# Implementasi Virtual Server Berbasis Container Pada Sistem Informasi Geografis Cagar Budaya Mojokerto

Muhammad Hussein Isron[1], Ricky Eka Putra[2]

[1,2] Jurusan Teknik Informatika Fakultas Teknik Universitas Negeri Surabaya
[1]muhammad.17051204065@mhs.unesa.ac.id
[2]rickyeka@unesa.ac.id

***Abstrak*— Sistem Informasi Geografis Cagar Budaya Kabupaten Mojokerto Merupakan Sistem informasi yang memuat detail dan lokasi cagar budaya di Kabupaten Mojokerto. Dalam sistem informasi geografis terdapat beberapa integrasi diantaranya maps dan database. Pembuatan SIG ini dilakukan menggunakan cara pengumpulan data melalui observasi baik secara offline di beberapa titik lokasi maupun daring melalui portal resmi Kemendikbud terkait cagar budaya yang ada di Mojokerto. Pada pengembangannya juga melakukan studi literatur yang mendukung penyelesaian masalah mengenai penyiapan lokal development server yang fleksibel, scaleable, efisien, dan mempunyai ketersediaan yang tinggi. Teknologi yang digunakan yaitu docker dengan arsitektur container. Pengembangan SIG dilakukan dalam docker container dan dibangun dengan arsitektur model view controller menggunakan penentuan titik lokasi google maps.**

***Penyiapan lokal development server terdiri dari beberapa requirement yang dibutuhkan untuk pengembangan SIG, Apache http server sebagai server menjalankan aplikasi, PHP sebagai bahasa pemrograman pembuatan aplikasi, MySQL sebagai rdbms penyimpanan data cagar budaya serta Adminer digunakan untuk mengelola administrasi database. Server tersebut dibangun dengan docker berbasis container. Pengujian pada sistem informasi geografis ini menggunakan blackbox testing dimana fungsi halaman pengunjung dan admin dapat berjalan dengan baik. Dilakukan juga evaluasi terhadap hasil pengembangan melalui docker container sebagai alternatif lokal development server yang efisien, fleksibel, scaleable dan memiliki ketersediaan yang tinggi.***

***Kata Kunci*— Virtualisasi Server, Docker Container, Sistem Informasi Geografis, Cagar Budaya, Model View Controller.***

## I. PENDAHULUAN

Di era modern ini perkembangan teknologi makin bertumbuh cepat, efisiensi dan efektifitas dalam pemanfaatan teknologi semakin dikedepankan. Pengembangan aplikasi berbasis web banyak dimanfaatkan dalam penyampaian informasi yang dapat dengan mudah diakses melalui berbagai perangkat. Aplikasi web dapat dikembangkan dengan membuat skrip kode dan menjalankan kode program yang disimpan(*hosting*) dalam sebuah server. *Web server* adalah aplikasi yang memiliki peran untuk menyalurkan halaman web ke pengguna, halaman web yang dikirimkan akan disesuaikan dengan permintaan dari pengguna. Menurut [1] mekanisme kerja *web server* ialah pada saat penelusur web(*web browser*) mengirim permintaan ke server maka permintaan data penelusur web tersebut dienkapsulasi dalam *Transmission Control Protocol (TCP)* dan dikirim ke protokol transfer jaringan internet yaitu *HTTP*.

Teknologi virtualisasi semakin marak digunakan dalam pengembangan web dikarenakan mudahnya dalam konfigurasi dan mempunyai mesin virtual yang terisolasi. Virtualisasi dapat didefinisikan sebagai kebalikan dari mesin fisik. Mesin fisik terdiri dari berbagai komponen fisik seperti *power supply*, *mainboard*, *processor*, *memory, disk*, dan sebagainya yang menjadi sebuah komputer untuk dapat melakukan komputasi secara fungsional. Virtualisasi menjadikan suatu komputasi dapat menjalankan program yang sama layaknya komputer, tetapi sebenarnya virtualisasi berjalan diatas mesin lain. Virtualisasi memungkinkan untuk menjalankan beberapa *operating system* dan *service* pada satu perangkat keras fisik dalam satu waktu. Hal tersebut menjadi fundamental implementasi virtualisasi server dapat menjadi solusi yang baik untuk menghemat *resources* penyediaan infrastruktur serta meningkatkan efisiensi penggunaan daya dan biaya.

Teknologi virtualisasi dapat dibagi menjadi dua jenis, yang pertama adalah mesin virtual berbasis *hypervisor* dan lainnya adalah virtualisasi *container*. Dalam virtualisasi berorientasi *hypervisor* sebagai pemisah perangkat keras dari perangkat lunak untuk menghasilkan efektivitas sistem yang lebih baik. *Container* dapat menjadi titik temu kinerja aplikasi sehubungan dengan aksesibilitas dan skalabilitas [2]. Virtualisasi mesin mempunyai sedikit kelemahan diantaranya *uptime* yang memakan waktu untuk mengkluster *web server*. Hal tersebut mengakibatkan munculnya berbagai macam teknik dan model sebagai sarana meningkatkan efisiensi, skalabilitas dan elastisitas. Teknologi baru yang dapat diadopsi adalah *containerization*. *Containerization* adalah teknologi yang mengemas aplikasi, dependensi terkait dan *library* sistem yang terorganisir untuk membangun layanan dalam bentuk wadah atau *container*. Aplikasi yang dibangun dan diatur dapat dijalankan serta digunakan sebagai sebuah wadah [3]. Teknologi *container* berbeda dengan mesin virtual. Mesin virtual membagi *resources hardware* untuk dialokasikan kedalam *operating system(OS)* yang dijalankan pada mesin virtual, sedangkan *container* berjalan pada *os host* yang sama. *Container* membungkus aplikasi serta dependensi yang dibutuhkan dalam beberapa konfigurasi untuk dapat menjalankan *service*. *Container* berjalan pada *os host* linux yang dapat menjalankan *os* linux lain dalam *container*. Metode perancangan virtualisasi *server* dengan tipe *cluster high availability* memberikan kemudahan dalam hal instalasi dan penggunaannya serta memiliki *uptime* yang tergolong cepat

daripada virtualisasi mesin. Teknologi *container* yang *popular* adalah Docker.

Pengembangan pada *web server* menggunakan docker cukup banyak digunakan dikalangan *developer*. Docker ialah suatu proyek *open-source* yang memberi keterbukaan bagi *developer* untuk dapat membangun, mengemas, dan menjalankan aplikasi dalam sebuah wadah atau *container* [4]. Docker mempunyai arsitektur *client* dan server. Docker *client* mengirimkan permintaan atau *request* ke docker daemon untuk membangun, mendistribusikan, dan menjalankan docker *container* [5]. Docker merupakan mekanisme virtualisasi tingkat *operating system* yang memungkinkan wadah linux untuk dieksekusi secara independen terhadap semua *host* di lingkungan yang terisolasi [6]. Docker menyediakan beberapa *tools* yang dapat digunakan secara sistematis menggunakan konsep *container*. Salah satunya adalah kemampuan untuk memeriksa *container* dan versinya, untuk membedakan dua *container* sehingga dapat dengan mudah memperbaiki aplikasi jika diperlukan [7].

Docker dapat dikonfigurasikan untuk membuat objek, objek yang terdapat pada docker antara lain *images, container,* dan *service*. Objek tersebut yang menjadi orkestrasi dalam proses *containerization*. Penerapan aplikasi ke dalam *platform* docker hanya membutuhkan mempersiapkan *file* instruksi (bernama dockerfile) tanpa mengubah infrastruktur aplikasi yang tidak diperlukan [8]. *Client* docker dan docker daemon dapat berjalan pada *host* yang sama serta dapat berkomunikasi menggunakan *request API*, melalui soket maupun jaringan. Docker meminimalisir penyediaan *resource* yang tidak dibutuhkan dalam *deployment* aplikasi. Docker mempunyai portabilitas yang tinggi dalam hal *uptime* dan *deployment* aplikasi, layanan *container* hanya membutuhkan waktu beberapa detik untuk melakukan penyediaan layanan(*uptime*), docker juga mempunyai repositori *online*(docker hub) yang bisa menyimpan image(bahan *deployment*) untuk nantinya dapat diunduh dan diaplikasikan kedalam *container* sehingga memudahkan dalam *production server*. *Deployment* pada docker memiliki *scalability* tinggi yang dapat diskalakan secara *horizontal*(menjadi beberapa server), server dapat direplikasi menjadi beberapa *node* sehingga apabila server yang lain mengalami *crash* masih ada server lain yang melayani *request*. Secara teknis, layanan web disediakan oleh aplikasi *web server* seperti Apache dan Nginx. Secara default, setiap *web server* menyediakan situs web untuk satu domain tertentu [9]. Kemudian aplikasi website bisa dikembangkan menggunakan bahasa pemrograman PHP, Javascript dan sebagainya. Semua dependensi yang dibutuhkan tersebut dipaketkan dalam *container* docker menggunakan beberapa konfigurasi diantaranya dockerfile dan YAML.

Setelah lingkungan pengembangan disiapkan, penulis dapat melakukan pengembangan aplikasi web sistem informasi geografis pada docker *container*. Pengembangan aplikasi menggunakan Bahasa PHP *framework* codeigniter dengan arsitektur *model view controller*. Menurut [10], Codeigniter merupakan sebuah *framework* atau kerangka kerja yang dibuat menggunakan bahasa pemrograman PHP yang mempunyai tujuan untuk memudahkan *programmer* web dalam membuat atau mengembangkan aplikasi berbasis web. Codeigniter merupakan alat bantu bagi *developer* yang ingin membangun aplikasi web menggunakan bahasa PHP. *MVC* merupakan kepanjangan dari *Model View Controller* yang merupakan alur pengembangan *framework* codeIgniter. Dengan adanya konsep *MVC*, logika program dan desain *layout* menjadi bagian yang terpisah, sehingga pengerjaan dapat dilakukan secara fokus. Penelitian yang dilakukan oleh [10] mampu mengimplementasikan konsep *MVC* dalam pengembangan sistem informasi geografis dengan *framework* codeigniter.

Pada sistem informasi geografis cagar budaya ini terdapat pemetaan titik lokasi serta sejumlah informasi mengenai cagar budaya yang ada di Kabupaten Mojokerto khususnya Kecamatan Trowulan. Sebagian besar cagar budaya yang telah ditemukan berlokasi di Kecamatan Trowulan. Kantor Balai Pelestarian Cagar Budaya Kabupaten Mojokerto juga berada di Kecamatan tersebut sehingga diharapkan dapat memudahkan penggalian informasi terkait cagar budaya di Kabupaten Mojokerto. Kabupaten Mojokerto mempunyai sejarah historis yang kuat mengenai kerajaan Majapahit. Hasil verifikasi dan validasi cagar budaya Kabupaten Mojokerto oleh Kemendikbud tahun 2016 menyatakan terdapat 10 cagar budaya yang telah diverifikasi, 1 cagar budaya di Kecamatan Trawas dan 9 cagar budaya di Kecamatan Trowulan [11]. Telah ditemukan beberapa situs cagar budaya baru di Kecamatan Trowulan dalam dua tahun terakhir. Hal ini memungkinkan situs cagar budaya masih banyak yang dapat ditemukan kembali di Kabupaten Mojokerto. Dengan dikembangkannya aplikasi ini diharapkan juga dapat membantu pemetaan dan penginformasian cagar budaya di Kabupaten Mojokerto.

Penelitian sebelumnya yang juga menjadi salah satu acuan dilakukan oleh [5] yang berhasil merancang arsitektur *web server* pada *container* dan mengimplementasikan *Lightweight Virtualization* dengan menggunakan *Linux Containers (LXC)* serta *deployment* aplikasi web menggunakan dockerfile, setelah dilakukan pengujian dimana layanan web berjalan sesuai konfigurasi yang diterapkan. Dalam pengujian tersebut dapat dihasilkan bahwa suatu *container* tidak harus membutuhkan *config resource* mesin virtual seperti di komputer yang terdapat *os* sepenuhnya. *Container* diciptakan supaya dapat menggunakan *resource* seperlunya agar menjadi ringan. *Installer* linux pada mesin virtual berformat ISO sedangkan *container* menggunakan format tar yang mana merupakan kompresi dalam pengemasan dependensi yang diperlukan saja sehingga ukuran file lebih sedikit.

Berdasarkan analisis dari topik penelitian tersebut, peneliti ingin melakukan penelitian mengenai "Implementasi *Virtual Server* Berbasis *Container* Pada Sistem Informasi Geografis Cagar Budaya Mojokerto". Penelitian ini mengimplementasikan arsitektur virtualisasi *web server* berbasis docker *container* menggunakan sistem operasi windows serta mengembangkan sistem informasi geografis menggunakan arsitektur *model view controller* pada lingkungan docker *container* sehingga diharapkan dapat melakukan pengembangan aplikasi web yang lebih efisien, fleksibel dan mempunyai ketersediaan yang tinggi (*high availability*).

## II. METODE PENELITIAN

Jenis penelitian ini merupakan penelitian rekayasa atau pengembangan, yaitu pengembangan sistem informasi geografis pada server lokal menggunakan docker *container*. Metode adalah suatu prosedur yang dirancang serta digunakan untuk mencapai suatu tujuan tertentu. Beberapa tahap yang telah dilakukan dalam perancangan server dan pembuatan aplikasi tersebut, yaitu:

### *A. Analisa Kebutuhan*

Dalam pengembangan aplikasi ini, terdapat beberapa kebutuhan yang perlu dianalisa. Kebutuhan tersebut akan digunakan sebagai bahan untuk membantu pengembangan aplikasi. Analisa kebutuhan dibagi menjadi beberapa bagian, yaitu:

#### *1. Kebutuhan Data*

Data yang diperlukan untuk penelitian ini diambil dari beberapa referensi. Untuk pengembangan sistem informasi geografis data cagar budaya diambil dari dokumen verifikasi dan validasi cagar budaya oleh Kemendikbud, situs registrasi cagar budaya Kemendikbud serta pengambilan data ke beberapa cagar budaya di Mojokerto. Perancangan server pengembangan lokal merujuk pada dokumentasi resmi docker. Pengumpulan data pada penelitian ini terbagi menjadi dua jenis, yaitu studi literatur dan observasi.

*a. Studi literatur*

Pada penelitian ini mengambil referensi dari berbagai macam literatur yang relevan dengan pengembangan aplikasi pada docker dan sistem informasi geografis untuk mendalami pengetahuan. Literatur yang digunakan diantaranya laporan, jurnal, makalah, situs resmi dan sumber dari internet.

*b. Observasi*

Penelitian ini juga melakukan observasi dengan mengunjungi beberapa cagar budaya di Kabupaten Mojokerto dan mengunjungi pula situs resmi Kemendikbud serta situs dokumentasi docker.

#### *2. Kebutuhan Alat*

Spesifikasi perangkat yang diperlukan guna mendukung penelitian dalam melakukan perancangan dan pengembangan sistem adalah :

a. Processor AMD Dual Core A9-9420 3.0 GHz
b. RAM 8 GB
c. Hardisk 500 GB
d. Sistem Operasi Windows 10 64-bit (*support* virtualisasi hyper-v atau terinstal windows subsystem for linux)

Sedangkan perangkat lunak yang diperlukan dalam penelitian ini adalah sebagai berikut :

a. Docker desktop sebagai media dalam membangun aplikasi dalam *container*
b. WSL2(Windows Subsystem for Linux) sebagai media komunikasi *backend* linux dengan *host OS* pada windows
c. Code Editor(VSCode) sebagai alat untuk menulis kode program

### *B. Desain Sistem*

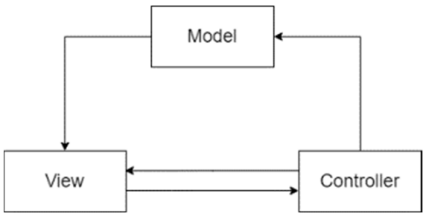
Sistem yang akan dikembangkan ialah sistem informasi geografis. Virtualisasi server lokal memberikan dampak pada proses pengembangan aplikasi yang modern, terisolasi dan efisien. Didalam desain sistem tahap-tahap yang dilakukan adalah sebagai berikut :

#### *1. Arsitektur Pengembangan Lokal*


Gbr 1. Arsitektur server pengembangan lokal

Vortualisasi server pengembangan lokal yang akan menjadi tempat pembuatan sistem informasi geografis berbasis docker *container*. *Image* yang dibangun dari docker memuat php dengan apache sebagai *web server* untuk menjalankan website menggunakan bahasa pemrograman php. Kemudian mysql sebagai sistem manajemen database relasional(RDBMS) dan adminer untuk mengelola konten atau data dalam database. Konfigurasi pembangunan *image* pada docker berisi masing-masing *environtment* yang sudah dirancang, kemudian dilakukan *build* image melalui dockerfile tersebut yang dijalankan pada docker. Setiap image tersebut didefinisikan dalam docker compose untuk mengintegrasikan multi *container* pada docker.

#### *2. Pola Pengembangan Sistem*



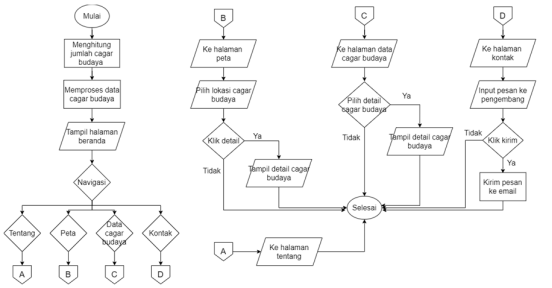

Gbr 2. Pola pengembangan sistem

Pengembangan sistem informasi geografis dilakukan menggunakan *design pattern Model View Controller* (MVC). Model mendefinisikan data yang harus ada dalam aplikasi. Jika data tersebut berubah, maka model akan memberi tahu *view* sehingga tampilan dapat berubah sesuai kebutuhan dan model dapat dimanipulasi oleh *controller*. *View* digunakan untuk menampilkan data ke sisi klien. *View* juga dapat

melakukan permintaan aksi lewat *controller*. *Controller* berperan sebagai logika yang memperbarui model maupun tampilan sebagai respons terhadap masukan dari pengguna aplikasi. Bahasa pemrograman yang dipakai pada aplikasi ini ialah php dengan *framework* codeigniter.

3. *Flowchart dan UML Diagram*

Untuk memudahkan dalam pengembangan aplikasi serta pengguna dapat dengan mudah memahami alur aplikasi dibuat perancangan alur dan proses diantaranya *flowchart, usecase* dan *class diagram*.




Gbr 3. Flowchart pengunjung

Gambar diatas merupakan alur program atau *flowchart* dari sistem informasi geografis cagar budaya untuk pengunjung. Tampilan dari sistem ini ialah *single page*, namun untuk halaman detail akan menampilkan laman baru agar detail cagar budaya tersampaikan dengan terperinci.

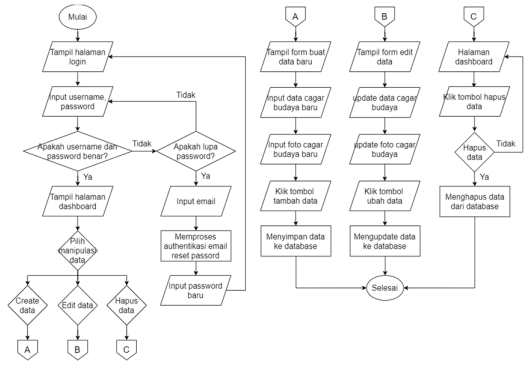
Sebelum menampilkan halaman utama sistem akan melakukan kalkulasi jumlah cagar budaya yang terverifikasi dan belum terverifikasi. Sistem juga akan memuat semua data cagar budaya untuk ditampilkan pada halaman peta dan data cagar budaya. Terdapat beberapa navigasi pada halaman beranda, diantaranya tentang, peta, data cagar budaya, dan kontak pengembang.

Ketika pengunjung menekan navigasi tentang maka akan dialihkan ke halaman tentang, begitu pula dengan navigasi lainnya akan berpindah ke halaman masing-masing. Halaman tentang berisi deskripsi beserta fitur dari sistem informasi geografis cagar budaya tersebut.

Ketika pengunjung menekan navigasi peta. Halaman akan dialihkan ke peta dari cagar budaya yang ada di Kabupaten Mojokerto. Titik lokasi yang ada di peta dapat diklik untuk melihat detail dari cagar budaya tersebut. Pengunjung dapat melihat deskripsi, informasi, dan foto cagar budaya.

Navigasi selanjutnya ialah data cagar budaya, ketika pengunjung menekan navigasi tersebut maka halaman akan dialihkan ke daftar data cagar budaya yang ada di Kabupaten Mojokerto. Data tersebut ditampilkan secara ringkas pada halaman ini, ketika pengunjung klik detail maka halaman akan dialihkan menuju halaman detail cagar budaya.

Navigasi terakhir adalah kontak. Ketika pengunjung menekan navigasi kontak maka akan dialihkan ke halaman kontak. Pengunjung dapat menginput pesan yang ingin disampaikan kepada pengembang, jika seluruh inputan sudah selesai, pengunjung bisa menekan tombol kirim untuk megirim pesan tersebut kepada pengembang.




Gbr 4. Flowchart admin

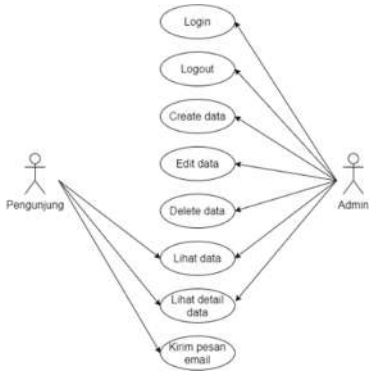
Pada gambar 4 dijelaskan gambar flowchart dari admin. Untuk memperoleh *privilege* sebagai admin harus login terlebih dahulu, setelah berhasil login admin dapat melakukan manipulasi data. Akun admin didaftarkan secara manual melalui fitur registrasi, setelah itu fitur registrasi disembunyikan. Terdapat fitur lupa password jika pengguna lupa password akun admin. Pengguna dapat memasukkan alamat email yang telah didaftarkan, setelah itu pengguna akan dikirimkan email untuk diminta mengisi password baru. Selanjutnya admin dapat melakukan login dengan kredensial baru.

Admin dapat menambahkan data cagar budaya baru, ketika klik tombol tambah data maka sistem akan menampilkan *form* input data cagar budaya baru yang terdiri dari nama, kode pengelolaan, deskripsi, alamat, desa, kecamatan, kabupaten, latitude, longitude dan foto cagar budaya. Kode pengelolaan didapatkan jika cagar budaya telah terdaftar atau terverifikasi oleh Kemendikbud. *Latitude* dan *longitude* didapatkan dari titik lokasi cagar budaya dalam maps. Setelah admin menekan tombol tambah data maka sistem akan menyimpan data tersebut dalam database.

Admin juga dapat mengupdate data melalui tombol edit data. halaman akan dialihkan ke *form* edit data. Data asli akan ditampilkan melalui form tersebut, namun apabila ada perubahan dapat memanipulasinya secara langsung. Ketika admin menekan tombol ubah data maka sistem akan memperbarui data cagar budaya tersebut ke database.

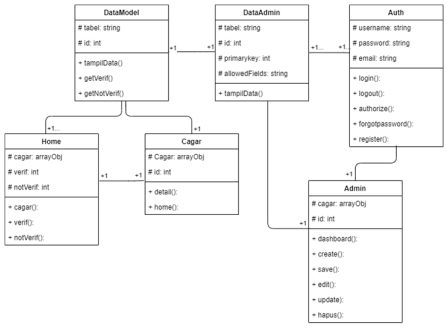
Hapus data dapat dilakukan oleh admin ketika menekan tombol hapus pada baris data cagar budaya yang ingin dihapus. Kemudian sistem akan menampilkan pesan konfirmasi hapus, tekan tidak untuk membatalkan hapus dan tekan ya untuk memproses hapus data. Sistem akan menghapus data cagar budaya tersebut pada database.

Pada gambar 5 merupakan usecase diagram dari sistem informasi geografis cagar budaya di Kabupaten Mojokerto.




Gbr 5. Usecase diagram

Dalam sistem ini terdapat dua aktor, pengunjung dan admin. Pengunjung merupakan pengguna yang secara langsung mengakses sistem tersebut, pengunjung hanya dapat melihat tampillah halaman, detail data cagar budaya dan mengirim pesan kepada pengembang. Sedangkan admin dapat melakukan login, logout, membuat data, mengedit data, menghapus dan juga melihat data cagar budaya. Proses manipulasi terhadap data cagar budaya tersebut tentunya harus melalui login terlebih dahulu.




Gbr 6. Class diagram

Pada gambar 6 dijabarkan struktur dari kelas yang ada pada sistem. Kelas tersebut berfungsi sebagai pengontrol utama alur pada sistem. Pada kelas home terdapat tiga atribut yakni cagar dalam bentuk array object, verif dan nonVerif yang memiliki tipe interger. kelas tersebut memiliki method cagar() yang memproses semua data cagar budaya, *method* verif() untuk menghitung data cagar budaya yang telah terverifikasi dan notVerif() untuk menghitung data cagar budaya yang belum terverifikasi. Kelas tersebut memiliki hubungan asosiasi terhadap kelas cagar dan dataModel. Kelas cagar dipanggil oleh kelas home untuk menampilkan detail cagar budaya, kelas cagar memiliki dua atribut yaitu cagar bertipe array object dan id bertpe interger. kelas cagar memiliki dua method, detail() dan home(). *Method* detail digunakan untuk menampilkan detail dari cagar budaya dengan parameter id cagar budaya, sedangkan *method* home() digunakan untuk kembali ke halaman beranda sistem. Kelas dataModel berhubungan dengan database, memiliki dua atribut yaitu tabel bertipe string dan id bertipe interger serta mempunyai tiga *method* yaitu tampilData(), getVerif dan getNotVerif(). Kelas dataModel dipanggil oleh kelas home dan cagar sebagai penghubung ke database.

Kelas admin merupakan kelas untuk mengontrol *behaviour* admin. Kelas admin memiliki dua atribut yaitu cagar bertipe array object dan id bertipe interger serta mempunyai beberapa *method* diantaranya dashboard(), create(), save(), edit(), update(), dan hapus(). Kelas admin mempunyai hubungan asosiasi dengan kelas auth dan dataAdmin. Admin harus memiliki kredensial terlebih dahulu dari kelas auth untuk mengakses kelas dataAdmin. Kelas auth berfungsi sebagai autentikasi dan authotisasi kelas admin. Kelas auth memiliki tiga atribut yaitu username bertipe string, password bertipe string dan email bertipe string serta mempunyai method login(), logout(), auth(), dan forgotpassword(). Kelas dataAdmin berperan sebagai penghubung ke database, memiliki atribut tabel dan allowedField bertipe string serta id dan primaryKey bertipe interger. kelas ini memiliki method tampilData() untuk menampilkan baik semua data maupun detail data. Kelas dataAdmin dan dataModel meiliki kelas parent yang sama.

### C. *Implementasi Sistem*

Setelah menganalisa data dan merancang sistem. Tahap selanjutnya ialah mengaplikasikan hasil rancangan dan desain yang telah dibuat untuk melakukan pengembangan aplikasi pada pola sistem tersebut. Perancangan arsitektur docker *container* sebagai server pengembangan lokal serta menerapkan *model view controller* sebagai pola pengembangan sistem dalam membangun sistem informasi geografis cagar budaya Kabupaten Mojokerto sesuai dengan data penelitian yang telah didapat.

### D. *Pengujian dan Evaluasi Sistem*

Pada tahap ini, setelah aplikasi selesai dikembangkan kemudian dilakukan pengujian. Pengujian ini dimaksudkan untuk mengetahui dan membuktikan bahwa aplikasi yang sudah dirancang sudah sesuai dengan yang diharapkan. Pengujian sistem informasi geografis dilakukan menggunakan metode *blackbox testing*. Evaluasi sistem pengembangan dilakukan untuk mengetahui hasil virtualisasi server yang telah diimplementasikan.

## III. HASIL DAN PEMBAHASAN

Setelah melakukan perancangan sistem dan analisa data yang akan diimplementasikan kedalam program, maka pada bagian hasil dan pembahasan dalam penelitian ini juga akan dituliskan tentang beberapa hal yang bersangkutan dengan hasil perancangan sistem informasi geografis pada server pengembangan virtual docker *container*.

### A. *Data Cagar Budaya*

Setelah melakukan observasi secara daring melalui situs resmi cagar budaya Kemendikbud serta dokumen laporan hasil verifikasi & validasi cagar budaya Mojokerto oleh Kemendikbud tahun 2016 dan langsung mengunjungi beberapa cagar budaya di Mojokerto peneliti mendapatkan data cagar budaya yang digunakan sebagai objek dari sistem informasi geografis. Berikut ini merupakan sebagian data cagar budaya tertera pada tabel 1.

TABEL I
DATA CAGAR BUDAYA

| No | Nama Cagar Budaya | Alamat | Kode pengelolaan |
|---|---|---|---|
| 1 | Petirtaan Jolotundo | Area Hutan Trawas, Seloliman, Trawas | KB000448 |
| 2 | Candi Minak Jinggo | Jl. Candi Minak Jinggo, Unggahan, Trowulan | KB000421 |
| 3 | Candi Brahu | Jl. Candi Brahu No.73, Siti Inggil, Bejijong, Trowulan | KB000416 |
| 4 | Candi Gentong | Jl. Candi Gentong Telogo Gede, Trowulan | KB000420 |
| 5 | Candi Kedaton | Jl. Pendopo Agung, Sidodadi, Sentonorejo, Trowulan | KB000422 |
| 6 | Candi Tikus | Jl. Trowulan-Jatirejo, Temon, Trowulan | KB000417 |
| 7 | Candi Wringin Lawang | Jl. Candi Wringin Lawang, Jatipasar, Trowulan | KB000418 |
| 8 | Candi Bajangratu | Jl. Candi Tikus No 9, Pelem, Temon, Trowulan | KB000415 |
| 9 | Kolam Segaran | Jl. Pendopo Agung, Nglinguk, Trowulan | KB000419 |
| 10 | Situs Umpak Sentonorejo | Jl. Pendopo Agung, Sidodadi, Sentonorejo, Trowulan | KB000423 |
| 11 | Situs Kumitir | Dusun Sedati, Kumitir | |
| 12 | Petirtaan Nglinguk | Dusun Nglinguk, Trowulan | |
| 13 | Petirtaan Tegalan | Dusun Tegalan, Trowulan | |

Data tersebut merupakan data utama yang dimasukkan dalam sistem, adapun beberapa data pendukung diantaranya deskripsi cagar budaya, gambar, kabupaten, provinsi serta latitude dan longitude.

### B. *Virtualisasi Server Docker Container*

Setelah kebutuhan data tercukupi selanjutnya mengembangkan virtual server berbasis docker sebagai server pengembangan lokal untuk membuat dan menjalankan sistem informasi geografis cagar budaya. Sebelum membuat server pengembangan lokal docker *container* peneliti menyusun direktori *file* dan folder secara sistematis agar mudah dibaca oleh pengguna dan docker. Berikut struktur folder dan *file* yang didefinisikan.

```
E:.
|   .env
|   docker-compose.yml
|
+---docker
|   +---apache
|   |       000-default.conf
|   |       Dockerfile
|   |
|   \---mysql
|           cagarbudaya_database.sql
|           Dockerfile
|
\---src
```

Gbr 7. Direktori folder dan file docker

Pada luar folder terdapat file '.env' dan 'docker-compose.yml' serta terdapat folder docker dan src. Di dalam folder docker terdapat dua sub folder apache dan mysql. Dalam folder apache terdapat 'Dockerfile' dan '000-default.conf'. Di folder mysql memuat database aplikasi 'cagarbudaya_database.sql' dan 'Dockerfile'.

```
FROM php:8.0-apache
RUN echo "ServerName localhost" >> /etc/apache2/apache2.conf
RUN apt-get update
RUN apt-get install -y git zip curl sudo unzip libicu-dev libbz2-dev \
    libpng-dev libjpeg-dev libmcrypt-dev libreadline-dev \
    libfreetype6-dev

RUN docker-php-ext-install bz2 intl \
    bcmath opcache calendar pdo_mysql mysqli

COPY docker/apache/000-default.conf /etc/apache2/sites-available/000-default.conf

RUN a2enmod rewrite headers

RUN mv "$PHP_INI_DIR/php.ini-development" "$PHP_INI_DIR/php.ini"

RUN curl -sS https://getcomposer.org/installer | php
RUN mv composer.phar /usr/local/bin/composer
RUN chmod +x /usr/local/bin/composer
RUN chown -R www-data:www-data /var/www/html
RUN composer self-update

COPY src/ /var/www/html/

ARG uid
RUN useradd -G www-data,root -u $uid -d /home/devuser devuser
RUN mkdir -p /home/devuser/.composer && \
    chown -R devuser:devuser /home/devuser

EXPOSE 80
```

Gbr 8. Dockerfile apache

Dokerfile tersebut membangun *image* dari *base* 'php:8-0.apache' lalu menambahkan nama server 'localhost' ke dalam konfigurasi 'apache2.conf'. Kemudian memperbarui daftar package yang ada serta menginstal librari dan ekstension yang didefinisikan. Librari dan ekstension yang banyak digunakan dalam lingkungan pengembangan adalah git, curl, zip, intl, opcache, mysqli dan sebagainya.

Kemudian docker akan menyalin konfigurasi apache dalam folder 'docker/apache' ke dalam folder '/etc/apache2/sites-available/' yang ada di dalam image tersebut. Lalu docker akan menjalankan fungsi 'a2enmod rewrite headers' untuk mengontrol dan memodifikasi rule *route*. Konfigurasi php dalam 'php.ini' diubah ke development agar memudahkan proses *debug*. Menginstal composer sebagai *php dependency manager* untuk mempermudah menggunakan *library* resmi maupun *third party*. Setelah itu docker akan menyalin file yang ada di folder 'src' ke root *directory hosting* apache '/var/www/html' dengan user 'www-data' dalam grup 'www-data'. Kemudian docker akan menambahkan user

yang memiliki user id sama dengan komputer lokal ke dalam grup 'www-data' dan 'root' serta membuat grup 'devuser' dengan *user* 'devuser' tujuannya unuk memberi izin user lokal dalam server apache. Terakhir docker akan mengekspose port default http yaitu 80 agar *web server image* dapat berkomunikasi.

```
<VirtualHost *:80>
  ServerAdmin webmaster@localhost
  DocumentRoot /var/www/html/public

  <Directory /var/www/html>
    Options Indexes FollowSymLinks
    AllowOverride All
    Require all granted
  </Directory>
</VirtualHost>
```

Gbr 9. Konfigurasi apache 000-default.conf

Konfigurasi apache digunakan untuk mengkonfigurasi virtual host apache, konfigurasi dalam file tersebut akan diprioritaskan untuk dibaca server. Dalam konfigurasi tersebut didefinisikan port host, admin server dan root dokumen yang merujuk pada folder public karena pada codeigniter 4 file index diamankan di folder tersebut. Untuk direktori 'var/www/html' memberi perintah apache untuk mengikuti link sesuai pada file sistem dan menampilkan list direktori jika tidak terdapat file index. Apache mengizinkan semua ip mengakses *web server* pada 'Require all granted'. Mendefinisikan 'AllowOverride All' agar mengubah direktori yang ada secara dinamis berbasis dari aplikasi.

```
FROM mysql


ENV MYSQL_ROOT_PASSWORD=${MYSQL_ROOT_PASSWORD}
ENV MYSQL_DATABASE=${MYSQL_DATABASE}

#COPY ./docker/mysql/cagarbudaya_database.sql /docker-entrypoint-initdb.d/init.sql

EXPOSE 3306
```

Gbr 10. Dockerfile mysql

Dockerfile mysql membangun *image* dari base mysql kemudian mendefinisikan mysql password dan database dari global variabel file env. Selanjutnya docker menginstruksikan untuk menyalin database dump dari cagar budaya untuk diunggah dalam *image* mysql kemudian docker akan mengekspos port 3306 agar layanan komunikasi database dapat tersedia.

```
UID=1000

MYSQL_ROOT_PASSWORD=isron
MYSQL_DATABASE=sig
```

Gbr 11. File .env

File '.env' digunakan untuk mendefinisikan *default* variabel agar dapat digunakan sebagai penyimpan *value* pada saat docker dijalankan. File '.env' tersebut mendefinisikan 'UID' yang digunakan pada docker apache sebagai variabel untuk user id, 'MYSQL_ROOT_PASSWORD' mendefinisikan password database pada image mysql, demikian pula untuk 'MY_SQL_DATABASE' mendefinisikan nama database. hal tersebut dilakukan agar *value* dari variabel dapat diubah secara dinamis dan mudah.

```
version: '3'
services:
    db:
        build:
            context: .
            dockerfile: docker/mysql/Dockerfile
        environment:
            MYSQL_DATABASE: ${MYSQL_DATABASE}
            MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
        command: --default-authentication-plugin=mysql_native_password
        restart: unless-stopped
        volumes:
            - ./db_data:/usr/data
        ports:
            - 3308:3306
    web:
        build:
            context: .
            dockerfile: docker/apache/Dockerfile
            args:
                uid: ${UID}
        environment:
            - APACHE_RUN_USER=#${UID}
            - APACHE_RUN_GROUP=#${UID}
        restart: unless-stopped
        volumes:
            - ./src:/var/www/html
            - ./apache_log:/var/log/apache2
        ports:
            - 8000:80
        depends_on:
            - db
        links:
            - db
    adminer:
        image: adminer
        restart: unless-stopped
        ports:
            - 8080:8080
volumes:
    db_data:
    src:
```

Gbr 12. File docker-compose.yml

File 'docker-compose.yml' merupakan suatu konfigurasi dalam file berformat YAML untuk mendefinisikan dan menjalankan aplikasi docker multi-*container*. Dalam pengembangan server lokal ini peneliti mendefinisikan tiga *image* yang akan di jalankan sebagai *container* yaitu mysql(db), php-apache server(web), dan adminer.

Dalam menjalankan *container* database mysql(db) docker akan membangun *image* dari dockerfile yang ada di folder 'docker/mysql' kemudian memasukkan *environtment* nama dan password database dari yang telah didefinisikan. Mysql 8 menggunakan caching_sha2 password untuk default autentikasi, untuk menetralkan autentikasi mysql driver dapat mengubah menjadi mysql_native_password. Dengan fungsi 'restart: unless-stopped' Ketika memulai *container* mysql, *container* tidak akan di-*restart* kecuali pengguna menjalankan fungsi untuk menghentikan *container*. Docker volume membuat jalur file pada folder 'usr/data' dari *image* ke lokal komputer. *Container* mysql akan diekspose oleh docker pada port 3306 dan diteruskan ke lokal pada port 3308.

*Container* server pengembangan aplikasi(web) dibangun dari *image* hasil menjalankan dockerfile yang ada di folder 'docker/apache' dan konfigurasi args UID yang didefinisikan di env, kemudian memasukkan *environtment* user id dan user grup sesuai UID yang didefinisikan. Ketika memulai *container* apache tidak akan di-*restart* kecuali pengguna menjalankan fungsi untuk menghentikan *container*. Docker volume './src:/var/www/html' membuat jalur file aplikasi ke folder 'src' lokal dan

'./apache_log:/var/log/apache2' menyimpan log apache server ke folder 'apache_log' lokal. Docker mengekspos port 80 dalam *container* diteruskan ke port 8000 pada lokal komputer. *Container* web akan dijalankan ketika *container* db sudah *running*. *Container* web juga menyimpan data yang dikirim dan diterima dari *container* db melalui links.

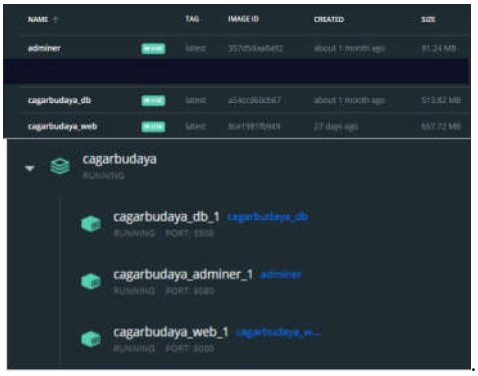
*Container* adminer dibangun dari *image* adminer di repositori(docker hub) apabila sebelumnya tidak di-*pull* ke lokal. Ketika menjalankan *container* adminer tidak akan di-*restart* kecuali pengguna menjalankan fungsi untuk menghentikan *container*. Docker akan mengekspos port 8080 dalam *container* untuk diteruskan ke port 8080 pada lokal komputer.

Docker volume mendefinisikan penyimpanan yang dapat dikelola dengan mudah melalui docker maupun *host os*. Docker volume juga merupakan jalan terbaik untuk mengelola file secara permanen pada *container*. Setelah Docker compose selesai ditulis maka akan dieksekusi untuk menjalankan beberapa *image* menjadi *container* dan saling terhubung dengan perintah 'docker-compose up'.

```
E:\Kuliah\Skripsi\Project\cagarbudaya>docker-compose up
Creating network "cagarbudaya_default" with the default driver
Creating cagarbudaya_db_1      ... done
Creating cagarbudaya_adminer_1 ... done
Creating cagarbudaya_web_1     ... done
```

Gbr 13. Fungsi docker-compose up

Setelah fungsi tersebut dijalankan maka docker akan membuat *image* dan *container* yang telah didefinisikan, *container* tersebut akan dihubungkan dalam satu jaringan. Berikut pada gambar 14 ialah docker container yang telah dijalankan melalui docker compose pada docker desktop




Gbr 14. Container pada docker desktop

## C. *Pengembangan Sistem Informasi Geografis*

Proses berikutnya setelah data kebutuhan sudah dilengkapi dan lingkungan pengembangan virtual berbasis docker disiapkan ialah mengembangkan sistem informasi geografis cagar budaya Kabupaten Mojokerto. Peneliti membuat sistem informasi geografis dengan bahasa php *framework* codeigniter 4. Untuk membuat file proyek codeigniter dapat melakukan *download* melalui composer pada *command container* 'cagar_budaya_web_1'.

Setelah proyek diinisiasi peneliti mengatur *environtment variable* dari proyek codeigniter 4 agar dapat berjalan pada server dan berkomunikasi dengan database. berikut pada gambar 15 merupakan susunan *environtment variable* dalam file '.env' pada proyek cagarmaja.

```
#--------------------------------------------------------------------
# APP
#--------------------------------------------------------------------

app.baseURL = 'http://127.0.0.1:8000'
app.forceGlobalSecureRequests = false
app.locale = "en"

#--------------------------------------------------------------------
# DATABASE
#--------------------------------------------------------------------

database.default.hostname = 172.20.0.1:3308
database.default.database = sig
database.default.username = root
database.default.password = isron
database.default.DBDriver = MySQLi
# database.default.DBPrefix =
```

Gbr 15. File .env proyek cagarmaja

Pada file *environtment* tersebut perlu merubah konfigurasi default. Base url menambahkan port 8000 sesuai dengan port yang diekspos oleh *container* 'cagarbudaya_web_1'. Untuk konfigurasi database perlu mengubah default hostname, database, username serta password. Untuk mengetahui ip *container* database yang di-*assign* oleh docker dengan menjalankan fungsi 'docker container inspect container_db_1' kemudian dapat dilihat pada ip gateway. Konfigurasi lainnya menyesuaikan dengan *environtment* terdahulu.


```
"Networks": {
    "cagarbudaya_default": {
        "IPAMConfig": null,
        "Links": null,
        "Aliases": [
            "67fb87727ebc",
            "db"
        ],
        "NetworkID": "665d836fa0fd4d9abfb75492b5af1ca8c8a637068e3ff205be18f7e9d6672032",
        "EndpointID": "7214d4dc0a923bb9c21b07607960a7eea0437ad9c647ca180347aa7027c5f75",
        "Gateway": "172.20.0.1",
        "IPAddress": "172.20.0.2",
        "IPPrefixLen": 16,
        "IPv6Gateway": "",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "MacAddress": "02:42:ac:14:00:02",
        "DriverOpts": null
    }
}
```

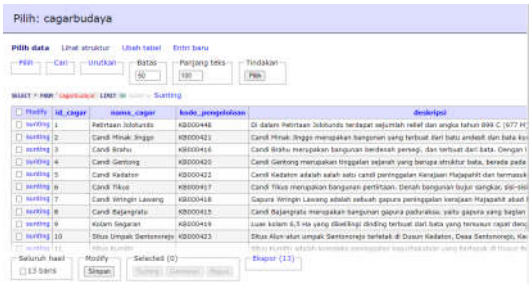

Gbr 16. Docker inspect container db

*Image* docker dapat di-*push* ke repositori sehingga dapat menghemat waktu untuk membuat ulang *image*, pada kasus ini lingkungan pengembangan php apache di-*push* ke dockerhub sebagai salinan *image*. Berikut pada gambar 17 merupakan *image* pada dockerhub.


Gbr 17. Docker image pada dockerhub

Setelah itu peneliti dapat memasukkan data cagar budaya yang telah disediakan ke dalam database mysql pada *container* melalui adminer. Berikut pada gambar 18

data cagar budaya yang sudah dimasukkan ke dalam database dan struktur data cagar budaya pada gambar 19.




Gbr 18. Database cagar budaya via adminer


Gbr 19. Struktur data cagar budaya via adminer

Data deskripsi tersebut diambil dari situs cagarbudaya Kemendikbud, *latitude* dan *longitude* diambil dari titik lokasi google maps. Data tersebut nantinya akan dijadikan sebagai objek yang akan ditampilkan dalam sistem.

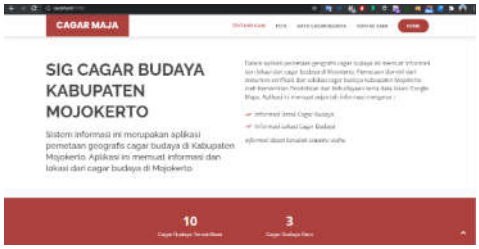
Sebelum membuat halaman yang terintegrasi terlebih dahulu mendefinisikan routes untuk menangani *request* yang diterima sehingga dapat memberikan *response* yang sesuai, *route* didefinisikan sesuai *path* dan fungsinya. Untuk menangani *request* dengan *path* '/' akan dikontrol oleh *controller* home untuk menampilkan halaman utama, begitu pula untuk *path* lainnya seperti admin akan dikontrol oleh *controller* admin. Konfigurasi *route* lainnya dapat dilihat pada gambar 20.

```
 * --------------------------------------------------------------------
 * Route Definitions
 * --------------------------------------------------------------------
 */
$routes->get('/', 'Home::index');
$routes->get('/cagar/(:segment)', 'Cagar::detail/$1');

$routes->get('/admin', 'Admin::index');
$routes->get('/admin/create', 'Admin::create');
$routes->get('/admin/edit/(:segment)', 'Admin::edit/$1');
$routes->delete('/admin/(:num)', 'Admin::hapus/$1');
```

Gbr 20. Routes aplikasi


Gbr 21. Tampilan halaman utama

Gambar diatas merupakan tampilan halaman utama dari sistem informasi geografis cagar budaya Kabupaten Mojokerto. Dapat mengakses 'localhost:8000' sesuai dengan port docker *container* 'cagarbudaya_web_1'. Pengguna dapat klik mulai untuk beralih ke halaman selanjutnya.




Gbr 22. Tampilan halaman utama

Halaman tentang pada gambar 22 menampilkan deskripsi dan fungsi sistem serta menampilkan jumlah cagar budaya terverifikasi dan belum terverifikasi. Cagar budaya terverifikasi memiliki kode pengelolaan sedangkan jika belum terverifikasi tidak memiliki kode pengelolaan. Data verifikasi tersebut diambil dari database melalui *controller* pada gambar 23.

```
class Home extends BaseController
{
    protected $dataM;
    public function __construct()
    {
        $this->dataM = new dataModel();
    }

    public function index()
    {
        $verif = $this->dataM->getVerif();
        $notVerif = $this->dataM->getNotVerif();
        $data = [
            'cagar' => $this->dataM->tampilData(),
            'verif' => $verif,
            'notverif' => $notVerif
        ];

        return view('pages/beranda', $data);
    }
}
```

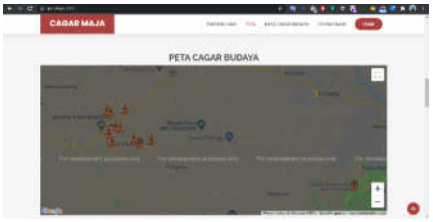
Gbr 23. Controller home

*Controller* home akan mengatur *backend* dari halaman utama, method getVerif() digunakan untuk mengambil cagar budaya terverifikasi dan getNotVerif() mengambil cagar budaya belum terverifikasi. tampilData() digunakan untuk mengambil semua data cagar budaya. Method tersebut terhubung ke database melalui model dataM kemudian ditampilkan ke halaman utama. Berikut pada gambar 24 ialah model dari *controller* home.

```
class dataModel extends Model
{
    protected $table      = 'cagarbudaya';
    protected $id         = 'id_cagar';
    protected $useTimestamps = true;

    public function tampilData($id = false)
    {
        if ($id == false) {
            return $this->findAll();
        }

        return $this->where(['id_cagar' => $id])->first();
    }

    public function getVerif()
    {
        $notverif = $this->where(['kode_pengelolaan' => ''])->countAllResults();
        $semua = $this->countAll();
        $hasil = $semua - $notverif;

        return $hasil;
    }

    public function getNotVerif()
    {
        return $this->where(['kode_pengelolaan' => ''])->countAllResults();
    }
}
```

Gbr 24. Model dataModel

Navigasi halaman selanjutnya ialah peta cagar budaya, peta tersebut menampilkan titik lokasi cagar budaya yang dapat diklik untuk melihat detail dari cagar budaya. Maps tersebut menggunakan API Google maps javascript. Berikut pada gambar 25 tampilan halaman maps cagar budaya.




Gbr 25. Halaman peta

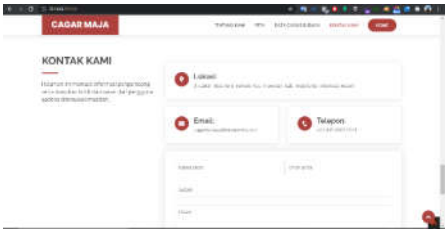
Pada halaman selanjutnya menampilkan daftar data cagar budaya yang ada dalam database. data cagar budaya diambil dari model melalui *controller* untuk diteruskan ke *view* halaman data cagar budaya. Pada *view* disisipkan tombol untuk melihat detail cagar budaya. Berikut gambar 26 merupakan tampilan halaman data cagar budaya.


Gbr 26. Model data cagar budaya

Ketika klik tombol detail maka akan beralih ke halaman detail untuk menampilkan titik lokasi dan detail dari cagar budaya tersebut. Berikut pada gambar 27 merupakan detail lokasi serta data cagar budaya.


Gbr 27. Tampilan detail data dan lokasi cagar budaya

Navigasi terakhir pada halaman utama ialah kontak, pada halaman ini menampilkan kontak dari pengembang dan pesan yang ingin disampaikan kepada pengembang. Pada kolom tersebut memiliki *exception* dan kondisi, kolom nama harus minimal 4 karakter, kolom email harus memasukkan format email yang valid, kolom subjek minimal 8 karakter dan semua kolom masukan harus terisi. Berikut pada gambar 28 merupakan halaman kontak.

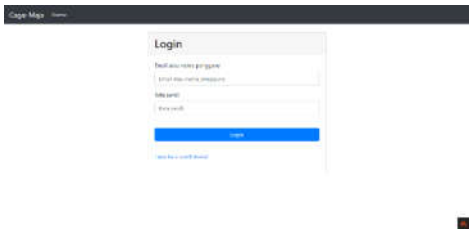



Gbr 28. Tampilan halaman kontak

Dalam sistem informasi geografis ini juga terdapat admin untuk mengelola data cagar budaya, sebelum memasuki halaman admin akan ditampilkan halaman login, untuk kasus ini peneliti memasang *library* pihak ketiga sebagai autentikasi yaitu *library* 'myth/auth'. Akun admin sudah didaftarkan terlebih dahulu kemudian menonaktfkan fitur registrasi. Data admin disisipkan ke database melalui migrasi dari *library* 'myth/auth'. Untuk database *user* dapat dilihat pada gambar 29.


Gbr 29. Database user

Setelah database auth/myth telah dimigrasi, konfigurasi email pengembang dan *routes* yang melewati *path* admin harus melakukan autentikasi *login* terlebih dahulu untuk masuk ke dalam halaman admin. Berikut tampilan halaman login pada gambar 30.




Gbr 30. Login admin

Form *login* admin memiliki kondisi jika *username* dan *password* tidak sesuai maka akan muncul pesan peringatan kredensial salah. Jika kredensial benar maka akan masuk ke halaman admin. Pada gambar 31 adalah tampilan halaman admin.


Gbr 31. Halaman admin

Pada halaman ini admin dapat menambah, menghapus dan mengubah data cagar budaya.

*D.* Pengujian dan Evaluasi Sistem

Setelah lingkungan pengembangan dan aplikasi telah selesai dibuat maka tahap berikutnya ialah melakukan pengujian pada aplikasi serta evaluasi terhadap sistem pengembangan yang telah dirancang. Untuk pengujian aplikasi peneliti menggunakan blackbox testing dan mengevaluasi hasil akhir server pengembangan lokal.

1. Blackbox Testing

Pengujian blackbox diterapkan pada sistem informasi geografis cagar budaya Kabupaten Mojokerto yang telah dikembangkan sesuai dengan analisis dan desain yang dirancang untuk fungsionalitas utama sistem.

TABEL II
BLACKBOX TESTING

| **Data Masukan** | **Yang Diharapkan** | **Hasil Pengamatan** | **Keterangan** |
|---|---|---|---|
| Halaman tentang sistem | Menampilkan jumlah cagar budaya terverifikasi dan belum verif | Tampil jumlah cagar budaya terverifikasi dan belum verif | Sukses |
| Halaman peta cagar budaya | Tampil titik lokasi dan tombol detail cagar budaya | Tampil titik lokasi dan tombol detail cagar budaya | Sukses |
| Halaman data cagar budaya | Menampilkan daftar data cagar budaya | Tampil daftar data cagar budaya | Sukses |
| Klik detail data cagar budaya | Beralih ke halaman detail cagar budaya | Beralih ke halaman detail cagar budaya | Sukses |
| Form pesan ke pengembang | Menampilkan pesan error ketika masukan salah | Tampil pesan error ketika masukan salah | Sukses |
| Submit form kontak pesan | Mengirim pesan ke email pengembang | Pesan terkirim ke email pengembang | Sukses |
| Request ke halaman admin | Menampilkan halaman login | Tampil halaman login | Sukses |
| Masukan username dan password | Menampilkan halaman admin jika benar, tampil error jika salah | Tampil halaman admin jika benar, tampil error jika salah | Sukses |
| Tambah, edit, hapus data role admin | Role admin dapat memanipulasi data | Role admin dapat memanipulasi data | Sukses |

2. Evaluasi Sistem

Evaluasi sistem dimaksudkan untuk mengetahui hasil dari server virtual pengembangan lokal berbasis docker *container*. Dari sistem yang telah dibuat didapatkan hasil sebagai berikut:

a. Konfigurasi dan pembangunan infrastruktur pengembangan web berbasis docker container dilakukan secara deklaratif pada dockerfile dan docker-compose sehingga fleksibel.

b. Pembangunan image server php, apache, mysql, dan adminer membutuhkan *resource* 1261 Mb serta dapat dijalankan menjadi beberapa container sehingga *scalable* dan efisien.

c. Image docker dapat di-*push* ke repositori *online* agar dapat digunakan kembali dan mudah diakses sehingga mempunyai ketersediaan yang tinggi.

## IV. KESIMPULAN

Berdasarkan penelitian yang telah dilakukan, peneliti berhasil melakukan pengembangan web sistem informasi geografis pada lingkungan pengembangan virtual berbasis docker *container*. Lingkungan pengembangan virtual docker dapat dimanfaatkan secara fleksibel, efisien dan *scalable* tanpa melakukan pendefinisian ulang. Pengujian aplikasi pada blackbox testing menunjukkan bahwa sistem mampu menampilkan titik lokasi pada maps, data cagar budaya pada database serta fungsionalitas halaman pengunjung dan admin berjalan normal. Hal ini menunjukkan bahwa aplikasi berjalan dengan baik dan lingkungan pengembangan virtual berbasis *container* dapat menjadi alternatif pengembangan aplikasi.

## V. SARAN

Berdasarkan hasil yang diperoleh peneliti memberikan saran ke depan dalam pengembangan aplikasi pada server virtual berbasis docker dengan arsitektur *microservices*, dikarenakan aplikasi dalam docker diusung sebagai *container* sehingga dapat memecah fungsionalitas sistem menjadi beberapa bagian yang independen serta dapat menyesuaikan kebutuhan dengan teknologi pengembangan.

## UCAPAN TERIMA KASIH

Penulis senantiasa mengucap syukur kepada Tuhan YME atas segala berkah, rahmat dan pertolongannya, sehingga penulis mampu menyelesaikan proyek dan artikel ilmiah ini dengan baik. Terimakasih penulis haturkan kepada kedua orangtua yang selalu memberi semangat dan dukungan, dosen pembimbing skripsi yang selalu memberikan masukan dan

saran yang membangun kepada penulis, sahabat dan teman yang selalu memberikan dukungan dan dorongan dalam melakukan penelitian. Terimakasih kepada diri sendiri karena dapat berkompromi untuk menggapai tujuan yang ingin dicapai.

## REFERENSI

[1] S. E. Prasetyo, "Design and Implementation of Lightweight Virtualization Using Docker Container in Distributing Web Application with Experimental Methods," *JITE (Journal of Informatics and Telecommunication Engineering),* pp. 270-276, 2021.

[2] M. Sri Raghavendra., & Priyanka Chawla, "A Review on Container-Based Lightweight Virtualization for Fog Computing," *International Conference on Reliability, Infocom Technologies and Optimization (ICRITO),* pp. 378-384, 2018.

[3] Amit, M., Narayan, D., Shivaraj, K., & Mohammed M., "Performance Evaluation of Docker Container and Virtual Machine," *Procedia Computer Science 171,* pp. 1419-1428, 2020.

[4] Apridayanti, S., Isnawaty & Adi S, R., "Desain dan Implementasi Virtualisasi Berbasis Docker Untuk Deployment Aplikasi Web," *semanTIK Vol.4, No.2,* pp. 37-46, 2018.

[5] Khalida R., Muhajirin, A., & Setiawati S., "Teknis Kerja Docker Container untuk Optimalisasi Penyebaran Aplikasi," *Jurnal Penelitian Ilmu Komputer, System Embedded & Logic,* pp. 167 -176, 2019.

[6] Hussein, Mohamed K., Mousa, Mohamed H & Alqarni, Mohamed A., "A placement architecture for a container as a service (CaaS) in a cloud environment," *Journal of Cloud Computing: Advances, Systems and Applications,* p. 8:7, 2019.

[7] Bellishree P., & Dr. Deepamala. N, "A Survey on Docker Container and its Use Cases," *International Research Journal of Engineering and Technology (IRJET),* pp. 2716-2720, 2020.

[8] Y. Li, "Towards fast prototyping of cloud-based environmental decision support systems for environmental scientists using R Shiny and Docker," *Dept. Of Civil, Environmental, and Geomatic Engineering, Institute of Environmental Engineering, ETH Zurich,* pp. 1-8, 2020.

[9] F. Adiputra, "Container dan Docker: Teknik Virtualisasi Dalam Pengelolaan Banyak Web," *Jurnal SimanteC,* p. 169, 2015.

[10] Yulanda & Mardeni, "Sistem Informasi Geografis Pemetaan Usaha Kecil Menengah (UKM) diwilayah Kota Pekanbaru Menggunakan Framework Codeigniter," *Jurnal Ilmu Komputer,* pp. 117-123, 2020.

[11] Pramesti, N. I. & Retnowati, F., "Verifikasi dan Validasi Cagar Budaya Kabupaten Mojokerto, Provinsi Jawa Timur," Kementerian Pendidikan dan Kebudayaan, Mojokerto, 2016.