                    foreach($costs as $cost){ // parsing ongkos kirimnya
                        $service_name = $cost->service;
                        $service_cost = $cost->cost[0]->value;
                        $service_estimation = str_replace('hari', '', trim($cost->cost[0]->etd));
                        $services[] = [

```

```

'service' => $service_name,
'cost' => $service_cost,
'estimation' => $service_estimation,
'resume' => $service_name . ' [ Rp.
'.number_format($service_cost).', Etd: '.$cost->cost[0]->etd.' day(s) ]'
];
}

// Response
if(count($services)>0){
    $data['services'] = $services;
    $status = "success";
    $message = "getting services success";
}
else{
    $message = "courier services unavailable";
}

// ketika ternyata jumlah beli berbeda dengan jumlah stok maka
tampilkan warninng
if($quantity_different){
    $status = "warning";
    $message = "Check cart data, ".$message;
}
} else {
    $message = "cURL Error #:" . $respon_services['error'];
}
}
else{
    $message = "weight invalid";
}
}
else{
    $message = "destination not set";
}
}
else{
    $message = "user not found";
}

return response()->json([
    'status' => $status,
    'message' => $message,
    'data' => $data
], 200);
}

```

Lalu jangan lupa daftarkan routingnya pada api.php bagian auth route.

```

// ...
// private

```

```
Route::middleware(['auth:api'])->group(function () {
    Route::post('logout', 'AuthController@logout');
    Route::post('shipping', 'ShopController@shipping');
    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
<input checked="" type="checkbox"/> courier	jne	
<input checked="" type="checkbox"/> 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.

```
{
    "status": "success",
    "message": "getting services success",
    "data": {
        "safe_carts": [
            {
                "id": 2,
                "title": "Esse quo rem explicabo",
                "cover": "5912d0c375d5181bb746ba495889bb0d.jpg",
                "price": 300000,
                "weight": 0.5,
                "quantity": 1
            }
        ],
        "total": {
            "quantity_before": 1,
            "quantity": 1,
            "price": 300000,
            "weight": 0.5
        },
        "services": [
            {
                "service": "OKE",
                "cost": 19000,
                "estimation": "2-3",
                "resume": "OKE [ Rp. 19,000, Etd: 2-3 day(s) ]"
            },
            {
                "service": "REG",
                "cost": 22000,
                "estimation": "1-2",
                "resume": "REG [ Rp. 22,000, Etd: 1-2 day(s) ]"
            }
        ]
    }
}
```

```

    },
{
    "service": "YES",
    "cost": 30000,
    "estimation": "1-1",
    "resume": "YES [ Rp. 30,000, Etd: 1-1 day(s) ]"
}
]
}
}

```

Jika jumlah stock berbeda antara yang dibeli dan yang di database maka hasilnya sama kecuali pada status dan message saja.

```
{
    "status": "warning",
    "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.

```

<v-subheader>Courier</v-subheader>
<div>
    <v-card flat>
        <v-container>
            <v-select
                v-model="courier"
                :items="couriers"
                @change="getServices"
                item-text="text"
                item-value="id"
                label="Courier"
                persistent-hint
                single-line
            ></v-select>

```

```

<v-select
    v-model="service"
    v-if="courier"
    :items="services"
    @change="calculateBill"
    item-text="resume"
    item-value="service"
    label="Courier Service"
    persistent-hint
    single-line
></v-select>

<v-card-actions>
    Subtotal
    <v-spacer />
    Rp. {{ shippingCost.toLocaleString('id-ID') }}
</v-card-actions>
</v-container>
</v-card>
</div>

<v-subheader>Total</v-subheader>
<v-card>
<v-container>
<v-layout row wrap>
    <v-flex xs6 text-xs-center>
        Total Bill ({{ totalQuantity }} items)
        <div class="title">{{ totalBill.toLocaleString('id-ID') }}</div>
    </v-flex>
    <v-flex xs6 text-xs-center>
        <v-btn color="orange">
            <v-icon light>attach_money</v-icon> &ampnbsp
            Pay
        </v-btn>
    </v-flex>
</v-layout>
</v-container>
</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.

```

courier: '',
couriers: [],
service: '',
services: [],
shippingCost: 0,
totalBill: 0,

```

Lalu kita perlu memetakan state action `cart/set` untuk mengupdate data belanja agar sesuai dengan database.

```
...mapActions({
  // ...
  setCart : 'cart/set'
}),
```

Kemudian kita buat method `getService()` yang akan merequest endpoint services

```
getServices(){
  let courier = this.courier
  let encodedCart = JSON.stringify(this.carts)
  console.log(encodedCart)
  let formData = new FormData()
  formData.set('courier', this.courier)
  formData.set('carts', encodedCart)

  let config = {
    headers: {
      'Authorization': 'Bearer ' + this.user.api_token,
    },
  }
  this.axios.post('/services', formData, config)
    .then((response) => {
      let response_data = response.data
      // jika tidak error maka data service dan cart akan diupdate.
      if(response_data.status != 'error'){
        this.services = response_data.data.services
        this.setCart(response_data.data.safe_carts)
      }

      this.setAlert({
        status : true,
        text   : response_data.message,
        type   : response_data.status,
      })
    })
    .catch((error) => {
      //console.log(error)
      let responses = error.response
      this.setAlert({
        status : true,
        text   : responses.data.message,
        type   : 'error',
      })
    })
},
```

Lalu kita juga buat method calculateBill() yang akan menghitung total pembayaran.

```
calculateBill(){
    let selectedService = this.services.find((service) => {
        return (service.service==this.service)
    })
    this.shippingCost = selectedService.cost
    this.totalBill = parseInt(this.totalPrice) + parseInt(this.shippingCost)
},
```

Mari kita uji coba.

localhost:8081/checkout

Your Shopping Cart

Esse quo rem explicabo
Rp. 300.000 (1 item)
0.5 kg

Subtotal Rp. 300.000

Courier

JNE

REG [Rp. 22,000, Etd: 1-2 day(s)]

Subtotal Rp. 22.000

Total

Total Bill (1 items)
322.000

\$ 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.

```
public function payment(Request $request){
    // validasi kelengkapan data

    // buat data order

    // buat data detail order

    // check ongkir

    // update data order

    // respon
}
```

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.

```
$this->validate($request, [
    'courier' => 'required',
    'service' => 'required',
    'carts' => 'required',
]);
```

Setelah itu, untuk membuat data order, kita akan menyimpan data awal pemesanan ke dalam record baru pada tabel orders, di mana total bill masih kosong.

```
```php
$order = new Order;
$order->user_id = $user->id;
$order->total_bill = 0;
$order->invoice_number = date('YmdHis');
$order->courier_service = $courier.'-'.$service;
$order->status = 'SUBMIT';
if($order->save()){
 // ...
}
```

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.

```
// perintah ini akan melakukan pengecekan kembali sebagaimana yang
dilakukan fungsi validateCarts()
foreach($carts as $cart){
 $id = (int)$cart['id'];
 // ...
}
```

```

$quantity = (int)$cart['quantity'];
$book = Book::find($id);
if($book){
 if($book->stock>=$quantity){
 $total_price += $book->price * $quantity;
 $total_weight += $book->weight * $quantity;
 // create book order
 $book_order = new BookOrder;
 $book_order->book_id = $book->id;
 $book_order->order_id = $order->id;
 $book_order->quantity = $quantity;
 if($book_order->save()){
 // kurangi stock
 $book->stock = $book->stock - $quantity;
 $book->save();
 }
 }
 else{
 $error++;
 throw new \Exception('Out of stock');
 }
}
else{
 $error++;
 throw new \Exception('Book is not found');
}
}

```

Untuk cek ongkir, kita akan gunakan fungsi `getServices()` yang telah kita buat sebelumnya.

```

$weight = $total_weight * 1000; // to gram
if($weight<=0) {
 $error++;
 throw new \Exception('Weight null');
}
$data = [
 "origin" => $origin,
 "destination" => $destination,
 "weight" => $weight,
 "courier" => $courier
];
$data_cost = $this->getServices($data);
if ($data_cost['error']){
 $error++;
 throw new \Exception('Courier service unavailable');
}

$response = json_decode($data_cost['response']);
$costs = $response->rajaongkir->results[0]->costs;
$service_cost = 0;
foreach($costs as $cost){
 $service_name = $cost->service;
}

```

```

if($service == $service_name){
 $service_cost = $cost->cost[0]->value;
 break;
}
if ($service_cost<=0){
 $error++;
 throw new \Exception('Service cost invalid');
}

```

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.

```

$total_bill = $total_price + $service_cost;
// update total bill order
$order->total_bill = $total_bill;
if($order->save()){
 if($error==0){
 DB::commit();
 $status = 'success';
 $message = 'Transaction success';
 $data = [
 'order_id' => $order->id,
 'total_bill' => $total_bill,
 'invoice_number' => $order->invoice_number,
];
 }
 else{
 $message = 'There are '.$error.' errors';
 }
}

```

Berikut ini kode lengkap untuk fungsi payment.

```

public function payment(Request $request)
{
 $error = 0;
 $status = "error";
 $message = "";
 $data = [];

 $user = Auth::user();
 if ($user) {
 // validasi kelengkapan data
 $this->validate($request, [
 'courier' => 'required',
 'service' => 'required',
 'carts' => 'required',
]);
 }
}

```

```

DB::beginTransaction();
try {
 // prepare data
 $origin = 153; // Jakarta Selatan
 $destination = $user->city_id;
 if($destination<=0) $error++;
 $courier = $request->courier;
 $service = $request->service;
 $carts = json_decode($request->carts, true);

 // create order
 $order = new Order;
 $order->user_id = $user->id;
 $order->total_bill = 0;
 $order->invoice_number = date('YmdHis');
 $order->courier_service = $courier.'-'.$service;
 $order->status = 'SUBMIT';
 if($order->save()){
 $total_price = 0;
 $total_weight = 0;
 foreach($carts as $cart){
 $id = (int)$cart['id'];
 $quantity = (int)$cart['quantity'];
 $book = Book::find($id);
 if($book){
 if($book->stock>=$quantity){
 $total_price += $book->price * $quantity;
 $total_weight += $book->weight * $quantity;
 // create book order
 $book_order = new BookOrder;
 $book_order->book_id = $book->id;
 $book_order->order_id = $order->id;
 $book_order->quantity = $quantity;
 if($book_order->save()){
 // kurangi stock
 $book->stock = $book->stock - $quantity;
 $book->save();
 }
 }
 } else{
 $error++;
 throw new \Exception('Out of stock');
 }
 }
 else{
 $error++;
 throw new \Exception('Book is not found');
 }
 }

 $totalBill = 0;
 $weight = $total_weight * 1000; // to gram
 if($weight<=0) {

```

```

 $error++;
 throw new \Exception('Weight null');
 }
 $data = [
 "origin" => $origin,
 "destination" => $destination,
 "weight" => $weight,
 "courier" => $courier
];
 $data_cost = $this->getServices($data);
 if ($data_cost['error']){
 $error++;
 throw new \Exception('Courier service unavailable');
 }

 $response = json_decode($data_cost['response']);
 $costs = $response->rajaongkir->results[0]->costs;
 $service_cost = 0;
 foreach($costs as $cost){
 $service_name = $cost->service;
 if($service == $service_name){
 $service_cost = $cost->cost[0]->value;
 break;
 }
 }
 if ($service_cost<=0){
 $error++;
 throw new \Exception('Service cost invalid');
 }

 $total_bill = $total_price + $service_cost;
 // update total bill order
 $order->total_bill = $total_bill;
 if($order->save()){
 if($error==0){
 DB::commit();
 $status = 'success';
 $message = 'Transaction success';
 $data = [
 'order_id' => $order->id,
 'total_bill' => $total_bill,
 'invoice_number' => $order->invoice_number,
];
 }
 else{
 $message = 'There are '.$error.' errors';
 }
 }
} catch (\Exception $e) {
 $message = $e->getMessage();
 DB::rollback();
}
}

```

```

else{
 $message = "User not found";
}

return response()->json([
 'status' => $status,
 'message' => $message,
 'data' => $data
], 200);
}

```

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 ▾	[{"id":2,"title":"Esse quo rem explicabo","cover":"5912d0c375d5181bb746ba495889bb0d.jpg","price":30000,"weight":0.5,"quantity":2}]
	Key		Description

Jika stocknya kurang maka akan muncul sebagai berikut.

```
{
 "status": "error",
 "message": "Out of stock",
 "data": []
}
```

Namun jika sukses akan menampilkan respon berikut.

```
{
 "status": "success",
 "message": "Transaction success",
 "data": {
 "order_id": 2,
 "total_bill": 322000,
 "invoice_number": "20180909005030"
 }
}
```

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.

```
<v-subheader>Total</v-subheader>
<v-card>
<v-container>
<v-layout row wrap>
 <v-flex xs6 text-xs-center>
 Total Bill ({{ totalQuantity }} items)
 <div class="title">{{ totalBill.toLocaleString('id-ID') }}</div>
 </v-flex>
 <v-flex xs6 text-xs-center>
 <v-btn color="orange" @click="dialogConfirm=true"
:disabled="totalBill==0">
 <v-icon light>attach_money</v-icon> &nbsp
 Pay
 </v-btn>
 </v-flex>
</v-layout>
</v-container>
</v-card>

<template>
<v-layout row justify-center>
 <v-dialog v-model="dialogConfirm" persistent max-width="290">
 <v-card>
 <v-card-title class="headline">Confirmation!</v-card-title>
 <v-card-text>If You continue, transaction will be processed</v-
card-text>
 <v-card-actions>
 <v-spacer></v-spacer>
 <v-btn color="green darken-1" flat @click="cancel">Cancel</v-btn>
 <v-btn color="green darken-1" flat @click="pay">Continue</v-btn>
 </v-card-actions>
 </v-card>
 </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`.

```
pay(){
 this.dialogConfirm = false
```

```

let courier = this.courier
let service = this.service
let safeCart = JSON.stringify(this.carts)
let formData = new FormData()
formData.set('courier', this.courier)
formData.set('service', this.service)
formData.set('carts', safeCart);
let config = {
 headers: {
 'Authorization': 'Bearer ' + this.user.api_token,
 },
}
this.axios.post('/payment', formData, config)
 .then((response) => {
 let response_data = response.data
 if(response_data && response_data.status=='success'){
 this.$router.push({path: "/payment"})
 this.setCart([])
 }

 this.setAlert({
 status : true,
 text : response_data.message,
 type : response_data.status,
 })
 })
 .catch((error) => {
 let responses = error.response
 this.setAlert({
 status : true,
 text : responses.data.message,
 type : 'error',
 })
 })
},
cancel(){
 this.dialogConfirm = false
}

```

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.

```

this.$router.push({path: "/payment"})
this.setCart([])

```

Mari kita coba.

Pada halaman checkout, klik tombol Pay



Subtotal Rp. 300.000

Courier

JNE

REG [ Rp. 22,000, Etd: 1-2 day(s) ]

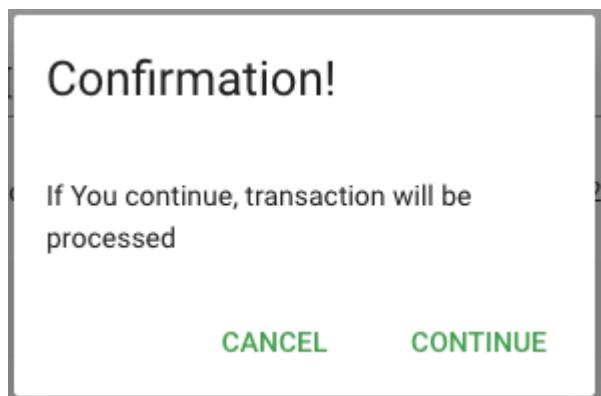
Subtotal Rp. 22.000

Total

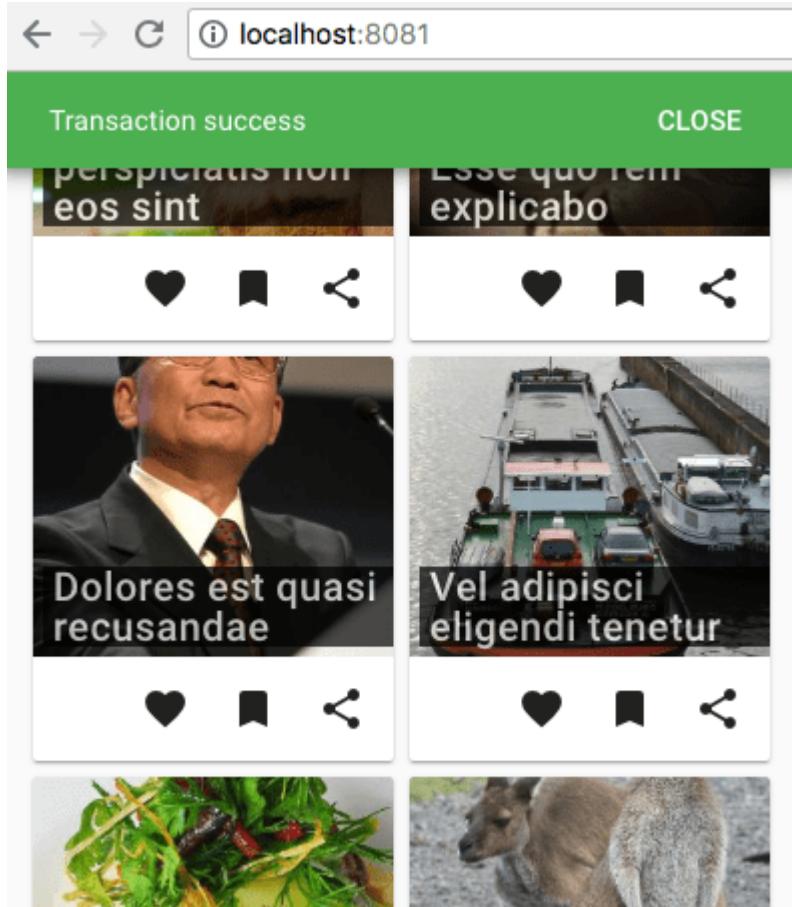
Total Bill (1 items)  
**322.000**

\$ PAY

Muncul dialog konfirmasi, klik Continue



Muncul pesan bahwa transaksi berhasil dan halaman diredirect ke halaman payment, namun karena belum kita definisikan routingnya maka akan diredirect ke halaman utama.



Oke, kita sebenarnya perlu state tambahan untuk menyimpan data transaksi berhasil hasil dari respon endpoint payment.

```
{
 "status": "success",
 "message": "Transaction success",
 "data": {
 "order_id": 2,
 "total_bill": 322000,
 "invoice_number": "20180909005030"
 }
}
```

Tujuannya adalah agar data ini bisa kita tampilkan pada halaman pembayaran.

Buka src/store.js, tambahkan state payment, termasuk mutation, action dan getternya.

```
state: {
 // state lain
 payment: []
},
mutations: {
 // mutation lain
 setPayment: (state, value) => {
 state.payment = value
 }
}
```

```

},
},
actions: {
 // action lain
 setPayment: ({commit}, value) => {
 commit('setPayment', value)
 },
},
getters: {
 // getter lain
 payment: state => state.payment,
},

```

Kemudian balik ke component Checkout, petakan action setPayment `setPayment : 'setPayment'`. Lalu gunakan untuk menyimpan respon data jika transaksi sukses.

```

if(response_data && response_data.status=='success'){
 this.setPayment(response_data.data) // <= ini
 this.$router.push({path: "/payment"})
 this.setCart([])
}

```

## 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`

```

<template>
 <div class="payment">
 <v-subheader>Payment Information</v-subheader>
 <v-card flat>
 <v-container>
 <table class="v-datatable v-table">
 <tbody>
 <tr><th class="column text-xs-left">Order ID</th><td>{{ payment.order_id }}</td></tr>
 <tr><th class="column text-xs-left">Invoice Number</th><td>{{ payment.invoice_number }}</td></tr>
 <tr><th class="column text-xs-left">Total Bill</th><td>Rp. {{ payment.total_bill.toLocaleString('id-ID') }}</td></tr>
 </tbody>
 </table>
 </v-container>
 </v-card>

 <v-subheader>Transfer To</v-subheader>
 <v-card flat>

```

```

<v-container>
 <table class="v-datatable v-table">
 <tbody>
 <tr>
 <td></td>
 <td>BCA KCP abc No Rek 123</td>
 </tr>
 <tr>
 <td></td>
 <td>BANK MANDIRI KCP xyz No Rek 456</td>
 </tr>
 </tbody>
 </table>
</v-container>
</v-card>

<v-subheader></v-subheader>
<v-card>
 <v-container>
 <v-layout row wrap>
 <v-flex xs12 text-xs-center>
 <v-btn color="success" @click="finish">
 Finish
 </v-btn>
 </v-flex>
 </v-layout>
 </v-container>
</v-card>
</div>
</template>
<script>
 import { mapGetters, mapActions } from 'vuex'
 export default {
 computed: {
 ...mapGetters({
 payment : 'payment',
 }),
 },
 methods: {
 finish(){
 this.$router.push('/')
 }
 },
 }
</script>

```

Daftarkan pada routing

```
{
 path: '/payment',
 name: 'payment',
 component: () => import(/* webpackChunkName: payment */
```

```
'./views/Payment.vue')
meta: { auth: true }
},
```

Saatnya kita uji coba.

localhost:8081/payment

Vueshop

Payment Information

Order ID	7
Invoice Number	20180909013442
Total Bill	Rp. 322.000

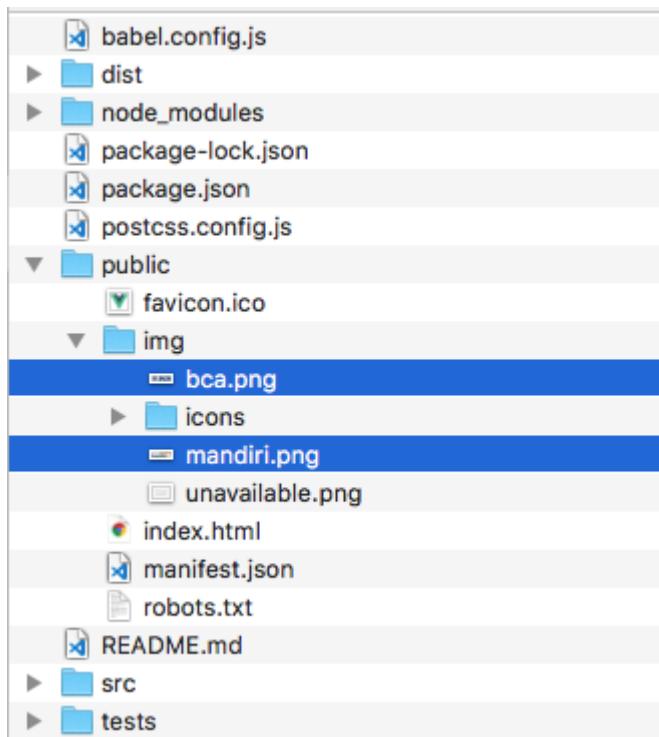
Transfer To

**BCA**  
BCA KCP abc No Rek  
123

**mandiri**  
BANK MANDIRI KCP xyz  
No Rek 456

**FINISH**

Catatan: data bank pada contoh ini masih hardcode, kamu perlu siapkan image logo bank dan letakkan pada folder `public/img`



## 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/components/CSideBar.vue`, pada properti data, tambahkan menu baru pada data items.

```
items: [
 { title: 'Home', icon: 'dashboard', route: 'home' },
 { title: 'Profile', icon: 'person', route: 'profile', auth: true },
 { title: 'About', icon: 'question_answer', route: 'about' },
]
```

Flag `auth: true` untuk membedakan menu yang buat publik atau menu untuk user yang sudah login.

Lalu pada templatenya ubah menjadi sebagai berikut.

```
<v-list-item
 v-for="(item,index) in items"
 :key="index"
 :href="item.route"
 :to="{name: item.route}"
 v-if="!item.auth || (item.auth && !guest)"
>
<v-list-item-action>
 <v-icon>{{ item.icon }}</v-icon>
</v-list-item-action>
```

```
<v-list-tile>
 <v-list-tile-content>{{ item.title }}</v-list-tile-content>
</v-list-tile>
```

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

## Membuat Component Profile

Component ini hanya akan menampilkan profile dari user yang sudah login. Simpan pada file `/src/components/Profile.vue`

```
<template>
 <div>
 <v-container>
 <ul v-for="(value, key) in user" :key="key">
 {{ key }}: {{ value }}

 </v-container>
 </div>
</template>
<script>
import { mapGetters } from 'vuex'
export default {
 computed: {
 ...mapGetters({
 'user':'auth/user'
 })
 }
}
</script>
```

Jangan lupa pada `src/router.js` tambahkan routing untuk component profile.

```
{
 path: '/profile',
 name: 'profile',
 component: () => import(/* webpackChunkName: "profile" */ './views/Profile.vue'),
 meta: { auth: true }
},
```

Mari kita test.

The screenshot shows a user profile page for 'Rosie Cremin' on 'Vueshop'. The page has a blue header with the title 'Vueshop' and a shopping cart icon. Below the header is a list of user details:

- id: 1
- name: Rosie Cremin
- email: mayer.glennie@example.com
- email\_verified\_at:
- created\_at: 2018-09-08 06:14:28
- updated\_at: 2018-09-09 02:01:50
- roles: ["CUSTOMER"]
- address: Jalan Bintaro Raya
- city\_id: 114
- province\_id: 1
- phone: 08121621212
- avatar: 49492abfed22947cefca2773b6344beb.jpg
- status: ACTIVE
- api\_token: krY0rL83wdRopkqjNEdgtjWxAnqQh7ELeqvGSsliU6j8z5CN

Yap berhasil, silakan dikembangkan menjadi misalnya form update profile dsb.

## 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.

```
public function myOrder(Request $request)
{
 $user = Auth::user();
 $status = "error";
 $message = "";
 $data = [];
 if($user){
 $orders = \App\Order::select('*')
 ->where('user_id', '=', $user->id)
 ->orderBy('id', 'DESC')
 ->get();

 $status = "success";
 $message = "data my order ";
 }
}
```

```
$data = $orders;
}
else{
 $message = "User not found";
}

return response()->json([
 'status' => $status,
 'message' => $message,
 'data' => $data
, 200);
}
```

Jangan lupa daftarkan routing myOrder pada api.php

```
// route lain

// private
Route::middleware(['auth:api'])->group(function () {
 // ... route lain
 Route::get('my-order', 'ShopController@myOrder'); // <= ini
 //...
});
```

Silakan dicoba.

Akses URL ini <http://larashop-api.test/v1/my-order> menggunakan postman method GET dan authentication bearer token, maka hasilnya.

```
{
 "status": "success",
 "message": "data my order ",
 "data": [
 {
 "id": 8,
 "user_id": 1,
 "total_bill": 322000,
 "invoice_number": "20180909021916",
 "courier_service": "jne-REG",
 "status": "SUBMIT",
 "created_at": "2018-09-09 02:19:16",
 "updated_at": "2018-09-09 02:19:17"
 }
]
}
```

Link My Order di SideBar

Buka file `src/components/CSideBar.vue`, pada properti data, tambahkan menu baru pada data items.

```
items: [
 { title: 'Home', icon: 'dashboard', route: 'home' },
 { title: 'Profile', icon: 'person', route: 'profile', auth: true },
 { title: 'My Order', icon: 'shop_two', route: 'my-order', auth: true },
 { title: 'About', icon: 'question_answer', route: 'about' },
]
```

## Membuat Component My Order

Component ini hanya akan menampilkan histori pemesanan dari user yang sudah login. Simpan pada file `/src/components/MyOrder.vue`

```
<template>
<div>
 <v-container>
 <v-list two-line v-if="items.length>0">
 <template v-for="item in items">
 <v-list-tile :key="item.id">
 <v-list-tile-content>
 <v-list-tile-title>INVOICE: {{ item.invoice_number }}</v-
list-tile-title>
 <v-list-tile-sub-title class="text--primary">Rp. {{
item.total_bill.toLocaleString('id-ID') }}</v-list-tile-sub-title>
 <v-list-tile-sub-title>date: {{ item.updated_at }}. courier:
{{ item.courier_service }}</v-list-tile-sub-title>
 </v-list-tile-content>
 <v-list-tile-action>
 <v-list-tile-action-text>{{ item.status }}</v-list-tile-
action-text>
 </v-list-tile-action>
 </v-list-tile>
 <v-divider :key="'div_'+item.id"></v-divider>
 </template>
 </v-list>
 </v-container>
</div>
</template>
<script>
 import { mapGetters, mapActions } from 'vuex'
 export default {
 data () {
 return {
 items: [],
 }
 },
 computed: {
 ...mapGetters({
 user : 'auth/user',

```

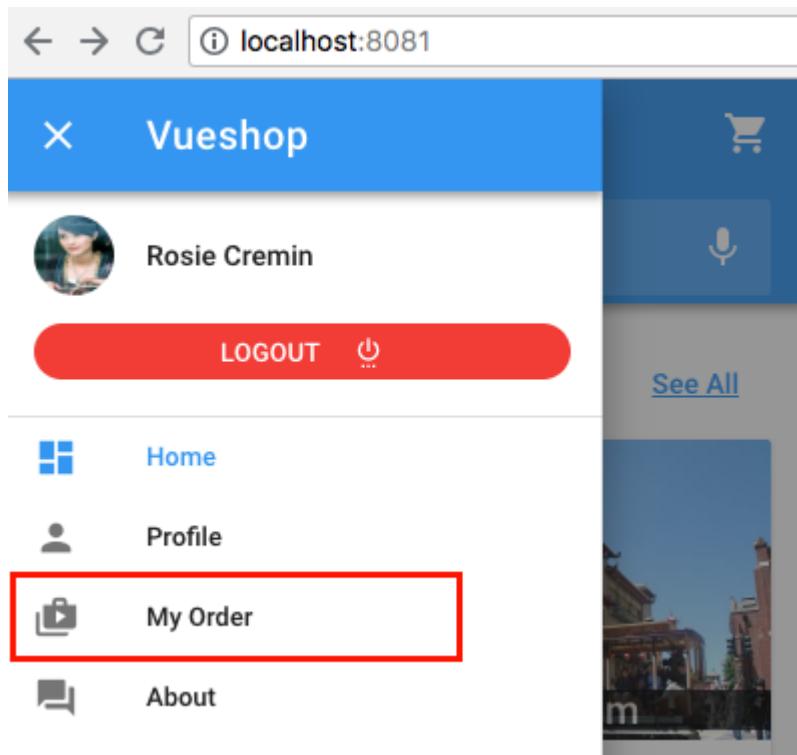
```
 })
 },
methods: {
 ...mapActions({
 setAlert : 'alert/set',
 }),
},
mounted(){
 let config = {
 headers: {
 'Authorization': 'Bearer ' + this.user.api_token,
 },
 }
 this.axios.get('/my-order',config)
 .then((response) => {
 let orders = response.data.data
 this.items = orders
 })
 .catch((error) => {
 let responses = error.response
 this.setAlert({
 status : true,
 text : responses.data.message,
 type : 'error',
 })
 })
 }
}
</script>
```

Jangan lupa pada `src/router.js` tambahkan routing untuk component MyOrder.

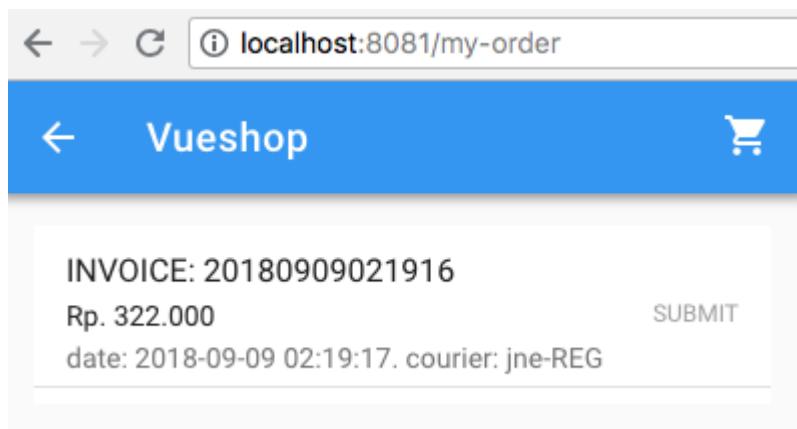
```
{
 path: '/my-order',
 name: 'my-order',
 component: () => import(/* webpackChunkName: "my-order" */ './views/MyOrder.vue'),
 meta: { auth: true }
},
```

Mari kita test.

Pada sidebar, pilih menu my order.



Hasilnya.



Yes selesai.

## 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 dapati 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.

```
VUE_APP_NAME=Vueshop
VUE_APP_BACKEND_URL=https://api.vueshop.id
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
```

```
vueshop — -bash — 82x49
[Hafids-MacBook-Pro:vueshop hafidmukhlasin$ npm run build
> vueshop@0.1.0 build /Users/hafidmukhlasin/Dev/vue-projects/vueshop
> vue-cli-service build

 Building for production...

 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.09cb4b6028eaae2 1.99 kb 0.69 kb
e21658e63157c5fba.js
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` itu lah 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](#).

```
APP_DEBUG=false
```

### Sesuaikan Konfigurasi Database

Sesuaikan konfigurasi untuk koneksi database dengan server production. Adapun pengaturannya juga pada file .env.

```
DB_CONNECTION=mysql
DB_HOST=mysql
DB_PORT=3306
DB_DATABASE=database
DB_USERNAME=username
DB_PASSWORD=password
```

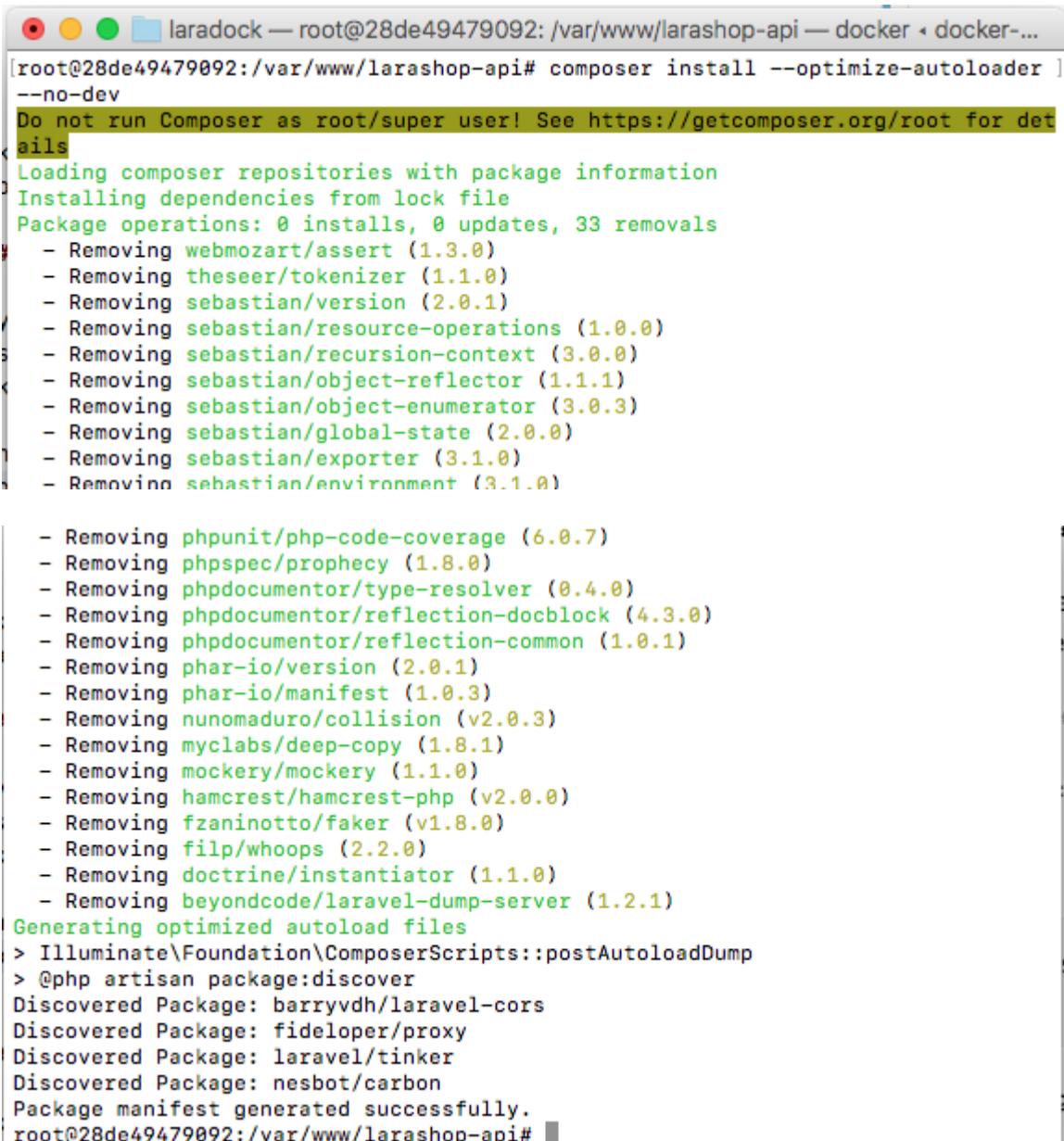
## Optimization

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.

```
composer install --optimize-autoloader --no-dev
```



```

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 fzaninotto/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.

```
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.

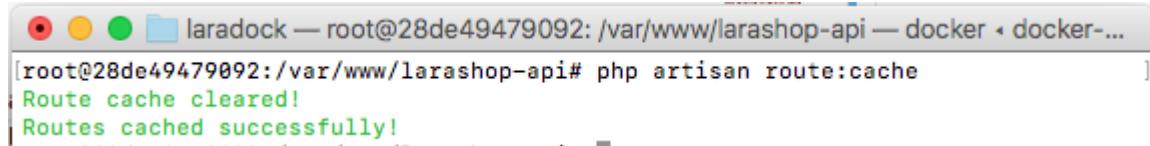
Nah hanya saja apabila ini kita lakukan di development maka akan menyebabkan error karena penggunaan absolute path pada config yang bisa jadi berbeda dengan path di server production. Jika terlanjur maka gunakan perintah berikut.

```
php artisan config:clear
```

### Optimizing Route Loading

Jika kita mengembangkan aplikasi dengan routing yang complex maka sebaiknya kita menjalankan perintah berikut sebelum deployment.

```
php artisan route:cache
```



```
laradock — root@28de49479092: /var/www/larashop-api — docker ↵ docker-...
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.

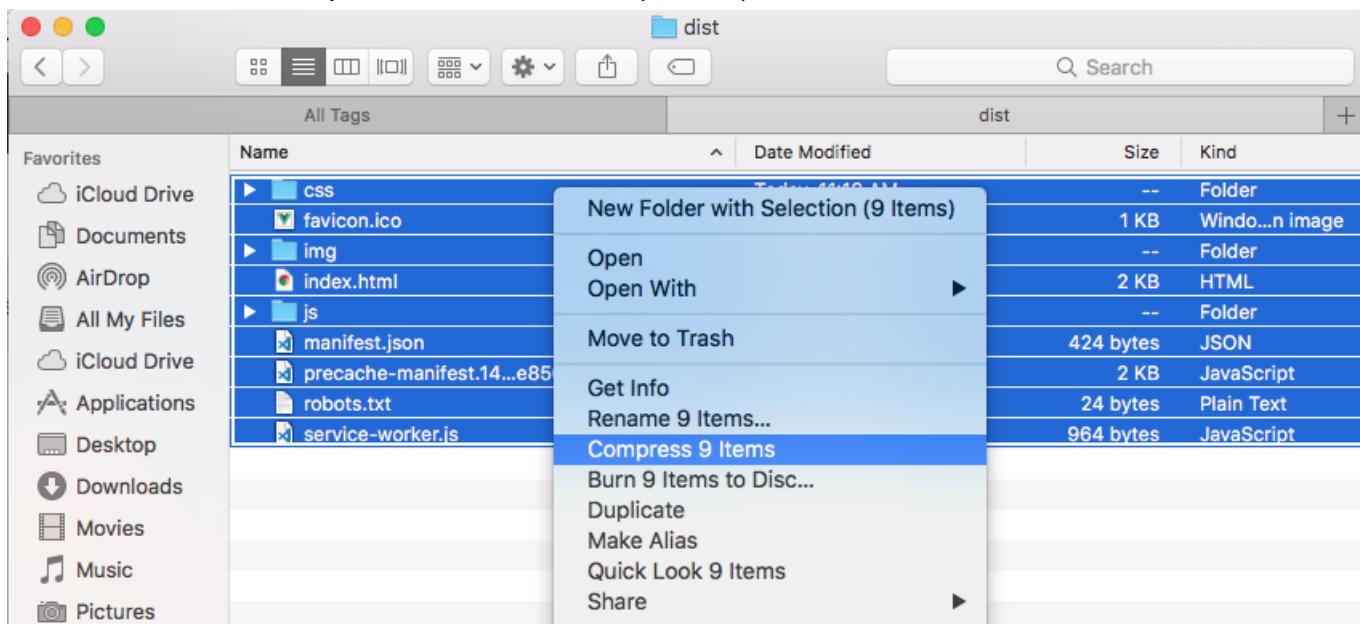
## Proses Deployment

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 tinggal 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.

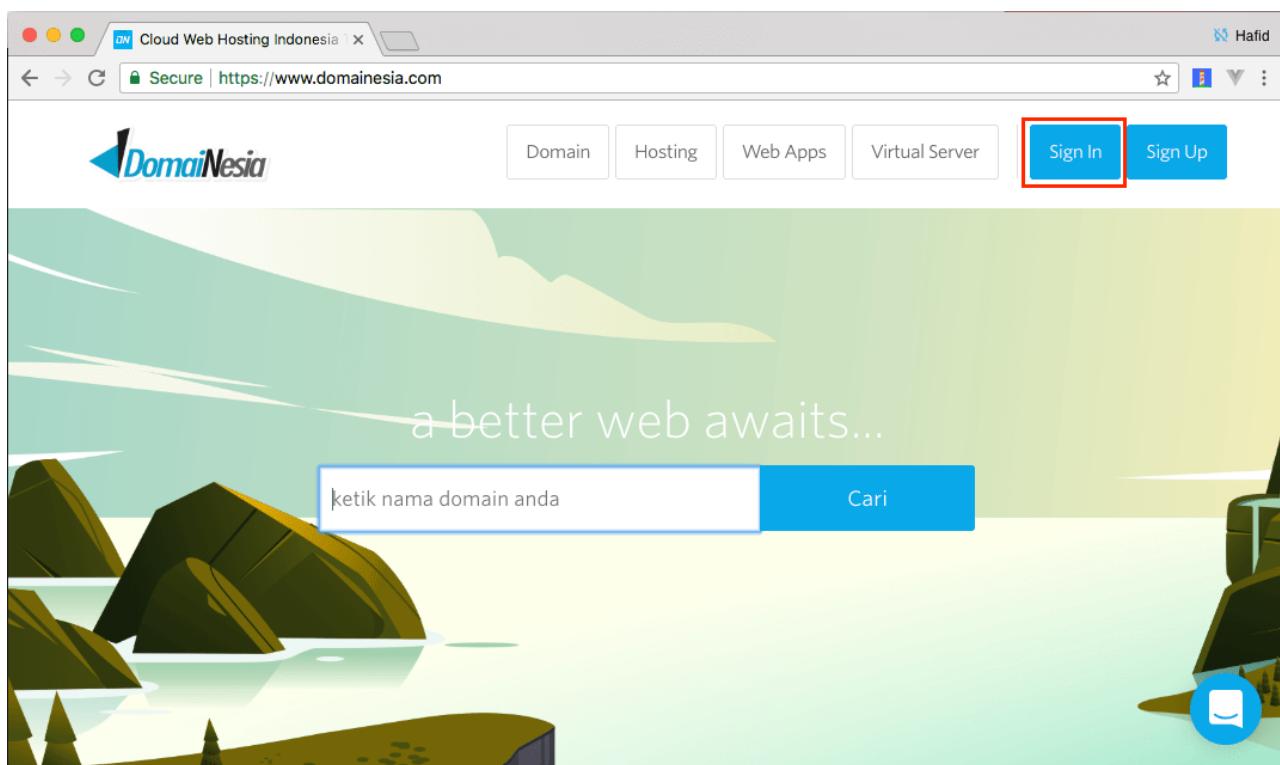
### Deployment Aplikasi Web Frontend

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.

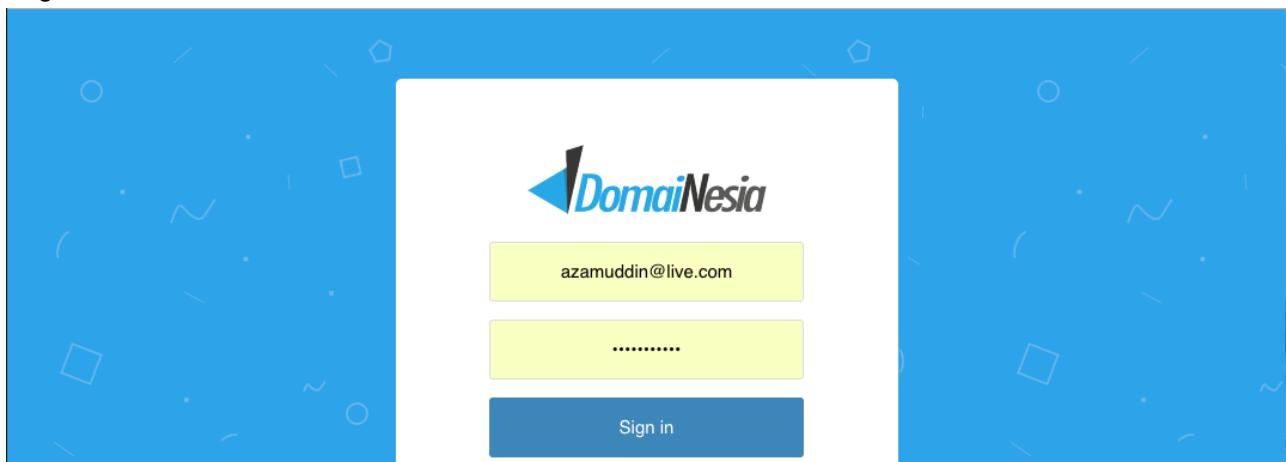


Pada contoh ini, kita menggunakan domain [vueshop.id](https://vueshop.id) dan aplikasi web frontend akan diunggah ke root dari vueshop.id (public\_html).

### 1. Buka domainesia



2. Login



3. Pada halaman dahsboard, pilih products

Hello, Muhammad!

Account Overview

1 Active Products	6 Active Domains	0 Unpaid Invoices	0 Active Affiliate
-------------------	------------------	-------------------	--------------------

Epicbox 0 [View Epicbox](#)

4. Pada halaman dashboard products, pilih hostingnya yaitu pada contoh ini Vueshop

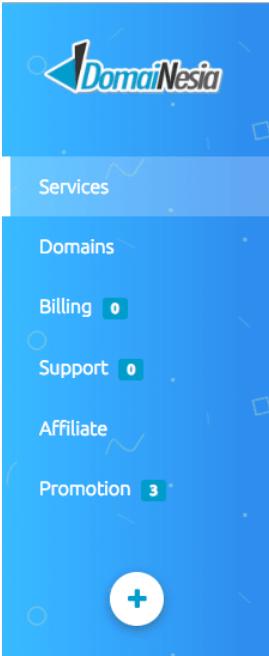
My Products & Services

Hosting

<input type="checkbox"/> ● Hosting Super (ID) - vueshop.id	Expiration
31/08/2019	

[Renew Hosting](#)

5. Pada halaman dashboard Vueshop, pilih file manager



Vueshop.id | Hosting Super (ID)
[profile](#)
[Switch Plan](#)

Overview
Access
Upgrade
Addons

#### Quick Shortcuts

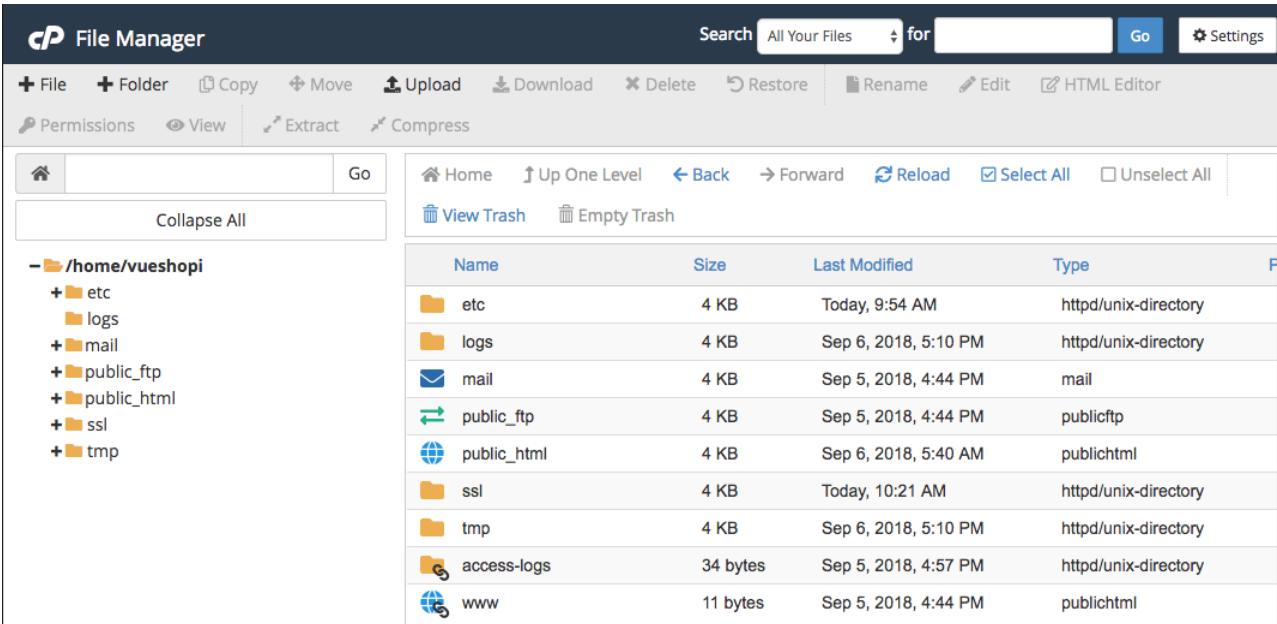
- [!\[\]\(40ec9f394acb5a1d60d07793bdce7e2f\_img.jpg\) Email Accounts](#)
- [!\[\]\(41a53a6864eea15d55d344d612b7945b\_img.jpg\) Forwarders](#)
- [!\[\]\(da581e3277a99d52195dadde72490bd7\_img.jpg\) Autoresponders](#)
- [!\[\]\(d9b766e7328457d8f4c3aa24b4aff8c3\_img.jpg\) File Manager](#)
- [!\[\]\(e26d810b5f6ed45b84440de5b9f38b0f\_img.jpg\) Backup](#)
- [!\[\]\(2fc61613ea31ed006c3f1c53b37f4435\_img.jpg\) Subdomains](#)
- [!\[\]\(f2533a40aece2c71a5c1218c0bbdb120\_img.jpg\) Addon Domains](#)
- [!\[\]\(ed0dd2361ba6b3b006d063983be63df5\_img.jpg\) Cron Jobs](#)
- [!\[\]\(bfd68948e19d6b0c1cee51c7f0f72e18\_img.jpg\) MySQL Databases](#)
- [!\[\]\(6b81607818f130c29a6ae6bcbd730c8c\_img.jpg\) phpMyAdmin](#)
- [!\[\]\(b79eef44604e7be513ca6ca8fa191d78\_img.jpg\) Awstats](#)

#### Usage Statistics

Disk Usage	Bandwidth Usage
 0 2 M / 2048 M	 0 0 M / Unlimited M

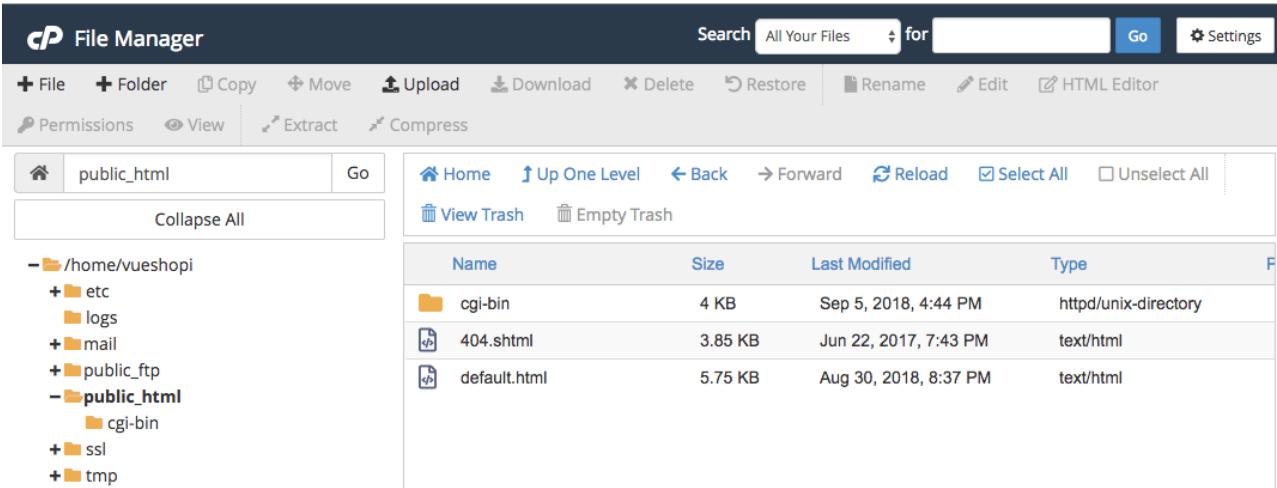
Last Updated 09/09/2018 (01:36)

6. Maka kita akan dibawah ke halaman file manager. Pilih public\_html yaitu folder root web.



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	publithml
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	publithml

7. Pada folder public\_html ini, pilih menu Upload untuk mengunggah file aplikasi web frontend



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.

The screenshot shows a 'File Upload' interface. At the top, it says 'Select the file you want to upload to "/home/vueshopi/public\_html"'. Below this is a message: 'Maximum file size allowed for upload: 1.99 GB'. There is a checkbox for 'Overwrite existing files'. A dashed box contains the text 'Drop files here to start uploading' and 'or' below it, followed by a 'Select File' button. At the bottom, there is a link to 'Go Back to "/home/vueshopi/public\_html"'.

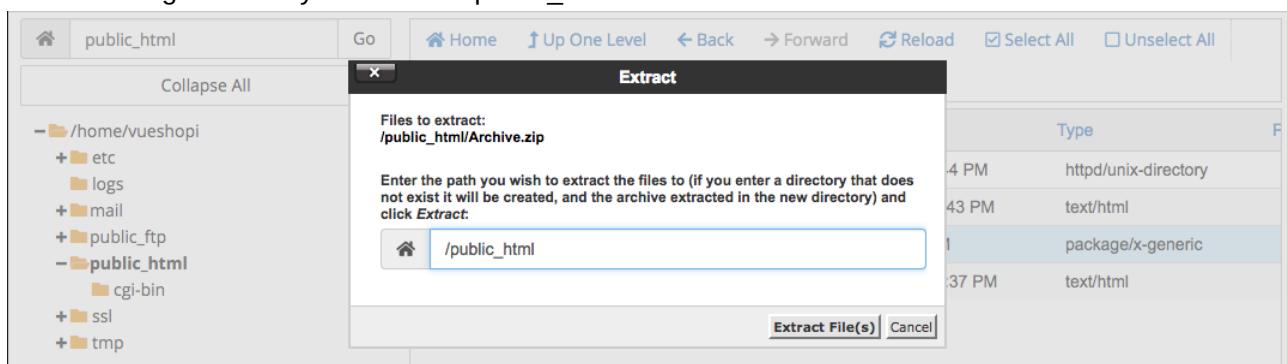
9. Jika sudah selesai, klik Go Back.

The screenshot shows a 'File Upload' interface, identical to the previous one but with a file named 'Archive.zip' being uploaded. A green progress bar indicates '100%' completion. Below the bar, it says '816.26 KB complete'. At the bottom, there is a link to 'Go Back to "/home/vueshopi/public\_html"'.

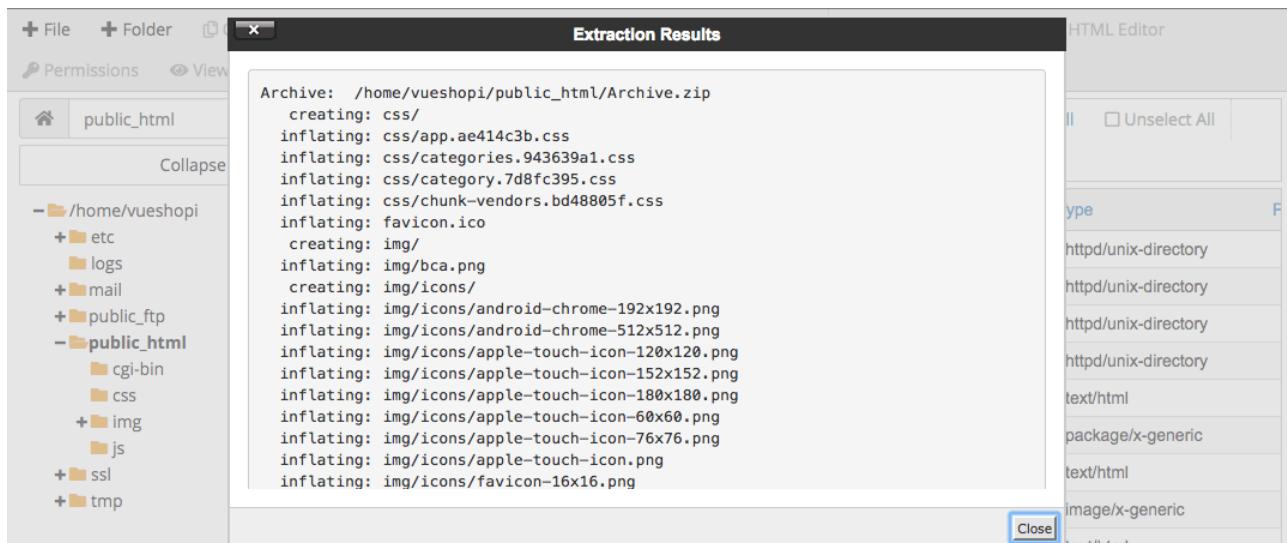
10. Pada file manager, pilih file archive hasil unggahan kita, kemudian extract.

The screenshot shows a 'File Manager' interface. The left sidebar shows a tree view of the directory structure under 'public\_html': '/home/vueshopi' (with 'etc', 'logs', 'mail', 'public\_ftp'), 'public\_html' (with 'cgi-bin'), and 'cc1'. The main area shows a list of files and folders: 'cgi-bin' (4 KB, httpd/unix-directory), '404.shtml' (3.85 KB, text/html), 'Archive.zip' (816.26 KB, package/x-generic), and 'default.html' (5.75 KB, text/html). At the top, there are buttons for File, Folder, Copy, Move, Upload, Download, Delete, Rename, Edit, and HTML Editor. There is also a search bar and settings icon.

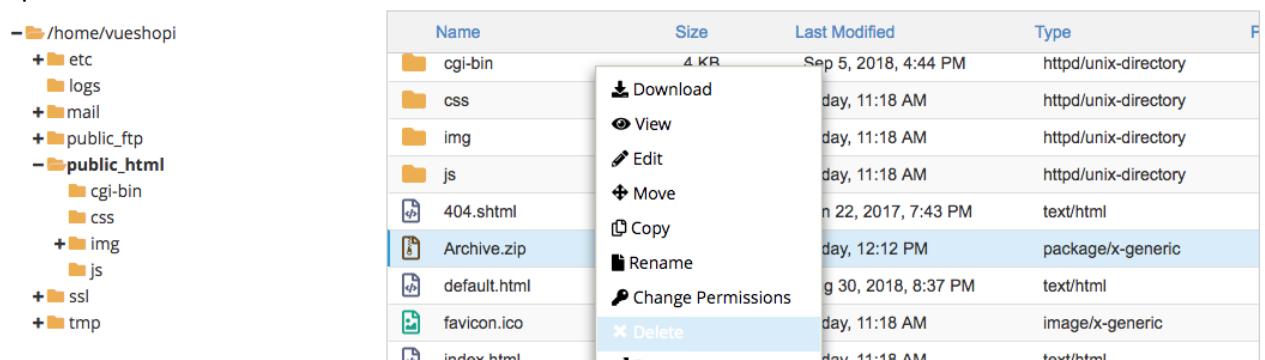
11. Tentukan target extract yaitu di folder public\_html



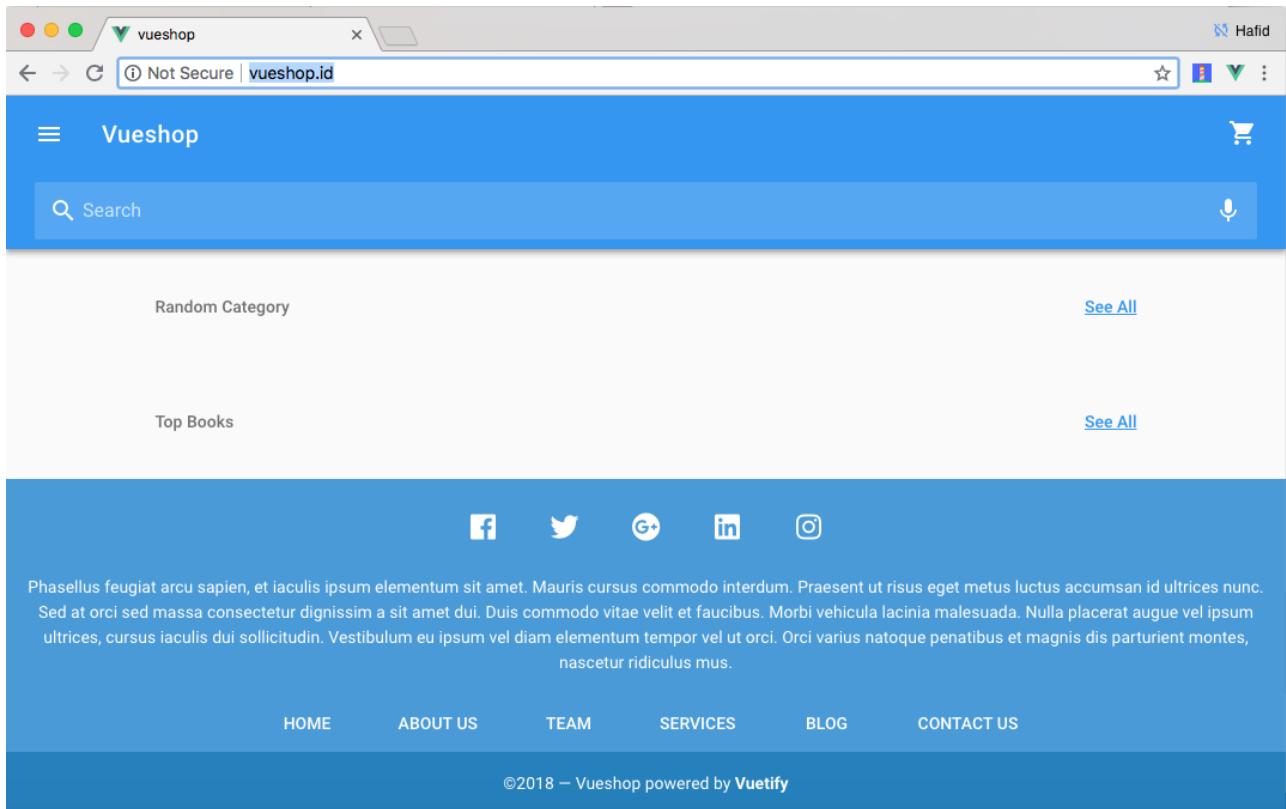
12. Jika selesai, klik Close.



13. Hapus file archive.



14. Aplikasi telah siap diakses. Silakan diuji coba.



Mudah sekali bukan? Ya karena cuman 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.

```
RewriteEngine On
RewriteCond %{HTTPS} !=on
RewriteRule ^(.*)$ https:// %{HTTP_HOST}%{REQUEST_URI} [L,R=301]
```

Pada public\_html, tambahkan file .htaccess

The screenshot shows the cPanel File Manager interface. In the top navigation bar, the 'File Manager' tab is selected. Below it, there are various file operations: + File (highlighted with a red box), + Folder, Copy, Move, Upload, Download, Delete, Restore, Rename, Edit, and HTML Editor. Underneath the toolbar, there are 'Permissions' and 'View' buttons. The left sidebar shows the directory structure: public\_html (selected), .cpanel, .cphorde, .htpasswd, .softaculous, and .snamassassin. The main area shows a 'New File' dialog box. The 'New File Name:' field contains '.htaccess' (highlighted with a red box, labeled '2'). The 'New file will be created in:' dropdown shows '/public\_html' (highlighted with a red box, labeled '3'). At the bottom right of the dialog are 'Create New File' and 'Cancel' buttons (highlighted with a red box, labeled '4'). To the right of the dialog, there is a list of files with their types: httpd/unix-directory for .cpanel, .cphorde, .htpasswd, .softaculous, and .snamassassin.

Edit file tersebut.

The screenshot shows the cPanel File Manager interface. In the left sidebar, the 'public\_html' directory is selected. In the main content area, a list of files and folders is shown. The '.htaccess' file is highlighted with a red box and selected for editing. The top navigation bar includes buttons for File, Folder, Copy, Move, Upload, Download, Delete, Restore, Rename, Edit (which is highlighted with a red box), and HTML Editor.

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 window. The 'Editing' field shows the path '/home/vueshopi/public\_h'. The 'Encoding' field is set to 'utf-8'. The 'Save Changes' button is visible. The editor contains the following configuration:

```

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.

```

server {
 listen 80;
 server_name vueshop.id;
 rewrite ^ https://$server_name$request_uri? permanent;
}

```

## 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](http://api.vueshop.id)

### Membuat Sub Domain

Pada contoh ini, kita akan menggunakan sub domain [api.vueshop.id](http://api.vueshop.id). Oleh karena itu, kita perlu membuatnya terlebih dahulu.

- Pada dashboard vueshop, pilih Sub Domain.

The screenshot shows the DomaiNesia Vueshop dashboard. On the left sidebar, there are sections for Services, Domains, Billing (0), Support (0), Affiliate, and Promotion (3). The main area is titled "Vueshop.id | Hosting Super (ID)". It has tabs for Overview, Access, Upgrade, and Addons. Under "Quick Shortcuts", there are icons for Email Accounts, Forwarders, Autoresponders, File Manager, Backup, Subdomains (which is highlighted with a red box), Addon Domains, and Cron Jobs. To the right, there are "Usage Statistics" for Disk Usage (0 / 2048 M) and Bandwidth Usage (0 / Unlimited M), with a note that it was last updated on 09/09/2018 at 01:36.

- 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 "api" and is highlighted with a red box. The "Domain" dropdown is set to "vueshop.id". The "Document Root" field shows "/api.vueshop.id". A large blue "Create" button is at the bottom, also highlighted with a red box.

- Setelah sub domain berhasil dibuat, klik link Go Back.

The screenshot shows the "Subdomains" list. A green success message at the top states "Success: "api.vueshop.id" has been created." Below it is a "Go Back" button.

4. Balik ke cpanel melalui tombol di sisi kiri atas.

The screenshot shows the DomaiNesia cPanel interface. On the left sidebar, there is a 'Create' button and an icon for creating new subdomains, which is highlighted with a red box. The main area displays a table titled 'Modify a Subdomain' with one entry: 'api.vueshop.id' with a document root of '/api.vueshop.id'. There are buttons for 'Remove' and 'Manage Redirection'. Below the table are pagination controls for 'Page Size' (set to 10) and navigation arrows. The top right corner shows user information and a 'LOGOUT' link.

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 with the 'DATABASES' section selected. On the left sidebar, there are icons for phpMyAdmin, MySQL® Database Wizard, PostgreSQL Databases, and phpPgAdmin. In the main area, there are links for 'MySQL® Databases', 'Remote MySQL®', and 'PostgreSQL Database Wizard'. The 'MySQL® Databases' link is highlighted with a red box. To the right, there is a sidebar with various system statistics: 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.

The screenshot shows the MySQL Databases section of the DomaiNesia control panel. A sidebar on the left has icons for databases and users. The main area title is "MySQL® Databases". Below it, a text block says: "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](#)". A blue link "Jump to MySQL Users" is at the bottom right. A form titled "New Database:" contains the input "vueshopi\_larashop" and a "Create Database" button.

3. Jika berhasil maka klik link Go Back.

The screenshot shows the MySQL Databases section again. A green success message box says "Added the database "vueshopi\_larashop"" with a checkmark icon. A "Go Back" button is at the bottom right.

4. Scrolldown, buat User database, masukkan username dan password. Lalu klik create user.

The screenshot shows the "Add New User" page. It has fields for "Username" (vueshopi\_app), "Password" (redacted), "Password (Again)" (redacted), and "Strength" (Very Strong (97/100)). A "Create User" button is at the bottom right. A "Password Generator" button is also present.

5. Jika berhasil, maka klik link Go Back.

The screenshot shows the MySQL Databases section again. A green success message box says "You have successfully created a MySQL user named "vueshopi\_app"" with a checkmark icon. A "Go Back" button is at the bottom right.

6. Scroll down, tambahkan user yang telah kita buat pada database, supaya user tersebut dapat berinteraksi dengan database. lalu klik add.

The screenshot shows the 'Add User To Database' section of the DomaiNesia control panel. It has two dropdown menus: 'User' set to 'vueshopi\_app' and 'Database' set to 'vueshopi\_larashop'. A blue 'Add' button is at the bottom.

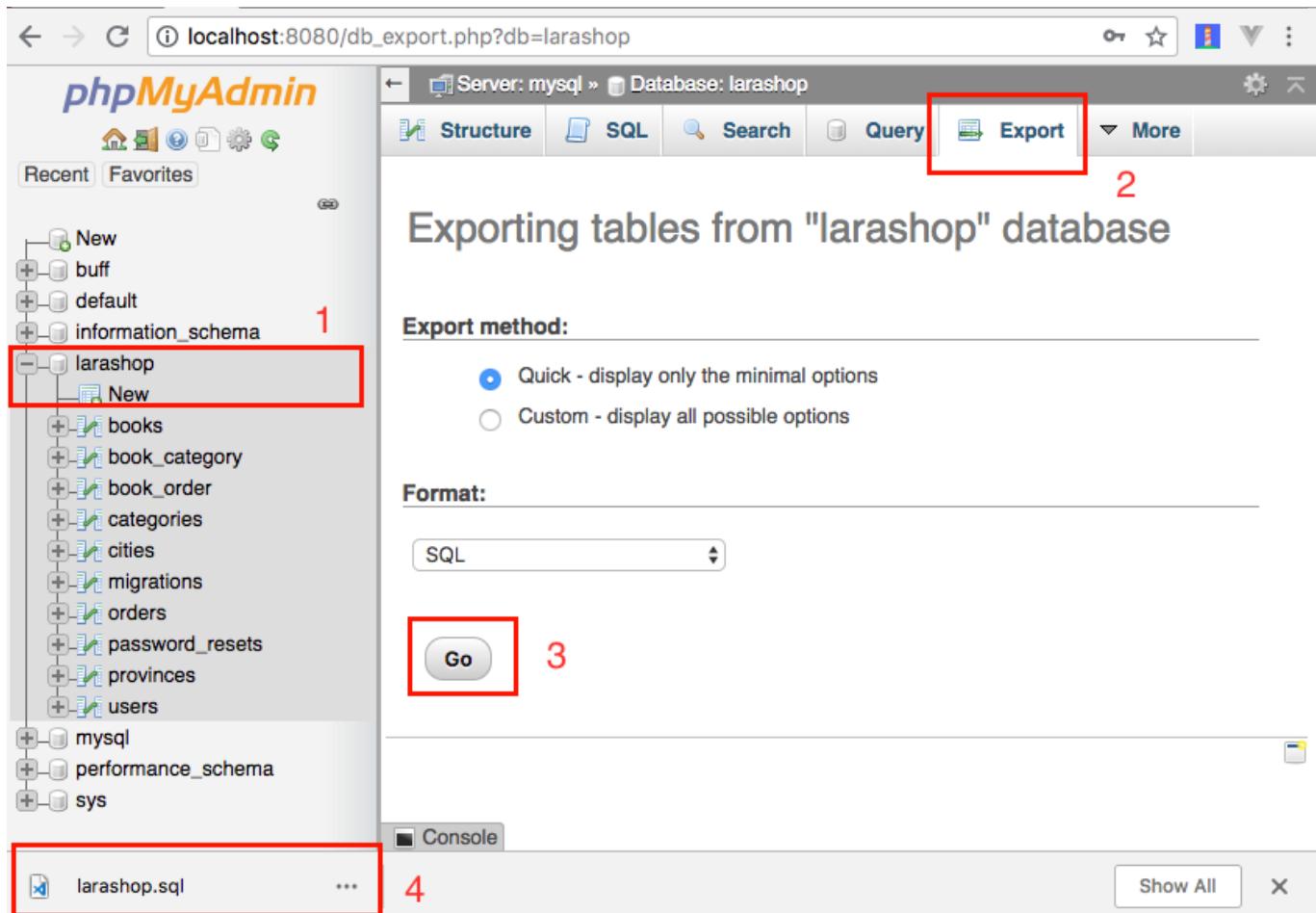
7. Tentukan privilege atau hak akses user tersebut pada database. Sebaiknya kamu punya minimal dua jenis user, user admin bisa semua dan user aplikasi yang hanya bisa select, insert, update dan delete saja.

The screenshot shows the 'Manage User Privileges' page for the user 'vueshopi\_app' on the database 'vueshopi\_larashop'. Under the heading 'ALL PRIVILEGES', several checkboxes are checked: ALTER, CREATE, CREATE TEMPORARY TABLES, DELETE, ALTER ROUTINE, CREATE ROUTINE, CREATE VIEW, and DROP.

8. Scrol down dan klik Make Changes

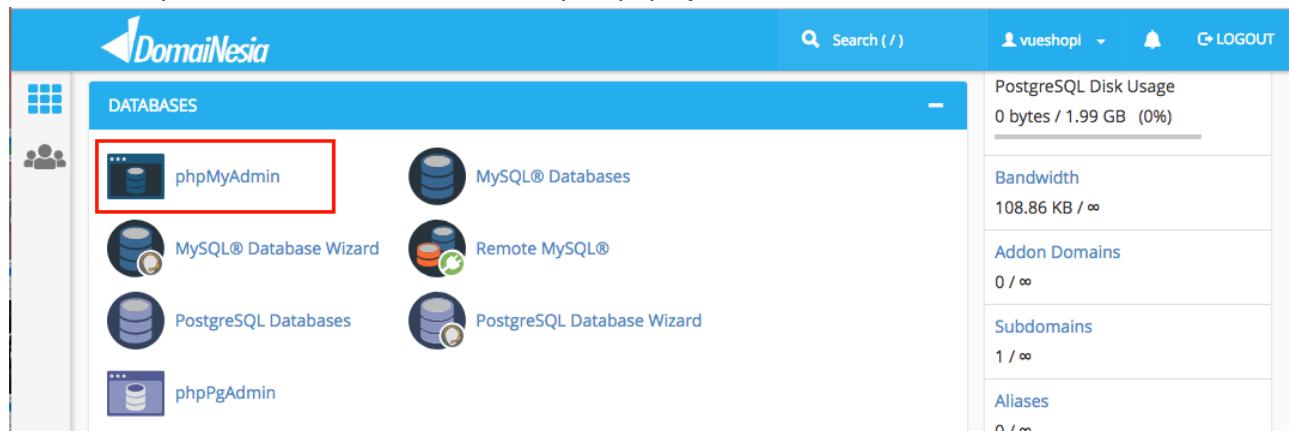
The screenshot shows a list of privileges with checkboxes. Most checkboxes are checked: EVENT, INDEX, LOCK TABLES, SELECT, TRIGGER, EXECUTE, INSERT, REFERENCES, SHOW VIEW, and UPDATE. At the bottom are 'Make Changes' and 'Reset' buttons.

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.



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



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 sections: "General settings" and "Appearance settings". In "General settings", the "Server connection collation" is set to "utf8mb4\_unicode\_ci". In "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, user cpses\_vugfavy7cz@localhost) and "Web server" (listing PHP version 5.6.30).

3. Pada panel database larashop, pilih import

The screenshot shows the phpMyAdmin interface for the "vueshopi\_larashop" database. The title bar says "Server: localhost:3306 > Database: vueshopi\_larashop". The top menu includes "Import". The main area displays a message: "No tables found in database." Below it is a "Create table" form with fields for "Name" and "Number of columns" (set to 4). A "Go" button is at the bottom right.

4. Pada panel import, pilih file sql hasil backup database larashop yang telah kita buat untuk diimport

The screenshot shows the phpMyAdmin interface for the "vueshopi\_larashop" database, specifically the "Import" tab. The title bar says "Server: localhost:3306 > Database: vueshopi\_larashop". The main area is titled "Importing into the database \"vueshopi\_larashop\"". It has a "File to import:" section with instructions: "File may be compressed (gzip, bzip2, zip) or uncompressed. A compressed file's name must end in .[format].[compression]. Example: .sql.zip". It includes fields for "Browse your computer:" (with "Choose File" button), "Character set of the file:" (set to "utf-8"), and a "Go" button.

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

```
DB_CONNECTION=mysql
DB_HOST=localhost
DB_PORT=3306
```

```
DB_DATABASE=vueshopi_larashop
DB_USERNAME=vueshopi_app
DB_PASSWORD=PASSWORD_DATABASEMU
```

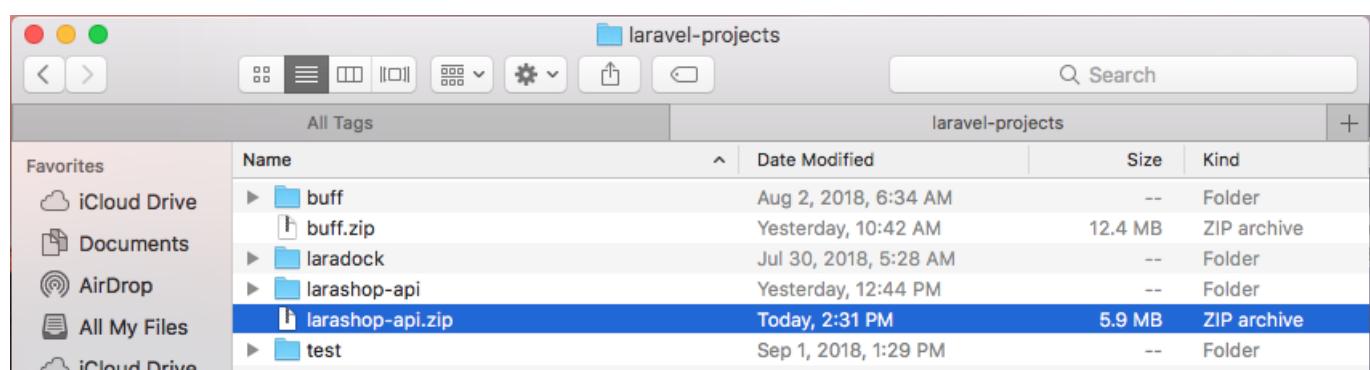
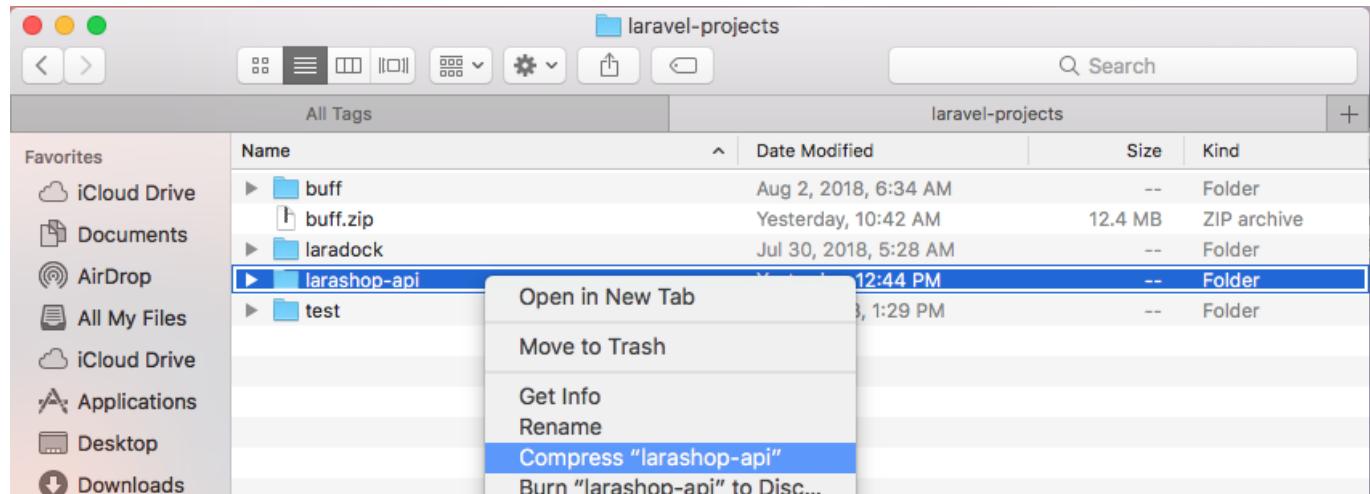
## Mengunggah File Aplikasi

Setelah database siap, maka aplikasi web service kita siap di deploy.

Jalankan perintah berikut pada.

```
php artisan cache:clear
composer install --optimize-autoloader --no-dev
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

The screenshot shows the DomaiNesia control panel interface. On the left, there's a sidebar with icons for users, files, domains, databases, and more. The 'FILES' section is expanded, showing various management tools: File Manager, Directory Privacy, Disk Usage, FTP Accounts, FTP Connections, Backup, Backup Wizard, CpCleaner, Git™ Version Control, and File and Directory Restoration. The 'File Manager' icon is highlighted with a red box. On the right, there are sections for Last Login IP Address (114.4.79.55), Theme (paper\_lantern), and Server Information. Below that is a 'STATISTICS' section showing Disk Usage (5.57 MB / 2 GB (0.27%)) and MySQL® Disk Usage (0 bytes / 1.99 GB (0%)).

2. Pada file manager pilih menu Upload

The screenshot shows the cPanel File Manager interface. At the top, there's a toolbar with File, Folder, Copy, Move, Upload, Download, Delete, Restore, Rename, Edit, and HTML Editor buttons. The 'Upload' button is highlighted with a red box. Below the toolbar is a navigation bar with Home, Up One Level, Back, Forward, Reload, Select All, and Unselect All buttons. Underneath is a trash bin section with View Trash and Empty Trash options. The main area shows a file tree for the '/home/vueshopi' directory and a table of files with columns for Name, Size, Last Modified, and Type.

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
ssl			
tmp			

3. Pada form upload, klik tombol Select untuk memilih file kode aplikasi web service yang telah dikompres

The screenshot shows the 'File Upload' form. It has a message 'Select the file you want to upload to "/home/vueshopi".' and a note 'Maximum file size allowed for upload: 1.99 GB'. There's a checkbox for 'Overwrite existing files'. Below is a dashed box with the text 'Drop files here to start uploading' and an 'or' option, followed by a 'Select File' button.

4. Tunggu sampai selesai dan klik Go Back.

**File Upload**

Select the file you want to upload to "/home/vueshopi".

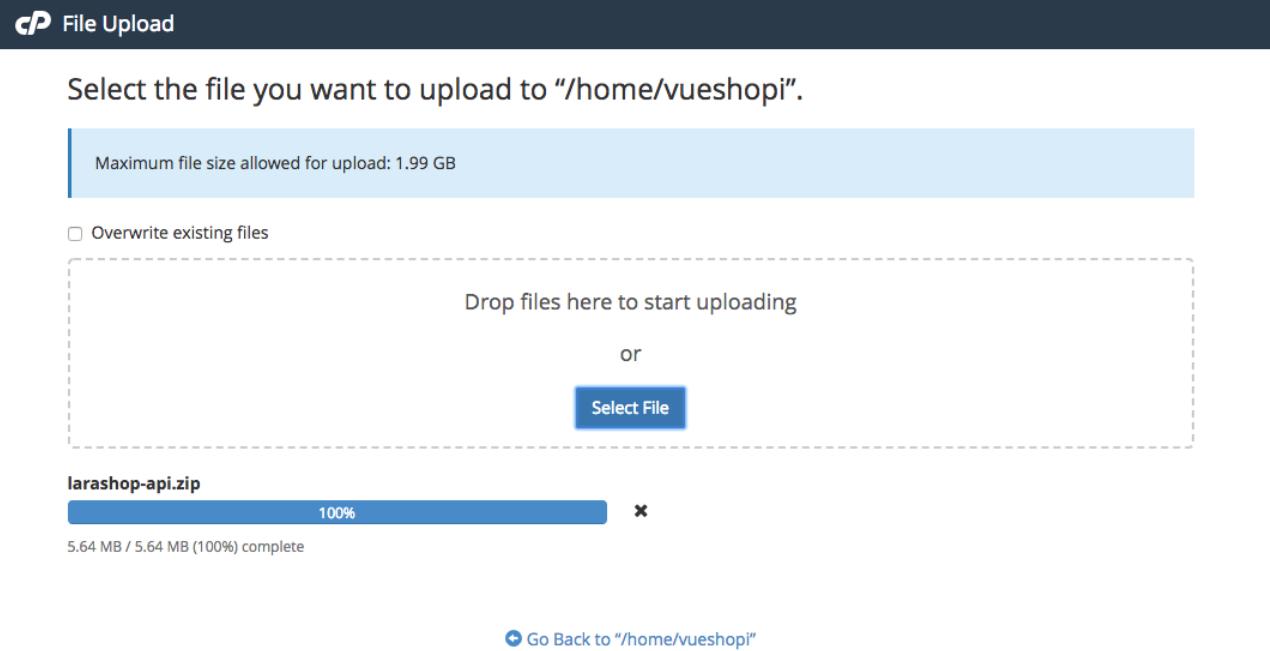
Maximum file size allowed for upload: 1.99 GB

Overwrite existing files

Drop files here to start uploading  
or  
[Select File](#)

**larashop-api.zip**  
100%  
5.64 MB / 5.64 MB (100%) complete

[Go Back to "/home/vueshopi"](#)



5. Extract file zip larashop-api.zip.

**File Manager**

Search All Your Files for [Go](#) Settings

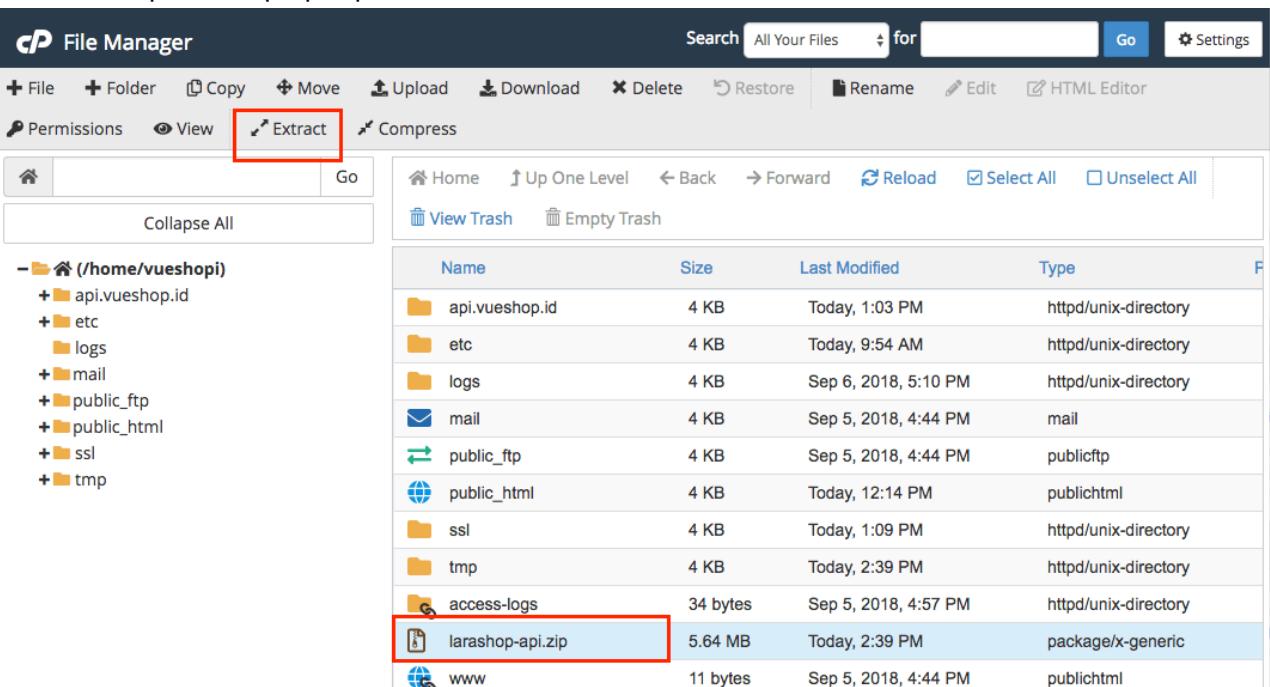
+ File + Folder Copy Move Upload Download Delete Rename Edit HTML Editor

Permissions View Extract Compress

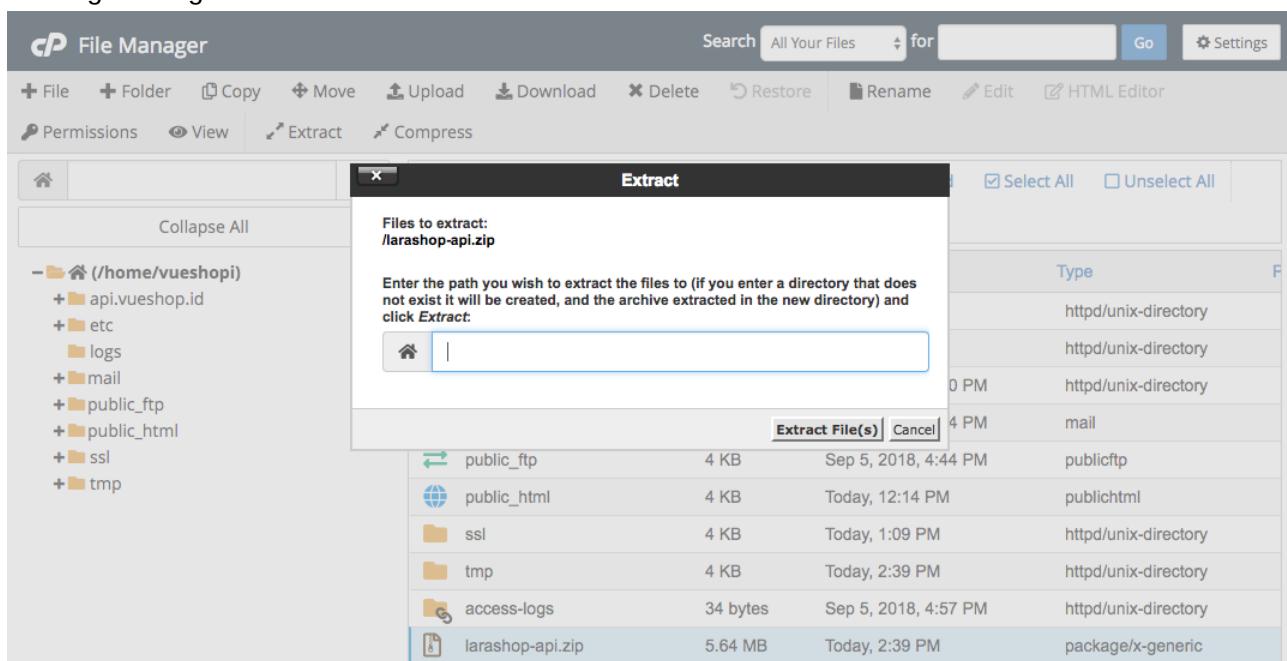
Home Up One Level Back Forward Reload Select All Unselect All

View Trash Empty Trash

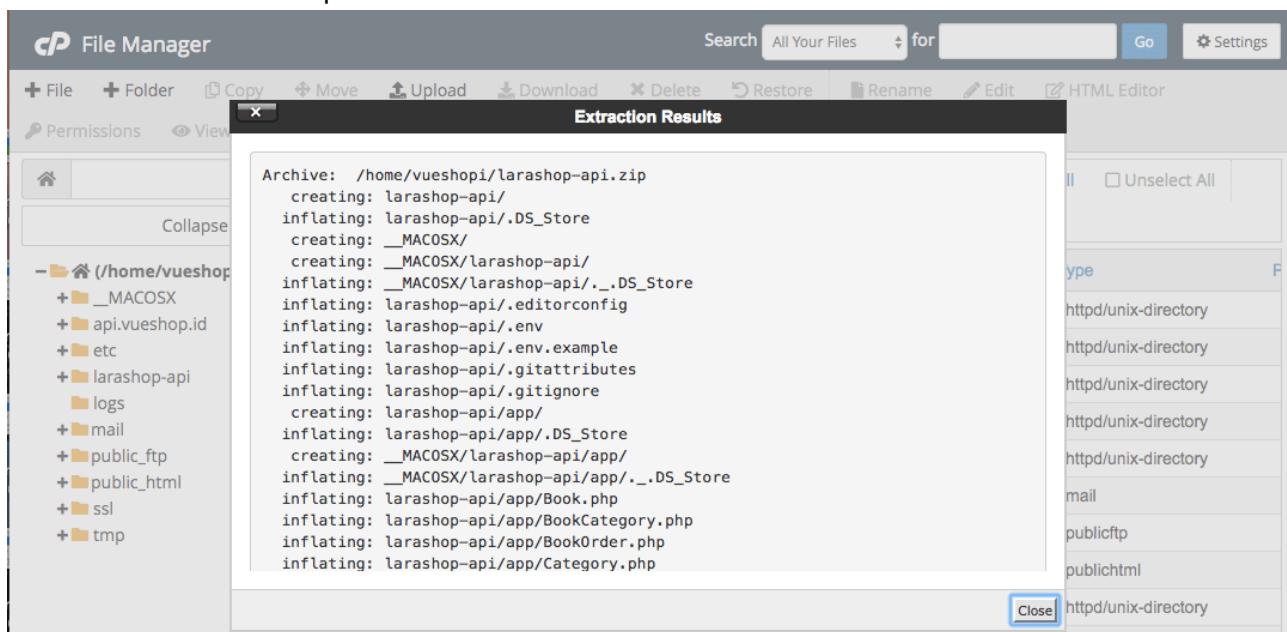
Name	Size	Last Modified	Type
api.vueshop.id	4 KB	Today, 1:03 PM	httpd/unix-directory
etc	4 KB	Today, 9:54 AM	httpd/unix-directory
logs	4 KB	Sep 6, 2018, 5:10 PM	httpd/unix-directory
mail	4 KB	Sep 5, 2018, 4:44 PM	mail
public_ftp	4 KB	Sep 5, 2018, 4:44 PM	publicftp
public_html	4 KB	Today, 12:14 PM	publichtml
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
<b>larashop-api.zip</b>	5.64 MB	Today, 2:39 PM	package/x-generic
www	11 bytes	Sep 5, 2018, 4:44 PM	publichtml



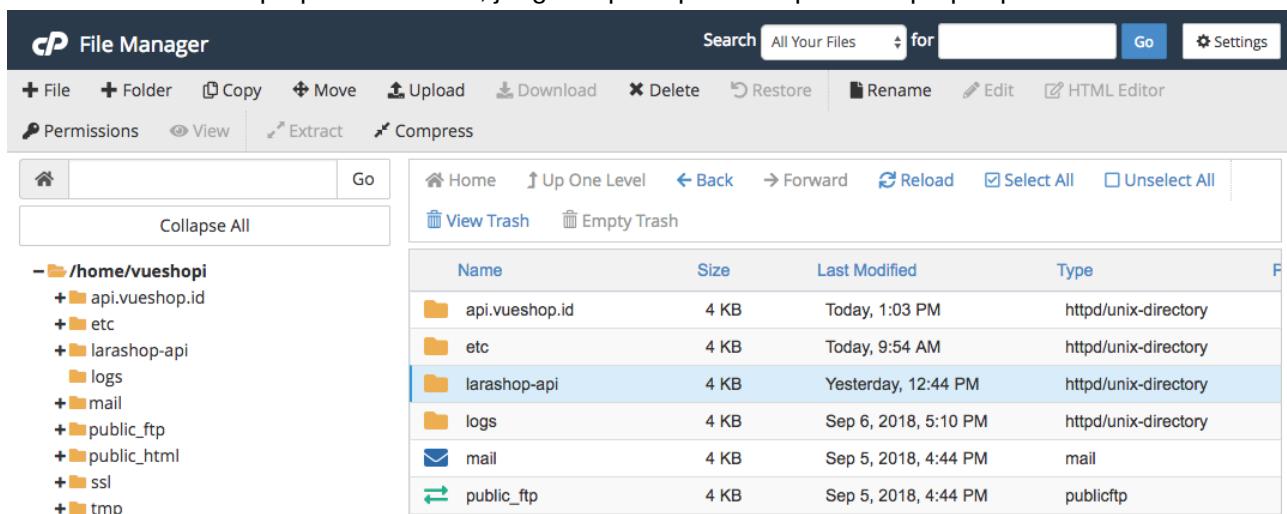
## 6. Kosongkan target extract dan klik tombol Extract ...



## 7. Klik tombol close setelah proses extract selesai



## 8. Posisi folder larashop-api hasil extract, jangan lupa hapus file zip larashop-api.zip



## 9. Pada folder larashop-api, klik folder public

The screenshot shows the cPanel File Manager interface. The left sidebar shows a tree view of the directory structure under '/home/vueshopi'. The 'public' folder is selected and highlighted with a dashed border. The right panel displays a table of files and folders within the 'public' directory. The table has columns for Name, Size, Last Modified, and Type. All files listed are of size 4 KB or less, modified yesterday, and are httpd/unix-directories.

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.

This screenshot is identical to the one above, showing the cPanel File Manager interface with the 'public' folder selected. However, all files in the list are now highlighted in blue, indicating they have been selected for a copy operation.

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.

The screenshot shows the cPanel File Manager interface again. The 'Settings' button in the top right corner is highlighted with a red box. This button is used to access advanced file management options.

Kemudian pada popup yang muncul, checklist show hidden files

The screenshot shows the cPanel File Manager interface. A 'Preferences' dialog box is open over the file list. In the preferences, there is a section for 'Always open this directory in the future by default:' with options for 'Home Directory', 'Web Root (public\_html or www)', 'Public FTP Root (public\_ftp)', and 'Document Root for:' which is set to 'vueshop.id'. Below this, there are two checkboxes: 'Show Hidden Files (dotfiles)' (which is checked and highlighted with a red box) and 'Disable Character Encoding Verification Dialogs'. At the bottom right of the dialog are 'Save' and 'Cancel' buttons.

## 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: .softaculous, .spamassassin, .trash, \_MACOSX, api.vueshop.id, etc, larashop-api (with app, bootstrap, config, database, public, resources, routes, storage, tests, vendor), logs, and mail. On the right, a list of files in 'larashop-api/public' is shown, including css, images, js, svg, .htaccess, favicon.ico, index.php, robots.txt, and web.config. A 'Copy' dialog box is overlaid on the interface, with the 'Copy' button highlighted with a red box.

## 12. Copy semua file tersebut ke dalam folder api.vueshop.id

The screenshot shows the cPanel File Manager interface. On the left, a tree view shows the directory structure: larashop-api/public (with css and images). On the right, a 'Copy' dialog box is open. It shows the 'File path:' as '/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', and '/larashop-api/public/web.config'. Below this, it says 'Enter the file path that you want to copy this file to:' followed by a text input field containing '/api.vueshop.id'. At the bottom right of the dialog are 'Copy File(s)' and 'Cancel' buttons.

13. Buka folder `api.vueshop.id`, lalu edit file `index.php` di dalamnya.

The screenshot shows the cPanel File Manager interface. The left sidebar lists the directory structure under `/home/vueshopi`, with `api.vueshop.id` selected. The main area displays a list of files and directories with columns for Name, Size, Last Modified, and Type. The `index.php` file is highlighted with a red box. The top right corner features a toolbar with various file management icons, and the 'Edit' button is specifically highlighted with a red box.

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

```
require __DIR__.'/../vendor/autoload.php';
$app = require_once __DIR__.'/../bootstrap/app.php';
```

Ubah menjadi

```
require __DIR__.'/../larashop-api/vendor/autoload.php';
$app = require_once __DIR__.'/../larashop-api/bootstrap/app.php';
```

The screenshot shows a file editor window with the following details:

- Editing: `/home/vueshopi/api.vues`
- Encoding: `utf-8`
- Buttons: Re-open, Use legacy editor, Save Changes, Close
- Toolbar: Keyboard shortcuts, Search, Go, Undo, Redo, Font Size (13px), PHP

The code in the editor is as follows:

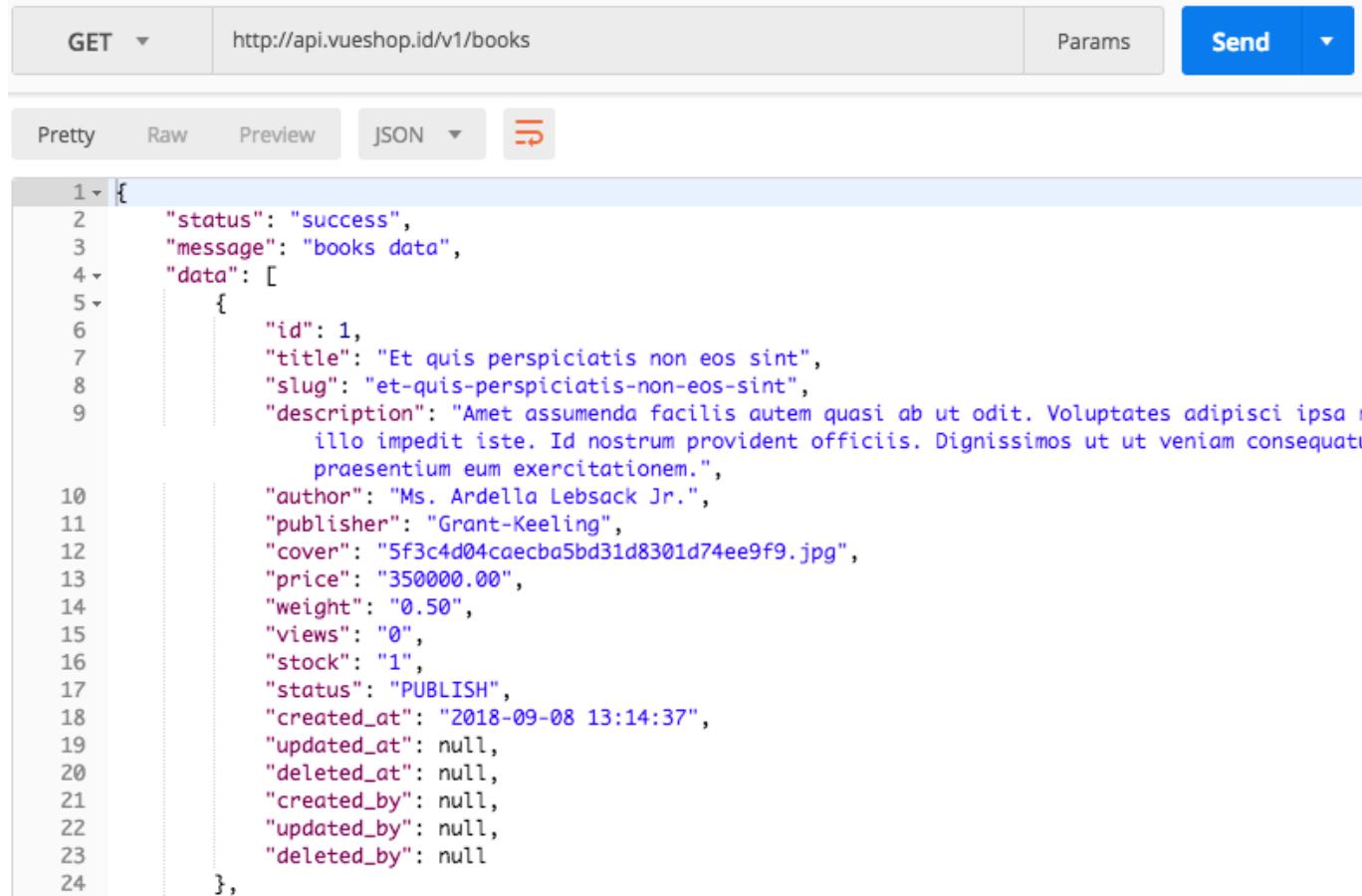
```
23
24 require __DIR__.'/../larashop-api/vendor/autoload.php';
25
26 /*
27 |-----
28 | Turn On The Lights
29 |-----
30 |
31 | We need to illuminate PHP development, so let us turn on the lights.
32 | This bootstraps the framework and gets it ready for use, then it
33 | will load up this application 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 */
```

Lines 24 and 38 are highlighted with red boxes.

Intinya arahkan ke folder `larashop-api`.

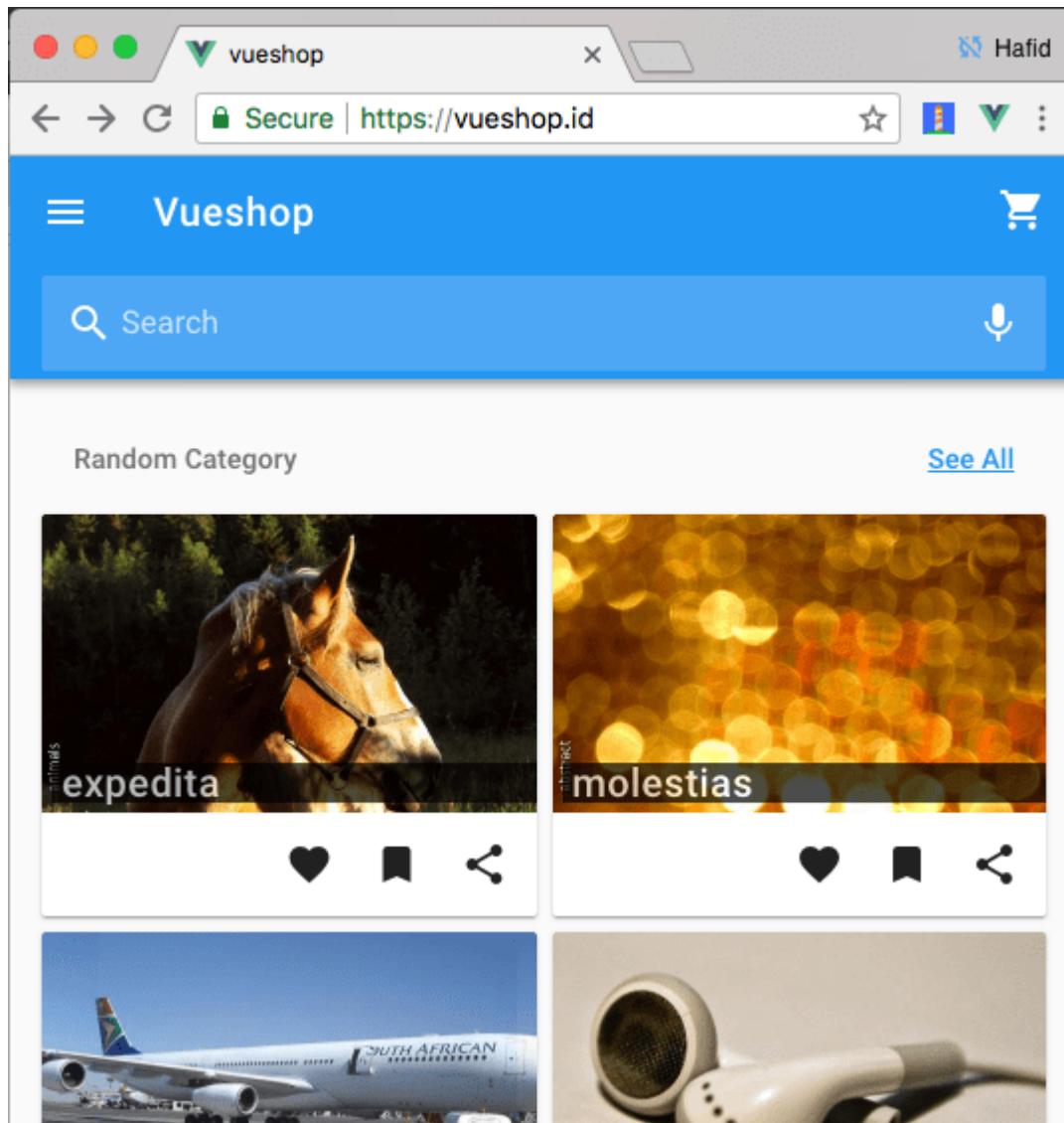
Lakukan uji coba untuk mengakses salah satu endpoint yang sifatnya publik misalnya:

<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 i 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 },
25]
26 }
```

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 di dukung 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.

The advertisement features a large, stylized blue server stack in the foreground, with several white clouds scattered around it. In the background, a white blimp with the 'Domainesia' logo is flying over a series of blue rectangular blocks representing data storage or network layers. The overall theme is cloud computing and web hosting.

**// 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 rankingshosting.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](mailto:info@domainesia.com) | (0274) 545653  
[www.domainesia.com](http://www.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.