SISTEM MONITORING DAN PELAPORAN KELUHAN DRAINASE DI KOTA MANADO BERBASIS CROWDSOURCING

SKRIPSI

OLEH

Arthur Manuel Anggow

NIM : 16021106026

UNIVERSITAS SAM RATULANGI

FAKULTAS TEKNIK

JURUSAN TEKNIK ELEKTRO

MANADO

2023

SISTEM MONITORING DAN PELAPORAN KELUHAN DRAINASE DI KOTA MANADO BERBASIS CROWDSOURCING

SKRIPSI

Disusun Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Komputer

Pada Program Studi S1 Teknik Informatika di Jurusan Teknik Elektro

Fakultas Teknik Universitas Sam Ratulangi

Oleh

Arthur Manuel Anggow

NIM : 16021106026

UNIVERSITAS SAM RATULANGI

FAKULTAS TEKNIK

JURUSAN TEKNIK ELEKTRO

MANADO

2023

LEMBAR PENGESAHAN

Judul Skripsi	: Sistem Monitoring dan Pelaporan Keluhan Drainase di Kota Manado Berbasis Crowrdsourcing
Nama	: Arthur Manuel Anggow
NIM	: 16021106026
Program Studi	: S1 Teknik Informatika
Jurusan	: Teknik Elektro Universitas Sam Ratulangi

Menyetujui :

Pembimbing I,	Pembimbing II,
Sherwin R. U. Aldo Sompie, ST, MT.	**Dr. Eng. Steven R. Sentinuwo, ST, MTI.**
NIP : 198007092005011002	**NIP : 198007092005011002**

Ketua Jurusan Teknik Elektro Universitas Sam Ratulangi,

Alwin M. Sambul, ST, M.Eng, Ph.D.

NIP : 19770929 200501 1 005

Dekan Fakultas Teknik Universitas Sam Ratulangi,

Prof. Dr. Ir. Fabian J. Manoppo. M.Agr

NIP : 19621014 199203 1 001

Tanggal Lulus : 14 Juli 2023

ABSTRAK

Hujan merupakan sesuatu berkat bagi kebanyakan orang, tetapi hal itu berbeda bila terjadi di Kota Manado, Pasalnya air hujan mengalir bukan hanya pada saluran air (drainase) tetapi juga sering mengalir hingga ke tengah jalan. Selain dapat merusak infrastruktur hal ini juga merugikan masyarakat bila drainase yang bermasalah mengakibatkan banjir. Pemerintah Kota Manado dan instansi yang terkait tentu selalu melakukan peremajaan drainase ataupun trotoar tetapi tentu masih ada titik-titik yang mungkin tidak terdeteksi. Crowdsourcing merupakan solusi dimana masyarakat Kota Manado secara bebas berpatisipasi dalam menginformasikan titik-titik (*google maps API*) drainase yang bermasalah maupun juga keluhan masyarakat diseputaran saluran air (trotoar, bekas galian saluran air yang dibiarkan, dll). Aplikasi *website* dibangun dengan *framework Laravel* dan juga memanfaatkan *sign-in* dari *google account* agar menjangkau masyarakat luas dan memudahkan dalam mengakses aplikasi ini.

Kata Kunci: *Crowdsourcing, Laravel, Google-maps*, *Google cloud*, Drainase.

ABSTRACT

Rain is a blessing for most people, but it is different when it happens in Manado City. This is because rainwater flows not only in waterways (drainage). but it also often flows to the middle of the road. Besides being able to damage infrastructure, this is also detrimental to the community if the drainage problem causes flooding.The Manado City Government and related agencies are of course always rejuvenating drainage or sidewalks, but there are still points that may not be detected. Crowdsourcing is a solution where the people of Manado City freely participate in informing the drainage points (google maps API) that have problems and also complaints from the people around the waterways (sidewalks, abandoned canal excavations, etc.). The website application is built with the Laravel framework and also utilizes sign-in from a Google account in order to reach the wider community and make it easier to access this application. .

Keywords: *Crowdsourcing, Laravel, Google-maps, Google Cloud*, *Drainage*.

KATA PENGANTAR

Segala Puji syukur dipanjatkan kepada Tuhan Yang Maha Esa atas segala tuntunan dan rahmatnya penulis dapat menyelesaikan tugas akhir yang berjudul "**Sistem Monitoring Dan Pelaporan Keluhan Drainase/Saluran Air Di Kota Manado Berbasis Crowdsourcing**" dengan baik dan tentu rasa bangga.

Maksud dari pembuatan skripsi ini sebagai salah satu persyaratan memperoleh gelar Sarjana Teknik pada Program Studi S1 Teknik Elektro Fakultas Teknik Universitas Sam Ratulangi Manado.

Halangan dan rintangan begitu melekat dalam pembuatan skripsi ini, pandemi adalah sebagian halangan dari beberapa halangan bagi penulis. Tapi halangan dan rintangan inilah yang memberikan pengalaman dan motivasi bagi penulis. Dan juga skripsi tidak akan berhasil diselesaikan tanpa dukungan dari semua pihak yang telah membantu baik secara langsung maupun tidak langsung. Untuk bentuk terima kasih dan syukur penulis, izinkan saya sebagai penulis mengucapkan terima kasih kepada:

1. Tuhan Yesus Karena perkenanannya saya bisa menyelesaikan skripsi saya dengan baik.
2. Prof. Dr. Ir. Oktovian Berty Alexander Sompie M.Eng. Selaku Rektor Universitas Sam Ratulangi.
3. Prof. Dr. Ir. Fabian J, Mannoppo, M.Agr. Selaku Dekan Fakultas Teknik Universitas Sam Ratulangi.
4. Virginia Tulenan, S.Kom., M.T.I selaku Ketua Program Studi S1 Teknik Informatika Jurusan Teknik Elektro Fakultas Teknik Universitas Sam Ratulangi.

5. (Alm) Stanley D. S. Karouw, ST, MTI selaku mantan Dosen Pembimbing Akademik yang selalu memberikan banyak semangat dan bimbingan selama perkuliahan
6. Muhamad Dwisnanto Putro ST, M.Eng selaku Dosen Pembimbing Akademik yang selalu memberikan banyak semangat dan bimbingan selama perkuliahan
7. Sherwin Reinaldo U Aldo Sompie ST, MT dan Dr. Eng. Steven R Sentinuwo, ST., MTI selaku Dosen Pembimbing Tugas Akhir dan juga Ketua Jurusan Teknik Elektro, yang telah memberikan banyak ilmu dan bimbingan arahan dalam penyusunan tugas akhir ini.
8. Seluruh dosen Program Studi S1 Informatika, Fakultas Teknik, Universitas Sam Ratulangi yang sudah banyak memberikan ilmu dan pengalaman yang bisa didapatkan.
9. Keluarga yang sudah selalu berkorban baik mama, papa, kaka, (almh) Ama Tan Siu Hoa, ape ko Titi, aem ci Hong, koko Anggun, koko Bryan yang senantiasa memberikan dorongan dan motivasi.
10. Teman-teman lingkaran ada Key, Davin, Epoy, Ocep, Pipo, Alex, Axel, Ai yang tentu selalu menghibur dan memotivasi.
11. Teman-teman Sobat misqueen ada Valdi, Sani, Orion yang selalu menghibur dan memotivasi.
12. Teman-teman Gereja Sion Ranomut dan para pendeta, Komisi Pelayanan Remaja Jemaat Sion Ranomuut.
13. Teman-teman yang sudah terlibat dalam tugas ini ada Eduardo.
14. Teman-teman FASE yang sudah memotivasi dan menghibur.
15. Teman-teman Mode Hardcore Orion dan Andre yang telah berjuang bersama ditugas akhir ini.

Pembuatan Skripsi ini tentu masih jauh dari kata sempurna, begitu banyak kekurangan dalam pembuatan aplikasi maupun penyusunan laporan tugas akhir ini. Untuk itu penulis sangat mengharapkan masukan-masukan agar menjadi bahan ajar introspeksi diri untuk penulis lebih baik lagi kedepannya. Sekali lagi penulis ucapkan terima kasih dan semoga tugas akhir ini dapat bermanfaat bagi kita semua.

Manado, 2023

Penulis

Arthur Manuel Anggow

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I

PENDAHULUAN

1.1. Latar Belakang

Bagi masyarakat Manado hujan merupakan berkat Tuhan yang patut disyukuri, namun dewasa ini hujan merupakan hal yang sangat diwaspadai oleh masyarakat Manado, pasalnya ketika hujan turun bukan air saja yang mengalir di saluran air (drainase) tetapi ada begitu banyak kotoran seperti sampah juga hanyut terbawa,menyebabkan aliran air terhambat. Sehingga dapat dipastikan drainase berjalan dengan tidak optimal, air didrainase tumpah kejalan-jalan, dan lebih parah lagi dapat merembes ke pemukiman warga. Selain itu masih banyak juga masalah-masalah drainase lainnya seperti drainase tidak terlalu dalam akibat lumpur, kotoran bekas galian drainase yang dibiarkan. Selain itu drainase sering dihubungkan dengan trotoar atau jalur pedestrian untuk pejalan kaki seperti masalah trotoar yang berlubang, pedestrian yang kurang ramah bagi disabiltas dan lain-lain.

Proses dalam masyarakat melakukan laporan umumnya cukup bertele-tele, dimulai dari kepala lingkungan lanjut bertahap ke kelurahan dan seterusnya sampai ditingkat pemerintah kota. Proses birokrasi tersebut tentu dirasa terlalu lambat. Di era digital sekarang proses birokrasi yang bertele tele sudah sangat ketinggalan jaman, dengan proses yang cepat tentunya membuat nilai tambah bagi Kota Manado sebagai *Smart City*. Adapun cara lainya dalam melakukan laporan yaitu menggunakan media sosial. Umumnya masyarakat Manado menggunakan Media sosial seperti *facebook*, warga sering memposting di forum-forum Kota Manado namun postingan tersebut tidak dapat memberi kepastian apakah laporan tersebut di tanggapi oleh dinas yang terkait.

Pemerintah Kota ataupun dinas yang terkait setiap bulan rutin berupaya melakukan peremajaan drainase-drainase yang ada di Kota Manado, namun sayang

ada begitu banyak drainase yang bermasalah di Kota Manado yang tidak terdeteksi oleh dinas terkait, maka dari itu sangat diperlukan peran aktif dari masyarakat dalam membantu menginformasikan titik buta dari permasalahan drainase tersebut. Konsep umum Crowdsourcing merupakan suatu sumberdaya yang berasal dari kerumunan orang, sumberdaya tersebut dapat sebuah aktifitas ataupun tindakan yang dilakukan tanpa memandang suatu bentuk fisik, suku dan ras, bagi setiap orang yang ingin berkontribusi atau memberikan solusinya atas suatu permasalahan dengan imbalan yang murah atau pun Cuma-Cuma. Disinilah *Crowdsourcing* menjembatani masyarakat dengan Pemerintah Kota atau dinas terkait melalui sistem monitoring dan pelaporan dalam pemberian informasi yang cepat dan aktual.

Pengimplementasian *Crowdsourcing* ini akan berupa Aplikasi Web , fitur dari aplikasi web menampilkan peta digital beserta titik kordinat drainase yang dikeluhkan,menampilkan informasi keluhan, dan untuk membedakan apakah laporan tersebut telah ditindak lanjuti akan diberikan tanda verifikasi oleh dinas yang terkait, dan yang terakhir akan tersedia kolom komentar. informasi yang diberikan oleh masyarakat seperti: keterangan masalah, bukti foto, keterangan lokasi ,dan titik koordinat peta digital.

Dengan adanya sistem monitoring dan pelaporan keluhan masyarakat terkait drainase/saluran air dikota Manado berbasis Crowdsourcing dapat menjadi wadah aspirasi masyarakat dan dapat membantu Instansi yang terkait dalam mendapatkan informasi yang aktual, untuk membuat Kota Manado semakin bersih, nyaman untuk ditinggali dan dikenal sebagai kota cerdas.

1.2. Rumusan Masalah

Berdasarkan latar belakang diatas maka rumusan masalah dari penilitian ini sebagai berikut :

1. Bagaimana membangun sebuah aplikasi web yang dapat memuat laporan keluhan masyarakat mengenai drainase/saluran air di Kota Manado ?
2. Bagaimana membangun Aplikasi Web yang dapat memonitoring hasil pelaporan serta menampilkan titik-titik permasalahan dan titik-titik sudah tertangani melaui *Google Map API* ?

1.3. Batasan Masalah

Adapun batasan masalah dari penelitian tugas akhir ini adalah :

1. Jenis permasalahan yang akan dijadikan objek penelitian adalah infrastruktur seperti drainase dan trotoar yang ada dikota Manado.
2. Tampilan Apllikasi Web ini akan berfokus pada tampilan *Web Browser PC*.
3. User yang dapat menggunakan Aplikasi Web ini adalah user yang memiliki *Google Account*.

1.4. Tujuan Penelitian

Tujuan dari penelitian tugas akhir ini adalah :

1. Untuk mengetahui manfaat Crowdsourcing bagi teknologi informasi dan di implementasikan pada Aplikasi web.

1.5. Manfaat Penelitian

Manfaat dari penelitian ini adalah menyajikan Sistem Monitoring dan pelaporan berbasis *Crowdsourcing* yang menjadikan suatu wadah yang dapat menampung aspirasi masyarakat mengenai permasalahan drainase maupun pedestrian yang ada di Kota Manado, dan dapat membantu pemerintah Kota dan instansi terkait

dalam mendapatkan dan mengumpulkan informasi yang dibutuhkan demi menjaga ekosistem infrastruktur drainase/saluran air dan pedestrian yang lebih optimal.

BAB II

DASAR TEORI

2.1. Penilitian Terdahulu

Sebelumnya terdapat penelitian yang mempunyai kemiripan dengan penelitian yang sementara dikebangkan. Penelitian tersebut menjadi acuan untuk penulis mengembangkan penelitian ini, antara lain:

Tabel 2. 1 Penelitan terdahulu

Nama peneliti	Judul Penelitian	Hasil Penelitian
Nina Setiyawati, Samodra Teguh Bowo Kesowo	Pembangunan Aplikasi Pelaporan Kecelakaan Lalu Lintas Berbasis Web Menggunakan Framework Laravel	Pada penelitian ini pengembangan aplikasi pelaporan kecelakaan lalu lintas berbasis web menggunakan *framework laravel* dan memanfaatkan Google Maps API guna mendapat lokasi kecelakaan dan pelapor serta menampilkan daerah rawan kecelakaan.
Perbedaan : Pada penelitian yang dilakukan Nina dan Samodra dalam konteks membuat laporan kecelakaan lalu lintas sehingga tidak ada perekayasaan terhadap warna marker peta, juga tidak ada fitur untuk mengupload foto, dan memuat komentar.		

2.2. Sistem Monitoring

Monitoring didefinisikan sebagai siklus kegiatan yang mencakup pengumpulan, peninjauan ulang, pelaporan, dan tindakan atas informasi suatu proses yang sedang diimplementasikan (Mercy, 2005). Pada umumnya monitoring digunakan dalam checking antara kinerja dan target yang telah ditentukan. Monitoring dapat memberikan informasi keberlangsungan proses untuk menetapkan langka menuju ke arah perbaikan yang bekesinambungan.

2.3. Drainase

2.3.1 Pengertian Drainase

Drainase berasal dari bahasa inggris "*drainage*" mempunyai arti mengalirkan, menguras, membuang, atau mengalirkan air. Secara umum, drainase didefinisikan sebagai serangkai bangunan yang berfungsi untuk mengurai dan membuang kelebihan air dari suatu kawasan atau lahan, sehingga lahan dapat difungsikan secara optimal. Drainase juga diartikan sebagai suatu cara pembuangan kelebihan air yang tidak diinginkan pada satu daerah, serta cara-cara penganggulangan akibat ditimbulkan oleh kelebihan air tersebut.(Suripin.2004)

Drainase merupakan usaha atau tindakan teknis untuk menangani kelebihan air, baik yang berasal dari air hujan, rembesan, kelebihan air irigasi atau air buangan lainnya sehingga fungsi dari suatu kawasan/lahan tidak terganggu. Sistem drainase menjadi salah satu prasarana untuk menciptakan kehidupan yang bersih dan menyenangkan bagi masyarakat yang tinggal disekitarnya. Sistem drainase yang buruk dapat menimbulkan dampak negatif bagi suatu kawasan/lahan.

2.3.2 Sistem Jaringan Drainase

Sistem jaringan drainase perkotaan umumnya dibagi atas 2 bagian, yaitu :

1. Sistem drainase mayor yaitu sistem saluran/badan air yang menanmpung dan mengalirkan air dari suatu daerah tangkapan air hujan (*catcment area*). Pada umumnya sistem drainase mayor ini disebut juga sebagai sistem saluran pembuangan utama (*major system*) atau drainase primer. Sistem jaringan ini

menampung aliran yang berskala besar dan luas drainase primer seperti kanal-kanal atau sungai-sungai.

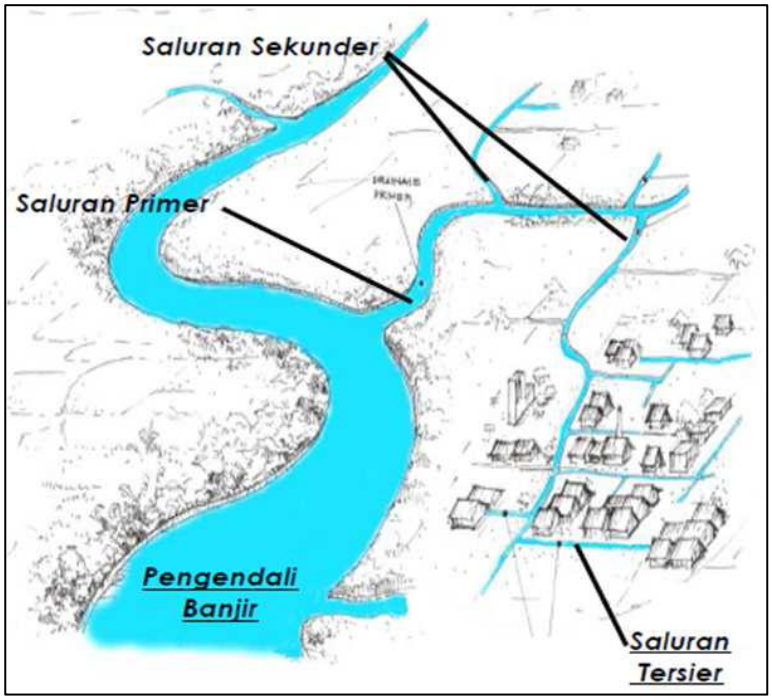
2. Sistem drainase mikro yaitu sistem saluran dan bangunan pelengkap drainase yang menampung dan mengalirkan air dari daerah tangkapan hujan. Secara keseluruhan yang termasuk dalam sistem drainase mikro adalah saluran di sepanjang jalan, saluran/selokan air hujan di sekitra bangunan, gorong-gorong dan lain sebagainnya.

2.3.3 Jenis-Jenis Drainase

Terdapat juga beberapa jenis drainase lebih spesifik yang ada dalam perkotaan:

1. Drainase Alamiah (*natural drainage)* drainase yang terbentuk secara alami tanpa bantuan manusia, artinya drainase tersebut memiliki pola limpasan air yang terbentuk secara alami, tetapi karena dasarnya adalah tanah jadi dapat terkikis. Contohnya: sungai kecil didaerah hulu (pegunungan atau bukit) hingga sungai besar dimuara.
2. Drainase Buatan (*artificial drainage)* drainase yang dengan sengaja dibuat oleh manusia berdasarkan analisis ilmu drainase. seperti contoh: Saluran drainase dibahu jalan, saluran pembuangan air.
3. Draianse Bawah Tanah (*Sub surface drainage*) drainase yang berada di bawah permukaan tanah yang bertujuan mengalirkan air limpasan melalui pipa-pipa, seperti gorong-gorong dibawah tanah, permukaan lapangan stadion, lapangan terbang, taman dan lain-lain.
4. Drainase Tersier atau jaringan tersier merupakan saluran untuk mengalirkan limbah rumah tangga ke draianse sekunder, berupa plesteran pipa, tanah dan lain-lain.
5. Drainase Sekunder atau saluran cabang adalah saluran yang berfungsi sebagai pengumpul debit yang diperoleh dari saluran drainase yang lebih kecil (drainase tersier) dan akhirnya dibuang ke saluran utama (Drainase Primer).

6. Drainase Primer atau saluran utama adalah saluran yang berfungsi sebagai pembawa air buangan dari saluran sekunder, ke dimensi yang lebih besar.



Gambar 2. 1 Perbedaan antara Saluran Primer, Saluran Sekunder, Saluran Tersier.

2.4. Crowdsourcing

Crowdsourcing diambil dari kata *Crowd yaitu* kerumunan dan *outsourcing* yaitu alih daya. Bila diartikan yaitu suatu kerumunan orang yang bersumber dari luar organisasi ataupun perorangan, kemudian berkontribusi (informasi, data, ide, opini dan lain-lain) dalam suatu proyek, yang dilakukan oleh suatu oraganisasi maupun perorangan. *Crowdsourcing* di populerkan oleh Jeff Howe dalam sebuah artikel The Rise Of Crowdsourcing menyebutkan ini merupakan era *Crowdsourcing,* tiap individu-individu telah terkoneksi dengan jaringan internet membentuk suatu komunitas, perusahaan besar memanfaatkan komunitas tersebut sebagai sukarelawan kemudian menyelesaikan permasalahan dengan lebih murah (Jeff Howe, 2006). Salah satu contoh *crowdsourcing* yang sering kita sering gunakan yaitu *Wikipedia* adalah ensikopledia berbasis website, memberikan akses bebas bagi komunitas untuk

berkontribusi dan memelihara *Wikipedia (https://en.wikipedia.org/wiki/Wikipedia, 2001).* Dari konsep tersebut *Crowdsourcing* dapat didefinisikan suatu layanan yang terdiri dari perorangan, komunitas, organisasi, lembaga, institusi, perusahaan yang terkoneksi dengan internet kemudian berkontribusi ke dalam suatu proyek ataupun pemecahan permasalahan dilakukan secara sukarela atau dengan imbalan yang murah untuk mencapai suatu tujuan kebaikan bersama.

Banyak metode untuk melaksanakan *Crowdsourcing* seperti mengisi kuisioner, melakukan *testing* aplikasi, melakukan suatu hal kreatif dan masih banyak lagi. Metode dipilih secara bebas oleh penyediah *Crowdsourcing* demi mencapai tujuan yang efisien. Proses pembuatan laporan untuk mengumpulkan data-data mengenai *drainase* bermasalah selaras dengan nilai-nilai dari *Crowdsourcing.*

2.5. NGINX

NGINX merupakan salah satu Web Server bersifat open source. NGINX memakai arsitektur asinkron dan dijalankan berdasarkan *event.* Dengan arsitektur tersebut NGINX dikenal dengan performanya yang cepat dalam menangani koneksi. NGINX adalah *web server* yang dibuat dengan menawarkan penggunaan memori yang rendah serta dapat berjalan pada proses yang tinggi (mampu menangani banyak permintaan dalam waktu bersamaan). NGINX juga memberikan kemudahan dalam hal mengkonfigurasi sehingga NGINX sangat mudah dipelajari, untuk memasang NGINX kedalam aplikasi hanya tinggal memanggil images NGINX yang tersedia pada *Docker Hub.* Lalu

2.6. Web Service

Web service merupakan suatu layanan berupa fungsi atau prosedur yang diakses melalui protokol web (HTTP). Dengan perkembangan teknologi yang memiliki tingkat kompleksitas aplikasi yang berbeda, web *service* hadir untuk mewujudkan koneksi atau hubungan antar aplikasi. Dengan adanya web service

memudahkan perpaduan fungsi dalam membangun sebuah program aplikasi tanpa bergantung lagi pada sistem operasi maupun bahasa pemrograman yang digunakan. Hal ini dimungkinkan karena web *service* berkomunikasi menggunakan sebuah standar format data yang universal yaitu XML, JSON dan menggunakan protokol Simple *Object Access Protocol* (SOAP) ataupun *Representational State Transfer* (REST). Dengan adanya web *service* menggunakan format data XML atau JSON, maka web *service* juga mendapatkan sifat multi-tier aplikasi.

2.7. MySQL

SQL (*Structured Query Language*) adalah bahasa yang digunakan dalam pemrograman dan dirancang untuk mengelola data yang disimpan dalam basis data relasional. RDBMS (*Relational Database Management System*) adalah program yang dirancang untuk mengatu sebuah basis data sebagai sekumpulan data yang disimpan secara terstruktur dan melakukan operasi-operasi pada data.

2.8. PHP

PHP Adalah bahasa *Server-side-scripting* yang menghubungkan HTML untuk membuat halaman web yang dinamis. Dengan *Server-side-scripting* maka sintaks dan perintah-perintah PHP akan dieksekusi diserver kemudian hasilnya akan dikirim ke browser dengan format *HTML*. *PHP* merupakan suatu bahasa pemrograman *open source* yang memudahkan programer dalam mengembangkan web secara luas. Keuntungan *PHP* antara lain adalah karena penggunaan *web* yang dinamis sehingga dalam proses maintenance situs web tersebut lebih mudah dan efisien. Selain itu *PHP* juga mampu sebagai antar muka dengan basis data secara baik, support dengan bermacam-macam server abasis data seperti *MySQL*, *Oracle*, *Sysbase*. *PHP-FPM (FastCGI Process Manager)* adalah pemrosesan canggih dan sangat efisien untuk skrip yang ditulis dalam bahasa pemrograman *PHP*.

2.9. Google Map API

API (*Application Programming Interface*) adalah sekumpulan perintah, fungsi, serta protokol yang dapat digunakan oleh *programmer*. API memungkinkan *programmer* untuk menggunakan fungsi standar untuk berinteraksi dengan sistem operasi. *Google Maps API* adalah suatu *library JavaScript*. Agar *Google Maps API* dapat muncul di website kita, diperlukan *API Key*. *API key* merupakan kode unik yang dapat kita generasikan (*generate*) dan di operasikan pada suatu *website*, agar server *Google Maps* dapat mengenali *website* tersebut.

2.10. Framework PHP

Framework PHP merupakan suatu wadah untuk membuat aplikasi web tanpa kita harus membuatnya lagi dari awal, karena Framework PHP adalah sekumpulan *library* fungsionalitas yang umum digunakan. Framework diorganisasikan pada rancangan arsitektur untuk memberikan kecepatan, ketepatan, kemudahan dan konsistensi di dalam pengembangan aplikasi dari definisi tersebut" (Siena,2009). Dasarnya *Framework PHP* menerapkan *MVC (Model, View, Controller).*

1. *Model* mencangkup semua proses yang terkait dengan pemanggilan struktur data baik berupa fungsi, input processing atau mencetak output ke dalam browser. Sebuah aplikasi web menggunakan basis data dalam menyimpan data, maka pada bagian *Model* biasanya akan berhubungan dengan perintah-perintah *query* SQL
2. *View* mencangkup semua proses yang terkait dengan layout output. Seperti menampilkan *interface website*. Seperti halaman web, rss, javascript dan lain-lain. Di dalam *view* hanya berisi variabel-variable yang berisi data yang siap ditampilkan.

3. *Controller* mencakup semua proses yang terkait dengan pemanggilan basis data dan kapsulisasi proses-proses utama. Controller bertindak sebagai penghubung *model* dan *view*. *Controller* berisikan class dan fungsi yang memproses permintaan dari view ke dalam struktur data di dalam *model*.

2.11. Laravel

2.11.1 Pengertian Laravel

Laravel merupakan suatu *framework PHP* yang gratis dan *open-sourc*e. Diciptakan oleh *Taylor Otwell* pada tahun 2011. *Laravel* merupakan salah satu *Framework PHP* yang paling populer pada 2021 berdasarkan beberapa artikel. Alasannya karena Laravel ditujukan untuk pengembangan aplikasi web. Laravel merupakan framework yang bersifat *highly extensible* yang artinya para pengembang dapat memanfaatkan *ready-to-use first-party-packages* seperti *Envoy, Passport, Horizon, Cashier, Jetsream* dan *fortify*. Salah satu fitur yang memudahkan pengembangan adalah proses manajemen database, dengan *Elequent ORM* penulisan tidak perlu menulis *query* secara rumit. *Elequent ORM* merupakan *API Interface* yang berinteraksi langsung dengan *database*. Ada juga fitur-fitur Laravel seperti *database seeder, migration* tentu sangat memudahkan pengembang dalam membangun aplikasi web yang diinginkan.

2.11.2 Tool Laravel

Dua Tool berikut merupakan ciri khas dari laravel yang jarang dimiliki oleh framework lain yaitu Composer dan Artisan. Berikut kegunaannya.

1. *Composer* adalah suatu tool yang berfungsi sebagai manajemen depedencies PHP. Pada dasarnya Laravel memiliki banyak *library* (*packages, modules, plug-ins* dan *component*) salah satu yang umum digunakan seperti contoh *User Authentication, No Captha* dan masih banyak lainnya. Untuk kita memanggil *library* Laravel secara manual tentu akan sangat sulit

dikarenakan tiap *library* diduga juga mempunyai ketergantungan (*Depedencies)* akan *library* lainnya, pada ujungnya akan menyebabkan kekacauan pada project yang dikembangkan. *Composer* berkerja untuk mengatur tiap ketergantungan/*depedencies* pada *PHP*, kita tinggal memanggil saja *library* yang kita inginkan tanpa perlu memikirkan *depedencies* akan *library* yang kita panggil.

Kumpulan *library* ditempatkan pada "packagist.com". *dependencies* disimpan menggunakan format file "composer.json" sehingga dapat ditempatkan didalam *folder* utama *website*. *Composer* membantu dalam menginstall sebuah *library* tanpa perlu menginstallnya secara satu persatu, *Composer* secara otomatis akan menginstall *library* yang kita butuhkan. Misalnya *Library Laravel* dan secara otomatis *library* lain yang terkait dengan akan terinstall. Begitu pun ketika ingin memperbarui *library*, cukup menggunakan perintah " `$ composer update`" dan seluruh *library* akan diperbarui secara otomatis.

2. *Artisan* merupakan *command line interface* yang dimiliki oleh Laravel. Artisan mencakup sekumpulan perintah yang membantu untuk membangun sebuah *website* atau aplikasi web. Fitur-fitur pada laravel dijalankan dengan awalan *php artisan* dengan *artisan* pengembang tidak perlu repot untuk membuat *folder* kemudian membuat *controller* secara manual, dengan artisan user hanya perlu memasukan *command line.*

   " `$ php artisan make:controller AdminController`".

   Kumpulan perintah Artisan juga termasuk penggabungan dengan *framework Symphony* yang menghasilkan fitur add-on di Laravel.

3. *Livewire* adalah *tools/*perkakas yang dibuat khusus untuk *framework Laravel*, yang berfungsi menjadikan *website* menjadi dinamis tanpa perlu melakukan kodingan yang rumit (*complexity*). *Livewire* merupakan suatu *tools full-stack*, artinya berkerja pada bagian frontend dan backend. *Tools full-stack* yang mungkin kita pernah dengar seperti *Vue.js* dan *React*, dua tools tersebut merupakan *framework Javascript* yang bertujuan

mempermudah *frontend development.* Dibandingkan dengan kedua *tools* tersebut *Livewire* dirancang khusus untuk *framework Laravel, Output* yang dihasilkan *Livewire* berupa *Component* dan `blade.php` yang merupakan extensi *View* yang hanya ditemukan pada *Laravel*.

4. *Laravel Breeze* adalah suatu *package* atau kumpulan fungsi yang mengolah autentikasi user seperti register user dan login user. Sebelumnya pada laravel versi 7 *package* Autentikasi ini sangat mudah bagi *developer* untuk mengkonfigurasi fungsi yang ada didalamnya karena *path component* tidak begitu rumit dan gampang diikuti. Berbeda dengan *Laravel 8* yang digunakan pada penelitian ini, dalam pembuatan sistem autetifikasi Laravel berkolaborasi dengan *Jetstream* dan *Tailwind*, menjadikan tampilan *Laravel* menjadi sangat cantik dan elegan dibandingkan dengan versi sebelumnya. Tetapi dengan perubahan yang besar itu menyebabkan *laravel* sangat sulit untuk dibedah karena tentunya sangat kompleks dan banyak component yang tersembunyi. Disini Taylor Otwell memberikan solusi agar *Laravel* lebih mudah digunakan tanpa adanya *Jetstream*. Dengan *Laravel breeze* menjadikan *Laravel 8* dapat seperti versi sebelumnya tapi dengan lebih elegan dengan sentuhan *Tailwind*.

2.12. JSfiddle

Jsfiddle merupakan aplikasi web yang berguna sebagai *IDE Online* yang dapat menjalankan dan menyimpan codingan *html*, *css*, *javascript* dan lain-lain. Untuk dapat menggunakan *API GoogleMaps* , dibutuhkan *API key* untuk dapat bisa diterapkan diaplikasi kita. *API key* diberikan jika telah memenuhi syarat kebijakan dari google seperti pembayaran dan lain-lain. Bila belum memiliki *API Key Jsfiddle* menjadi solusi alternatif jika kita ingin melakukan pengembangan sebelum diterapkan pada aplikasi web yang kita buat.

2.13. Adobe Photoshop

Photoshop adalah sebuah *software* yang populer untuk memodifikasi foto maupun objek gambar dengan bermacam ekstensi file. Disamping dari itu Photoshop dapat digunakan untuk membuat logo, pamflet, poster dan masih banyak lainnya. Photoshop mempunyai banyak sekali fitur, tapi salah satu fitur yang sering digunakan adalah melakukan suatu seleksi dengan *magic wand*. Pemakaiannya yang simpel dan bantuan video ajar yang sudah lengkap diberbagai media sosial, tentu membuat siapa saja dapat menggunakan *software photoshop*.

2.14. Docker

Docker merupakan suatu *platform* untuk mempermudah pengembangan aplikasi. Karena didalam *docker*, aplikasi yang dirancang akan dipisah (*images*) berdasarkan infrastrukturnya, sehingga penyaluran aplikasi dalam lingkungan pengembangan akan lebih cepat. Docker memiliki kemampuan untuk mem-package aplikasi dan menjalankan aplikasi yang dimana aplikasi tersebut insfrastuktur-insfrastrukturnya dalam keadaan terisolasi yang disebut *container*. Docker merupakan sebuah *open source* pada *platform* kontainerisasi. Setiap konteiner berkerja secara isolasi sehingga konteiner-konteiner yang lain tidak akan saling terpengaruh dan dengan begitu konteiner-konteiner tersebut dapat terkontrol, Docker dibuat untuk *Linux kernel* seperti *Ubuntu* namun docker dapat juga diakses dan diunduh pada *Mac* dan *Windows*. Kontainer memiliki tujuan yang sama seperti dengan *Virtual Machine* tetapi jika dibandingkan konteiner jauh lebih ringan karena tidak akan menghidupkan seluruh layanan pada OS dan tidak perlu menggunakan *hypervisior* ,sehingga konteiner hanya perlu menggunakan proses OS dan dependencies yang dibutuhkan oleh kode aplikasi. Dari sisi lain juga Docker merupakan salah satu *Container Manager* yang paling populer sehingga hampir semua *cloud provider* sudah menyediakan layanan Docker.

2.14.1 *Container*

Container merupakan suatu wadah untuk membungkus *(package)* aplikasi dan menjanlankan aplikasi secera terpisah. Dengan terisolasi dan keamanan yang tinggi memastikan pengembang menjalankan banyak container-container secara berkala dalam sebuah host saja. *Container* bersifat ringan, jika ada suatu *images* yang telah terinstal atau pernah terinstal dalam *container* atau *container* yang berbeda maka developer tidak perlu lagi untuk mengunggah dari awal lagi.

2.14.2 *Images*

Sebuah *image* merupakan suatu file digunakan untuk mengekseksui kode yang terdapat di Docker container. Dokcer imgaes digambarkan sebagai suatu instruksi-instruksi untuk membagun docker conteiner dan juga titik awal ketika menggunakan docker.

2.15. Sistem Operasi Ubuntu

Ubuntu Merupakan OS (Operating Sistem) berbasis pada Linux yang bersifat open source. Kepanjangan LTS yaitu *Long Term Services* yang berarti versi tersebut mendapatkan *maintenance* dan keamanannya terjaga selama 5 tahun (sampai tahun 2027). Sistem Operasi Ubuntu sangat cocok digunakan diberbagai perangkat. Seperti laptop atau komputer, selular genggam, dan digunakan pada server. Sistem Operasi Ubuntu hanya perlu memerlukan spesifikasi komputer yang rendah sehingga sangat cocok pada *server* atau jenis *server* lainnya seperti *database server*, *web server* dll. Ubuntu ini dapat diunduh dan digunakan pada OS Windows khususnya Ubuntu dengan *terminal environment* yang akan dijalankan dengan *Windows Subsystem for Linux* (WSL).

2.16. Google Cloud Platform

Google cloud merupakan penyedia layanan-layanan *cloud* yang tentu bermanfaat untuk bisnis maupun tahap pengembangan. Untuk dapat mengakses *Google Cloud* tentu harus memiliki akun *Google* terlebih dahulu, kemudian mengisi data-data seperti kartu kredit dan lain-lain. Setelah berhasil melakukan pengsian data,

Google cloud dapat diakses. Halaman *Google Cloud* disebut dengan *Google Cloud Console* disana terdapat *Dashboard, activity, recommendations* berfungsi untuk memonitoring produk-produk yang sedang dipakai juga menganalisis dan merekomendasikan kebutuhan-kebutuhan kita yang *google cloud* tawarkan, mengenai pembayaran *Google cloud* dalam tahap pengembangan belum dibebani pembayaran dalam hal ini pengguna perdana mendapatakan 400$ selama 4 bulan, jika penggunaan layanan melewati limit dari yang disediakan maka akan dibebani uang sesuai apa yang sudah digunakan. Pada sistem pelaporan drainase ini menggunakan beberapa produk seperti:

1. *Google Maps API* merupakan penyedia layanan peta dari pengembangnya yaitu *Google.* Dengan *Google maps API* aplikasi yang kita kebangkan dapat menggunakan layanan *google maps* selain itu pengembang juga dapat mengkonstruksi peta yang ada di *google maps* sehingga peta dapat dinamis sesuai kebutuhan pengembang. Seluruh variabel-variabel dan contoh yang terdapat pada google maps dapat diakses pada *google docs* sehingga sangat membantu pengembang. *Google Maps API* berupa link dan berisikan kode unik yang bersifat rahasia sehingga *Google maps* terbatas dan dapat di monitoring tiap penggunaan *Google maps API* ini.
2. *Virtual Machine (VM)* merupakan instansi dari *Compute Engine. Virtual Machine* merupakan layanan yang dapat menjalankan *images* yang umum pada *Linux* dan *Windows Server* , pengembang juga dapat memasukan *Docker containers* yang dimana docker tersebut akan secara otomatis diluncurkan dan dijalankan oleh *Container-Optimized OS.* VM terhosting dengan infrastruktur *google.* Sehingga pengembang dapat membangun VM melalui Google Cloud Console, Google Cloud CLI atau Compute Engine API
3. *Cloud SQL* merupakan layanan database yang mengatur keseluruhan dari persiapan, memelihara, mengelola dan mengadministrasikan relasional database MySQL dalam lingkungan *Google Cloud Platform.*

Layanan *Google Cloud* tentu masih banyak lagi yang menarik dan penulis menggunakan beberapa layanan yang diperlukan saja. Selanjutnya masih ada layanan dari *google cloud* yaitu OAuth yang akan dibahas pada subbab 2.16 .

2.17. OAuth

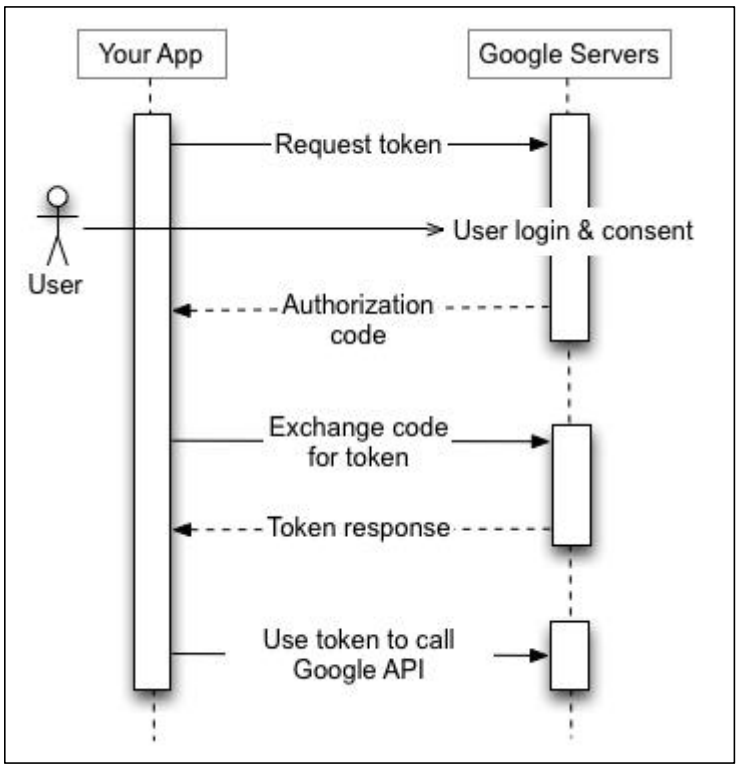
2.17.1 Pengertian OAuth (*Open Autorization*)

OAuth (*Open Authorization*) adalah protokol otorisasi standar terbuka yang memungkinkan pengguna mengakses aplikasi tanpa perlu melakukan *sign-up* kepada aplikasi web tersebut. Pemilik aplikasi web mengintegrasikan *credential* milik pengguna dengan teknologi otentikasi yang berasal dari penerbit (dalam hai ini *Google API*). Dengan *OAuth* 2.0 memungkinkan user/masyarakat yang terintegrasi dengan *Google Account* dengan mudah melakukan *sign-in* pada suatu aplikasi web.

2.17.2 Skenario Server Aplikasi Web

Google OAuth 2.0 mendukung server aplikasi web yang menggunakan bahasa pemrograman dan *framework* seperti PHP, *Java*, *Python*, *Ruby*, dan ASP.NET. Urutan otorisasi dimulai ketika aplikasi web mengalihkankan browser ke *Google URL*, URL mencangkup *parameter Query* yang mengindikasi jenis dari akses yang diminta. Google menangani *User Authentication*, pemilihan sesi (*session selection*), dan persetujuan pengguna (*user consent*). Hasilnya adalah kode otorisasi yang dapat di tukar *access token* dan *token refresh*.

Aplikasi harus menyimpan *refresh token* untuk dapat digunakan kembali dimasa mendatang dan menggunakan *access token* untuk mengakses Google API. Setelah token akses kadaluwarsa, aplikasi menggunakan refresh *token* untuk mendapatkan *refresh token* yang baru.

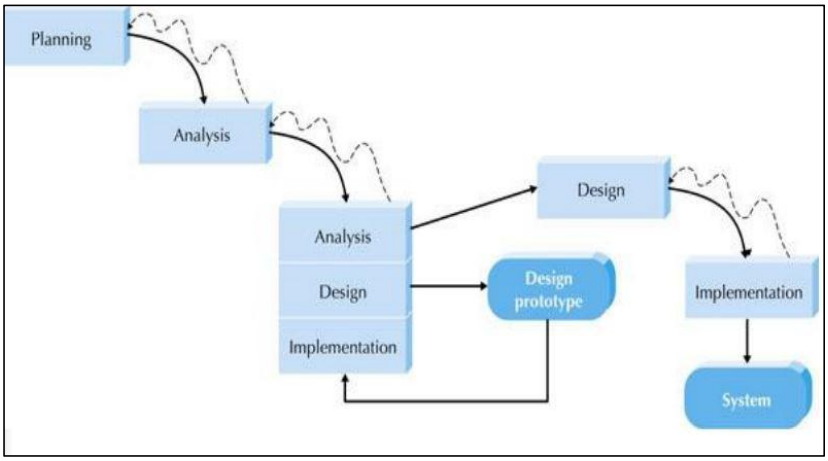


Gambar 2. 2 Skenario Server Aplikasi Web.

2.18. Rapid Application Development

Rapid Application Development adalah sebuah proses pengembangan perangkat lunak, didesain agar pengembangan dilakukan secara cepat dengan hasil akhir yang berkualitas. Proses pemodelan dalam pengembangan RAD ini menggunakan metode iteratif, dengan menekankan pada proses adaptif untuk memenuhi kebutuhan user. Pendekatan yang sering dipakai dalam RAD adalah prototipe.

2.18.1 *Throwaway Prototyping*



Gambar 2. 3 Model RAD dengan Throwaway Prototyping

Throwaway Prototyping dalam pengembangan berobjektif untuk memahami apa yang menjadi kebutuhan *customer*. Protipe berkonsetrasi pada percobaan dengan kebutuhan *customer*. Teknik *Throwaway Prototyping* ini termasuk dalam pengembangan prototipe, prototipe dalam teknik ini digunakan dalam mengeksplorasi *alternative design* dan tidak di masukan kedalam sistem yang baru. Teknik ini memiliki tahapan analis yang cukup menyeluruh untuk mengumpulkan kebutuhan dan mengembangakan ide-ide konsep sistem. Setiap masalah diperiksa dengan menganalisis, merangcang, dan membangun prototipe desain.

2.19. Unified Modeling Language (UML)

*Unified Modeling Langua*ge adalah suatu standar bahasa pemodelan untuk *software* dan pengembangan sistem. Dalam merancang sistem tidak terlepas dari kompleksitas, untuk mengelola sesuatu yang kompleks ini dibutuhkan sebuah modeling. *Modeling* membantu developer untuk tetap fokus pada dokumentasi dan mengkomunikasikan aspek utama tentang suatu rancangan sistem. *Modeling Language* dapat berupa *pseudoo-code*, kodingan, gambar, digaram, atau juga berupa penjelasan deskripsi. Harus diketahui bahwa UML memiliki syarat penting seperti

tidak boleh ambigu, bertele-tele (*verbose*), atau menggunakan bahasa sehari-hari. Umumnya penjabaran sistem yang berfokus pada *source code* saja tidak dapat memberitahu bagaimana penggunaan *software* dan oleh siapa yang menjadi akan menjadi penggunanya. Tetapi tentu kodingan adalah suatu bahasa untuk *software* developer dan bukan untuk *stakeholder* yang lain, seperti *customers* dan *system designers*.

UML berkembang seiring dengan perkembangan *software*, pada versi mula-mula *UML* di buat agar orang mengkomunikasikan desain secara tidak ambigu, menjabarkan inti dari suatu *design* dan memetakan kebutuhan fungsi untuk solusi sistem orang-orang. Versi awal *UML* adalah *UML* 1.x lalu *UML* 1.5 dan *UML* 2.0 . Berikut adalah penejelasan pemodelan tipe diagram pada UML:

1. *Use Case* : Berinteraksi antara Sistem dengan sistem yang lain atau juga dengan pengguna. Sangat membantu dalam pemetaan kebutuhan sebuah sistem. Sudah ada sejak versi *UML* 1.x .
2. *Activity* : *Activity* diagram dapat menentukan bagaimana cara suatu sistem mencapai tujuannya. *Activity* diagram menampilkan proses aksi tingkat tinggi yang terjadi didalam sistem. *Activity* diagrams salah satu diagram *UML* yang paling mudah dijumpai semenjak penerapannya menggunakan simbol layaknya notasi *flowchart*. Sudah ada sejak versi *UML* 1.x .
3. *Class* : *Classes* mendeskripsikan perbedaan tipe-tipe dari sebuah objek di miliki oleh sistem tersebut lalu *class* diagram menampilkan para *classes* ini bersama hubungan antar *class* yang lain. Dalam sebuah *class* berisikan detail *class* seperti nama *class*, attribut, operasi, dan visibilitas. Sudah ada sejak versi *UML* 1.x
4. *Object* : Membantu mendefinisikan *classes* ke suatu *class* diagram, didalam mengkonfigurasi ini sangat penting bagi sistem. *Object* diagram membantu menjelaskan bagaimana sebuah objek didalam sistem bekerja secara berasama dengan skenario khusus. Sudah ada sejak versi *UML* 1.x .

5. *Sequence* : Menampilkan interaksi antara objek dengan perintah. *Sequence diagrams* berkaitan penting dalam member dari grup *interaction* diagram, karena berinteraksi penting antara *runtime* dengan sistem juga bagian dari model sudut pandang *logical*. Disamping itu *sequence* diagram juga berkaitan dengan *communication diagrams* dan *timing* diagrams karena membantu secara tepat memodelkan bagaimana bagian-bagian dari sistem berinteraksi. Sudah ada sejak versi *UML* 1.x .

6. *Communication* : *Communication* diagrams menambahkan prespektif yang lain dengan lebih menghubungkan (*links*) antara para pengguna. Ini sangat bagus dalam hal menampilkan kebutuhan suatu hubungan (*links*), misalnya sesama pengguna mengirimkan suatu pesan. Pada versi *UML* 1.x *Communication diagrams* ini dulunya bernama *collaboration diagrams*.

7. *Timing* : Interaksi antara objek-objek dimana waktu yang menjadi fokus utama. Interaksi *timing* pada umumnya berorientasi dengan *real-time*, dalam sebuah timing diagram setiap *event* memiliki informasi waktu yang terkandung didalamnya. Secara rinci menjelaskan berapa lama suatu *event* yang dilaksanakan. Baru ada pada versi *UML* 2.0 .

8. *Interaction Overview* : membantu mengumpulkan baik itu *Sequence*, *Communication*, dan *timing diagrams* semuanya itu diambil khususnya interkasi penting yang terjadi dalam sistem.Interaction *overview* mengikat bersama suatu interaksi(*Sequence*, *Communication*, *timing*) yang berbeda-beda kedalam satu gambar utuh berisi interaksi-interaksi yang menjadi perhatian sistem tersebut. Baru ada pada versi *UML* 2.0 .

9. *Composite Structure* : membantu dalam mengisi detail mengenai sistem karena ada suatu jurang pemisah diantara beberapa *UML* diagrams bila digabungkan seperti *class* dan *sequence* diagrams. *Composite Structure* menampilkan bagaimana objek-objek membentuk sebuah gambaran besar, dengan memodelkan bagaiman objek tersebut bekerja bersama didalam sebuah *class* dan mencapai tujuannya. Baru ada pada versi *UML* 2.0 .

10. *Component*: Digunakan untuk mengorganisir sistem menjadi sesuatu yang dapat dikelola kembali, digunakan kembali ,dan bagian didalam sistem dapat diganti. Sudah ada sejak versi *UML* 1.x .
11. *Package* : *Package diagrams* sering sekali digunakan karena bertujuan menampilkan ketergantungan antara *packages*. *Package* akan sangat berpengaruh jika *package* yang lain ada yang terubah, memahami dependensi antara package merupakan hal yang vital dalam menstabilkan suatu aplikasi. *Package diagrams* baru dirilis pada *UML* 2.0 .
12. *State Machine* : Membantu memodelkan sebuah *states* dari suatu objek dan sebuah *events* yang menyebabkan perubahan atau transisi keadaan pada sebuah objek. Sudah ada sejak versi *UML* 1.x .
13. *Deployment* : Menampilkan pemodelan pada tingkat *Physical view*, *Physical view* berfokus pada elemen fisik pada sebuah sistem seperti melakukan eksekusi file *software* atau pada *hardware*. *Deployment* ini sudah ada sejak versi *UML* 1.x .

2.20. WebSocket

WebSocket adalah protokol komunikasi dua arah yang memungkinkan pertukaran data *real-time* antara klien dan server melalui koneksi jaringan yang tetap terbuka. Dalam konteks pengembangan aplikasi web, *WebSocket* memberikan kemampuan interaktif dan responsif, memungkinkan pembaruan data secara langsung tanpa harus melakukan permintaan ulang (request) dari klien. *Broadcasting*, dalam konteks aplikasi web, adalah teknik untuk mengirimkan pesan dari server ke banyak klien secara bersamaan. Ini memungkinkan aplikasi untuk menyebarkan pembaruan data secara *real-time* kepada pengguna yang terhubung.

2.20.1 *Pusher*

Pusher adalah layanan yang menyediakan infrastruktur untuk memfasilitasi broadcast real-time dalam aplikasi web. Dalam konteks *framework Laravel*, *Pusher* sering digunakan sebagai layanan *broadcasting* untuk memungkinkan komunikasi

real-time antara server dan klien. Dalam implementasi *Laravel*, dengan menggunakan *Pusher* sebagai layanan *broadcasting*, aplikasi dapat menggunakan fitur-fitur seperti *event broadcasting* dan *channel broadcasting*. *Event broadcasting* memungkinkan aplikasi untuk mendefinisikan peristiwa (*event*) yang akan disiarkan kepada klien, sedangkan *channel broadcasting* memungkinkan aplikasi untuk mengelompokkan klien dalam saluran *(channel)* tertentu sehingga pembaruan data hanya diterima oleh klien yang terhubung ke saluran tersebut.

2.21. Chart.JS

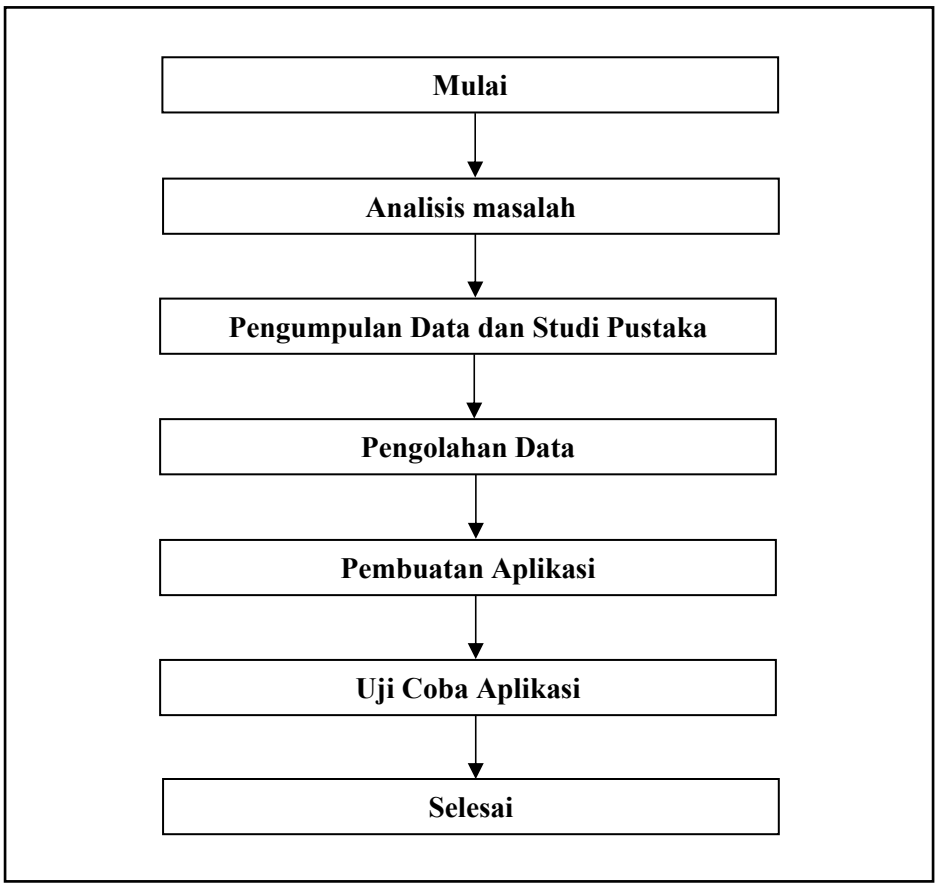
Chart.js adalah sebuah library JavaScript yang digunakan untuk membuat grafik interaktif di halaman web. Library ini menyediakan berbagai jenis grafik seperti grafik garis, grafik batang, grafik lingkaran, grafik area, dan banyak lagi. Chart.js sangat populer karena sederhana, mudah digunakan, dan memiliki dokumentasi yang lengkap. Chart.js juga mendukung fitur-fitur seperti animasi, skala sumbu yang disesuaikan, label kustom, tooltips, dan banyak lagi. Dengan demikian, Chart.js menjadi pilihan populer untuk memvisualisasikan data dalam aplikasi web.

BAB III

METODOLOGI PENELITIAN

3.1. Alur Pembuatan Aplikasi

Tahapan melakukan penelitian ini dimulai dari Analasis masalah, dan berakhir pada Uji coba aplikasi. Untuk lebih rinci dapat dilihat sebagai berikut:



Gambar 3. 4 Skenario Server Aplikasi Web.

Penilitian ini muncul dari suatu ide atau buah pemikiran yang dimana adalah merancang suatu aplikasi yang dapat membantu masyarakat melaporkan suatu kejadian permasalahan drainase atau selokan air, ditambah juga dengan ada postingan-postingan dimedia sosial tentang keluhan masyarakat seputar drainase, ternyata hal ini sangat ditanggapi serius oleh masyarakat. Dengan dasar itulah topik ini diangkat menjadi sebuah penelitian penulis, kemudian mulai menganalisis masalah bagaimana suatu kejadian permasalahan mengenai drainase akan berjalan beriringan dengan konsep *Crowdsourcing* dan dengan pengaplikasiannya ke dalam *Website*. Aplikasi website tentu bagus karena dapat menjangkau pengguna *PC desktop* dan Pengguna *Smartphone*. Tentu untuk merancang sebuah aplikasi website dibutuhkan sebuah *framework*. *Framework* yang dipilih adalah *framework Laravel.* Bagaimana juga menggambarkan sebuah proses dari cara pembuatan laporan hingga bisa direspon oleh pemerintah atau dinas terkait, disamping itu penulis harus menganalisa isi laporan seperti apa yang menjadi pokok persoalan dan bagaimana tanggapan yang tepat oleh admin jika diterapkan pada aplikasi website. Dari gambaran tersebut data-data dan studi pustaka yang berkaitan dikumpulkan untuk menjadi acuan dalam pembuatan aplikasi dalam penelitian ini.

Dengan menggunakan *Framework Laravel* kemudian mendefinisikan fungsi yang digunakan pada *backend* dan *frontend* juga dengan rancangan tampilan *user-interface*.

3.2 Tempat Dan Waktu Penelitian

Penelitian ini akan bertempat di Kota Manado dengan target instansi yang terkait dalam hal menangani drainase atau saluran air yang bermasalah di Kota Manado, dan juga masyarakat yang berada di Kota Manado. Waktu penelitian dan juga deskripsi kegiatan dapat dilihat pada tabel dibawah ini.

Tabel 3. 2 Jadwal rencana kegiatan penelitian

No.	Deskripsi Kegiatan	Tahun 2022 Bulan											
		2	3	4	5	6	7	8	9	10	11	12	1
1.	Pengambilan data	■									■	■	
2.	Perancangan dan pemodelan	■	■										
3.	Pembuatan aplikasi	■	■	■	■	■	■	■	■	■	■		
4.	Implementasi prototipe											■	
5.	Pengujian dan analisa data											■	
6.	Pembuatan laporan tengah penelitian	■	■	■	■	■	■	■	■	■	■	■	
7.	Pengujian prototipe dan penyempurnaan	■	■	■	■	■	■	■	■	■	■	■	■
8.	Dokumentasi dan penulisan skripsi	■	■	■	■	■	■	■	■	■	■	■	■

3.3 Alat Dan Bahan

Adapun alat dan bahan yang digunakan dalam penelitian dan penulisan skripsi ini seperti *hardware* dan *software* berikut ini merupakan alat dan bahan yang digunakan:

Tabel 3. 3 Alat dan bahan beserta keterangan

No.	Langkah-langkah aktivitas penelitian	Alat dan bahan yang digunakan	keterangan
1.	Analisis dan pengembagan rancangan prototipe aplikasi	1. Laptop 2. PowerPoint 3. Sublime Text 3 4. Brave browser 5. gitbash	Spesifikasi: ● Tipe: ASUS A442uf ● RAM : 12 GB DDR4 ● Prosesor : Intel core i5 8th gen ● Hardisk 246 GB ● Windows 10 Versi 2019 Versi 4143 Versi 1.45.133
2.	Perancangan aset marker peta	1. Adobe photoshop CS4	
3.	Pembuatan aplikasi	1. Docker Desktop 2. Ubuntu LTS 3. HeidiSQL 4. Microsoft Edge	Versi 4.9.0 Versi 22.04.1

3.4 Metode Pengambilan Data

3.4.1 Metode Observasi

Penulis melakukan observasi diberbagai media sosial untuk mendapatkan informasi bentuk keluhan masyarakat, demi bisa memahami dan dapat menkonstruksi sebuah wadah yang tepat bagi masyarakat menyalurkan keluhan mereka.

3.4.2 Metode Kepustakaan

Metode Kepustakaan dilkakukan agar penulis dapat juga memahami dasar teori dari setiap instrumen yang akan dipakai. Metode ini dilakukan dengan cara membaca literatur-literatur baik dalam bentuk buku juga tulisan dari sumber terpercaya di internet.

Sebagai salah satu contoh adalah literatur dari F*ramework Laravel ,* pengertian *Crowrdsourcing, Google Cloud, Docker* dan pengertian dari drainase. Sumber dan redaksi tersebut dapat dilihat pada daftar pustaka.

3.4.3 Metode Wawancara

Dalam penelitian ini penulis melakukan wawancara dengan pihak-pihak yang berkaitan seperti kepala lingkungan dan masyarakat. Dalam hal ini data-data yang diperlukan peneliti sehingga melakukan metode wawancara ini adalah:

1. Bagaimana cara atau proses masyarakat jika ingin melaporkan drainase/got bermasalah (penuh sampah, penuh lumpur, trotoar berlubang dll)?
2. Pernahkah kepala lingkungan mendapatkan intruksi dari Pemerintah Kota Manado untuk menangani peremajaan drainase/got disekitar domisili anda?

Selain melakukan wawancara secara langsung dibagikan kuisioner untuk pengembangan aplikasi ini.

3.5 Metode Perancangan Sistem

3.5.1 Analisis Sistem Aktual

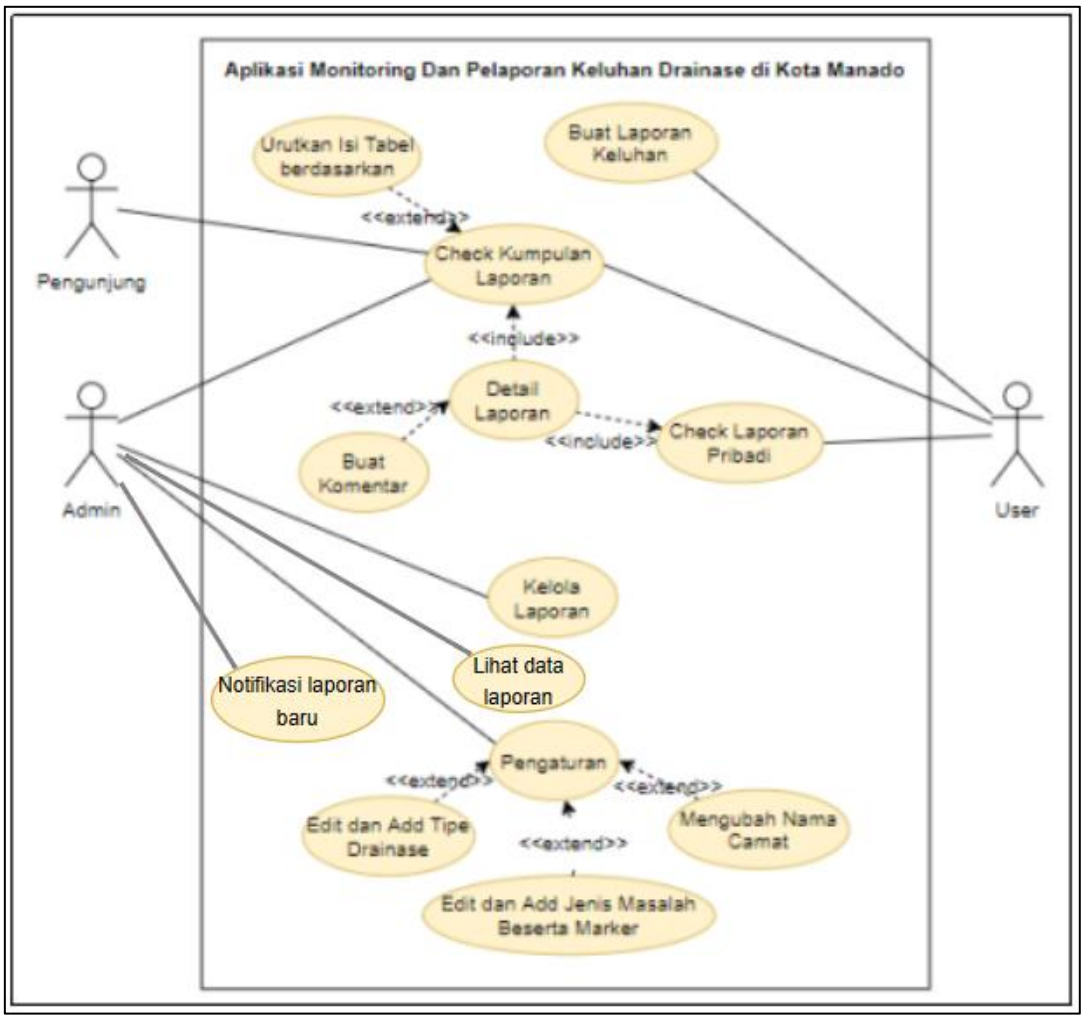
Dalam analisis ini, penulis menemukan aplikasi pelaporan drainase tetapi memiliki kekurangan-kekurangan yang dapat digambarkan seperti aplikasi formulir. Disinilah penulis ingin menciptakan aplikasi pelaporan drainase yang dapat menjadi wadah yang tepat bagi masyarakat mengaspirasikan keluhan mereka.

3.5.2 Perancangan Sistem Baru

Dalama perancangan aplikasi ini ada fitur-fitur yang menjadi fokus utama dalam pembuatannya. Seperti pembuatan laporan ditambah titik kordinat peta agar membantu ketepatan titik lokasi yang bermasalah. Pada halaman awal akan langsung ditampilkan kumpulan laporan, juga tiap-tiap jenis permasalahan drainase akan memiliki warna markernya tersendiri yang ditampilkan dipeta. Selain itu ditiap laporan akan memiliki kolom komentar sehingga masyarakat dan admin dapat saling berkomunikasi. Pada admin selain mengolah laporan, admin juga ada fitur untuk dapat mengkostumisasi baik marker warna marker menambahkan jenis permasalahan, menambah jenis tipe drainase, mengubah nama camat. Fitur lainnya adalah masyarakat dapat melakukan login dengan akun google.

3.6 Perancangan Aplikasi

Perancangan dimulai dengan tampilan *interface* terlebih dahulu dengan mengilustrasikan User akan membuat sebuah laporan, lalu menganalisis tampilan awal yang efisien bagi pengujung dan lain sebagainnya. kemudian diikuti dengan fungsi-fungsi yang mendukung berdasarkan analisis interface yang sebelumnya. Untuk lebih rinci dalam perencanaan aplikasi yang dibangun akan digambarkan dengan *UML*, diagram *UML* pada bagian ini memakai diagram *Use Case, Activity, Class, Object, Sequence.* Use Case diagram digunakan untuk menggambarkan hubungan antara aktor dan sistem.disini terdapat 3 aktor, aktor User yang merupakan masyarakat atau warga yang telah Melakukan Login, Pengunjung merupakan masyarakat atau warga yang tidak login, dan aktor Admin.



Gambar 3. 5 Use Case Interkasi pengguna dengan Aplikasi.

Tabel 3. 4 Use Case dari Buat Laporan Keluhan.

NAMA USE CASE	BUAT LAPORAN KELUHAN
TUJUAN KONTEKS	User dapat membuat laporan dengan memasukan data berupa formulir, titik koordinat, dan memasukan foto.
PRAKONDISI	User harus terlebih dahulu terautentikasi didalam database.
KONDISI AKHIR JIKA SUKSES	Akan muncul sebuah label bertuliskan “Laporan berhasil dibuat”. Laporan yang dibuat akan terlihat di

	halaman utama, kelola admin dan check laporan saya.
KONDISI AKHIR JIKA GAGAL	Laporan akan ditolak jika tidak melengkapi formulir , titik koordinat dan foto.
AKTOR UTAMA	User
AKTOR KEDUA	-
TRIGGER	User menekan tombol buat laporan.
ALUR DALAM KONTEKS	1. User membuka sidebar, dengan menekan logo humberger pada pojok kiri atas. 2. User menekan tombol “Buat Laporan”. 3. User memasukan formulir keluhan. 4. User menekan tombol “Buka Map” untuk menampilkan peta pada sebuah modal. 5. User mencari titik koordinat dengan cara klik marker kemudian tahan dan tarik. Atau bisa memasukan lokasi pada kolom “cari lokasi”. 6. User menekan tombol “Save Changes”. 7. User mengupload foto dengan menekan tombol “Choose Files”. 8. User menekan tombol Submit. 9. Akan terpampang label “Laporan berhasil dibuat”.

Tabel 3. 5 Use Case dari Check Kumpulan Laporan.

NAMA USE CASE	CHECK KUMPULAN LAPORAN
Tujuan Konteks	Semua orang dapat melihat laporan-laporan yang telah dibuatkan sebelumnya oleh user dan sudah dikelola oleh admin
PRAKONDISI	-
KONDISI AKHIR JIKA	Marker yang ditampilkan dalam peta sesuai dengan

SUKSES	jumlah isi tabel.
KONDISI AKHIR JIKA GAGAL	Peta tidak memiliki isi.
AKTOR UTAMA	User
AKTOR KEDUA	Admin
AKTOR KETIGA	Pengunjung
TRIGGER	Pengguna telah membuat laporan-laporan akan ditampilkan dalam kumpulan laporan.
ALUR DALAM KONTEKS	1. Pengguna mengakses URL http://localhost:8000/dashboard 2. Tunggu Map selesai memuat. 3. Daftar laporan dan laporan ditampilkan. 4. Pengguna dapat mengurutkan isi tabel
EKSTENSI	4.1 Setiap baris yang terdapat anak panah (ascending) keatas atau kebawah (descending). 4.2 Pengguna menekan panah salah satu baris untuk mengurutkan isi tabel. 4.3 Tabel telah diurutkan.

Tabel 3. 6 Use Case dari detail laporan.

NAMA USE CASE	DETAIL LAPORAN
Tujuan Konteks	Semua orang dapat melihat secara rinci sebuah laporan.
PRAKONDISI	Laporan harus tersedia didalam daftar laporan yang ada didashboard dan di Laporan Pribadi.
KONDISI AKHIR JIKA SUKSES	Data dan map dapat ditampilkan.
KONDISI AKHIR JIKA	Data dan map kosong.

GAGAL	
AKTOR UTAMA	User
AKTOR KEDUA	Admin
AKTOR KETIGA	Pengujung
TRIGGER	Pengguna menekan salah satu laporan yang ada dalam tabel. Atau marker yang ada peta
INCLUDED CASES	**CHECK KUMPULAN LAPORAN, CHECK LAPORAN PRIBADI.**
ALUR DALAM KONTEKS	1. Pada tampilan dashboard, Pengguna memilih salah satu laporan yang ingin dilihat secara detail, yang terpampang pada tabel. 2. Pengguna menekan salah satu kolom ditabel. 3. Akan muncul modal yang berisi map dan rincian dari laporan terpilih. 4. Tekan tombol close jika ingin menutup modal.
EKSTENSI	3.1 Pengguna dan admin dapat memberikan komentar. 3.2 Pengguna yang dapat memberikan komentar adalah pengguna yang sudah login. 3.3 pengguna scroll kebawah untuk melihat kolom komentar. 3.4 Pengguna mengisi komentar pada kolom komentar. Dan menekan tombol kirim. 3.5 Pengguna dapat melihat isi komentar-komentar sebelumnya dibagian bawah.

Tabel 3. 7 Use Case dari Check Laporan Pribadi

NAMA USE CASE	**CHECK LAPORAN PRIBADI**

Tujuan Konteks	User dapat melihat hasil laporannya sendiri tanpa terganggu dengan laporan orang lain.
PRAKONDISI	User sudah pernah membuat laporan minimal sekali. Dan sudah melakukan login
KONDISI AKHIR JIKA SUKSES	Laporan tersedia.
KONDISI AKHIR JIKA GAGAL	Laporan tidak tersedia
AKTOR UTAMA	User
AKTOR KEDUA	-
TRIGGER	Id User identik dengan id user didalam laporan
ALUR DALAM KONTEKS	1. Pengguna yang telah sign in dan berada didashboard membuka tombol humberger agar sidebar yang tersembunyi akan ditampilkan 2. Pengguna menekan tombol “Laporan anda”. Atau bisa langsung memsukan URL localhost:8000/show 3. Sistem akan menampilkan tabel yang berisi laporan-laporan yang pernah dibuat oleh pengguna. 4. Pengguna dapat menekan salah satu laporan didalam tabel dan menampilkan detail suatu laporan yang di tekan tersebut.

Tabel 3. 8 Use Case dari Kelola Laporan

NAMA USE CASE	**KELOLA LAPORAN**
Tujuan Konteks	Admin merespon laporan-laporan dari pengunjung, pengolahan laporan dengan memberikan status, jenis permasalahan dan tipe drainase pelapor.

PRAKONDISI	Admin yang dapat mengakses kelola laporan. Dan memiliki pemahaman konsep tentang drainase.
KONDISI AKHIR JIKA SUKSES	Laporan akan mendapatkan status “telah dibaca”, dan pergantian jenis dan tipe.
KONDISI AKHIR JIKA GAGAL	Laporan yang tidak dipilih akan mendapatkan status ”belum dibaca”
AKTOR UTAMA	Admin
AKTOR KEDUA	-
TRIGGER	Saat Pengunjung membuat Laporan baru akan dimasukan dalam antrian untuk dikelola admin.
ALUR DALAM KONTEKS	1. User membuat laporan baru. 2. Laporan akan masuk dalam antrian dan status ”belum dibaca”. 3. Admin akan memilih laporan yang akan dikelola 4. Admin masuk kedalam detail laporan. 5. Status “telah dibaca”. 6. Admin membaca informasi detail dari laporan. 7. Admin menyimpulkan jenis permasalahan, tipe drainase. 8. Admin memberikan perkembangan laporan tersebut kedalam status. 9. Admin tekan tombol submit.
EKSTENSI	6.1 Jika informasi yang diberikan kurang jelas admin dapat menanyakan dikolom komentar.

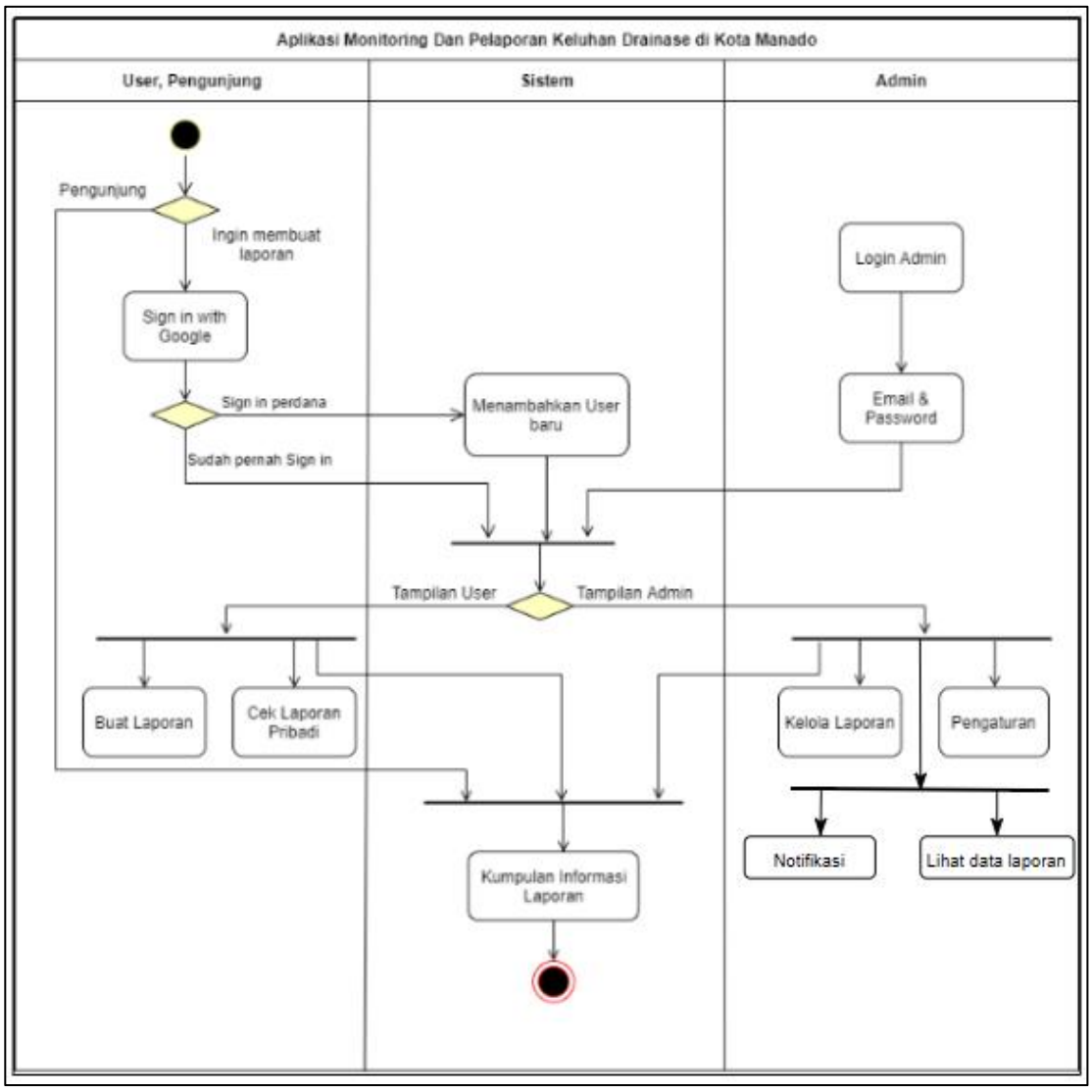
Tabel 3. 9 Use Case dari Pengaturan

NAMA USE CASE	PENGATURAN
Tujuan Konteks	Admin dapat merubah dan menambahkan baik Tipe

	drainase, Kecamatan, Jenis permasalahan dan marker sesuai kebutuhan admin.
PRAKONDISI	Admin yang dapat mengakses Pengaturan.
KONDISI AKHIR JIKA SUKSES	berhasil menambahkan atau mengubah terhadap suatu entitas-entitas.
KONDISI AKHIR JIKA GAGAL	Gagal mengubah atau menambahkan terhadap suatu entitas-entitas.
AKTOR UTAMA	Admin
AKTOR KEDUA	-
TRIGGER	Admin ingin merubah atau menambahkan nama Camat, Tipe drainase, jenis masalah beserta pilihan warna marker.
ALUR DALAM KONTEKS	1. Admin membuka pengaturan 2. Admin mengatur kecamatan. 3. Admin mengatur Tipe drainase. 4. Admin mengatur Jenis permasalahan beserta marker.
EKSTENSI	2.1 Admin memilih kecamatan yang camatnya ingin diubah. 2.2 admin memasukan camat yang ingin diubah ke dalam kolom, Tekan ok disamping kolom jika ingin menyimpan. 3.1 admin ingin membuat tipe drainase baru 3.2 admin ingin mengubah salah satu tipe drainase 3.3 admin ingin menghapus salah satu tipe drainase 4.1 Admin membuat Jenis Permasalahan baru. 4.2 Kemudian admin memilih marker warna marker

	untuk permasalahan yang baru. 4.3 admin mengubah keterangan jenis permasalahan dan warna pada marker. 4.4 admin menghapus salah satu jenis permaslahan

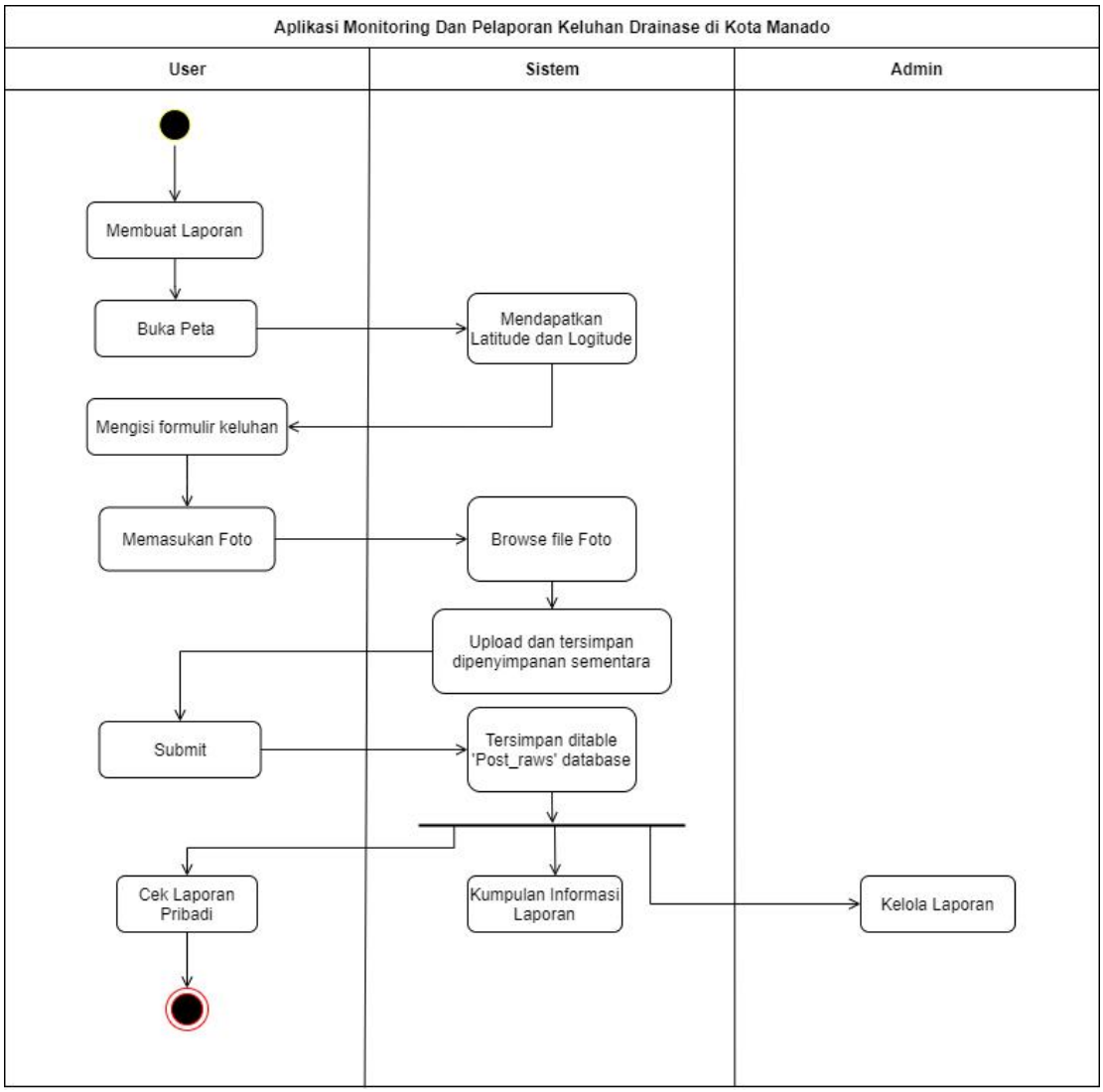
Selanjutnya menentukan alur dari *UML Activity Diagram*, alur yang akan dituturkan ada beberapa sudah disebutkan sebelumnya pada *Use Case*. Dalam alur *Activity diagram* nanti akan ada empat alur tujuan diantaranya alur pertama yang memperlihatkan keseluruhan obyek yang dapat diakses oleh User, pengunjung dan admin, alur kedua memperlihatkan proses dari Buat Laporan oleh user, alur ketiga memperlihatkan proses kelola laporan yang dilakukan oleh Admin , dan alur keempat memperlihatkan proses Pengaturan yang dilakukan oleh admin. Berikut adalah alur-alur *activity diagram* beserta penjelasannya.



Gambar 3. 6 Activity diagram yang memperlihatkan keseluruhan obyek yang dapat diakses oleh User, pengunjung dan admin

Activity diagram pada gambar 3.6 merupakan alur diagram yang memperlihatkan keseluruhan obyek yang dapat diakses oleh User, pengunjung dan admin. Tujuan dari *activity diagram* ini memperlihatkan bagaimana perbedaan antara User, Pengunjung dan admin, dan memperlihatkan bagaimana aktivitas sistem jika User baru melakukan *Sign-in*. Penggunaan *Activity diagram* dengan presentasi *swimlane* terdapat 3 lajur lajur User dengan Pengunjung, lajur Sistem, lajur Admin. Dimulai dari lajur User, Pengunjung yang memasuki aplikasi Pengujung tentu akan

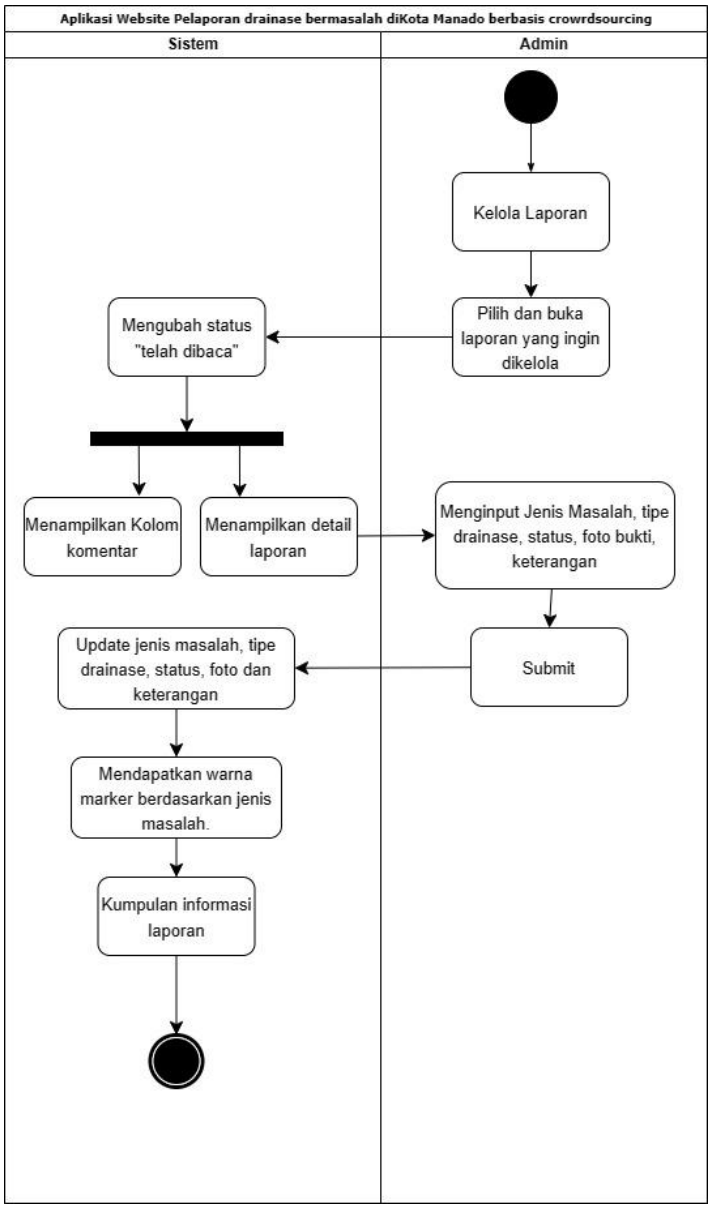
langsung disajikan dengan Informasi laporan yang tertera pada halaman awal jika Pengunjung ingin mengajukan laporan terlebih dahulu harus melakukan proses *Sign-in with google.* Kemudian sistem akan menganilisis Pengunjung yang masuk adalah user baru atau sudah pernah sign-in sebelummnya. Jika user baru perdana melakukan sign-in maka sistem akan mencoba mendaftarkan user kedalam database dengan data yang diambil dari google account. Selanjutnya user akan diarahkan ke bentuk tampilan User begitu juga dengan admin jika terautentikasi sebagai admin, akan diarahkan ketampilan admin, hal ini dilakukan agar fitur yang ada pada admin tidak akan menyatu dengan tampilan user begitu pun sebaliknya. Pada tampilan user dapat mengakses buat laporan , cek laporan pribadi dan kumpulan laporan informasi. kemudian admin dapat mengakses kelola laporan, pengaturan dan kumpulan informasi laporan dan berakhir.



Gambar 3. 7 Activity diagram Buat Laporan.

Pada gambar 3.7 Memperlihatkan alur proses pembuatan laporan yang dilakukan oleh user membuat laporan hingga dapat mencapai cek laporan pribadi. Dimulai saat user menekan tombol membuat laporan. Kemudian membuka peta kemudian mencari titik koordinat dimana permasalahan itu berasal, sistem mendapatkan titik koordinat antara *latitude* dan *longitude* tersebut dan mengirimnya ke formulir. Dalam formulir user diminta untuk memasukan Kecamatan (*dropdown menu*), Kelurahan (*dropdown menu*), Deskripsi Lokasi, Deskripsi Masalah. Lalu proses mengupload foto dimulai saat user menekan tombol *choose file* akan

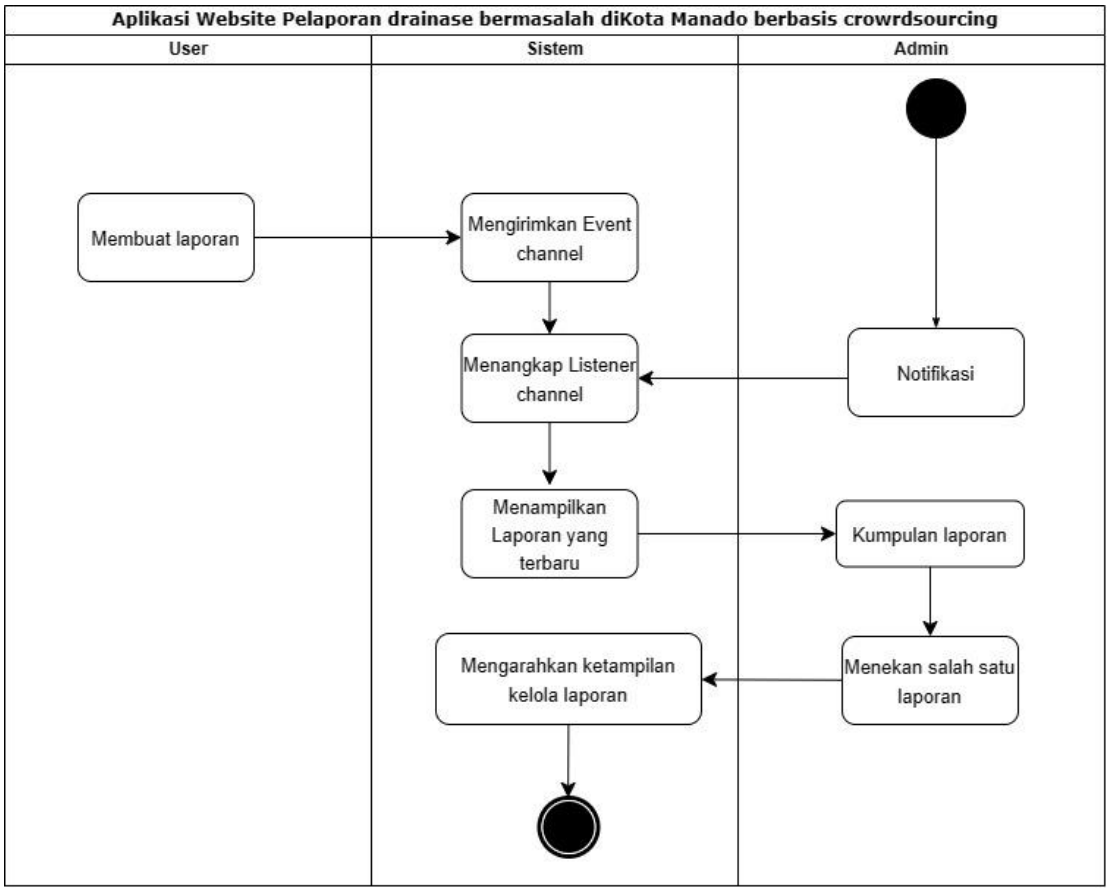
terpampang menu menjelajah file agar dengan bebas memilih foto. Setelah memilih beberapa foto dan menekan ok , sistem akan menyimpan foto tersebut kedalam penyimpanan sementara agar dapat ditampilkan pada *preview photo.* Setelah selesai, user menekan tombol *submit* untuk mempublikasikan laporan dan memastikan setiap data sudah terisi pada formulir. Sejalan dengan itu pada sisi Sistem, data-data tersebut tersimpan kedalam database dengan nama tabel “Post_raws” dan “addtionalPhotos”. Selanjutnya dari database tersebut dapat dilihat kembali kedalam kumpulan informasi laporan, kelola admin dan cek laporan pribadi. Pada cek laporan pribadi dapat melihatkan kembali laporan yang telah dibuat sebelumnya oleh user tersebut dan berakhirlah alur pembuatan laporan.



Gambar 3. 8 Activity diagram Kelola Laporan.

Pada gambar 3.8 Terdapat 2 lajur *swimlane* antara admin dengan sistem. Dimulai ketika admin menekan tombol kelola laporan, setelah itu sistem akan menampilkan tabel dengan isi laporan-laporan yang telah dibuat user. Admin kemudian memilih salah satu laporan dengan menekan tombol view. Saat admin menekan *view* sistem akan mengubah status menjadi "Telah dibaca" dan menampilkan detail dari laporan yang terpilih, juga kolom komentar untuk

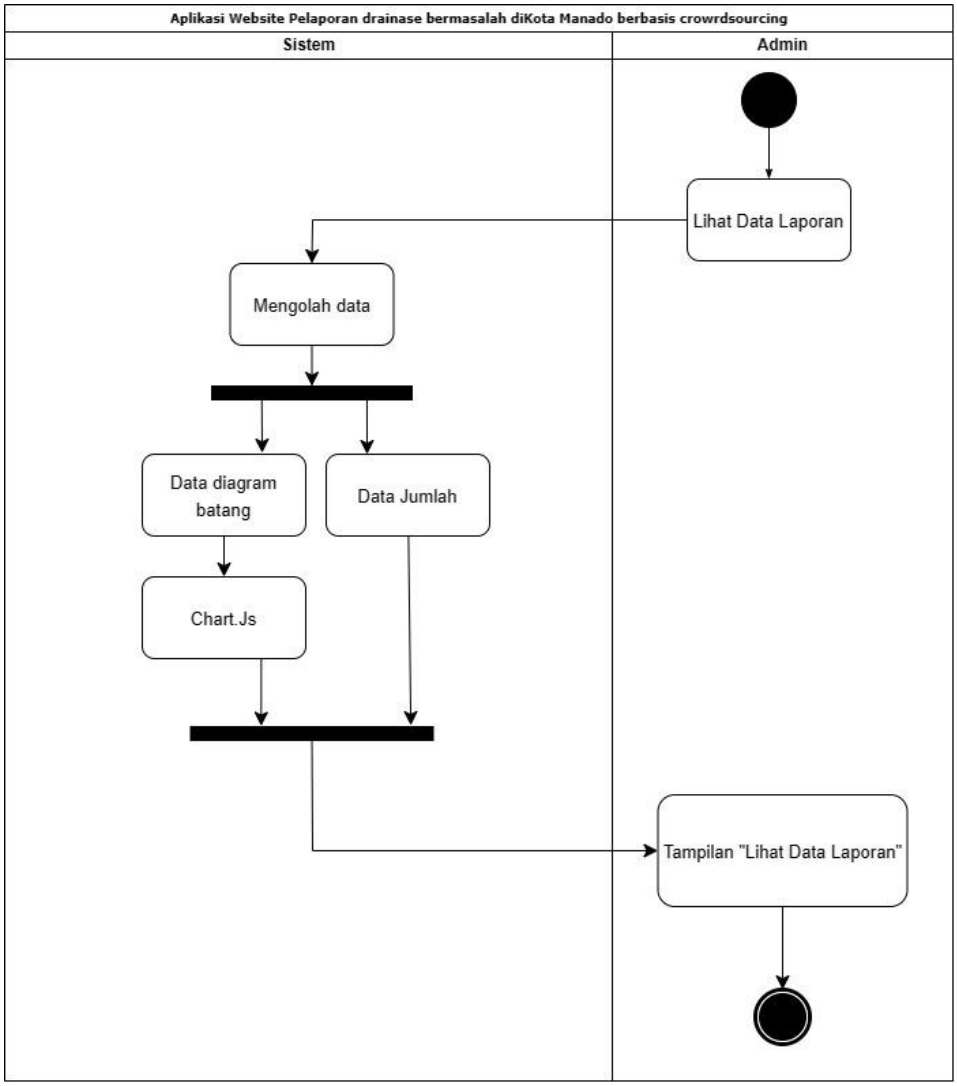
mengamati komentar-komentar dari user ataupun admin juga yang ingin memberikan komentar. Pada detail laporan terdapat Peta dengan sebuah marker yang merupakan titik yang dikeluhkan user atau pelapor, ada foto-foto yang diunggah user, Tipe drainase (yang akan diberikan oleh admin), Jenis Masalah (yang akan diberikan oleh admin), Deskripsi masalah, Deskripsi Lokasi, Kecamatan beserta camat, Kelurahan, dan Status Laporan.



Gambar 3. 9 Activity diagram Notifikasi.

Pada gambar 3.9 Terdapat 3 lajur *swimlane* antara user,admin dan sistem. Dimulai ketika admin menekan tombol notifikasi, setelah itu sistem akan menampilkan tabel dengan isi laporan-laporan yang telah dibuat user berdasarkan yang terbaru. Ketika admin dalam keadaan membuka halaman notifikasi, sistem telah dan sementara menangkap *listener* dari *channel broadcast*. Sehingga saat user

mengirimkan laporan yang baru maka *event channel* akan ikut terpanggil dan mencari *channel* yang serupa kemudian dikirimnya data terbaru dalam notifikasi secara *realtime*.



Gambar 3. 10 Activity diagram Lihat Data Laporan.

Pada gambar 3.10 Terdapat 2 lajur *swimlane* antara admin dengan sistem. Dimulai ketika admin menekan tombol “Lihat Data Laporan”, setelah itu sistem akan menampilkan mengolah data seperti menjumlahkan keseluruhan laporan, menjumlahkan kecamatan mana saja yang terlapor dan lain sebagainya. Terdapat dua tampilan data yang akan ditampilkan pertama data total keseluruhan angka dan data

yang diolah ketampilan diagram batang. Data diagram batang untuk dapat ditampilkan ke halaman menggunakan *library javascript* 'Chart.JS'.

Untuk bagian admin tersedia pada sebuah panel khusus dengan nama Admin Panel. Didalam admin panel berisi Jenis masalah (*dropdown*), Tipe Saluran/Drainase (*dropdown*), Status Laporan (*dropdown*). Setelah Admin selesai menginput data-data tersebut Admin menekan tombol *Submit* dan sistem akan menyimpan data tersebut kedalam database kemudian menyesuaikan Marker dengan warna yang telah terhubung dengan jenis permasalahan yang telah dikonfigurasi di dalam pengaturan. Laporan yang telah berhasil dikelola dapat langsung dilihat pada kumpulan informasi laporan halaman awal.

Tabel 3. 10 Penejelasan setiap jenis permasalahan drainase/saluran air.

No.	Jenis Permasalahan	Marker	Keterangan
1.	Empty	Marker putih	Belum diidentifikasi
2.	Saluran kurang dalam	Marker pink	Saluran air dianggap kurang dalam untuk dialiri air.
3.	Saluran banyak lumpur dan tanah	Marker oranye	Permukaan saluran air dipenuhi material lumpur atau pun tanah.
4.	Air disaluran kejalan saat hujan	Marker ungu	Dalam saat hujan atau pasca hujan drainase mengeuluarkan air resapan ke jalan.
5.	Saluran air tak bisa dialiri air dari jalan	Marker lime	Dalam saat hujan atau pasca hujan genangan air dijalan lama diserap atau dialirkan drainase.
6.	Sisa penggalian belum diangkat	Marker violet	Hasil pembersihan drainase yang dibiarkan dibahu jalan.

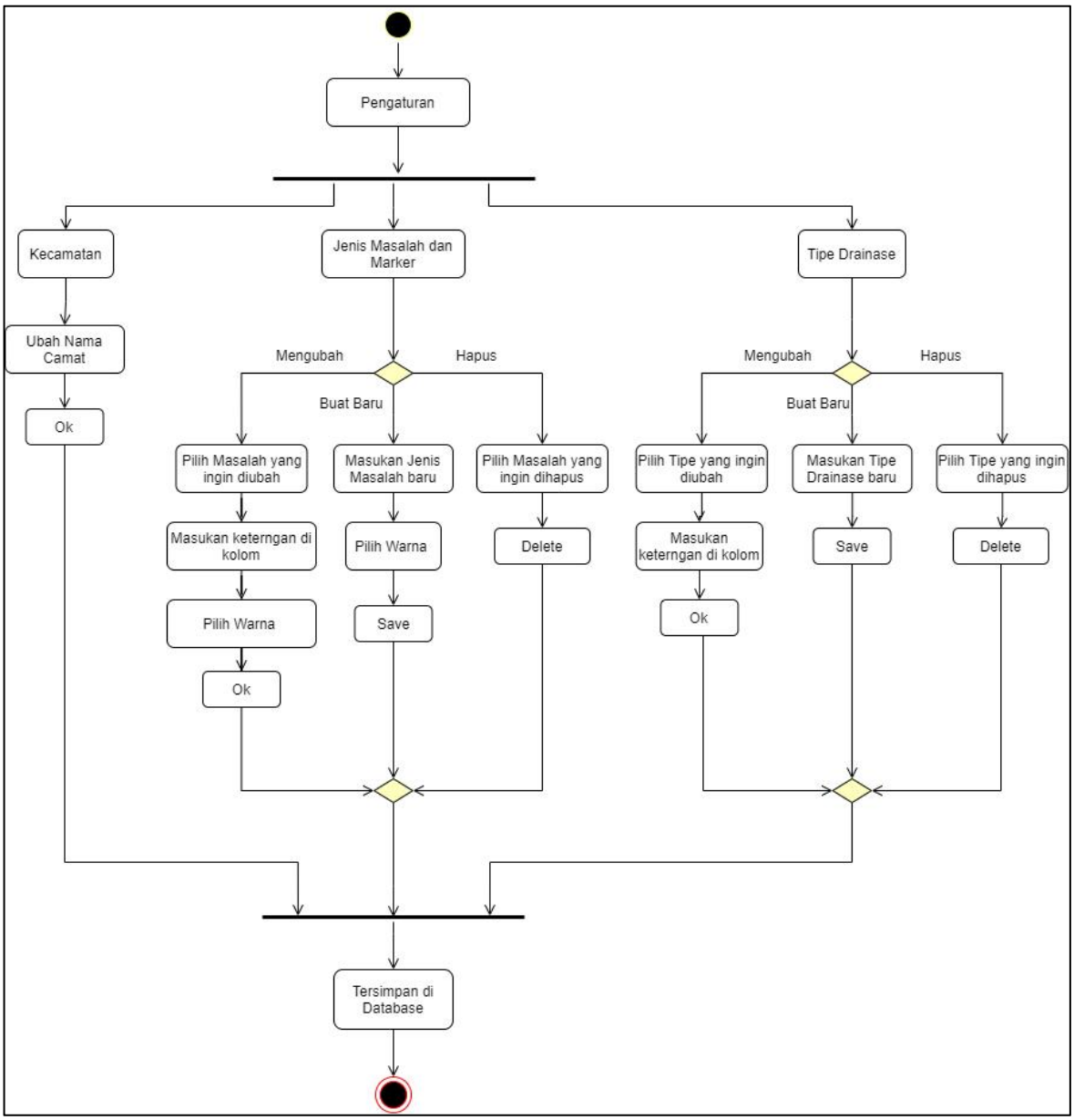
7.	Trotoar tidak tertutup atau berlubang	Marker kuning	Trotoar pejalan kaki yang memilki lubang.
8.	Trotoar tidak tertutup dan penuh sampah	Marker krem	Trotoar pejalan kaki yang memilki lubang. Dan dalamnya terlihat dipenuhi sampah
9.	Trotoar tidak tertutup dan penuh lumpur	Marker Hitam	Trotoar pejalan kaki yang memilki lubang. Dan dalamnya terlihat dipenuhi material lumpur

Tabel 3. 11 Penejelasan setiap tipe drainase/saluran air.

No.	**Tipe drainase**	**Keterangan**
1.	Empty	Belum diidetnifikasi
2.	*natural drainage*	Drainase yang terbentuk secara alami. Umumnya terbentuk dari topografi yang tinggi
3.	*artificial drainage*	Seperti saluran air dipinggiran jalan, saluran air ber-trotoar,
4.	*Sub surface drainage*	Gorong-gorong dibawah tanah, Pipa bawah tanah.
5.	Drainase Tersier	Drainase ini berada pada pemukiman warga. Umumnya saluran air berintensitas kecil
6.	Drainase Sekunder	Drainase ini mengalirkan air drainase tersier ke drainase primer.
7.	Drainase Primer	Drainase ini yang mengalirkan air menuju ke anak sungai atau ke sungai.

Tabel 3. 12 Penejelasan setiap status pada laporan.

No.	Status	Keterangan
1.	Belum dibaca	Admin belum membuka laporan tersebut.
2.	Telah dilihat	Admin telah membuka laporan tersebut.
3.	Kasus akan dirposes	Admin telah memberikan konfirmasi bahwa laporan akan diproses.
4.	Kasus sementara diproses	Admin telah menfonkirmasi bahwa laporan sementara dikerjakan pihak yang terkait.
5.	Kasus tertangani	Admin sudah memastikan bahwa laporan tersebut telah tertangani.



Gambar 3. 11 Activity diagram Pengaturan admin.

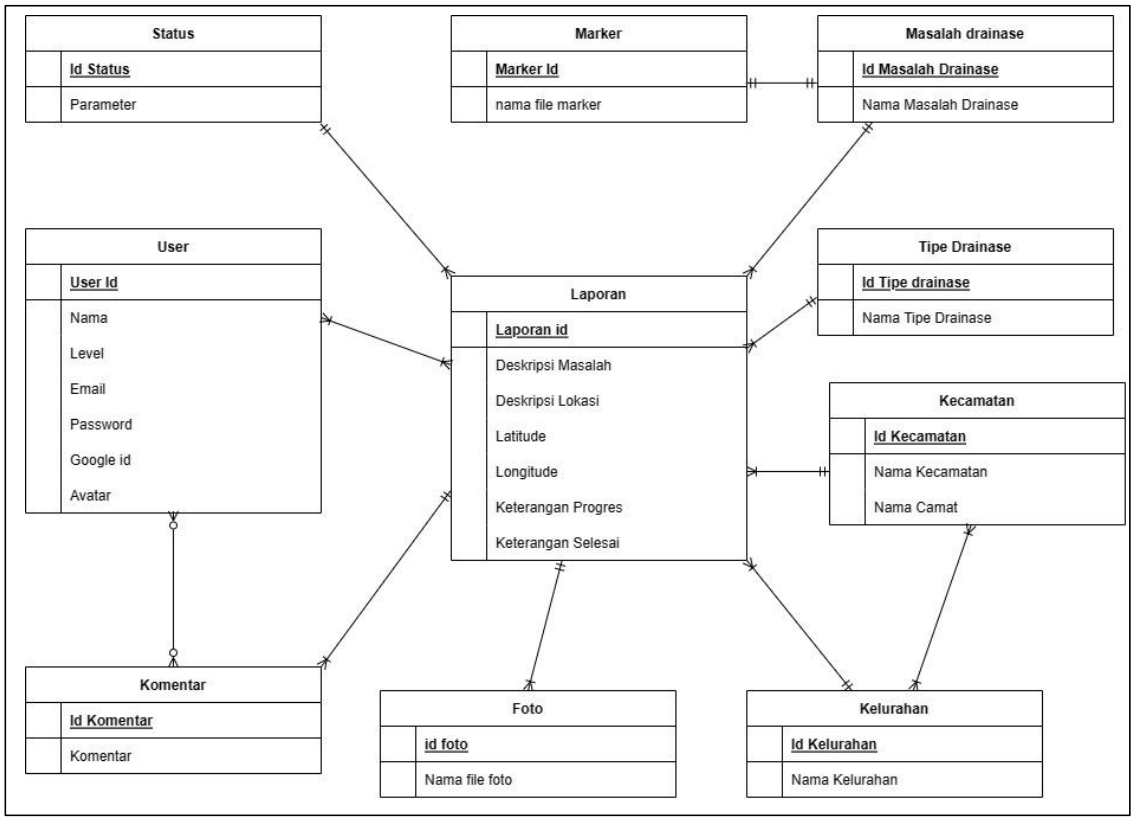
Pada gambar 3.11 merupakan alur pengaturan yang dapat dilakukan oleh admin. Tentu ini hanyalah sebuah opsi tambahan namun penting bagi admin jika ada perubahan-perubahan yang dibutuhkan. Dimulai ketika admin menekan tombol pengaturan, akan ada tampilan dimana terdapat 3 buah panel yang tersedia, yaitu Kecamatan (kiri), Jenis Masalah dan Marker (tengah), Tipe Drainase (kanan). Dari

Kecamatan, disini pengaturannya cukup untuk mengubah nama Camatnya saja, dalam panel Kecamatan akan ada field nama kecamatannya dan dibawahnya ada field nama camatnya, admin memilih salah satu kecamatan yang camatnya ingin diubah, lalu ketikan camat baru dan tekan tombol ok yang ada disampingnya. Kemudian akan tersimpan kedalam database.

Selanjutnya pada sisi jenis permasalahan dan marker terdapat tiga pilihan yaitu mengubah (*edit*), buat baru (*add*) dan hapus. Dimulai dari mengubah jenis permasalahan dan marker, admin menentukan hal yang ingin diubah baik itu keterangan jenis permasalahannya atau warna markernya ataupun kedua-duannya. Setelah admin sudah menentukan , admin memilih jenis permasalahan yang sudah ditentukan, yang keterangannya tertera pada placeholder kolom input, admin menekan kolom input tersebut dan memasukan keterangan jenis permasalahan yang sesuai atau juga admin dapat memilih warna untuk jenis permasalahan tersebut, dibawah kolom input ada pilihan warna-warni marker dan dapat dipilih admin sesuai pilihan. Yang markernya sedang digunakan ada border pinggiran berwarna biru menandakan pilihan yang sedang dipakai, selanjutnya admin mengubah marker sebelumnya , klik warna yang baru diikuti border biru disamping marker. Setelah sudah admin menekan tombol ok kemudian akan tersimpan kedalam database. Pilihan hapus jenis permasalahan , admin pertama menentukan jenis yang ingin dihapus, kemudian cari kolom input yang placeholder berisikan keterangan jenis permasalahan yang ingin dihapus, admin menekan tombol delete, beberapa saat sistem akan men-refresh halaman pengaturan dan akan tersimpan kedalam database.

Pengaturan yang terakhir yaitu tipe drainase sama seperti sebelumnya pada jenis masalah ada terdapat tiga pilihan mengubah (edit), buat baru (add) dan hapus (delete). Jika admin ingin mengubah tipe drainase, admin menentukan lalu kemudian memilih tipe drainase yang ingin diubah, dapat dilihat pada kolom input dengan placeholder yang berisikan keterangan tipe drainase. Klik kolom input yang ingin diubah dan masukan keterangannya, selanjutnya simpan dengan menekan tombol ok. Selanjutnya admin menambahkan tipe drainase , cari kolom input dengan label

diatasnya berupa “Tambahkan tipe drainase”. Admin menekan kolom input tersebut dan memasukan keterangan tipe drainase yang baru kemudian tekan save untuk menyimpannya. Selanjutnya adalah menghapus tipe drainase, admin memilih tipe drainase yang akan dihapus kemudian tekan tombol delete.



Gambar 3. 12 Conseptual ERD

Pada Gambar 3.12 merupakan Conseptual Entitiy Reletionship Diagram dengan menggunakan notasi *Martin notation,* Didalamnya terdapat entitas (e*ntities*), atribut (*attributes*), relasi hubungan (*relationship*). *Conseptual ERD* digunakan untuk memetakan konsep perencaan basis data pada aplikasi ini, *Conceptual ERD* pada perencanaan ini memiliki 10 entitas. Berikut ini 10 entitas yaitu:

1. Entitas *User* : Berisikan data-data pengguna baik itu nama, *email*, *password*, id google dan *avatar*.

2. Entitas Laporan : Berisikan kumpulan laporan drainase bermasalah baik data yang belum diproses dan juga telah diproses. Data yang harus dimasukan oleh *user* berupa deskripsi lokasi, deskripsi masalah, kecamatan, kelurahan, *latitude* dan *longitude*. Dan untuk *admin* berupa jenis masalah drainase, tipe drainase, Status, keterangan sementara diperbaiki, dan keterangan telah selesai diperbaiki.

3. Entitas Masalah Drainase : Berisikan data jenis-jenis masalah drainase yang akan dicocokan pada laporan warga. Dan setiap jenis masalah drainase yang ada akan terelasi dengan masing-masing entitas marker.

4. Entitas Tipe Drainase : Berisikan data tipe drainase yang akan dicocokan pada laporan warga.

5. Entitas Status : Berisi beberapa data yang menjelaskan status laporan berdasarkan tindakan admin, seperti “Belum dibaca”,”Telah dilihat”, ”Kasus akan diproses”, ”Kasus akan diproses”, ”Kasus sementara diproses” & ”Kasus tertangani”.

6. Entitas Kecamatan : Berisikan data-data mengenai kecamatan di Kota Manado, seperti nama nama kecamatan dan nama camat ditiap kecamatan.

7. Entitas Kelurahan : Berisikan data kelurahan di Kota Manado dan tiap kelurahan berelasi dengan kecamatan yang ada di Kota Manado.

8. Entitas Komentar : Berisikan komentar-komentar dari pengguna baik *user* dan *admin.* Tiap laporan akan memiliki kolom komentarnya sendiri.

9. Entitas Marker : Bersikan data marker berupa nama file seperti contoh marker-hitam, marker-merah, marker-hijau dan lain-lain.

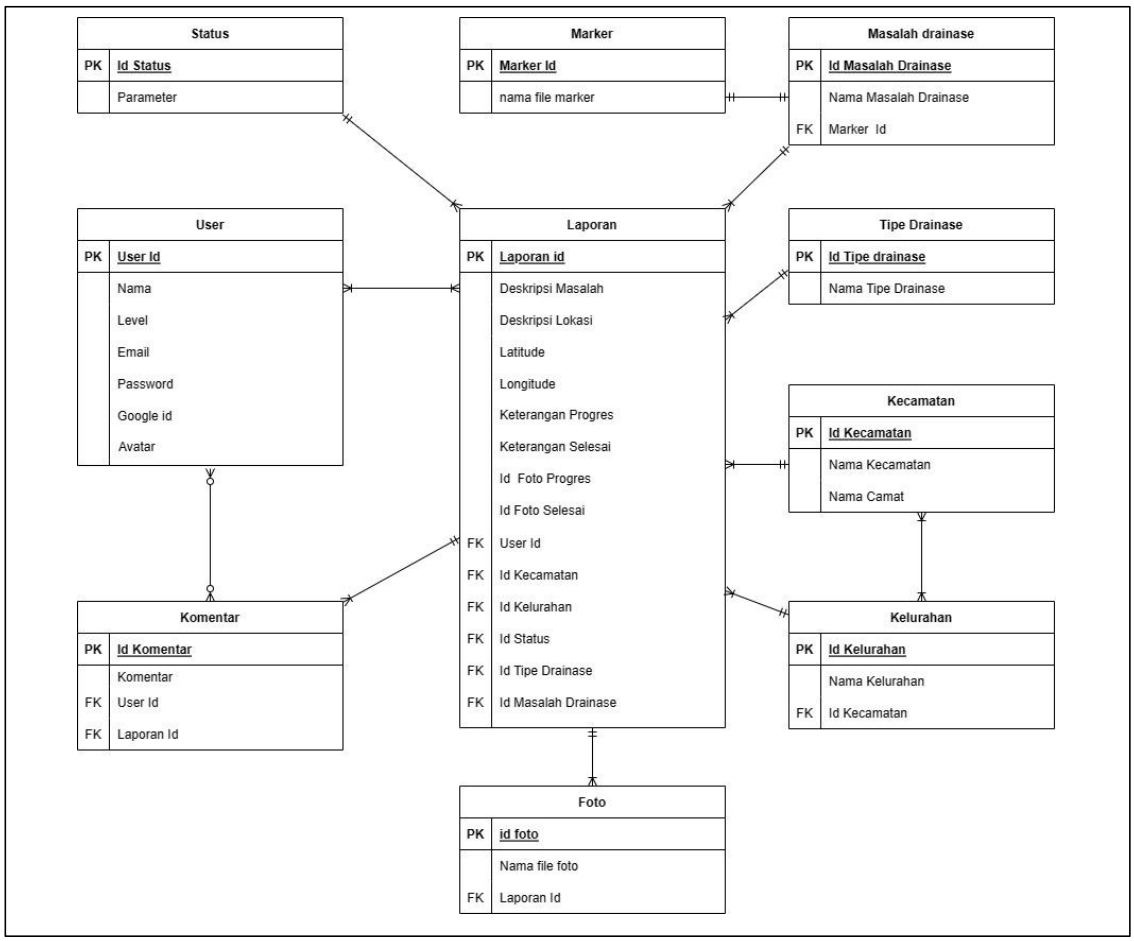
10. Entitas Foto: Berisikan data foto unggahan pengguna dalam melengkapi laporan berupa nama file.

Tabel 3. 13 Penjabaran Entitas beserta atribut dan relasinya.

No.	Entitas	Atribut	Relasi
1.	User	1. Nama 2. Level 3. Email 4. Password	1. User - Laporan (*Many to Many*) Pelapor dapat membuat laporan lebih dari sekali dan admin dapat megolah laporan kapanpun. 2. User - Komentar (*Many to Many*) Pelapor dan Admin dapat membuat komentar lebih dari sekali.
2.	Laporan	1. Deskripsi Masalah Drainase 2. Deskripsi Lokasi 3. Longitude 4. Latitude 5. Kecamatan 6. Kelurahan 7. Ket.Progres 8. Ket.Selesai	1. Laporan - User (*Many To Many*) Sebuah laporan dapat digunakan lebih dari satu pengguna (*user* dan *admin*). 2. Laporan - Jenis Masalah (*One To Many*) Sebuah laporan hanya dapat memilih satu Jenis masalah. 3. Laporan - Tipe drainase (*One To Many*) Sebuah laporan hanya dapat memilih satu Tipe drainase. 4. Laporan - Kecamatan (*One To Many*) Sebuah laporan hanya dapat memilih satu Kecamatan 5. Laporan - Kelurahan (*One To Many*) Sebuah laporan hanya dapat memilih satu kelurahan. 6. Laporan - Status (*One To Many*) Sebuah laporan hanya dapat memilih satu status. 7. Laporan - Ket.Progress

			(*One To Many*) Sebuah laporan hanya dapat memilih satu Keterangan progres. 8. Laporan - Ket.Selesai (*One To Many*) Sebuah laporan hanya dapat memilihi satu keterangan selesai.
3.	Masalah Drainase	1. Masalah	1. Masalah Drainase - Marker (*One to One*) setiap Masalah drainase hanya dapat memilih salah satu dari entitas marker. 2. Masalah Drainase - Laporan (Many to One) Dari sekian Masalah drainase hanya dapat memilih satu Jenis masalah untuk sebuah entitas laporan.
4.	Tipe Drainase	1. Tipe Drainase	1. Tipe Drainase - Laporan *(Many To One*) Dari sekian tipe drainase hanya dapat memilih satu tipe pada sebuah entitas laporan.
5.	Kecamatan	1. Nama Kecamatan 2. Camat	1. Kecamatan - Laporan (*Many To One*) Dari sekian Kecamatan hanya dapat memilih satu kecamatan pada sebuah entitas laporan
6.	Kelurahan	1. Nama Kelurahan	1. Kelurahan - Laporan (*Many To One*) Dari sekian Kelurahan hanya dapat memilih satu Kelurahan pada sebuah laporan 2. Kelurahan - Kecamatan (*Many To Many*) satu atau lebih Kelurahan dapat berelasi dengan berbagai entitas kecamatan.
7.	Status	1. Parameter	1. Status - Laporan (*One To Many*) Dari sekian Status

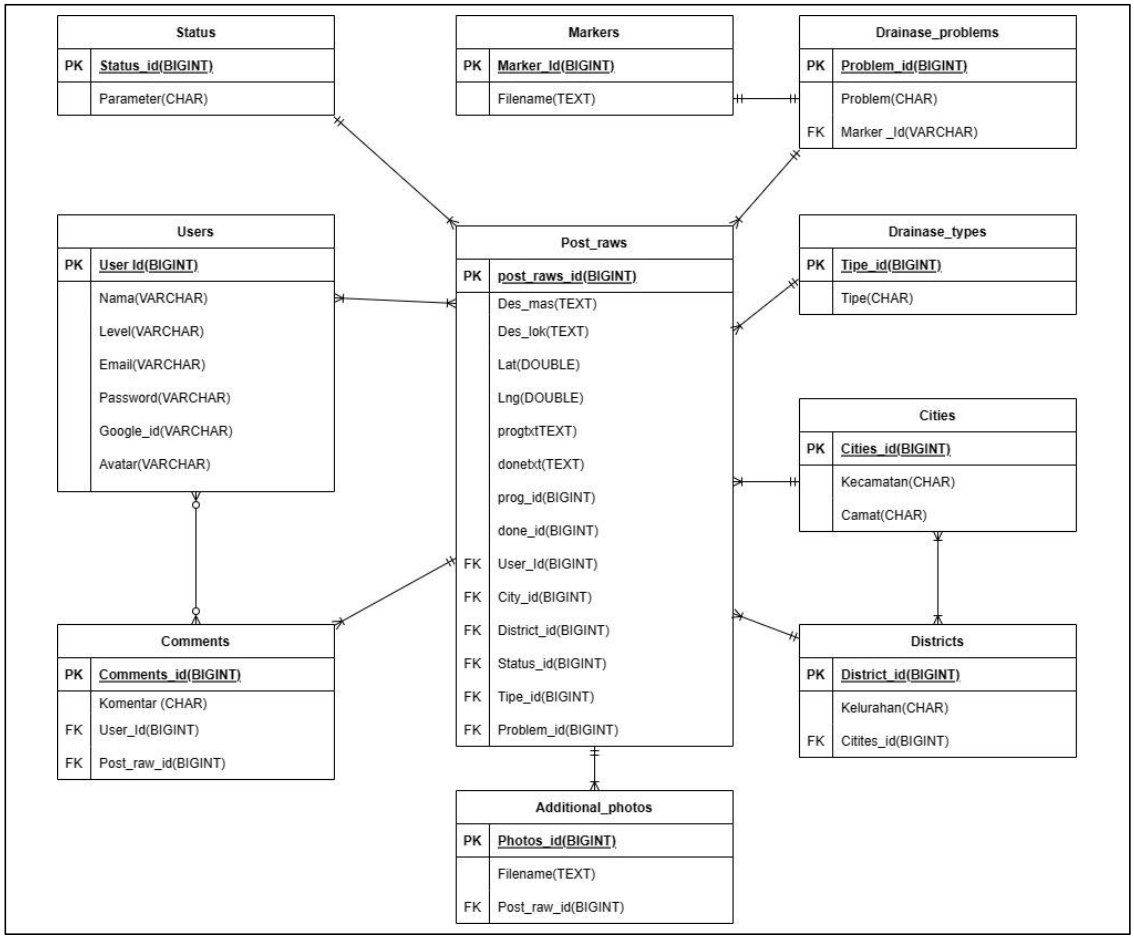
			hanya dapat satu memilhi Parameter pada sebuah entitas laporan.
8.	Komentar	1. Komentar	1. Komentar - Laporan (*Many To One*) Kumpulan komentar hanya dapat tergantung pada satu entitas laporan 2. Komentar - User (*Many To Many*) Satu atau lebih Komentar dapat diciptakan dari berbagai User.
9.	Marker	1. Filename	1. Marker - Masalah Drainase (*One To One*) Satu Marker hanya dapat dimiliki oleh satu entitas Masalah Drainase
10.	Foto	1. Filename	1.Foto - Laporan (*Many To One*) Satu atau lebih Foto hanya dapat dimiliki oleh satu entitas laporan.



Gambar 3. 13 Logical ERD

Pada Gambar 3.11 memperlihatkan struktur data dalam entitas seperti *Primary Key* (PK) dan *Foreign Key* (FK). Pada entitas terdapat *Foreign Key* sebagai struktur data menghubungkan relasi antar entitas yang satu dengan entitas lainnya.

Selain dari itu *Logical ERD* juga berguna sebagai fundamental dalam membuat struktur data dalam bentuk *Physical ERD.* Dengan menggunakan fitur yang terdapat dalam *framework Laravel* seperti *Migrate,* tentunya ada syarat yang harus di ikuti yaitu dalam membuat tabel baru (migrate) harus menggunakan nama tabel dengan ejaan bahasa inggris seperti kata jamak dan singular karena migrate dapat secara otomatis membuat Model (singular) dan tabel(jamak) dapat berelasi jika memenuhi syarat tersebut.



Gambar 3. 14 Physical ERD

Pada Gambar 3.12 merupakan representasi dari *Physical ERD* yang akan dibangun diaplikasi ini, menggunakan struktur data yang lebih mendalam untuk setiap atribut. Berikut ini merupakan pemaparan dari gambar 3.12.

1. Users *:* User_id (*Primary key*) BigINT, nama Varchar, level Varchar, email Varchar, password Varchar, google_id Varchar, avatar Varchar.

2. Post_raws : post_raw_id (*Primary key*) BigINT, des_lok Text, des_mas Text, lat Double, lng Double, progtxt Text, donetxt Text, prog_id BigINT, done_id BigINT city_id (*Foreing key*), district_id (*Foreing key*), status _id (*Foreing key*), tipe_id (*Foreing key*), problem_id (*Foreing key*), user_id (*Foreing key*).

3. Cities : Cities_id (*Primary key*) BigINT, kecamatan Char, camat Char.

4. Districts : District_id (*Primary key)* BigINT, kelurahan Char, Cities_id (*Foreing key*).

5. Drainase_problems : problem_id (*Primary key)* BigINT, problem Char, marker_id Varchar.

6. Drainase_types : tipe_id (*Primary key*) BigINT, tipe Char.

7. Status : Status_id (*Primary key)* BigINT, parameter Char.

8. Comments : Comments_id (*Primary key*) BigINT, komentar Char, User_id (*Foreign key),* post_raw_id (*Foreign key).*

9. Markers : Markers_id (*Primary key*) BigINT, filename Text.

10. Additional_photos : photos_id(*Primary key*), filename Text.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Pembuatan Aplikasi

Setelah sudah melakukan perancangan pada BAB III, pada BAB IV ini akan memperlihatkan hasil pembuatan aplikasi dan juga melakukan testing. Dalam tahap pembuatan aplikasi ini ada terdapat tahap pembuatan prototipe yang masih menggunakan localhost dan tahap pembuatan dengan hosting.

4.1.1 Pembuatan Fungsi-Fungsi Utama

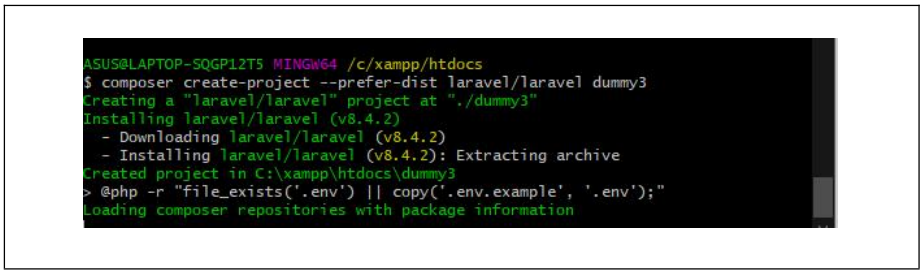
Buat folder baru yang menjadi tempat *project* prototipe. Untuk melakukan direktori dapat Terminal dengan `cd /c/xampp/htdocs`. Disini penulis menggunakan *Terminal console* (Git bash).

Gambar 4. 15 terminal gitbash dengan direktori yang akan menjadi

kemudian mengunduh dan menginstall perangkat lunak *Composer.* Caranya dengan menggunakan sintaks:

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"

php composer-setup.php

php -r "unlink('composer-setup.php');"
```

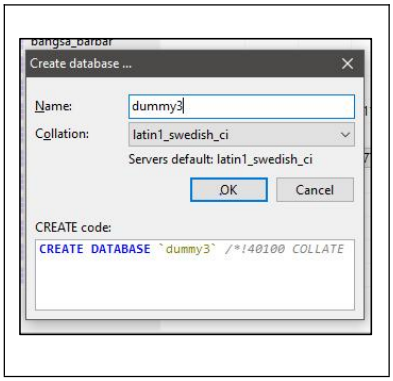
Jika sudah maka kita dapat masukan sintaks seperti berikut. Sintaks ini berdasarkan instruksi dokumentasi dari *framework Laravel.* Dummy3 adalah nama *project* yang kita buat .



Gambar 4. 16 Proses mengunduh dan penginstalasi package ketika membuat project baru. Versi laravel yaitu 8.4.2.

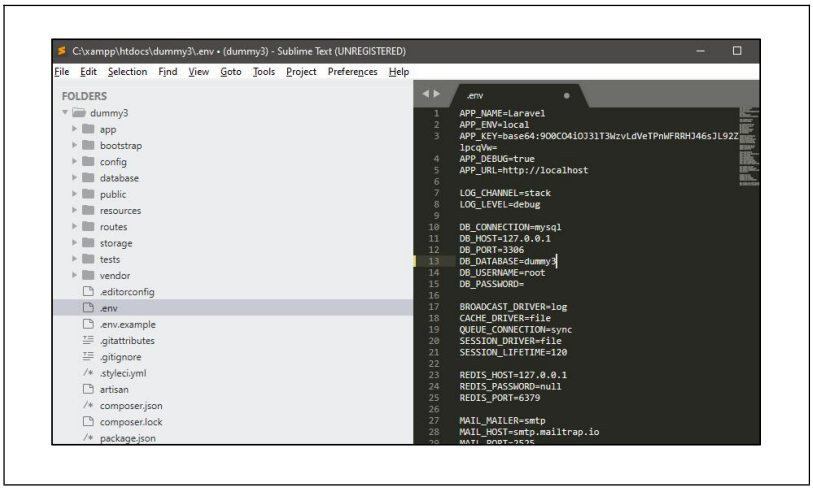
A) Pembuatan Database

Kemudian membuat sebuah *database* yang dapat terhubung dengan *project dummy3* . dan database di *project* ini adalah diberi nama *dummy3* sama dengan nama *project*.



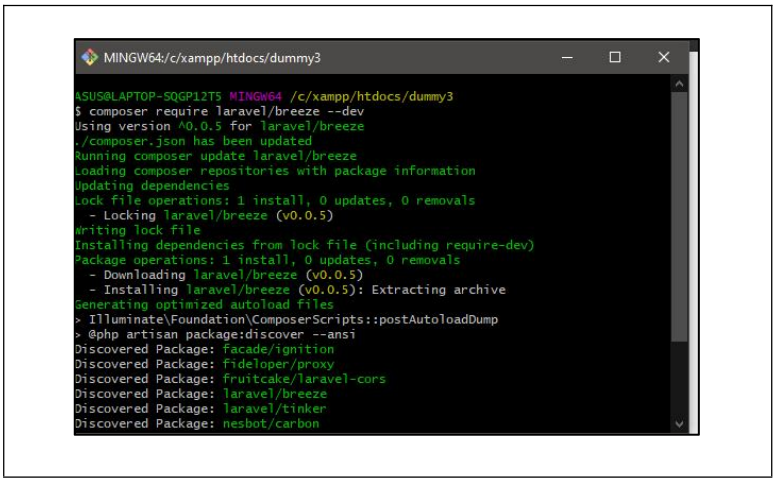
Gambar 4. 17 Pembuatan database dengan sistem database MariaDB. Pembuatan database ini menggunakan softaware HeidiSQL.

Buka aplikasi text editor php, modifikasi *file* .env sesuai dengan nama database yang baru dibuat.



Gambar 4. 18 file .env merupakan yang sangat penting karena berperan sebagai konfigurasi, misalnya database, API Key dan lain-lain .

Penginstalasi dependensi Laravel breeze yang fungsi dan kegunaanya sudah disebutkan pada BAB II yaitu membantu membuat sistem autentifikasi.



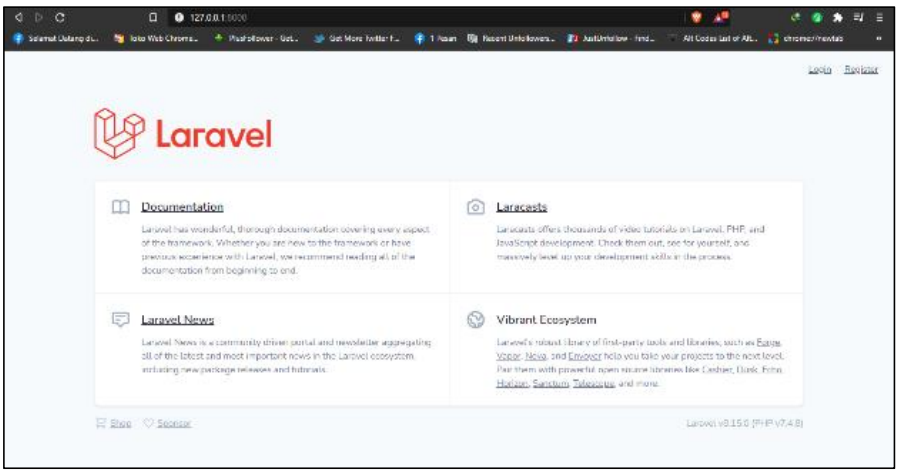
Gambar 4. 19 Sintaks diatas digunakan agar laravel breeze diunduh dan diinstall.

Sebelum kita dapat langsung menggunakan aplikasi web ini, hal yang harus dilakukan adalah melakukan `php artisan migrate` supaya Laravel akan membuatkan database default yang berisi tabel user kedalam database “dummy3”.

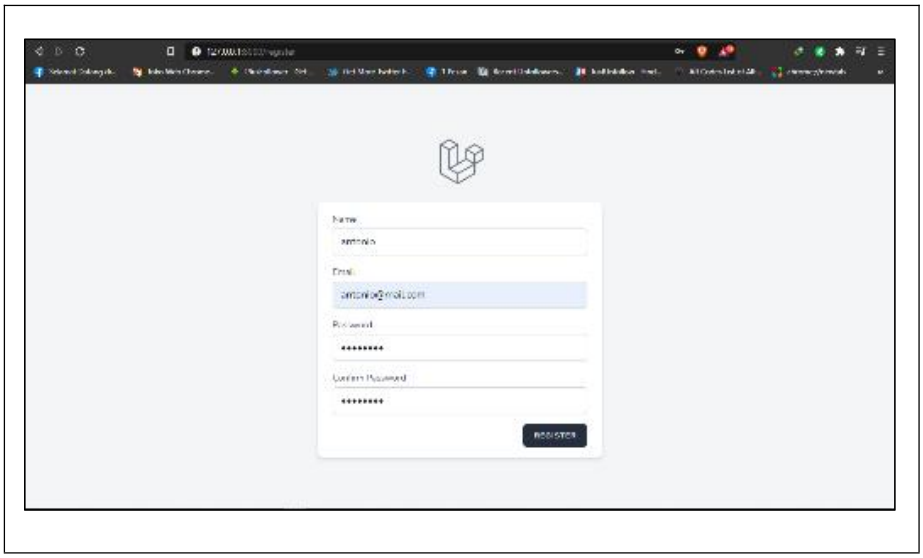
Gambar 4. 20 sisntaks php artisan migrate.

Setelah semua telah terinstal kita dapat langsung melihat hasil tampilan awal project yang telah kita buat, dengan cara memasukan sintaks seperti dibawah.

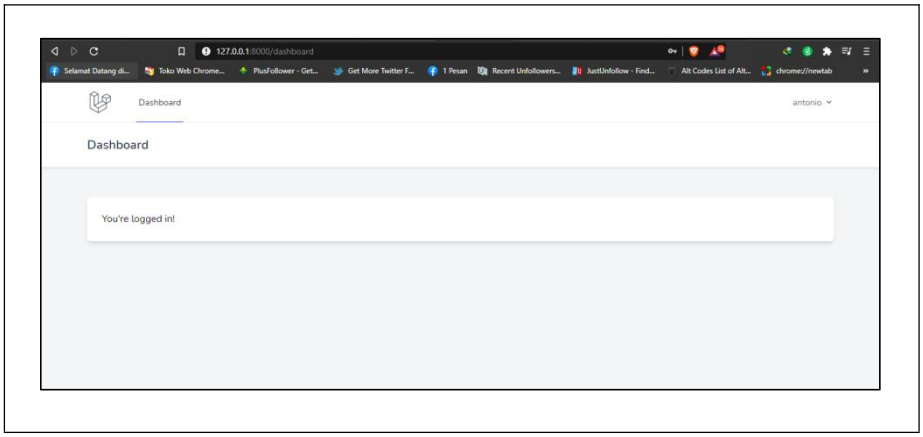
Gambar 4. 21 setelah memasukan sintaks php artisan serve , Laravel akan menghidupkan server local.



Gambar 4. 22 Tampilan default dari project dummy3 dimana sudah dapat langsung melakukan registrasi dan login tanpa perlu membuat controller dan model.



Gambar 4. 23Tampilan default register user.

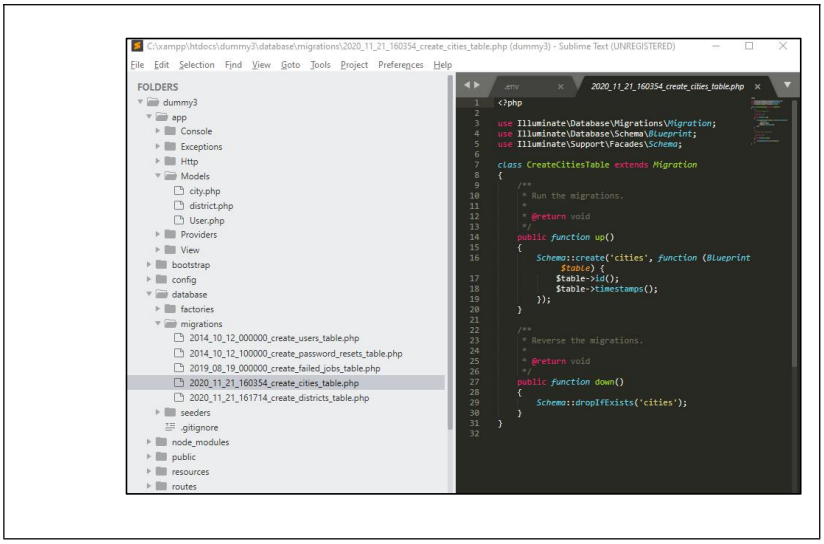


Gambar 4. 24 Tampilan dashboard ketika telah login.

Selanjutnya penulis akan membangun tabel database beserta model Kecamatan dan Kelurahan. Dimana kita akan membuat model tersebut menjadi masing-masing model yang terpisah. Kita akan memberi nama model Kecamatan dengan “City” dan Kelurahan adalah “District”.

Gambar 4. 25 Penambahan -m pada sintaks berguna agar Laravel secara otomatis membuatkan database migrasi yang terhubung langsung dengan model City.

Didalam laravel berlaku sebuah aturan bahwa penamaan model harus bersifat singular (City) dan untuk database bersifat jamak (Cities) hal ini berlaku dalam penggunaan bahasa Inggris saja.

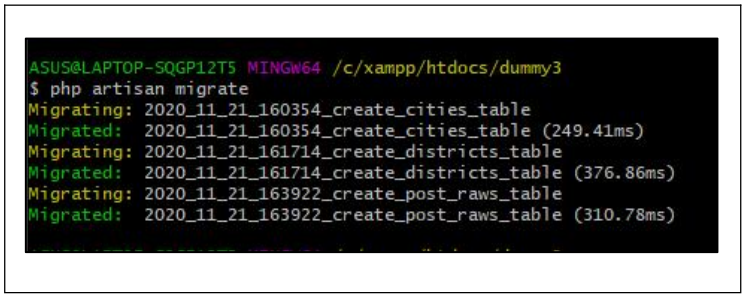


Gambar 4. 26 file migrate dari tabel cities, terlihat struktur data dari tabel cities.

Kemudian kita membuat model dan migrasi Post_raw yang nanti akan menjadi tabel yang berisi laporan yang mentah dibuat oleh *User* dalam hal ini masyarakat.

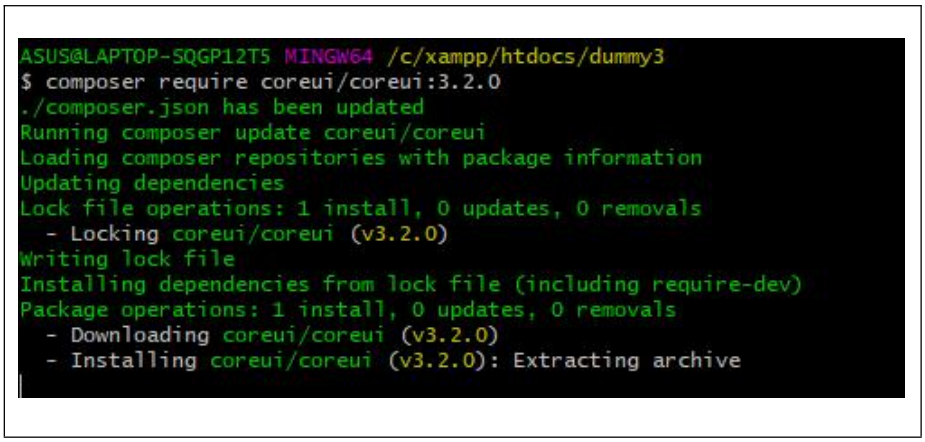
Gambar 4. 27 proses pembuatan model post_raw dan file migrate.

Lakukan php artisan *migrate* agar hasil dari modifikasi agar dapat termigrasi ke dalam *database MariaDB*.



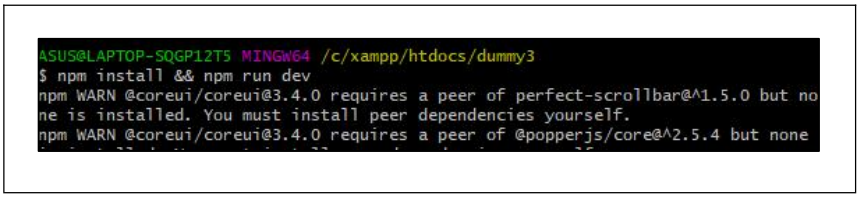
Gambar 4. 28 Tabel cities, district, post_raw telah berhasil di migrasi.

Untuk menghemat waktu dalam perancangan tampilan, penulis menggunakan dependensi *Core UI*.



Gambar 4. 29 Proses unduh dan dan penginstalan Core UI.

Untuk dapat menjalankan tampilan *template* dari *core ui*, terlebih dahulu kita harus menginstal npm.

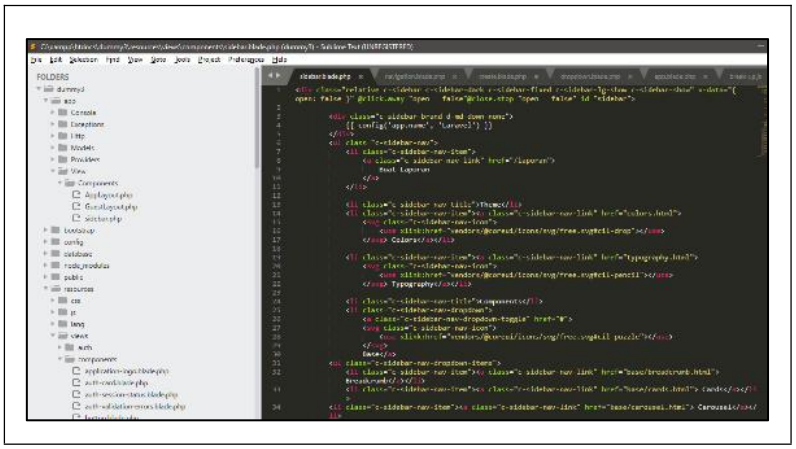


Gambar 4. 30 Setelah nmp berhasil terinstall, npm kemudian akan menjalankan javascript pada project kita.

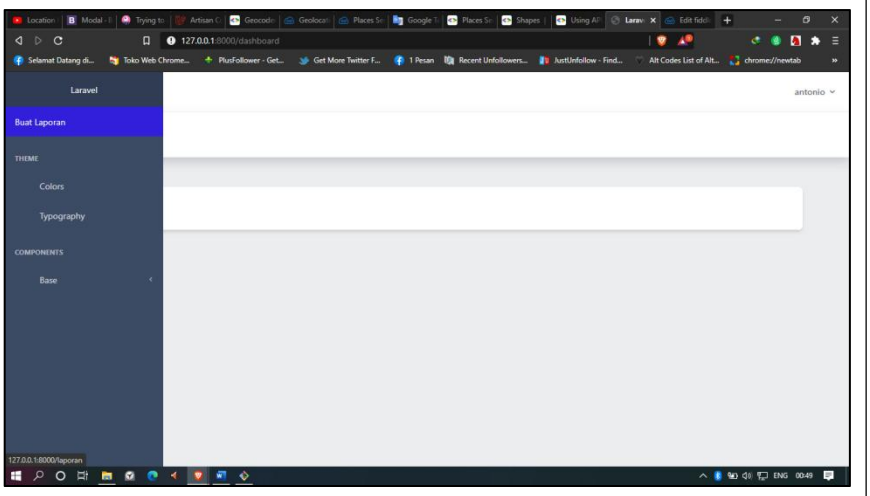
Kemudian penulis membuat sebuah komponen *Sidebar*, yang berguna sebagai menu. hal ini dilakukan agar pengetikan koding *sidebar* tidak tertumpuk disatu halaman. Saat sintaks membuat komponen ini dijalankan nantinya akan ada dua file yang terkompilasi dan diletakan difolder yang berbeda. File yang pertama terdapat di *PATH* `app/view/components/sidebar.php` dimana berfungsi sebagai mengatur fungsi-fungsi yang akan dijalankan pada tampilan komponen tersebut. Sedangkan pada file kedua terdapat di *PATH* `app/public/resources/views/components/sidebar.blade.php` bertindak sebagai tampilan antarmuka dengan pengguna.

Gambar 4. 31 Sintaks untuk membuat component yang file nya terdapat di.

Kemudian di folder ‘views’, dibuat lagi folder dengan nama ‘frontend’. Didalam folder ini terdapat folder-folder yang memuat content dan fungsinya masing-masing.

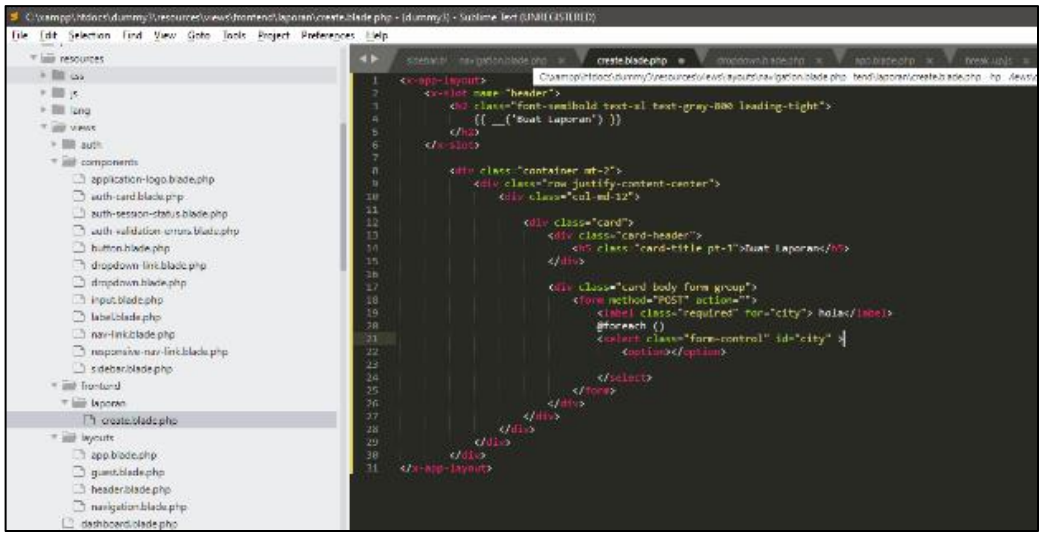


Gambar 4. 32 Gambar diatas merupakan component sidebar.blade.php.



Gambar 4. 33 Gambar diatas merupakan tampilan sidebar sementara waktu. Dan sudah ditambahkan konten “Buat Laporan”.

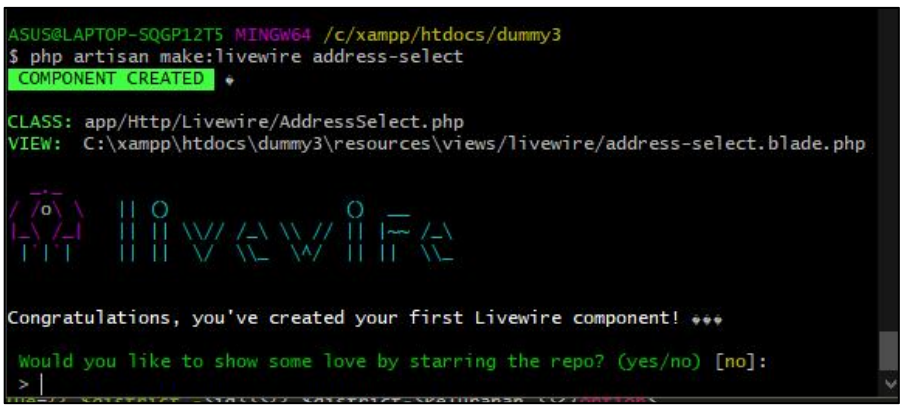
Sintaks dalam komponen *sidebar* merupakan *template* dari *Core UI*. Kemudian kita tinggal hanya memodifikasi baik dari nama hingga *link*-nya.



Gambar 4. 34 Gambar diatas merupakan create.blade.php difolder laporan.

Koding diatas menggunakan component yang telah disediakan oleh laravel `<x-app-layout>`. Dengan menggunakan komponen tersebut nantinya kita tinggal

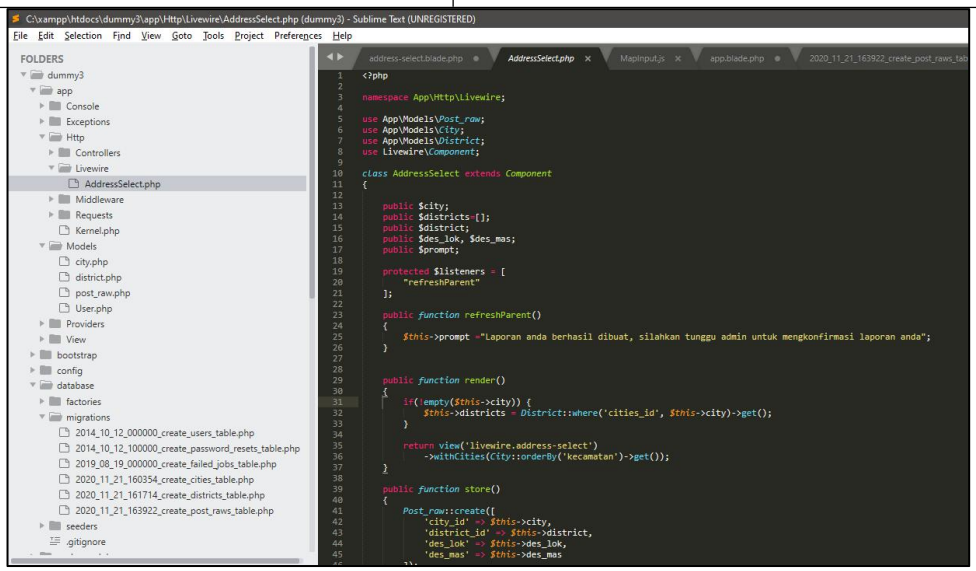
memasukan nama *header* dan *content* yang ingin kita tampilkan (dalam hal ini form laporan) dan akan tertera pada halaman `app.blade.php` (merupakan template utama).



Gambar 4. 35 Sintaks membuat component livewire, disertai terbentuk dua file.

File yang terbentuk merupakan *file Controller* dan View, disini AddressSelect.php merupakan *Controller* dan address-select.blade.php merupakan View.

AddressSelect.php	Livewire



Baris Code :

```
<?php

namespace App\Http\Livewire;
```

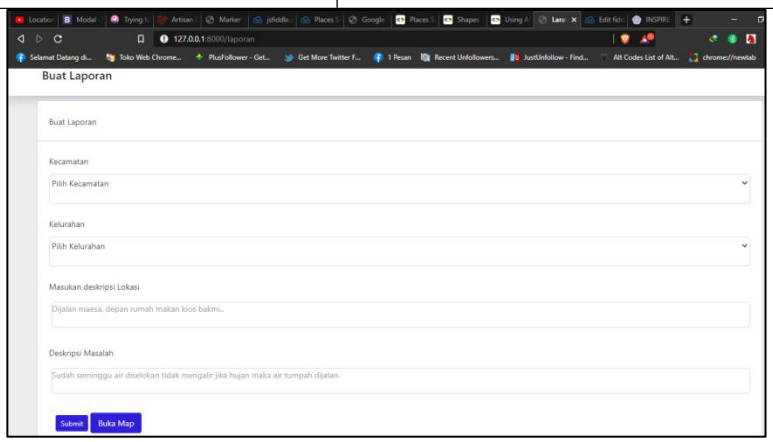
```
use App\Models\post_raw;
use App\Models\City;
use App\Models\district;
use App\Models\AdditionalPhotos;
use Livewire\Component;
use Livewire\WithFileUploads;
use Illuminate\Http\Request;

class AddressSelect extends Component
{
    public $city;
    public $districts = [];
    public $district;
    public $des_lok, $des_mas, $lat, $lng, $latitude, $longitude;
    public $prompt;
    public $modelId;
    public $photos = [];
    use WithFileUploads;
    protected $listeners = [
        "refreshParent",
        "getLatitudeForInput",
        "getLongitudeForInput",
        "getModelId",
    ];

    protected $messages = [
        "city.required" => "Kecamatan tidak boleh kosong.",
        "district.required" => "Kelurahan tidak boleh kosong.",
        "des_lok.required" => "Deskripsi lokasi tidak boleh kosong.",
        "des_mas.required" => "Deskripsi masalah tidak boleh kosong.",
        "lat.required" =>
            "Belum memilih titik lokasi silahkan tekan tombol Buka Map",
        "lng.required" =>
            "Belum memilih titik lokasi silahkan tekan tombol buka Map.",
        "photo" => "Masukan bukti foto",
    ];
    protected function rules()
    {
        return [
            "city" => "required",
            "district" => "required",
            "des_lok" => "required",
            "des_mas" => "required",
            "lat" => "required",
            "lng" => "required",
            "photos.*" => "image|max:1024|required",
        ];
    }

    public function removeImg($index)
    {
        array_splice($this->photos, $index);
    }

    public function refreshParent()
    {
        $this->prompt =
            "Laporan anda berhasil dibuat, silahkan tunggu admin untuk
mengkonfirmasi laporan anda";
```

```
    }

    public function getLatitudeForInput($value)
    {
        if (!is_null($value)) {
            $this->lat = $value;
        }
    }

    public function getLongitudeForInput($value)
    {
        if (!is_null($value)) {
            $this->lng = $value;
        }
    }

    public function render()
    {
        if (!empty($this->city)) {
            $this->districts = District::where("cities_id", $this->city)-
>get();
        }

        return view("livewire.address-select")->withCities(
            City::orderBy("kecamatan")->get()
        );
    }

    public function getModelId($modelId)
    {
        $this->modelId = $modelId;
        $model = Post_raw::find($this->modelId);
    }

    public function store()
    {
        $this->validate();

        if ($this->modelId) {
            Post_raw::find($this->modelId);
            $postInstanceId = $this->modelId;
        } else {
            $postInstance = Post_raw::create([
                "city_id" => $this->city,
                "district_id" => $this->district,
                "des_lok" => $this->des_lok,
                "des_mas" => $this->des_mas,
                "lat" => $this->lat,
                "lng" => $this->lng,
                "user_id" => auth()->user()->id,
                "status_id" => 1,
                "problem_id" => 1,
                "tipe_id" => 1,
            ]);
            $postInstanceId = $postInstance->id;
        }

        foreach ($this->photos as $photo) {
            $photo->store("additional_photos", "public");
```

```
            AdditionalPhotos::create([
                "post_raw_id" => $postInstanceId,
                "filename" => $photo->hashName(),
            ]);
        }

        $this->emit("refreshParent");
    }
}
```

Gambar 4. 36 Isi dari AddressSelect.php.

AddressSelect.blade.php	Views



Baris Code :

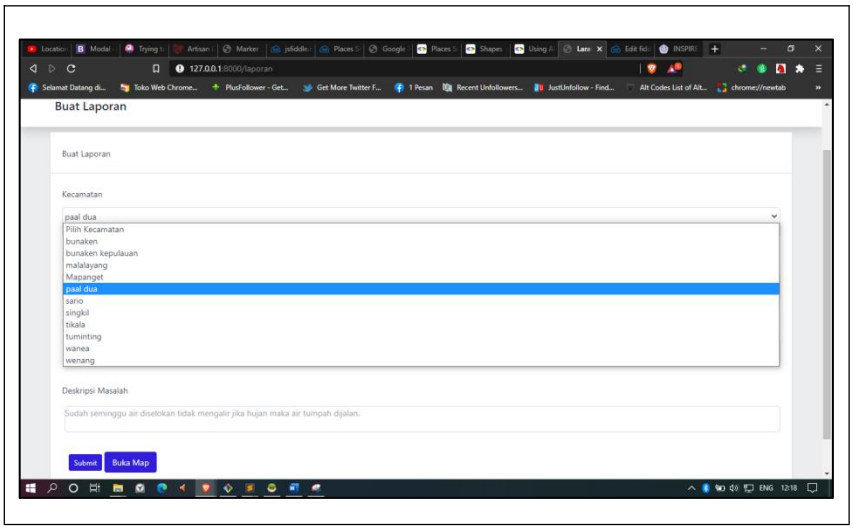
```
      <div >
          <button type="button" class="btn btn-primary" data-toggle="modal"
data-target="#exampleModal">
            Buka Map
          </button>
                  @error('lat') <span class="error">{{ $message }}</span>
@enderror
          <form wire:submit.prevent="store">

              <div class="form-group " >

                <div class="relative mt-1 mb-4" >
                          <label for="city"> Kecamatan </label>
                                <div class="mt-1 mb-4 border rounded-md">
                    <select
                    name="city"
                    wire:model="city"
                    class="mt-1 mb-4 w-full outline-primary" >
                          <option value=''>Pilih Kecamatan</option>
                          @foreach($cities as $city)
                                <option value={{ $city ->id}}>{{ $city-
>kecamatan }}</option>
                          @endforeach
                    </select>
```

```
                        @error('city') <span
class="error">{{ $message }}</span> @enderror
                                </div>
                </div>

                <div class="mt-1 mb-4">
                  <label for="district"> Kelurahan </label>
                    <div class="mt-1 mb-4 border rounded-md">
                    <select
                    name="district"
                    wire:model="district"
                    class="mt-1 mb-4 w-full outline-primary" >
                           <option value=''>Pilih Kelurahan</option>
                           @foreach($districts as $district)
                                  <option value={{ $district -
>id}}>{{ $district->kelurahan }}</option>
                           @endforeach
                    </select>
                        @error('district') <span
class="error">{{ $message }}</span> @enderror
                   </div>
                </div>

              </div>

          <div class="form-group">
           <label for="des_lok"> Masukan deskripsi Lokasi </label>
              <div class="mt-1 mb-4 pl-1 pt-1 pb-1 " >
              <textarea class="resize-none border rounded-md focus:outline-
none w-full" placeholder="Dijalan maesa, depan rumah makan kios bakmi.."
name="des_lok"  wire:model="des_lok" wire:ignore></textarea>
              @error('des_lok') <span class="error">{{ $message }}</span>
@enderror
              </div>
           </div>

           <div class="form-group">
           <label for="des_mas">Deskripsi Masalah</label>
              <div class="mt-1 mb-4 pl-1 pt-1 pb-1 " >
              <textarea class="resize-none border rounded-md focus:outline-
none w-full" placeholder="Sudah seminggu air diselokan tidak mengalir jika
hujan maka air tumpah dijalan. " wire:model="des_mas"></textarea>
              @error('des_mas') <span class="error">{{ $message }}</span>
@enderror
              </div>

          </div>

              <label for="photo_input"> Masukan Gambar </label>
              <input type="file" wire:model="photos" multiple>
          <div class="flex">
          @if ($photos)
              Photo Preview:
              @foreach($photos as $photo)
              <i wire:click.prevent="removeImg({{$loop->index}})">X</i>
              <img width="62" height="22" class="responsive p-2"
src="{{ $photo->temporaryUrl() }}">

              @endforeach
          </div>
```

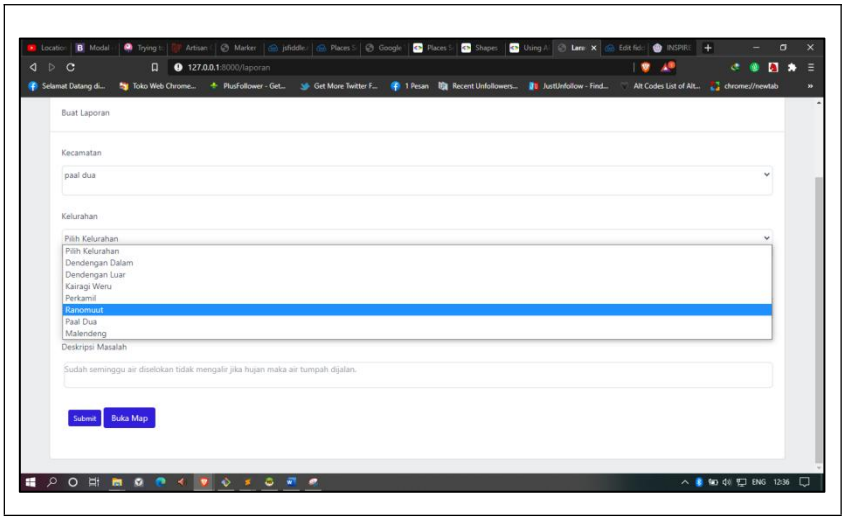
```
          @endif


          @error('photos.*') <span class="error">{{ $message }}</span>
@enderror

          <div class="form-group" >
              <label for="latitudehide">latitude</label>
              <input id="latitudehide" name="latitudehide" wire:model="lat"
wire:model="lat"  >

              <label for="longitudehide">longitude</label>
              <input id="longitudehide" name="longitudehide"
wire:model="lng" wire:model="lng" >
          </div>

           <label for="status_id"></label>
             <input type="text" name="status_id" class="">

          <button type="submit" class="btn btn-sm btn-
primary" >Submit</button>

          </form>
          {{$prompt}}
          <!-- Button trigger modal -->

      <!-- Modal Buka map -->
          <div class="modal fade" id="exampleModal" tabindex="-1" aria-
labelledby="exampleModalLabel" aria-hidden="true">
            <div class="modal-dialog">
              <div class="modal-content">
                <div class="modal-header">
                  <h5 class="modal-title" id="exampleModalLabel">Modal
title</h5>
                  <button type="button" class="close" data-dismiss="modal"
aria-label="Close">
                    <span aria-hidden="true">&times;</span>
                  </button>
                </div>
                <div class="modal-body">

              <input id="pac-input" type="text" placeholder="cari lokasi">
              <div wire:ignore id="map" class="mb-2" style=" width: 500px;
height: 400px; float: left;"></div>


              <input wire:ignore id="infolat" value="as" type="text"
name="infolat" class="hidden">
              <input wire:ignore id="infolng" value="rs" type="text"
name="infolng" class="hidden">
              <div id="infoPanel" class="hidden">
              <b>Marker status:</b>
              <div id="markerStatus"><i>Click and drag the marker.</i></div>
              <b>Current position:</b>
                </div>
                <div class="modal-footer">
                  <button type="button" class="btn btn-secondary" data-
dismiss="modal">Close</button>
                  <button onclick="myFunction()" id="save" type="button"
class="btn btn-primary" data-dismiss="modal">Save changes</button>
```

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

      </div>

      @section('scripts')
      <script async defer
src="https://maps.googleapis.com/maps/api/js?key={{config('services.google.k
ey')}}&libraries=places&callback=initialize"
        type="text/javascript"></script>
       <script src="{{ asset('js/mapInput.js') }}" defer></script>

      <script >
      // supaya menu dalam searchbox tidak membelakangi modal
      var pacContainerInitialized = false;
      $("#pac-input").keypress(function() {
        if (!pacContainerInitialized) {
          $(".pac-container").css("z-index", "9999");
          pacContainerInitialized = true;
        }
      });

      // ambil data latitude dan longitude dari input yang dimodal
          function myFunction() {
            var lat = document.getElementById("infolat").value;
            var lng = document.getElementById("infolng").value;

          // ambil value lewat DOM, tidak cocok buat livewire
            // x.setAttribute("value",lat );

          // agar Javascript dapat berkomunikasi dng Livewire
            Livewire.emit('getLatitudeForInput',lat);
            Livewire.emit('getLongitudeForInput',lng);

          //memasukan value pada text input, tidak cocok dgn livewire
            // document.getElementById("infolat").value ;
            // document.getElementById("latutudehide").value =
document.getElementById("infolat").value ;
            // document.getElementById("infolng").value ;
            // document.getElementById("longitudehide").value =
document.getElementById("infolng").value ;
          }
       // Enter button disabling
          $(document).keypress(
               function(event){
                  if (event.which == '13') {
               event.preventDefault();
          }
      });

      </script>
      @endsection
```

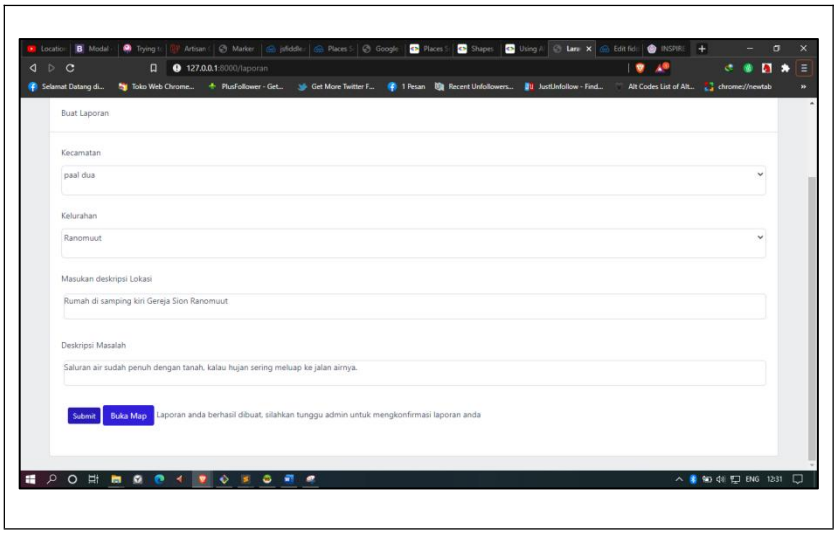
Gambar 4. 37 Isi dari AddressSelect.blade.php.



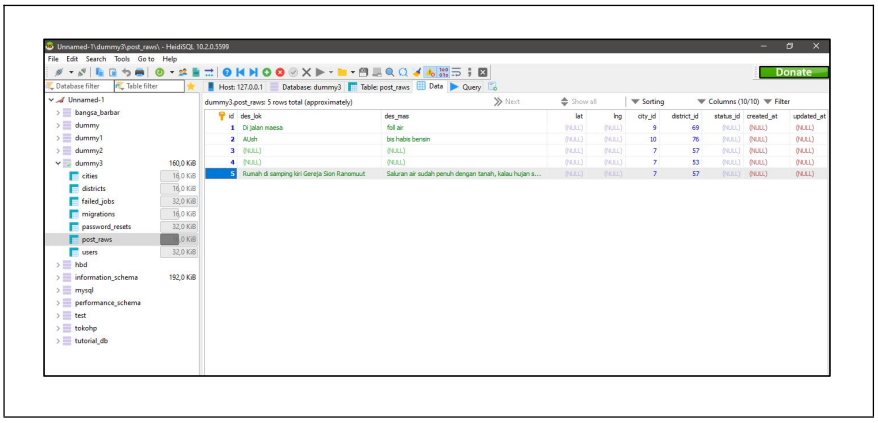
Gambar 4. 38 Percobaan memilih kecamatan,dipilih kecamatan paal dua.



Gambar 4. 39 Percobaan memilih kecamatan, data kelurahan berhasil ditampilkan



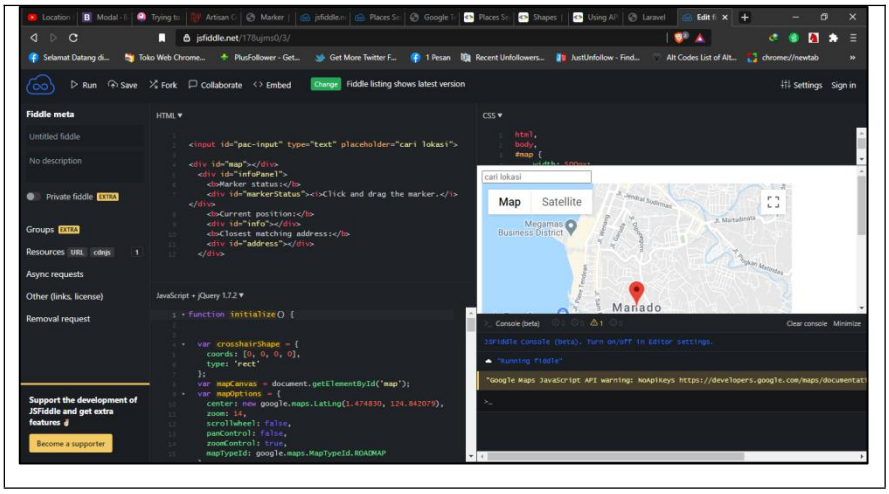
Gambar 4. 40 Percobaan melakukan stored data, semua data disi baik deskripsi lokasi dan deskripsi masalah kemudian tekan submit.



Gambar 4. 41 Data berhasil tersimpan didatabase

B) Pembuatan peta *google maps*

Buka aplikasi web Jsfiddle.



Gambar 4. 42 Tampilan dari jsfiddel, dapat terlihat digambar terdapat 4 bagian utama , HTML (kiri atas) Javascript + JQuery (kiri bawah), CSS (kanan atas), dan Output beserta Console (kanan bawah).

```
HTML ▼

<input id="pac-input" type="text" placeholder="cari lokasi">

<div id="map"></div>
  <div id="infoPanel">
    <b>Marker status:</b>
    <div id="markerStatus"><i>Click and drag the marker.</i>
</div>
    <b>Current position:</b>
    <div id="info"></div>
    <b>Closest matching address:</b>
    <div id="address"></div>
  </div>
```

Gambar 4. 43 Coding HTML pada project Map di Jsfiddel.

```
CSS ▼

html,
body,
#map {
    width: 500px;
    height: 400px;
    float: left;
}

  #infoPanel {
    float: left;
    margin-left: 10px;
  }
  #infoPanel div {
    margin-bottom: 5px;
  }
```

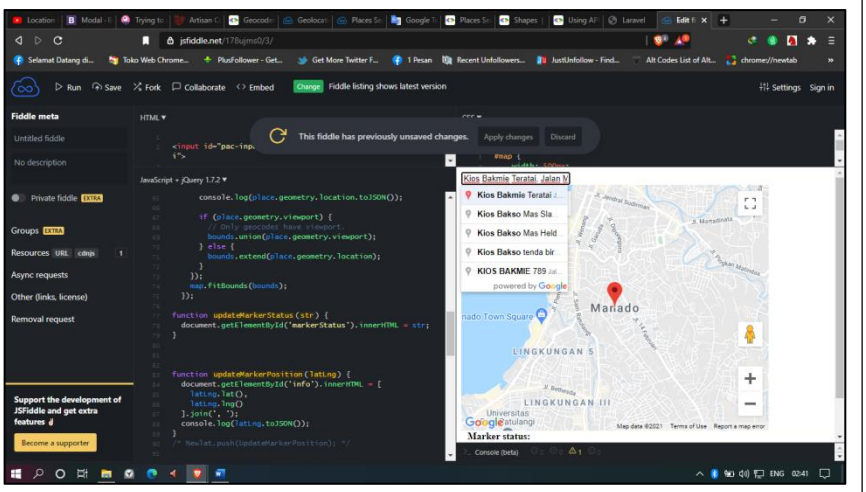
Gambar 4. 44 Coding CSS pada project Map di Jsfiddel.

"#Map" mendeskripsikan variabel peta. ukuran yang terterah masih akan berubah menyesuaikan dengan aplikasi web kita nanti. "#infoPanel" dan "#infoPanel". kemudian "div" mendeskripsikan status "marker", "longitude" dan "latitude". Status "marker" merupakan indikator saat marker melakukan sesuatu. misalnya user memencet marker dan melakukan aksi *drag* (tarik), maka statusnya adalah "marker dragged", begitu halnya jika *marker* tersebut dilepas maka status "dragend".

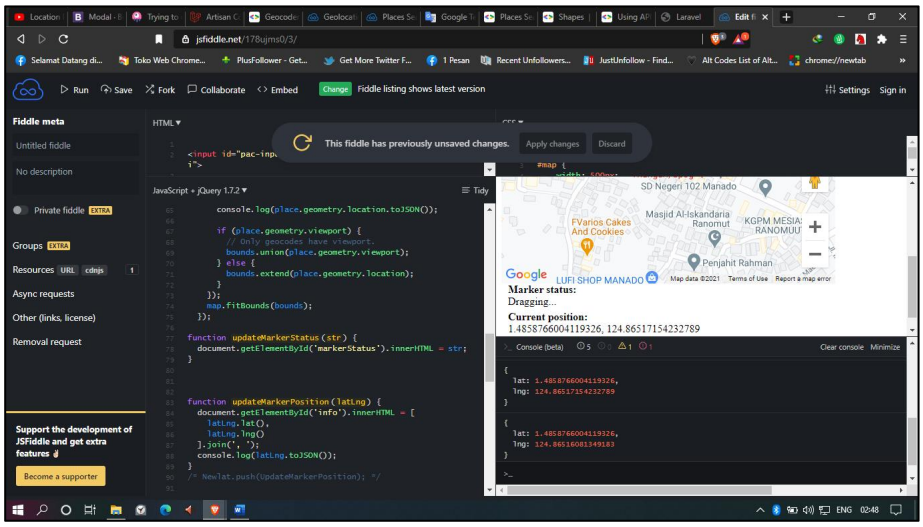
Gambar 4. 45 Coding Javascript pada project Map di Jsfiddel. Kodingan digambar tidak ditampilkan secara menyeluruh, nanti variabel coding akan dijelaskan halaman selanjutnya.

Sebelumnya untuk mengetahui sumber variabel-variabel pada google maps dapat dilihat pada *Google documentation*. Hal pertama melakukan *pharsing* antara variabel map di javascript dan di HTML merupakan tugas dari variabel mapCanvas. MapOptions mengatur tampilan posisi awal, *zoom* dan lain-lain. variabel marker dikonfigurasi, tambahkan draggable dengan nilai *true* agar bisa ditarik-tarik.

Kemudian ada variabel searchBox yang berfungsi sebagai kolom pencarian lokasi. Untuk dapat menampilkan *longitude* dan *latitude* terlebih dahulu tambahkan *listener* pada *marker*, sehingga jika *marker* melakukan aksi akan dapat terpantau. Kemudian untuk mendapatkan data longitude dan latitude pada marker tambahkan variabel updateMarkerPosition (LatLng).



Gambar 4. 46 Percobaan pencarian alamat menggunakan Search box. Alamat yang dimasukan adalah Kios bakmi.

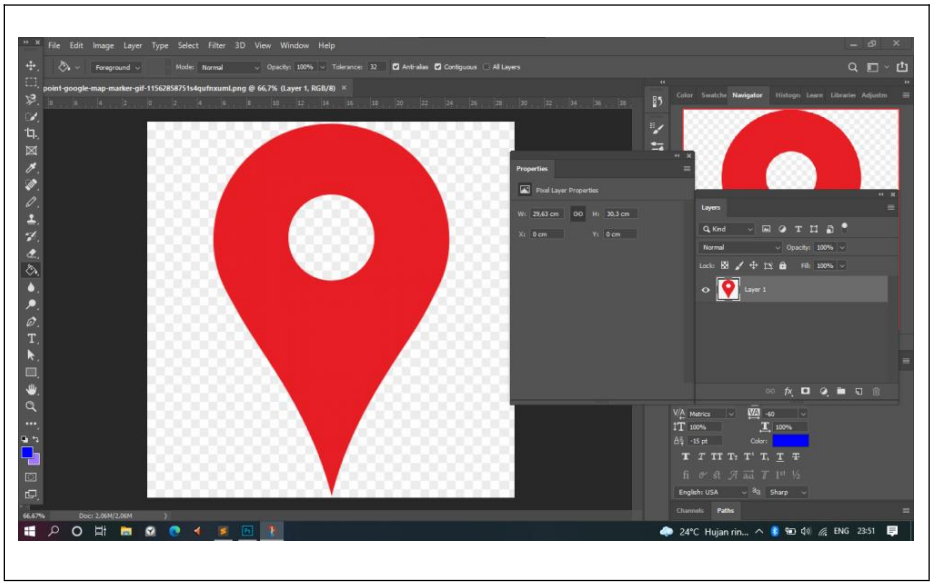


Gambar 4. 47 data longitude dan latitude berhasil didapatkan pada bagian console dan halaman html.

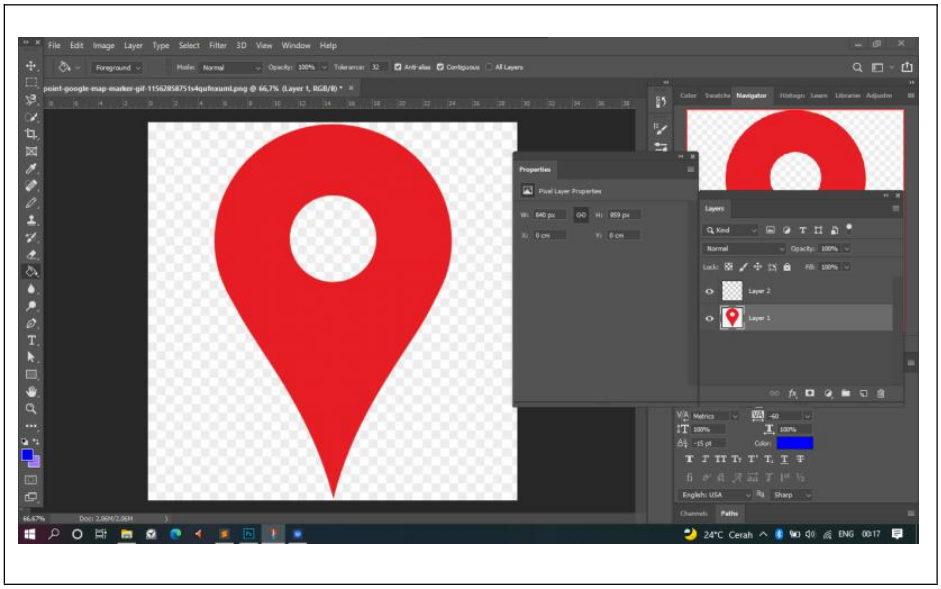
C) Pembuatan *logo marker* dengan Photoshop

Hal pertama untuk memdofikasi gambar dibutuhkan sampel *marker*. Sampel ini diambil bentukan dari *marker* kemudian diubah menjadi berbagai macam warna.

Sampel merupakan file berekstensi png. Untuk memasukan ke dalam *Adobe Photoshop*, click dan tahan sampel gambar lalu tarik dan lepaskan kedalam *adobe photoshop* yang telah dibuka.



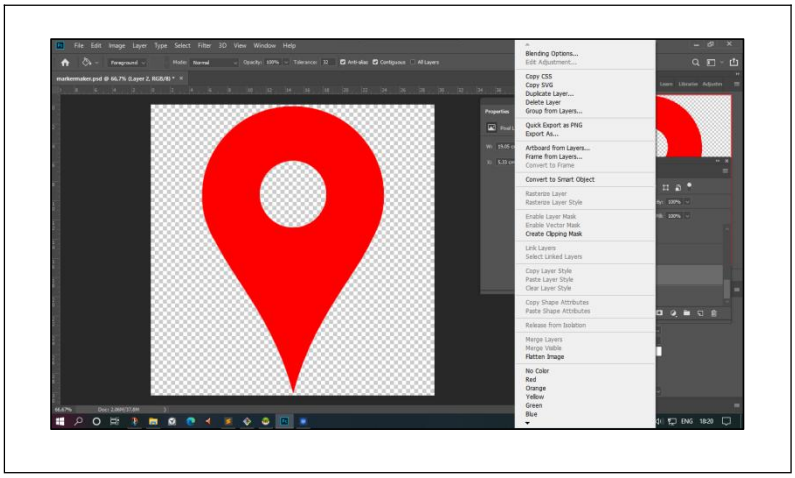
Gambar 4. 48 Tampilan Adobe Photoshop dan sampel marker berekstensi png.



Gambar 4. 49 Menambahkan layer baru.

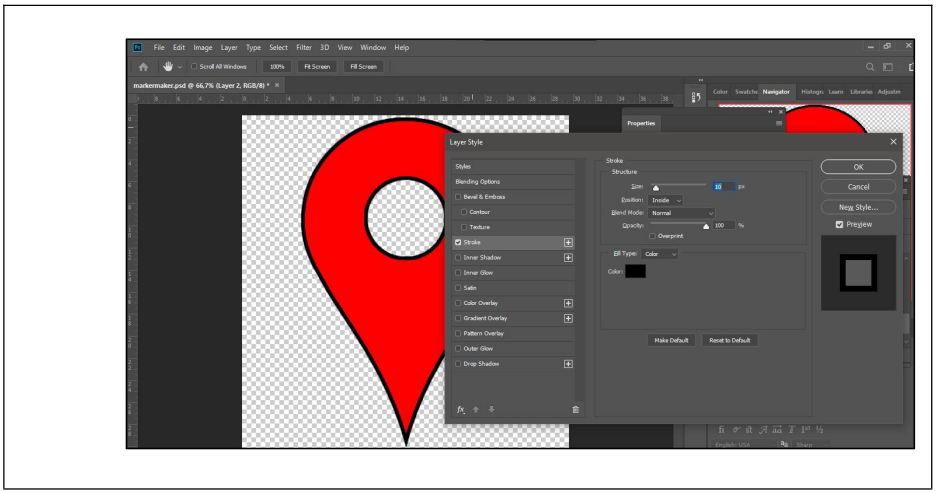
Selanjutnya kita menggunakan *tool magicwand*, dengan fitur *magicwand* dengan mudah menyeleksi suatu pola gambar secara utuh. Dengan cara setelah

mengaktifkan fitur *magicwand* arahkan *mouse* ke gambar dalam kasus ini adalah *marker*, keefektifan *magicwand* tergantung pada pola mudah dan warna jika antara gambar dan *background* memiliki warna yang kontras akan semakin bagus *magicwand* akan menyeleksi gambar.

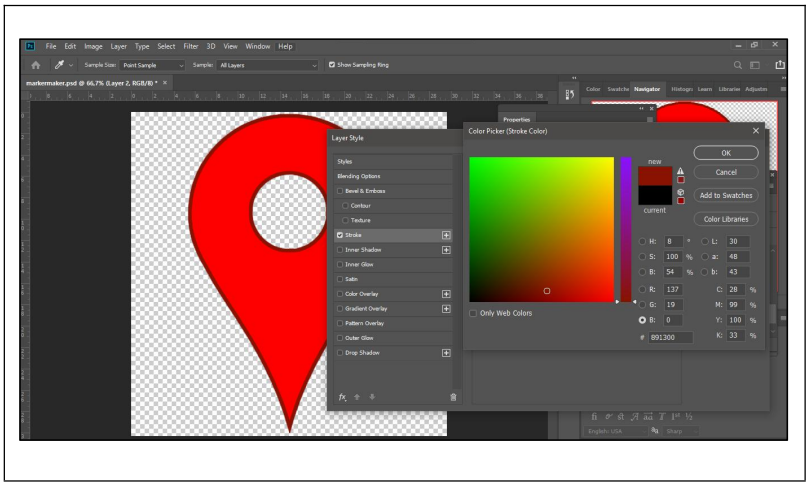


Gambar 4. 50 Saat melakukan klik kanan pada layer untuk menampilkan Properties, kemudian pilih Blending Options.

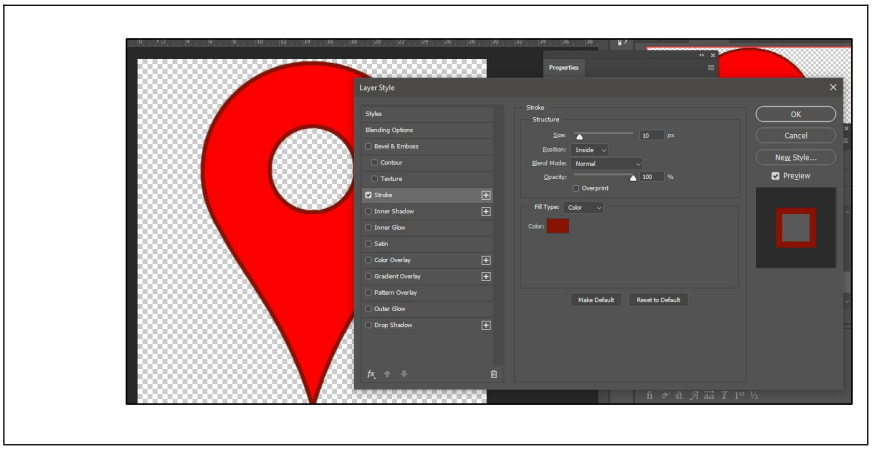
Langkah ini adalah cara untuk membuat storke pada gambar. Setelah pilih blended options akan muncul *Layer Style* pada pilihan dibagian kiri tekan menu *Stroke* disini dapat memilih ketebalan sebuah *stroke* disini *developer* memilih *size* 10 *px.*



Gambar 4. 51 Tampilan Menu Layer Style untuk mengatur stroke.

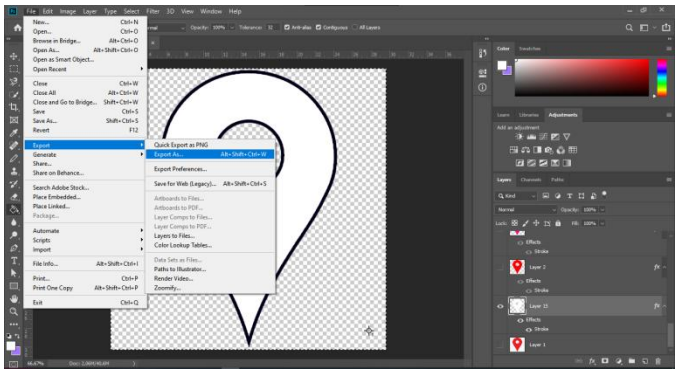


Gambar 4. 52 Pengaturan warna pada stroke.



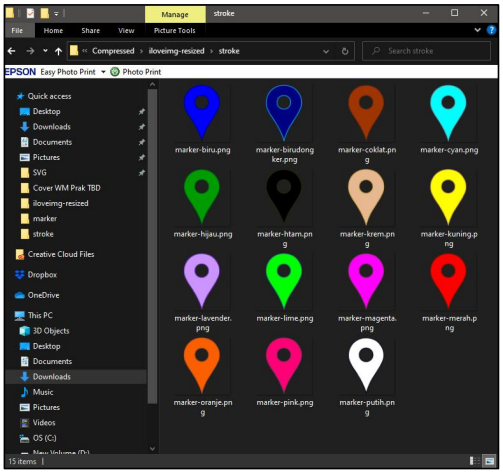
Gambar 4. 53 Setelah sudah sesuai pastikan menekan OK

Simpan dengan cara *File -> Export -> Export A*s.



Gambar 4. 54 Proses menyimpan nanti akan tersimpan dalam bentuk extensi PNG.

Dalam aplikasi web ini *marker* yang dibuat berjumlah 15 marker. Jadi pada *photoshop*, penulis melakukan 15 buah marker dengan warna yang berbeda-beda.

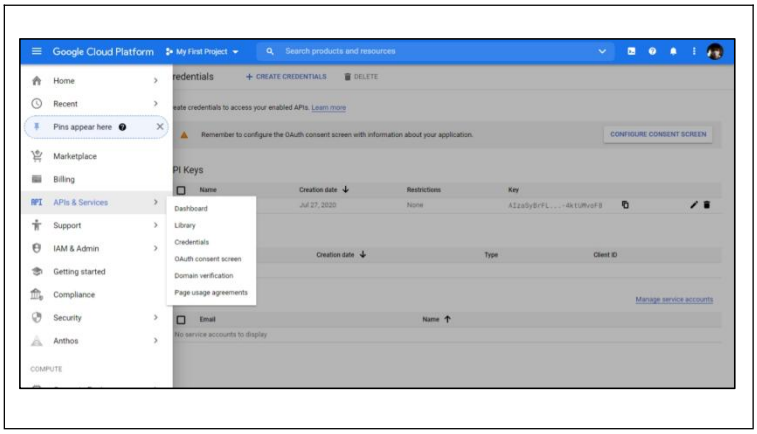


Gambar 4. 55 Hasil pembuatan marker dengan photoshop tiap marker berukuran width 26 px height 36 px.

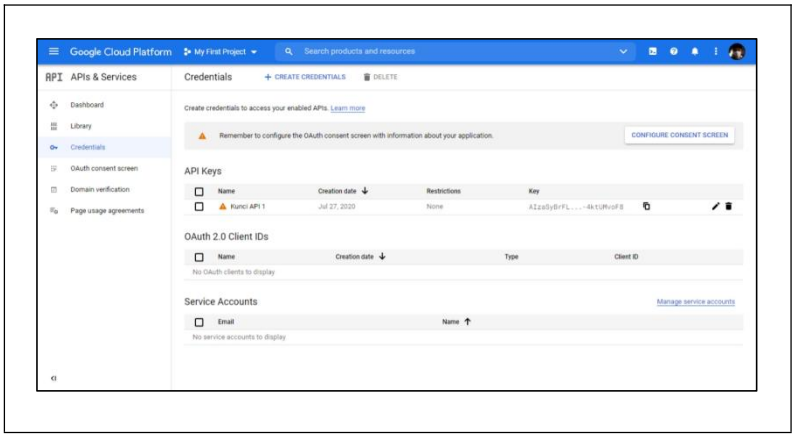
D) Pembuatan kredensial *OAuth Google* pada aplikasi

Untuk mendapatkan data diri pembuat laporan, sistem ini menerapkan *Sign-in* dengan Google API. Sehingga nantinya user hanya perlu menggunakan *Google Account* mereka agar dapat *login* dan bisa membuat laporan selain mudah dan cepat Sign-in dengan *Google API* tentu aman karena aplikasi nantinya hanya mendapat *key token* dan informasi yang terbatas dimana berasal dari Google account (seperti Id, nama, email dan avatar dari Google account). berikut adalah langka-langka membuat Login dengan Google API.

Buka *sidebar google cloud platform* pilih *APIs & Services,* pilih *Credentials*.

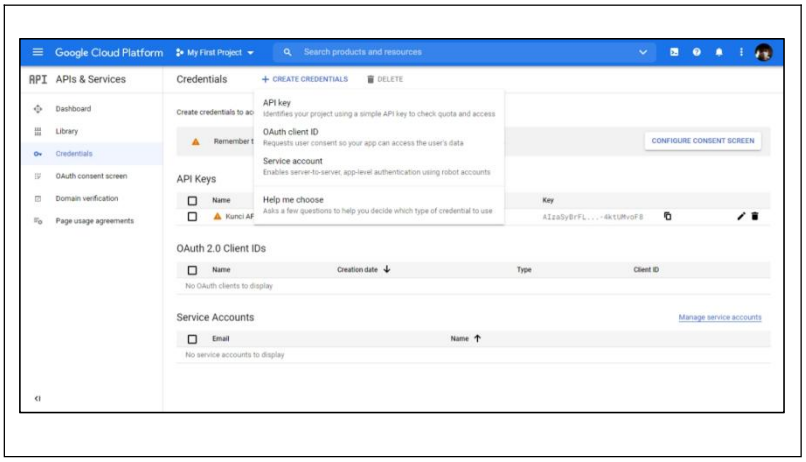


Gambar 4. 56 Langkah awal pembuatan credentials pada halaman console google cloud.



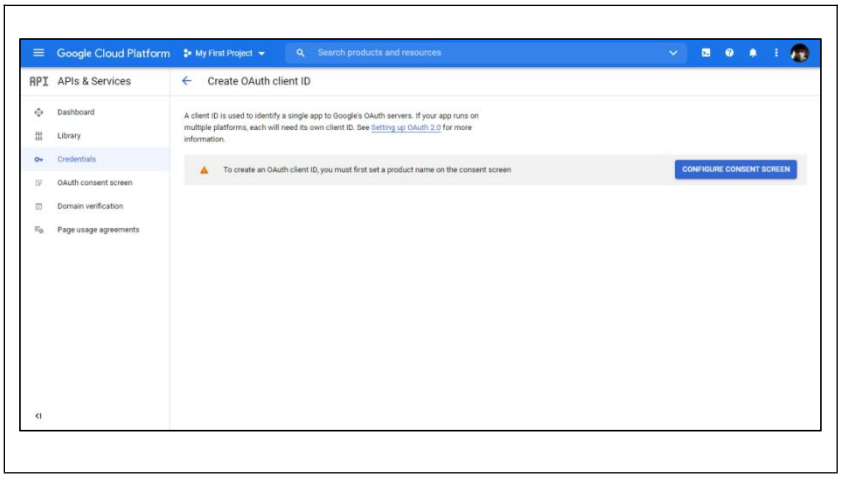
Gambar 4. 57 Halaman pengaturan credential, disini dapat mengatur, membuat dan menghapus credential

Tambahkan Credentials dengan menekan *CREATE CREDENTIALS*, pilih *OAuth Client ID*.



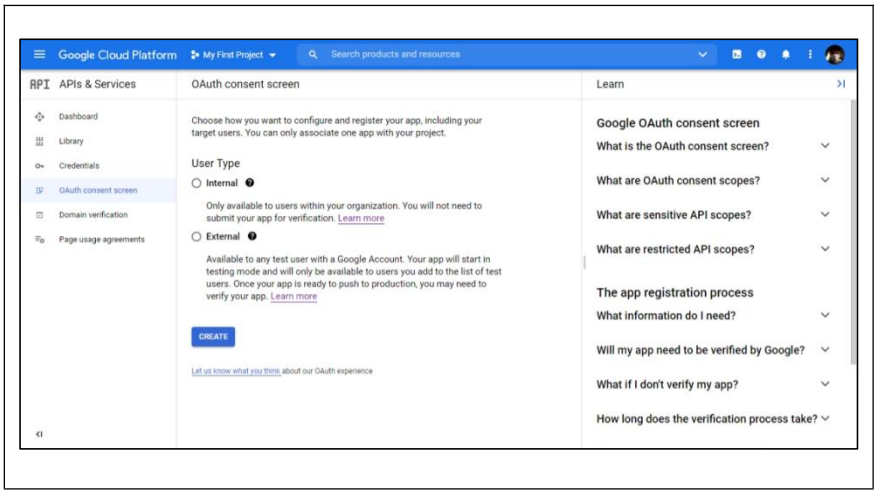
Gambar 4. 58 Langkah menambahkan credential

Pada tahap ini diminta untuk melakukan set Product name terlebih dahulu. Tekan Configure Consent Screen untuk melanjutkan.



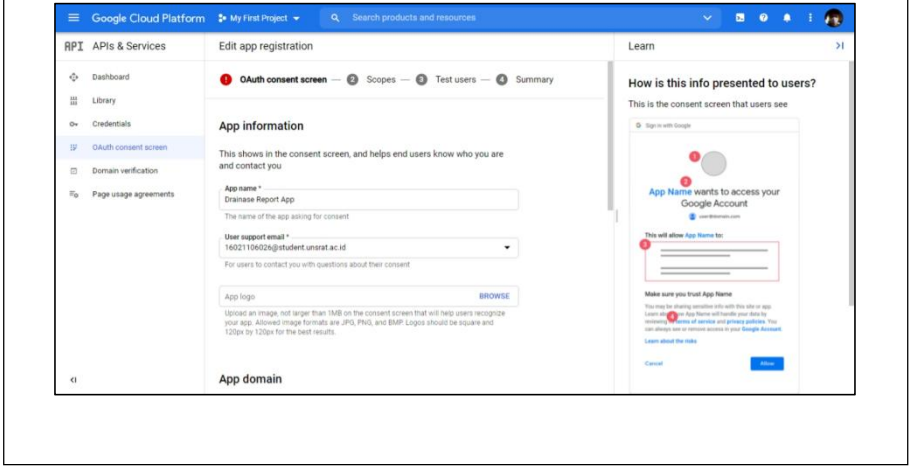
Gambar 4. 59 Langkah menambahkan Product Name.

Tekan *Create* untuk melanjutkan.

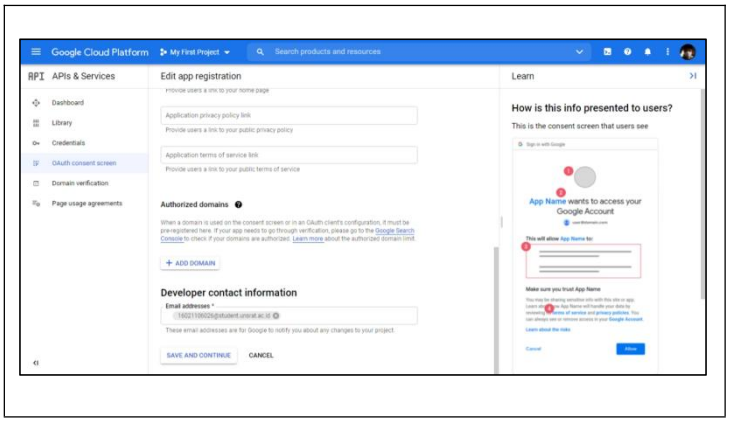


Gambar 4. 60 Tampilan OAuth consent screen lewati User Type.

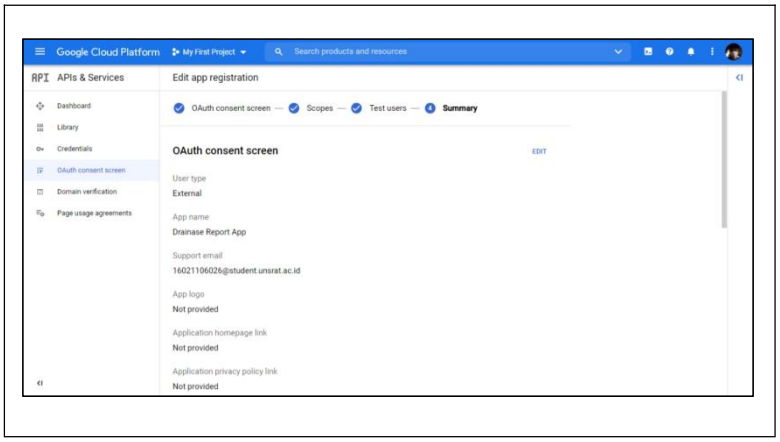
Ini adalah bagian meregistrasi aplikasi. Yang nantinya data seperti nama aplikasi dan detail-detail lainnya akan terpampang pada *sign-in user*.



Gambar 4. 61 Pengisian informasi guna memberitahukan user tentang informasi aplikasi yang ingin sign-in.

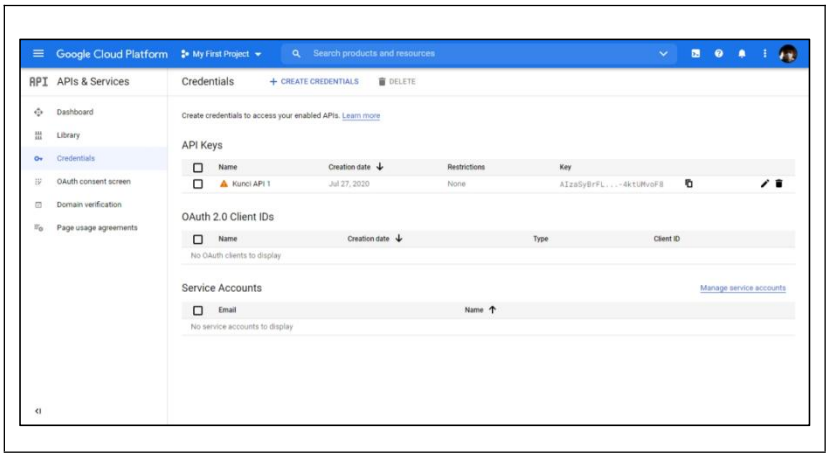


Gambar 4. 62 Halaman registrasi discroll kebawah lalu tekan tombol Save and continue



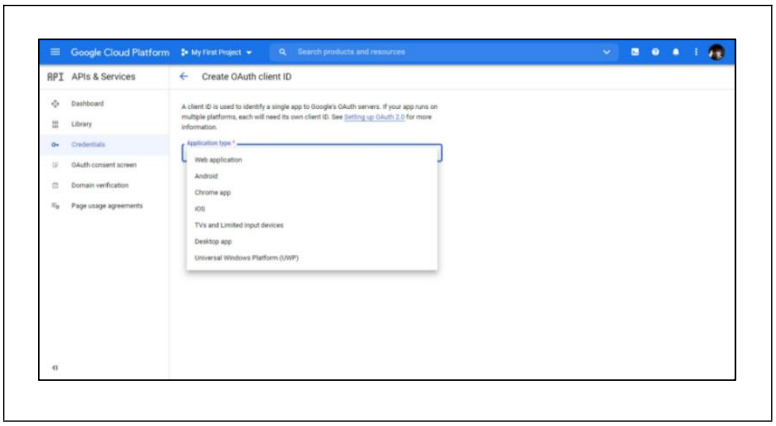
Gambar 4. 63 Rangkuman OAuth yang telah dimasukan sebelumnya

Setelah menekan *Save* and *continue*, kemudian kembali lagi ke *dashboard credentials*. Lalu tekan “CREATE CREDENTIALS” pilih “OAuth Client ID”.



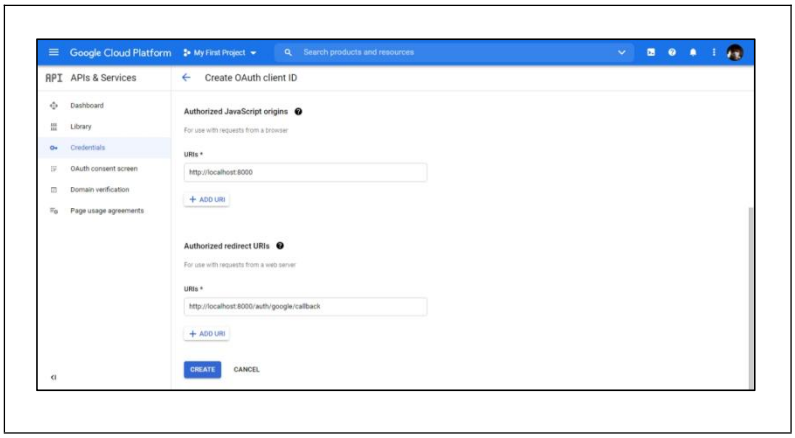
Gambar 4. 64 Halaman pengaturan credential.

Pada halaman *Create OAuth client ID* kita diminta untuk mengidentifikasi jenis aplikasi seperti apa yang ingin diterapkan. Pada kasus ini adalah *Web application.*



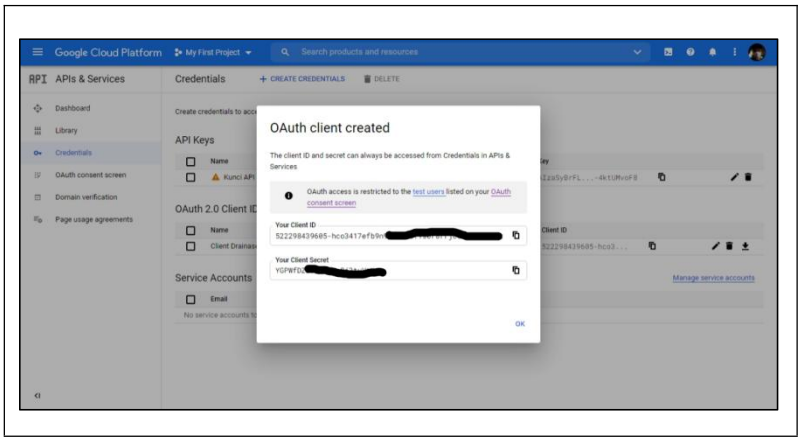
Gambar 4. 65 Tampilan Create OAuth client ID berbeda dari sebelumnya

Bagian ini merupakan pembuatan *OAuth client* dari sisi *backend*. Ada dua *field* yang harus dimasukan yaitu *Authorized JavaScript origins dan Authorized redirect URLs*. Kolom *Authorized JavaScript origins* dimasukan *url atau HTTP* yang merupakan asal dari aplikasi web yang dibuat, *URL* sementara yang diinputkan adalah `http://localhost:8000`. Kolom *Authorized redirect URLs* merupakan sebuah *URL* yang berguna untuk melakukan *trigger* atau *redirect* aplikasi web yang dibuat untuk menampilkan tampilan *google sign-in*. Pada kolom *Authorized redirect URLs* yang diinputkan adalah `http://localhost:8000/auth/google/callback`.



Gambar 4. 66 Create OAuth dengan input Authorized JavaScript origins dan Authorized redirect URLs.

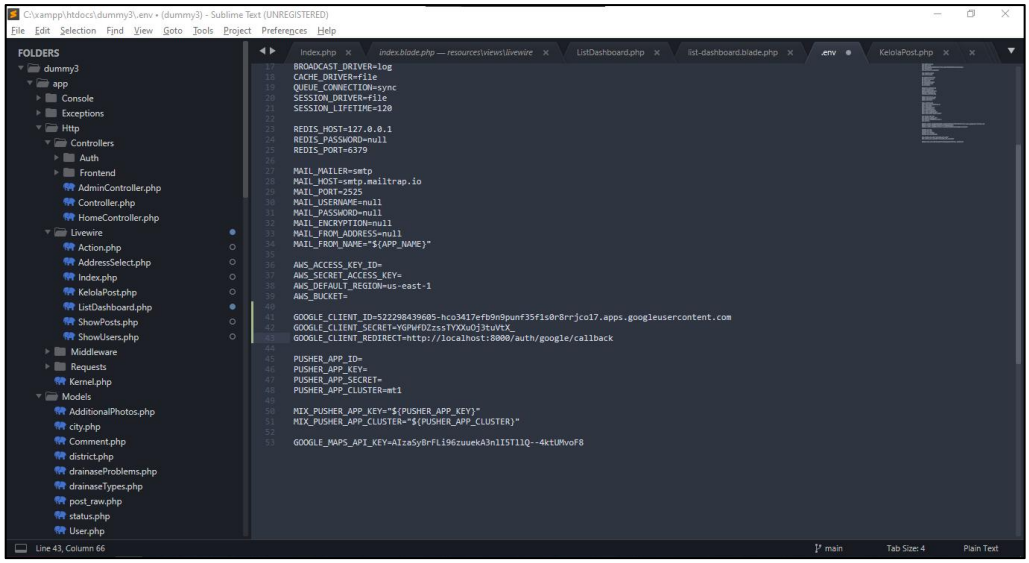
Setelah berhasil dibuat, *OAuth* client yang telah dibuat akan menghasilkan *Client ID* dan *Client Secret. Client ID* dan *Client Secret* merupakan code untuk mengakses layanan API, kode ini bersifat rahasia karena menyangkut dengan data pribadi user.



Gambar 4. 67 OAuth client created, beserta Client ID dan Client Secret sengaja disensor untuk kerahasiaan data.

Untuk merahasiakan data, data yang bersifat rahasia akan disematkan pada file *environment* (.env). Laravel memiliki fitur yang mempermudah pengembang dalam hal mengkonfigurasi segala sesuatu dalam hal *server local* atau *server production*. Misalnya saat pengembangan aplikasi user masih menggunakan server local, isinya baik *database*, *API google key* dll menggunakan driver sang pengembang aplikasi kemudian disimpan pada (.env) dan dotenv ini hanya berjalan sendiri pada *driver* pengembang tanpa takut tersebar. bila sudah tahap produksi dotenv akan kembali secara default (.env.example).

.env	Configuration



Baris Code :

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:rIJXYmDhelJFp1PqXJRG0FgrfIyGMWbHRkvPGh4MuOA=
APP_DEBUG=true
APP_URL=https://34.80.17.203/

LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=34.128.72.109
DB_PORT=3306
DB_DATABASE=mysqlbaru
DB_USERNAME=root
DB_PASSWORD=

BROADCAST_DRIVER=pusher
CACHE_DRIVER=file
FILESYSTEM_DRIVER=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=smtp
MAIL_HOST=mailhog
MAIL_PORT=1025
MAIL_USERNAME=null
MAIL_PASSWORD=null
```

```
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS=null
MAIL_FROM_NAME="${APP_NAME}"

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=
AWS_USE_PATH_STYLE_ENDPOINT=false

PUSHER_APP_ID=1622328
PUSHER_APP_KEY=471562929500248b03e0
PUSHER_APP_SECRET=1ebb82282a88f687e4a9
PUSHER_APP_CLUSTER=ap1

MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"

GOOGLE_CLIENT_ID=522298439605-
hco3417efb9n9punf35f1s0r8rrjco17.apps.googleusercontent.com
GOOGLE_CLIENT_SECRET=YGPWfDZzssTYXXuOj3tuVtX_
GOOGLE_CLIENT_REDIRECT=http://lapordrainasemdo.com/auth/google/callbac
k
GOOGLE_KEY=AIzaSyAvxGVyl8zi8_JkdC6Duk71-OYwxpju0t0

PW_DEFAULT=asddjNSAdDJbbBA91555
```

Gambar 4. 68 baris kode (.env) Pada line 41 sampai 43 merupakan hasil dari OAuth sebelumnya. Dilanjut pada line 53 merupakan key dari google maps.

Untuk lebih aman lagi, data dari dotenv (.env) dikonversikan kedalam suatu file php yaitu Services.php. Services.php ini bertujuan mengirimkan hal-hal *credentials* dari layanan pihak ketiga (*third party services*).

Baris Code :

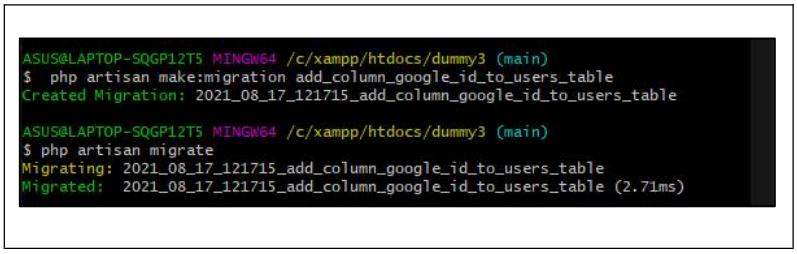
```
<?php

return [

    /*
    |--------------------------------------------------------------------
---------
    | Third Party Services
    |--------------------------------------------------------------------
---------
    |
    |  This  file  is  for  storing  the  credentials  for  third  party
services such
    | as Mailgun, Postmark, AWS and more. This file provides the de
facto
    | location for this type of information, allowing packages to have
    | a conventional file to locate the various service credentials.
    |
    */
```

```
        'mailgun' => [
            'domain' => env('MAILGUN_DOMAIN'),
            'secret' => env('MAILGUN_SECRET'),
            'endpoint' => env('MAILGUN_ENDPOINT', 'api.mailgun.net'),
        ],

        'postmark' => [
            'token' => env('POSTMARK_TOKEN'),
        ],

        'default' =>[
            'emergency' =>env('PW_DEFAULT'),
        ],

        'ses' => [
            'key' => env('AWS_ACCESS_KEY_ID'),
            'secret' => env('AWS_SECRET_ACCESS_KEY'),
            'region' => env('AWS_DEFAULT_REGION', 'us-east-1'),
        ],

        'google' => [
            'client_id' => env('GOOGLE_CLIENT_ID'),
            'client_secret' => env('GOOGLE_CLIENT_SECRET'),
            'redirect' => env('GOOGLE_CLIENT_REDIRECT'),
            'key' => env('GOOGLE_KEY')
        ],
    ];
```

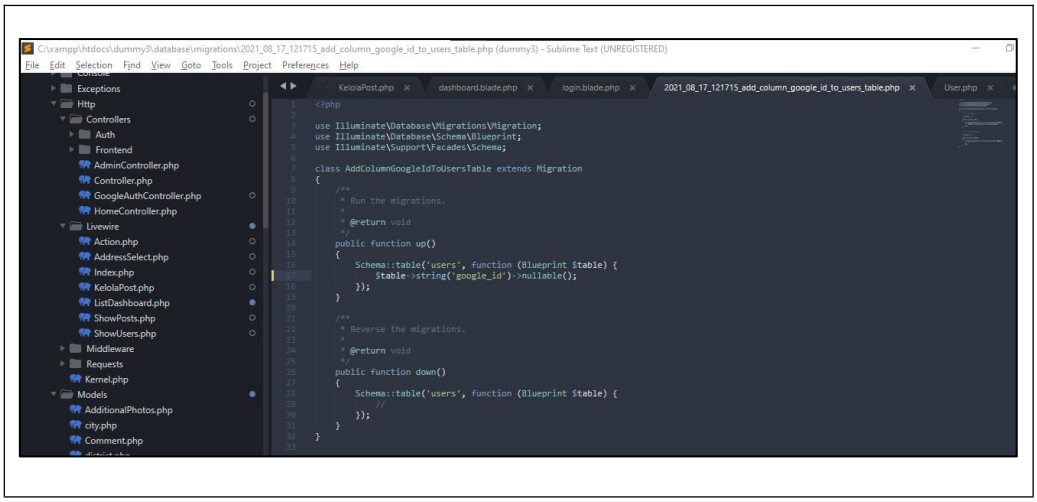
Gambar 4. 69 Berikut adalah isi dari services.php. Pada line ke 33 sampai 37 merupakan konversi dari dotenv(.env) ke variabel services.php.

Setiap *google account* tentunya memiliki ID. Untuk menyimpan ID tersebut haruslah menambahkan kolom pada tabel user. Dengan sintaks

```
php artisan make:migration add_column_google_id_to_users_table
```



Gambar 4. 70 menambahkan kolom baru pada tabel database dengan php artisan, kemudian menjalankan perintah migrate.



Gambar 4. 71 tampilan file add_column_google_id_to_users_table. Setelah itu lakukan perintah php artisan migrate untuk mengeksekusi file-file dari folder migrations.

Baris Code :

```
<?php

namespace App\Models;

use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable
{
    use HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'name',
        'email',
        'password',
        'level',
        'google_id',
        'avatar'
    ];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
```

```
            'password',
            'remember_token',
        ];

        /**
         * The attributes that should be cast to native types.
         *
         * @var array
         */
        protected $casts = [
            'email_verified_at' => 'datetime',
        ];
    }
```

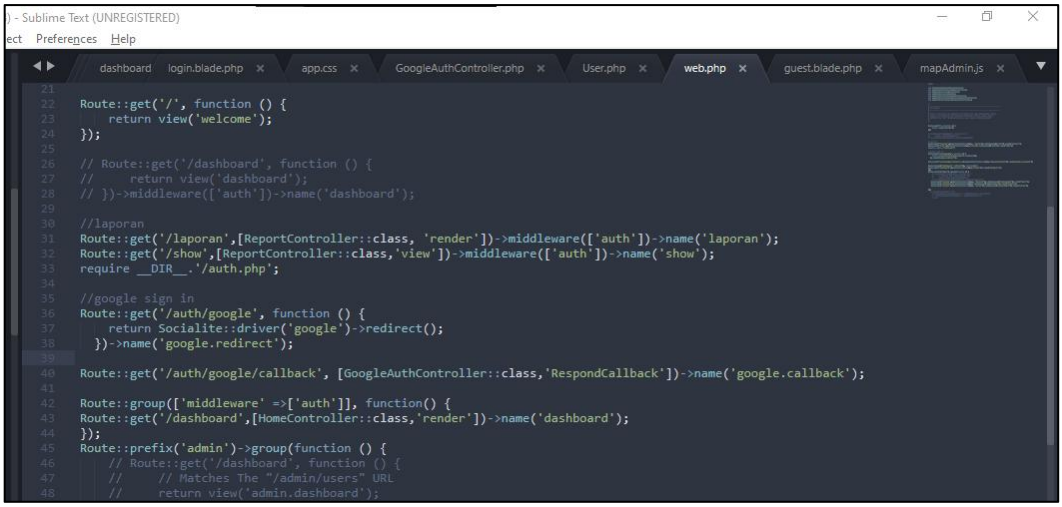
Gambar 4. 72 menambahkan kolom baru pada tabel database dengan php artisan, kemudian menjalankan perintah migrate.

Laravel Socialite merupakan suatu *package* yang berfungsi untuk menghandel segala sesuatu *OAuth authentication* baik dari *github, Facebook, Twitter, Google, LinkedIn, GitLab* dan *Bitbucket*.

```
Composer require laravel/socialite
```

Pada bagian *Route*, sebelumnya kita telah mengistallasi laravel *socialite* untuk memanggil *socialite* harus melakukan fungsi *redirect*. Fungsi *redirect* akan men-*trigger* bila user menekan tombol sign-in yang merupakan url /auth/google seperti pada line 36 sampai *line* 38. Kemudian pada *line* ke 40, harus pastikan jika path *Controller GoogleAuthController* sudah sesuai dan *class* RespondCallback akan bertindak jika url '/auth/google/callback' telah dijalankan.

Web.php	Routes



Baris Code :

```
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\AdminController;
use App\Http\Livewire\KelolaPost;
use App\Http\Livewire\Action;
use App\Http\Controllers\HomeController;
use App\Http\Controllers\Frontend\ReportController;
use App\Http\Controllers\GoogleAuthController;

/*
|--------------------------------------------------------------------------
-----
| Web Routes
|--------------------------------------------------------------------------
-----
|
| Here is where you can register web routes for your application.
These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get('/', function () {
    return view('dashboard');
});


//laporan
Route::get('/laporan',[ReportController::class, 'render'])-
>middleware(['auth'])->name('laporan');
Route::get('/show',[ReportController::class,'view'])-
>middleware(['auth'])->name('show');
require __DIR__.'/auth.php';

//google sign in
```

```
      Route::get('/auth/google', function () {
          return Socialite::driver('google')->redirect();
        })->name('google.redirect');

      Route::get('/auth/google/callback',
[GoogleAuthController::class,'RespondCallback'])->name('google.callback');


      Route::get('/dashboard',[HomeController::class,'render'])-
>name('dashboard');


      Route::prefix('admin')->group(function () {

         Route::get('/kelola',[AdminController::class, 'ShowPost'])-
>middleware(['admin:admin'])->name('posts');
         Route::get('/kelola/{id}',KelolaPost::class)-
>middleware(['admin:admin'])->name('kelola');
         Route::get('/action',[AdminController::class, 'Action'])-
>middleware(['admin:admin'])->name('action');

      });
```

Gambar 4. 73 Route Aplikasi web pelaporan pada line 31 sampai dengan line 40 berhubungan dengan google sign-in.

Dalam Controller GoogleAuthController mengatur proses *sign-in user*. Pertama lakukan pemanggilan class-class yang akan digunakan seperti Request, *Socialite, model user,* dan *Auth.* Pada *controller* ini memiliki *public class* yaitu RespondCallback. Variabel $user berisikan data *Request* yang berasal dari Socialite dimana data tersebut secara otomatis didapatkan saat user melakukan *sign-in*, data *google account* kemudian tersaring kedalam *request Socialite* yang tentu hanya sementara saja. Variabel $users disini untuk melihat apakah *user* sebelumnya sudah pernah *sign-in* atau belum pernah jika *user* pernah *sign-in* dan ter-*record* oleh *database* maka *user* dipersilahkan langsung ke *dashboard*. Jika baru pertama kali mencoba *sign-in* dan belum terdaftar ditabel *user database*, maka variabel $newUser akan dijalankan pemrosesan variabel $newUser adalah model *user* dibuat baru dengan data yang diambil dari variabel $user (*request Socialite*) seperti nama*,email* dan id. Kolom 'level' merupakan suatu status pada *user*, status pada *user* terbagi menjadi dua yang pertama 'admin' dan kedua '2' (*user*). *password user* terenkripsi dengan bantuan fungsi *encrypt*. Jika sudah variabel $newUser yang beriskan data baru tadi akan langsung ke fungsi login, kemudian kembali ke dashboard.

Baris Code GoogleAuthController.php :

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Laravel\Socialite\Facades\Socialite;
use App\Models\User;
use Illuminate\Support\Facades\Auth;


class GoogleAuthController extends Controller
{
    public function RespondCallback()
    {
       try {
            $user = Socialite::driver('google')->user();
            $users = User::where('google_id',$user->id)->first();
            if ($users) {
              Auth::login($users);
              return redirect()->route('dashboard');
            }else{
              $newUser = User::create([
                           'name' => $user->name,
                           'email' => $user->email,
                           'google_id' => $user->id,
                           'level'=> 2,
                           'password'=>
encrypt(config('services.default.emergency')),
                'avatar'=>$user->avatar,
                ]);
                Auth::login($newUser);
                return redirect()->route('dashboard');

            }
       } catch (Exception $e) {
                    dd($e->getMessage());
       }
    }
}
```

Gambar 4. 74 Controller GoogleAuthController mengatur jalannya sign-in user.

Berikutnya pada tahap *View*. *Layout* terbagi menjadi tiga *file blade view* yang terpisah yaitu app.blade.php, untuk *layout user* yang telah berhasil *login*, Admin.blade.php untuk *layout admin*, dan terakhir guest.blade.php layout untuk melakukan sign-in baik user maupun admin. Penggunaan layout bermanfaat dalam efisiensi koding seperti *layout* guest, dimana satu *layout view* ini dapat digunakan ke bebearapa komponen view lainnya, tinggal tiap content akan disematkan pada {{ $slot }}.

Penerapan *Google OAuth* pada *view*. Sebelumnya informasi tentang *google secret* maupun *client id* sudah disimpan pada dotenv(.env) dan dikonversi ke services.php. Saatnya untuk memanggil informasi *google* tersebut dari services.php kemudian menempatkan ke *head* seperti *line* ke 6.

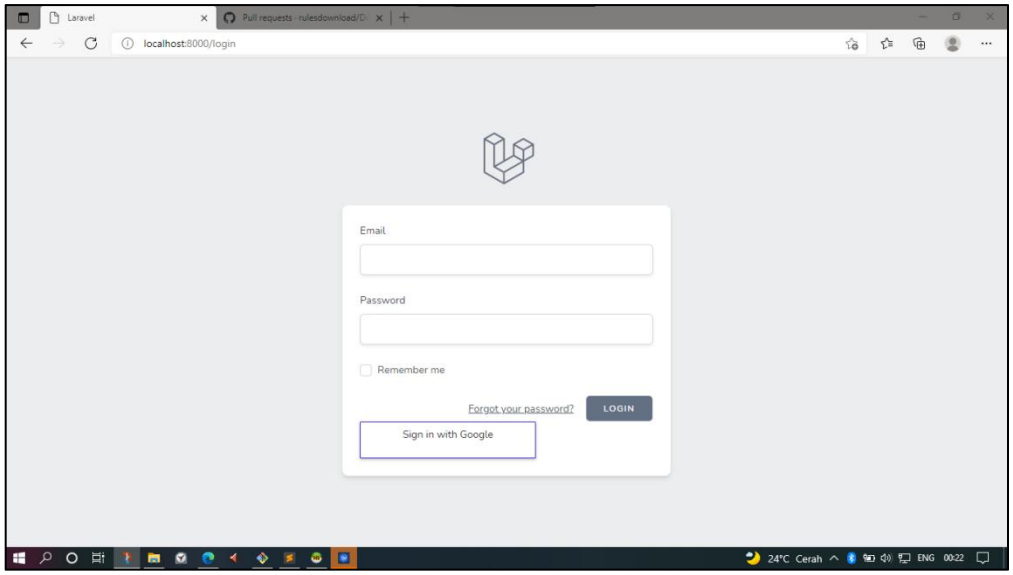
Baris Code guest.blade.php:

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1">
        <meta name="google-signin-client_id"
content="{{config('services.google.client_id')}}">
        <meta name="csrf-token" content="{{ csrf_token() }}">

        <title>{{ config('app.name', 'Laravel') }}</title>

        <!-- Fonts -->
        <link rel="stylesheet"
href="https://fonts.googleapis.com/css2?family=Nunito:wght@400;600;700&displ
ay=swap">

        <!-- Styles -->
        <link rel="stylesheet" href="{{ asset('css/app.css') }}">

        <!-- Scripts -->
        <script src="{{ asset('js/app.js') }}" defer></script>


    </head>
    <body>
        <div class="font-sans text-gray-900 antialiased">
            {{ $slot }}
        </div>
    </body>
</html>
```

Gambar 4. 75 Layout guest.blade.php

Berikut merupakan hasil dari *view* login.blade.php. Untuk gambar dibawah sementara digabungkan *login* dari *admin* dan *login user*. Yang membedakan *login* admin dan user. Admin melakukan *login* dengan *field form* email dan password kemudian tekan *login*. Sedangkan *Login User* hanya perlu memencet tombol '*Sign in with Google*' tanpa memasukan form email dan password.

Adminlogin.blade.php	View



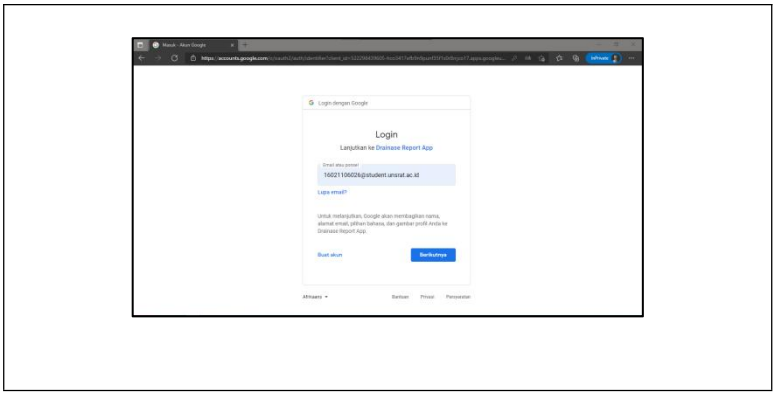
Baris Code :

```
      <x-guest-layout>
          <x-auth-card>
              <x-slot name="logo">
                  <a href="/">
                      <x-application-logo class="w-20 h-20 fill-current
text-gray-500" />
                  </a>
              </x-slot>

              <!-- Session Status -->
              <x-auth-session-status class="mb-
4" :status="session('status')" />


              <!-- Validation Errors -->
              <x-auth-validation-errors class="mb-4" :errors="$errors" />

              <form method="POST" action="{{ route('adminlogin') }}">
                  @csrf

                  <!-- Email Address -->
                  <div>
                      <x-label for="email" :value="__('Email')" />

                      <x-input id="email" class="block mt-1 w-full"
type="email" name="email" required autofocus />
                  </div>

                  <!-- Password -->
                  <div class="mt-4">
                      <x-label for="password" :value="__('Password')" />

                      <x-input id="password" class="block mt-1 w-full"
                                      type="password"
```

```
                                        name="password"
                                        required autocomplete="current-
password" />
                    </div>

                    <!-- Remember Me -->
                    <div class="block mt-4">
                        <label for="remember_me" class="flex items-center">
                            <input id="remember_me" type="checkbox"
class="form-checkbox" name="remember">
                            <span class="ml-2 text-sm text-gray-
600">{{ __('Remember me') }}</span>
                        </label>
                    </div>

                    <div class="flex items-center justify-end mt-4">
                        @if (Route::has('password.request'))
                            <a class="underline text-sm text-gray-600
hover:text-gray-900" href="{{ route('password.request') }}">
                                {{ __('Forgot your password?') }}
                            </a>
                        @endif

                        <x-button class="ml-3">
                            {{ __('Login') }}
                        </x-button>
                    </div>
                </form>
            </x-auth-card>
        </x-guest-layout>
```

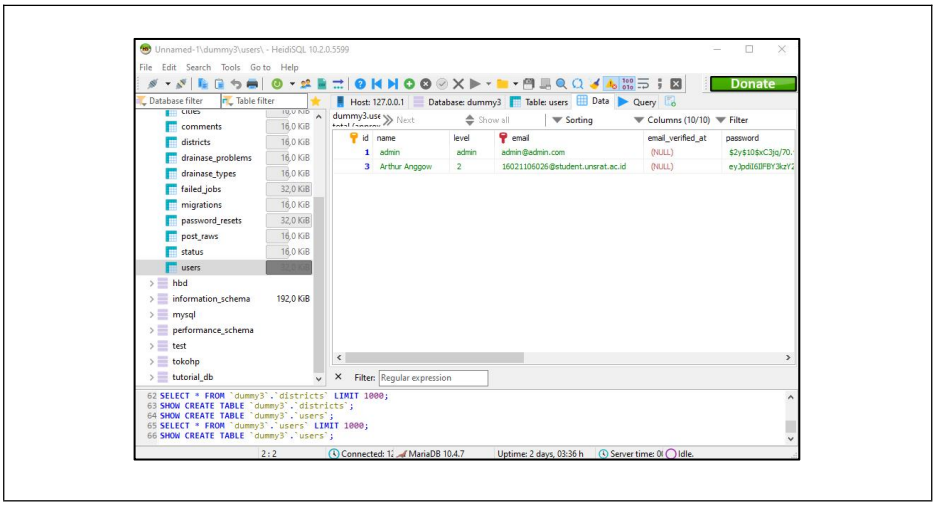
Gambar 4. 76 tampilan halaman login dengan url localhost:8000/login.

Setelah User menekan tombol '*Sign in with Google*' secara otomatis akan di-direct ke google OAuth. Terdapat informasi seperti nama aplikasi dan form untuk memasukan email atau ponsel milik user.



Gambar 4. 77 Halaman dari Google OAuth, proses direct dijalankan pada halaman tab yang sama, artinya tidak akan membuka tab baru atau jendela baru bila menekan tombol Sign in with Google.

Berikut adalah data dari *google account* yang berhasil disimpan pada *database local*, dengan nama tabel “Users”, kolom satu merupakan admin dan kolom ke dua dan selanjutnya merupakan user.



Gambar 4. 78 Menampilkan data yang tersimpan menggunakan HeidySql.

E) Pembuatan fungsi Seeder

Seeders merupakan salah satu fitur yang dimiliki oleh *laravel* bertujuan untuk menyusun data *testing* atau *data default* agar saat melakukan *testing* tidak perlu lagi mengisi data secara berulang-ulang. Untuk membuat *seeder* hanya menggunakan sintaks `php artisan make:seeder namaSeeder` kemudian komponen Seeder berada pada direktori `database/seeders/` .

Gambar 4. 79 Sintaks membuat Seeder dengan php artisan.

Dalam direktori database/seeders/ terdapat sebuah komponen dengan nama DatabaseSeeder.php. Komponen ini yang nanti akan mengidentifikasi file-file seeder tersebut melalui nama file *seeder* yang berada pada direktori yang sama. Untuk

memanggil variabel seeder harus berada pada *public function run* dan sintaksnya sebagai berikut:

```
$this->call(namaSeeder::class);
```

Setelah itu pada tahap seeder tinggal mengisi data yang akan menjadi data *testing*. Pada MarkerSeeder.php data yang dimasukan adalah nama *marker* kolom dari filename.

Baris Code :

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;
use Illuminate\Support\Str;
use App\Models\User;

class userSeeder extends Seeder
{
    /**
     * Run the database seeds.
     * @return void
     */
    public function run()
    {
       User::truncate();
     User::create([
               'name' => 'admin',
             'level' => 'admin',
             'email' => 'admin@admin.com',
             'password' => bcrypt('admin'),
             'remember_token' => Str::random(60),
         ]);
       }
}
```

Gambar 4. 80 Seeder UserSeeder.php.

Baris Code :

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;
use App\Models\Marker;

class MarkerSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
```

```
     * @return void
     */
    public function run()
    {
        Marker::truncate();
        Marker::create([
                 'filename' => 'marker-putih.png'
                        ]);

        Marker::create([
                 'filename' => 'marker-merah.png'
                        ]);

        Marker::create([
                 'filename' => 'marker-pink.png'
                        ]);

        Marker::create([
                 'filename' => 'marker-oranje.png'
                        ]);

        Marker::create([
                 'filename' => 'marker-magenta.png'
                        ]);

        Marker::create([
                 'filename' => 'marker-lime.png'
                        ]);

        Marker::create([
                 'filename' => 'marker-lavender.png'
                        ]);

        Marker::create([
                 'filename' => 'marker-kuning.png'
                        ]);

        Marker::create([
                 'filename' => 'marker-krem.png'
                        ]);

        Marker::create([
                 'filename' => 'marker-htam.png'
                        ]);

        Marker::create([
                 'filename' => 'marker-hijau.png'
                        ]);

        Marker::create([
                 'filename' => 'marker-cyan.png'
                        ]);

        Marker::create([
                 'filename' => 'marker-coklat.png'
                        ]);

        Marker::create([
                 'filename' => 'marker-birudongker.png'
                        ]);
```

```
            Marker::create([
                     'filename' => 'marker-biru.png'
                            ]);

        }
    }
```

Gambar 4. 81 Seeder MakerSeeder.php.

Baris Code :

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;
use App\Models\drainaseTypes;

class DrainaseTypesSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
     drainaseTypes::truncate();
     drainaseTypes::create([
             'tipe' => 'empty'
         ]);
     drainaseTypes::create([
             'tipe' => 'natural drainage'
         ]);
     drainaseTypes::create([
             'tipe' => 'artificial drainage'
         ]);
     drainaseTypes::create([
             'tipe' => 'Sub surface drainage'
         ]);
     drainaseTypes::create([
             'tipe' => 'Drainase Tersier'
         ]);
        drainaseTypes::create([
             'tipe' => 'Drainase Sekunder'
         ]);
        drainaseTypes::create([
             'tipe' => 'Drainase Primer'
         ]);
    }
}
```

Gambar 4. 82 Seeder DrainaseTypesSeeder.php.

Baris Code :

```
<?php
```

```
namespace Database\Seeders;

use Illuminate\Database\Seeder;
use App\Models\drainaseProblems;

class DrainaseProblemsSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
     drainaseProblems::truncate();
     drainaseProblems::create([
             'problem' => 'empty',
             'marker_id'=> 1,
         ]);
     drainaseProblems::create([
             'problem' => 'Saluran penuh sampah',
             'marker_id'=> 2,
         ]);
     drainaseProblems::create([
             'problem' => 'Saluran kurang dalam',
             'marker_id'=> 3,
         ]);
     drainaseProblems::create([
             'problem' => 'Saluran banyak lumpur atau tanah',
             'marker_id'=> 4,
         ]);
     drainaseProblems::create([
             'problem' => 'Air disaluran keluar kejalan saat hujan',
             'marker_id'=> 5,
         ]);
     drainaseProblems::create([
             'problem' => 'Saluran tak bisa dimasuki air dari jalan',
             'marker_id'=> 6,
         ]);
     drainaseProblems::create([
             'problem' => 'Sisa penggalian belum diangkat',
             'marker_id'=> 7,
         ]);
     drainaseProblems::create([
             'problem' => 'Trotoar tidak tertutup',
             'marker_id'=> 8,
         ]);
     drainaseProblems::create([
             'problem' => 'Trotoar tidak tertutup dan penuh sampah',
             'marker_id'=> 9,
         ]);
     drainaseProblems::create([
             'problem' => 'Trotoar tidak tertutup dan penuh lumpur',
             'marker_id'=> 10,
         ]);
     drainaseProblems::create([
             'problem' => 'Trotoar berlubang-lubang',
             'marker_id'=> 11,
         ]);
```

```
    }
}
```

Gambar 4. 83 Seeder DrainaseTypesSeeder.php.

Baris Code :

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;
use App\Models\district;

class DistrictSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
     District::truncate();
     District::create([
             'kelurahan' => 'Bumi Nyiur',
             'cities_id' => 1
                    ]);
     District::create([
             'kelurahan' => 'Karombasan Selatan',
             'cities_id' => 1
                    ]);
     District::create([
             'kelurahan' => 'Karombasan Utara',
             'cities_id' => 1
                    ]);
     District::create([
             'kelurahan' => 'Tingkulu',
             'cities_id' => 1
                    ]);
. . .
     District::create([
             'kelurahan' => 'Paniki Bawah',
             'cities_id' => 11
                    ]);
    }
}
```

Gambar 4. 84 Seeder DistrictSeeder.php.

Baris Code :

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;
```

```
use App\Models\City;

class CitySeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
     City::truncate();
     City::create([
             'Kecamatan' => 'Wanea',
             'Camat' => 'Mario Reivalino Rio Karundeng, S.STP'
                    ]);
     City::create([
             'Kecamatan' => 'Wenang',
             'Camat' => 'Deysie Kalalo, SE MAP'
                    ]);
     City::create([
             'Kecamatan' => 'Tuminting',
             'Camat' => 'Dany H. Kumajas, S.SOS'
                    ]);
     City::create([
             'Kecamatan' => 'Tikala',
             'Camat' => 'Argo B Sangkay'
                    ]);
     City::create([
             'Kecamatan' => 'Singkil',
             'Camat' => 'Zainal Naway, S.Sos'
                    ]);
    City::create([
             'Kecamatan' => 'Sario',
             'Camat' => 'Handry Jouke Lasut, SE'
                    ]);
     City::create([
             'Kecamatan' => 'Pall Dua',
             'Camat' => 'Glennstiano Kowaas, SH MH'
                    ]);
     City::create([
             'Kecamatan' => 'Malalayang',
             'Camat' => 'Reintje Heidemans'
                    ]);
     City::create([
             'Kecamatan' => 'Bunaken Kepulauan',
             'Camat' => 'Christian Salindeho'
                    ]);
     City::create([
             'Kecamatan' => 'Bunaken',
             'Camat' => 'Drs. Boyke Pandean'
                    ]);
     City::create([
             'Kecamatan' => 'Mapanget',
             'Camat' => 'Robert Dauhan S,STP'
                    ]);
    }

}
```

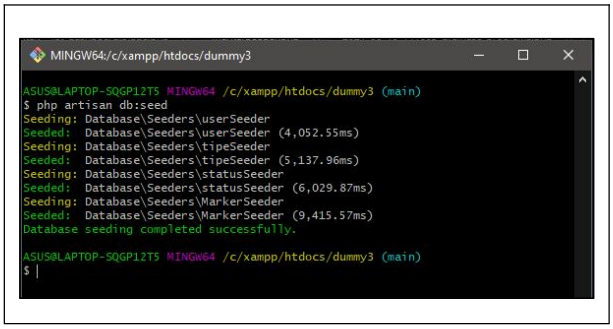
Gambar 4. 85 Seeder CitySeeder.php.

Pada tahap ini merupakan langkah eksekusi untuk menjalankan Seeders tersebut. Komponen DatabaseSeeders.php diolah dengan perintah “php artisan db:seed”. Jika ada seed yang error proses akan diberhentikan dan developer diharapkan melihat seedersnya kembali.

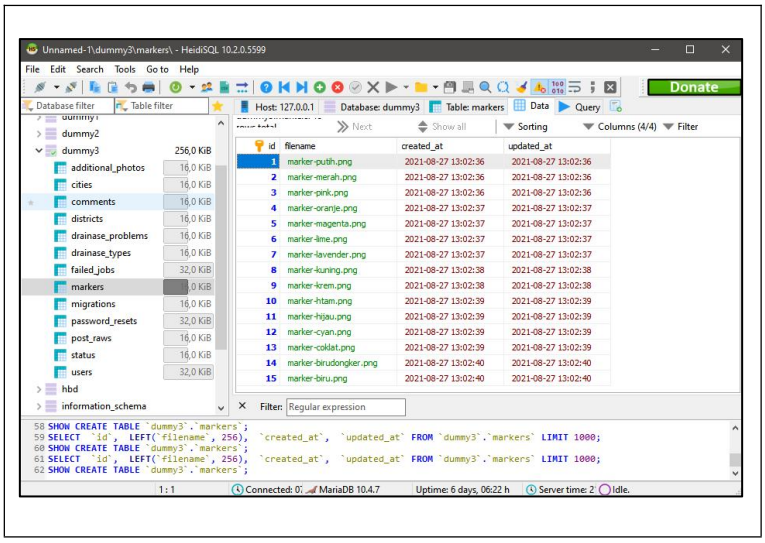
Baris Code :

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     *
     * @return void
     */
    public function run()
    {
        // \App\Models\User::factory(10)->create();
        $this->call(drainaseProblemsSeeder::class);
        $this->call(drainaseTypesSeeder::class);
        $this->call(CitySeeder::class);
        $this->call(DistrictSeeder::class);
        $this->call(StatusSeeder::class);
        $this->call(MarkerSeeder::class);
        $this->call(UserSeeder::class);
    }
}
```

Gambar 4. 86 Seeder DatabaseSeeder.php.



Gambar 4. 87 Perintah untuk mengeksekusi Seed dengan php artisan.

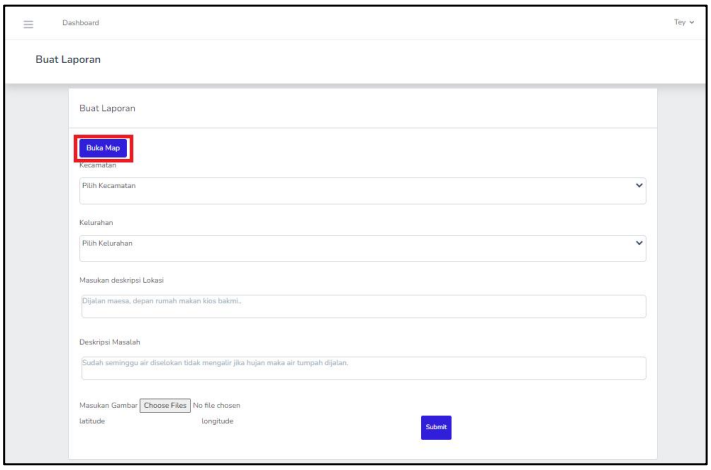


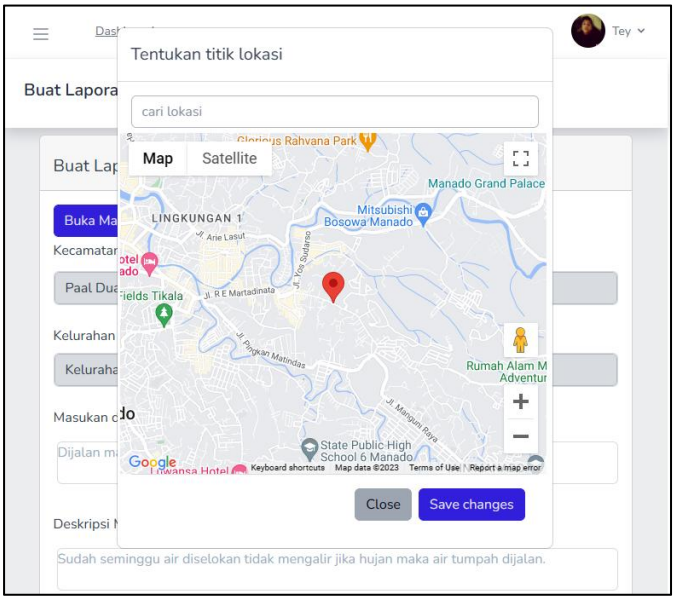
Gambar 4. 88 Tampilan tabel Markers telah berisi data lewat proses seeders.

F) Pengintegrasian map beserta marker pada aplikasi.

Pada bagian ini penulis akan memprogram agar *marker* dan jenis-jenis permasalahan drainase terhubung satu sama lain. Sebelumnya penulis telah berhasil melakukan percobaan fungsi-fungsi *Google Maps* dengan menggunakan *Jsfiddle*, mendapatkan *API Keys*, membuat *logo marker,* membuat tabel dan model dari drainase_problem dan marker. Peta akan digunakan pada Halaman address-select.blade.php (pembuatan laporan), list-dashboard.blade.php (*dashboard*), index.blade.php (hasil laporan), kelola-post.blade.php (pengolahan laporan oleh admin) dan Action.blade.php (untuk mengubah dan menambah marker baru). Pada halaman address-select.blade.php yang merupakan halaman pembuatan laporan, pada halaman tersebut terdiri dari halaman form dan halaman modal untuk membuka map.

address-select.blade.php	View





Baris Code :

```
<div >
          <button type="button" class="btn btn-primary" data-toggle="modal"
data-target="#exampleModal" data-backdrop="false">
            Buka Map
          </button>
                  @error('lat') <span class="error">{{ $message }}</span>
@enderror
          <form wire:submit.prevent="store">
               <div class="form-group " >

                <div class="relative mt-1 mb-4" >
```

```
                  <label for="city"> Kecamatan </label>
                    <div class="mt-1 mb-4 border rounded-md">
                    <select
                    name="cities"
                    wire:model.lazy="cities"
                    class=" w-full outline-primary form-control"disabled >
                       value='{{$cities}}'>Kecamatan</option>

                        <option class="disabled" id="dropkec"
value={{$cities}}>Kecamatan</option>

                    </select>
                      @error('city') <span
class="error">{{ $message }}</span> @enderror
                        </div>
                </div>

                <div class="mt-1 mb-4 disabled">
                  <label for="district"> Kelurahan </label>
                    <div class="border rounded-md ">
                    <select
                    name="districts"
                    wire:model.lazy="districts"
                    class=" w-full outline-primary form-control" disabled>
                      <option value='{{$districts}}'>Kelurahan</option>

                        <option id="dropkel" value={{ $districts}}></option>

                    </select>
                        @error('district') <span
class="error">{{ $message }}</span> @enderror
                      </div>
                </div>

              </div>
          <div class="form-group">
           <label for="des_lok"> Masukan deskripsi Lokasi </label>
              <div class="mt-1 mb-4 pl-1 pt-1 pb-1 " >
              <textarea class="resize-none border rounded-md focus:outline-
none w-full" placeholder="Dijalan maesa, depan rumah makan kios bakmi.."
name="des_lok"  wire:model.defer="des_lok" wire:ignore></textarea>
              @error('des_lok') <span class="error">{{ $message }}</span>
@enderror
              </div>
            </div>

            <div class="form-group">
            <label for="des_mas">Deskripsi Masalah</label>
              <div class="mt-1 mb-4 pl-1 pt-1 pb-1 " >
              <textarea class="resize-none border rounded-md focus:outline-
none w-full" placeholder="Sudah seminggu air diselokan tidak mengalir jika
hujan maka air tumpah dijalan. " wire:model.defer="des_mas"></textarea>
              @error('des_mas') <span class="error">{{ $message }}</span>
@enderror
              </div>

          </div>

              <label for="photo_input"> Masukan Gambar </label>
              <input type="file" wire:model.lazy="photos" multiple>
```

```
              <div wire:loading wire:target="photos">Uploading...</div>
          @error('photos.*') <span class="error">{{ $message }}</span>
@enderror

          <div class="form-group" >
              <input type="" id="latitudehide" name="latitudehide"
wire:model.lazy="lat" wire:model.lazy="lat"  >
              <input type="" id="longitudehide" name="longitudehide"
wire:model.lazy="lng" wire:model.lazy="lng" >
          </div>

           <label for="status_id"></label>
            <input type="hidden" name="status_id" class="">

          <button type="submit" class="btn btn-sm btn-primary"
wire:loading.class="bg-gray" >Submit</button>

          </form>
          {{$prompt}}
          <!-- Button trigger modal -->

      <!-- Modal Buka map -->
          <div class="modal" id="exampleModal" tabindex="-1" aria-
labelledby="exampleModalLabel" aria-hidden="true">
            <div class="modal-dialog">
              <div class="modal-content"style="width: 512px;right: 7px;">
                <div class="modal-header">
                  <h5 class="modal-title" id="exampleModalLabel">Tentukan
titik lokasi</h5>
                  <button type="button" class="close" data-dismiss="modal"
aria-label="Close">
                    <span aria-hidden="true">&times;</span>
                  </button>
                </div>
                <div class="modal-body">

              <input id="pac-input" type="text" placeholder="cari
lokasi"class="mb-1.5">
              <div wire:ignore id="map" class="" style="width: 500px;height:
400px;float: left;position: relative;overflow: hidden;left: -23;right:
13px;left: -10;"></div>

              <input id="infolat"  type="text" name="infolat"
class="hidden">
              <input id="infolng"  type="text" name="infolng"
class="hidden">
              <input id="infokec" name="infokecamatan" class="hidden">
       <input id="infokel" name="infokelurahan" class="hidden">
              <div id="infoPanel" class="hidden">
                  <b>Marker status:</b>
                  <div id="markerStatus"><i>Click and drag the
marker.</i></div>
                  <b>Current position:</b>
              </div>
                <div class="modal-footer">
                  <button type="button" class="btn btn-secondary" data-
dismiss="modal">Close</button>
                  <button onclick="myFunction()" id="save" type="button"
class="btn btn-primary" data-dismiss="modal">Save changes</button>
                </div>
```

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

      </div>

      @section('scripts')
      <script async defer
src="https://maps.googleapis.com/maps/api/js?key={{config('services.google.k
ey')}}&libraries=places&callback=initialize"
        type="text/javascript"></script>
       <script src="{{ asset('public/js/mapInput.js') }}" defer></script>

      <script >
      // supaya menu dalam searchbox tidak membelakangi modal
      var pacContainerInitialized = false;
      $("#pac-input").keypress(function() {
        if (!pacContainerInitialized) {
          $(".pac-container").css("z-index", "9999");
          pacContainerInitialized = true;
        }
      });
// ambil data latitude dan longitude dari input yang dimodal
    function myFunction() {
      var lat = document.getElementById("infolat").value;
      var lng = document.getElementById("infolng").value;
      var kec = document.getElementById("infokec").value;
      var kel = document.getElementById("infokel").value;
      console.log(kec);
    // agar Javascript dapat berkomunikasi dng Livewire
      Livewire.emit('getLatitudeForInput',lat);
      Livewire.emit('getLongitudeForInput',lng);
      Livewire.emit('getKecamatan',kec);
      Livewire.emit('getKelurahan',kel);
 $(document).keypress(
         function(event){
            if (event.which == '13') {
         event.preventDefault();
    }
});

</script>
@endsection
```

Gambar 4. 89 Addres-select.blade.php

Kode pada gambar 4.87 terdapat dua jenis bahasa pemrograman yaitu *HTML* dan *Javascript.* Pada html terdapat form (*input field*, *option field*, *input photo field*), dan tombol (“buka map” dan “submit”). Data yang dimasukan dalam form akan disetorkan kedalam *database* post_raw ketika user menekan tombol submit akan menjalankan *class* `wire:submit.prevent="store"`. Tantangannya adalah untuk

memindahkan *data* dari *google maps* (*Javascript*) ke *form* yang proses tersebut dijalankan oleh *Livewire*. Pada *line 97* merupakan div dengan *class* modal yang bertujuan untuk membuka peta, dalam modal terdapat `id="map"` dengan atribut *livewire* `wire:ignore` sintaks tersebut digunakan agar *Livewire* tidak melakukan perubahan atau manipulasi pada bagian *component* dari *HTML* yang ditandai. karena sulit bagi *Livewire* melacak *DOM* yang ingin dipertahankan, karena tentu terdapat banyak macam *libraries* yang memanipulasi *DOM* diwaktu yang sama. Dalam modal juga ada *input field* dengan id "infolat" & "infolng" yang berfungsi untuk menerima angka *latitude* dan *longitude* yang dihasilkan dari *Googlemaps* (*javascript*) dengan tipe data *JSON,* input *field* tersebut termasuk dalam *class* `wire:submit.prevent="store"` yang berarti tinggal menunggu *user* memencet tombol *submit* sebagai *trigger* agar class *store* berjalan. Selanjutnya adalah bagian kode *Javascript* yang terdapat pada *line 129* terdapat sintaks diawali `@section('scripts')` dan diakhiri `@endsection` hal ini dilakukan agar *blade* tersebut dapat memasukan sintaks-sintaks dari bahasa pemrograman *Javascript.* Dalam bagian *script* tersebut ada terdapat *script* kode link beserta kode Google Maps API , *script* koding peta yang telah direkayasa pada *line 132* hanya berupa sumber jalur penyimpanan kode tersebut (mapinput.js), dan yang terpenting adalah script dengan fungsi `myFunction()` dimana fungsi tersebut yang menangkap nilai *latitude* dan *longitude* dari koding peta yang telah direkayasa ke *form input field* pada halaman *blade.*

Baris Code Mapinput.js :

```
function initialize() {


  var crosshairShape = {
    coords: [0, 0, 0, 0],
    type: 'rect'
  };
  var mapCanvas = document.getElementById('map');
  var mapOptions = {
    center: new google.maps.LatLng(1.474830, 124.842079),
    zoom: 14,
    scrollwheel: false,
    panControl: false,
    zoomControl: true,
    mapTypeId: google.maps.MapTypeId.ROADMAP
```

```
  }
  var map = new google.maps.Map(mapCanvas, mapOptions);

  var marker = new google.maps.Marker({
    map: map,
    draggable: true
  });
 var geocoder = new google.maps.Geocoder();

function geocodePosition(pos) {
  geocoder.geocode({
    latLng: pos
  }, function(responses) {
    if (responses && responses.length > 0) {
      updateMarkerAddress(responses[0].formatted_address);
    } else {
      updateMarkerAddress('Cannot determine address at this location.');
    }
  });

}
//Search box
  const input = document.getElementById("pac-input");
  const searchBox = new google.maps.places.SearchBox(input);

    map.addListener("bounds_changed", () => {
    searchBox.setBounds(map.getBounds());
  });

searchBox.addListener("places_changed", () => {
    const places = searchBox.getPlaces();

    if (places.length == 0) {
      return;
    }


    // For each place, get the icon, name and location.
    const bounds = new google.maps.LatLngBounds();
    places.forEach((place) => {
      if (!place.geometry) {
        console.log("Returned place contains no geometry");
        return;
      }

      /* google.maps.event.addListener(map,'dragend', function (event) {
            marker(this, event.latLng);
      }); */

      console.log(place.geometry.location.toJSON());

      if (place.geometry.viewport) {
        // Only geocodes have viewport.
        bounds.union(place.geometry.viewport);
        document.getElementById('infolat').value =
place.geometry.location.lat();
        document.getElementById('infolng').value =
place.geometry.location.lng();
      } else {
        bounds.extend(place.geometry.location);
```

```
      }
    });
    map.fitBounds(bounds);
  });

function updateMarkerStatus(str) {
  document.getElementById('markerStatus').innerHTML = str;
}



function updateMarkerPosition(latLng) {
  var objlat = JSON.parse(latLng.lat(),);
  var objlng = JSON.parse(latLng.lng(),);
/*   document.getElementById('info').innerHTML = [
    latLng.lat(),
    latLng.lng()
  ]; */
document.getElementById("infolat").innerHTML = objlat;
document.getElementById("infolng").innerHTML = objlng;
  console.log(latLng.toJSON());

 $('input[name=infolat]').val(objlat);
 $('input[name=infolng]').val(objlng);

}
/* Newlat.push(UpdateMarkerPosition); */


function updateMarkerAddress(str) {
  document.getElementById('address').innerHTML = str;
}

  marker.bindTo('position', map, 'center');
  //marker.bindTo('position', map, 'center');


  map.addListener('dragend', function() {
    var Newlat = map.getCenter().toJSON();
    console.log(Newlat);
  });



  // Add dragging event listeners.
  google.maps.event.addListener(marker, 'dragstart', function() {
    updateMarkerAddress('Dragging...');
  });

  google.maps.event.addListener(marker, 'drag', function() {
    updateMarkerStatus('Dragging...');
    updateMarkerPosition(marker.getPosition());

  });
  google.maps.event.addListener(map, 'drag', function() {
    updateMarkerStatus('Dragging...');
    updateMarkerPosition(map.getCenter());

  });
  google.maps.event.addListener(marker, 'dragend', function() {
```

```
    updateMarkerStatus('Drag ended');
    geocodePosition(marker.getPosition());
  });

//tambahan melakukan pencarian koordinat dengan  HTML5 Geolocation (gps)
    if (navigator.geolocation) {
      navigator.geolocation.getCurrentPosition(
        (position) => {
          const pos = {
            lat: position.coords.latitude,
            lng: position.coords.longitude,
          };

          marker.setPosition(pos);
          map.setCenter(pos);

/*           const gpslat = marker.getCenter(pos).toJSON(); */
                  console.log(pos);
        },
        () => {
          handleLocationError(true, infoWindow, map.getCenter());
        }
      );
    } else {
      // Browser doesn't support Geolocation
      handleLocationError(false, infoWindow, map.getCenter());
    }

};
google.maps.event.addDomListener(window, 'load', initialize);
google.maps.event.addDomListener(window, "load", initialize);
```

Gambar 4. 90 Baris kode dari Mapinput.js

Baris Code AddresSelect.php (controller) :

```
<?php

namespace App\Http\Livewire;

use App\Models\Post_raw;
use App\Models\City;
use App\Models\District;
use App\Models\additionalphotos;
use Livewire\Component;
use Livewire\WithFileUploads;
use Illuminate\Http\Request;

class AddressSelect extends Component
{

       public $city;
       public $districts=[];
       public $district;
    public $des_lok, $des_mas, $lat, $lng, $latitude, $longitude;
    public $prompt;
    public $modelId;
    public  $photos = [];
```

```
    use WithFileUploads;
    protected $listeners = [
        "refreshParent",
        "getLatitudeForInput",
        "getLongitudeForInput",
        "getModelId"
    ];
        protected $messages = [
        'city.required' => 'Kecamatan tidak boleh kosong.',
        'district.required' => 'Kelurahan tidak boleh kosong.',
        'des_lok.required' => 'Deskripsi lokasi tidak boleh kosong.',
        'des_mas.required' => 'Deskripsi masalah tidak boleh kosong.',
        'lat.required' => 'Belum memilih titik lokasi silahkan tekan tombol
Buka Map',
        'lng.required' => 'Belum memilih titik lokasi silahkan tekan tombol
buka Map.',
        'photo' => 'Masukan bukti foto',

    ];
    protected function rules() {
        return [

        'city' =>'required',
        'district' =>'required',
        'des_lok'=>'required',
        'des_mas'=>'required',
        'lat' => 'required',
        'lng' =>'required',
        'photos.*' => 'image|max:1024|required',
        ];
    }

    public function removeImg($index){
        array_splice($this->photos, $index);
    }

    public function refreshParent()
    {
        $this->prompt ="Laporan anda berhasil dibuat, silahkan tunggu admin
untuk mengkonfirmasi laporan anda";
    }

    public function getLatitudeForInput($value)
    {
    if(!is_null($value))
        $this->lat = $value;

    }
    public function getLongitudeForInput($value)
    {
        if(!is_null($value) )
            $this->lng = $value;
    }

    public function render()
    {
       if(!empty($this->city)) {
              $this->districts = District::where('cities_id', $this->city)-
>get();
       }
```

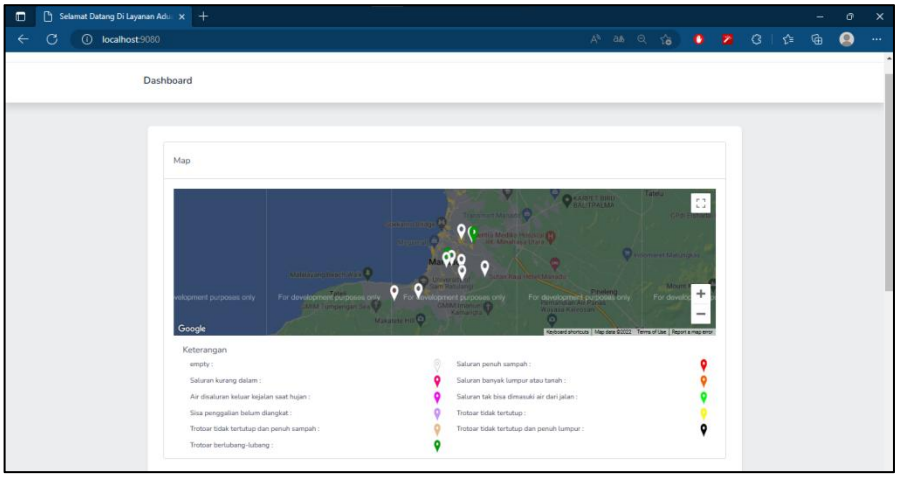
```
        return view('livewire.address-select')
             ->withCities(City::orderBy('kecamatan')->get());
    }

    public function getModelId($modelId){

        $this->modelId = $modelId;
        $model = Post_raw::find($this->modelId);

    }
    public function store()
    {
        $this->validate();

        if ($this->modelId) {
            Post_raw::find($this->modelId);
            $postInstanceId  = $this->modelId;
        }
        else {
            $postInstance = Post_raw::create([
            'city_id' => $this->city,
            'district_id' => $this->district,
            'des_lok' => $this->des_lok,
            'des_mas' => $this->des_mas,
            'lat'=>$this->lat,
            'lng'=>$this->lng,
            'user_id' => auth()->user()->id,
            'status_id'=>1,
            'problem_id'=>1,
            'tipe_id'=>1,
        ]);
            $postInstanceId = $postInstance->id;
        }

        foreach ($this->photos as $photo) {
            $photo->store('additional_photos','public');

            additionalphotos::create([
                'post_raw_id' => $postInstanceId,
                'filename' => $photo->hashName()
            ]);
        }
        $this->emit('refreshParent');
    }
}
```

Gambar 4. 91 Baris kode dari AddressSelect.php sebagai controller.

Pada halaman *dashboard* penggunaan peta berfungsi untuk memperlihatkan seluruh titik koordinat yang merupakan titik laporan-laporan warga, selanjutnya ditambahkan *Listener* didalam peta agar ketika user menekan salah satu marker atau titik didalam peta nantinya akan muncul suatu modal yang merupakan detail laporan

dari laporan tersebut. Dan juga diberikan penjelasan mengenai warna *marker* beserta jenis masalahnya.

List-dashboard.blade.php	View



Baris Code :

```
<div>
 <div class="row justify-content-center">
  <div class="m-2 col">
   <div class="card">
    <div class="card-header">
     <h5 class="card-title pt-3">Map</h5>
    </div>
    <div class="card-body">
     <div id="maps" class="p-2 bd-highlight" style=" width: 100%; height:
300px; float: left;"></div>
    </div>
    <div class="container-fluid">
     <h5 class="ml-4" for="">Keterangan</h5>
     <div class="row m-1 ml-4"> @foreach($problemos as $problemo) <div
class="col-12 col-lg-6 mb-1">
       <label for="">{{$problemo->problem}} :</label> @foreach($fotos as
$foto) @if($problemo->marker_id == $foto->id) <img width="20" height="25"
src="{{ asset('storage/app/public/marker/'.$foto->filename)}}" alt="{{$foto-
>filename}}" class="float-right"> @endif @endforeach
      </div> @endforeach </div>
    </div>
   </div>
  </div>
 </div>
 <div class="table-responsive-md">
  <table class="table">
   <thead>
    <tr>
     <th scope="col"> No <a href="#"
wire:click="sorting('id','asc')">&uarr;</a>
```

```
       <a href="#" wire:click="sorting('id','desc')">&darr;</a>
      </th>
      <th scope="col"> Deskripsi Masalah </th>
      <th scope="col">
       <a href="#" wire:click="sorting('city_id','asc')">&uarr;</a>
       <a href="#" wire:click="sorting('city_id','desc')">&darr;</a>
Kecamatan
      </th>
      <th scope="col">
       <a href="#" wire:click="sorting('district_id','asc')">&uarr;</a>
       <a href="#" wire:click="sorting('district_id','desc')">&darr;</a>
Kelurahan
      </th>
      <th scope="col">
       <a href="#" wire:click="sorting('status_id','asc')">&uarr;</a>
       <a href="#" wire:click="sorting('status_id','desc')">&darr;</a>
Status Laporan
      </th>
      <th scope="col">
       <a href="#" wire:click="sorting('problem_id','asc')">&uarr;</a>
       <a href="#" wire:click="sorting('problem_id','desc')">&darr;</a>
Jenis Permasalahan
      </th>
      <th scope="col">
       <a href="#" wire:click="sorting('tipe_id','asc')">&uarr;</a>
       <a href="#" wire:click="sorting('tipe_id','desc')">&darr;</a> Tipe
Drainase
      </th>
     </tr>
    </thead>
    <tbody> @foreach($lists as $id => $list) <tr class="cursor-pointer"
wire:click.prevent="$emitTo('index', 'open', {{ $id }})">
      <div wire:key="{{ $id }}">
       <th scope="row">{{ $list->id }}</th>
       <td>{{ $list->des_mas }}</td>
       <td>{{ $list->city->kecamatan }}</td>
       <td>{{ $list->district->kelurahan }}</td>
       <td>{{ $list->status->parameter}}</td>
       <td>{{ $list->problem->problem }}</td>
       <td>{{ $list->type->tipe}}</td>
      </div>
     </tr> @endforeach {{$lists->links()}}
    </tbody>
   </table>
   <div class="modal fade" id="detail-post-modal" tabindex="1" role="dialog"
aria-hidden="true"> @livewire('index') </div>
  </div>
 </div> @section('scripts') <script type="text/javascript">
  window.livewire.on('toggleGalaxyFormModal', () => $('#detail-post-
modal').modal('toggle'));
  window.livewire.on('confirmDestroy', event => {
   $('#detail-post-modal').modal('hide');
  })
 </script> @endsection <script async defer
src="https://maps.googleapis.com/maps/api/js?key={{config('services.google.m
aps')}}&libraries=places&callback=initmap"></script>
 <script type="text/javascript">
  setTimeout(function initmap() {
   //Center the map to manado.
   const lat = 1.474830;
```

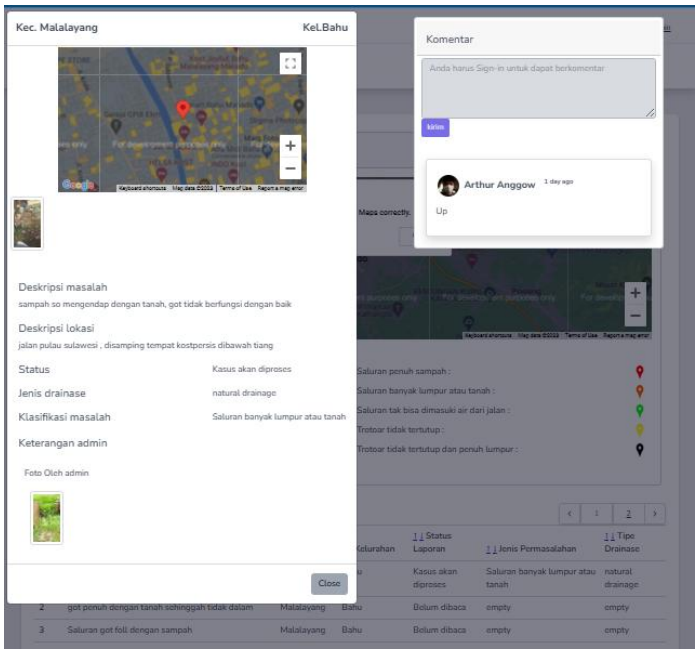
```
   const lng = 124.842079;
   const manado = new google.maps.LatLng(lat, lng);
   const map = new google.maps.Map(document.getElementById("maps"), {
    zoom: 12,
    scrollwheel: true,
    panControl: false,
    zoomControl: true,
    mapTypeId: google.maps.MapTypeId.roadmap,
    center: manado,
    styles: [{
     "featureType": "poi",
     "stylers": [{
      "visibility": "off"
     }]
    }]
   });
   //Retrieve values from JSON to List
   //CHANGES: Multiple marker showed up done.
   //add: problem id as an indicator if the problem is diffrent and changes
the marker colours
   var latJSON = JSON.parse(@this.latitudes);
   var lngJSON = JSON.parse(@this.longitudes);
   //markerjson data berupa nama marker yang telah disesuaikan dengan
permasalahan yang dipilih
   var markerJSON = JSON.parse(@this.markers);
   var postJSON = JSON.parse(@this.problems);
   //text variabel untuk mengarahkan path storage sebuah gambar dari marker
   var text = '/storage/app/public/marker/';
   for (var i = 0; i < latJSON.length; i++) {
    var lats = latJSON[i];
    var lngs = lngJSON[i];
    var posts = postJSON[i];
    var markers = markerJSON[i];
    var latsObject = Object.values(lats);
    var lngsObject = Object.values(lngs);
    var postsObject = Object.values(posts);
    var marksObject = Object.values(markers);
    var marksValue = marksObject[1];
    var latitudeValue = latsObject[0];
    var longitudeValue = lngsObject[0];
    var problemValue = latsObject[1];
    var idValue = postsObject[0];
    // var imagesObject = Object.values(image[problemValue]);
    var images = text + marksValue;
    const marker = new google.maps.Marker({
     position: new google.maps.LatLng(latitudeValue, longitudeValue),
     map,
     icon: images,
     title: String(i)
    });
    attachSecretMessage(marker, i);
   }
  }, 1500);

  function attachSecretMessage(marker, i) {
   google.maps.event.addListener(marker, 'click', function() {
    window.Livewire.emitTo('index', 'open', i);
    $('#detail-post-modal').modal('toggle');
   });
  }
```

```
  // function attachMarker(marker) {
  //   const infowindow = new google.maps.InfoWindow({
  //     content: secretMessage,
  //   });
</script>
```

Gambar 4. 92 Baris kode dari List-Dashboard.blade.php.

Blade selanjutnya adalah Index.blade.php berfungsi untuk menampilkan detail laporan, baik itu peta, gambar, deskripsi masalah, komentar dan lain-lain. Peta tersebut hanya menampilkan satu titik koordinat yang sesuai laporan yang telah dibuat

index.blade.php	View



Baris Code :

```
      <div>
       <div class="modal-dialog modal-xl" style="">
        <div class="modal-content">
         <div class="modal-header">
          <h5 class="modal-title" id="exampleModalLabel">Modal title</h5>
          <button type="button" class="btn-close" data-dismiss="modal" aria-
label="Close"></button>
         </div>
         <div class="row g-5" style="height: auto; padding: 2px;">
          <div class="d-flex align-items-start flex-column bd-highlight top-
50 start-0 translate-middle-y ml-3" style="height: 400px;">
```

```
            <div wire:ignore id="map" class="p-2 bd-highlight" style=" width:
460px; height: 300px; float: left;"></div>
            <div class="flex p-2 bd-highlight"> @foreach($photos as $photo)
<img width="62" height="22"
src="{{ asset('storage/app/public/additional_photos/'.$photo->filename)}}"
alt="post images" class="img-fluid mr-2"> @endforeach </div>
           </div>
           <div class="modal-dialog col-md-10 col-lg-9 order-md-last ">
@foreach ($cities as $city) <span class="text-xl font-bold">Kecamatan
{{ $city->kecamatan }},</span> @endforeach @foreach ($districts as
$district) <span class="text-xl font-bold"> Kelurahan {{ $district-
>kelurahan }}</span> @endforeach @foreach ($posts as $post) @if ($loop-
>first) <div class="modal-dialog col-md-10 col-lg-9 order-md-last ">
             <div class="row">
              <div class="col-sm-6 ">
               <p class="h5">deskripsi masalah</p>
              </div>
              <div class="col-sm-6 ">
               <p>{{ $posts->des_mas }}</p>
              </div>
             </div>
            </div>
            <div class="modal-dialog col-md-10 col-lg-9 order-md-last ">
             <div class="row">
              <div class="col-sm-6 ">
               <p class="h5">deskripsi lokasi</p>
              </div>
              <div class="col-sm-6 ">
               <p>{{ $posts->des_lok }}</p>
              </div>
             </div>
            </div>
            <div class="modal-dialog col-md-10 col-lg-9 order-md-last ">
             <div class="row">
              <div class="col-sm-6 ">
               <p class="h5">status</p>
              </div>
              <div class="col-sm-6 ">
               <p>{{ $posts->status->parameter }}</p>
              </div>
             </div>
            </div>
            <div class="modal-dialog col-md-10 col-lg-9 order-md-last ">
             <div class="row">
              <div class="col-sm-6 ">
               <p class="h5">jenis drainase</p>
              </div>
              <div class="col-sm-6 ">
               <p>{{ $posts->type->tipe}}</p>
              </div>
             </div>
            </div>
            <div class="modal-dialog col-md-10 col-lg-9 order-md-last ">
             <div class="row">
              <div class="col-sm-6 ">
               <p class="h5">klasifikasi masalah</p>
              </div>
              <div class="col-sm-6 ">
               <p>{{ $posts->problem->problem }}</p>
              </div>
```

```
            </div>
           </div> @endif @endforeach
          </div>
         </div>
         <div class="modal-footer">
          <button wire:click="confirmDestroy()" class="btn btn-secondary"
data-dismiss="modal">Close</button>
         </div>
        </div>
       </div>
       <form wire:submit.prevent="addComment">
        <div class="card m-3 w-75 mx-auto"> @if (Route::has('login') ||
Route::has('adminlogin')) @auth
         <!-- komentar bila user telah login -->
         <div class="card-header">
          <h5 class="card-title pt-3">Komentar</h5>
         </div>
         <div class="mt-1 mb-4 mr-2 ml-2">
          <textarea wire:model="comment_input" class="form-control"
id="textAreaExample1" rows="4"></textarea>
          <button type="submit" class="btn btn-sm btn-
primary">kirim</button>
         </div>
         {{ $commentprompt }} @else <fieldset disabled>
          <div class="card-header">
           <h5 class="card-title pt-3">Komentar</h5>
          </div>
          <div class="mt-1 mb-4 mr-2 ml-2 ">
           <textarea placeholder="Anda harus Sign-in untuk dapat
berkomentar" class="form-control " rows="4"></textarea>
           <button type="submit" class="btn btn-sm btn-
primary">kirim</button>
          </div>
         </fieldset> @endauth @endif @foreach($koment as $komentar)
@if($komentar->post_raw_id == $postId) <div class=" card rounded border
shadow p-3 m-3">
          <div class="flex justify-start my-2">
           <p class="font-bold text-lg"> {{$komentar->user->name}}</p>
           <p class="mx-3 py-2 text-xs text-grey-500 font-semibold">
{{\Carbon\Carbon::parse($komentar->created_at)->diffForHumans()}} </p>
          </div>
          <p> {{$komentar->komentar}}</p>
         </div> @endif @endforeach
        </div>
       </form>
       <script type="text/javascript">
        function initialize() {
         window.addEventListener('latitude-loaded', event => {
          var lat = @this.latitude;
          var lng = @this.longitude;
          var latlng = new google.maps.LatLng(lat, lng);
          var crosshairShape = {
           coords: [0, 0, 0, 0],
           type: 'rect'
          };
          var mapCanvas = document.getElementById('map');
          var mapOptions = {
           center: latlng,
           zoom: 17,
           scrollwheel: false,
```

```
            panControl: false,
            zoomControl: true,
            mapTypeId: google.maps.MapTypeId.ROADMAP
           }
           var map = new google.maps.Map(mapCanvas, mapOptions);
           var marker = new google.maps.Marker({
            map: map,
            position: latlng,
           });
          });
         }
        </script>
        <script async defer
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBrFLi96zuuekA3nlI5Tll
Q--4ktUMvoF8&libraries=places&callback=initialize"
type="text/javascript"></script>
```

Gambar 4. 93 Baris kode dari index.blade.php.

Baris Code index.php (controller) :

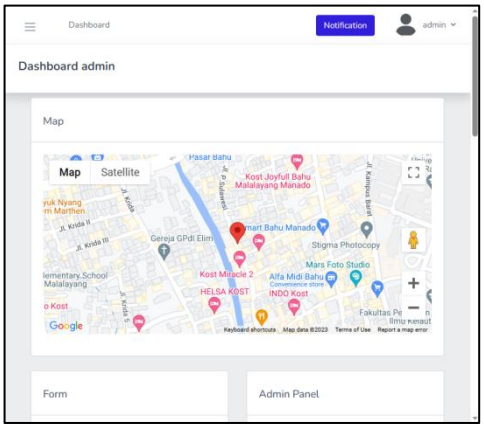
```
<?php

namespace App\Http\Livewire;

use App\Events\ReportSent;
use Livewire\Component;
use App\Models\post_raw;
use App\Models\City;
use App\Models\User;
use App\Models\district;
use App\Models\AdditionalPhotos;
use App\Models\comment;
use Livewire\WithFileUploads;
class Index extends Component
{

    protected $listeners = ['open'=>'loadPosts',
'CommentCreated','refreshParentComponent' => '$refresh'];
    public $cities = [];
    public $districts= [];
    public $posts = [];
    public $postId = [];
    public $reports;
    public $latitude;
    public $longitude;
    public $photos = [];
    public $progresphotoshow = [];
    public $donephotoshow = [];
    //komentar
    public $created_at;
    public $comment_input;
    public $commentprompt;

    public function loadPosts($uid)
    {
        $reports = post_raw::all()->get($uid);
        $this->posts = $reports;
        $this->postId = $reports->id;
```

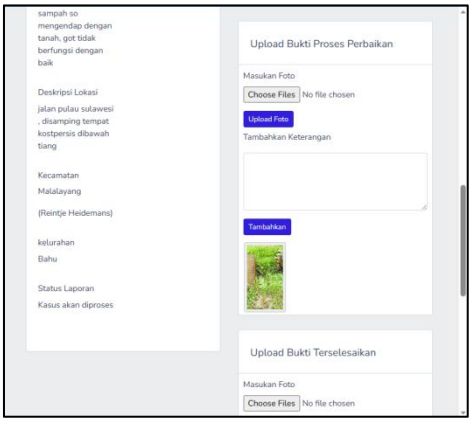
```
        $this->dispatchBrowserEvent('latitude-loaded', ['alat' => $this-
>latitude = $reports->lat]);
        $this->dispatchBrowserEvent('longitude-loaded', ['alng' => $this-
>longitude = $reports->lng]);

        $this->photos = AdditionalPhotos::where('post_raw_id', $reports-
>id)->get();
        $this->donephotoshow = additionalphotos::where('post_raw_id',
$reports->done_id)->get();
        $this->progresphotoshow = additionalphotos::where('post_raw_id',
$reports->prog_id)->get();
        // binding Kecamatan dan Kelurahan berdasarkan post yang dipilih
user
        $this->cities = City::where('id', $reports->city_id)->get();
        $this->districts = District::where('id', $reports->district_id)-
>get();
       $this->emit('toggleGalaxyFormModal');
        $this->emit('confirmDestroy');
        }

    protected function rules()
    {
        return[

            'comment_input' =>'',
        ];
    }

    public function CommentCreated()
    {
        $this->commentprompt = "Komentar berhasil terkirim";
    }

    public function confirmDestroy()
    {
       $this->emit('modalDestroy');
    }

    public function addComment()
    {
        Comment::create([
            'komentar' => $this->comment_input,
            'user_id'=> auth()->user()->id,
            'post_raw_id' => $this->posts->id,
        ]);

        $this->emit('CommentCreated');
        $this->reset('comment_input');
    }

    public function render()
    {

        return view('livewire.index', [
            'koment'=> comment::all()
        ]);

    }
}
```

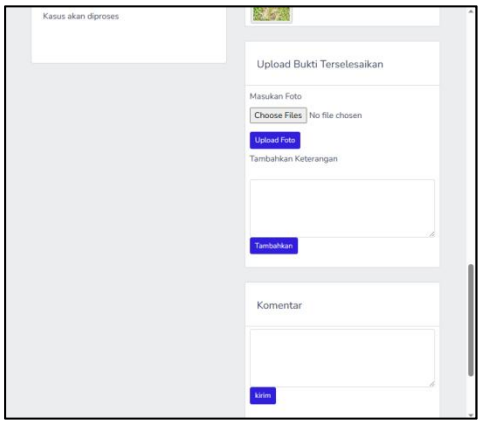
Gambar 4. 94 Baris kode dari index.blade.php.

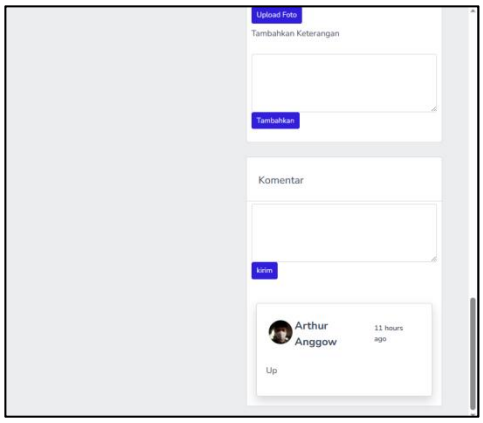
Pada bagian admin, terdapat *blade* khusus kelola-post.blade.php yang didalamnya terdapat panel informasi, *panel* admin, *panel* komentar dan *panel* peta yang lebih lebar. Fitur yang terdapat dalam peta tidak jauh beda dengan index.blade.php .

Kelola-post.blade.php	View









Baris Code :

```
<div>
 <x-slot name="header">
  <h2 class="font-semibold text-xl text-gray-800 leading-tight">
   {{ __('Dashboard admin') }}
  </h2>
 </x-slot>
 <div class="container mt-2">
  <div class="row justify-content-center">
   <div class="col-md-12">
    <div class="card">
```

```
     <div class="card-header">
      <h5 class="card-title pt-3">Map</h5>
     </div>
     <div class="card-body form-group">
      <div wire:ignore id="map" class="p-2 bd-highlight" style=" width:
100%; height: 300px; float: left;"></div>
     </div>
    </div>
    <div class="row">
     <div class="col-sm-6">
      <div class="card">
       <div class="card-header">
        <h5 class="card-title pt-3">Form</h5>
       </div>
       <div class="card-body form-group">
        <div class="row pb-3">
         <div class="col-sm-6 flex p-2 bd-highlight">
          <p class="h6">Gambar</p> @foreach($photos as $photo) <img
width="62" height="22"
src="{{ asset('storage/app/public/additional_photos/'.$photo->filename)}}"
alt="post images" onclick="openModal();currentSlide({{$loop->iteration}})"
class="img-fluid mr-2" wire:ignore> @endforeach
         </div>
        </div>
        <div class="row pb-3">
         <div class="col-sm-6 ">
          <p class="h6">Tipe Drainase</p>
          <p>{{ $posts->type->tipe }}</p>
         </div>
        </div>
        <div class="row pb-3">
         <div class="col-sm-6 ">
          <p class="h6">Jenis Masalah Drainase</p>
          <p>{{ $posts->problem->problem }}</p>
         </div>
        </div>
        <div class="row pb-3">
         <div class="col-sm-6 ">
          <p class="h6">Deskripsi Masalah</p>
          <p>{{ $posts->des_mas }}</p>
         </div>
        </div>
        <div class="row pb-3">
         <div class="col-sm-6 ">
          <p class="h6">Deskripsi Lokasi</p>
          <p>{{ $posts->des_lok }}</p>
         </div>
        </div>
        <div class="row pb-3">
         <div class="col-sm-6 ">
          <p class="h6">Kecamatan</p>
          <p>{{ $posts->city->kecamatan }}</p>
          <p>({{ $posts->city->camat }})</p>
         </div>
        </div>
        <div class="row pb-3">
         <div class="col-sm-6 ">
          <p class="h6">kelurahan</p>
          <p>{{ $posts->district->kelurahan }}</p>
         </div>
```

```
        </div>
        <div class="row pb-3">
         <div class="col-sm-6 ">
          <p class="h6">Status Laporan</p>
          <p>{{ $posts->status->parameter }}</p>
         </div>
        </div>
       </div>
      </div>
     </div>
     <div class="col-sm-6">
      <form wire:submit.prevent="update">
       <div class="card">
        <div class="card-header">
         <h5 class="card-title pt-3">Admin Panel</h5>
        </div>
        <div class="card-body form-group">
         <div class="row pb-3">
          <div class="col-sm-6 ">
           <p class="h6">Jenis Masalah</p>
           <select name="problem" wire:model="problem" class="mt-1 mb-4 w-
full outline-primary">
            <option value=''>Pilih Jenis Masalah</option>
@foreach( $problems as $id => $problem ) @if($problem->id == 1) <option
class="hidden" value={{ $problem->id }}></option> @else <option
value={{ $problem->id }}>{{ $problem->problem }}</option> @endif @endforeach
           </select>
          </div>
         </div>
         <div class="row pb-3">
          <div class="col-sm-6 ">
           <p class="h6">Tipe Saluran/Drainase</p>
           <select name="type" wire:model="type" class="mt-1 mb-4 w-full
outline-primary">
            <option value=''>Pilih Tipe</option> @foreach($types as $id =>
$type) @if($type->id == 1) <option class="hidden" value={{ $type-
>id }}></option> @else <option value={{ $type->id }}>{{ $type-
>tipe }}</option> @endif @endforeach
           </select>
          </div>
         </div>
         <div class="row pb-3">
          <div class="col-sm-6 ">
           <p class="h6">Status laporan</p>
           <select name="status" wire:model="status" class="mt-1 mb-4 w-full
outline-primary">
            <option value=''>Pilih <s></s>tatus laporan </option>
@foreach($stats as $id => $status) @if($status->id == 1 || $status->id ==
2 ) <option class="hidden" value={{ $status->id }}></option> @else <option
value={{ $status->id }}>{{ $status->parameter }}</option> @endif @endforeach
           </select>
          </div>
         </div>
         <button type="submit" class="btn btn-sm btn-
primary">Submit</button>
         {{ $prompt }}
        </div>
       </div>
      </form>
      <div class="card">
```

```
       <div class="card-header">
        <h5 class="card-title pt-3">Komentar</h5>
       </div>
       <div class="mt-1 mb-4 mr-2 ml-2">
        <textarea wire:model="comment_input" class="form-control"
id="textAreaExample1" rows="4"></textarea>
        <button wire:click="addComment" class="btn btn-sm btn-
primary">kirim</button>
       </div>
       {{ $commentprompt }} @foreach($koment as $komentar) @if($komentar-
>post_raw_id == $postId) <div class=" card rounded border shadow p-3 m-3">
        <div class="flex justify-start my-2">
         <p class="font-bold text-lg"> {{$komentar->user->name}}</p>
         <p class="mx-3 py-2 text-xs text-grey-500 font-semibold">
{{\Carbon\Carbon::parse($komentar->created_at)->diffForHumans()}} </p>
        </div>
        <p> {{$komentar->komentar}}</p>
       </div> @endif @endforeach
      </div>
     </div>
    </div>
   </div>
  </div>
  <div id="ModalGambar" class="moodal">
   <span class="close cursor" onclick="closeModal()">&times;</span>
   <div class="modal-konten"> @foreach ($photos as $photo) <div
class="mySlides">
     <div class="numbertext">{{$loop->iteration}}</div>
     <img src="{{ asset('storage/app/public/additional_photos/'.$photo-
>filename)}}" style="width:100%">
    </div> @endforeach
    <!-- Next/previous controls -->
    <a class="prev" onclick="plusSlides(-1)">&#10094;</a>
    <a class="next" onclick="plusSlides(1)">&#10095;</a>
   </div>
  </div>
 </div>
 <script type="text/javascript">
  function initialize() {
   console.log(@this.longitude);
   var lat = @this.latitude;
   var lng = @this.longitude;
   console.log(@this.latitude);
   var latlng = new google.maps.LatLng(lat, lng);
   var crosshairShape = {
    coords: [0, 0, 0, 0],
    type: 'rect'
   };
   var mapCanvas = document.getElementById('map');
   var mapOptions = {
    center: latlng,
    zoom: 17,
    scrollwheel: false,
    panControl: false,
    zoomControl: true,
    mapTypeId: google.maps.MapTypeId.ROADMAP
   }
   var map = new google.maps.Map(mapCanvas, mapOptions);
   var marker = new google.maps.Marker({
    map: map,
```

```
    position: latlng,
   });
   markers.push(marker);
  }
  document.addEventListener('livewire:load', function() {
   google.maps.event.addDomListener(window, "load", initialize);
  });
 </script>
 <script async defer
src="https://maps.googleapis.com/maps/api/js?key={{config('services.google.k
ey')}}&libraries=places&callback=initialize"
type="text/javascript"></script>
 <script type="text/javascript">
  // Open the Modal
  function openModal() {
   document.getElementById("ModalGambar").style.display = "block";
  }
  // Close the Modal
  function closeModal() {
   document.getElementById("ModalGambar").style.display = "none";
  }
  var slideIndex = 1;
  showSlides(slideIndex);
  // Next/previous controls
  function plusSlides(n) {
   showSlides(slideIndex += n);
  }
  // Thumbnail image controls
  function currentSlide(n) {
   showSlides(slideIndex = n);
  }

  function showSlides(n) {
   var i;
   var slides = document.getElementsByClassName("mySlides");
   var dots = document.getElementsByClassName("demo");
   var captionText = document.getElementById("caption");
   if (n > slides.length) {
    slideIndex = 1
   }
   if (n < 1) {
    slideIndex = slides.length
   }
   for (i = 0; i < slides.length; i++) {
    slides[i].style.display = "none";
   }
   for (i = 0; i < dots.length; i++) {
    dots[i].className = dots[i].className.replace(" active", "");
   }
   slides[slideIndex - 1].style.display = "block";
   dots[slideIndex - 1].className += " active";
   captionText.innerHTML = dots[slideIndex - 1].alt;
  }
 </script>
```

Gambar 4. 95 Baris kode dari kelola-post.blade.php.

Baris Code kelolapost.php (controller) :

```php
<?php

namespace App\Http\Livewire;

use Livewire\Component;
use App\Models\post_raw;
use App\Models\drainaseProblems;
use App\Models\drainaseTypes;
use App\Models\AdditionalPhotos;
use Livewire\WithFileUploads;
use App\Models\Status;
use App\Models\comment;
use Illuminate\Http\Request;

use Illuminate\Session\SessionManager;

class KelolaPost extends Component
{
    public $posts = [];
    public $postId = [];
    public $reports;
    public $latitude;
    public $longitude;
    public $type = [];
    public $tipe = [];
    public $problem = [];
    public $prompt;
    public $photos = [];
    public $status;
    public $status_value = [];
    //komentar
    public $created_at;
    public $comment_input;
    public $commentprompt;
    protected $listeners = [
        "postViewed" => "adminHandle",
        "postUpdated",
        "CommentCreated",
        "showComment" => "Comment",
    ];

    public function mount($id)
    {
        $reports = post_raw::find($id);
        $this->posts = $reports;
        //saat proses update dibutuhkan sebuah nilai. nilainya yaitu
property post Id.
        $this->postId = $reports->id;
        $this->dispatchBrowserEvent("latitude-loaded", [
            "latitude" => ($this->latitude = $this->posts->lat),
        ]);

        $this->dispatchBrowserEvent("longitude-loaded", [
            "longitude" => ($this->longitude = $this->posts->lng),
        ]);
        $this->photos = additionalphotos::where(
            "post_raw_id",
            $reports->id
        )->get();
```

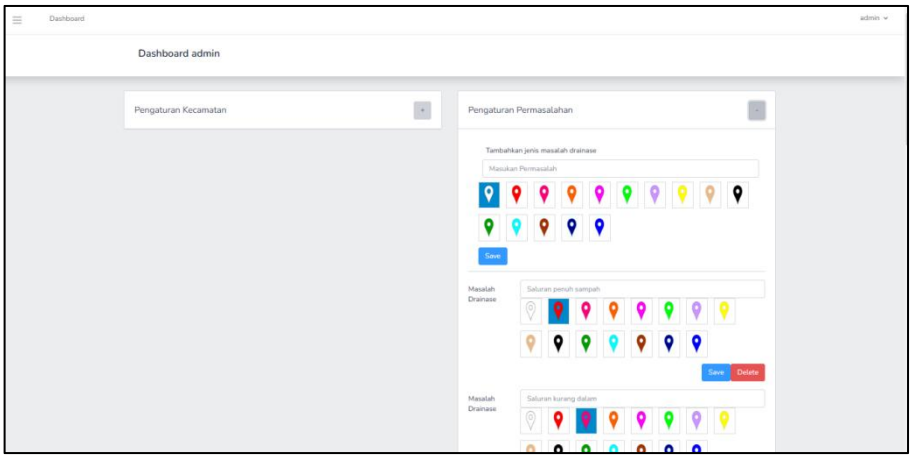
```
        //jika status id = 1(belum dibaca) maka nilai berubah jadi 2 (telah
dibaca)
        if ($this->posts->status_id == 1) {
            $this->posts->status_id = 2;
            $this->posts->save();
        }

    }

    public function postUpdated()
    {
        $this->prompt = "berhasil diperbaharui";
    }

    public function CommentCreated()
    {
        $this->commentprompt = "Komentar berhasil terkirim";
    }

    protected function rules()
    {
        return [
            "problem" => "required",
            "type" => "required",
            "status" => "required",
            "comment_input" => "",
        ];
    }

    public function update()
    {
        $this->validate();

        if ($this->postId) {
            $post = post_raw::find($this->postId);
            $post->update([
                "problem_id" => $this->problem,
                "tipe_id" => $this->type,
                "status_id" => $this->status,
            ]);
        }
        $this->emit("postUpdated");
    }

    public function addComment()
    {
        Comment::create([
            "komentar" => $this->comment_input,
            "user_id" => auth()->user()->id,
            "post_raw_id" => $this->posts->id,
        ]);

        $this->emit("CommentCreated");
        $this->reset("comment_input");
    }

    public function render()
    {
        return view("livewire.kelola-post", [
            "problems" => DrainaseProblems::all(),
```

```
            "types" => DrainaseTypes::all(),
            "stats" => Status::all(),
            "koment" => Comment::all(),
        ])->layout("layouts.admin");
    }
}
```

Gambar 4. 96 Baris kode dari kelola-post.php.

Dan yang terakhir adalah *Blade* Action.blade.php . pada *blade* ini berfungsi sebagai tempat pengaturan bagi admin, khususnya untuk pengaturan untuk mengubah, menambahkan atau pun menghapus jenis masalah beserta marker yang ada dalam aplikasi, disamping itu juga admin dapat mengatur Tipe drainase dan kecamatan. Dalam *blade* ini tidak ada tampilan peta dan deskripsi permasalahan juga deskripsi tipe drainase laporan warga.

Action.blade.php	View



Baris Code :

```
<div>
    <div class="container mt-2">
       <div class="row">
              <div class="col-sm-6">
                     <div class="card">
                            <div class="card-header">
                        <button id="buttonKecamatan" type="button"class="btn
btn-secondary active float-right mt-2" data-toggle="button" aria-
pressed="false" autocomplete="off">+</button>
                        <h5 class="card-title pt-3">Pengaturan
Kecamatan</h5>
                     </div>
```

```
                    <div wire:ignore.self id="kecamatancontent" class="card-
body hiddenclass">
                    @foreach($cities as $id => $city)
                        <fieldset disabled>
                          <div class="form-group row">
                                <label for="inputKecamatan" class="col-sm-4
col-form-label">Kecamatan</label>
                                <div class="col-sm-10">
                                        <input type="text"
id="disabledTextInput.{{$city->id}}" class="form-control col-xs-4"
placeholder="{{$city->kecamatan}}">
                                </div>
                          </div>
                        </fieldset>
                            <form
wire:submit.prevent="updateKecamatan({{$city->id}})">
                                <div class="form-group row">
                                    <label for="input Camat" class="col-sm-4
col-form-label ">Camat</label>
                                    <div class="col-sm-10 input-group-
prepend">
                                            <input type="text"
wire:model.defer="camatinput.{{$city->id}}" class="form-control mb-4 "
placeholder="{{$city->camat}}">
                                            <button type="submit" class="btn
btn-outline-secondary h-60">Ok</button>
                                    </div>
                                </div>
                            </form>
                    @error('updatecamat') <span
class="error">{{ $message }}</span> @enderror
                    @endforeach
                    </div>
                    </div>
             </div>

            <div class="col-sm-6">
                <div class="card">
                    <div class="card-header">
                            <button id="buttonPermasalahan"
type="button"class="btn btn-secondary active float-right mt-2" data-
toggle="button" aria-pressed="false" autocomplete="off">+</button>
                            <h5 class="card-title pt-3">Pengaturan
Permasalahan</h5>
                    </div>
                    <div wire:ignore id="problemcontent" class="card-body
hiddenclass">
                        <form wire:submit.prevent="createproblems">
                        @error('baruproblem') <span
class="error">{{ $message }}</span> @enderror
                            <div wire:ignore id="createproblem" class="
card-header mb-3">
                                <label for="baruproblem" class="col
">Tambahkan jenis masalah drainase</label>
                                <input  type="text" id="baruproblem"
class="form-control mb-2 ml-2" wire:model="baruproblem" placeholder="Masukan
Permasalah">

                                    <div>
                                        @if (session()->has('message'))
```

```
                                            <div class="alert alert-
success">
                                                {{ session('message') }}
                                            </div>
                                        @endif
                                    </div>

                                    <select id="my-select" class="image-
picker" wire:model="optionmarkerbaru" data-container="#createproblem" >
                                        @foreach($Markers as $marker)
                                        <option value="{{$marker->id}}"
data-img-src="{{ asset('storage/marker/'.$marker->filename)}}">{{$marker-
>id}}</option>
                                        @endforeach
                                    </select>
                                @error('optionmarkerbaru') <span
class="error">{{ $message }}</span> @enderror
                                    <button type="submit" class="btn btn-info
">Save</button>
                                </div>
                            </form>
                            @foreach($problems as $problem)
                            <form>
                                <div wire:ignore class="form-group row">
                                    <label for="inputproblem" class="col-sm-2
col-form-label">Masalah Drainase</label>
                                    <div wire:ignore class="col-sm-10">
                                        <input wire:ignore type="text"
wire:model.defer="probleminput.{{$problem->id}}" class="form-control"
placeholder="{{$problem->problem}}">
                                        <select wire:ignore class="image-
pickerupd" wire:model.defer="optionmarkerupdate.{{$problem->id}}" data-
container="#createproblem" >
                                        @foreach($Markers as $marker)
                                            <option value="{{$marker->id}}"
data-img-src="{{ asset('storage/marker/'.$marker->filename)}}">{{$marker-
>id}}</option>
                                        @endforeach
                                        </select>
                                    <button
wire:click="hapusproblem({{$problem->id}})" class="btn btn-danger float-
right">Danger</button>
                                    <button
wire:click="updateProblem({{$problem->id}})" class="btn btn-info float-
right">Save</button>
                                    </div>
                                </div>
                            </form>
                            @endforeach
                        </div>
                    </div>
                </div>

                <div class="col-sm-6">
                    <div class="card">
                        <div class="card-header">
                            <button id="buttonTipe" type="button"class="btn btn-
secondary active float-right mt-2" data-toggle="button" aria-pressed="false"
autocomplete="off">+</button>
                            <h5 class="card-title pt-3">Pengaturan Tipe</h5>
```

```
                    </div>
                    <div wire:ignore id="tipecontent" class="card-body
hiddenclass">
                     <form wire:submit.prevent="createtipe">
                        <div wire:ignore id="createtipe" class=" card-header
mb-3">
                            @error('barutipe') <span
class="error">{{ $message }}</span> @enderror
                                <div wire:ignore id="createtipe" class="
card-header mb-3">
                                    <label for="barutipe" class="col
">Tambahkan tipe drainase</label>
                                    <input  type="text" id="barutipe"
class="form-control mb-2 ml-2" wire:model="barutipe" placeholder="Masukan
Tipe ">
                            <button type="submit" class="btn btn-info
">Save</button>
                        </div>
                    </form>
                    @foreach($types as $tipe)
                            <form>
                                <div class="form-group row">
                                    <label for="input tipe" class="col-sm-4
col-form-label ">Tipe drainase</label>
                                    <div class="col-sm-10 input-group-
prepend">
                                        <input type="text"
wire:model.defer="tipeupdate.{{$tipe->id}}" class="form-control mb-4 "
placeholder="{{$tipe->tipe}}">
                                        <button
wire:click="updateTipe({{$tipe->id}})" class="btn btn-outline-secondary h-
60">Ok</button>
                                        <button
wire:click="hapustipe({{$tipe->id}})" class="btn btn-outline-secondary h-
60">delete</button>
                                    </div>
                                </div>
                            </form>
                    @error('updatetipe') <span
class="error">{{ $message }}</span> @enderror
                    @endforeach
                    </div>
                </div>
            </div>

        </div>

        <div class="container">
            <div class="row">
            </div>
        </div>
       </div>
</div>

<script type="text/javascript">

    $("#buttonKecamatan").click(function(){
    if($(this).html() == "+"){
        $(this).html("-");
    }
```

```
    else{
        $(this).html("+");
    }
         $('#kecamatancontent').slideToggle();
    });

    $("#buttonPermasalahan").click(function(){
    if($(this).html() == "+"){
        $(this).html("-");
    }
    else{
        $(this).html("+");
    }
         $('#problemcontent').slideToggle();
    });

    $("#buttonTipe").click(function(){
    if($(this).html() == "+"){
        $(this).html("-");
    }
    else{
        $(this).html("+");
    }
         $('#tipecontent').slideToggle();
    });

     jQuery("select.image-picker").imagepicker({
          selected: function(option){
                     var values = this.val();
                     @this.set('optionmarkerbaru',values);

                }
    });
        document.addEventListener('livewire:load', function () {

        var probJSON =  JSON.parse(@this.problemdis);

           for(var i = 0; i < probJSON.length;i++){
                var probs = probJSON[i];
                var selects = $('.image-pickerupd');
                var selectjson = selects[i];
                var selectobj = Object.values(selectjson);
                var probsObject = Object.values(probs);
                var probsValue = probsObject[0];
                console.log(probsValue);
                $(selectjson).val(probsValue);
                $(selectjson).data('picker').sync_picker_with_select();

                }
        });
               jQuery("select.image-pickerupd").imagepicker({
                selected: function(option){
                    var valuesupdate = this.val();
                    @this.set('optionmarkerupdate',valuesupdate);
                }
             });
</script>
```

Gambar 4. 97 Baris kode dari action.blade.php.

Dalam Action.blade.php lebih berfokus pada fungsi *create, update, delete* untuk memodifikasi tiap konten yang ada. Sebelumnya dalam blade Address-Select.blade.php penulis telah membuat fungsi *create,* pengimplementasiannya juga sama dengan action.blade.php contohnya `<form wire:submit.prevent="store">` kemudian seluruh input *field* dan tombol diahkiri dengan `</form>`. Pada dalam proses *update* dan *delete* bisa juga menggunakan metode yang serupa dalam proses update dan delete, tetapi *livewire* mempunyai cara lain dengan membuat tombol yang mentrigger langsung ke *class* UpdateProblem dan hapusproblem dengan cara menambahkan fungsi *livewire* `wire:click` Kemudian nama *class* yang menjadi target.

Baris Code Action.php (controller) :

```
<?php

namespace App\Http\Livewire;

use Livewire\Component;
use App\Models\City;
use App\Models\drainaseProblems;
use App\Models\DrainaseTypes;
use App\Models\Marker;
use Illuminate\Support\Arr;


class Action extends Component
{
    public $camatinput = [];
    public $updatecamat = [];
    public $updateproblem = [];
    public $createproblem = [];
    public $kecupdate;
    public $probupdate;
    public $problemdelete;
    public $tipemdelete;
    public $inputcamat;
    public $probleminput;
    public $tipearr = [];
    public $tipeupdate;
    public $updatetipe;
    public $barutipe;
    public $baruproblem = [];
    public $optionmarkerbaru = [];
    public $optionmarkerupdate = [];
    public $problemlexp = [];
    public $problemlist = [];
    public $problemlistArr = [];
    public $problemdis = [];

    public function mount()
```

```
    {
        $this->problemlist = drainaseProblems::whereNotIn('id',[1])-
>get('marker_id')->toJSON();
        $this->dispatchBrowserEvent('problem-loaded',[
            'problemdis' => $this->problemdis = $this->problemlist
        ]);

    }

    public function updateKecamatan($idCamat)
    {

        if($idCamat){
        $this->updatecamat = Arr::get($this->camatinput, $idCamat );
            $kecupdate = City::find($idCamat);
            $kecupdate->update([
                'camat' => $this->updatecamat,
            ]);

        }
    }

    public function updateProblem($id)
    {

        if($id){

            $this->updateproblem = Arr::get($this->probleminput, $id);
            $probupdate = drainaseProblems::find($id);
            $probupdate->update([
                'problem' => $this->updateproblem,
                'marker_id' => $this->optionmarkerupdate
            ]);
        }
    }

    public function hapusproblem($id)
    {
        if($id){
            $problemdelete = drainaseProblems::find($id);
            $problemdelete->delete();
        }
        session()->flash('messagedelete', 'Data Berhasil Dihapus.');
        return redirect()->route('action');
    }

    public function hapustipe($id)
    {
        if($id){
            $tipemdelete = DrainaseTypes::find($id);
            $tipemdelete->delete();
        }
        session()->flash('messagedelete', 'Data Berhasil Dihapus.');
        // return redirect()->route('action');
    }

    public function createtipe()
    {
        $validateData = $this->validate(
            ['barutipe'=> 'required'],
```

```
            ['barutipe.required' => 'Tipe kosong']);

        $this->createtipe = DrainaseTypes::create([
            'tipe' => $this->barutipe
        ]);
        session()->flash('message', 'Data Berhasil Dibuat.');
        // return redirect()->route('action');

    }

    public function createproblems()
    {

        $validateData = $this->validate(
            ['baruproblem' => 'required'],
            [
                'optionmarkerbaru.required' => 'Belum memilih marker.',
                'baruproblem.required' => 'Belum mengisi masalah.'
            ],
            ['optionmarkerbaru' => 'required']

        );

        $this->createproblem = drainaseProblems::create([
            'problem' => $this->baruproblem,
            'marker_id' => $this->optionmarkerbaru
        ]);

        session()->flash('message', 'Data Berhasil Dibuat.');
        // return redirect()->route('action');
    }

    public function updateTipe($id)
    {
        if($id){
            $this->tipearr =  Arr::get($this->tipeupdate, $id);
            $updatetipe = DrainaseTypes::find($id);
            $updatetipe->update([
                'tipe' => $this->tipearr,
            ]);
        }
    }

    public function render()
    {

        return view('livewire.action',[
            'cities' => City::all(),
            'problems'=> drainaseProblems::whereNotIn('id',[1])->get(),
            'Markers' => Marker::all(),
            'types' => DrainaseTypes::whereNotIn('id',[1])->get(),
        ]);
    }
}
```

Gambar 4. 98 Baris kode dari action.php.

G) Penerapan Routes dan Middleware pada aplikasi.

Routes merupakan penghubung dan mengatur antara user dengan *framework.* Tanpa disadari *Routes* sangat berperan penting bagi user. Segala pencantuman alamat web aplikasi ini dibangun *Routes* lah menjadi hal yang dicari dalam menangani hal tersebut. Dalam aplikasi ini routes menjalani hal-hal seperti menentukan halaman utama, menjembatani *Google auth*, mengatur *middleware* admin, mengatur pendefinisian link berdasarkan tiap-tiap halaman. Routes atau web.php berada pada *folder routes* didalamnya terdapat web.php, auth.php, channels.php dan lain-lain. Middleware merupakan filter atau memverifikasi apakah yang masuk adalah admin atau user.

Baris Code web.php (routes) :

```
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\ReportController;
use App\Http\Controllers\HomeController;
use App\Http\Controllers\GoogleAuthController;
use App\Http\Livewire\KelolaPost;
use App\Http\Controllers\AdminController;
/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get('/', function () {
    return view('dashboard');
});

Route::get('/adminlogin', function () {
    return view('auth.adminlogin');
});


//laporan
Route::get('/laporan',[ReportController::class, 'render'])-
>middleware(['auth'])->name('laporan');
Route::get('/show',[ReportController::class,'view'])->middleware(['auth'])-
>name('show');

Route::get('/dashboard',[HomeController::class,'render'])-
>name('dashboard');
require __DIR__.'/auth.php';

Route::get('/auth/google/callback',
```

```
[GoogleAuthController::class,'RespondCallback'])->name('google.callback');

//google sign in
Route::get('/auth/google', function () {
    return Socialite::driver('google')->redirect();
  })->name('google.redirect');


Route::prefix('admin')->group(function () {

   Route::get('/kelola',[AdminController::class, 'ShowPost'])-
>middleware(['admin:admin'])->name('posts');
   Route::get('/kelola/{id}',KelolaPost::class)-
>middleware(['admin:admin'])->name('kelola');
   Route::get('/action',[AdminController::class, 'Action'])-
>middleware(['admin:admin'])->name('action');
});
```

Gambar 4. 99 Baris kode dari web.php.

Konfigurasi *middleware* dengan bawaan *Laravel breeze* sebelumnya penulis telah membuat fitur login, yang login ini akan digunakan saja bagi Admin. Hal yang perlu dirubah adalah pada file auth.php yang terdapat pada folder routes. Dengan mengubah link lama dari /login ke /adminlogin.

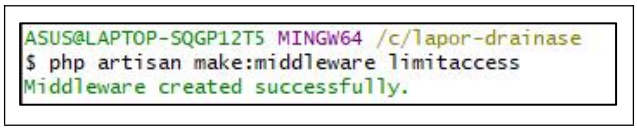
Baris Code web.php (routes) :

```
<?php

use App\Http\Controllers\Auth\AuthenticatedSessionController;
use App\Http\Controllers\Auth\ConfirmablePasswordController;
use App\Http\Controllers\Auth\EmailVerificationNotificationController;
use App\Http\Controllers\Auth\EmailVerificationPromptController;
use App\Http\Controllers\Auth\NewPasswordController;
use App\Http\Controllers\Auth\PasswordResetLinkController;
use App\Http\Controllers\Auth\RegisteredUserController;
use App\Http\Controllers\Auth\VerifyEmailController;
use Illuminate\Support\Facades\Route;

Route::get('/register', [RegisteredUserController::class, 'create'])
                ->middleware('guest')
                ->name('register');

Route::post('/register', [RegisteredUserController::class, 'store'])
                ->middleware('guest');

Route::get('/adminlogin', [AuthenticatedSessionController::class,
'createadmin'])
                ->middleware('guest')
                ->name('adminlogin');

Route::post('/adminlogin', [AuthenticatedSessionController::class, 'store'])
                ->middleware('guest');

Route::post('/logout', [AuthenticatedSessionController::class, 'destroy'])
```

```
                ->middleware('auth')
                    ->name('logout');
```

Gambar 4. 100 Baris kode dari auth.php.

Selanjutnya adalah membuat *middleware* dengan diberi nama limitaccess. Nantinya file ini yang menjadi pemisah antara user dan admin.



Gambar 4. 101 Proses pembuatan middleware limitaccess telah berhasil.

Baris Code limitaccess.php (middleware) :

```
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;

class limitaccess
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure(\Illuminate\Http\Request):
(\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse)  $next
     * @return \Illuminate\Http\Response|\Illuminate\Http\RedirectResponse
     */
    public function handle($request, Closure $next, ...$levels)
    {
        if($request->user() == null)
        {
            return redirect('/');
        }

        if(in_array($request->user()->level,$levels)){
            return $next($request);
        }
            return redirect('/');
    }
}
```

Gambar 4. 102 Baris kode limitaccess.blade.php.

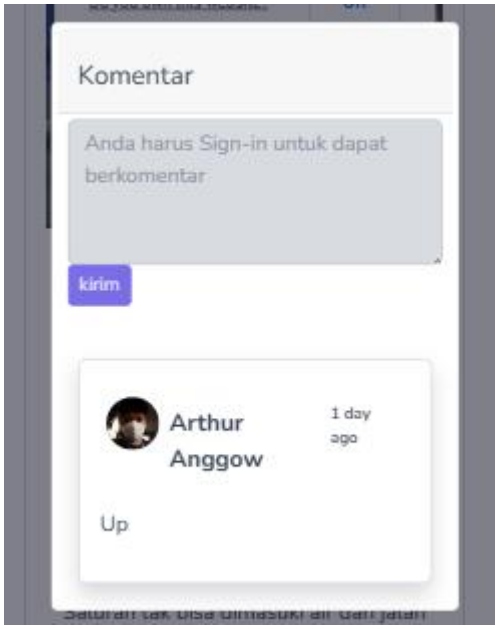
Dalam tabel user, penulis memasukan kolom *level*, kolom *level* ini hanya akan memiliki dua nilai antara “0” dan “1” maksud dari nilai tersebut seperti “admin” dan “user”. Setiap user baru yang terdaftar melalui *google auth* kolomnya *level* akan

diberi “1”. Sehingga middleware disini akan mendeteksi nilai level pada tabel user dan menampilkan tampilan sesuai porsi user. Lakukan pendefinisian ruting *middleware* “limitacces” pada file kernel.php.

H) Penerapan Komentar pada aplikasi.

Panel komentar terdapat pada *blade* “index.blade.php” dan “kelola-post.blade.php”. *Panel* ini dapat membuat komentar (*create*) dan dapat diakses bagi yang dalam keadan aktif atau sementara *login*, selain dari itu komentar hanya dapat dilihat saja.

Index.blade.php	View



Baris Code :

```
<form wire:submit.prevent="addComment" class="col">

 <div class="card m-3 w-75 mx-auto p-1">@if (Route::has('login') ||
Route::has('adminlogin')) @auth <div class="card-header">

   <h5 class="card-title pt-3">Komentar</h5>

  </div>

  <div class="mt-1 mb-4 mr-2 ml-2">

   <textarea wire:model="comment_input" class="form-control"
```

```
id="textAreaExample1" rows="4"></textarea>

   <button type="submit" class="btn btn-sm btn-primary">kirim</button>

  </div>{{ $commentprompt }} @else <fieldset disabled="disabled">

   <div class="card-header">

    <h5 class="card-title pt-3">Komentar</h5>

   </div>

   <div class="mt-1 mb-4 mr-2 ml-2">

    <textarea placeholder="Anda harus Sign-in untuk dapat berkomentar"
class="form-control" rows="4"></textarea>

    <button type="submit" class="btn btn-sm btn-primary">kirim</button>

   </div>

  </fieldset>@endauth @endif @foreach($koment as $komentar) @if($komentar-
>post_raw_id == $postId) <div class="card rounded border shadow p-3 m-3">

   <div class="flex justify-start my-2">

    <img src="{{$komentar->user->avatar}}" alt="avatar" class="rounded-
circle m-1" style="width:40px;height:40px">

    <p class="font-bold mt-2.5 pl-1 text-lg">{{$komentar->user->name}}</p>

    <p class="mx-3 py-2 text-xs text-grey-500 font-
semibold">{{\Carbon\Carbon::parse($komentar->created_at)-
>diffForHumans()}}</p>

   </div>

   <p>{{$komentar->komentar}}</p>

  </div>@endif @endforeach

 </div>

</form>
```

Gambar 4. 103 Baris kode index.blade.php.

```
ASUS@LAPTOP-SQGP12T5 MINGW64 /c/lapor-drainase
$ php artisan make:model comment -m
Model created successfully.
Created Migration: 2022_02_09_055257_create_comments_table
```

Gambar 4. 104 membuat model dan tabel komentar.

Baris Code comment.php (middleware) :

```
<?php

namespace App\Models;
```

```
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class comment extends Model
{
    use HasFactory;

    protected $fillable = ['komentar','user_id','post_raw_id'];

       public function user()
    {
        return $this->belongsTo(User::class, 'user_id');
    }

           public function post_raw()
    {
        return $this->belongsTo(Post_raw::class, 'post_raw_id');
    }
}
```

Gambar 4. 105 Isi kode dari model comment.php.

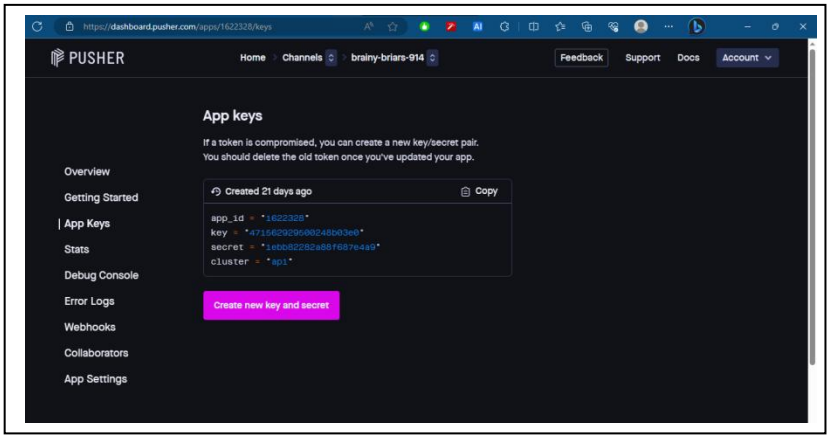
Dalam baris kode model comment.php menjelaskan pertama *protected "*$fillable" kolom-kolom tersebut dapat dimasukan nilai. Tabel *comment* ini akan bergantung baik kepada tabel user dan post_raw .

I) Notifikasi pada admin menggunakan Package Laravel-echo dan Pusher

Pembuatan diawali dengan menginstall *package Laravel-echo* dan *Pusher* dengan menggunakan *NPM.*

```
npm i Laravel-echo

npm i pusher-js
```

Selanjutnya untuk dapat menggunakan *package pusher* diperlukan autentikasi terhadap proyek aplikasi yang akan dipakai pada *website Pusher*. Sehingga diperoleh app_id, *key*, *secret*, dan *cluster.* Untuk autentikasi hanya perlu untuk mendaftar atau langsung *sign-in* dengan *google account* atau *github account*.



Gambar 4. 106 Kode rahasia dari Pusher

Selanjutnya kode tersebut disalin dan *paste* pada file environment (.env) dan file resources/js/bootstrap.js.

```
import Echo from 'laravel-echo';
import Pusher from 'pusher-js';

window.Pusher = Pusher;
window.Echo = new Echo({
    broadcaster: 'pusher',
    key: '471562929500248b03e0',
    cluster: 'ap1',
    forceTLS: true
});
```

Gambar 4. 107 Insialisasi Pusher Key pada Bootstrap.js

Bila *Pusher Key* telah diinisiliasasi pada file Bootstrap.js selanjutnya menjalankan `npm run dev` agar *package* dan dependensi *laravel-echo dan pusher.js* dapat diproses oleh *NPM* dan berjalan pada *javascript* aplikasi. Dan untuk mencoba apakah package tersebut telah berjalan lakukan `window.Echo` pada *console browser.*

Gambar 4. 108 Percobaan memanggil Laravel-echo

Selanjutnya yaitu penerapan skema *boradcasting* dengan membuat *channel*, dan *event*. Terlebih dahulu dengan membuat *event* dengan nama notifevent.php.

```
php artisan make:event notivevent
```

Penginisialisasi *Channel* akan dilakukan pada file *event* yang baru dibuat. Insilisasi *channel* dilakukan pada *component* 'broadcastOn()' dengan nama *channel* yaitu 'notifchannel'.

```
<?php
namespace App\Events;
use Illuminate\Broadcasting\Channel;
use Illuminate\Broadcasting\InteractsWithSockets;
use Illuminate\Broadcasting\PresenceChannel;
use Illuminate\Broadcasting\PrivateChannel;
use Illuminate\Contracts\Broadcasting\ShouldBroadcast;
use Illuminate\Foundation\Events\Dispatchable;
use Illuminate\Queue\SerializesModels;
use app\Models\post_raw;

class notifevent implements ShouldBroadcast
{
    Use         Dispatchable,         InteractsWithSockets,
SerializesModels;

       public function __construct($message){
        $this->message = $message;
       }
    /**
     * Get the channels the event should broadcast on.
     *
     * @return \Illuminate\Broadcasting\Channel|array
     */
    public function broadcastOn()
    {
        return new Channel('notifchannel');
    }
```

Gambar 4. 109 notifevent.php

Pengalamatkan *Event broadcasting channel* juga perlu dilakukan pada agar sistem dapat mengontrol atau mengautentikasian user manakah yang dapat mendengar (*listen*) *channel* yang dipakai. Hal tersebut dilakukan pada file 'routes/channels.php'.

```
<?php

use Illuminate\Support\Facades\Broadcast;
use app\Models\Post_raw;
/*
|--------------------------------------------------------------
-----------------
| Broadcast Channels
|--------------------------------------------------------------
-----------------
|
| Here you may register all of the event broadcasting
channels that your
| application supports. The given channel authorization
callbacks are
| used to check if an authenticated user can listen to the
channel.
|
*/

Broadcast::channel('App.Models.User.{id}',        function
($user, $id) {
    return (int) $user->id === (int) $id;
});

Broadcast::channel('notifchannel', function () {
    return true;
});
```

Gambar 4. 110 Channels.php

Selanjutnya adalah tahapan memanggil event dan menangkap listener pada tampilan notifikasi admin. Membuat tampilan notifadmin dengan *livewire*.

```
php artisan make:livewire notifadmin
```

Pada file *controller* 'notifadmin' untuk dapat menangkap *listener* khususnya pada *component livewire* diperlukan sintaks: `Protected $listeners`

```
Protected $listeners=["echo:channel,.event"=> 'class_baru'];
```

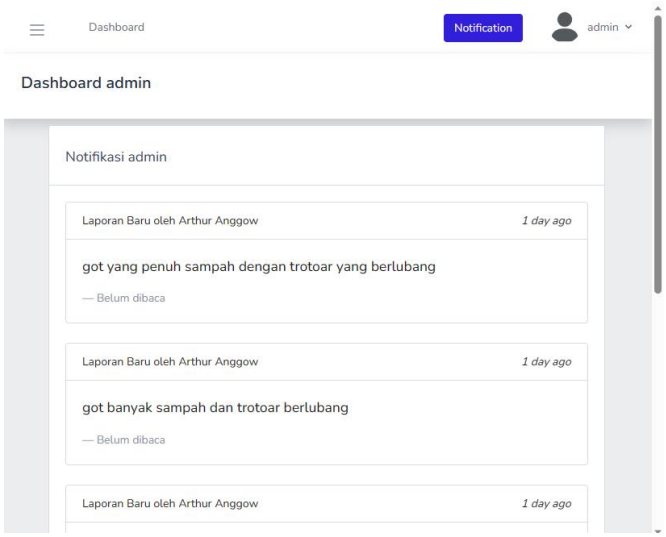
Dalam *class* baru berikan *value* "*true*" agar *callback* dari *channel* dapat ditampilkan. Kemudian data yang tampilkan berdasarkan yang terbaru dengan fungsi 'latest()'. Untuk mengakses fitur ini dapat diakses melalui tombol 'notifikasi' yang akan

langsung diarahkan pada link ‘/admin/notifi’ dan tombol ini terdapat pada ‘header’ ekslusif untuk admin.

```
<?php

namespace App\Http\Livewire;

use Livewire\Component;
use App\Models\post_raw;
use App\Events\notifevent;
use App\Models\User;
use Livewire\WithPagination;

class Notifadmin extends Component
{
protected $listeners = ["echo:notifchannel,.notifevent" =>
'notifyNewOrder'];
public $data=[];
public $reportnewid=[];
public $showNewReportNotification = false;

    public function notifyNewOrder($event)
    {
       $this->showNewReportNotification = true;

    }

    public function render()
    {
        return view('livewire.notifadmin', [
       'reports' => Post_raw::latest()->paginate(5),
       ]);
    }
}
```

Gambar 4. 111 Notifadmin.php

notifadmin.blade.php	View



Baris Code :

```
<div>
@foreach($reports as $id => $report)
<div class="card">
  <a href="/admin/kelola/{{$report->id}}" class="text-decoration-none"style="color:
#1d1c26">
  <div class="card-header">
    Laporan Baru oleh {{$report->user->name}}
    <cite                  class="blocquote-footer                  position-sticky
"style="left:1400px;"title="Source          Title">{{\Carbon\Carbon::parse($report-
>created_at)->diffForHumans()}} </cite>
  </div>
  <div class="card-body">
    <blockquote class="blockquote mb-0">
      <p>{{$report->des_mas}}</p>
      <footer class="blockquote-footer">{{$report->status->parameter}}</footer>
    </blockquote>
  </div>
  </a>
</div>
    @endforeach
{{ $reports->links() }}
</div>
```

Gambar 4. 112 Notifadmin.blade.php

J) Penerapan unggahan gambar dan menampilkan gambar pada laporan

Gambar menjadi komponen terpenting dalam pembuatan laporan tentu. Penggunaan Gambar ini akan diaplikasikan pada *blade* kelola-laporan.blade

(pengelolaan laporan oleh admin).php, index.blade.php (laporan yang telah dibuat) dan address-select.blade.php (halaman pembuatan laporan). Hal awal yang harus dibuat adalah sebuah tabel yang akan menjadi penyimpanan nama file dan juga id laporan.

Gambar 4. 113 Pembuatan model dan tabel AdditionalPhotos.

Selanjutnya adalah melakukan definisi suatu jalur penyimapanan gambar pada "config/filesystem.php" kemudian tinggal perlu menggunakan sintaks laravel seperti pada gambar 4.104.

Gambar 4. 114 Artisan storage:link berguna membuat link simbolis.

K) Fitur menampilkan data laporan.

Pada bagian ini. Fungsi utamanya untuk menampilkan data laporan menjadi diagram batang menggunakan *library javascript* seperti 'Chart.js'. pertama membuat *Component View* dengan nama 'data-show'.

```
Php artisan make:livewire data-show
```

Pada *Component livewire* 'data-show' lebih berfokus pada pengolahan data seperti menghitung jumlah laporan yang telah dibuat, jumlah laporan telah dibaca, dsb. Pengolahan data ini dibantu dengan '*Eloquent ORM'*.

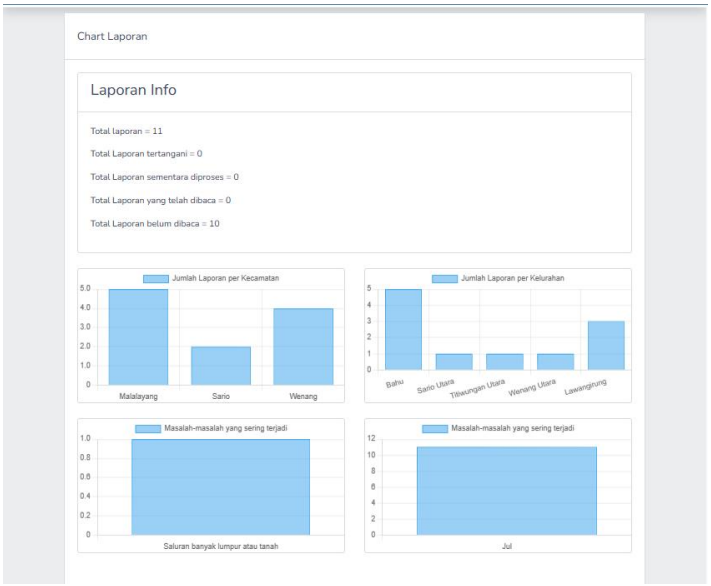
```
<?php

namespace App\Http\Livewire;

use Livewire\Component;
use App\Models\post_raw;
use App\Models\district;
use App\Models\city;
use App\Models\drainaseProblems;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Carbon;

class ShowData extends Component
{
    public $kecamatan = [];
    public $kec;
    public $kel;
    public $kectot = [];
    public $post;
    public $statdone;
    public $statprog;
    public $statunread;
    public $statread;
    public $probcount;
    public $probnama;
    public $postdate;
    public $monthpost;
    public $monthcount;

    public function mount()
    {
        $post = Post_raw::all()->count();
        $this->jumposts = $post;

        $statdone = Post_raw::where("status_id", "=", 5)-
>count();
        $this->statdone = $statdone;

        $statprog = Post_raw::where("status_id", "=", 4)-
>count();
        $this->statprog = $statprog;

        $statunread = Post_raw::where("status_id", "=",
1)->count();
        $this->statunread = $statunread;

        $statread = Post_raw::where("status_id", "=", 2)-
>count();
        $this->statread = $statread;

        $kec = Post_raw::groupBy("city_id")
            ->selectRaw("COUNT(*)as countkec,city_id")
            ->get("countkec", "city_id");

        foreach ($kec as $item) {
            $data["kecamatan"][] = $item->city->kecamatan;
            $data["countkec"][] = $item->countkec;
        }
```

```
        $this->kectot = json_encode($data);

        $kel = Post_raw::groupBy("district_id")
            ->selectRaw("COUNT(*)as countkel,district_id")
            ->get("countkel", "district_id");

        foreach ($kel as $item) {
            $data["kelurahan"][]     =     $item->district-
>kelurahan;
            $data["countkel"][] = $item->countkel;
        }
        $this->keltot = json_encode($data);

        $probcount = Post_raw::groupBy("problem_id")
            ->selectRaw("COUNT(*)as countprob,problem_id")
            ->where("problem_id", ">", 1)
            ->get("countprob", "problem_id");

        foreach ($probcount as $item) {
            $data["countprob"][] = $item->countprob;
            $data["probnama"][] = $item->problem->problem;
        }
        $this->probcount = json_encode($data);

        $monthpost                                          =
Post_raw::groupby("month(created_at)")
            ->selectRaw("COUNT(*)as            monthcount,
month(created_at)")
            ->selectRaw("month(created_at)as created")
            ->get("monthcount",
"month(created_at),created");

        foreach ($monthpost as $item) {
            $data["month"][] = Carbon::createFromDate(
                "M",
                $item->created
            )->format("M");
            $data["monthcount"][] = $item->monthcount;
        }
        $this->month = json_encode($data);
    }
    public function render()
    {
        return view("livewire.show-data")->layoutData([
            "unread" => $this->statunread,
        ]);
    }
}
```

Gambar 4. 115 ShowData.php

Dan untuk tampilan lebih banyak menggunakan javascript nantinya javascript tersebut akan mencari suatu ‘id’ kemudian menapilkan data pada *tag* khusus `<canvas>`. Fitur ini eksklusif untuk admin, dengan link ‘admin/showdata’, dan link tersebut terdapat di menu *sidebar* dengan nama ‘Lihat Data Laporan (Admin)’.

show-data.blade.php	View



Baris Code :

```
<div>
 <div class="">
  <div class="card">
   <div class="card-header">
    <h3>Laporan Info</h3>
   </div>
   <div class="card-body">
    <p> Total laporan = {{$jumposts}}
    </p>
    <p> Total Laporan tertangani = {{$statdone}}
    </p>
    <p> Total Laporan sementara diproses = {{$statprog}}
    </p>
    <p> Total Laporan yang telah dibaca = {{$statread}}
    </p>
    <p> Total Laporan belum dibaca = {{$statunread}}
    </p>
   </div>
  </div>
 </div>
 <div class="row">
  <div class="col-sm-6">
   <div class="card" style="position: relative;">
    <canvas height="40vh" width="80vw" id="totalkecamatan"></canvas>
   </div>
  </div>
  <div class="col-sm-6">
   <div class="card" style="position: relative;">
    <canvas id="totalkelurahan" height="40vh" width="80vw"></canvas>
   </div>
  </div>
  <div class="col-sm-6">
```

```
    <div class="card" style="position: relative;">
     <canvas id="totalproblem" height="40vh" width="80vw"></canvas>
    </div>
   </div>
   <div class="col-sm-6">
    <div class="card" style="position: relative;">
     <canvas id="totaldate" height="40vh" width="80vw"></canvas>
    </div>
   </div>
  </div>
 </div>
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<script type="text/javascript">
 const ctx = document.getElementById('totalkecamatan');
 var kectot = JSON.parse(' < ? php echo $kectot ? > ');
 new Chart(ctx, {
  type: 'bar',
  data: {
   labels: kectot.kecamatan,
   datasets: [{
    label: 'Jumlah Laporan per Kecamatan',
    data: kectot.countkec,
    borderWidth: 1
   }]
  },
  options: {
   scales: {
    y: {
     beginAtZero: true
    }
   }
  }
 });
 const ctr = document.getElementById('totalkelurahan');
 var keltot = JSON.parse(' < ? php echo $keltot ? > ');
 new Chart(ctr, {
  type: 'bar',
  data: {
   labels: keltot.kelurahan,
   datasets: [{
    label: 'Jumlah Laporan per Kelurahan',
    data: keltot.countkel,
    borderWidth: 1
   }]
  },
  options: {
   scales: {
    y: {
     beginAtZero: true
    }
   }
  }
 });
 const cty = document.getElementById('totalproblem');
 var probtot = JSON.parse(' < ? php echo $probcount ? > ');
 new Chart(cty, {
  type: 'bar',
  data: {
   labels: probtot.probnama,
   datasets: [{
    label: 'Masalah-masalah yang sering terjadi',
    data: probtot.countprob,
    borderWidth: 1
```

```
    }]
   },
   options: {
    scales: {
     y: {
      beginAtZero: true
     }
    }
   }
  });
  const ctu = document.getElementById('totaldate');
  var postdate = JSON.parse(' < ? php echo $month ? > ');
  new Chart(ctu, {
   type: 'bar',
   data: {
    labels: postdate.month,
    datasets: [{
     label: 'Masalah-masalah yang sering terjadi',
     data: postdate.monthcount,
     borderWidth: 1
    }]
   },
   options: {
    scales: {
     y: {
      beginAtZero: true
     }
    }
   }
  });
</script>
```

Gambar 4. 116 Notifadmin.blade.php

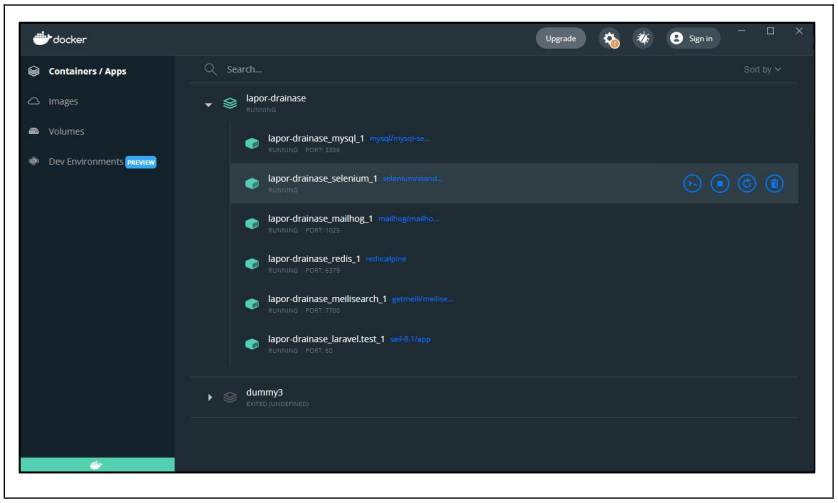
L) Penerapan Docker pada aplikasi.

Untuk menggunakan docker disini penulis melakukan penginstalasi aplikasi kembali dari awal dengan menggunakan perintah laravel yang telah diperbaharui, dimana dalam perintah tersebut disarankan menggunakan depedensi *Laravel Sail* yang adalah *docker*. Pengerjaan baik dari proses pembuatan hingga proses "running" semua dilkukan pada *WSL Ubuntu LTS.*

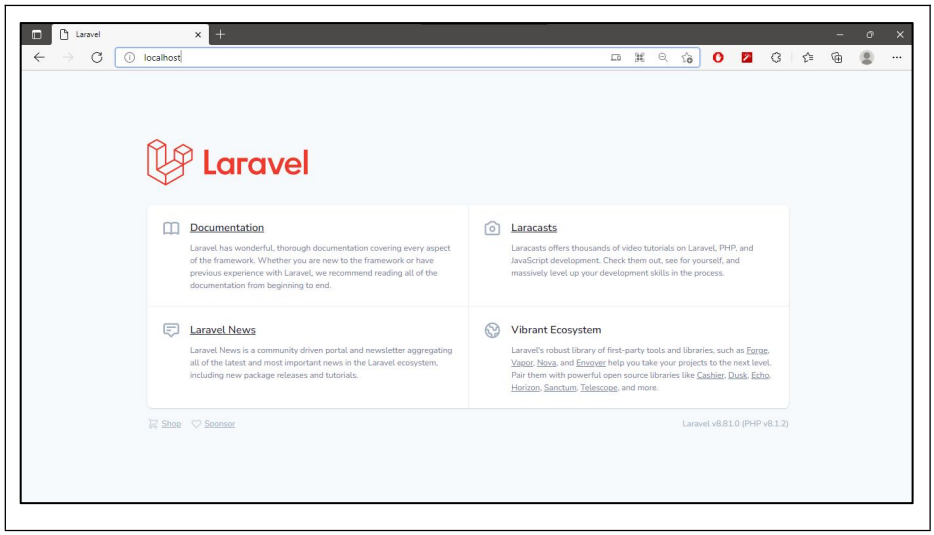
```
Curl -s https://laravel.buld/nama-aplikasi | bash
```

Gambar 4. 117 Proses docker membuat images dan container dengan perintah sail.

Jika proses telah berjalan dengan sukses, selanjutnya memastikan jika *conteiners* telah tersedia atau berjalan, Untuk melihat dan mengontrol *Conteiners* yang telah dibuat sebelumnya, conteiner tersebut dapat dilihat pada program *Docker Desktop*. Dari gambar 4.106 dapat dilihat *conteiner* dengan nama lapor-drainase dengan indikator "Running" bertanda sementara berjalan dan didalamnya tedapat *images-images* dengan indikator "Running" atau sedang berajalan seperti *mysql, selenium, mailhog, redis, meilsearch,* dan terakhir *images* dari aplikasi yang dirancang dengan nama laravel.test. Aplikasi yang telah berjalan dapat dilihat pada web *browser* dengan memasukan dalam URL dan memasukan *localhost* atau dengan menekan didalam menu "*open in browser*" pada *images* aplikasi web (laravel.test).



Gambar 4. 118 Tampilan docker desktop dengan containers lapor-drainase yang telah aktif.



Gambar 4. 119 Tampilan awal projek baru laravel dengan dijalankan melalui docker.

Pada tahap ini merupakan proses memigrasikan seluruh fungsi-fungsi aplikasi yang dilakukan pada file “dummy3” kedalam aplikasi baru docker ini terdapat pada file “lapor-drainase”. Setelah semua telah berhasil dimigrasi secara manual, selanjutnya melakukan pembersihan terhadap berbagai *docker* bawahan dari *laravel* yang tidak akan digunakan seperti *selenium, mailhog, redis, meilsearch.* Untuk *Laravel Sail* tentu sangat berguna ketika dalam karena dapat memahami lingkungan *docker* lebih mudah, tetapi *Laravel Sail* ini memiliki kekurangan karena *Images Laravel sail* hanya berfokus pada *server local* saja. Maka dari itu penulis menggunakan *images NGINX* dan *PHP-FPM* sebagai *web-server.* Docker menyediakan file docker-compose.yml untuk memudahkan pemasangan berbagai Images yang telah tersedia.

Baris Code docker-compose.yml (docker) :

```
# For more information: https://laravel.com/docs/sail
version: '3'
services:
    nginx:
        image: nginx
        depends_on:
            - php-fpm
        networks:
            - sail
        volumes:
            - ./:/var/www/
            - ./docker:/etc/nginx/templates
```

```
        ports:
            - "443:443"
            - "80:80"
        environment:
            - NGINX_HOST=34.80.17.203
            - NGINX_PORT=443
    php-fpm:
        image: cyberduck/php-fpm-laravel:8.1
        networks:
            - sail
        volumes:
            - ./:/var/www
        environment:
            - XDEBUG=TRUE

networks:
    sail:
        driver: bridge
```

Gambar 4. 120 Baris kode docker-compose.yml

Pada Gambar 4.109 didalamnya terdapat 3 *images* yaitu *mysql, Nginx, & php-fpm* kemudian ada juga *networks* dan *volumes*. Untuk menggunakan *Nginx* harus membuat *file template* yang ada pada *folder docker.*

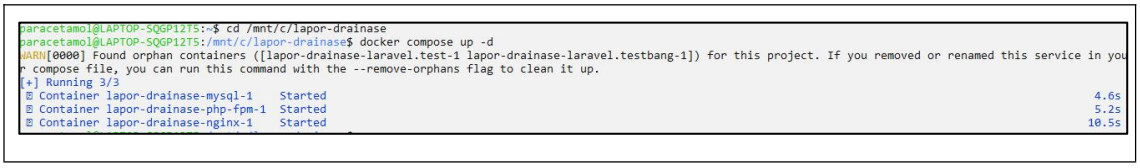
Baris Code default.conf.template (NGINX) :

```
server {
    listen       ${NGINX_PORT} default_server ssl;
    listen       [::]:${NGINX_PORT} ipv6only=on default_server ssl;
    server_name  ${NGINX_HOST} www.lapordrainasemdo.com;

#    return 301 https://${NGINX_HOST}$request_uri;

#    listen       80;
#    listen       [::]:80;
#    server_name  ${NGINX_HOST} www.lapordrainasemdo.com
lapordrainasemdo.com;

    index index.php;

    charset utf-8;

    proxy_read_timeout 300;
    proxy_connect_timeout 300;
    proxy_send_timeout 300;

#    ssl on;
    ssl_certificate /etc/nginx/templates/lapordrainasemdo.crt;
    ssl_certificate_key /etc/nginx/templates/lapordrainasemdo.key;

    ssl_session_timeout 5m;

    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers "HIGH:!aNULL:!MD5 or HIGH:!aNULL:!MD5:!3DES";
    ssl_prefer_server_ciphers on;
```

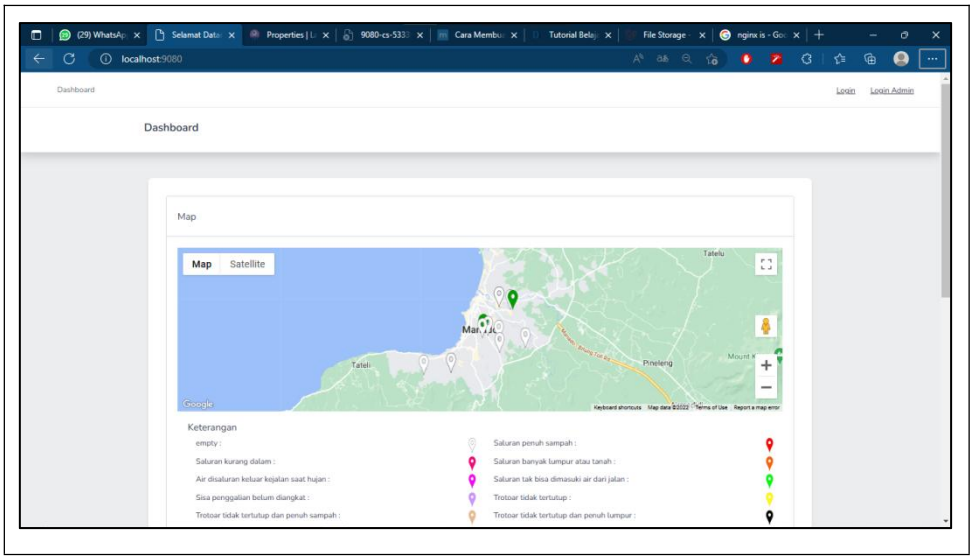
```
    error_page   500 502 503 504  /50x.html;

    client_max_body_size 10M;
    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    location = /favicon.ico { access_log off; log_not_found off; }
    location = /robots.txt  { access_log off; log_not_found off; }

    # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000

    location ~ \.php$ {
       root           /var/www/public;
       fastcgi_pass   php-fpm:9000;
       fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;
       include        fastcgi_params;
    }

   location ~ \.(gif|jpg|png|js)$ {
       root /var/www;
   }
   location ~ \.(css){
       root /var/www/public;
   }

    location ~ /\.ht {
       deny  all;
    }
}
        driver: local
```

Gambar 4. 121 Baris kode default.conf.template

Selanjutnya lakukan percobaan kembali dengan menjalankan perintah `docker compose up -d .`



Gambar 4. 122 Proses docker membuat images dan container dengan perintah docker.



Gambar 4. 123 aplikasi berjalan dengan baik menggunakan docker.

4.1.2 Pembuatan Server Cloud

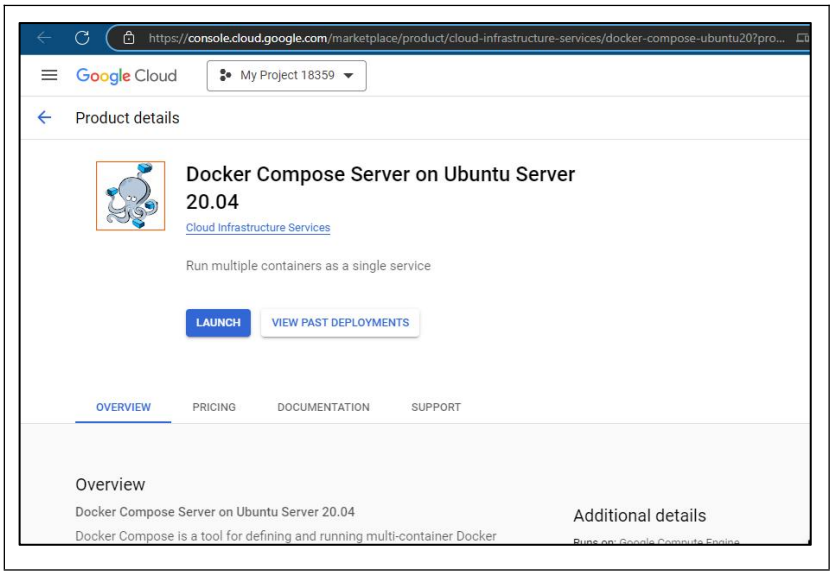
Server yang akan digunakan masih menggunakan salah satu *service* yang ada pada *google cloud console*, dan untuk memasukan seluruh kode aplikasi ke dalam *server* tersebut, Github menjadi solusi dalam menyimpan kode secara *online*.

Pada github ini, akun yang digunakan memiliki nama "Rulesdownload" dan *repositories* yang digunakan memilki nama "tugasakhir". dan *branch* yang dipakai adalah "branchrevisi".

```
ASUS@LAPTOP-SQGP12T5 MINGW64 /c/lapor-drainase/database/seeders (origin)
$ git push origin
fatal: credential-cache unavailable; no unix socket support
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 516 bytes | 516.00 KiB/s, done.
Total 5 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To https://github.com/rulesdownload/tugasakhir.git
   7978dcd..648bf02  origin -> origin
```

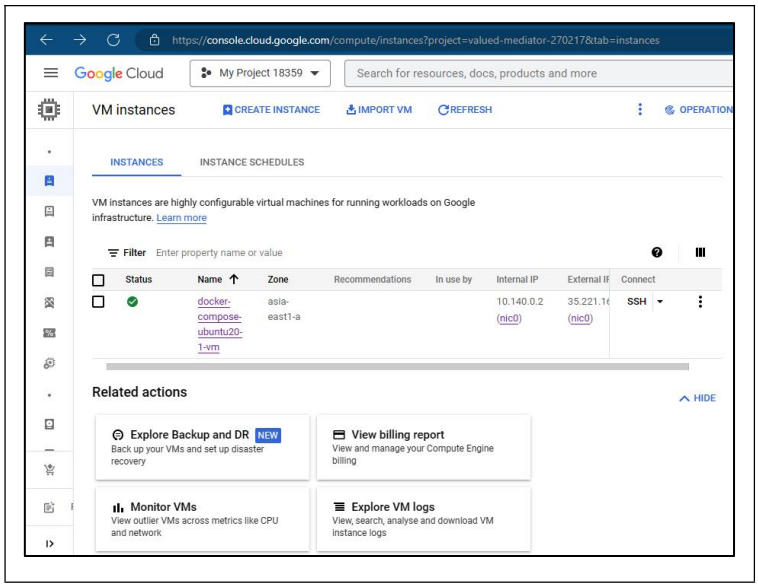
Gambar 4. 124 Proses "Push" ke Repositories.

Selanjutnya adalah proses untuk mengaktifkan layanan "*Compute Engine*" lebih spesifik lagi yaitu "*VM Instances*". Dalam *VM Instance* terdapat produk "*Docker Compose Server on Ubuntu Server 20.04*".



Gambar 4. 125 Salah satu produk pada VM instances.

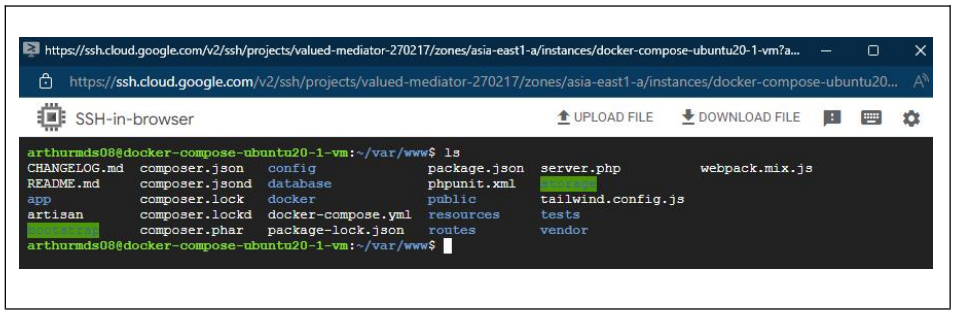
Setelah menekan "*launch*" akan diminta untuk memasukan seperti *zone,* kapasitas penyimapanan yang dibutuhkan dan lain-lain. Selanjutnya setelah berhasil akan nama dan centang berwarna hijau akan muncul dalam tabel. Pada tabel tersebut khususnya pada kolom "External IP", terdapat IP 'https://34.80.17.203'. IP tersebut merupakan output bagi *Virtual Machine*.



Gambar 4. 126 Salah satu produk pada VM instances.

Untuk masuk kedalam *VM* ini akan dibantu dengan *SSH* yang telah tersedia. Proses awal adalah melakukan *clone* dari *github* ke dalam VM yang baru dibuat.

Gambar 4. 127 proses clone ke dalam VM dengan SSH.



Gambar 4. 128 Proses clone berhasil dijalankan.

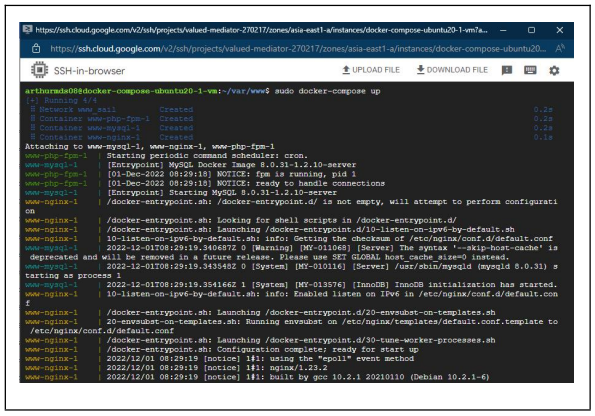
Lakukan pembaharuan docker didalam *VM* dengan perintah :

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin.
```

Jika folder `storage` dan `bootstrap` belum terdefinisikan oleh *VM* lakukan perintah:

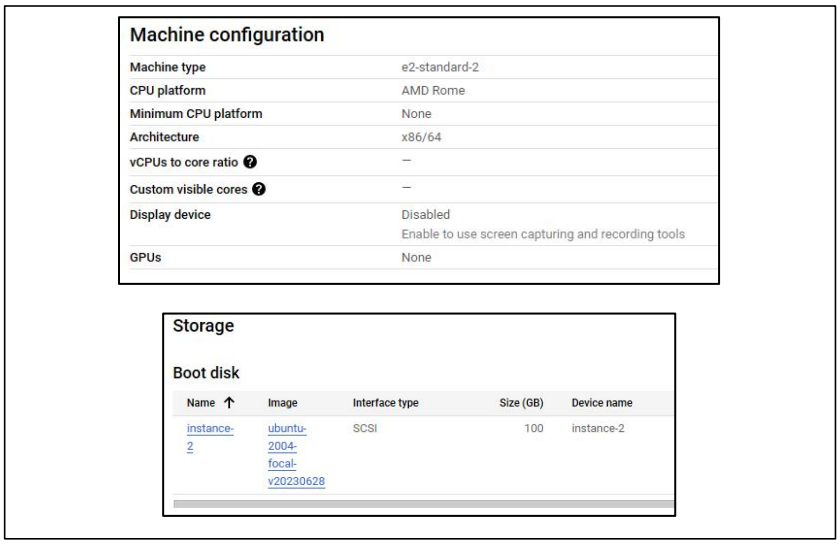
```
"sudo chmod -R 775 storage"

"sudo chmod -R ugo+rw storage"

"sudo chmod -R 775 bootstrap"

"sudo chmod -R ugo+rw bootstrap"
```

Jalankan perintah untuk menghidupkan *docker images* dengan memanggil docker-compose.yml dengan perintah `"sudo docker-compose up"`



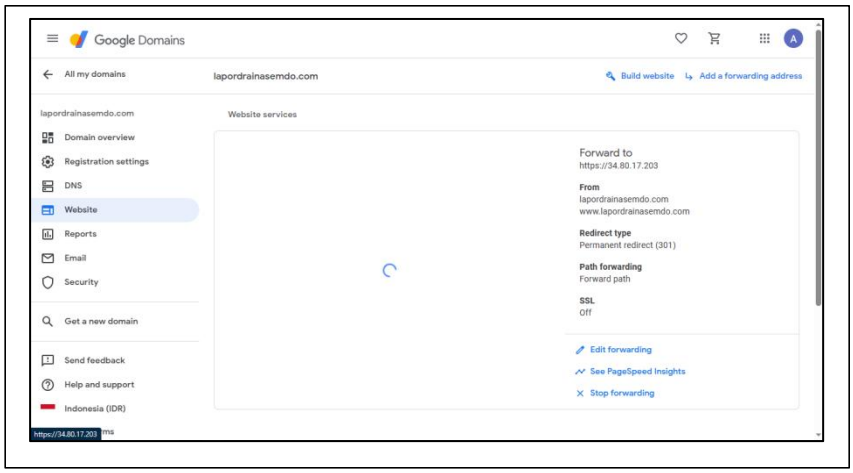
Gambar 4. 129 Proses docker-compose up dalam SSH VM.

Adapun spesifikasi *Virtual Mechine* tertera pada gambar berikut



Gambar 4. 130 Spesifikasi Virtual Machine Cpu dan Storage.

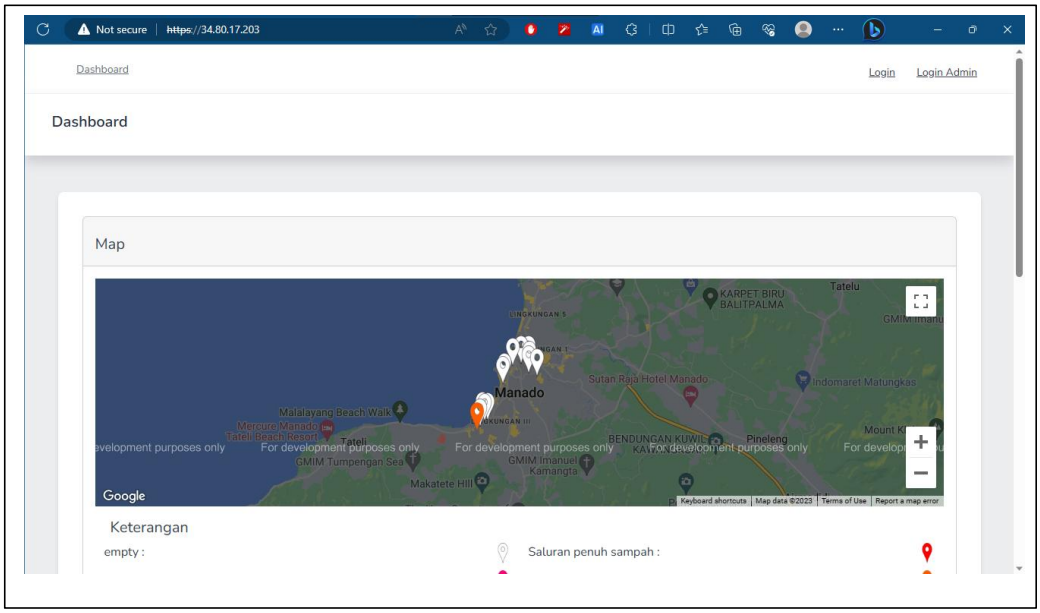
Untuk *DNS* atau *Domain* dibeli pada layanan *Google Domains. Domains* yang dibeli yaitu : ‘www.lapordrainasemdo.com’ / ‘lapordrainasemdo.com’. jika *Domains* tersebut dicari nantinya akan diarahkan pada IP dari *Virtual Machine*.



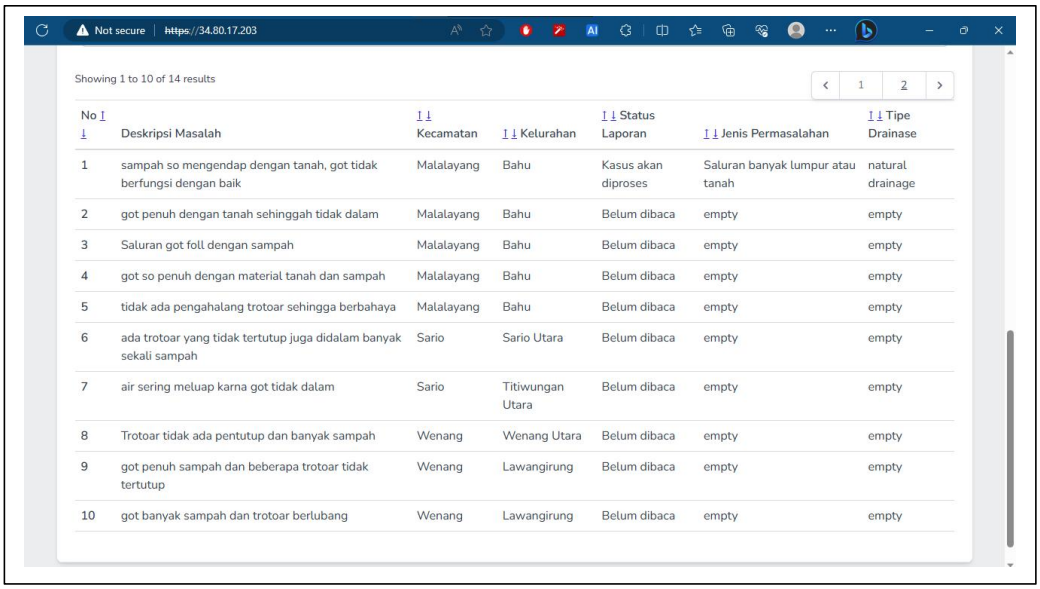
Gambar 4. 131 Layanan Google Domains.

4.2 *Testing* dan Implementasi

4.2.1 Tampilan sebelum sign-in.

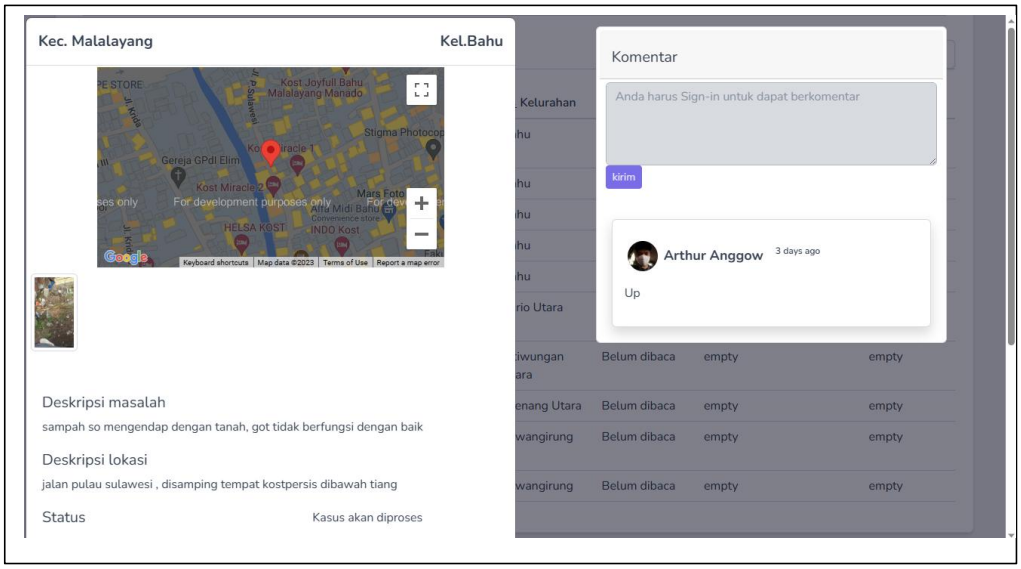


Gambar 4. 132 Halaman utama sebelum sign-in bagian atas.

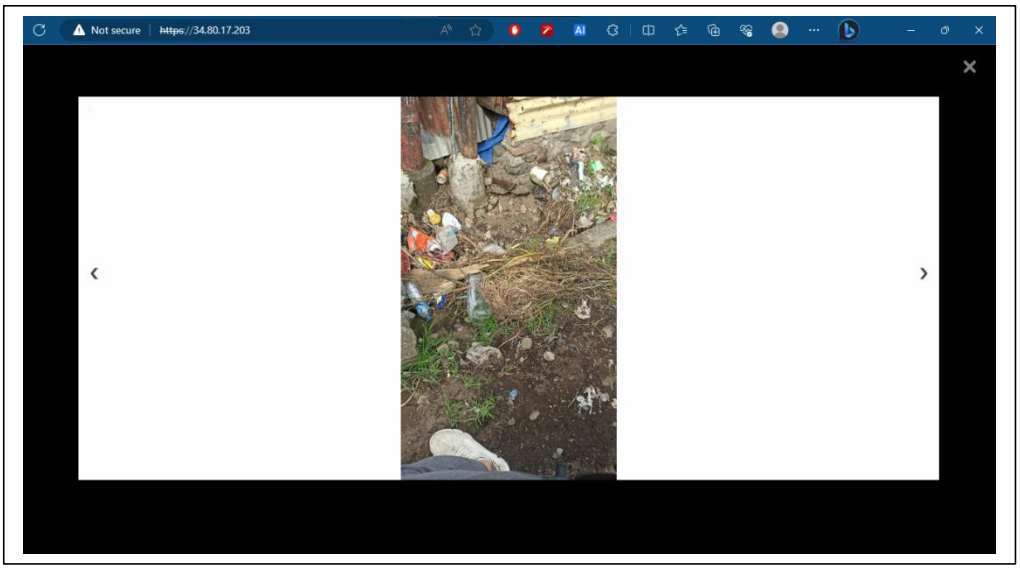


No	Deskripsi Masalah	Kecamatan	Kelurahan	Status Laporan	Jenis Permasalahan	Tipe Drainase
1	sampah so mengendap dengan tanah, got tidak berfungsi dengan baik	Malalayang	Bahu	Kasus akan diproses	Saluran banyak lumpur atau tanah	natural drainage
2	got penuh dengan tanah sehinggah tidak dalam	Malalayang	Bahu	Belum dibaca	empty	empty
3	Saluran got foll dengan sampah	Malalayang	Bahu	Belum dibaca	empty	empty
4	got so penuh dengan material tanah dan sampah	Malalayang	Bahu	Belum dibaca	empty	empty
5	tidak ada pengahalang trotoar sehingga berbahaya	Malalayang	Bahu	Belum dibaca	empty	empty
6	ada trotoar yang tidak tertutup juga didalam banyak sekali sampah	Sario	Sario Utara	Belum dibaca	empty	empty
7	air sering meluap karna got tidak dalam	Sario	Titiwungan Utara	Belum dibaca	empty	empty
8	Trotoar tidak ada pentutup dan banyak sampah	Wenang	Wenang Utara	Belum dibaca	empty	empty
9	got penuh sampah dan beberapa trotoar tidak tertutup	Wenang	Lawangirung	Belum dibaca	empty	empty
10	got banyak sampah dan trotoar berlubang	Wenang	Lawangirung	Belum dibaca	empty	empty

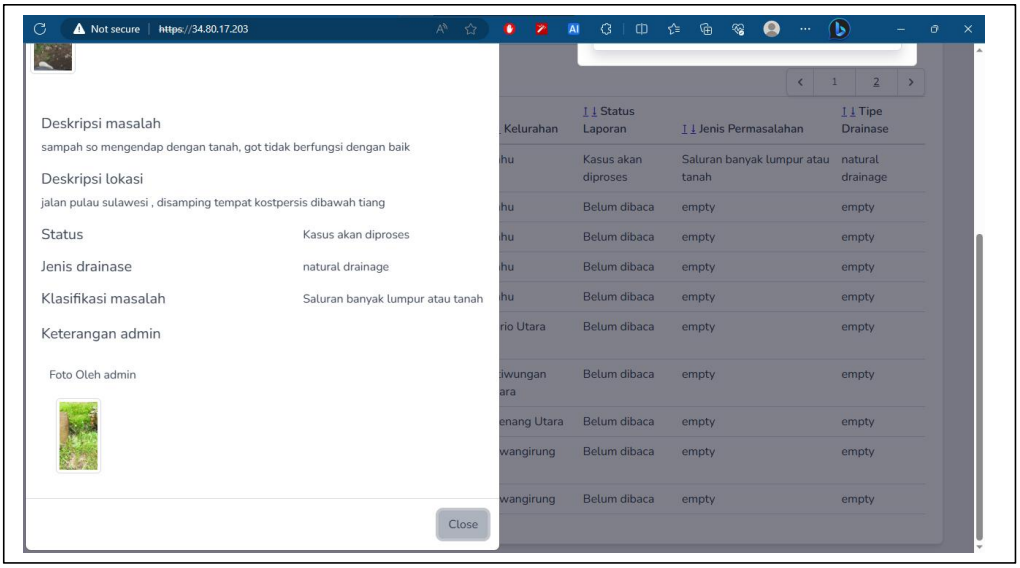
Gambar 4. 133 Halaman utama sebelum sign-in bagian bawah.



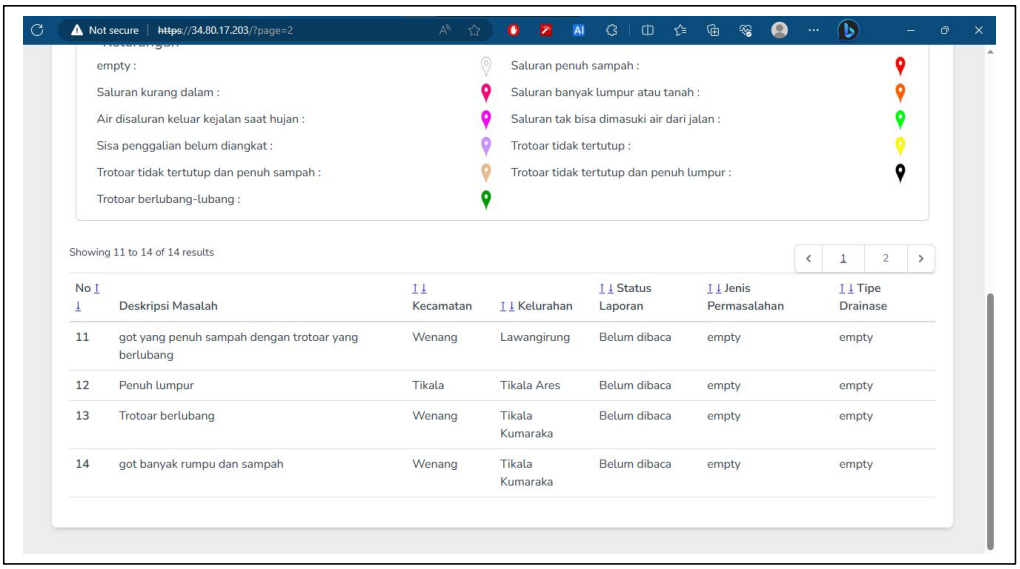
Gambar 4. 134 Modal detail post sebelum sign-in bagian atas.



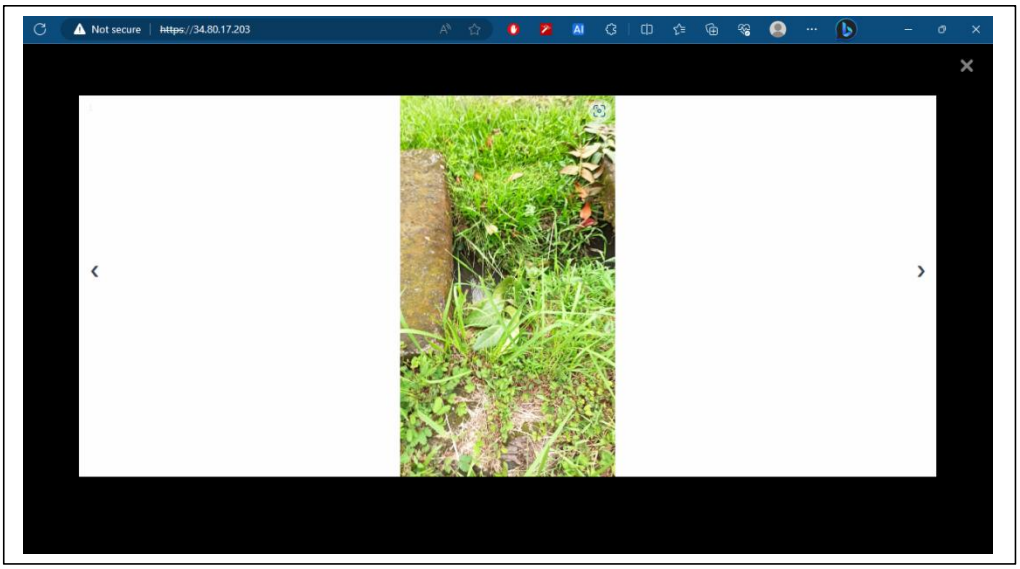
Gambar 4. 135 Modal membuka gambar.



Gambar 4. 136 Modal detail post sebelum sign-in bagian bawah.

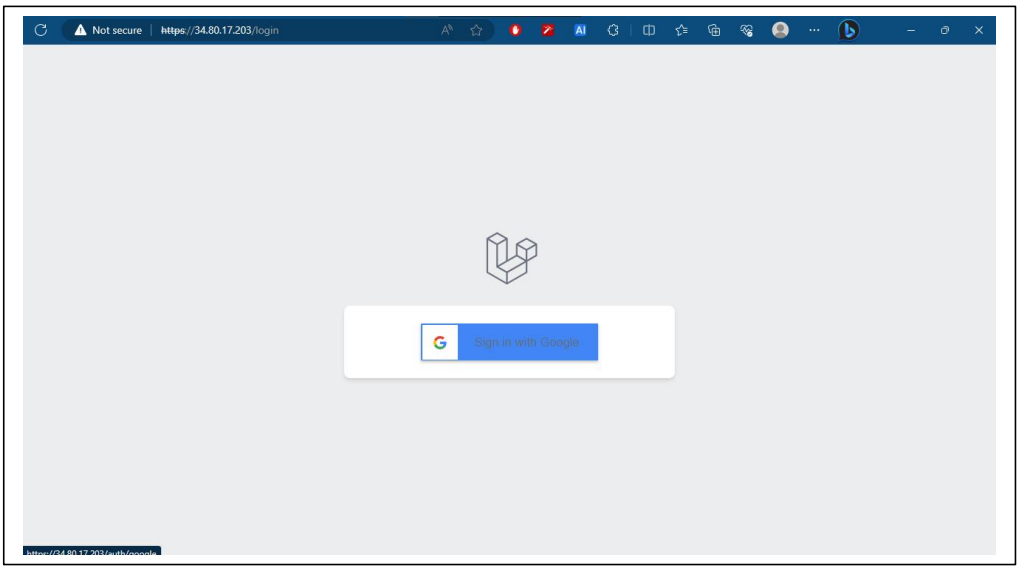


Gambar 4. 137 Halaman utama sebelum sign-in saat berada dihalaman ke-‘2’.

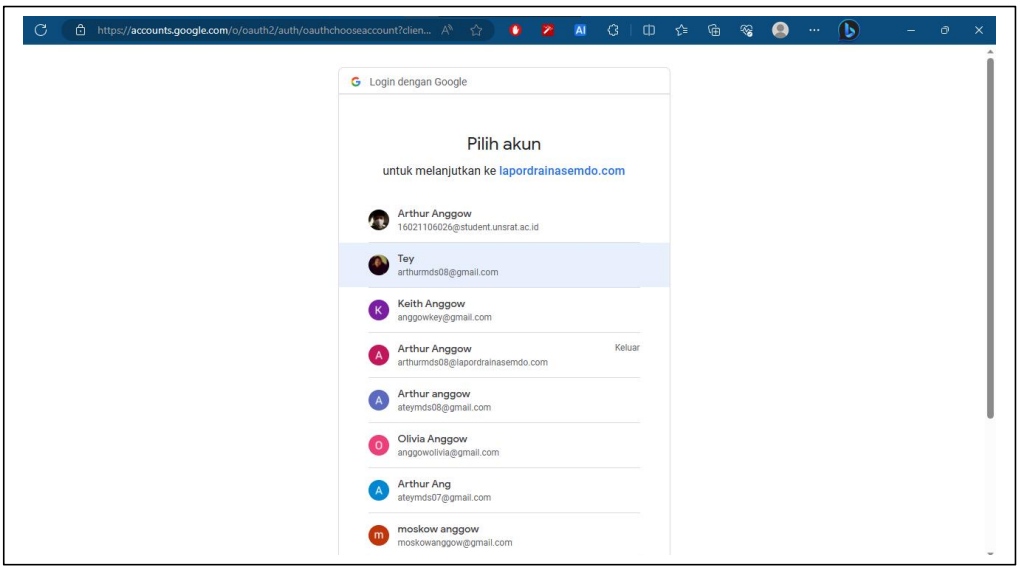


Gambar 4. 138 Modal membuka gambar kiriman Admin.

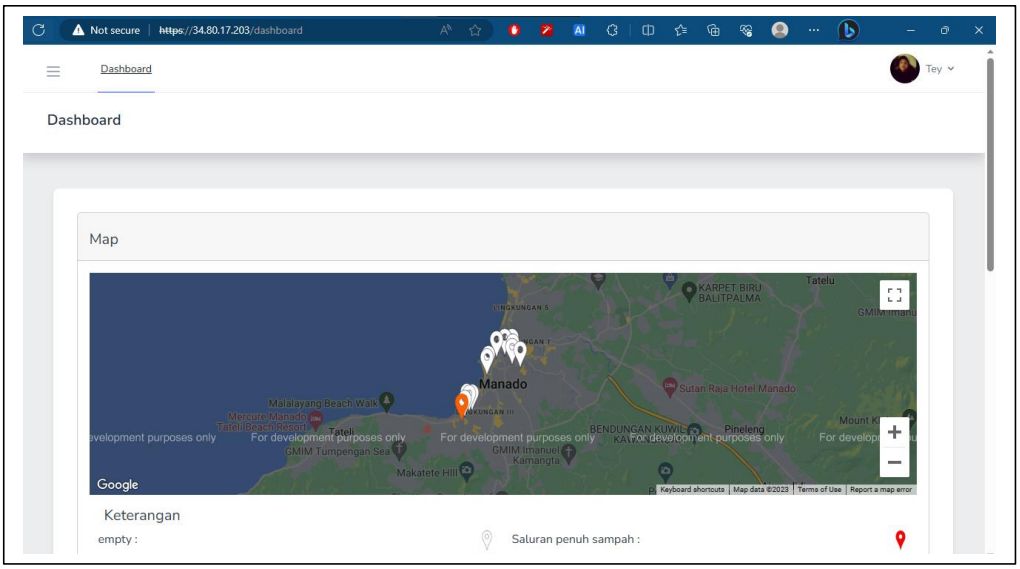
4.2.2 Tampilan sign-in User.



Gambar 4. 139 Tampilan Sign-in.



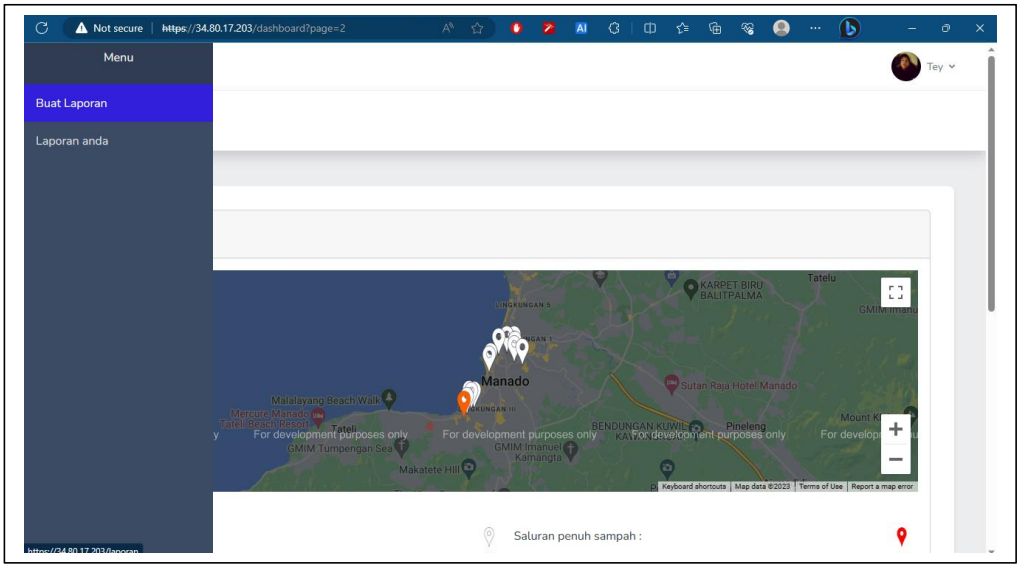
Gambar 4. 140 Tampilan memilih akun google.



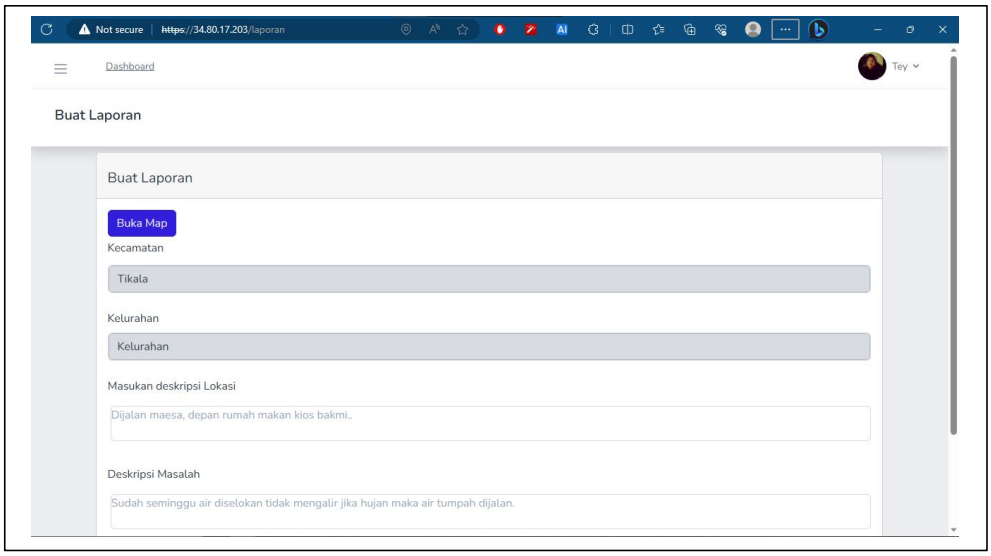
Gambar 4. 141 Tampilan memilih akun google.

Jika telah berhasil *sign-in* pada header atas akan terpampang foto avatar dan nama pengguna seperti pada gambar 4.141.

4.2.3 Fitur membuat laporandan melihat laporan yang telah dibuat.



Gambar 4. 142 Halaman utama pada saat user membuka sidebar.



Gambar 4. 143 Form membuat laporan.

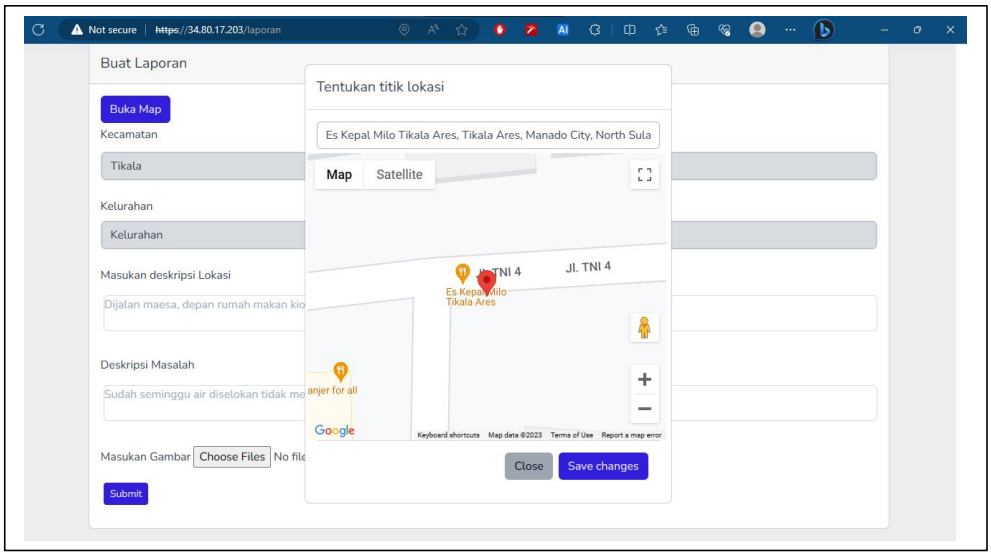
Mempersiapkan data laporan yang akan diisi:

1. Posisinya ialah “Es kepal milo tikala ares”.

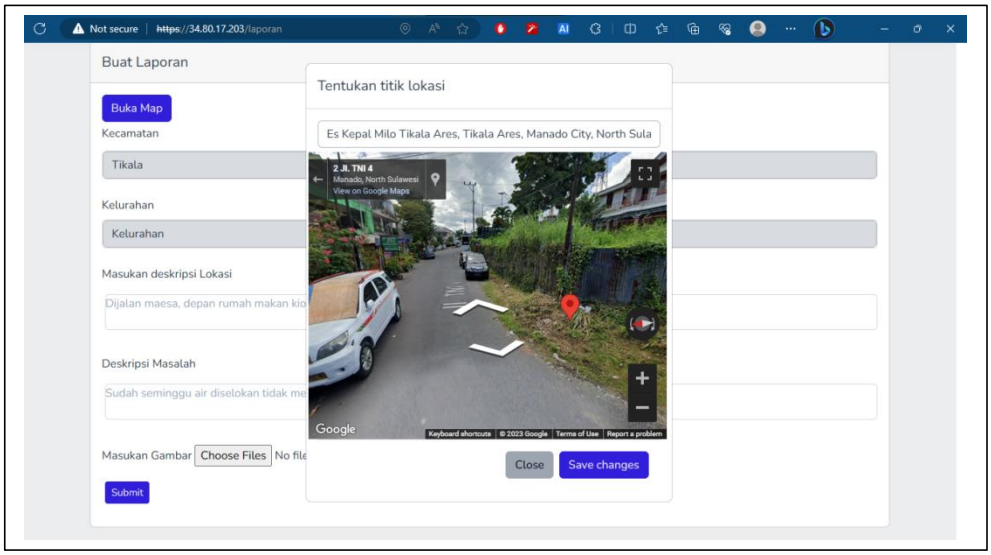
2. Deskripsi lokasi “pertigaan sebelah kiri dari sesudah RRI”

3. Deskripsi masalah “tumpukan galian tanah dan sampah disamping got”.

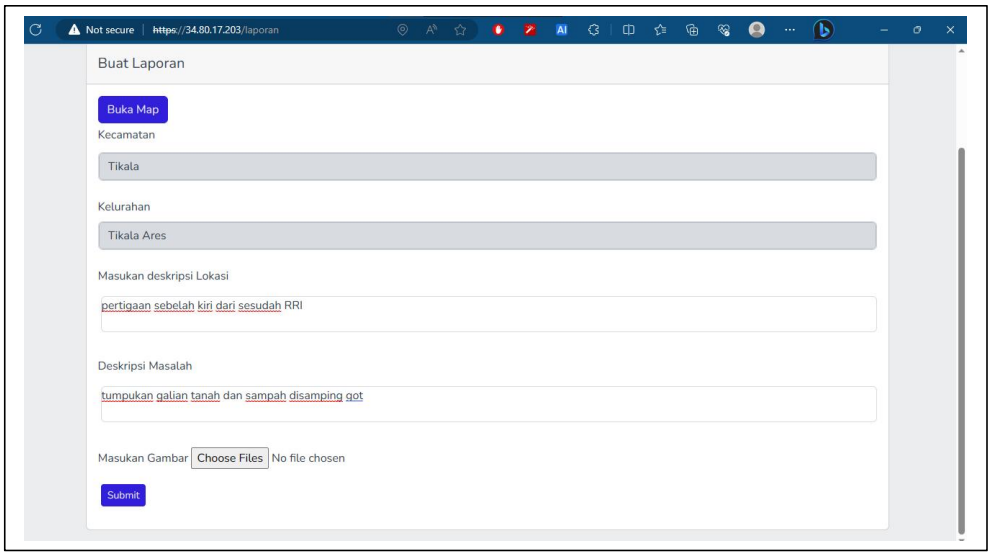
Gambar 4. 144 Foto laporan yang akan diunggah.



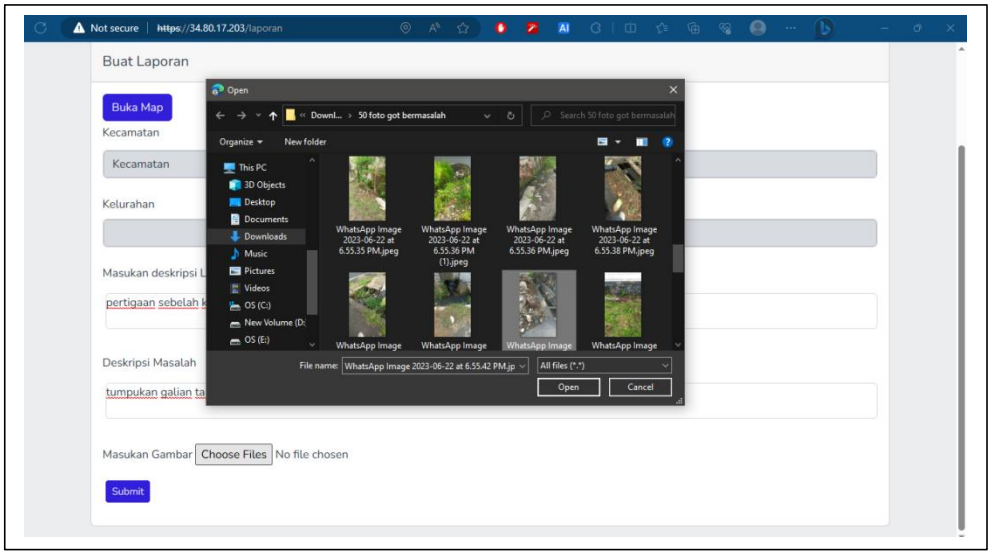
Gambar 4. 145 Memasukan titik lokasi sesuai berdasarkan input.



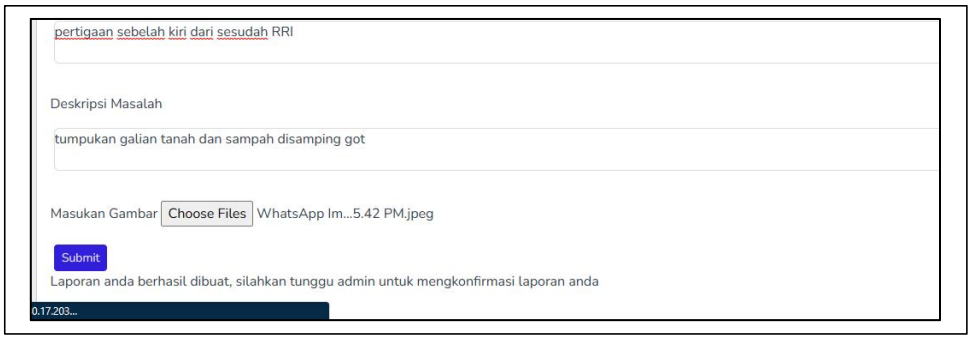
Gambar 4. 146 Input lokasi dengan tampilan Google Street.



Gambar 4. 147 Kecamatan dan kelurahan otomatis terisi, dan deskrispi masalah dan lokasi telah disi.

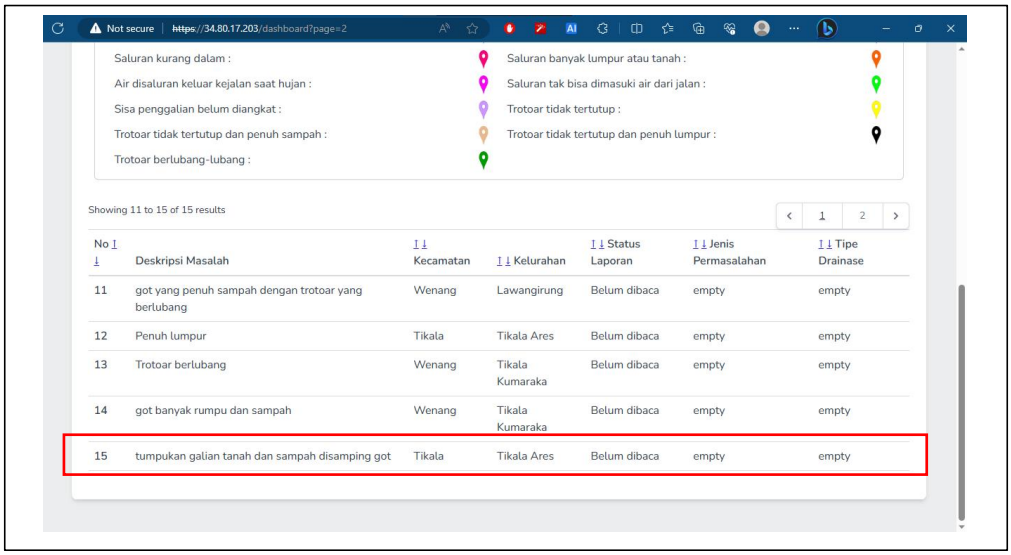


Gambar 4. 148 Tombol “Choose Files” berfungsi untuk menginput foto kedalam server.

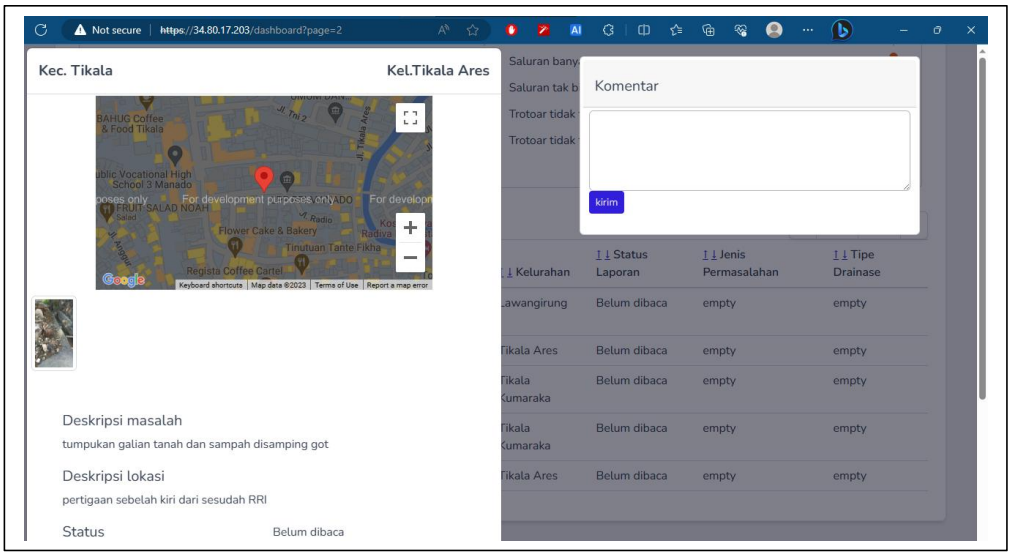


Gambar 4. 149 Setelah menekan submit akan ada pemberitahuan.

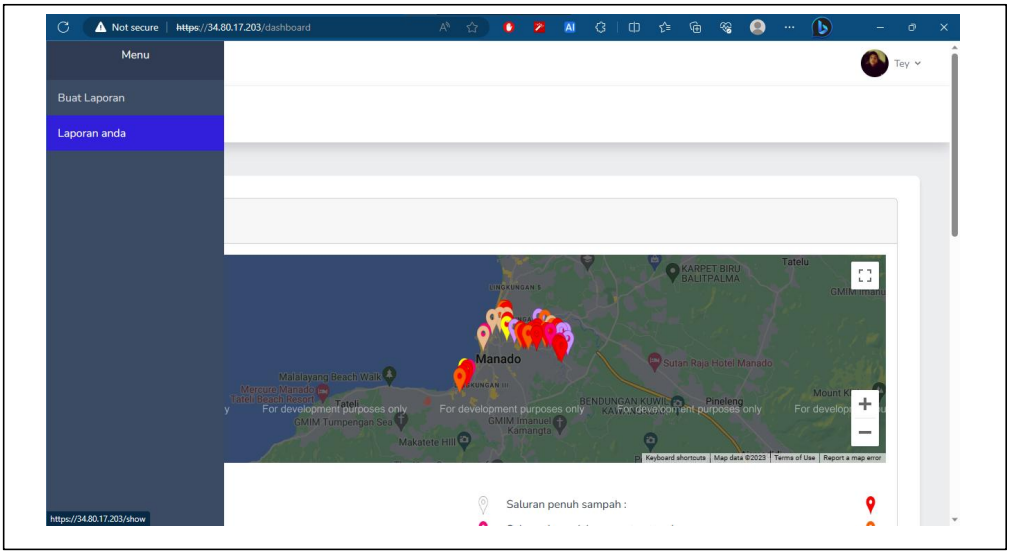
Jika seluruh dalan form telah terisi dengan baik maka tidak akan ada error waktu menekan tombol ‘submit’, bila berhasil akan muncul teks dengan kalimat “Laporan anda berhasil dibuat, silahkan tunggu admin untuk mengkonfirmasi laporan anda”. Setelah itu secara otomatis akan diarahkan ke halaman utama (*dashboard*).



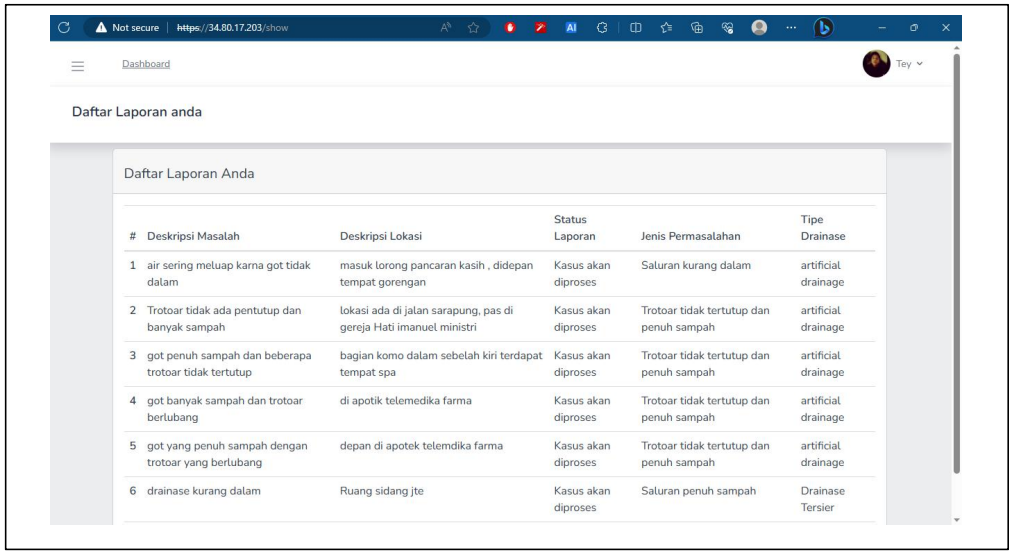
Gambar 4. 150 Halaman Utama dengan laporan yang baru dibuat (ditandai pada kotak merah).



Gambar 4. 151 Halaman detail laporan yang baru saja dibuat.



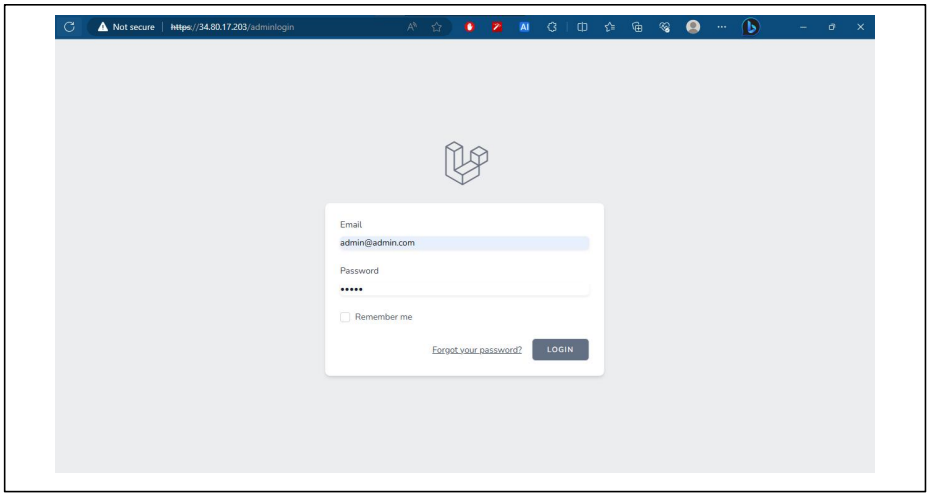
Gambar 4. 152 Menu untuk melihat laporan-laporan yang telah dibuat oleh pengguna.



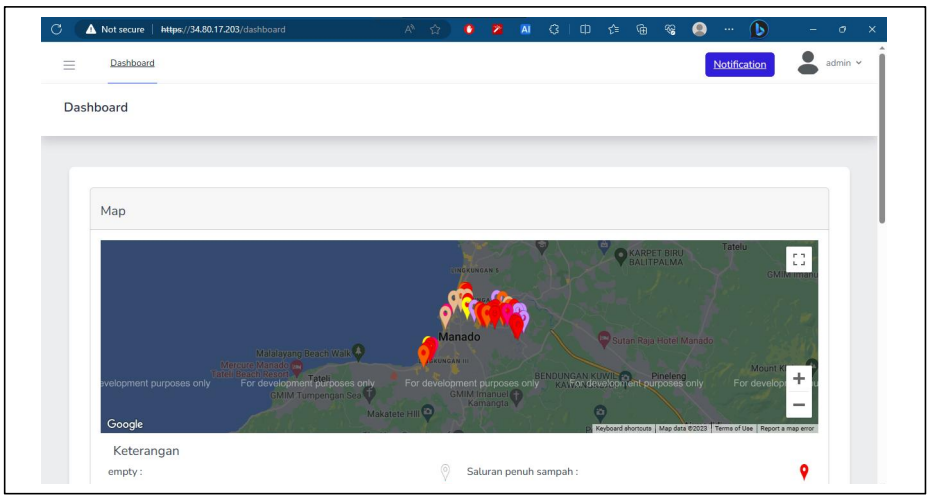
Gambar 4. 153 Tabel yang berisikan laporan-laporan pribadi yang telah dibuat.

4.2.4 Login Admin dan tampilan menu.

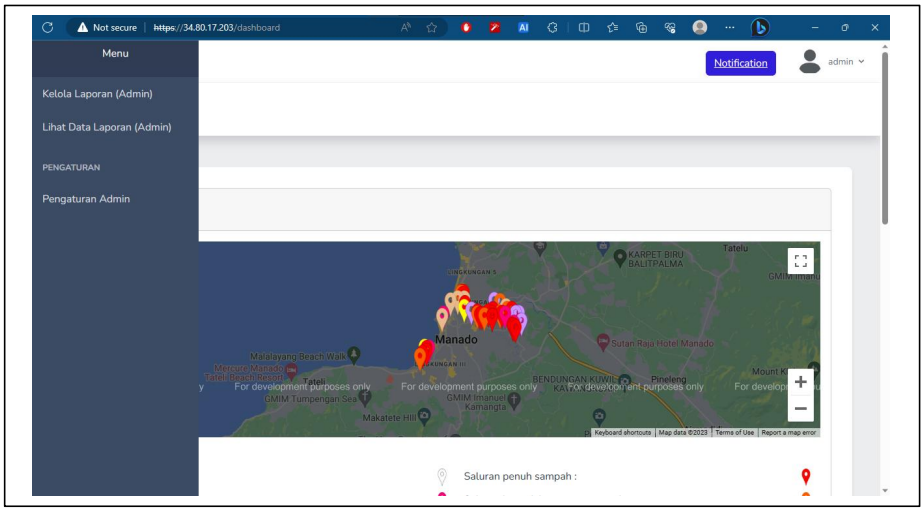
Pada bagian *header* terdapat tombol “login admin”, kemudian akan ke diarahkan pada halaman(https://34.80.17.203/adminlogin) untuk memasukan *email* dan *password* admin. Disini emailnya adalah “admin@admin.com” dan *password*”*admin*”



Gambar 4. 154 Halaman Admin.

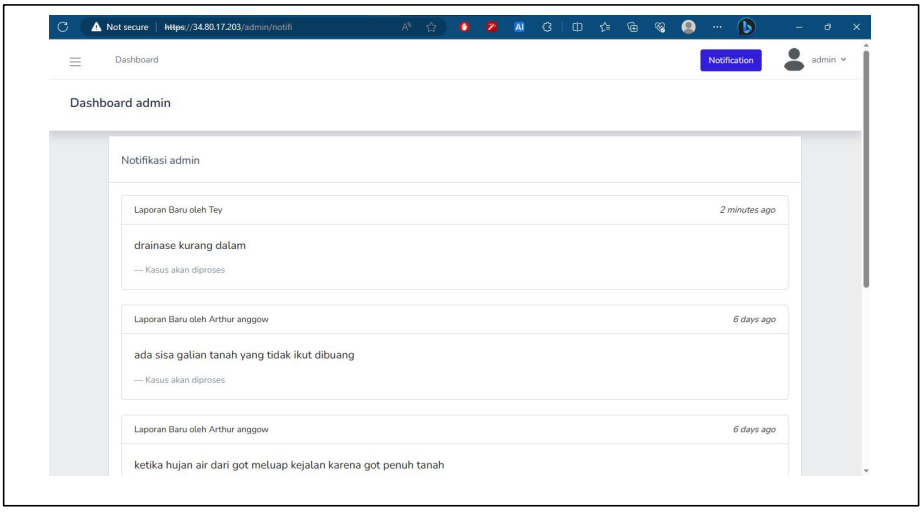


Gambar 4. 155 Dashboard Admin.

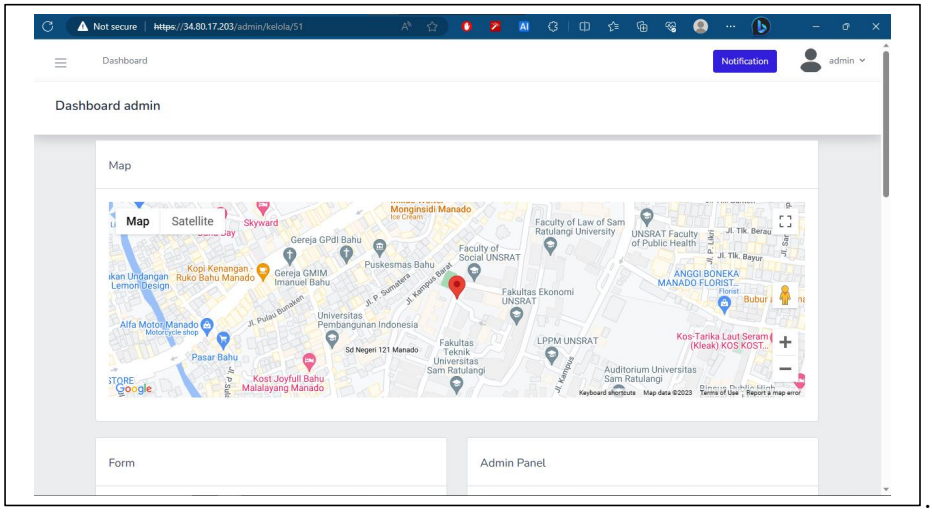


Gambar 4. 156 Menu Sidebar Admin.

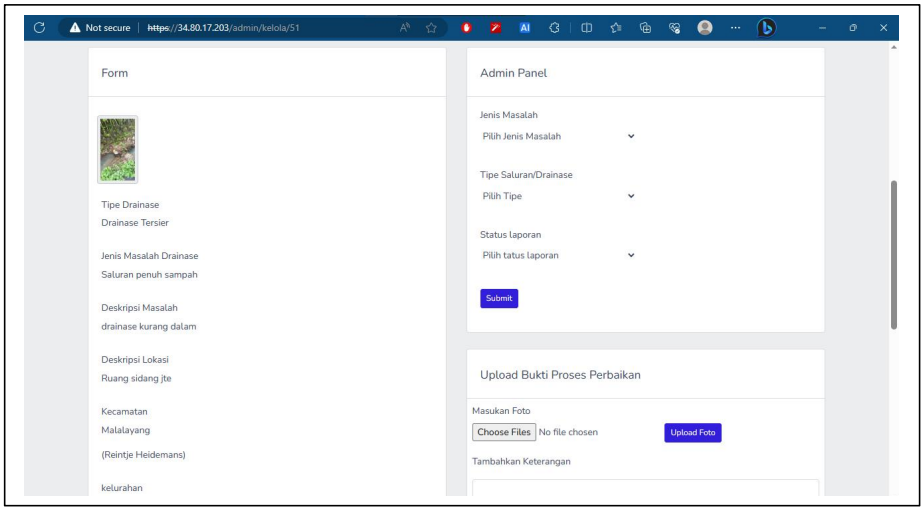
4.2.5 Fitur Notification dan kelola laporan.



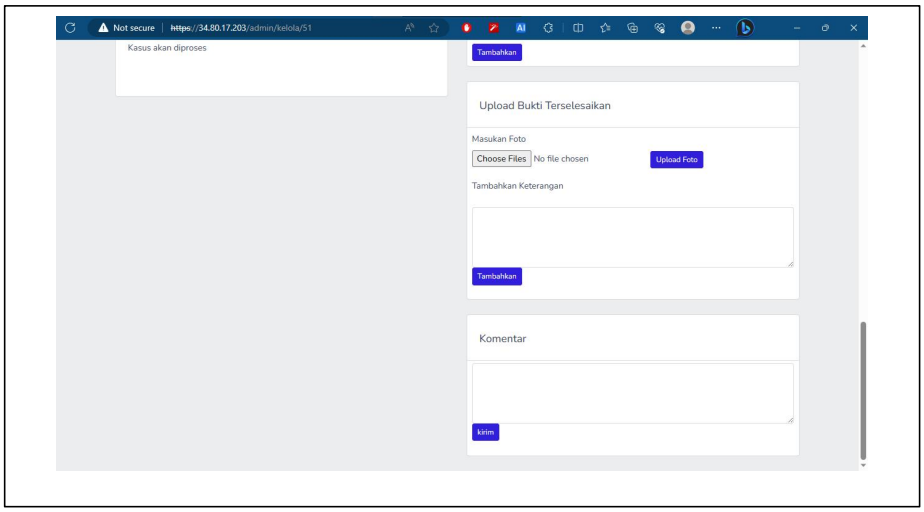
Gambar 4. 157 Fitur notifikasi Admin



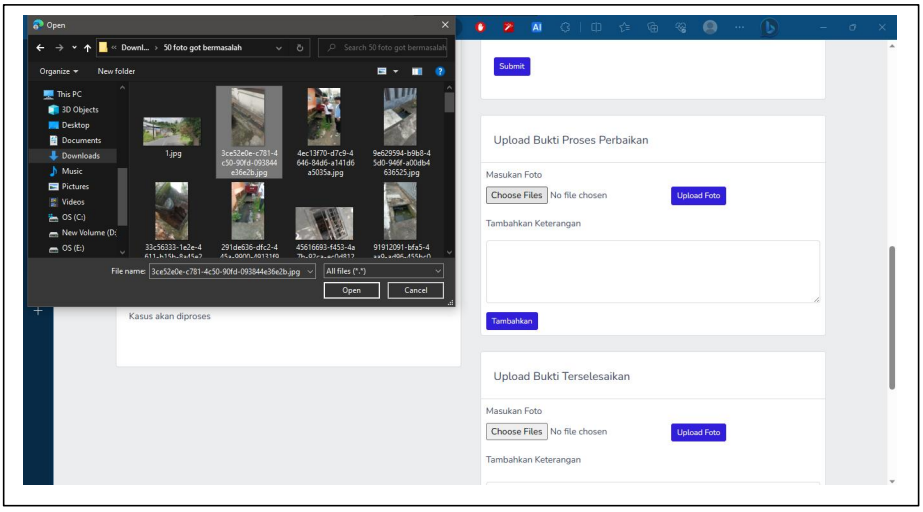
Gambar 4. 158 Kelola laporan (bagian atas).



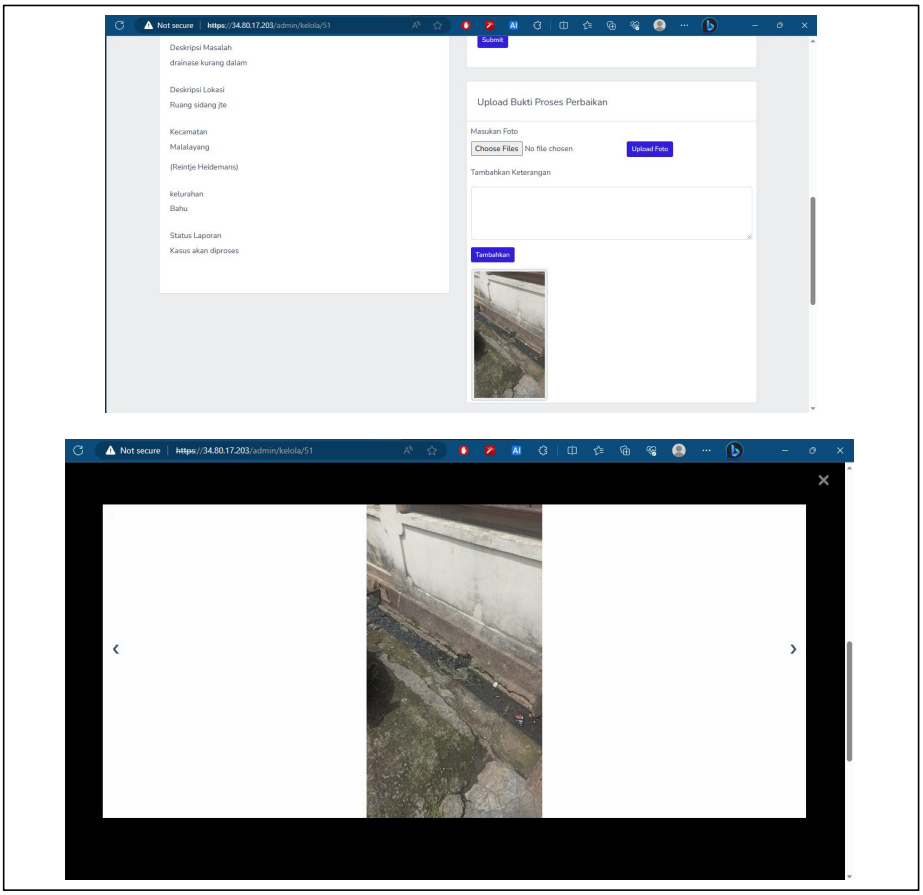
Gambar 4. 159 Kelola laporan (bagian tengah).



Gambar 4. 160 Kelola laporan (bagian bawah).

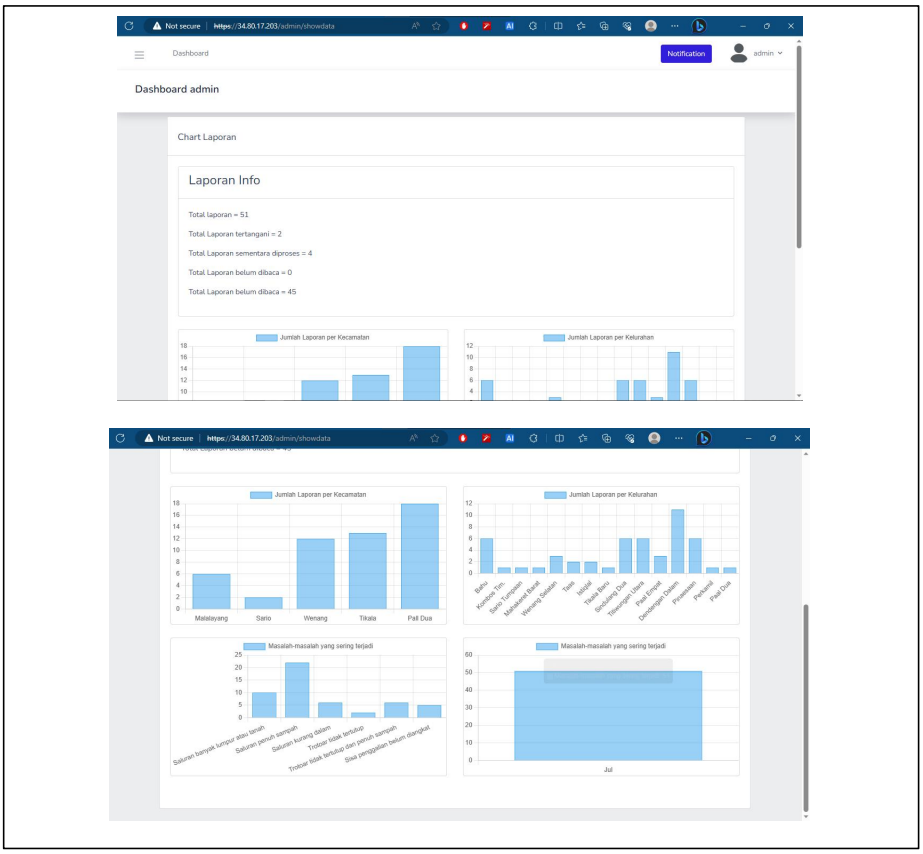


Gambar 4. 161 Melakukan uploud bukti proses perbaikan berupa foto



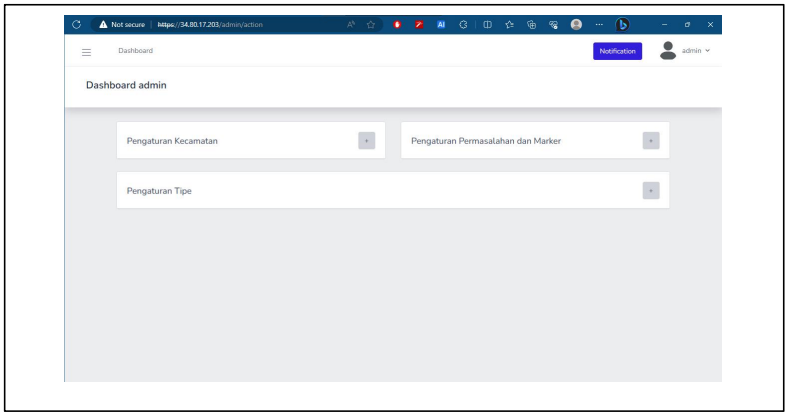
.Gambar 4. 162 Uploud bukti proses perbaikan telah berfungsi dengan baik.

4.2.6 Fitur lihat data laporan oleh admin.

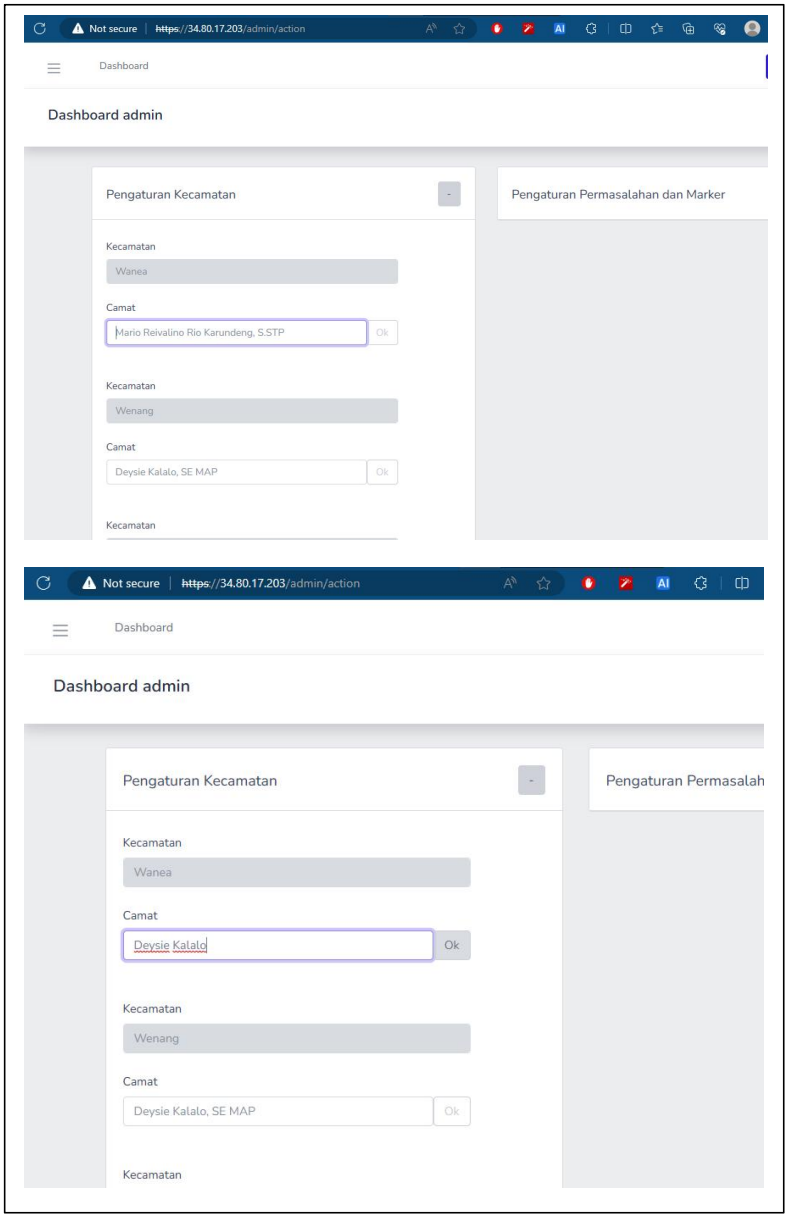


.Gambar 4. 163 Halaman “Lihat Data Laporan (atas dan bawah)”

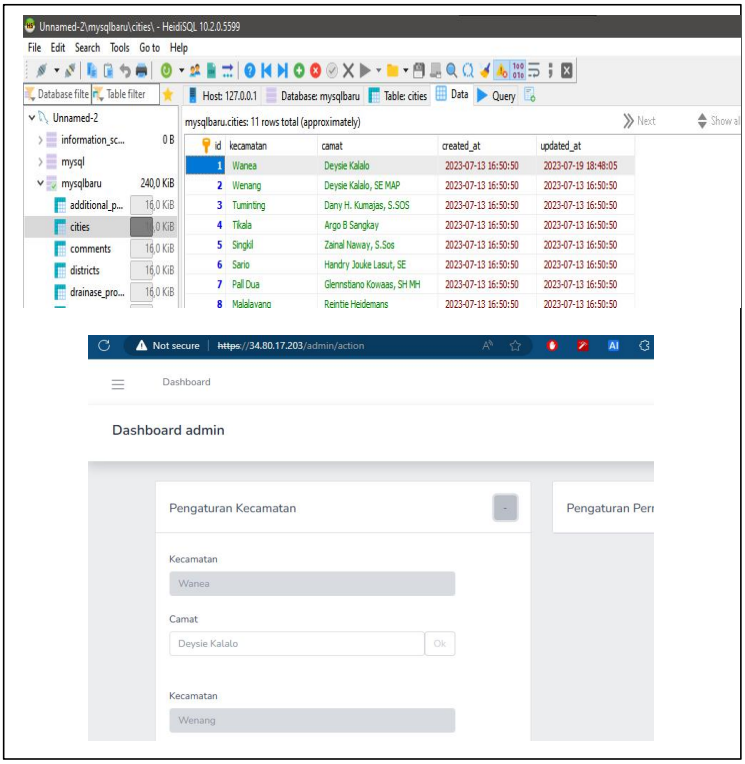
4.2.5 Fitur lihat data laporan oleh admin.



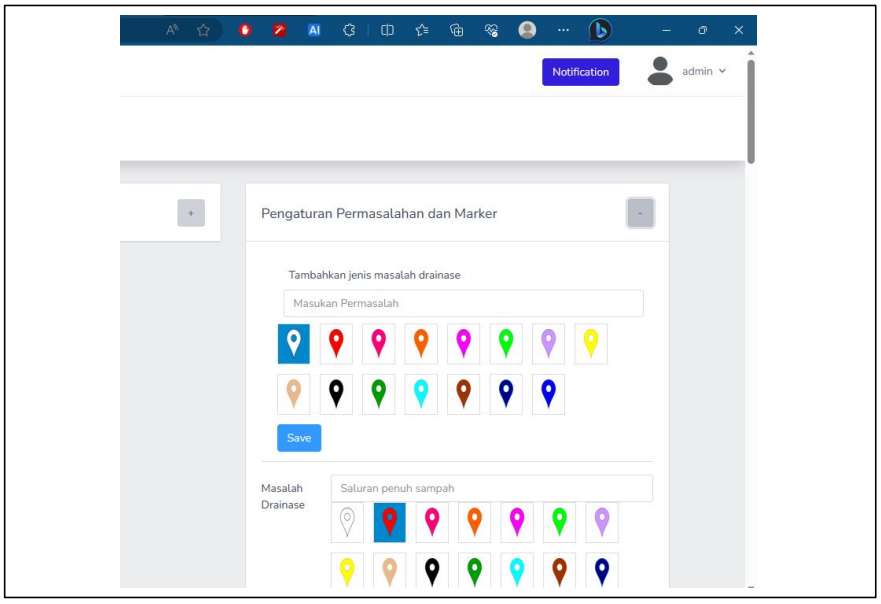
.Gambar 4. 164 Halaman Pengaturan Admin.



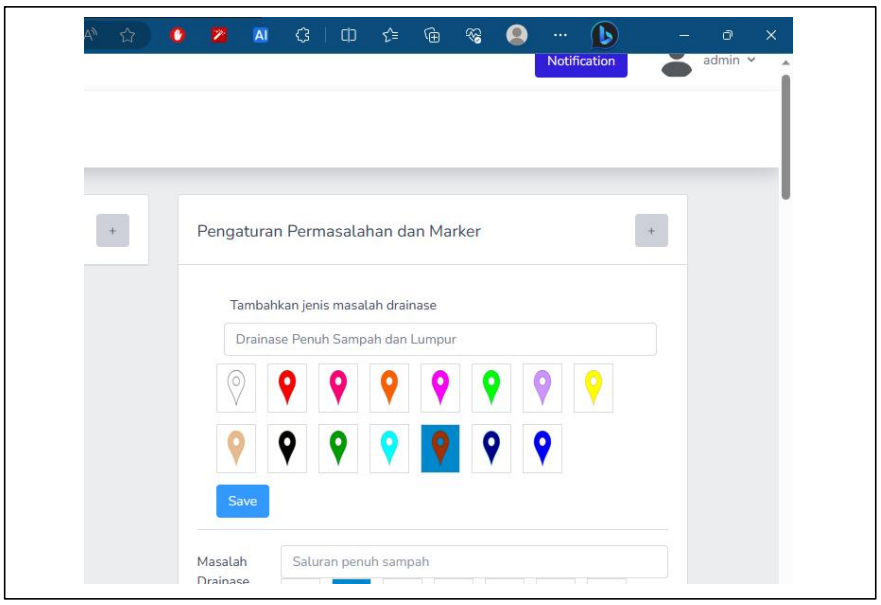
.Gambar 4. 165 Proses merubah nama camat dalam hal ini camat Wanea.



.Gambar 4. 166 data berhasil diperbarui dilihat dari database dan halaman pengaturan.

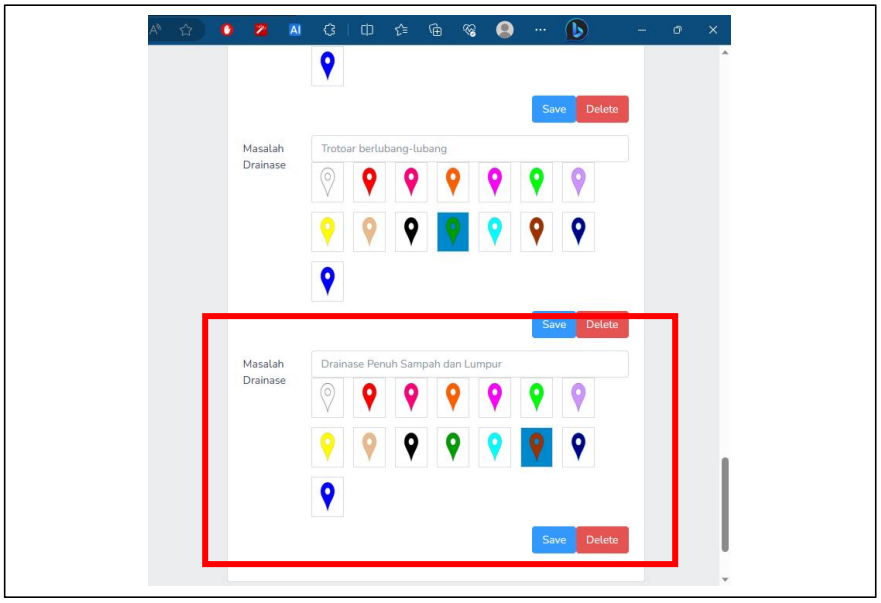


.Gambar 4. 167 Menambahkan jenis masalah baru.

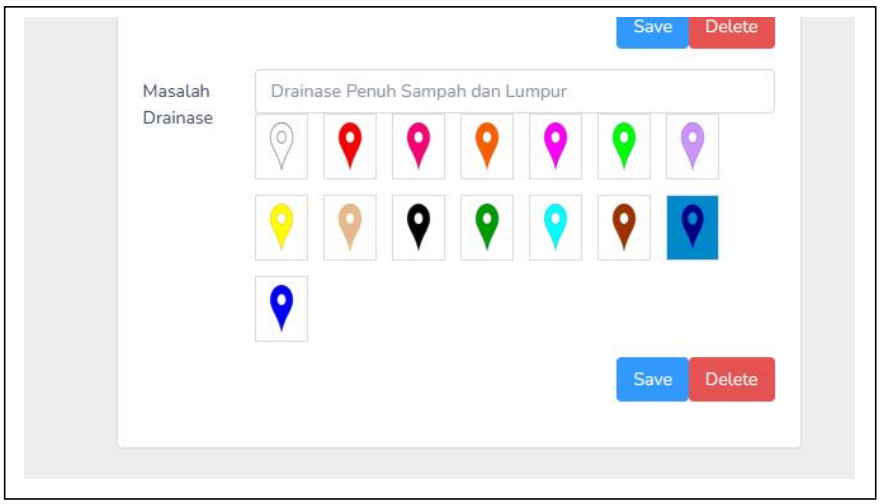


.Gambar 4. 168 Menambahkan jenis masalah baru dengan marker baru.

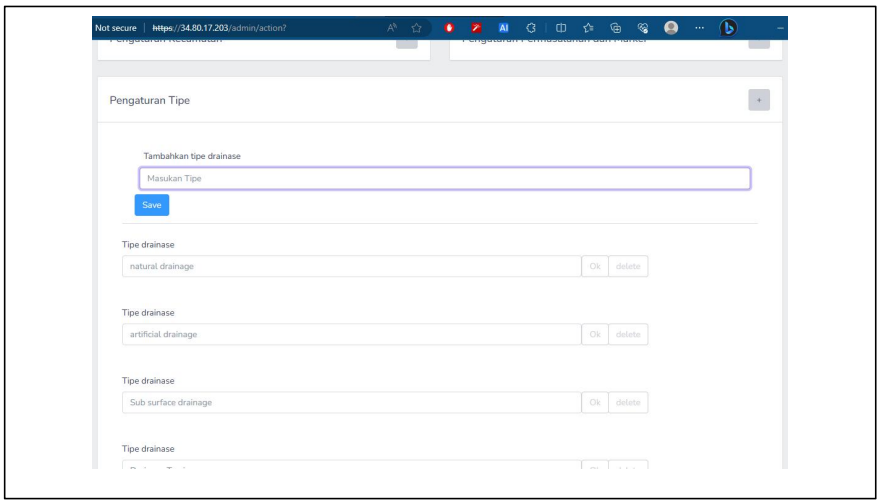
Masalah yang ditambahkan dalam *testing* ini adalah "Drainase Penuh Sampah dan Lumpur" dengan warna marker ialah coklat seperti pada gambar 4.168.



.Gambar 4. 169 Data jenis masalah baru dengan marker baru berhasil ditambahkan.



.Gambar 4. 170 Mengubah warna dari “coklat” ke warna “biru gelap”.



.Gambar 4. 171 Mengubah dan menambahkan tipe drainase.

BAB V

PENUTUP

5.1 Kesimpulan

Setelah berhasil merancang dan membangun aplikasi, tentu terdapat beberapa hal yang dapat disimpulkan, dan juga menjawab beberapa pokok dari tujuan penelitian ini. Berikut ini merupakan hal-hal yang dapat disimpulkan:

1. Pembuatan aplikasi menggunakan *framework Laravel* versi 8, *Composer* digunakan untuk mengunduh dan menginstall *package Laravel* dan berbagai *package* seperti *Livewire, breeze, Sociallite, Sail, dll*. Konfigurasikan *file environment* (*User* dan *Password Mysql, Google credential key*, dll), tabel-tabel database dapat dibuat pada *Laravel Migration* dengan perintah khusus tiap tabel memiliki *model*-nya tersendiri. Tabel-tabel (seperti tabel Kecamatan, Kelurahan, Marker) dapat ditanamkan data dengan menggunakan fitur *Seeder. Livewire* mempercepat dalam mengintegrasikan MVC (*Model View Controller*). Google Maps memakai bahasa pemrograman Javascript sehingga pengembangannya dilakukan dengan bantuan website JSFiddle.

2. *Crowdsourcing* merupakan cara yang memanfaatkan suatu individu-individu untuk mencapai tujuan dengan cepat dan murah. Dan menjawab tantangan yang membutuhkan informasi yang banyak, dengan tampilan peta bersama *marker* informasi yang didapatkan akan lebih presisi dan menarik.

DAFTAR PUSTAKA

Google Developers 2020. Using OAuth 2.0 to Access Google APIs. https://developers.google.com/identity/protocols/OAuth2. Di Akses 8 Maret 2020.

Yasin K. 2019. Laravel Framework : Pengertian, Keunggulan & tips untuk Pemula. https://www.niagahoster.co.id/blog/laravel-adalah/. Di akses 8 Maret 2020.

Andika Kurni Adi Pradana. 2016. Pembuatan Aplikasi Berbasis Crowdsourcing Dalam Upaya Penaggulangan Penduduk Miskin. Teknik Informatika. Politeknik Kediri. Kediri

Russ Miles, Kim Hamilton. 2006. Learning UML 2.0 (A Pragmatic Introduction to UML). O'REILLY. USA.

Ian Sommerville.2007. Software Engineering 8. Addison-Wesley. England.

Arief M Rudianto. 2011. Pemrograman Web Dinamis menggunakan PHP dan MySQL. C.V ANDI OFFSET.Yogyakarta.

Za'imatus Sa'diyah. 2019. Pengembangan Web Service Sistem Informasi Sekolah. Teknik Informatika. Universitas Islam Negeri Maulana Malik Ibrahim Malang. Malang.

Miftah Andriansyah, Teddy Oswari, Budi Prijanto. 2009. Crowdsourcing : Konsep Sumber Daya Kerumunan Dalam Abad Partisipasi Komunitas Internet. Teknik Informatika. Universitas Gunadarma.

Adhitia Listiawati, 2016. Implementasi peraturan daerah kota serang nomor 6 Tahun 2011 tentang rencana tata ruang wilayah kota serang tahun 2010-2030. Universitas Sultan Ageng Tirtayasa Serang. Serang.

Reggy P. Mamonto, Analisis Sistem Jaringan Drainase di Kecamatan Kotamobagu Barat, Kota Kotamobagu. Universitas Sam Ratulangi Manado.

Mercy Corps. 2005. *Design, Monitoring and evaluation guidebook.*

Igor Sysoev. Nginx About. *https://nginx.org/en/* . Di akses 10 November 2022

Docker. Docker Overview https://docs.docker.com/get-started/overview/. Di akses 10 November 2022

Wikipedia. Ubuntu https://id.wikipedia.org/wiki/Ubuntu . Di Akses 29 November 2022

Muhammad Sedawo. Mengenal Ubuntu & Keuntungan Menggunakannya. https://www.niagahoster.co.id/blog/ubuntu-adalah/

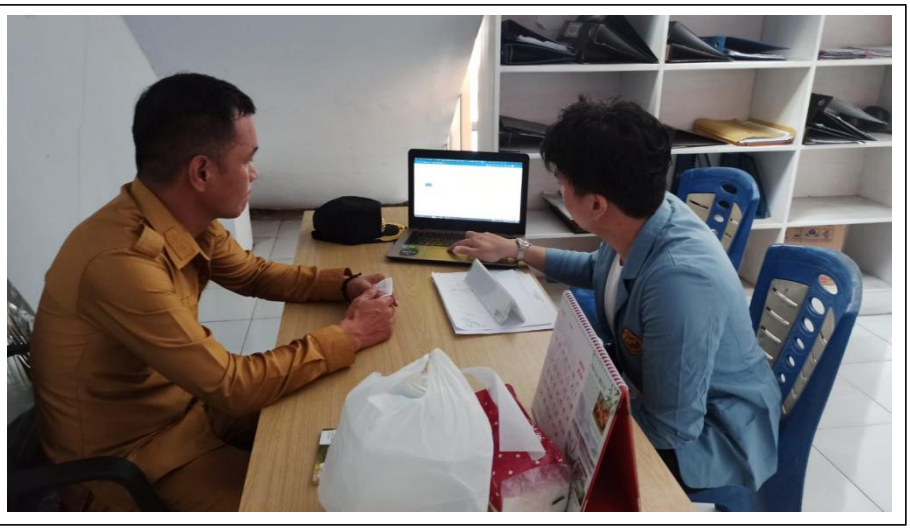
Qigang Liu, Xiangyang Sun. 2012.Research of Web Real-Time Communication Based on Web Socket. Sydney Institute of Language & Commerce Sanghai University, Shanghai China.

Pusher. Channels Overview https://pusher.com/docs/channels/. Di Akses 11 Juli 2023

Chartjs. Why Chart.js https://www.chartjs.org/docs/latest/.Di Akses 12 Juli 2023

LAMPIRAN

Pertemuan dengan pemerintah Kota (Kelurahan)



Pengecekan drainase bermasalah berdasarkan laporan warga (bersama dengan kepala lingkungan)

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan dibawah ini:

Nama : Arthur Manuel Anggow

NIM : 16021106026

Program Studi : S-1 Teknik Informatika

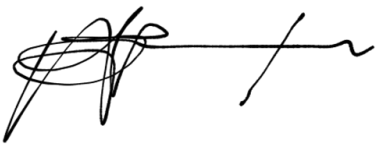
Jurusan : Teknik Elektro

Menyatakan dengan sebenarnya bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil-alihan tulisan atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri.

Apabila di kemudian hari terbukti atau dapat dibuktikan skripsi ini hasil jiplakan, maka saya bersedia menerima sanksi perbuatan tersebut.

Manado, 20 Juli 2023

Yang Membuat Pernyataan



Arthur Manuel Anggow

RIWAYAT HIDUP

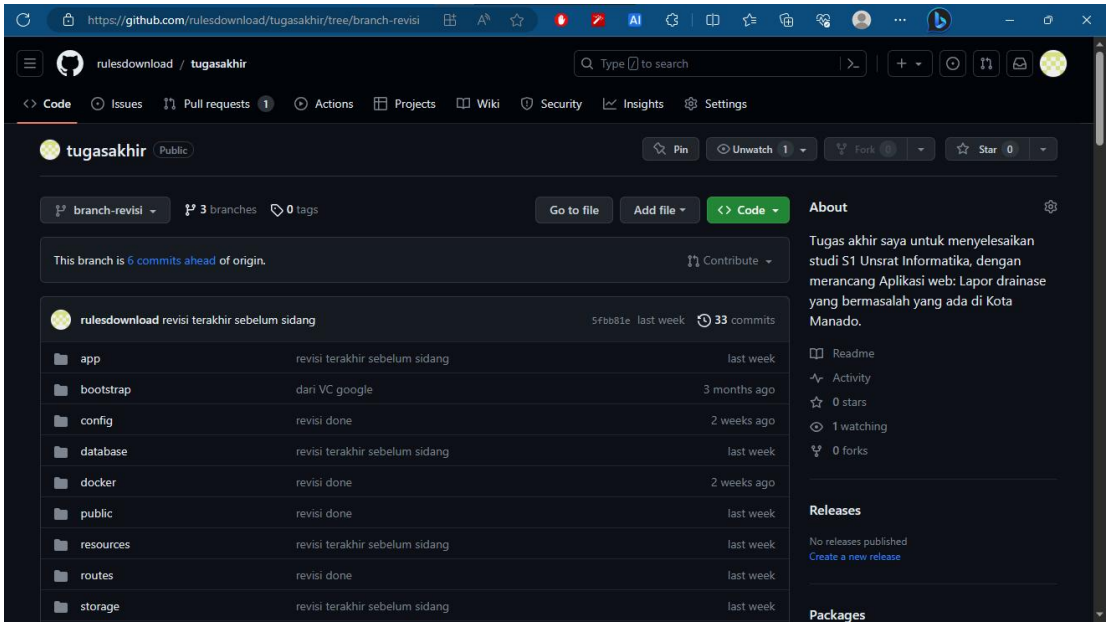
Penulis bernama lengkap Arthur Manuel Anggow, anak Kedua dari dua bersaudara. Lahir di Manado ,Sulawesi Utara pada tanggal 21 April 1999. Dengan alamat tempat tinggal sekarang di Kelurahan Ranomuut, Kecamatan Pall dua, Kota Manado.

Penulis mulai menempuh pendidikan di TK. GMIM Sion Ranomuut (2003-2004). Setelah itu lanjut ke Sekolah Dasar di SD Advent 3 Pall dua (2004-2010). Kemudian melanjutkan pendidikan tingkat pertama di SMP Negeri 1 Manado (2010-2013). Selanjutnya penulis menempuh pendidikan ke sekolah tingkat atas di SMA Negeri 1 Manado (2013-2016).

Setelah itu, di tahun 2016 penulis melanjutkan pendidikan S1 di Program Studi Teknik Informatika, Jurusan Teknik Elektro, Fakultas Teknik, Universitas Sam Ratulangi. Selama berada di bangku kuliah, penulis bergabung dalam organisasi kemahasiswaan yaitu Himpunan Mahasiswa Elektro (HME). Dan akhirnya berhasil menyelesaikan studi di Program Studi Informatika Unsrat.

Selama pembuatan skripsi saya mendapat bimbingan dari para dosen pembimbing Sherwin Reinaldo U. Aldo Sompie, ST, MT. dan Dr. Eng. Steven R. Sentinuwo, ST, MTI. Sehingga penulis dapat menyelesaikan studi di Teknik Elektro Program Studi Teknik Informatika Universitas Sam Ratulangi dengan menyandang gelar Sarjana Komputer (S.Kom).

Keseluruhan Koding dapat dilihat pada Github "Rulesdownload/tugasakhir" branch "branchrevisi"



50 foto drainase bermasalah