Pengembangan *Single Website Application* untuk *Multiple Domain* Menggunakan *Laravel Frameworks*

Danny Sebastian[1)]

[1)] Informatika, Fakultas Teknologi Informasi, Universitas Kristen Duta Wacana
Jln Dr. Wahidin Sudirohusodo No 5-25, Yogyakarta
[1)] danny.sebastian@staff.ukdw.ac.id

Abstrak

Aplikasi website menjadi salah satu komponen penting bagi sebuah perusahaan. Setiap aplikasi *website* terdaftar ke sebuah alamat *domain* untuk mengakses aplikasi tersebut. Sebuah *website* biasanya memiliki sebuah halaman yang bersifat umum dan sebuah halaman *private* yang digunakan untuk pengelolaan *data* yang ditampilkan. Tetapi hal ini membuat *developed time* menjadi lama karena developer perlu membuat dua buah muka website. *Laravel frameworks* memungkinkan sebuah aplikasi menerima *request* dari lebih dari 1 *domain*. Penulis membuat sebuah aplikasi *website* yang dapat menerima *request* dari banyak *domain*. Arsitektur ini memungkinkan pengelolaan semua *data* hanya menggunakan sebuah antarmuka aplikasi *website*. Skenario pengujian aplikasi menggunakan *automated testing* dan *user acceptance test* mendapatkan hasil yang sesuai dengan luaran yang diharapkan. Arsitektur yang diujikan dapat mengurangi *development time* sampai 1/3,3125 kali.

Kata kunci: multiple domain, aplikasi website, Laravel frameworks

Abstract

Website application is one of the important components for a company. Each website application is registered to a domain address to access the application. A website usually has a general page and a private page that is used for managing the displayed data. But this makes the developed time long because the developer needs to make two website faces. Laravel frameworks allow an application to receive requests from more than 1 domain. The author creates a website application that can accept requests from multiple domains. Automated testing and user acceptance test shows the same output between expected and actual output. Tested architecture reduces development time up to 1/3,3125 times.

Keywords: *multiple domain, website application, Laravel frameworks*

1. PENDAHULUAN

Perkembangan teknologi menuntut pelaku usaha untuk mengembangkan cara berniaga, salah satunya adalah dengan memanfaatkan teknologi *website*. Banyak perusahaan yang berusaha untuk membuat aplikasi *website* untuk perusahaan mereka. Hal ini menjadi sebuah kesempatan bagi *developer* aplikasi untuk mendapatkan pekerjaan dengan lebih mudah. Tetapi pertumbuhan jumlah *developer* tidak dapat mengimbangi jumlah permintaan untuk pembuatan aplikasi. Sehingga banyak *developer* berusaha mengembangkan teknologi yang dapat mengurangi waktu pembuatan aplikasi atau *development time*.

Perusahaan beranggapan dengan adanya *website*, kredibilitas perusahaan dalam berniaga menggunakan *internet* akan naik dan lebih dipercaya oleh masyarakat. Tetapi dalam pengelolaan aplikasi *website*, diperlukan pembaruan konten yang terus menerus. Banyak sekali aplikasi *website* yang dibangun dengan baik, tetapi tidak diimbangi dengan pembaruan konten secara berkesinambungan. Hal ini menjadi tantangan bagi perusahaan karena mereka harus mengalokasikan dana untuk membayar seorang *administrator* atau *content creator* untuk

DOI: 10.34128/jsi.v7i1.300 *Received: March 13th, 2021*
Accepted: June 9th, 2021

melakukan pembaruan pada aplikasi website mereka. Biaya seorang *administrator* atau *content creator* tidak murah, sehingga muncul lapangan usaha baru, yaitu *digital media management*, atau agensi pengelolaan konten *digital*.

Digital media management membantu perusahaan mengelola konten media dengan konsep 1 orang *administrator* mengelola beberapa aplikasi *website*. Permasalahan bagi *Digital media management* adalah dengan semakin bertambahnya jumlah aplikasi website yang dikelola, maka *administrator* harus mengingat setiap *username* dan *password* untuk masing-masing *website*. Hal ini dikarenakan pada umumnya sebuah aplikasi *website* dibangun menggunakan konsep 1 aplikasi website memiliki 1 halaman pengelolaan. Sehingga *administrator* yang mengelola 10 website, harus mengingat 10 pasang *username* dan *password*.

Berdasarkan permasalahan yang ada, penulis mengangkat rumusan masalah penelitian ini adalah Bagaimana membangun sebuah aplikasi website yang dapat mempermudah proses admin mengelola banyak *website*/banyak alamat *domain* dengan menggunakan 1 halaman pengelolaan yang terpusat (1). Aplikasi website yang dibangun harus mudah di replikasi oleh *developer* dan dapat menghasilkan *development time* yang sekecil-kecilnya (2).

Tujuan dari penelitian ini adalah membuat sebuah aplikasi *website* yang dapat mempermudah *admin* mengelola banyak *website* (banyak alamat *domain*) melalui satu antarmuka *website* secara terpusat. Harapannya aplikasi *website* hasil dari penelitian ini dapat mempermudah *admin website* dalam mengelola *data* konten. Sedangkan bagi *developer* pengembang aplikasi *website* diharapkan dapat mempermudah duplikasi aplikasi *website* dengan fitur serupa dan mengurangi *development time*.

2. TINJAUAN PUSTAKA

Aplikasi website adalah aplikasi yang diakses menggunakan *browser*, dan memungkinkan pertukaran data antara *client* dan *server* [1]. Aplikasi website banyak digunakan oleh perusahaan kecil dan menengah karena aplikasi *website* dapat diakses banyak perangkat [2] [3]. Aplikasi website dapat dibangun menggunakan banyak Bahasa pemrograman. Bahasa pemrograman aplikasi *website* yang banyak digunakan adalah Bahasa PHP. Pengembangan aplikasi *website* dapat dilakukan menggunakan *frameworks* [4].

Frameworks adalah kerangka kerja. Dalam kasus pengembangan aplikasi, frameworks dapat mempercepat proses pengembangan aplikasi karena frameworks sudah menyediakan fungsi-fungsi dasar dalam pengembangan aplikasi [4]. Ada banyak *frameworks* untuk pengembangan aplikasi menggunakan PHP, antara lain Laravel Frameworks, codeigniter [5], Cake PHP [5], Yii [6] dan lain sebagainya. Masing-masing frameworks memiliki kerangka kerja masing-masing dan memiliki keunggulannya masing-masing. Salah satu *frameworks* yang banyak digunakan adalah *Laravel Frameworks*. *Laravel frameworks* cocok digunakan untuk pengembangan aplikasi yang berskala besar dan membutuhkan waktu pengembangan yang cepat [7]. *Laravel frameworks* menghasilkan performa yang lebih baik dibandingkan beberapa *frameworks* lain [7] [8].

Laravel Frameworks menggunakan *routing* untuk mengatur setiap request yang masuk ke aplikasi *website*. *Routing* pada *Laravel frameworks* memungkinkan adanya pengelompokan/*grouping request* yang masuk berdasarkan *domain* [9]. Hal ini memungkinkan beberapa alamat *domain* mengakses sebuah aplikasi yang sama.

Berdasarkan tinjauan pustaka, penulis memilih *Laravel Frameworks* untuk membangun sebuah aplikasi *website* yang dapat menerima request dari beberapa alamat *domain* berbeda. Dan membuat sebuah *website* pengelolaan untuk mengelola semua *data* pada beberapa *website* dengan alamat *domain* yang berbeda.

3. METODE PENELITIAN

Penelitian terdiri dari 4 tahap utama, yaitu penentuan daftar kebutuhan, perancangan, pengembangan aplikasi, pengujian.

3.1 Penentuan Daftar Kebutuhan

Tahap awal penelitian adalah menentukan daftar kebutuhan. Daftar kebutuhan didapatkan melalui observasi dan wawancara terhadap calon pengguna. Selain itu penulis mencari refrensi aplikasi *website* sejenis.

3.2 Perancangan

Pada tahap kedua, peneliti melakukan perancangan aplikasi berdasarkan daftar kebutuhan yang sudah didapatkan pada Langkah sebelumnya. Perancangan dilakukan menggunakan *Entity Relationship Diagram* (ERD) untuk perancangan *database*. Perancangan antar muka dilakukan menggunakan tools Balsamiq.

Secara umum penelitian ini akan membuat sebuah aplikasi *website*, dengan dua muka, yaitu *private facing website* dan *public facing website*. *Private facing website* digunakan oleh pengguna yang sudah login ke aplikasi untuk melakukan pengaturan *data*. *Data* tersebut ditampilkan pada *public facing website*. Hasil dari tahap ini adalah rancangan aplikasi *website* yang akan dibangun

3.3 Pengembangan Aplikasi

Tahap pengembangan aplikasi adalah tahapan untuk membuat kode program aplikasi berdasarkan rancangan yang sudah dilakukan sebelumnya. Bahasa pemrograman yang digunakan adalah PHP versi 8.0.6, dengan *frameworks* Laravel versi 8. Beberapa *tools* yang digunakan adalah Visual Studio Code sebagai *text editor*. Database dan webserver yang digunakan adalah MariaDB versi 10.4 dan Apache *Webserver* versi 2.4 pada *software bundling* XAMPP *for* windows versi 8.0.3. Hasil dari tahap ini adalah program aplikasi *website* yang siap diuji coba.

3.4 Pengujian

Pengujian aplikasi dilakukan dalam 3 kali, yaitu pengujian pada lingkup pengembang aplikasi, pengujian dengan melibatkan pengguna, dan pengujian *development time*.

3.4.1 Pengujian pada lingkup pengembang

Pengujian pada lingkup pengembang aplikasi atau *developer*, dilakukan untuk memastikan aplikasi website berjalan dengan normal tanpa adanya *error* atau *bugs*. Pengujian ini dilakukan menggunakan metode *whitebox testing*, karena penguji atau tester memiliki pengetahuan tentang alur program. Pengujian dilakukan menggunakan tools pengujian otomatis pada *Laravel Frameworks*.

Pengujian *multiple domain* dilakukan menggunakan *virtual host* pada *web server* apache. Penulis melakukan pengujian dengan mendaftarkan 3 buah *domain* untuk tampilan aplikasi *website public* dan 1 buah domain untuk tampilan aplikasi *private facing website*. Dari masing-masing *domain* yang terdaftar, penulis melakukan pengujian dengan cara mengisi data melalui aplikasi *private facing website*. Kemudian dibandingkan dengan tampilan dari *public facing website*. 4 alamat domain yang didaftarkan untuk pengujian dapat dilihat pada Tabel 1.

Tabel 1. Domain yang didaftarkan

No	Domain	Keterangan
1	domainpublic.com	Aplikasi website *public facing primary*
2	domainpublic2.com	Aplikasi website *public facing secondary*
3	domainpublic3.com	Aplikasi website *public facing tertiary*
4	admin.domainpublic.com	Aplikasi website *private facing*

3.4.2 Pengujian *user acceptance test*

Pada tahap ini, pengujian dilakukan dengan melibatkan pengguna. Pengujian ini bertujuan untuk memastikan aplikasi website sudah sesuai dengan kebutuhan pengguna. Metode yang dipilih adalah metode *blackbox testing*. Pengguna yang pada tahap awal menentukan daftar kebutuhan diminta mempelajari aplikasi dan coba menggunakan aplikasi, kemudian pengguna

diminta untuk melakukan pengujian berdasarkan skenario-skenario yang ditentukan. Jumlah responden adalah 3 orang *admin website* yang merupakan perwakilan dari masing-masing *public facing website*. Beberapa pengujian yang dilakukan dapat dilihat pada Tabel 2.

Tabel 2. Daftar pengujian *user acceptance test*

No	Fitur	*Data* yang dibutuhkan	Poin yang diujikan
1	*Login* dan *logout*	- *Username* dan *password admin website*	- *Admin website* dapat masuk/*login* ke halaman *private facing website*. - *Admin website* dapat keluar/*logout* dari halaman *private facing website*.
2	Pengaturan entitas utama bagian 1	- *Admin website* sudah masuk ke *private facing website* - Menentukan entitas yang akan dikelola (artikel, produk, dll)	- *Admin website* dapat menambah *data* baru entitas utama - *Admin website* dapat mengubah *data* entitas utama - *Admin website* dapat menghapus *data* entitas utama - Melihat kesesuaian data yang ditampilkan pada public facing website
3	Pengaturan entitas utama bagian 2	- *Admin website* sudah masuk ke *private facing website* - Masuk ke menu pengaturan entitas produk dan artikel	- Validasi range harga produk, minimum harga < maksimum harga - Validasi stok produk, stok tidak boleh bernilai negatif - Validasi tanggal terbit artikel. - Melihat kesesuaian data yang ditampilkan pada public facing website
4	Pengaturan entitas *parameter*	- *Admin website* sudah masuk ke *private facing website* - Menentukan level entitas parameter (level 1-4)	- Pengujian entitas parameter yang dapat ditambah (seperti link media sosial, gambar, dll) - Pengujian entitas parameter yang dapat dihapus (seperti link media sosial, gambar, dll) - Pengujian entitas parameter yang hanya dapat diubah (seperti profil perusahaan, deskripsi alamat, dll) - Melihat kesesuaian data yang ditampilkan pada public facing website

3.4.3 Pengujian *development time*

Pengujian *development time* dilakukan untuk mendapatkan pembanding antara pengembangan aplikasi website dengan arsitektur 1 domain memiliki 1 *public facing website* dan 1 *private facing website* dan solusi arsitektur 1 *private facing website* dapat mengelola banyak *public facing website*. Parameter pada pengujian ini adalah *developer* yang melakukan pengujian berjumlah 1 orang dan waktu dihitung dalam satuan jam. Fitur yang dibangun sebagai pembanding sama dengan fitur utama aplikasi website yang didefinisikan. Sebagai skenario pengujian, *developer* diminta menduplikasi sebuah *website* dengan fungsi penuh (termasuk halaman pengaturan/*private facing website*) dengan menggunakan 2 arsitektur yang dibandingkan.

4. PEMBAHASAN

4.1 Implementasi aplikasi

4.1.1 Fitur utama

Berdasarkan hasil observasi dan wawancara kepada pengguna, didapatkan beberapa fitur utama yang harus dipenuhi pada aplikasi website yang dibangun. Daftar fitur dapat dilihat pada Tabel 3.

Tabel 3. Daftar fitur aplikasi *private facing website*

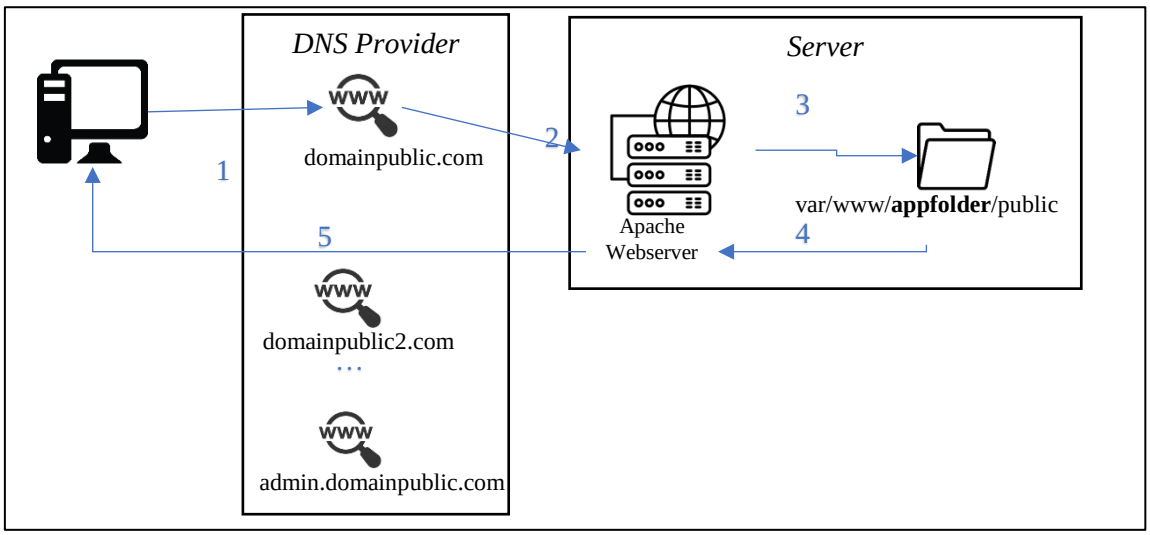
No	Fitur	Deskripsi
1	Pengaturan entitas website	Entitas website adalah entitas website yang terdaftar. Jumlah entitas website sama dengan jumlah *domain* atau *subdomain* yang digunakan.
2	Pengaturan administratif *user* dan hak akses	Pengaturan administratif *website* dilakukan untuk menambahkan daftar *user* untuk pengguna *login*. Hak akses harus dapat membedakan pengguna tidak *login*, pengguna *login* untuk mengelola salah satu *website* spesifik, dan pengguna *login* yang dapat mengelola semua website yang terdaftar melalui 1 *facing website*.
3	*Login* dan *logout*	Fitur *login* dan *logout* berada di *private facing website*. Digunakan untuk membedakan pengguna yang login dan tidak *login*. Serta membedakan *level* hak akses yang ada untuk masing-masing user.
4	Pengaturan entitas utama	Entitas utama adalah artikel, produk, dan portfolio.

p-ISSN: 2460-173X
e-ISSN: 2598-5841

		Sebuah *website* dapat memiliki ketiga entitas tersebut, atau hanya 2 dari 3 entitas tersebut, atau bahkan hanya 1 entitas saja. Hal ini disesuaikan dengan kebutuhan data yang mau ditampilkan pada website tersebut. Fitur pada *private facing website* adalah CRUD atau *Create Read Update Delete*. Fitur pada *public facing website* adalah *Read* saja atau melihat data yang sudah diatur melalui *private facing website*.
5	Pengaturan entitas parameter	Entitas parameter adalah entitas yang digunakan sebagai tampungan untuk *data* yang hanya terdiri dari 1 baris saja. Umumnya digunakan untuk pengaturan label pada *public facing website*. Setiap label dapat diubah, tetapi tidak semua label dapat ditambah atau dihapus. Tersedia 4 level parameter, yang membedakan adalah jumlah value yang dapat disimpan. Untuk *parameter level* 1, hanya 1 *value* yang dapat disimpan, untuk *parameter level* 2, hanya 2 *value* yang dapat disimpan, dst.
6	Menu dinamis pada *private facing website*	Menu dibuat secara dinamis untuk mengakomodasi perbedaan entitas utama untuk masing-masing website.

4.1.2 Request flow dan Arsitektur Aplikasi

Request flow aplikasi dari *browser* pengguna sampai dengan folder aplikasi yang sudah dibangun dapat dilihat pada Gambar 1. Pengguna mengisikan alamat pada browser, misalnya domain.com, kemudian *request* tersebut diterima oleh *DNS Provider* (1). Pada penelitian ini DNS Provider yang digunakan adalah *cloudflare*. DNS Provider yang menentukan alamat ip dari server tempat aplikasi website yang menangani domain.com dan meneruskan request ke *server* (2). *Request* tersebut diterima oleh *webserver* Apache pada komputer *server*. Pada penelitian ada 2 *server* yang digunakan, yaitu windows *server* yang digunakan untuk pengembangan aplikasi dan linux *server* yang digunakan untuk pengujian kepada pengguna. *Webserver* bertugas melakukan pengecekan domain mana yang diminta dengan daftar virtual host pada webserver (3). Selanjutnya *webserver* mengarahkan *request* ke *document root folder* dari aplikasi yang diakses, dan selanjutnya aplikasi website yang akan memproses *request* tersebut. Setelah halaman tersedia, struktur halaman dalam bentuk teks HTML dikirimkan ke pengguna melalui *webserver* dan internet (4 & 5).

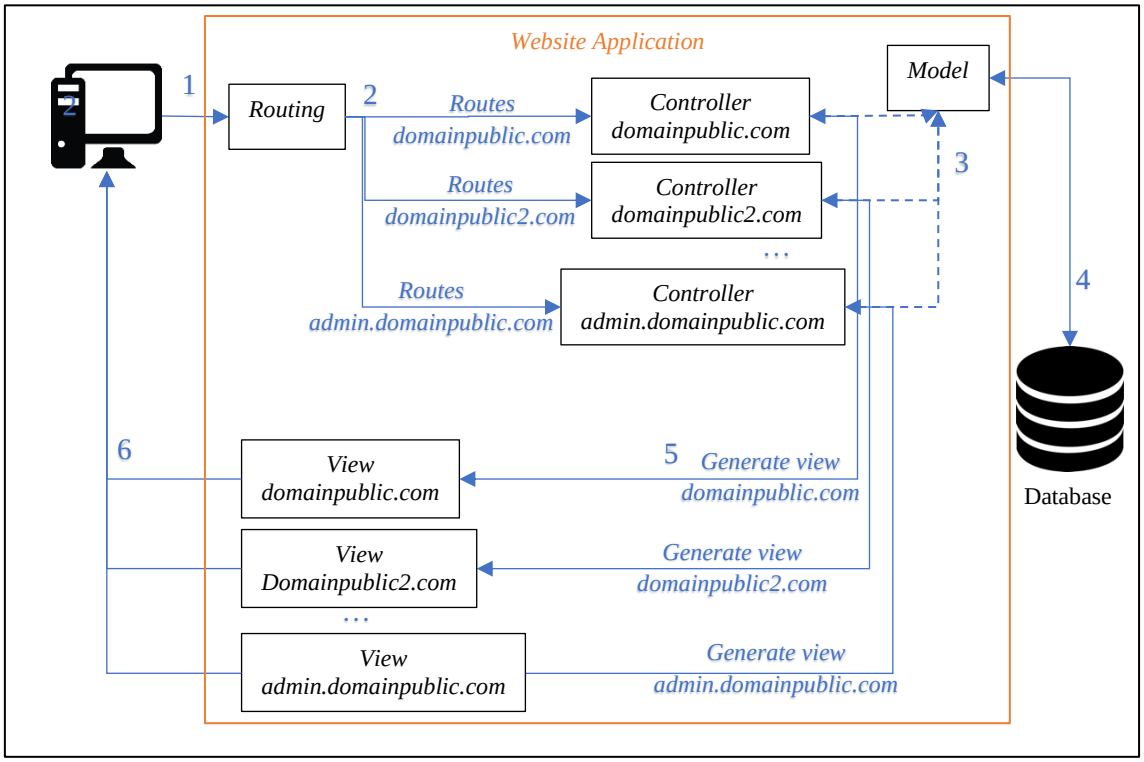


Gambar 1. Request flow dari browser ke aplikasi *website multi-domain*

Pada aplikasi *website* yang dibangun menggunakan *Laravel frameworks*, *request* dari pengguna diterima oleh *routing* (1). *Routing* pada *Laravel frameworks* bertugas menentukan kelas *Controller* yang bertugas mengelola *request* dan memberikan *response* (2). Kelas Controller untuk masing-masing *domain* aplikasi *website* akan berbeda karena logika yang dibangun juga berbeda satu sama lain. Seperti misalnya, domainprivate.com digunakan untuk *private facing website* dan menampilkan *form* pengelolaan entitas artikel. Sedangkan domainpublic.com digunakan untuk *public facing website* menampilkan entitas artikel kepada pengguna umum. Selanjutnya apabila diperlukan, *Controller* akan mengakses *Model* atau Eloquent *Object*

p-ISSN: 2460-173X
e-ISSN: 2598-5841

Relational Mapping (ORM) untuk mendapatkan data dari database (3 & 4). Dengan menggunakan *data* tersebut, *Controller* akan membangun *View* secara dinamis berdasarkan *blade templating engine* untuk masing-masing *facing website* (5). View tersebut dikirimkan berupa text dengan format HTML ke pengguna (6). Ilustrasi untuk pemrosesan request pada aplikasi website yang dibangun dapat dilihat pada Gambar 2.



Gambar 2. Arsitektur aplikasi *website* menggunakan *Laravel frameworks*

4.1.3 Hak akses

Aplikasi yang dibangun memiliki 3 level hak akses pengguna, yaitu pengguna umum, *admin website*, dan *super admin*. Daftar hak akses dan fitur yang dapat diakses dapat dilihat pada Tabel 4.

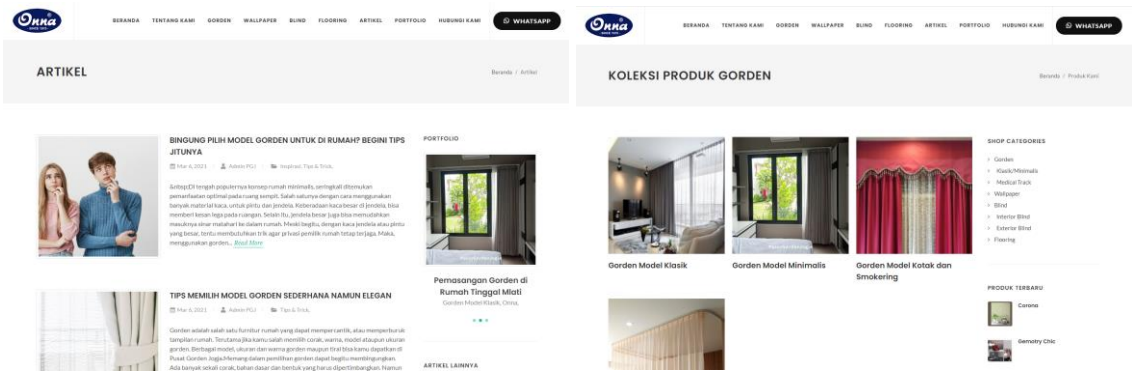
Tabel 4. Hak akses dan fitur yang dapat diakses

No	Hak akses	Deskripsi	Fitur yang dapat diakses
1	Pengguna Umum	Pengguna tidak *login*	- Mengakses semua halaman *public facing website*
2	*Admin Website*	Pengguna *login*, hanya dapat mengakses pengelolaan spesifik website.	- Semua fitur dari pengguna umum - Mengakses halaman pada *private facing website* dan melakukan pengaturan khusus untuk *website* spesifik.
3	*Super Admin*	Pengguna login, dapat mengakses pengelolaan semua *website*	- Semua fitur dari pengguna *Admin website* - Mengakses pengaturan daftar *website*, pengaturan daftar pengguna - Mengakses pengaturan untuk semua *website*.

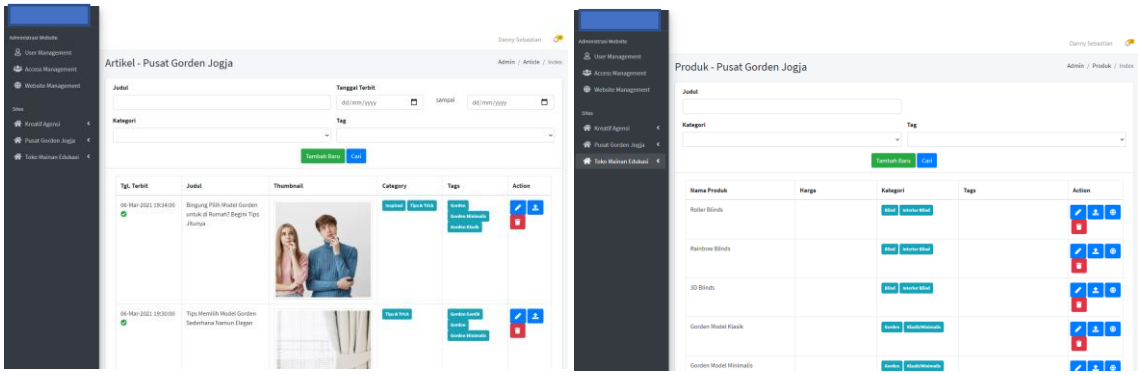
4.1.4 Antarmuka aplikasi website

Secara umum aplikasi memiliki 2 muka utama, yaitu *public facing website* dan *private facing website*. Beberapa halaman yang ada pada salah satu *public facing website* dapat dilihat pada Gambar 3. Gambar 3 sebelah kiri merupakan halaman pencarian artikel, sedangkan gambar sebelah kanan merupakan halaman pencarian produk. Sedangkan pada Gambar 4 adalah gambar *private facing website* halaman pengaturan artikel dan halaman pengaturan produk.

p-ISSN: 2460-173X
e-ISSN: 2598-5841



Gambar 3. Antarmuka *public facing website*, halaman pencarian artikel (kiri) dan halaman pencarian produk (kanan)



Gambar 4. Antarmuka *private facing website*, halaman pengaturan artikel (kiri) dan halaman pengaturan produk (kanan)

4.2 Hasil pengujian aplikasi

4.2.1 Pengujian pada lingkup pengembang

Pengujian pada lingkup pengembang dilakukan untuk memastikan program berjalan tanpa adanya *error* atau *bugs*. Pengujian dilakukan menggunakan phpunit yang merupakan fitur bawaan *Laravel frameworks* untuk melakukan *automated testing*. Pengujian dilakukan pada level fungsi, dengan *data* yang digunakan disesuaikan untuk semua kemungkinan *input data*.

Tabel 5 merupakan contoh skenario *automated testing* yang dilakukan. Pada contoh, pengujian dilakukan ke kelas *ArticleController* dengan fungsi *store()*. Fungsi *store()* digunakan untuk menyimpan *data* artikel berdasarkan *input* dari pengguna. *Data input* disesuaikan dengan semua kemungkinan yang ada, seperti ada *data* kosong, ada *data* yang hanya bisa berisi angka, ada *data* yang diisikan lebih dari satu, dan lain sebagainya.

Tabel 5. Beberapa skenario pengujian *automated testing* pada skenario menyimpan artikel baru

No	Skenario	Kelas/Fungsi	*Data*	*Expected Result*	Hasil
1	Menyimpan artikel baru	ArticleController store()	*Data* benar: *publish_date, publisher, title, link, content, categories, tags* *Data* salah: -	Berhasil menyimpan *data* artikel	Berhasil menyimpan *data* artikel (**OK**)
2	Menyimpan artikel baru	ArticleController store()	*Data* benar: *publish_date, publisher, title, link, content* *Data* salah: *categories* atau *tags* dikosongi	Berhasil menyimpan *data* artikel	Berhasil menyimpan *data* artikel (**OK**)
3	Menyimpan artikel baru	ArticleController store()	*Data* benar: *publish_date, publisher, categories, tags* *Data* salah: *title*, *link*, atau *content* kosong	Gagal menyimpan *data* artikel, karena *title*, *link*, dan *content* tidak bolek kosong.	Gagal menyimpan *data* artikel, karena *title*, *link*, dan *content* tidak bolek kosong (**OK**)

4	Menyimpan artikel baru	ArticleController store()	*Data* benar: *publish_date, publisher, title, link, content* *Data* salah: - *publish_date* lebih dari hari ini	Berhasil menyimpan *data* artikel. Status artikel belum terbit.	Berhasil menyimpan *data* artikel. Status artikel belum terbit. (**OK**)

Berdasarkan hasil pengujian *automated testing*, semua skenario pengujian untuk semua kelas dan fungsi dinyatakan berhasil. Karena semua skenario dapat dilakukan dengan baik tanpa adanya beda antara *expected result* dengan *actual result*. Berdasarkan hasil pengujian automated testing, pengujian dapat dilanjutkan ke tahap pengujian *user acceptance test*.

4.2.2 Pengujian *user acceptance test*

Pengujian *user acceptance test* atau pengujian yang melibatkan pengguna bertujuan untuk memastikan aplikasi yang dibangun sudah sesuai dengan kebutuhan dari pengguna dan dapat digunakan untuk kondisi aktual. Pada pengujian ini, dilakukan semua skenario yang memungkinkan pada kondisi aktual sehari-hari pengguna. Mulai dari *login* ke *private facing website*, mengelola entitas-entitas pada *public facing website* melalui *menu* pada *private facing website*, sampai dengan mengganti label-label non-entitas pada *public facing website*. Semua scenario tersebut dilakukan menggunakan kombinasi dari level hak akses pengguna yang ada. Beberapa skenario yang sudah dilakukan oleh pengguna dapat dilihat pada Tabel 6.

Tabel 6. Beberapa skenario pengujian *private facing website* oleh pengguna

No	Judul Skenario	Tujuan	Langkah yang dilakukan pengguna	Hasil
1	*Login* menggunakan akun level *super admin*	Menguji *menu* yang dapat diakses oleh *super admin*	1. Membuka alamat domain aplikasi *private facing website* 2. Memasukkan *username* dan *password* milik *user super admin*.	*Menu* yang dapat diakses adalah *menu* milik semua *website*
2	Menambahkan artikel menggunakan akun level *super admin*	Menguji proses penambahan artikel untuk *website* spesifik menggunakan akun level *superadmin*	1. *Login* menggunakan akun *super admin* ke halaman *private facing website* 2. Pilih menu Artikel pada salah satu *website* spesifik, klik tombol tambah baru, kemudian halaman *form* tambah artikel baru akan muncul. 3. Isikan data artikel, simpan.	Artikel yang ditambahkan muncul di halaman *public facing website* spesifik
3	Menghapus kategori yang masih digunakan artikel/produk/portfolio menggunakan akun *level admin website*.	Menguji apakah kategori yang masih digunakan dapat dihapus	1. *Login* menggunakan akun *admin website* ke halaman *private facing website* 2. Masuk ke menu kategori, kemudian pilih salah kategori yang masih digunakan oleh artikel. 3. Klik tombol hapus	Kategori yang masih digunakan pada artikel masih dapat dihapus. Seharusnya tidak dapat dihapus
4	Menambahkan lebih dari 1 kategori untuk produk menggunakan akun *level super admin*	Menguji fitur 1 artikel tergabung ke banyak kategori untuk salah satu domain website	1. Login menggunakan akun *level super admin* ke halaman *private facing website*. 2. Pilih salah satu website yang memperbolehkan artikel tergabung ke banyak kategori. 3. Tambahkan artikel dan kelompokkan ke banyak kategori. Simpan 4. Lakukan Langkah 2 dan 3 untuk website yang hanya memperbolehkan satu artikel tergabung ke 1 kategori	Artikel dapat dikelompokkan ke beberapa kategori untuk salah satu website, sedangkan pada website lain hanya satu kategori

Dari semua skenario-skenario yang sudah dilakukan, ada beberapa skenario yang masih gagal dilaksanakan, seperti skenario nomor 3 pada Tabel 6. Skenario tersebut bertujuan untuk menguji validasi restriksi penghapusan kategori yang masih digunakan pada artikel atau produk atau portfolio. Pada pengujian oleh pengguna, aplikasi masih memperbolehkan penghapusan. Sehingga dicatat sebagai *bugs*, dan diperbaiki oleh *developer*. Setelah perbaikan dilakukan oleh *developer*, pengguna melakukan pengujian kembali tanpa perlu didampingi oleh *developer*. Pada tahap pengujian kedua, sudah tidak ditemukan *bugs* oleh pengguna sehingga aplikasi dinyatakan lulus proses pengujian oleh pengguna.

4.2.3 Pengujian *development time*

Pengujian development time dilakukan untuk mendapatkan gambaran perbandingan waktu yang dibutuhkan untuk pengembangan aplikasi menggunakan solusi 1 *private facing website* dapat digunakan untuk banyak *domain*/*public facing website*. Daftar perbandingan pekerjaan untuk *developer* dalam mengembangkan masing-masing aplikasi dapat dilihat pada Tabel 7.

Tabel 7. Perbedaan pengembangan aplikasi website biasa dan solusi pada penelitian ini

Bagian website yang dibangun	Pengembangan aplikasi *website* 1 *domain* memiliki 1 *public facing website* dan 1 *private facing website*	Solusi pada penelitian ini 1 *private facing* dapat digunakan untuk banyak *domain*/*public facing website*
Kode program yang sudah tersedia sebagai asset	- Template HTML *public facing website* dan *private facing website* - Contoh aplikasi website sejenis.	- Template HTML *public facing website* dan *private facing website* - Obyek entitas untuk Model Eloquent - Fungsi dasar
Public facing website, *developer* membuat membuat aplikasi *website* yang menampilkan data dari *database*	- Menduplikasi struktur database untuk masing-masing entitas dari contoh aplikasi website sejenis - Menduplikasi obyek untuk masing-masing entitas dari contoh aplikasi website sejenis - Mengubah *template HTML* menjadi bentuk dinamis	- Mengubah *template HTML* menjadi bentuk dinamis - Membuat *routing* pada *Laravel Frameworks* untuk *public facing website.*
P*rivate facing website*, *developer* membuat aplikasi *website* pengelolaan *data* pada *database*	- Menggunakan struktur *database* dan obyek masing-masing entitas pada tahap pengembangan *public facing website*. - Mengubah *template HTML* menjadi bentuk *form* dinamis untuk pengelolaan *data* - Membuat fungsi pengelolaan *data* untuk masing-masing entitas.	- Tidak membuat kode program apa-apa kare sudah tersedia. - Melakukan pengaturan data awal pendaftaran website baru. - Mendaftarkan parameter data awal.
Waktu yang dibutuhkan	53 jam atau 6.625 hari kerja* *Asumsi 1 hari kerja = 8 jam.	16 jam atau 2 hari kerja*

Berdasarkan hasil pengujian development time, waktu pengembangan aplikasi website dengan arsitektur 1 *domain* memiliki 1 *public facing website* dan 1 *private facing website* adalah 53 jam, atau setara dengan 6.625 hari kerja. Dengan asumsi 1 hari kerja adalah 8 jam. Hasil aplikasi *website* yang didapat masih belum sempurna karena belum menerapkan validasi pada setiap *form* entitas yang ada, dan masih belum membuat fungsi pengelolaan pengguna.

Sedangkan pada arsitektur 1 *private facing website* dapat digunakan untuk banyak *domain/public facing website* membutuhkan *development time* sebanyak 16jam atau setara dengan 2 hari kerja. Pada proses pengembangan, developer dapat fokus kepada pengembangan public facing website dengan memanfaatkan entity yang memang sudah tersedia tanpa perlu melakukan duplikasi. Pada proses pengaturan *private facing website*, *developer* tidak perlu membuat kode program karena solusi pada penelitian ini sudah dapat memfasilitasi penambahan website dengan hanya mengisi data pada *database*.

Berdasarkan pengujian, disimpulkan solusi arsitektur 1 *private facing website* untuk banyak *public facing website/domain* dapat mengurangi *development time*. Perbandingan *development time* adalah 53jam dibandingkan dengan 16jam, atau $^{1}/_{3,3125}$. *Developer* dapat fokus ke pengembangan *public facing website* baru. Solusi ini cocok untuk pengembangan aplikasi dengan fitur yang sejenis atau didefinisikan sama.

5. KESIMPULAN DAN SARAN

Kesimpulan dari penelitian ini adalah:

- Aplikasi *website* dengan arsitektur *single private facing website* untuk *multiple domain* yang dibangun sudah lulus pengujian pada lingkup pengembang menggunakan *unit testing Laravel Frameworks*.
- Aplikasi *website* dengan arsitektur *single private facing website* untuk *multiple domain* yang dibangun sudah lulus pengujian oleh pengguna atau *user acceptance test* dan

pengguna dapat mengelola banyak *public facing website/domain* melalui 1 *website* pengelolaan.

- Aplikasi *website* dengan arsitektur *single private facing website* untuk *multiple domain* yang dibangun dapat mengurangi *development time*. Perbandingan development time yang dihasilkan adalah $^{1}/_{3,3125}$.

DAFTAR PUSTAKA

[1] D. Sebastian and K. A. Nugraha, "Pendampingan Pengembangan Sistem Informasi Antrian Disabilitas Rumah Sakit Panti Waluyo Purworejo," *Jurnal PATRIA*, vol. 3, no. 1, pp. 32-41, 2021.

[2] M. Sulayman, C. Urquhart, E. Mendes and S. Seidel, "Software process improvement success factors for small and medium Web companies: A qualitative study," *Information and Software Technology*, vol. 54, no. 5, pp. 479-500, 2012.

[3] D. Sebastian, K. A. Nugraha and M. N. A. Rini, "Pendampingan Pembangunan Aplikasi Penilaian Guru SMA Kolese De Britto," in *Prosiding Seminar Nasional Hasil Pengabdian kepada Masyarakat (Sendimas 2018)*, Jakarta, 2018.

[4] M. Stauffer, Laravel: Up & Running: A Framework for Building Modern PHP Apps, O'Reilly Media, 2019.

[5] A. K. Himawan, "Performance analysis framework codeigniter and CakePHP in website creation," *International Journal of Computer Applications*, vol. 94, no. 20, 2014.

[6] A. B. Warsito, M. Yusup and Yulianto, "Kajian Yii Framework dalam Pengembangan Website Perguruan Tinggi," *Creative Communication and Innovative Technology Journal*, vol. 7, no. 3, pp. 437-451, 2014.

[7] M. Laaziri, K. Benmoussa, S. Khoulji and M. L. Kerkeb, "A Comparative study of PHP frameworks performance," *Procedia Manufacturing*, vol. 32, pp. 864-871, 2019.

[8] X. Li, S. Karnan and J. A. Chishti, "An empirical study of three PHP frameworks," in *2017 4th International Conference on Systems and Informatics (ICSAI)*, 2017.

[9] Laravel LLC, "Laravel Documentation," 2020. [Online]. Available: https://laravel.com/docs/8.x/. [Accessed 13 03 2021].

Biodata Penulis

Danny Sebastian S.Kom., M.M., M.T., lahir di Pekalongan pada tanggal 26 November 1988. Meraih gelar Sarjana Komputer (S.Kom.) di Program Studi Informatika Universitas Kristen Duta Wacana pada tahun 2011, gelar Magister Manajemen (M.M.) di Universitas Pelita Harapan pada tahun 2014 dan gelar Magister Teknik (M.T.) di Universitas Atma Jaya Yogyakarta pada tahun 2016. Saat ini berprofesi sebagai dosen di Program Studi Informatika Universitas Kristen Duta Wacana.