

Kebutuhan sistem menggunakan framework laravel

Nama : Bayu wahyu pambudi

Universita : Universita Negeri Surabaya

Kelompok : Kelompok 5

Link Repository: <https://gitlab.com/bayu.22140/pw1-bast7-bayu>

Laravel adalah framework PHP open-source yang dirancang untuk memudahkan pengembangan aplikasi web dengan pendekatan arsitektur Model-View-Controller (MVC). Laravel menawarkan berbagai fitur seperti routing yang sederhana, autentikasi bawaan, ORM melalui Eloquent, serta blade templating engine yang memudahkan pembuatan antarmuka dinamis. Framework ini sangat populer di kalangan pengembang karena sintaksisnya yang bersih dan mudah dipahami, serta kemampuannya untuk meningkatkan efisiensi dalam membangun aplikasi web yang kompleks.

Secara historis, Laravel pertama kali diperkenalkan oleh Taylor Otwell pada tahun 2011 dengan tujuan untuk memberikan alternatif yang lebih canggih dibandingkan CodeIgniter, yang saat itu masih memiliki keterbatasan. Sejak awal, Laravel dirancang untuk mendukung fitur-fitur yang lebih modern seperti routing yang fleksibel dan sistem autentikasi yang terintegrasi. Pada versi kedua, Laravel menjadi framework penuh berbasis MVC, dan seiring perkembangan versi selanjutnya, Laravel 3 memperkenalkan fitur-fitur utama seperti Artisan CLI, sistem migrasi database, dan dukungan paket. Dengan Laravel 4, yang dirilis pada tahun 2013, framework ini dibangun ulang menggunakan komponen dari Symfony, menjadikannya lebih modular dan fleksibel. Pada versi 5 dan seterusnya, Laravel semakin berkembang dengan menambahkan fitur seperti Laravel Scheduler, middleware, dan peningkatan manajemen dependency. Versi terbaru dari Laravel terus menyempurnakan performa dan menambahkan fitur-fitur baru yang mendukung kebutuhan pengembangan modern, seperti job queues, event broadcasting, dan integrasi dengan Vue.js. Laravel kini menjadi salah satu framework PHP paling populer karena dokumentasinya yang lengkap, ekosistem yang luas, serta dukungan komunitas yang aktif.

Perangkat keras menggunakan framework laravel

Untuk menggunakan framework Laravel, baik pada **server development** maupun **perangkat komputer/laptop**, ada beberapa spesifikasi perangkat keras yang perlu diperhatikan agar pengembangan berjalan lancar. Berikut adalah spesifikasi yang direkomendasikan:

A. Server Development:

Server development adalah lingkungan yang digunakan untuk menjalankan aplikasi Laravel selama proses pengembangan dan testing. Beberapa hal yang perlu dipersiapkan antara lain:

- **Prosesor:** Minimal prosesor quad-core, tetapi lebih disarankan menggunakan prosesor modern seperti Intel Xeon, AMD EPYC, atau Intel i7/i9 untuk performa yang lebih baik, terutama jika server juga menangani tugas berat seperti kompilasi, menjalankan banyak proses secara bersamaan, atau simulasi beban tinggi.
- **RAM:** Minimal 4GB, tetapi disarankan 8GB atau lebih, terutama jika server menjalankan banyak layanan seperti database (MySQL, PostgreSQL), server web (Apache/Nginx), serta komponen lain seperti Redis atau Elasticsearch.
- **Storage (Hard Disk atau SSD):** Disarankan menggunakan SSD karena lebih cepat dalam akses data dan pengelolaan file daripada HDD. Minimal 50GB storage, namun lebih besar akan dibutuhkan tergantung pada ukuran aplikasi dan file-file pendukung.
- **Operating System:** Laravel dapat berjalan di server dengan OS berbasis Unix/Linux (seperti Ubuntu, CentOS) atau Windows. Namun, kebanyakan pengembang lebih memilih sistem operasi Linux karena stabilitas dan kompatibilitas yang baik dengan alat-alat pengembangan berbasis web.
- **Web Server:** Nginx atau Apache, dengan dukungan untuk PHP dan konfigurasi yang baik untuk menjalankan aplikasi Laravel.
- **PHP:** Versi PHP minimal 8.0 direkomendasikan, meskipun Laravel juga dapat berjalan di versi 7.4. Sebaiknya menggunakan versi terbaru untuk memaksimalkan fitur dan keamanan.

B. Perangkat Komputer/Laptop:

Untuk pengembangan Laravel di komputer atau laptop, berikut adalah spesifikasi yang disarankan:

- **Prosesor:** Minimal Intel Core i5 atau AMD Ryzen 5. Prosesor multi-core akan sangat membantu dalam multitasking, seperti menjalankan server lokal, kompilasi kode, dan menjalankan tools lain secara bersamaan. Prosesor yang lebih tinggi seperti i7/i9 atau Ryzen 7/9 akan memberikan performa yang lebih baik.
- **RAM:** Minimal 8GB untuk pengembangan dasar, tetapi lebih baik 16GB atau lebih, terutama jika Anda menggunakan virtual machine atau menjalankan banyak aplikasi seperti editor kode (VS Code, PHPStorm), browser, database server, dan alat pengujian lainnya.
- **Storage:** SSD sangat disarankan dengan kapasitas minimal 256GB. SSD meningkatkan kecepatan akses file, terutama saat menjalankan server lokal dan melakukan build aplikasi.
- **Grafik:** Untuk pengembangan Laravel, kartu grafis (GPU) bukanlah hal utama. Namun, jika Anda bekerja dengan desain antarmuka atau fitur lain yang melibatkan rendering grafik, maka GPU kelas menengah (seperti Intel Iris atau GPU terintegrasi modern) sudah cukup.
- **Operating System:** Laravel dapat dikembangkan di berbagai OS seperti Windows, macOS, atau Linux. Namun, banyak pengembang memilih

menggunakan macOS atau Linux karena kemudahan instalasi dan dukungan yang lebih baik untuk alat-alat pengembangan berbasis Unix.

- **PHP dan Development Tools:** Pastikan Anda menginstal PHP (minimal versi 8.0), Composer (untuk mengelola dependensi Laravel), dan database seperti MySQL atau PostgreSQL di komputer Anda.

Dengan spesifikasi perangkat keras yang memadai, pengembangan aplikasi Laravel dapat berjalan lancar dan efisien tanpa hambatan performa.

Perangkat lunak menggunakan framework laravel

Untuk menggunakan framework Laravel, ada beberapa kebutuhan perangkat lunak yang harus dipenuhi agar pengembangan dan eksekusi aplikasi berjalan dengan baik. Berikut adalah spesifikasi perangkat lunak yang diperlukan:

1. Web Server:

Laravel membutuhkan web server untuk melayani permintaan HTTP dan menjalankan aplikasi. Dua web server yang paling umum digunakan adalah:

- **Nginx atau Apache:**
 - **Nginx:** Ini adalah web server ringan dan cepat yang sering digunakan dalam skenario produksi karena efisiensi dan kemampuannya menangani beban tinggi. Nginx juga memiliki dukungan yang baik untuk pengelolaan cache dan proxy.
 - **Apache:** Salah satu web server paling populer dan fleksibel, Apache memiliki banyak modul tambahan dan dukungan konfigurasi yang luas. Laravel sering digunakan dengan Apache, terutama pada server pengembangan.
- **Kebutuhan Konfigurasi:**
 - Web server harus mendukung **PHP-FPM** (untuk Nginx) atau **mod_php** (untuk Apache) agar dapat menjalankan aplikasi Laravel yang berbasis PHP.
 - Pastikan modul URL rewriting diaktifkan. Untuk Apache, **mod_rewrite** sangat penting agar routing Laravel dapat berfungsi dengan benar.

2. Database:

Laravel mendukung berbagai jenis database. Beberapa database yang sering digunakan adalah:

- **MySQL/MariaDB:** Ini adalah pilihan umum karena performa yang baik dan skalabilitas. MySQL dan MariaDB mudah diintegrasikan dengan Laravel melalui ORM Eloquent. Selain itu, banyak tutorial Laravel menggunakan MySQL sebagai contoh.
- **PostgreSQL:** PostgreSQL adalah database relasional open-source dengan fitur-fitur canggih seperti dukungan JSON dan performa yang sangat baik. Laravel memiliki dukungan bawaan untuk PostgreSQL.

- **SQLite:** SQLite adalah pilihan yang bagus untuk pengembangan atau aplikasi kecil, karena menggunakan file tunggal untuk penyimpanan data. Ini ringan dan sederhana untuk digunakan.
- **SQL Server:** Laravel juga mendukung Microsoft SQL Server, yang berguna jika Anda bekerja di lingkungan yang menggunakan teknologi Microsoft.

Persyaratan:

- Anda harus menginstal driver database yang sesuai pada sistem Anda.
- Laravel menggunakan Eloquent ORM, yang secara otomatis menangani query database dan pengelolaan relasi antara tabel.

3. PHP:

Laravel adalah framework PHP, sehingga versi PHP yang tepat diperlukan:

- **Versi PHP:** Minimal versi 8.0, meskipun Laravel juga mendukung PHP 7.4 pada versi-versi sebelumnya. Namun, menggunakan versi terbaru dari PHP sangat disarankan karena fitur-fitur baru dan perbaikan keamanan yang diperkenalkan di setiap rilis.

Ekstensi PHP yang diperlukan:

- **OpenSSL:** Digunakan untuk pengelolaan sertifikat dan koneksi aman.
- **PDO:** Driver yang dibutuhkan untuk koneksi database.
- **Mbstring:** Digunakan untuk menangani string multibyte.
- **Tokenizer:** Digunakan untuk parsing kode.
- **XML:** Diperlukan oleh beberapa library internal PHP.
- **Ctype:** Ekstensi untuk memeriksa tipe karakter.
- **Fileinfo:** Digunakan untuk memeriksa jenis file yang diupload.

4. Composer:

Composer adalah dependency manager untuk PHP, dan Laravel sangat bergantung pada Composer untuk mengelola paket-paketnya.

- **Composer** berfungsi untuk menginstal Laravel serta dependensi yang dibutuhkan oleh proyek, seperti library dan paket tambahan. Laravel sendiri terdaftar sebagai paket di Composer, sehingga Anda dapat menginstalnya dan memperbarui framework dengan mudah.

Persyaratan Composer:

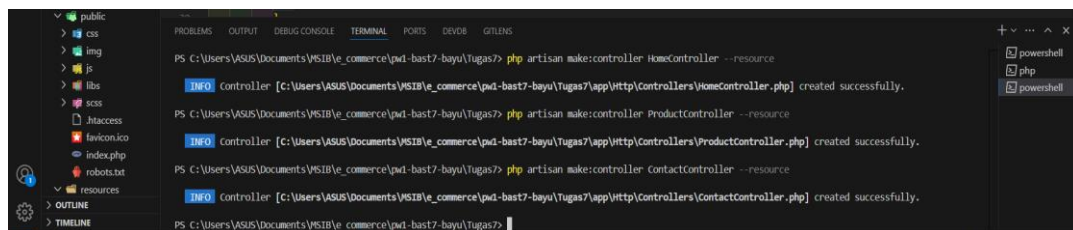
- Composer harus diinstal di sistem Anda. Setelah terinstal, Anda dapat menggunakan perintah seperti `composer install` untuk menginstal semua dependensi yang didefinisikan dalam file `composer.json`.
- Composer juga menyediakan cara untuk manage autoloading, sehingga Anda dapat dengan mudah mengimpor kelas-kelas dalam aplikasi Laravel tanpa harus menulis manual `require statements`.

konsep routing pada laravel

Routing adalah mekanisme penting dalam aplikasi web untuk menentukan jalur yang diambil oleh permintaan (request) dari client menuju aplikasi. Fungsi utamanya adalah memetakan URL yang diakses oleh pengguna ke logika yang bertanggung jawab untuk menanganinya, seperti controller atau fungsi anonim. Routing memungkinkan pengembang untuk memisahkan URL dari logika bisnis, sehingga memudahkan pengelolaan dan pemeliharaan aplikasi, serta meningkatkan keamanan dengan membatasi akses hanya pada rute tertentu.

Di Laravel, routing diatur dalam beberapa file di dalam folder *routes*. Laravel menangani permintaan HTTP menggunakan berbagai metode seperti *GET*, *POST*, *PUT*, dan *DELETE*. Saat permintaan diterima, Laravel memetakan rute ke controller atau closure yang telah didefinisikan untuk memprosesnya. Contohnya, file *web.php* menyimpan rute yang berhubungan dengan permintaan dari browser, lengkap dengan middleware seperti session dan perlindungan CSRF. File *api.php* digunakan untuk mendefinisikan rute API yang biasanya tidak memerlukan session atau CSRF protection, dan sering kali menggunakan namespace khusus seperti */api*. File *console.php* mendefinisikan perintah yang bisa dijalankan melalui Command Line Interface (CLI) Laravel menggunakan artisan, sedangkan *channels.php* berfungsi untuk mendefinisikan saluran broadcast event yang digunakan dalam komunikasi real-time. Dengan pembagian ini, Laravel memudahkan pengelolaan rute untuk berbagai jenis permintaan, baik dari web, API, CLI, maupun real-time event broadcasting.

A. Membuat controller



```
PS C:\Users\ASUS\Documents\VSIB\commerce\pud-bast7-bayu\Tugas7> php artisan make:controller HomeController --resource
INFO Controller [C:\Users\ASUS\Documents\VSIB\commerce\pud-bast7-bayu\Tugas7\app\Http\Controllers\HomeController.php] created successfully.

PS C:\Users\ASUS\Documents\VSIB\commerce\pud-bast7-bayu\Tugas7> php artisan make:controller ProductController --resource
INFO Controller [C:\Users\ASUS\Documents\VSIB\commerce\pud-bast7-bayu\Tugas7\app\Http\Controllers\ProductController.php] created successfully.

PS C:\Users\ASUS\Documents\VSIB\commerce\pud-bast7-bayu\Tugas7> php artisan make:controller ContactController --resource
INFO Controller [C:\Users\ASUS\Documents\VSIB\commerce\pud-bast7-bayu\Tugas7\app\Http\Controllers>ContactController.php] created successfully.

PS C:\Users\ASUS\Documents\VSIB\commerce\pud-bast7-bayu\Tugas7>
```

B. Menghubungkan controller dengan route

```
Route::resource('home', App\Http\Controllers\HomeController::class);
Route::resource('products', App\Http\Controllers\ProductController::class);
Route::resource('contact', App\Http\Controllers>ContactController::class);
```

Data route yang di buat

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL	PORTS	DEVD	GLTENS
GET HEAD	/		dashboard			
POST	_ignition/execute-solution		ignition.executeSolution > Spatie\LaravelIgnition > ExecuteSolutionController			
GET HEAD	_ignition/health-check		ignition.healthCheck > Spatie\LaravelIgnition > HealthCheckController			
POST	_ignition/update-config		ignition.updateConfig > Spatie\LaravelIgnition > UpdateConfigController			
GET HEAD	api/user					
GET HEAD	contact		contact.index > ContactController@index			
POST	contact		contact.store > ContactController@store			
GET HEAD	contact/create		contact.create > ContactController@create			
GET HEAD	contact/{contact}		contact.show > ContactController@show			
PUT PATCH	contact/{contact}		contact.update > ContactController@update			
DELETE	contact/{contact}		contact.destroy > ContactController@destroy			
GET HEAD	contact/{contact}/edit		contact.edit > ContactController@edit			
GET HEAD	home		home.index > HomeController@index			
POST	home		home.store > HomeController@store			
GET HEAD	home/create		home.create > HomeController@create			
GET HEAD	home/{home}		home.show > HomeController@show			
PUT PATCH	home/{home}		home.update > HomeController@update			
DELETE	home/{home}		home.destroy > HomeController@destroy			
GET HEAD	home/{home}/edit		home.edit > HomeController@edit			
GET HEAD	login		login > Auth\LoginController@showLoginForm			
POST	login		Auth\LoginController@login			
POST	logout		logout > Auth\LoginController@logout			
GET HEAD	password/confirm		password.confirm > Auth\ConfirmPasswordController@showConfirmForm			
POST	password/confirm		Auth\ConfirmPasswordController@confirm			
POST	password/email		password.email > Auth\ForgotPasswordController@sendResetLinkEmail			
GET HEAD	password/reset		password.request > Auth\ForgotPasswordController@showLinkRequestForm			
POST	password/reset		password.update > Auth\ResetPasswordController@reset			
GET HEAD	password/reset/{token}		password.reset > Auth\ResetPasswordController@showResetForm			
GET HEAD	products		products.index > ProductController@index			
POST	products		products.store > ProductController@store			
GET HEAD	products/create		products.create > ProductController@create			
GET HEAD	products/{product}		products.show > ProductController@show			
PUT PATCH	products/{product}		products.update > ProductController@update			
DELETE	products/{product}		products.destroy > ProductController@destroy			
GET HEAD	products/{product}/edit		products.edit > ProductController@edit			
GET HEAD	register		register > Auth\RegisterController@showRegistrationForm			

C. Menambahkan prefix route

```
Route::prefix('admin')->group(function () {
    Route::resource('home', App\Http\Controllers\HomeController::class);
    Route::resource('products', App\Http\Controllers\ProductController::class);
    Route::resource('contact', App\Http\Controllers>ContactController::class);
});
```

Data route yang telah di buat

ps C:\Users\ASUS\Documents\MSIB\re-commerce\pw1-bast7-bayu\Tugas7> php artisan route:list						
GET HEAD	/		dashboard			
POST	_ignition/execute-solution		ignition.executeSolution > Spatie\LaravelIgnition > ExecuteSolutionController			
GET HEAD	_ignition/health-check		ignition.healthCheck > Spatie\LaravelIgnition > HealthCheckController			
POST	_ignition/update-config		ignition.updateConfig > Spatie\LaravelIgnition > UpdateConfigController			
GET HEAD	admin/contact		contact.index > ContactController@index			
POST	admin/contact		contact.store > ContactController@store			
GET HEAD	admin/contact/create		contact.create > ContactController@create			
GET HEAD	admin/contact/{contact}		contact.show > ContactController@show			
PUT PATCH	admin/contact/{contact}		contact.update > ContactController@update			
DELETE	admin/contact/{contact}		contact.destroy > ContactController@destroy			
GET HEAD	admin/contact/{contact}/edit		contact.edit > ContactController@edit			
GET HEAD	admin/home		home.index > HomeController@index			
POST	admin/home		home.store > HomeController@store			
GET HEAD	admin/home/create		home.create > HomeController@create			
GET HEAD	admin/home/{home}		home.show > HomeController@show			
PUT PATCH	admin/home/{home}		home.update > HomeController@update			
DELETE	admin/home/{home}		home.destroy > HomeController@destroy			
GET HEAD	admin/home/{home}/edit		home.edit > HomeController@edit			
GET HEAD	admin/products		products.index > ProductController@index			
POST	admin/products		products.store > ProductController@store			
GET HEAD	admin/products/create		products.create > ProductController@create			
GET HEAD	admin/products/{product}		products.show > ProductController@show			
PUT PATCH	admin/products/{product}		products.update > ProductController@update			
DELETE	admin/products/{product}		products.destroy > ProductController@destroy			
GET HEAD	admin/products/{product}/edit		products.edit > ProductController@edit			