

**PEMBANGUNAN SISTEM PELAPORAN *SURVEYOR*  
LAPANGAN UNTUK KEMENTERIAN KELAUTAN &  
PERIKANAN REPUBLIK INDONESIA**

**Tugas Akhir**

**Diajukan untuk Memenuhi Salah Satu Persyaratan Mencapai Derajat  
Sarjana Komputer**



Dibuat Oleh:

**Agung Prio Rismawan**

**160709001**

**PROGRAM STUDI INFORMATIKA  
FAKULTAS TEKNOLOGI INDUSTRI  
UNIVERSITAS ATMA JAYA YOGYAKARTA  
2020**

## HALAMAN PENGESAHAN

Tugas Akhir Berjudul

PEMBANGUNAN APLIKASI PELAPORAN SURVEYOR LAPANGAN UNTUK KEMENTERIAN  
KELAUTAN & PERIKANAN REPUBLIK INDONESIA

yang disusun oleh

AGUNG PRIO RISMAWAN

160709001

dinyatakan telah memenuhi syarat pada tanggal 29 Juli 2020

Dosen Pembimbing 1	: Eduard Rusdianto, ST., MT.	Keterangan	Telah menyetujui
Dosen Pembimbing 2	: Joseph Eric Samodra, S.Kom, MIT.		Telah menyetujui
Tim Penguji			
Penguji 1	: Eduard Rusdianto, ST., MT.		Telah menyetujui
Penguji 2	: Dr. Andi Wahyu Rahardjo, BSEE., MSSE		Telah menyetujui
Penguji 3	: Stephanie Pamela Adithama, ST., MT.		Telah menyetujui

Yogyakarta, 29 Juli 2020

Universitas Atma Jaya Yogyakarta

Fakultas Teknologi Industri

Dekan

ttd

Dr. A. Teguh Siswanto, M.Sc



# PERNYATAAN ORISINALITAS & PUBLIKASI ILMIAH

Saya yang bertanda tangan di bawah ini:

Nama Lengkap : **Agung Prio Rismawan**  
NPM : **160709001**  
Program Studi : Informatika  
Fakultas : Teknologi Industri  
Judul Penelitian : **Pembangunan Sistem Pelaporan *Surveyor*  
Lapangan Untuk Kementerian Kelautan & Perikanan Republik Indonesia**

Menyatakan dengan ini:

1. Tugas Akhir ini adalah benar tidak merupakan salinan sebagian atau keseluruhan dari karya penelitian lain.
2. Memberikan kepada Universitas Atma Jaya Yogyakarta atas penelitian ini, berupa Hak untuk menyimpan, mengelola, mendistribusikan, dan menampilkan hasil penelitian selama tetap mencantumkan nama penulis.
3. Bersedia menanggung secara pribadi segala bentuk tuntutan hukum atas pelanggaran Hak Cipta dalam pembuatan Tugas Akhir ini.

Demikianlah pernyataan ini dibuat dan dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 21 Juli 2020

Yang menyatakan,

Agung Prio Rismawan

160709001

## **HALAMAN PERSEMBAHAN**

Semua akan ke jogja pada waktu-Nya



## KATA PENGANTAR

Puji dan syukur penulis haturkan kepada Tuhan Yang Maha Esa karena berkat rahmat dan karunia-Nya penulis dapat menyelesaikan pembuatan tugas akhir “Pembangunan Sistem Pelaporan *Surveyor* Lapangan Untuk Kementerian Kelautan & Perikanan Republik Indonesia” ini dengan baik. Penulisan tugas akhir ini bertujuan untuk memenuhi salah satu syarat untuk mencapai derajat sarjana Komputer dari Program Studi Informatika, Fakultas Teknologi Industri di Universitas Atma Jaya Yogyakarta.

Penulis menyadari bahwa dalam pembuatan tugas akhir ini penulis telah mendapatkan bantuan, bimbingan, dan dorongan dari banyak pihak. Untuk itu, pada kesempatan ini penulis ingin mengucapkan terima kasih kepada:

1. Tuhan Yang Maha Esa atas berkat dan rahmat-Nya penulis dapat memiliki semangat untuk dapat menyelesaikan tugas akhir ini.
2. Bapak Dr. A. Teguh Siswantoro, selaku Dekan Fakultas Teknologi Industri, Universitas Atma Jaya Yogyakarta.
3. Bapak Eduard Rusdianto S.T., M.T. selaku dosen pembimbing I yang telah membimbing dan memberikan masukan serta motivasi kepada penulis untuk menyelesaikan tugas akhir ini.
4. Bapak Joseph Eric S, S.Kom., MIT. selaku dosen pembimbing II yang telah membimbing dan memberikan masukan serta motivasi kepada penulis untuk menyelesaikan tugas akhir ini.

Demikian laporan tugas akhir ini dibuat, dan penulis mengucapkan terima kasih kepada semua pihak. Semoga laporan ini dapat bermanfaat bagi pembaca.

Yogyakarta, 29 Juli 2020

Agung Prio Rismawan

160709001

# DAFTAR ISI

LEMBAR PENGESAHAN .....	<b>Error! Bookmark not defined.</b>
PERNYATAAN ORISINALITAS & PUBLIKASI ILMIAH.....	iii
HALAMAN PERSEMBAHAN .....	iv
KATA PENGANTAR .....	v
DAFTAR ISI.....	vi
DAFTAR GAMBAR .....	ix
DAFTAR TABEL.....	xi
INTISARI.....	i
BAB I. PENDAHULUAN .....	1
1.1. Latar Belakang .....	1
1.2. Rumusan Masalah .....	2
1.3. Batasan Masalah.....	2
1.4. Tujuan Penelitian .....	2
1.5. Metode Penelitian.....	2
BAB II. TINJAUAN PUSTAKA.....	7
BAB III. LANDASAN TEORI.....	12
3.1. Kementerian Kelautan dan Perikanan Republik Indonesia (KKP) .....	12
3.2. Aplikasi <i>Mobile</i> .....	14
3.2.1 iOS .....	14
3.2.2 Android .....	14
3.3. Sistem Pelaporan.....	15
3.4. Geographic Information System .....	15
3.5. Flutter .....	16
3.6. MariaDB.....	17
3.7. Basis Data .....	17

3.8	Sistem Informasi .....	17
3.9.	BLoC (Business Logic Component) .....	18
BAB IV. ANALISIS DAN PERANCANGAN SISTEM .....		19
4.1.	Analisis Sistem.....	19
4.2.	Lingkup Masalah.....	20
4.3.	Perspektif Produk .....	21
4.3.1.	Kebutuhan Antarmuka Eksternal .....	21
4.3.2.	Antarmuka Pengguna.....	22
4.3.3.	Antarmuka Perangkat Keras .....	24
4.3.4.	Antarmuka Perangkat Lunak .....	25
4.4.	Fungsi Produk .....	26
4.4.1.	Entity Relational Diagram.....	26
4.4.2.	Use Case Diagram.....	28
4.4.1.1	Otentifikasi Pengguna .....	28
4.4.1.2	Membuat Akun Baru.....	29
4.4.1.3	Dashboard Surveyor.....	30
4.4.1.4	OTS Report .....	31
4.4.1.5	Melihat Daftar Laporan.....	32
4.4.1.6	Melihat Detail Laporan .....	32
4.4.1.7	Melihat Daftar Seluruh Laporan <i>Surveyor</i> .....	32
4.4.1.8	Melihat Daftar Seluruh <i>Surveyor</i> .....	33
4.4.1.9	Pengelolaan Data Pengguna.....	33
4.5.	Perancangan Sistem .....	34
4.5.1.	Arsitektur Sistem.....	34
4.5.2.	<i>Package</i> Diagram .....	36

4.5.3.	Kelas Diagram.....	37
4.5.4.	Deskripsi Perancangan Antarmuka.....	39
BAB V. IMPLEMENTASI DAN PENGUJIAN SISTEM.....		59
5.1.	Implementasi Sistem.....	59
5.1.1.	Halaman <i>Login</i> .....	59
5.1.2.	Halaman Register Surveyor.....	61
5.1.3.	Halaman Buat Laporan.....	63
5.1.4.	Tampil Data <i>Surveyor</i> .....	66
5.1.5.	Tampil Daftar Laporan <i>Client</i> .....	68
5.1.6.	Tampil Detail Laporan <i>Surveyor</i> .....	69
5.1.7.	Tampil Data Profil <i>Surveyor</i> .....	71
5.1.8.	Tampil Daftar Laporan Admin.....	72
5.1.9.	Tampil Detail Laporan Admin.....	74
5.1.10.	Tampil Daftar <i>Surveyor</i> Admin.....	76
5.1.11.	Tambah Data <i>Surveyor</i> Admin.....	77
5.1.12.	Ubah Status <i>Surveyor</i> .....	79
5.2.	Pengujian Fungsionalitas Perangkat Lunak.....	83
5.3.	Hasil Pengujian Terhadap Pengguna.....	91
5.4.	Analisis Kelebihan dan Kekurangan Aplikasi.....	92
BAB VI. PENUTUP.....		94
6.1.	Kesimpulan.....	94
6.2.	Saran.....	94
DAFTAR PUSTAKA.....		96

## DAFTAR GAMBAR

Gambar 3.1 Segitiga Terumbu Karang pada daerah kelautan Indonesia .....	12
Gambar 3.2 Gambar BLoC Architecture pada framework Flutter .....	18
Gambar 4.1 Diagram Alir Proses Bisnis .....	19
Gambar 4.2 Entity Relational Diagram MOTS-PeR API .....	26
Gambar 4.3. Use Case Diagram MOTS-PeR.....	28
Gambar 4.4 Arsitektur Sistem MOTS-PeR.....	34
Gambar 4. 5 Rancangan Antarmuka Otentifikasi .....	39
Gambar 4.6 Rancangan Antarmuka Halaman Register .....	41
Gambar 4.7 Rancangan Antarmuka Halaman Dashboard Surveyor.....	43
Gambar 4.8 Rancangan Antarmuka Halaman OTS Report .....	45
Gambar 4.9 Rancangan Antarmuka Halaman Detail Laporan.....	47
Gambar 4.10 Rancangan Antarmuka Halaman History Laporan .....	49
Gambar 4.11 Rancangan Antarmuka Halaman Daftar Seluruh Surveyor .....	50
Gambar 4.12 Rancangan Antarmuka Halaman Tambah Data Surveyor .....	51
Gambar 4.13 Rancangan Antarmuka Halaman Detail Surveyor .....	53
Gambar 4.14 Rancangan Halaman Daftar Seluruh Laporan Surveyor .....	55
Gambar 4.15 Rancangan Halaman Activated/Deactivated Surveyor .....	57
Gambar 5.1 Halaman Login.....	59
Gambar 5.2 Kode Kelas AuthenticatingStarted pada Login.....	60
Gambar 5.3 Halaman Register Client .....	61
Gambar 5.4 Kode Registration BLoC.....	62
Gambar 5.5 Halaman Membuat Laporan Baru .....	63
Gambar 5.6 Potongan Kode Geolocator .....	64
Gambar 5.7 Kode Pengecekan <code>_isValid()</code> Pada Halaman Input Laporan .....	64
Gambar 5.8 Kode Untuk Upload Gambar .....	65
Gambar 5.9 Halaman Dashboard Surveyor .....	66

Gambar 5.10 Potongan Kode Untuk Menampilkan Data Secara Realtime .....	67
Gambar 5.11 Daftar Laporan Surveyor.....	68
Gambar 5.12 Kode Untuk Menampilkan ListView.....	68
Gambar 5.13 Halaman Detail Laporan Surveyor.....	69
Gambar 5.14 Potongan Kode ReportDetailBloc.....	70
Gambar 5.15 Potongan Kode Penampilan Detail Laporan Dalam TextView ....	70
Gambar 5.16 Halaman Profil Pengguna.....	71
Gambar 5.17 Kode UserOverViewBloc.....	71
Gambar 5.18 Potongan Kode Menampilkan Data Pengguna pada TextView....	72
Gambar 5.19 Tampil Daftar Laporan Seluruh Surveyor.....	72
Gambar 5.20 Kode Untuk Menghitung Laporan Surveyor dan Menampilkan Dalam CardView.....	73
Gambar 5.21 Detail Laporan.....	74
Gambar 5.22 Potongan Kode Detail Laporan .....	74
Gambar 5.23 Potongan Kode Penampilan Detail Laporan Dalam TextView ....	75
Gambar 5.24 Halaman Tampil Daftar Surveyor .....	76
Gambar 5.25 Potongan Kode Untuk Menampilkan Jumlah Surveyor dengan cardView.....	76
Gambar 5.26 Tambah Data Surveyor.....	77
Gambar 5.27 Kode Registrasi BLoC .....	78
Gambar 5. 28 Halaman Pengelolaan Surveyor .....	79
Gambar 5.29 Kode Untuk Mendapatkan Status Activate/Deactivate Surveyor ....	80
Gambar 5.30 Halaman Activate dan Deactivate User .....	80
Gambar 5.31 Potongan Kode Untuk Activated/Deactivated Surveyor.....	81

## DAFTAR TABEL

Tabel 2.1. Perbandingan Penelitian.....	<b>Error! Bookmark not defined.</b>
Tabel 5.1.Pengujian Fungsionalitas Perangkat Lunak .....	83
Tabel 5.2.Hasil Pengujian Pada Halaman Surveyor .....	91
Tabel 5. 3. Hasil Pengujian Pada Halaman Admin.....	92



# INTISARI

## Pembangunan Sistem Pelaporan *Surveyor* Lapangan Untuk Kementerian Kelautan & Perikanan Republik Indonesia

Intisari

Agung Prio Rismawan  
160709001

Sektor kelautan di Indonesia memiliki potensi yang besar karena sumber kelautan yang banyak dan beragam. Salah satu cara untuk menjaga dan mengawasi potensi alam tersebut adalah dengan melakukan kegiatan pelaporan dalam ruang lingkup Kementerian Kelautan dan Perikanan. Tapi dalam praktiknya, banyak sekali oknum yang melakukan kecurangan saat survei. Kecurangan tersebut di antaranya adalah memberikan laporan palsu, tidak benar-benar survei di lapangan dan data-data pelaporan dimanipulasi sehingga data pelaporan tidak akurat.

Aplikasi pelaporan dibuat dengan *platform mobile* dan menggunakan arsitektur BLoC (*Business Logic Component*). Arsitektur BLoC digunakan karena mudah dipahami dan *powerful*. Arsitektur BLoC membagi setiap komponen menjadi modul yang kecil-kecil dan tidak bergantung satu-sama lain (*low coupling*). Arsitektur BLoC juga dapat memisahkan antara komponen *logic* dan *presentation layer*. Pemisahan ini akan memudahkan proses *maintenance* kode. Misalnya, pada saat suatu implementasi kode diubah, maka tidak diperlukan banyak perubahan pada kode yang lainnya.

Aplikasi pelaporan ini sudah berhasil dikembangkan untuk mengatasi masalah manipulasi data pelaporan di lapangan. Aplikasi pelaporan ini setelah diuji sudah dapat membantu meningkatkan kualitas dan keakuratan data yang dilaporkan oleh *surveyor*. Aplikasi Pelaporan ini dapat melacak secara *realtime* lokasi *surveyor* saat sedang melakukan pelaporan di lapangan. Lokasi ini diambil dari *latitude* dan *longitude* lokasi pelaporan *surveyor*.

**Kata Kunci:** Pelaporan, *Surveyor*, *Mobile*, *Realtime*, Arsitektur BLoC.

Dosen Pembimbing I : Eduard Rusdianto, S.T., M.T.

Dosen Pembimbing II : Joseph Eric S, S.Kom., MIT.

Jadwal Sidang Tugas Akhir : 29 Juli 2020

# BAB I. PENDAHULUAN

## 1.1. Latar Belakang

Sektor kelautan di Indonesia memiliki potensi yang besar karena kekayaan sumber kelautan yang beragam yaitu berupa hasil tambang, perikanan, mineral, dan budidaya laut lainnya. Hingga akhir Desember 2019, kawasan konservasi perairan saat ini memiliki luas mencapai 23,14 juta hektar atau sekitar 7,12 persen dari luas perairan yang dimiliki Indonesia. Dari jumlah itu, 166 kawasan dikelola oleh Kementerian Kelautan dan Perikanan serta 30 kawasan lain dikelola oleh Kementerian Lingkungan Hidup dan Kehutanan. Kementerian Kelautan dan Perikanan menargetkan terbentuknya kawasan konservasi perairan seluas 32,5 juta hektar atau sekitar 10 persen dari luas perairan Indonesia pada tahun 2030. Dari kekayaan alam tersebut, pemerintah membentuk suatu badan untuk melakukan pengelolaan konservasi dan keanekaragaman laut sebagai salah satu aset negara. Tanggung jawab tersebut diberikan sepenuhnya kepada Kementerian Kelautan dan Perikanan[1].

Untuk memenuhi tanggung jawab tersebut, Kementerian Kelautan dan Perikanan selalu melakukan survei. Survei tersebut dilakukan di setiap kepulauan yang memiliki potensi kelautan dan perikanan. Survei tersebut bertujuan sebagai pertimbangan pemerintah untuk membangun atau memperbaiki fasilitas yang sudah ada di dalam ruang lingkup Kementerian Kelautan, seperti pelaporan kondisi mercusuar, pelaporan hasil perikanan, pelaporan tambak ikan nelayan, pengecekan TEWS (*Tsunami Early Warning System*), dan lain-lain. Untuk mempermudah melakukan survei tersebut, pihak Kementerian Kelautan dan Perikanan sudah memiliki tim yang bertugas untuk melakukan survei yang disebut *surveyor*.

Para *surveyor* ini nantinya akan melakukan pelaporan terkait kondisi laut, perikanan, sarana, dan prasarana yang ada di lapangan. Tapi dalam praktiknya, banyak sekali oknum yang melakukan kecurangan saat survei dan pelaporan di

lapangan. Kecurangan tersebut di antaranya adalah memberikan laporan palsu dan data-data yang dilaporkan tidak benar.

Laporan-laporan tersebut dimanipulasi oleh *surveyor* sehingga data yang dilaporkan oleh *surveyor* tidak akurat. Salah satu contoh adalah *surveyor* tidak melakukan survei di lapangan. Mereka hanya menggunakan patokan data-data sebelumnya untuk dimanipulasi dan dilaporkan pada pelaporan selanjutnya. Hal itu sering dilakukan oleh *surveyor* sehingga seolah-olah terlihat melakukan survei di lapangan.

Oleh karena itu, Kementerian Kelautan dan Perikanan perlu merancang aplikasi *mobile* untuk mengurangi kecurangan tersebut. Aplikasi tersebut dirancang untuk para *surveyor* yang digunakan dalam melaporkan data-data di lapangan. Manfaat aplikasi ini dapat melacak lokasi terkini dari *surveyor* sehingga keberadaan *surveyor* dapat diketahui secara *realtime* saat melakukan survei di lapangan.

Aplikasi pelaporan berbasis *mobile* sudah banyak diciptakan dan dianggap efektif untuk meningkatkan akurasi pelaporan. Harapan dari aplikasi tersebut adalah data yang dilaporkan akurat dan sesuai dengan fakta yang ada di lapangan. Oleh karena itu, penulis melakukan pengembangan aplikasi *mobile* untuk Kementerian Kelautan dan Perikanan yang nanti dapat dimanfaatkan oleh *surveyor* dalam melakukan pelaporan yang lebih baik.

## **1.2. Rumusan Masalah**

Berdasarkan latar belakang di atas, maka diperoleh rumusan masalah sebagai berikut:

1. Bagaimana aplikasi ini dapat mengurangi kecurangan terkait laporan *surveyor*?
2. Bagaimana aplikasi ini dapat menyajikan data pelaporan *surveyor* yang akan diserahkan kepada Kementerian Kelautan dan Perikanan?
3. Bagaimana Aplikasi ini dapat dibangun menggunakan arsitektur BLoC (Business Logic Component)

### **1.3. Batasan Masalah**

1. Aplikasi ini hanya mencakup ruang lingkup Kementerian Kelautan dan Perikanan Republik Indonesia.
2. Aplikasi ini hanya digunakan untuk pengelolaan data pelaporan dari *surveyor* di lapangan.
3. Aplikasi ini dibangun hanya dengan platform *mobile* untuk pelaporan *surveyor* dan juga manajemen data utama oleh *administrator*.

### **1.4. Tujuan Penelitian**

Pembangunan sistem informasi ini memiliki tujuan sebagai berikut:

1. Mengurangi kecurangan *surveyor* terhadap data-data yang dilaporkan pada saat survei di lapangan.
2. Menyajikan data pelaporan *surveyor* yang akan diserahkan kepada Kementerian Kelautan dan Perikanan.
3. Membangun aplikasi pelaporan berbasis *mobile* menggunakan arsitektur BLoC (Business Logic Component)

### **1.5. Metode Penelitian**

#### **a. Wawancara**

Wawancara adalah tahap pertama yang dilakukan sebelum pembuatan aplikasi dilakukan. Tahap wawancara digunakan untuk mengumpulkan informasi yang dibutuhkan untuk membangun sebuah aplikasi. Metode wawancara dipercaya sangat efektif untuk mencari informasi dari pengguna aplikasi dikarenakan kita dapat melakukan interaksi secara langsung dan narasumber juga dapat memberikan informasi secara cepat dan jelas.

#### **b. Analisis Kebutuhan aplikasi**

Tahap selanjutnya adalah melakukan analisis terhadap kebutuhan yang didapat pada tahap wawancara. Pada tahap ini kebutuhan perangkat lunak akan disusun dan dirancang melalui beberapa

dokumen. Dokumen tersebut adalah SKPL (Spesifikasi Kebutuhan Perangkat Lunak) dan DPPL (Deskripsi Perancangan Perangkat Lunak). Apabila SKPL dan DPPL sudah terbentuk langkah selanjutnya adalah melakukan validasi kepada pengguna untuk memastikan kebutuhan aplikasi yang ditulis pada dokumen sudah mencakup kebutuhan yang diminta.

c. Merancang Aplikasi

Setelah mendapatkan validasi dari pengguna, tahap selanjutnya adalah melakukan perancangan aplikasi. Perancangan tersebut dilakukan berdasarkan dokumen SKPL dan DPPL yang sudah divalidasi oleh pengguna pada tahap sebelumnya. Pada tahap ini pengembang akan melakukan perancangan antarmuka, pembuatan ERD, arsitektur yang digunakan, dan lain-lain.

d. Pengkodean Aplikasi

Tahap selanjutnya adalah pengkodean. Pada tahap ini pengembang akan melakukan implementasi aplikasi terhadap kebutuhan yang sudah dirancang sebelumnya. Hasil dari pengkodean ini adalah aplikasi jadi yang berdasarkan dokumen SKPL dan DPPL.

e. Pengujian Aplikasi

Tahap selanjutnya adalah pengujian. Pengujian dilakukan melalui beberapa tahap, yaitu pengujian oleh pengguna dan pengujian fungsionalitas aplikasi. Pengujian fungsionalitas digunakan untuk menguji aplikasi secara fungsionalitas, sedangkan pengujian pengguna digunakan untuk menguji *usability* dari aplikasi yang sudah dibuat. Dari pengujian ini apabila terdapat kesalahan maka akan segera diperbaiki secara fungsionalitas maupun *usability*.

f. Laporan pembuatan aplikasi.

Tahap terakhir adalah pembuatan laporan. Pada tahap ini mencakup beberapa hal yaitu latar belakang masalah, tinjauan pustaka, perancangan aplikasi, pengujian aplikasi, dan lain-lain. Pada laporan juga akan dilampirkan hasil aplikasi yang sudah

dikembangkan oleh pengembang.

## 1.6. Sistematika Penulisan

### **BAB 1 - Pendahuluan**

Bab ini berisi latar belakang masalah, tujuan pembangunan aplikasi, batasan-batasan masalah, metode yang digunakan, dan sistematika penulisan laporan tugas akhir.

### **BAB II - Tinjauan Pustaka**

Bab ini berisi tentang uraian singkat tentang hasil-hasil penelitian terdahulu yang berkaitan dengan permasalahan yang akan ditinjau penulis dengan topik penelitian di tugas akhir ini.

### **BAB III - Landasan Teori**

Bab ini berisi penjelasan tentang dasar teori yang digunakan penulis dalam melakukan perancangan dan pembuatan perangkat lunak yang digunakan sebagai pembandingan atau acuan dalam pembahasan masalah.

### **BAB IV - Analisis dan Perancangan Sistem**

Bab ini akan membahas analisis dan perancangan sistem pelaporan *surveyor* lapangan, seperti lingkup masalah, perspektif produk, kebutuhan antarmuka eksternal, kebutuhan fungsionalitas perangkat lunak, *Entity Relationship Diagram* (ERD), *use case diagram*, *class diagram*, dan deskripsi perancangan antarmuka.

### **BAB V - Implementasi dan Pengujian Sistem**

Bab ini berisi tentang pembahasan penggunaan sistem pelaporan *surveyor* lapangan meliputi implementasi dan pengujian perangkat lunak yang telah dibuat. Implementasi digunakan untuk menjabarkan bagian-bagian sistem. Pengujian aplikasi digunakan untuk menganalisis apakah sistem yang dibuat sudah memenuhi tujuan yang hendak dicapai.

## **BAB VI - Kesimpulan dan Saran**

Bab ini merupakan penutup, kesimpulan, dan saran yang didapat selama pembuatan tugas akhir akan disertakan pada bab ini.



## BAB II. TINJAUAN PUSTAKA

Terdapat penelitian terkait dengan pembangunan aplikasi pelaporan yang pernah dilakukan sebelumnya. Pada penulisan laporan tugas akhir ini, penulis menggunakan empat pustaka sebagai bahan acuan dan bahan pembanding.

Penelitian pertama dilakukan oleh Mambu, dkk. yaitu pembuatan aplikasi pelaporan Manado *Smart City* untuk Pemerintah Kota Manado. Penelitian ini bertujuan untuk memudahkan partisipasi masyarakat dalam pelaporan Manado *Smart City*. Pengembangan *e-report* pada Manado *Smart City* dibangun dengan menggunakan teknologi Ionic dengan bantuan Cordova. Penelitian ini menghasilkan aplikasi *mobile* yang mendukung kegiatan pelaporan dalam Manado *Smart City*. Layanan aplikasi *mobile* digunakan untuk meningkatkan kemampuan dan fleksibilitas bagi lembaga yang bersangkutan dalam mengumpulkan data yang diperlukan agar meminimalisir kesalahan pengumpulan data [2].

Penelitian kedua dilakukan oleh Haloho dan Pribadi. Penelitian ini berjudul Perancangan Aplikasi Berbasis Android untuk Survei Kondisi Kapal oleh *Owner Surveyor*. Penelitian ini dibangun menggunakan bahasa pemrograman Java dengan *tools* Android Studio. Aplikasi ini mempunyai kelebihan dapat menyajikan laporan secara otomatis ketika *user* telah melakukan pengisian *form*. Pengisian data dilakukan dengan cara survei secara berkala untuk pengecekan kondisi kapal sekaligus pendataan kapal di lapangan. Tujuan perawatan kapal adalah menjamin terlaksananya sistem pemeliharaan terencana PMS (*Planned Maintenance System*) [3].

Penelitian berikutnya dilakukan oleh Hati, dkk. yang berjudul Aplikasi Penanda Lokasi Peta Digital Berbasis *Mobile GIS* Pada *Smartphone* Android. Penelitian ini melibatkan penggunaan GIS (*Geographic Information System*) yang digunakan untuk menandakan sebuah tempat atau lokasi melalui aplikasi *mobile* Android. Hasil dari penelitian ini adalah aplikasi *mobile* dengan fitur untuk menandakan lokasi dengan menggunakan GIS serta beberapa fitur utama seperti

*input* data, menampilkan data yang tersimpan, dan menampilkan rute pada peta [4].

Penelitian yang terakhir dilakukan oleh Somya. Penelitian ini berjudul Sistem *Monitoring* Kendaraan Secara *realtime* Berbasis Android menggunakan teknologi CouchDB di PT. Pura Barutama. Penelitian ini digunakan untuk *monitoring* kendaraan secara *realtime* menggunakan teknologi CouchDB dan GPS *tracker* pada studi kasus PT. Pura Barutama. CouchDB adalah salah satu contoh dari RTDBS (*Realtime Database System*) yang dikembangkan oleh Apache. Penelitian ini bertujuan untuk mendeteksi keberadaan kendaraan secara *realtime*. Pada penelitian ini, GPS berfungsi sebagai pendeteksi keberadaan kendaraan. Kendaraan yang sudah dipasang GPS apabila berpindah tanpa diketahui oleh pemiliknya maka akan otomatis mengirimkan notifikasi melalui aplikasi Android [5].



Pada tabel 2.1 merupakan penelitian yang terdahulu dan dijadikan sebagai referensi penulisan laporan tugas akhir. Terdapat 4 referensi yang digunakan, diantaranya adalah sebagai berikut:

Tabel 2.1. Perbandingan Penelitian

Peneliti	Mambu, dkk. [2]	Haloho dan Pribadi [3]	Hati [4]	Somya [5]	*) Rismawan
<b>Judul</b>	Pembangunan Aplikasi <i>E-Report</i> Layanan Masyarakat untuk Manado	Perancangan Aplikasi Komputer Berbasis Android untuk Survei Kondisi Kapal oleh Owner <i>Surveyor</i>	Aplikasi Penanda Lokasi Peta Digital Berbasis <i>Mobile GIS</i> Pada <i>Smartphone</i> Android	Sistem <i>Monitoring</i> Kendaraan Secara <i>Realtime</i> Berbasis Android menggunakan Teknologi CouchDB di PT. Pura Barutama	Pembangunan Sistem Pelaporan <i>Surveyor</i> Lapangan Untuk Kementerian Kelautan & Perikanan Republik Indonesia
<b>Bahasa</b>	Javascript dan Cordova	Java dan PHP	Java	Java dan JavaScript	Flutter, Dart, dan Golang
<b>Basis Data</b>	MySQL	PostgreSQL	SQLite	CouchDB	MySQL
<b>Akses Lokasi Pengguna Secara <i>Realtime</i></b>	X	X	X	√	√

<b>Fitur Aplikasi:</b>					
------------------------	--	--	--	--	--



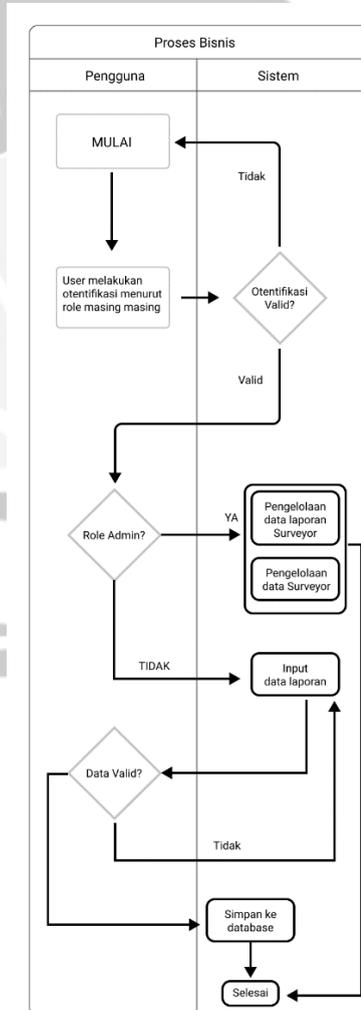
• <i>Login</i>	X	X	X	√	√
• <b>Pengelolaan Pengguna</b>	X	X	X	√	√
• <i>Signup</i>	X	X	X	√	√
• <i>Input Laporan</i>	√	√	√	X	√
• <b>Profil Pengguna</b>	X	X	X	X	√
• <i>Preview Laporan</i>	√	√	√	X	√
• <i>Tracking Lokasi Secara Realtime</i>	X	X	√	√	√
• <i>Active/Nonactive Pengguna</i>	X	X	X	√	√

\*) Masih dalam penelitian

## BAB IV. ANALISIS DAN PERANCANGAN SISTEM

Bab ini akan membahas analisis dan perancangan perangkat lunak MOTS-PeR yang merupakan aplikasi pencatatan laporan untuk *surveyor* dalam ruang lingkup Kementerian Kelautan dan Perikanan Republik Indonesia. Pembahasan yang dicakup seperti kebutuhan fungsionalitas perangkat lunak, *entity relationship diagram*, *sequence diagram*, lingkup masalah, kebutuhan antarmuka eksternal, dan deskripsi perancangan antarmuka.

### 4.1. Analisis Sistem



Gambar 4.1 Diagram Alir Proses Bisnis

Pada Gambar 4.1 menunjukkan proses bisnis MOTS-PeR. Terdapat 2 aktor pada diagram alir pada gambar 4.1, yakni pengguna dan sistem. Proses bisnis yang digambarkan adalah proses pengadaan laporan dimulai dari pengumpulan data. Pada diagram Gambar 4.1, pengguna dapat mengirimkan data laporan ke dalam aplikasi, sistem kemudian akan memvalidasi data yang masuk untuk memastikan data yang masuk adalah data yang valid dan lengkap. Jika data dinyatakan valid, maka sistem akan menyimpan data tersebut. Untuk pengguna yang tidak memasukkan laporan ke dalam sistem (admin) dapat melakukan pengelolaan *surveyor* dan melihat seluruh laporan yang masuk oleh *surveyor*.

#### **4.2. Lingkup Masalah**

Proyek ini bertujuan untuk mempermudah proses pelaporan dan mengurangi tindak kecurangan pada saat pelaporan di lapangan yang dilakukan oleh *surveyor* saat melakukan survei dalam ruang lingkup Kementerian Kelautan dan Perikanan Republik Indonesia. Selain untuk mempermudah proses pelaporan, proyek ini dibuat agar data yang dilaporkan oleh *surveyor* dapat diolah dengan mudah oleh Kementerian Kelautan dan Perikanan Republik Indonesia. Proyek ini dikembangkan dalam bentuk *mobile*. MOTS-PeR menyediakan layanan utama, antara lain:

1. Pengelolaan data *surveyor*,
2. Pengelolaan laporan *surveyor*,
3. Menampilkan lokasi *surveyor* saat melakukan survei,
4. Melakukan pelaporan oleh *surveyor*, dan
5. *Activate/Deactivate surveyor* oleh admin.

Dengan adanya aplikasi ini diharapkan dapat meningkatkan proses pelaporan, dapat mengelola data dengan cepat dan tepat, dan mengurangi tindak kecurangan agar laporan yang disampaikan oleh *surveyor* kepada Kementerian Kelautan dan Perikanan (KKP) dapat lebih akurat dan terpercaya.

### 4.3. Perspektif Produk

Pada MOTS-PeR akan menghasilkan beberapa fitur utama untuk mendukung proses pelaporan di lapangan. Fitur tersebut adalah pengolahan data *surveyor* oleh admin, laporan OTS oleh *surveyor*, pengelolaan data laporan oleh admin, dan *activate/deactivate surveyor* oleh admin. Untuk mengurangi kecurangan saat pelaporan *surveyor* di lapangan, terdapat fitur pelacakan lokasi secara *realtime* menggunakan Google Maps yang disisipkan saat *OTS Report* oleh *surveyor*.

Beberapa perangkat lunak/peralatan yang akan digunakan untuk mendukung jalannya MOTS-PeR, antara lain:

1. MariaDB sebagai *Database Management System* yang digunakan untuk menyimpan data utama dari MOTS-PeR.
2. Ubuntu sebagai sistem operasi untuk server komputer.
3. Flutter sebagai *framework* yang digunakan untuk pembuatan antarmuka pengguna yang interaktif.
4. Figma sebagai *design tools* yang digunakan untuk perancangan antarmuka sebelum produk masuk ke dalam tahap *development*.

#### 4.3.1. Kebutuhan Antarmuka Eksternal

Kebutuhan antarmuka eksternal pada perangkat lunak MOTS-PeR meliputi kebutuhan antarmuka pengguna, antarmuka perangkat keras, antarmuka perangkat lunak, dan antarmuka komunikasi.

### 4.3.2. Antarmuka Pengguna

Pada aplikasi MOTS-PeR memiliki 10 antarmuka perangkat lunak yang terdiri dari aplikasi MOTS-PeR client dan aplikasi MOTS-PeR admin yang memiliki fungsi yang berbeda-beda. Tabel 4.1 menjelaskan deskripsi dan peran dari antar muka tersebut.

Tabel 4.1. Tabel antarmuka MOTS-PeR

No.	Nama Antarmuka	Deskripsi Fungsi Antarmuka	Peran
1	<i>Log in</i>	Aplikasi akan meminta masukan berupa nama pengguna dan <i>password</i> untuk masuk ke dalam aplikasi berdasarkan peran yang dimilikinya.	<i>Surveyor</i> , Admin
2	<i>Register</i>	Antarmuka ini berguna untuk melakukan pendaftaran oleh <i>surveyor</i> sebelum dapat melakukan proses pelaporan di lapangan. <i>Surveyor</i> diminta untuk memasukan <i>username</i> , nama lengkap, jenis kelamin, unit kerja, posisi pekerjaan, email, KTP, nomor telepon, kata sandi, dan konfirmasi kata sandi.	<i>Surveyor</i>
3	Halaman <i>Dashboard</i>	Antarmuka ini berisi rangkuman data-data pribadi <i>surveyor</i> , lokasi <i>surveyor</i> saat ini, ID pegawai, unit pekerjaan, dan posisi pekerjaan saat	<i>Surveyor</i>

		ini.	
4	Halaman Daftar Laporan yang Telah Dikerjakan	Antarmuka ini berisi daftar laporan yang sudah dikerjakan oleh <i>surveyor</i> tersebut.	<i>Surveyor</i>
5	Halaman Detail Laporan	Antarmuka ini berisi detail laporan yang sudah dibuat oleh <i>surveyor</i> sebelumnya.	<i>Surveyor</i>
6	Halaman Tambah Laporan	Antarmuka ini digunakan untuk memasukkan data-data laporan saat berada di lapangan. Data-data yang harus dimasukkan adalah nama pekerjaan, nomor kontrak, durasi pekerjaan, deskripsi rinci pekerjaan yang sudah dilakukan, dan foto. Pada saat mengirimkan laporan, sistem akan otomatis mengambil data lokasi dari <i>surveyor</i> tersebut. Data lokasi berupa <i>latitude</i> dan <i>longitude</i> dari lokasi saat itu, sehingga admin dapat memantau apakah data yang dilaporkan sesuai dengan lokasinya atau tidak.	<i>Surveyor</i>
7	<i>Log out</i>	Antarmuka ini berfungsi agar <i>surveyor</i> dapat keluar dari aplikasi dan akan diarahkan ke halaman <i>login</i> . Apabila <i>surveyor</i> ingin	<i>Surveyor</i>

		masuk ke dalam aplikasi, maka harus login terlebih dahulu.	
8	Halaman Daftar Laporan <i>Surveyor</i>	Aplikasi akan menampilkan seluruh laporan yang sudah dilakukan oleh pengguna. Data laporan tersebut disajikan dalam bentuk daftar yang berisi tanggal laporan, id laporan, dan judul laporan tersebut.	Admin
9	Halaman Daftar Seluruh <i>Surveyor</i>	Antarmuka ini akan menampilkan <i>surveyor</i> yang sudah mendaftar pada aplikasi ini. Di dalamnya terdapat tanggal pendaftaran, id <i>surveyor</i> , dan nama.	Admin
10	Halaman Pengelolaan Pengguna	Antarmuka ini berfungsi untuk menampilkan seluruh <i>surveyor</i> dan admin dapat melakukan <i>activate/deactivate surveyor</i> . Apabila statusnya aktif, maka <i>surveyor</i> tersebut bisa membuat laporan. Apabila statusnya tidak aktif, maka <i>surveyor</i> tidak dapat membuat laporan.	Admin

#### 4.3.3. Antarmuka Perangkat Keras

Perangkat keras yang digunakan untuk berkomunikasi dengan MOTS-PeR yaitu *smartphone* Android. *Smartphone* Android merupakan perangkat keras yang berfungsi untuk mengakses aplikasi berbasis *mobile*. Android yang digunakan adalah versi Oreo 8.1.

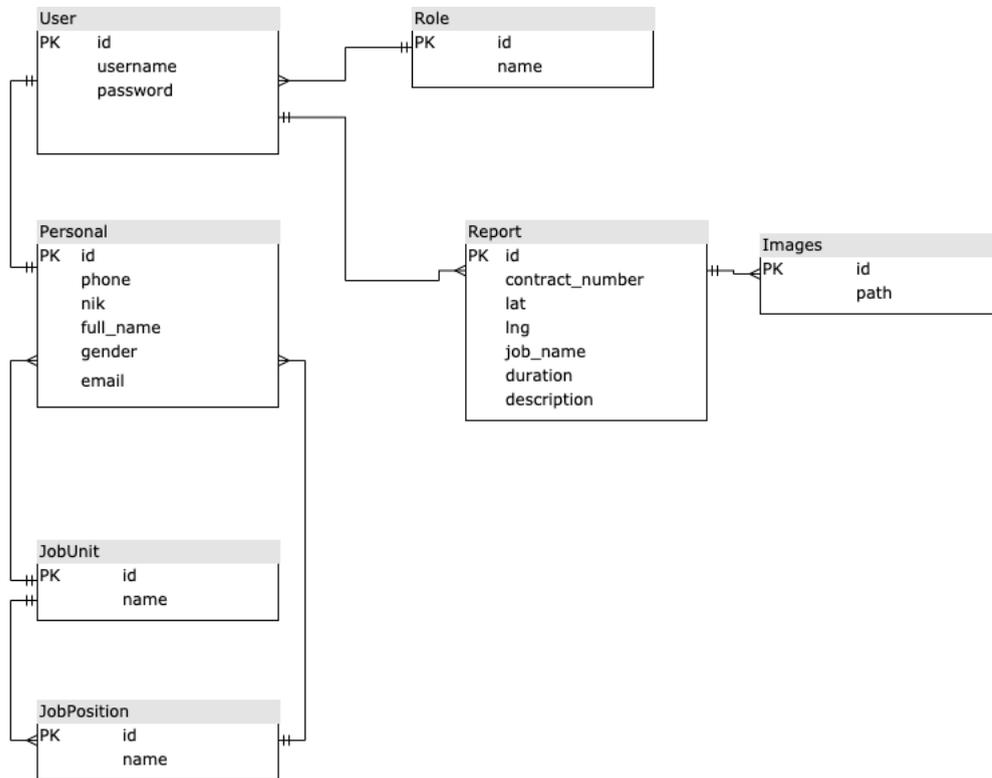
#### 4.3.4. Antarmuka Perangkat Lunak

Perangkat lunak yang digunakan untuk mengoperasikan aplikasi MOTS-PeR adalah sebagai berikut:

1. Nama : Ubuntu 18.04  
Sumber : Canonical  
Deskripsi : Sistem operasi *server* untuk MOTS-PeR
2. Nama : MariaDB 10.3  
Sumber : MariaDB Foundation  
Deskripsi : Basis data yang digunakan MOTS-PeR
3. Nama : Nginx  
Sumber : Nginx  
Deskripsi : *Web server* untuk MOTS-PeR
4. Nama : Google Chrome  
Sumber : Google  
Deskripsi : Peramban untuk mengakses API
5. Nama : Docker 18.09.4  
Sumber : Docker, Inc.  
Deskripsi : Sebagai perangkat lunak virtual berbasis *container* untuk mengisolir dan mengemas MOTS-PeR ketika proses *deployment*.
6. Nama : GitLab CI  
Sumber : GitLab  
Deskripsi : Sebagai *platform* penyedia layanan *continuous integration/continuous deployment* yang digunakan untuk *deployment* MOTS-PeR.
7. Nama : Android Oreo 8.1  
Sumber : Google Android  
Deskripsi : Sistem operasi untuk pengguna MOTS-PeR

## 4.4. Fungsi Produk

### 4.4.1. Entity Relational Diagram



Gambar 4.2 Entity Relational Diagram MOTS-PeR API

Pada Gambar 4.2 menunjukkan ERD dari MOTS-PeR API. Analisis sistem secara detail dapat dilihat pada Spesifikasi Kebutuhan Perangkat Lunak yang telah dilampirkan bersama laporan ini. Dalam alur bisnis utama sistem ini, entitas yang memegang peranan penting adalah *User*, *Personal*, dan entitas *Report*. Kombinasi data dari 3 entitas tersebut yang akan menghasilkan laporan surveyor pada aplikasi MOTS-PeR .

Entitas *User* yang didukung oleh entitas *role* dan entitas *personal* merupakan entitas yang merepresentasikan data Pengguna pada aplikasi MOTS-PeR. Entitas *User* nantinya akan memiliki detail dari *user* yang sudah mendaftarkan diri pada aplikasi dan mempunyai *role* untuk masing-masing user. Entitas user memiliki atribut utama sebagai berikut:

1. *id* yang menjadi *primary key* dari entitas ini.
2. *username* yang merupakan nama pengguna untuk dapat masuk ke dalam aplikasi.
3. *password* merupakan kata sandi untuk dapat masuk ke dalam aplikasi.

Entitas *Personal* merupakan representasi detail data *user*. Entitas ini memiliki atribut sebagai berikut:

1. *id* merupakan *primary key* dari entitas ini.
2. *nik* merupakan nomor identitas dari *surveyor* yang mendaftar pada aplikasi.
3. *Full\_name* merupakan nama lengkap *surveyor*.
4. *gender* merupakan jenis kelamin dari *surveyor* tersebut.
5. *Email* merupakan surel dari *surveyor* yang mendaftar.
6. *Phone* merupakan nomor *handphone* dari *surveyor* yang mendaftar.

Entitas *Report* merupakan representasi data laporan yang dilaporkan oleh *surveyor* pada saat di lapangan. Entitas *Report* ini memiliki atribut sebagai berikut:

1. *id* merupakan *primary key* dari entitas ini.
2. *lat* merupakan nama data koordinat *latitude* lokasi pelaporan.
3. *lng* merupakan nama data koordinat *longitude* lokasi pelaporan.
4. *Job\_name* merupakan nama pekerjaan saat ini.
5. *Duration* merupakan lama pengerjaan proses pelaporan pada saat di lapangan.
6. *Description* merupakan deskripsi singkat pelaporan di lapangan.
7. *Contract\_number* merupakan representasi nomor kontrak laporan *surveyor* dan satu laporan mempunyai satu nomor kontrak yang berbeda.

8. *Images* merupakan gambar yang diambil pada saat pelaporan di lapangan.

#### 4.4.2. Use Case Diagram



Gambar 4.3. Use Case Diagram MOTS-PeR

Pada Gambar 4.3 menunjukkan *use case diagram* Mots-PeR. Mots-PeR memiliki 2 level pengguna, yakni Admin dan *Surveyor*. Admin memiliki hak akses pada *use case* mengelola data *surveyor* dan mengelola laporan *surveyor*, sedangkan *Surveyor* hanya memiliki akses untuk pengadaan laporan saja.

##### 4.4.1.1 Otentifikasi Pengguna

<i>ID Requirement</i>	:	FR 001
Deskripsi	:	Dalam fungsi <i>log in</i> , pengguna dapat masuk ke dalam sistem

		sesuai dengan perannya masing-masing. Pengguna yang dapat menggunakan fungsi ini adalah <i>surveyor</i> dengan cara mendaftarkan diri terlebih dahulu dan admin yang sudah didaftarkan oleh pengembang.
<i>Validity Check</i>	:	<ul style="list-style-type: none"> <li>• Nama pengguna terdiri dari 1-255 karakter alfanumerik. Simbol dan spasi di antaranya tidak diperbolehkan</li> <li>• Kata sandi terdiri dari 1-255 karakter alfanumerik. Simbol dan spasi di antaranya diperbolehkan.</li> </ul>
Rasional	:	Autentikasi mengamankan fitur-fitur internal MOTS-PeR agar tidak terekspos ke pihak-pihak yang tidak memiliki akses.
Referensi	:	-

#### 4.4.1.2 Membuat Akun Baru

<i>ID Requirement</i>	:	FR 002
Deskripsi	:	Dalam fungsi ini, pengguna dapat mengisikan data diri untuk didaftarkan ke dalam sistem. Pengguna yang dapat membuat akun baru yaitu <i>surveyor</i> .
<i>Validity Check</i>	:	<ul style="list-style-type: none"> <li>• Nama pengguna terdiri dari 1-255 karakter alfanumerik. Simbol dan spasi di antaranya tidak diperbolehkan</li> <li>• Nama lengkap terdiri dari 1-255 karakter alfanumerik. Simbol dan spasi di antaranya tidak diperbolehkan.</li> <li>• Jenis Kelamin Pengguna memilih antara laki-laki dan perempuan.</li> <li>• Nomor KTP terdiri dari 16 digit angka numerik dan</li> </ul>

		<p>spasi tidak diperbolehkan.</p> <ul style="list-style-type: none"> <li>• Nomor HP terdiri dari 8 hingga 12 digit, yang diperbolehkan angka 0 hingga 9. Huruf, karakter khusus, dan spasi tidak diperbolehkan.</li> <li>• Email terdiri 3 hingga 50 karakter. Spasi tidak diperbolehkan.</li> <li>• Unit kerja terdiri dari 1-255 karakter alfanumerik. Simbol dan spasi di antaranya tidak diperbolehkan</li> <li>• Posisi Pekerjaan terdiri dari 1-255 karakter alfanumerik. Simbol dan spasi di antaranya tidak diperbolehkan</li> <li>• Kata sandi terdiri dari 1-255 karakter alfanumerik. Simbol dan spasi di antaranya diperbolehkan.</li> <li>• Konfirmasi kata sandi terdiri dari 1-255 karakter alfanumerik. Simbol dan spasi di antaranya diperbolehkan.</li> </ul>
Rasional	:	Fungsi ini dapat menambahkan data akun baru yang telah didaftarkan.
Referensi	:	-

#### 4.4.1.3 Dashboard Surveyor

<i>ID Requirement</i>	:	FR 003
Deskripsi	:	Dalam fungsi ini pengguna dapat melihat rangkuman data pribadinya, melihat lokasi saat ini, dan dapat membuat laporan.
<i>Validity Check</i>	:	<ul style="list-style-type: none"> <li>• ID: alfanumerik dengan panjang 3 hingga 10 karakter, merupakan kombinasi angka dan huruf. Karakter khusus tidak diperbolehkan.</li> </ul>

		<ul style="list-style-type: none"> <li>• Posisi Pekerjaan: alfanumerik dengan panjang 3 hingga 10 karakter, merupakan kombinasi angka dan huruf. Karakter khusus tidak diperbolehkan.</li> <li>• Unit Pekerjaan: alfanumerik dengan panjang 3 hingga 10 karakter, merupakan kombinasi angka dan huruf. Karakter khusus tidak diperbolehkan.</li> </ul>
Rasional	:	Fungsi ini dapat membantu pengguna untuk melihat data pribadi <i>surveyor</i> pada sistem.
Referensi	:	-

#### 4.4.1.4 OTS Report

<i>ID Requirement</i>	:	FR 004
Deskripsi	:	Dalam fungsi ini <i>surveyor</i> dapat melakukan pelaporan pada saat di lapangan.
<i>Validity Check</i>	:	<ul style="list-style-type: none"> <li>• Nomor kontrak: alfanumerik dengan panjang 3 hingga 10 karakter, merupakan kombinasi angka dan huruf. Karakter khusus tidak diperbolehkan.</li> <li>• Nama Pekerjaan: alfanumerik dengan panjang 3 hingga 10 karakter, merupakan kombinasi angka dan huruf. Karakter khusus tidak diperbolehkan.</li> <li>• Deskripsi: alfanumerik dengan panjang 50 hingga 255 karakter, merupakan kombinasi angka dan huruf. Karakter khusus diperbolehkan.</li> <li>• Lama Bekerja: numerik dari angka 1 – 8. Karakter khusus tidak diperbolehkan.</li> <li>• Gambar: masukan berupa gambar yang ditangkap melalui kamera.</li> </ul>
Rasional	:	Fungsi ini dapat membantu pengguna untuk melaporkan

	:	kejadian yang ada di lapangan.
Referensi	:	-

#### 4.4.1.5 Melihat Daftar Laporan

<i>ID Requirement</i>	:	FR 005
Deskripsi	:	Dalam fungsi ini pengguna dapat melihat laporan yang sudah pernah dibuat sebelumnya.
<i>Validity Check</i>	:	-
Rasional	:	Fungsi ini dapat membantu <i>surveyor</i> melihat riwayat laporan yang sudah pernah dibuat sebelumnya.
Referensi	:	-

#### 4.4.1.6 Melihat Detail Laporan

<i>ID Requirement</i>	:	FR 006
Deskripsi	:	Dalam fungsi ini pengguna dapat melihat detail laporan yang sudah pernah dibuat sebelumnya.
<i>Validity Check</i>	:	-
Rasional	:	Fungsi ini dapat membantu <i>surveyor</i> melihat detail laporan yang sudah pernah dibuat sebelumnya.
Referensi	:	-

#### 4.4.1.7 Melihat Daftar Seluruh Laporan *Surveyor*

<i>ID Requirement</i>	:	FR 007
Deskripsi	:	Dalam fungsi ini admin dapat melihat daftar seluruh laporan yang masuk dari <i>surveyor</i> yang telah mendaftar.

<i>Validity Check</i>	:	-
Rasional	:	Fungsi ini dapat membantu admin untuk mengetahui siapa saja yang sudah mengumpulkan laporan.
Referensi	:	-

#### 4.4.1.8 Melihat Daftar Seluruh *Surveyor*

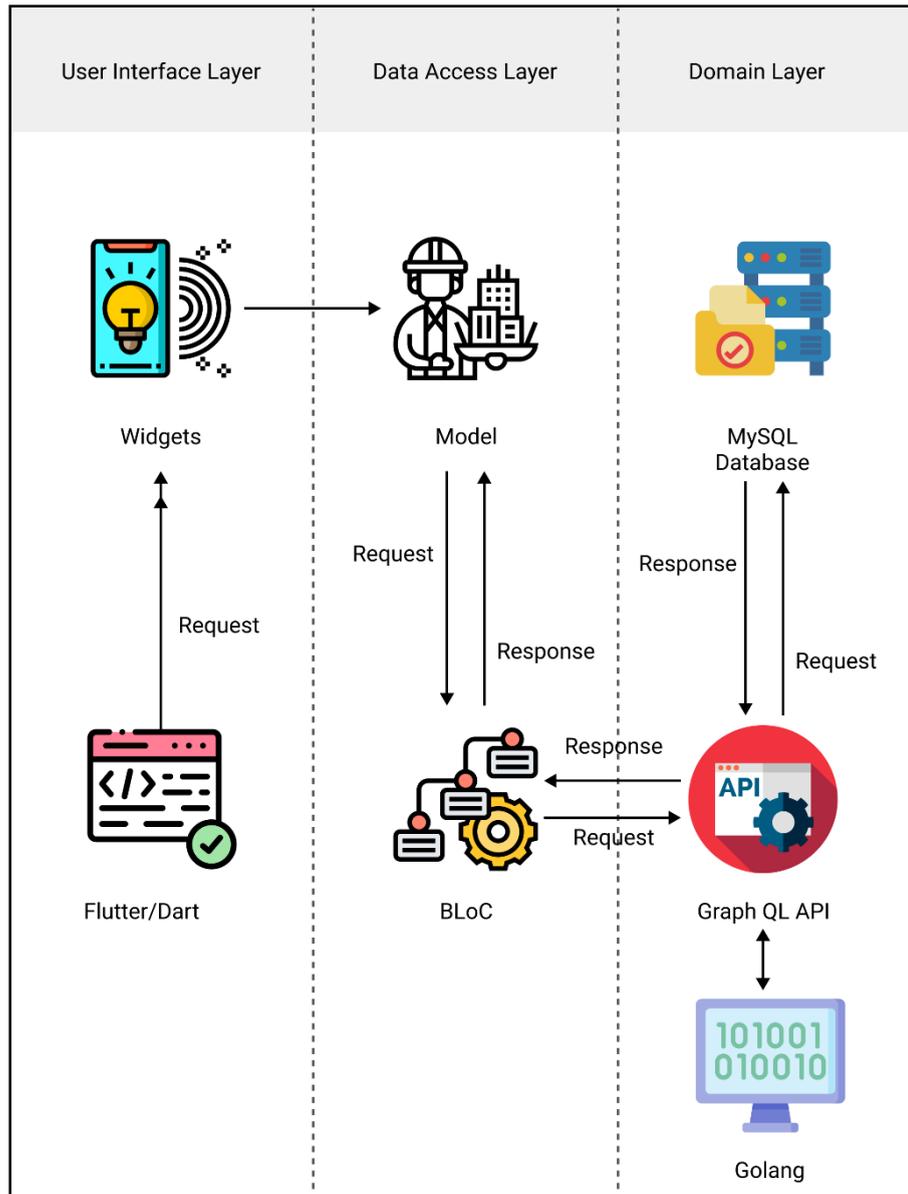
<i>ID Requirement</i>	:	FR 008
Deskripsi	:	Dalam fungsi ini admin dapat melihat daftar seluruh laporan yang masuk dari <i>surveyor</i> yang telah mendaftar.
<i>Validity Check</i>	:	-
Rasional	:	Fungsi ini dapat membantu admin untuk mengetahui siapa saja yang sudah mengumpulkan laporan.
Referensi	:	-

#### 4.4.1.9 Pengelolaan Data Pengguna

<i>ID Requirement</i>	:	FR 009
Deskripsi	:	Dalam fungsi ini admin dapat melakukan <i>activate/deactivate surveyor</i> .
<i>Validity Check</i>	:	-
Rasional	:	Fungsi ini dapat membantu admin untuk mengontrol pengguna apabila pengguna tersebut melakukan kecurangan dengan cara <i>activate/deactivate</i> .
Referensi	:	-

## 4.5. Perancangan Sistem

### 4.5.1. Arsitektur Sistem



Gambar 4.4 Arsitektur Sistem MOTS-PeR

Rancangan arsitektur MOTS-PeR dapat dilihat pada Gambar 4.4. Terdapat 3 komponen utama dari MOTS-PeR, yakni MOTS API (*back-*

*end*), BLoC (*Business Logic Component*), dan *Widget*. MOTS API merupakan pusat pengolahan data yang bertanggung jawab pada manajemen data-data utama, pengelolaan data *surveyor*, dan pengelolaan data laporan.

MOTS API dibangun dengan arsitektur MVC (*Model View Controller*) menjadi GraphQL API (*Query Language Application Programming Interface*). Basis data MOTS API menggunakan MariaDB 10.3. MOTS-PeR *Mobile* dibangun dengan arsitektur BLoC (*Business Logic Component*). MOTS-PeR *Mobile* hanya berperan sebagai antarmuka grafis untuk mengakses GraphQL API. Melalui MOTS-PeR *Mobile* pengguna dapat mengoperasikan manajemen data utama, manajemen data laporan, dan manajemen data pengguna.

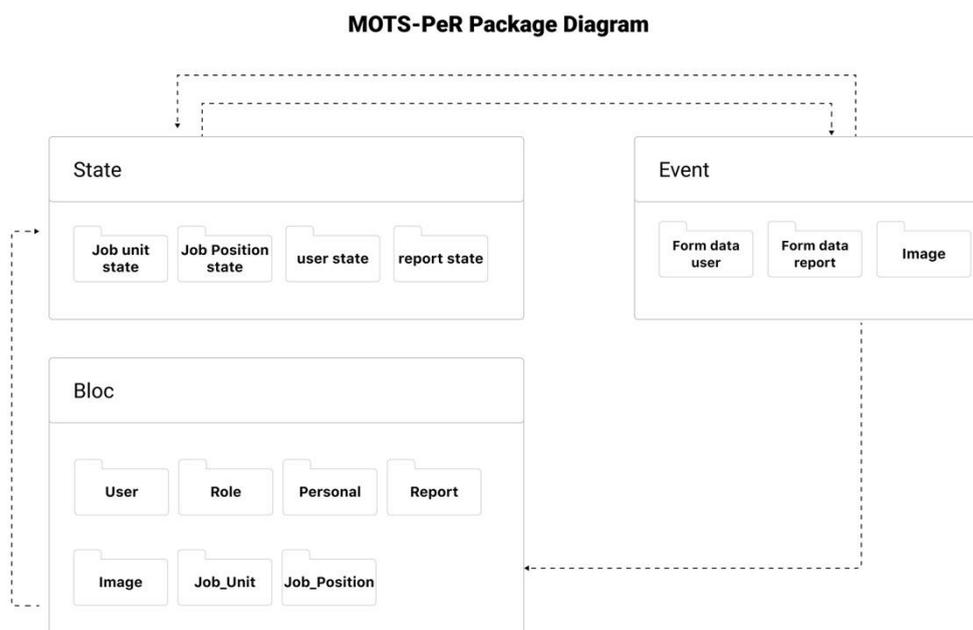
Komunikasi antara 3 komponen ini menggunakan protokol yang sama, yakni HTTP/HTTPS (*HyperText Transfer Protocol*). Dalam komunikasi ini, MOTS API akan berperan sebagai *server* yang akan menunggu *request* dari MOTS-PeR *Mobile*. *Request* ini memiliki data spesifik yang berisi tujuan dari *request* tersebut, berdasarkan *request* ini MOTS API akan memberikan *response* yang berisi data sesuai *request* yang datang. Format data yang digunakan adalah JSON (*JavaScript Object Notation*).

Dalam operasi internal MOTS API yang memiliki arsitektur MVC, *request* yang masuk akan diterima oleh *controller* untuk kemudian diekstraksi datanya. Data ini kemudian diberikan ke *service layer* tertentu sesuai konteks *request*. *Service layer* akan mengolah data dari *request* dan *model* untuk kemudian menghasilkan keluaran data yang sesuai. *Model* adalah komponen yang bertugas untuk melakukan interaksi dengan basis data (*querying*). Pemisahan tugas dalam arsitektur MVC ini akan memudahkan proses pengembangan dan memastikan tiap lapisan arsitektur ini dapat diuji dengan lebih teliti.

MOTS-PeR *Mobile* yang bertugas berinteraksi dengan MOTS API akan menerima masukan pengguna dalam bentuk antarmuka grafis seperti

*textbox*, *dropdown select*, dan *checkbox* untuk kemudian dikonversi menjadi bentuk JSON dan akan dikirim ke MOTS API. MOTS API akan memberikan *response* sesuai *request* yang dikirimkan oleh MOTS-PeR *Mobile*. Salah satu tugas krusial yang dilakukan oleh MOTS-PeR *Mobile* adalah mempresentasikan data primitif dari JSON menjadi bentuk yang lebih mudah dibaca oleh pengguna seperti menampilkan data dalam bentuk *table* dan *list*.

#### 4.5.2. Package Diagram

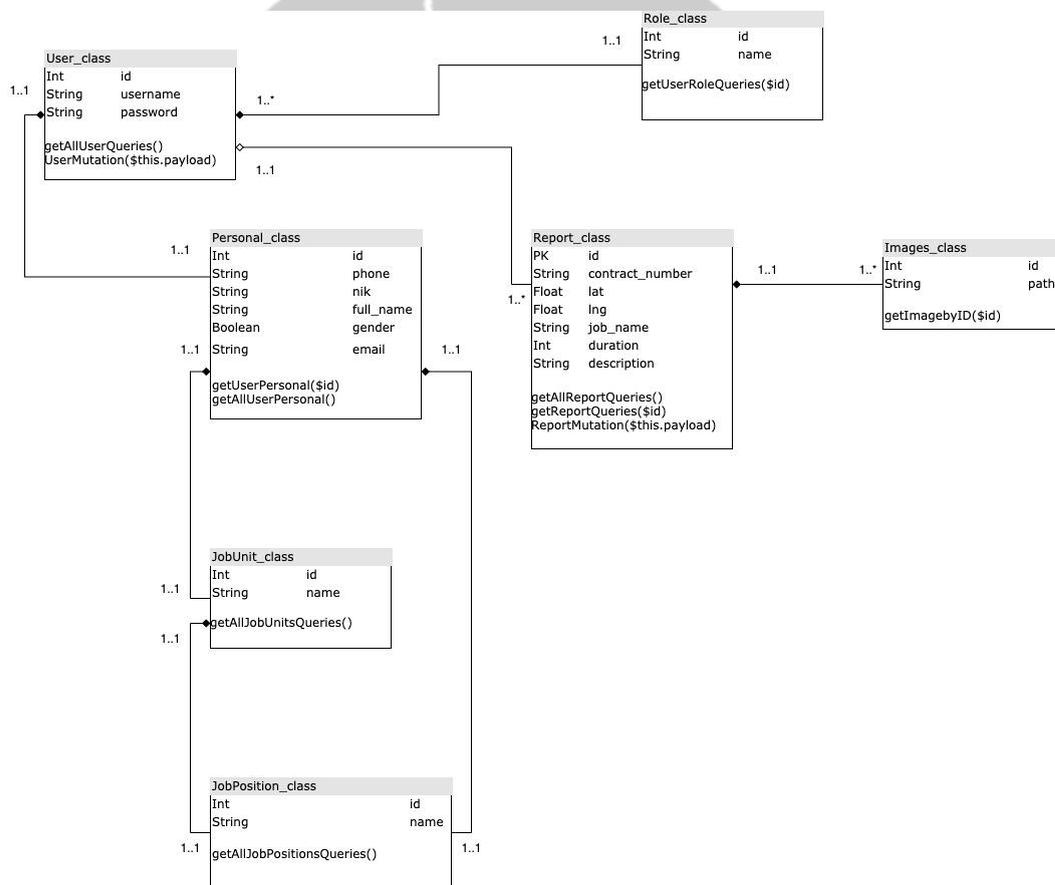


Gambar 4.5. Gambar Package Diagram MOTS-PeR

Pada gambar 4.5 menjelaskan hubungan antar komponen dalam arsitektur BLoC. Pada arsitektur BLoC terdapat 3 komponen utama yaitu *state*, *event* dan *Bloc*. Pada *package state* berisi *response* atas masukan dari pengguna. *Package state* terhubung dengan *package event* dan *bloc*. *Package state* dihubungkan dengan *package event* untuk melakukan pertukaran data dan *response* dari kedua *package* tersebut. Pada *package event* melakukan

masukan data kepada *package state*, lalu *package state* akan melakukan *response* atas masukan data tersebut sesuai dengan fungsi yang dijalankan. Pada *package Bloc* yang terhubung dengan *state* dan *event*, akan mengolah *output* dari masing-masing *package* tersebut menjadi *stream*. *Stream* tersebut berguna untuk melakukan komunikasi antar komponen yang ada didalam BLoC. Komponen tersebut akan eksekusi oleh program menjadi sebuah antarmuka sehingga aplikasi dapat diakses oleh pengguna.

### 4.5.3. Kelas Diagram



Gambar 4.6. Gambar Kelas Diagram MOTS-PeR

Pada gambar 4.6 memperlihatkan struktur dan hubungan diagram kelas pada aplikasi MOTS-PeR. Struktur kelas pada gambar 4.6 meliputi nama kelas, atribut kelas (variabel), tipe data, fungsi yang terdapat dalam kelas tersebut dan relasi antar kelas. Dalam alur bisnis utama sistem ini,

kelas yang memegang peranan penting adalah *User*, *Personal*, dan *Report*. Kombinasi data dari 3 kelas tersebut yang akan menghasilkan laporan surveyor pada aplikasi MOTS-PeR .

Kelas *User* yang didukung oleh kelas *role* dan kelas *personal* merupakan kelas yang merepresentasikan data Pengguna pada aplikasi MOTS-PeR. kelas *User* nantinya akan memiliki detail dari *user* yang sudah mendaftarkan diri pada aplikasi dan mempunyai *role* untuk masing-masing user. Kelas user memiliki *method* utama yaitu `getAllUserQueries()` dan `UserMutation($this.payload)`. pada kelas `getAllUserQueries()` digunakan untuk mengambil seluruh user yang sudah dibedakan berdasarkan *role* nya masing-masing. Pada kelas `UserMutation($this.payload)` digunakan untuk melakukan penambahan pengguna yang dapat mengakses aplikasi berdasarkan *role* nya masing-masing.

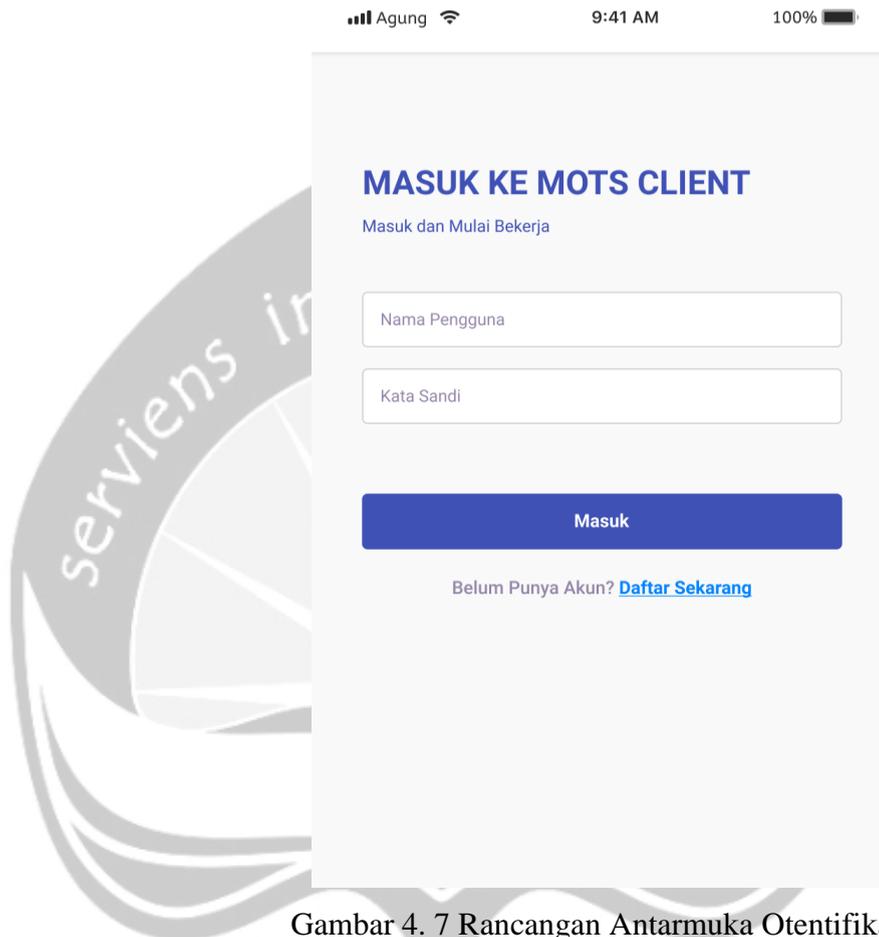
Kelas *Personal* merupakan representasi detail data *user*. Kelas ini memiliki fungsi utama yaitu `getUserPersonal($id)` yang digunakan untuk mengambil detail pengguna tertentu dan `getAllUserPersonal()` yang bisa mengambil detail seluruh pengguna yang suda terdaftar dalam aplikasi MOTS-PeR.

Kelas *Report* merupakan representasi data laporan yang dilaporkan oleh surveyor pada saat di lapangan. Entitas *Report* ini memiliki fungsi utama yaitu `getAllReportQueries()` yang berfungsi untuk mengambil seluruh data laporan pengguna, `getReportQueries($id)` berfungsi untuk mengambil detail laporan tertentu dan `ReportMutation(this.payload)` yang berfungsi untuk melakukan pelaporan dan menyimpan laporan ke dalam basis data.

#### 4.5.4. Deskripsi Perancangan Antarmuka

##### 4.5.2.1 MOTS-PeR Client

###### a) Otentifikasi Pengguna



Gambar 4. 7 Rancangan Antarmuka Otentifikasi

Pada Gambar 4.5 menunjukkan rancangan antarmuka Otentifikasi *surveyor* pada MOTS-PeR. Untuk memasuki sistem, pengguna perlu memberikan nama pengguna dan kata sandinya untuk dicocokkan dengan data pengguna yang dimiliki aplikasi. Tombol “Masuk” memiliki *event listener on-click* yang ketika ditekan sistem akan memulai proses autentikasi. Proses autentikasi akan dimulai dari *query* seperti di bawah ini.

```
mutation Authenticate($username: String, $password: String) { authenticate(username: $username, password: $password) }
```

Apabila terdapat data pengguna yang ditemukan, sistem akan mencocokkan kata sandi yang telah dimasukkan dengan kata sandi dari data pengguna yang ditemukan. Apabila kata sandi yang dimasukkan cocok dengan data pada sistem, maka pengguna akan diarahkan ke halaman *Dashboard* dan apabila tidak cocok maka pengguna akan diberikan peringatan bahwa nama pengguna/kata sandi salah atau tidak ditemukan.



## b) *Register Surveyor*

Agung 9:41 AM 100%

### DAFTAR KE MOTS CLIENT

Daftarkan diri anda dan Mulai Bekerja

Nama Pengguna

Nama Lengkap

Jenis Kelamin

Laki-laki  Perempuan

Unit Kerja

Posisi

Email

KTP

Nomor Telepon

Kata sandi

Konfirmasi Kata sandi

**Daftar**

Sudah Punya Akun? [Login Sekarang](#)

Gambar 4.8 Rancangan Antarmuka Halaman *Register*

Pada Gambar 4.6 menunjukkan rancangan antarmuka halaman *Register Surveyor*. Halaman ini berfungsi untuk pengguna baru yang ingin mendaftarkan diri secara mandiri ke dalam aplikasi. Tombol “Daftar” memiliki *event listener on-click* yang ketika ditekan sistem akan memulai proses pendaftaran dan akan mengecek apakah nama pengguna dan nomor KTP ada yang sama dengan pengguna lainnya. Proses pendaftaran akun dimulai dari ketika pengguna menekan tombol daftar dan mengeksekusi kode di bawah ini.

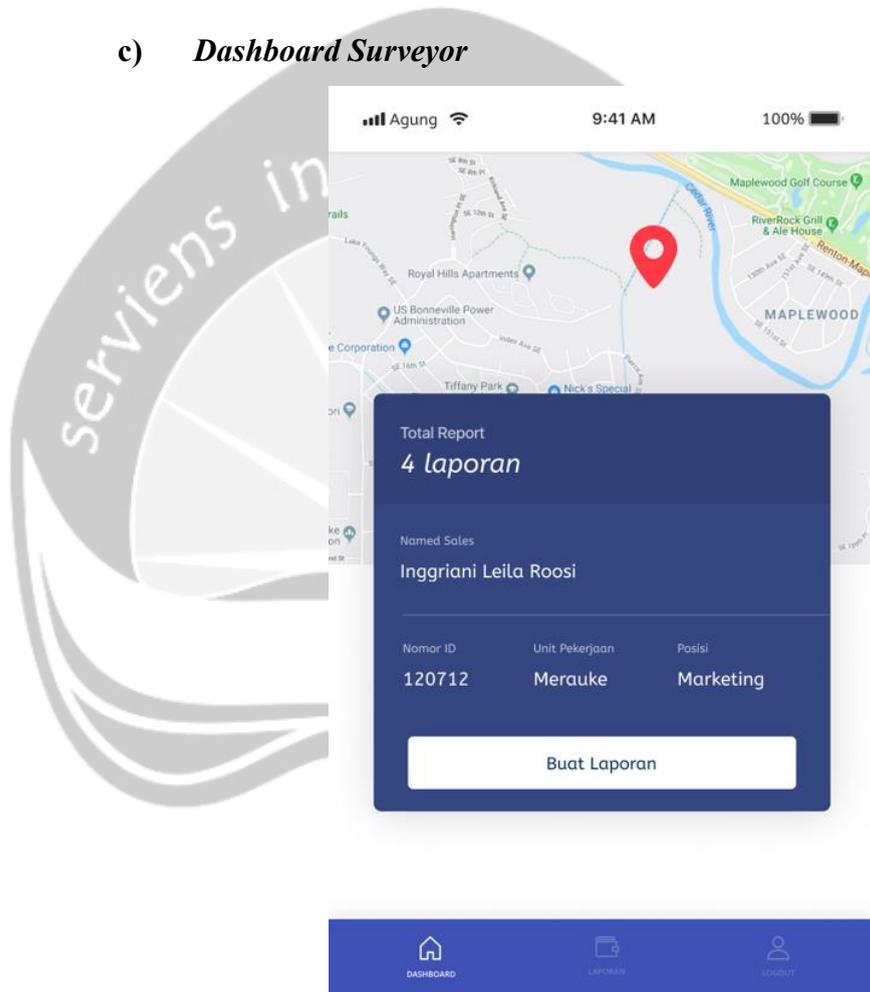
```
String RegisterMutation = r''  
mutation register(
```

```
$username: String,  
$password: String,  
$confirmPassword: String,  
$roleId: Int,  
$jobUnit: String,  
$jobPosition: String,  
$ktp: String,  
$fullName: String,  
$gender: String,  
$email: String,  
$phone: String  
) {  
  register (  
    username: $username,  
    password: $password,  
    confirmPassword: $confirmPassword,  
    roleId: $roleId,  
    jobUnit: $jobUnit,  
    jobPosition: $jobPosition,  
    ktp: $ktp,  
    fullName: $fullName,  
    gender: $gender,  
    email: $email,  
    phone: $phone  
  ) {  
    id,  
    username,  
    role {  
      id,  
      name  
    },  
    personal {  
      fullName,  
      email,  
      phone,  
      jobUnit,  
      jobPosition  
    }  
  }  
}
```

}}

Pada *query* tersebut akan melakukan *mutation* data yang ada pada *params register* dan akan melemparkan data yang di masukan oleh pengguna ke dalam basis data. Sebelum dimasukan ke dalam basis data, masukan dari pengguna akan diperiksa nama pengguna dan nomor KTP, apabila terdapat data yang sama dengan yang ada dalam basis data, maka akan mengeluarkan peringatan.

c) **Dashboard Surveyor**



Gambar 4.9 Rancangan Antarmuka Halaman *Dashboard Surveyor*

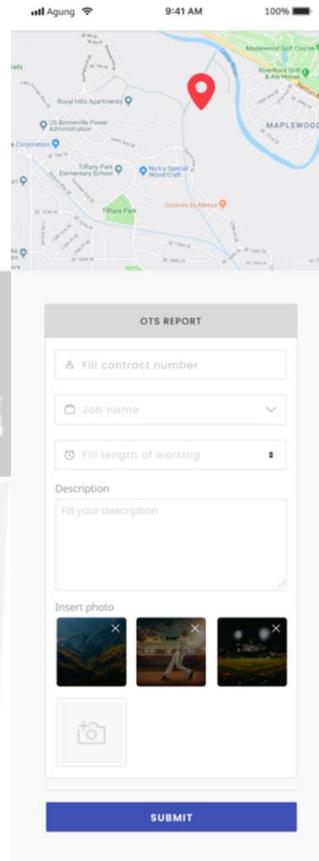
Gambar 4.7 menunjukkan rancangan antarmuka halaman *dashboard surveyor*. Pada halaman ini, *surveyor* dapat melihat data dirinya dan jumlah laporan yang sudah pernah dimasukan sebelumnya. Pada halaman ini juga terdapat CTA yang mengarahkan

*surveyor* untuk membuat laporan baru. Saat mengakses halaman ini, program akan mengeksekusi kode di bawah ini.

```
query whoami {
  whoami {
    id,
    isActive,
    username,
    personal {
      fullName,
      email,
      gender,
      ktp,
      phone,
      jobCount,
      jobPosition,
      jobUnit
    }
  }
}
```

Pada *query* tersebut mengambil data dari *end-point whoAmi* yang berisi data diri dan jumlah laporan yang telah dimasukkan sebelumnya.

d) *OTS Report*



Gambar 4.10 Rancangan Antarmuka Halaman *OTS Report*

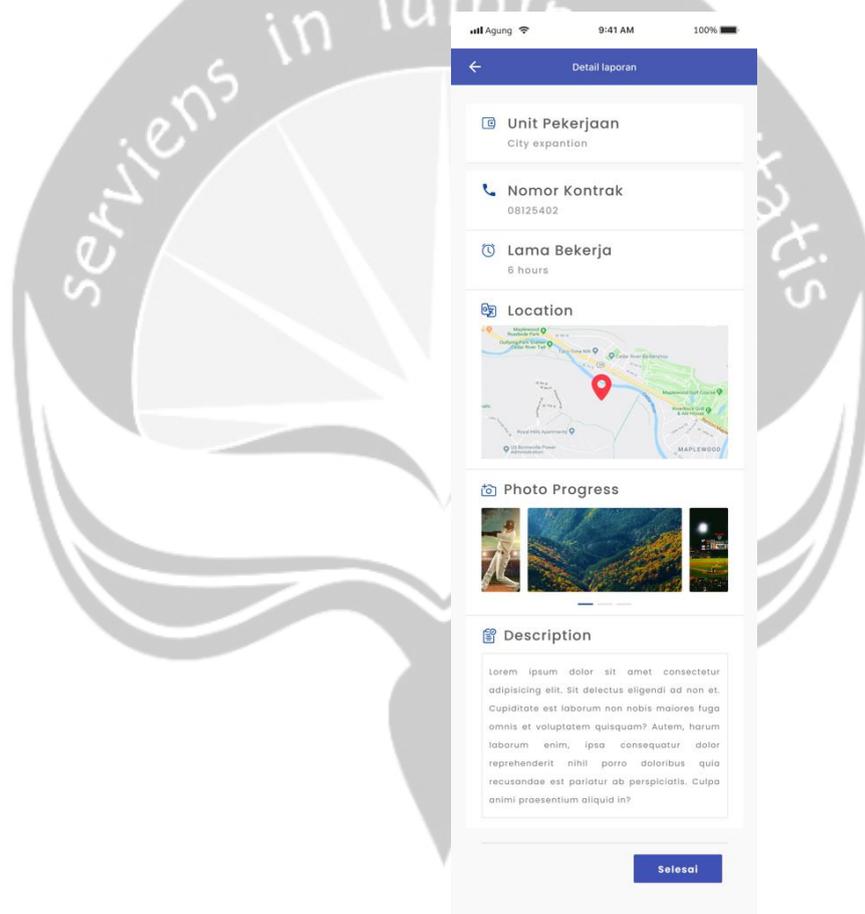
Pada Gambar 4.8 menunjukkan rancangan antarmuka halaman *OTS Report*. Satu laporan *surveyor* merepresentasikan satu nomor kontrak yang dimasukkan secara manual oleh *surveyor*. Nomor kontrak tersebut diberikan oleh admin pada saat pelaporan di lapangan. Pada halaman ini *surveyor* dapat melakukan pelaporan dan otomatis data lokasi *surveyor* pada saat melakukan pelaporan diambil *latitude* dan *longitude*-nya, sehingga kecurangan *surveyor* yang tidak melakukan pelaporan di lapangan dapat diatasi. Tombol “Submit” memiliki *event listener on-click* yang ketika ditekan sistem akan memulai proses pembuatan laporan dan akan mengecek apakah masukan *surveyor* ada yang kosong. Apabila terdapat masukan yang

kosong, maka akan mengeluarkan peringatan. Saat *surveyor* mengeksekusi pemasukan laporan, maka dilakukan eksekusi kode berikut ini:

```
CreateReportMutation = r'''
mutation createReport(
  $id: String!,
  $lat: Float!,
  $lng: Float!,
  $jobName: String!,
  $duration: Int!,
  $description: String!,
  $contractNumber: String!,
  $images: [ImageInput]!
) {
  createReport (
    id: $id,
    lat: $lat,
    lng: $lng,
    jobName: $jobName,
    duration: $duration,
    description: $description,
    contractNumber: $contractNumber,
    images: $images
  ) {
    id,
    jobName,
    images {
      mimeType,
      fileName,
      url
    },
    pengguna {
      id,
      username
    }
  }
}
```

Pada *query* tersebut akan melakukan mutasi data yang ada pada *params createReport* dan akan melemparkan data yang dimasukan oleh pengguna ke dalam basis data. Sebelum dimasukan ke dalam basis data, masukan dari pengguna akan diperiksa apakah terdapat masukan yang kosong atau tidak. Apabila terdapat data yang kosong maka data tidak akan dimasukkan ke dalam basis data dan aplikasi akan menampilkan peringatan bahwa terdapat data kosong.

e) **Melihat Detail Laporan**



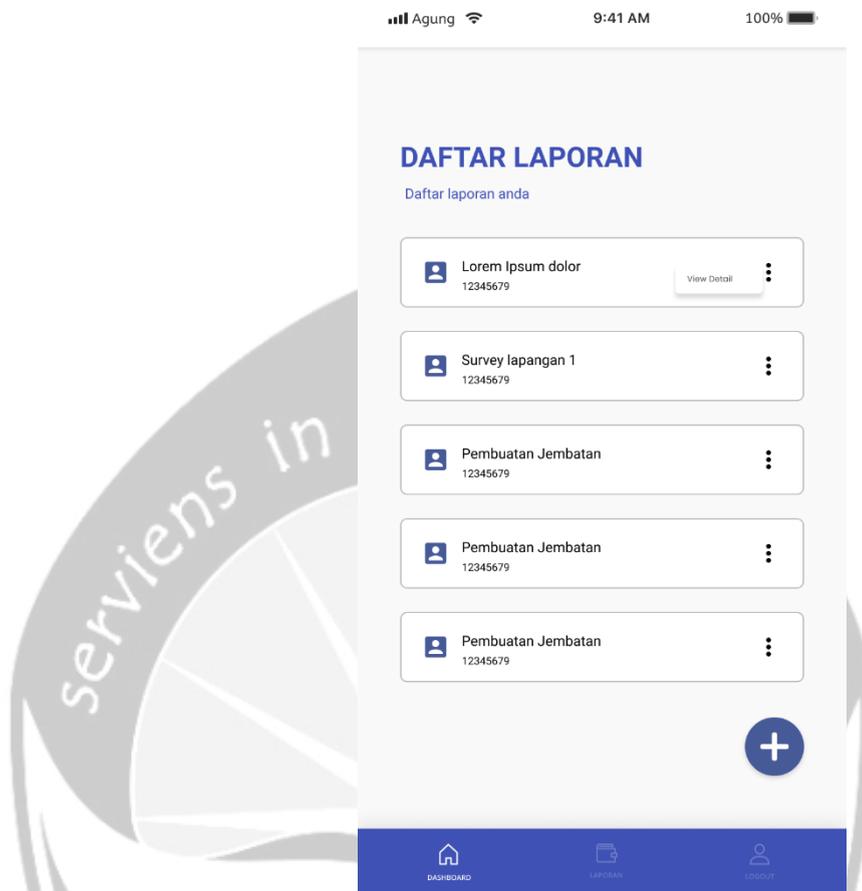
Gambar 4.11 Rancangan Antarmuka Halaman Detail Laporan

Gambar 4.9 menunjukkan rancangan antarmuka halaman detail laporan. Pada halaman ini *surveyor* dapat melihat detail laporan yang pernah dia masukan sebelumnya. Saat mengakses halaman ini, program akan mengeksekusi kode berikut ini:

```
GetMyReportsQuery = r'''
    query getMyReports {
      myReports {
        id,
        contractNumber,
        jobName,
        duration,
        description,
        images {
          url
        }
        lat
        lng
      }
    }
  '''
```

Pada *query* tersebut mengambil data dari *end-point* `GetMyReportQuery` yang berisi data laporan yang telah dimasukkan sebelumnya.

## f) Melihat *History* Laporan



Gambar 4.12 Rancangan Antarmuka Halaman *History* Laporan

Gambar 4.10. menunjukkan rancangan antarmuka halaman *history* laporan. Pada halaman ini *surveyor* dapat melihat *history* laporan yang pernah dia masukan sebelumnya. Saat mengakses halaman ini, program akan mengeksekusi kode berikut ini:

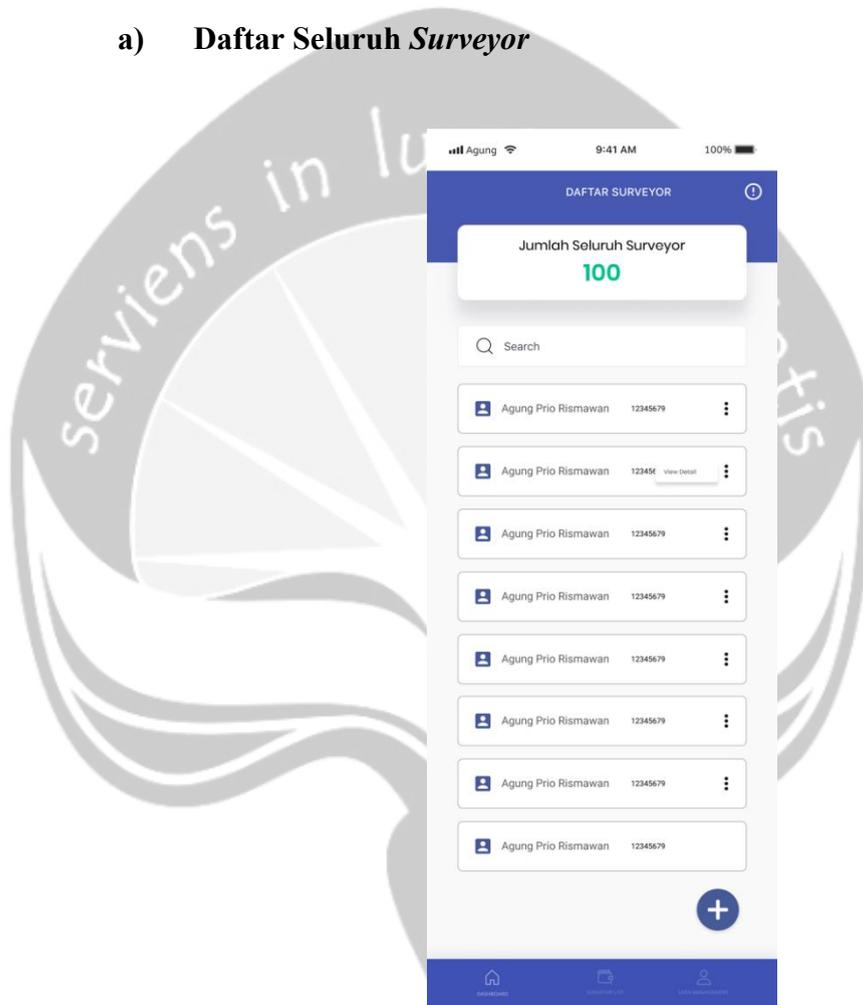
```
GetAllJobsQuery = r'''
query {
  jobUnits{
    id,
    name
  }
}
```

}

Query di atas berfungsi untuk mengambil data *history* dari *endpoint* `getAllJobsQuery` yang kembaliannya adalah id dan nama pekerjaan yang sedang dikerjakan dan akan ditampilkan dalam bentuk *list* pada aplikasi.

#### 4.5.2.2 MOTS-PeR Admin

##### a) Daftar Seluruh *Surveyor*



Gambar 4.13 Rancangan Antarmuka Halaman Daftar Seluruh *Surveyor*

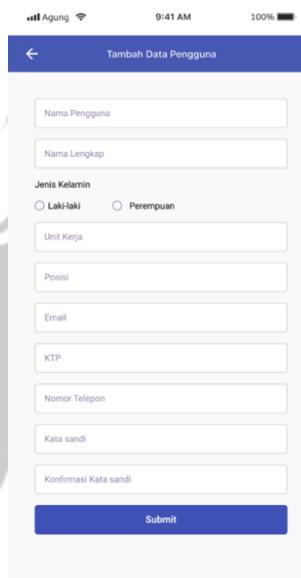
Pada gambar 4.11 menunjukkan rancangan antarmuka halaman daftar seluruh *surveyor*. Pada halaman ini, admin dapat melihat daftar seluruh *surveyor* yang terdaftar di dalam aplikasi ini, jumlah

*surveyor* yang terdaftar dan bisa mencari *surveyor* berdasarkan nama dan id. Saat mengakses halaman ini, program akan mengeksekusi kode berikut ini:

```
getAllPenggunaQuery = r'''
query {
  pengguna {
    id
    username
  }
}
'''
```

Query di atas berfungsi untuk mengambil data seluruh *surveyor* dari *endpoint* `getAllPenggunaQuery` yang mengembalikan id dan nama *surveyor* dan ditampilkan dalam bentuk *list*.

#### b) **Tambah Data *Surveyor***



The image shows a mobile application interface for adding a user. The screen is titled "Tambah Data Pengguna" and contains several input fields: "Nama Pengguna", "Nama Lengkap", "Jenis Kelamin" (with radio buttons for "Laki-laki" and "Perempuan"), "Unit Kerja", "Posisi", "Email", "KTP", "Nomor Telepon", "Kata sandi", and "Konfirmasi Kata sandi". A blue "Submit" button is at the bottom.

Gambar 4.14 Rancangan Antarmuka Halaman Tambah Data *Surveyor*

Pada Gambar 4.12 menunjukkan rancangan antarmuka

halaman tambah *surveyor*. Pada halaman ini, admin dapat menambahkan *surveyor* secara manual di dalam aplikasi. Saat mengakses halaman ini, program akan mengeksekusi kode berikut ini:

```
String AddPegunaMutation = r'''
mutation AddPeguna(
  $username: String!,
  $password: String!,
  $confirmPassword: String!,
  $roleId: Int!,
  $jobUnit: String!,
  $jobPosition: String!,
  $ktp: String!,
  $fullName: String!,
  $gender: String!,
  $email: String!,
  $phone: String!
) {
  AddPeguna (
    username: $username,
    password: $password,
    confirmPassword: $confirmPassword,
    roleId: $roleId,
    jobUnit: $jobUnit,
    jobPosition: $jobPosition,
    ktp: $ktp,
    fullName: $fullName,
    gender: $gender,
    email: $email,
    phone: $phone
  ) {
    id,
    username,
    role {
      id,
      name
    }
  }
}
```

```

    },
    personal {
      fullName,
      email,
      phone,
      jobUnit,
      jobPosition
    }
  }
}

```

Pada *query* tersebut akan melakukan mutasi data yang ada pada *params* `addPengguna` dan akan melemparkan data yang dimasukan oleh admin ke dalam basis data. Sebelum dimasukkan ke dalam basis data, masukan dari pengguna akan diperiksa nama pengguna dan nomor KTP. Apabila terdapat data yang sama dengan yang ada dalam basis data, maka akan mengeluarkan peringatan.

### c) Detail Data Pengguna

Field	Value
Nama Pengguna	agungskak22
Nama Lengkap	Agung prio rismawan
Jenis Kelamin	Laki-Laki
Unit Kerja	Apa Saja Boleh
Posisi	Apa Saja Boleh
Email	agungskak22@gmail.com
KTP	1607090010101010
Nomor Telepon	08121433

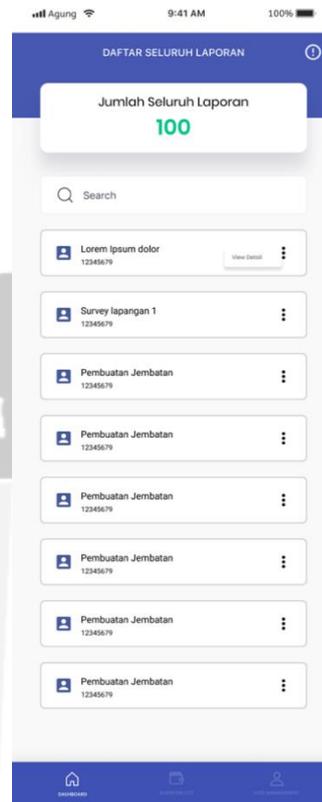
Gambar 4.15 Rancangan Antarmuka Halaman Detail *Surveyor*

Pada Gambar 4.13 menunjukkan rancangan antarmuka halaman detail *surveyor*. Pada halaman ini, admin dapat melihat detail *surveyor* yang sudah ditambahkan. Saat mengakses halaman ini, program akan mengeksekusi kode berikut ini:

```
query GetPeggunaDetail {
  GetPeggunaDetail {
    id,
    isActive,
    username,
    personal {
      fullName,
      email,
      gender,
      ktp,
      phone,
      jobCount,
      jobPosition,
      jobUnit
    }
  }
}
```

*Query* di atas berfungsi untuk mengambil data dari *end-point* *GetPeggunaDetail* yang berisi data diri *surveyor* dan status. Saat pertama kali didaftarkan, status *surveyor* belum aktif dan harus diaktifkan terlebih dahulu pada menu pengelolaan data *surveyor*.

#### d) Daftar Seluruh Laporan



Gambar 4.16 Rancangan Halaman Daftar Seluruh Laporan *Surveyor*

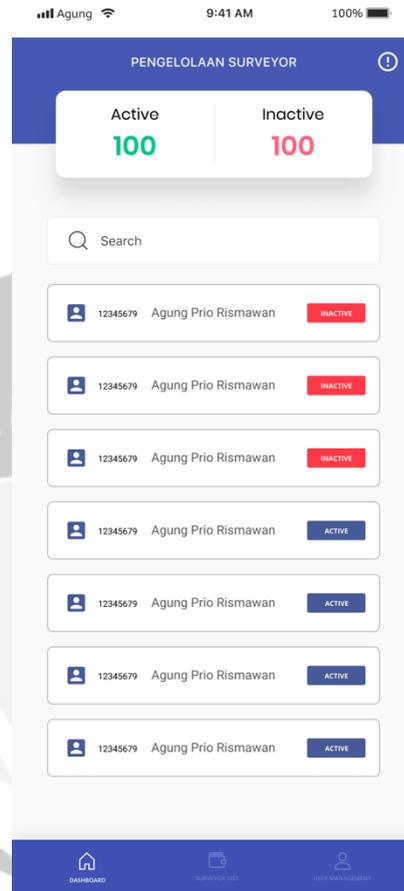
Gambar 4.14 menunjukkan rancangan halaman daftar seluruh laporan yang sudah dimasukkan oleh *surveyor* sebelumnya. Pada halaman ini, admin dapat mencari laporan berdasarkan nama pekerjaan atau nomor kontrak pada *form* cari, selain itu admin juga dapat mengetahui berapa jumlah laporan yang sudah dimasukan sebelumnya. Pada saat mengakses halaman ini, aplikasi akan mengeksekusi kode berikut ini:

```
AllReportsQueries = r'''
query{
  reports{
    contractNumber,
    description,
```

```
duration,  
id,  
images{  
  fileName,  
  mimeType,  
  url  
},  
jobName,  
lat,  
lng,  
pengguna {  
  id  
  personal {  
    fullName  
  }  
}
```

*Query* di atas berfungsi untuk mengambil data seluruh laporan dari *end-point* AllReportsQuery. Kembalian dari *endpoint* tersebut adalah objek laporan pengguna yang berisi data-data laporan yang ditampilkan dalam bentuk *list*.

e) *Activated/Deactivated Surveyor*



Gambar 4.17 Rancangan Halaman *Activated/Deactivated Surveyor*

Gambar 4.15 menunjukkan rancangan halaman *Activated/Deactivated Surveyor*. Menu ini digunakan untuk mengaktifkan dan menonaktifkan *surveyor* apabila terdapat kecurangan saat bekerja di lapangan. Pada halaman ini, admin juga dapat melihat jumlah *surveyor* yang aktif dan tidak aktif. Selain itu, admin dapat mencari berdasarkan nama dan id *surveyor* pada *form* cari. Menu ini juga digunakan oleh admin apabila terdapat *surveyor* baru pada saat *surveyor* melakukan pendaftaran pada MOTS Client, karena saat pertama kali mendaftar status *surveyor* diset belum aktif. Tombol “Active” dan “Deactive” memiliki *event listener on-click*

yang ketika ditekan sistem akan memulai proses *Activated/Deactivated Surveyor* dan mengeksekusi kode berikut ini:

```
ActivatePenggunaMutation = r'''
mutation($id: Int) {
  enablePengguna(id: $id)
},
DeactivatePenggunaMutation = r'''
mutation($id: Int) {
  enablePengguna(id: $id)
},
```

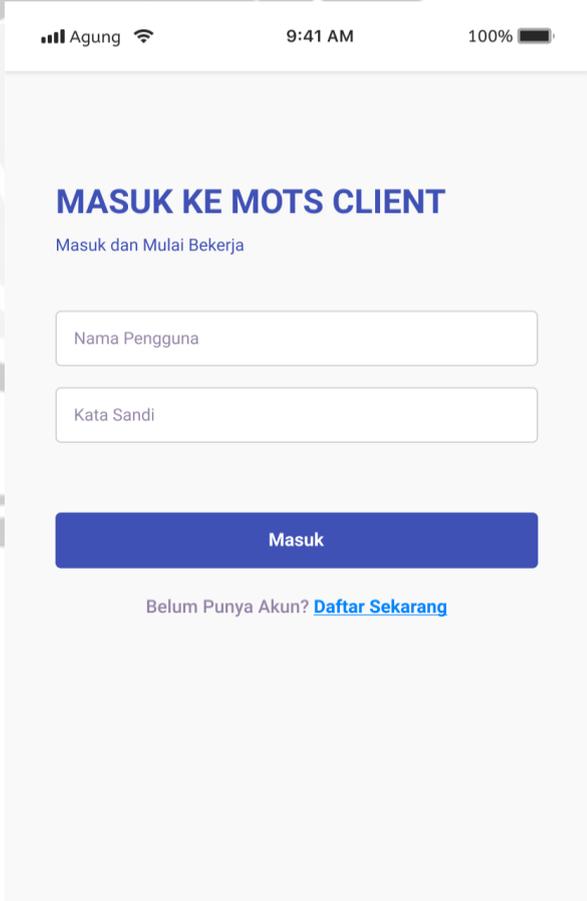
Pada saat menekan tombol *active* maka *state* dari *surveyor* yang dipilih akan berubah dan membuat *trigger* terhadap *end-point* *ActivatePenggunaMutation*. Lalu *surveyor* yang sebelumnya berstatus *nonactive* akan berubah menjadi aktif. Apabila admin menekan tombol *deactive* maka *state* dari *surveyor* yang dipilih akan berubah dan membuat *trigger* terhadap *end-point* *DeactivatePenggunaMutation*. Lalu *surveyor* yang sebelumnya berstatus aktif akan berubah menjadi *nonactive*.

## BAB V. IMPLEMENTASI DAN PENGUJIAN SISTEM

Bab ini akan membahas penggunaan sistem informasi penghasil berkas laporan yang meliputi implementasi dan pengujian perangkat lunak yang dibuat. Implementasi ini digunakan untuk menjelaskan dan mendeskripsikan bagian-bagian yang ada pada sistem. Pada aplikasi diberlakukan dua pengujian, yaitu pengujian kepada pengguna dan pengujian fungsionalitas aplikasi. Pengujian ini digunakan untuk menganalisis apakah sistem yang dibuat sudah memenuhi target yang ingin dicapai.

### 5.1. Implementasi Sistem

#### 5.1.1. Halaman *Login*

The image shows a mobile application login screen. At the top, the status bar displays 'Agung' with signal strength, Wi-Fi, the time '9:41 AM', and '100%' battery. The main content area has a light gray background. The title 'MASUK KE MOTS CLIENT' is in bold blue text. Below it, the subtitle 'Masuk dan Mulai Bekerja' is in a smaller blue font. There are two white input fields with rounded corners: the first is labeled 'Nama Pengguna' and the second is labeled 'Kata Sandi'. Below these fields is a solid blue button with the white text 'Masuk'. At the bottom, there is a link that says 'Belum Punya Akun? [Daftar Sekarang](#)'.

Gambar 5.1 Halaman *Login*

Gambar 5.1 merupakan implementasi *login* pada aplikasi *surveyor*. Pada proses *login* terdapat dua buah masukan yang diharapkan, yaitu nama

pengguna dan kata sandi. Ketika pengguna menekan tombol *login*, maka halaman *login* akan melakukan *API request* terhadap *endpoint* *authenticate* menggunakan BLoC dengan melakukan *dispatch* fungsi *event* `AuthenticatingStarted()`, seperti yang ditunjukkan pada Gambar 5.2.

```
), // TextStyle
), // Text
onPressed: () {
  authBloc.dispatch(AuthenticatingStarted());
},
```

Gambar 5.2 Kode Kelas *AuthenticatingStarted* pada *Login*

Pada kelas `AuthenticatingStarted()`, jika *API request* berhasil, maka API akan mengembalikan `accessToken` yang kemudian disimpan pada penyimpanan lokal *device surveyor*. Penyimpanan lokal bawaan Flutter ini disebut `localstorage` yang berfungsi sebagai tempat penyimpanan *token surveyor* yang *login*, *token* tersebut berfungsi sebagai identifikasi untuk setiap *surveyor* yang *login* dan bersifat unik. *Token* tersebut berisi data berupa nama pengguna, *id*, *role*, status, dan data personal *surveyor* tersebut. Selanjutnya API akan melakukan pengecekan. Jika nama pengguna dan kata sandi salah, maka API akan mengembalikan pesan *error* dan akan ditampilkan kepada pengguna melalui *snackbar*.

### 5.1.2. Halaman Register Surveyor

Agung 9:41 AM 100%

#### DAFTAR KE MOTS CLIENT

Daftarkan diri anda dan Mulai Bekerja

Nama Pengguna

Nama Lengkap

Jenis Kelamin

Laki-laki  Perempuan

Unit Kerja

Posisi

Email

KTP

Nomor Telepon

Kata sandi

Konfirmasi Kata sandi

Daftar

Sudah Punya Akun? [Login Sekarang](#)

Gambar 5.3 Halaman *Register Client*

Pada halaman *register* terdapat beberapa masukan yang wajib untuk diisi, ketika *user* menekan tombol daftar maka akan mengeksekusi kode pada gambar 5.4.

```
if (!state.isRegistering && state.isRegisteringSuccess) {  
  Navigator.pushNamed(context, '/');  
}
```

```

_handleRegisterStarted() async {
  try {
    var form = currentState.form;

    final MutationOptions options = MutationOptions(
      document: RegisterMutation,
      variables: <String, dynamic>{
        'username': form.username,
        'password': form.password,
        'confirmPassword': form.confirmPassword,
        'roleId': form.roleId,
        'jobUnit': form.jobUnit,
        'jobPosition': form.jobPosition,
        'ktp': form.ktp,
        'fullName': form.fullName,
        'gender': form.gender,
        'email': form.email,
        'phone': form.phone,
      },
    );

    QueryResult result = await _gqlClient.mutate(options);

    if (result.hasErrors) {
      print('ckck ' + result.errors.first.toString());
      this.dispatch(RegisterError(error: InvalidRegisterDataException()));
      return;
    }

    this.dispatch(RegisterSuccess());
  } catch (e) {
    print('ckck ' + e.toString());
    this.dispatch(RegisterError(error: RegistrationFailedException()));
  }
}

RegisterState _mapRegisterSuccess() =>
  currentState.copyWith(
    isRegistering: false,
    isRegisteringSuccess: true,
    form: RegisterForm.initial(),
  );

if (!state.isRegistering && state.isRegisteringSuccess) {
  Navigator.pushNamed(context, '/');
}

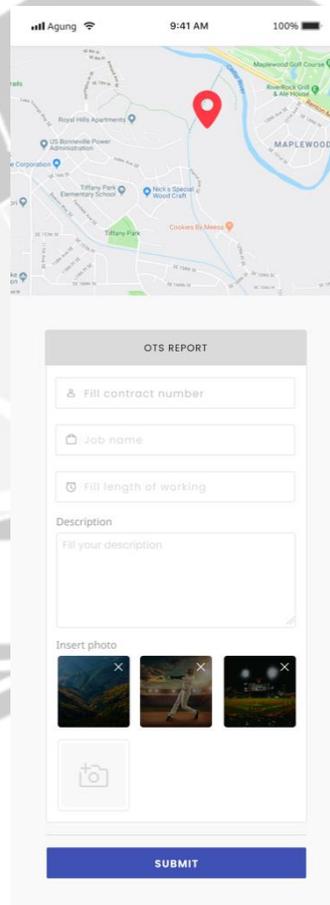
```

Gambar 5.4 Kode *Registration* BLoC

Pada potongan kode Gambar 5.4, data-data *surveyor* akan diinisialisasi pada BLoC `_handleRegisterStarted()`. Selanjutnya pada kelas `_handleRegisterStarted()`, data tersebut diberi nilai sesuai dengan *params* yang ada pada API lalu akan disimpan pada variabel `result`. Variabel `result` diberi kelas `await` yang menunggu proses pengisian *form* pada halaman pendaftaran. Apabila sudah selesai maka hasil isian tersebut akan dilakukan mutasi menggunakan fungsi `_gqlClient.mutate(options)`. Langkah selanjutnya adalah program

akan melakukan pengecekan apabila terdapat data yang kosong atau belum lengkap maka dimasukkan pada *exception* `invalidRegisterException()`. Apabila sukses maka akan memanggil BLoC *event* `registrationSuccess()` dan akan mengarahkan ke halaman *login*. Data dari *surveyor* yang mendaftar akan disimpan pada basis data.

### 5.1.3. Halaman Buat Laporan



Gambar 5.5 Halaman Membuat Laporan Baru

Pada halaman pembuatan laporan baru, menggunakan *plugin* yang disediakan oleh Flutter untuk menampilkan Google Maps. *Plugin* tersebut adalah `geolocator`. Pada halaman pembuatan laporan pertama kali dilakukan pemanggilan pada fungsi `geolocator` untuk mendapatkan data

lokasi terkini berupa koordinat *latitude* dan *longitude* menggunakan kode pada Gambar 5.6.

```
_geolocator.getCurrentPosition(desiredAccuracy: LocationAccuracy.best)
.then((Position position) async {
  final coordinates = new Coordinates(position.latitude, position.longitude);
  final addresses = await Geocoder.local.findAddressesFromCoordinates(coordinates);
  final cityName = addresses.first.locality;

  _currentLocationCtrl.sink.add(new Location(LatLng(position.latitude, position.longitude), cityName));
});
```

Gambar 5.6 Potongan Kode Geolocator

Pada Gambar 5.6, `geolocator` akan mengambil posisi terkini menggunakan fungsi `getCurrentPosition(desiredAccuracy: LocationAccuracy.best)`. Fungsi ini akan mengambil lokasi terbaik dari *surveyor* yang sedang *login*. Lokasi terbaik yang dimaksud adalah ketepatan akurasi pengambilan data *latitude* dan *longitude*, yaitu hingga 7 angka di belakang koma. Selanjutnya adalah proses *input* data laporan pada *form* yang sudah disediakan. Pada saat *surveyor* menekan tombol *submit*, maka terlebih dahulu dilakukan pengecekan atau validasi pada setiap *input field* yang ada. Validasi ini dilakukan dengan memanggil fungsi `isFormValid()` pada *file* `report_bloc.dart` seperti pada gambar 5.7.

```
bool _isFormValid(ReportForm form, List<File> files) {
  if (form.contractNumber.isEmpty || form.description.isEmpty || form.jobName.isEmpty) {
    this.dispatch(ReportStoringError(error: Exception('Data tidak boleh kosong')));
    return false;
  }

  if (form.duration <= 0) {
    this.dispatch(ReportStoringError(error: Exception('Durasi tidak valid')));
    return false;
  }

  if (files.length == 0) {
    this.dispatch(ImagesUploadingError(error: Exception('Gambar tidak boleh kosong')));
    return false;
  }

  return true;
}
```

Gambar 5.7 Kode Pengecekan `_isFormValid()` Pada Halaman *Input* Laporan

Pada potongan kode pada Gambar 5.7, akan dilakukan pengecekan

apabila data dari *form* yang dimasukan valid, maka proses *submit report* akan dilanjutkan dengan melakukan mutasi pada *endpoint* `createReport`. Pada potongan kode pada Gambar 5.7, terdapat contoh pengecekan *form* durasi yang tidak boleh berupa teks, kurang dari nol, dan gambar tidak boleh kosong.

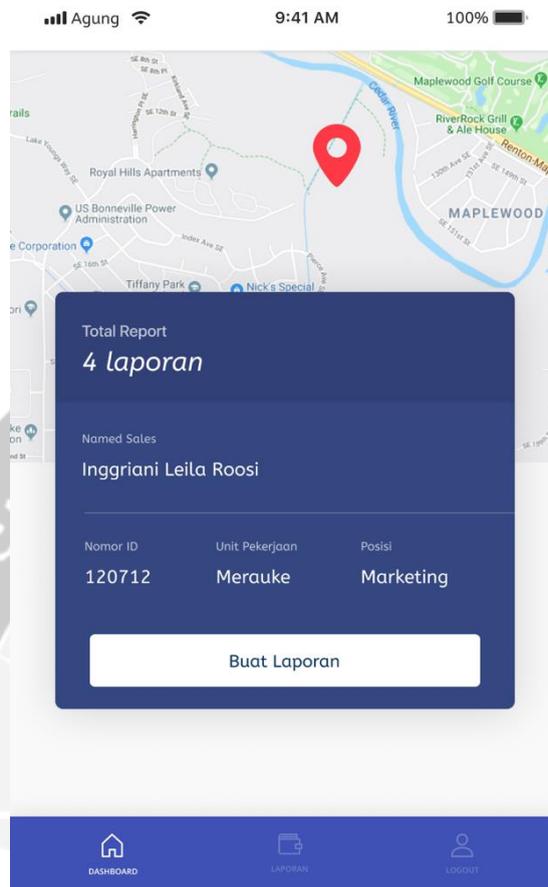
```
Future<ReportForm> _handleImagesUpload(ReportForm form, List<File> files) async {
  try {
    List<Future<Image>> imagesUploadFutures = files
      .map<Future<Image>>((item) {
        return _fileService.upload(item);
      })
      .toList();
    List<Image> images = await Future.wait(imagesUploadFutures)..toList();

    dispatch(ImagesUploadingSuccess());
    return form.copyWith(images: images);
  } catch (error) {
    dispatch(ImagesUploadingError(error: error));
    throw error;
  }
}
```

Gambar 5.8 Kode Untuk *Upload* Gambar

Pada Gambar 5.8 dijelaskan bahwa *upload* gambar dilakukan satu persatu dengan memanggil fungsi `_handleImagesUpload()`. Fungsi `_handleImagesUpload()` memiliki dua buah parameter, yaitu data laporan dan gambar. Data gambar diberikan tipe *array list*, dengan harapan gambar bisa dimasukkan lebih dari satu yang ditujukan pada *endpoint* `FileService`. *Endpoint* `FileService` akan memberikan *response* berupa `mimeType` dan `ImageLink` pada gambar yang sudah di-*upload*. Selanjutnya fungsi `_handleImagesUpload()` akan mengambil `mimeType` dan `ImageLink` pada *endpoint* `FileService` dan mengembalikan data gambar tersebut dalam bentuk *list* pada halaman detail laporan.

#### 5.1.4. Tampil Data *Surveyor*



Gambar 5.9 Halaman *Dashboard Surveyor*

Pada Gambar 5.9 halaman *dashboard client*, akan dilakukan API *Call* pada *endpoint* *whoami*. *Endpoint* ini akan mengembalikan data *user* dan data personal seperti nama, *id*, unit pekerjaan, posisi, dan data *report*-nya. Pada peta, tampilan *marker* pada peta di-*update* secara *realtime*. *Update* ini dilakukan pada *userOverviewBloc* dengan memanggil fungsi `_handleGetCurrentLocation()` seperti pada Gambar 5.10.

```

_handleGetCurrentLocation() async {
  Geolocator _geolocator = Geolocator()..forceAndroidLocationManager;
  await _geolocator.getCurrentPosition(desiredAccuracy: LocationAccuracy.best)
    .then((Position position) async {
      final coordinates = new Coordinates(position.latitude, position.longitude);
      final addresses = await Geocoder.local.findAddressesFromCoordinates(coordinates);
      final cityName = addresses.first.locality;

      final location = new model.Location(new LatLng(position.latitude, position.longitude), cityName);

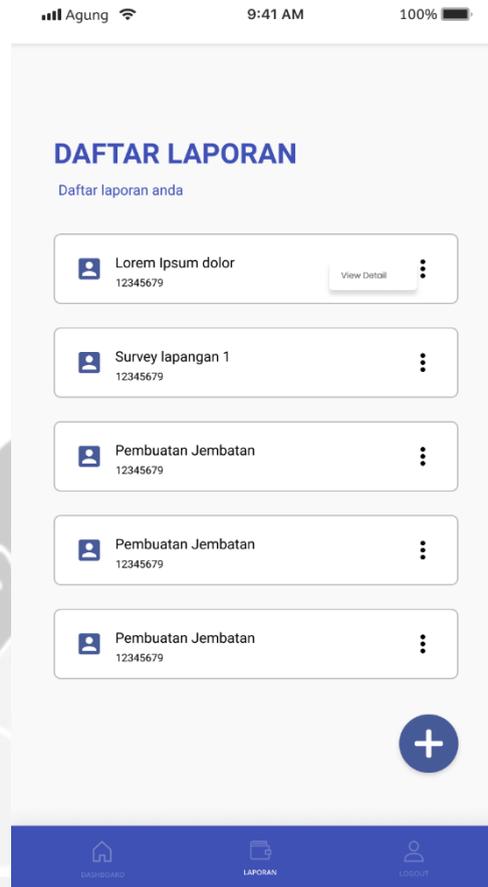
      dispatch(UserOverviewLocationFetchDone(location: location));
    });
}

```

Gambar 5.10 Potongan Kode Untuk Menampilkan Data Secara *Realtime*

Pada gambar 5.10 dijelaskan proses *update* lokasi disesuaikan dengan koordinat *latitude* dan *longitude* dari *surveyor* yang *login* tersebut. Pada potongan kode pada Gambar 5.10, *geolocator* akan mengambil posisi terkini menggunakan fungsi `getCurrentPosition(desiredAccuracy:LocationAccuracy.best)` dengan mengambil posisi terbaik dari *surveyor* yang sedang *login*. Posisi terbaik yang dimaksud adalah ketepatan akurasi yaitu akurasi *latitude* dan *longitude*-nya hingga 7 angka di belakang koma, sehingga hasil pengambilan lokasinya benar-benar akurat.

### 5.1.5. Tampil Daftar Laporan *Client*



Gambar 5.11 Daftar Laporan Surveyor

Pada halaman ini akan dilakukan *API Call* pada *endpoint* `getMyReports` dan akan memberikan kembalian berupa data laporan *surveyor* yang *login*. Data kumpulan laporan ini yang kemudian dimasukkan ke dalam `ListView`.

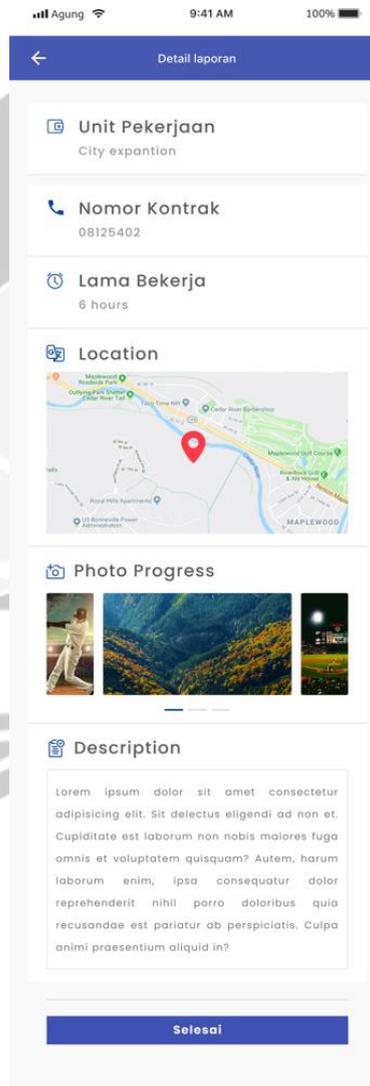
```
child: ListView.builder(  
  itemCount: state.reports.length,  
  itemBuilder: (context, index) => reportCard(context, state.reports[index]),  
), // ListView.builder
```

Gambar 5.12 Kode Untuk Menampilkan `ListView`

Pada potongan kode seperti Gambar 5.12, setiap laporan *surveyor* akan ditampilkan menggunakan `ListView` yang berisi nama laporan dan

nomor kontraknya. Pada halaman ini apabila *surveyor* melakukan klik pada salah satu *ListView*, maka data laporan pada *ListView* tersebut akan diarahkan pada halaman detail laporan. Apabila *surveyor* melakukan klik pada *floating button* tambah, maka halaman akan dialihkan menuju halaman tambah laporan.

### 5.1.6. Tampil Detail Laporan *Surveyor*



Gambar 5.13 Halaman Detail Laporan *Surveyor*

Pada halaman detail laporan, data laporan tersebut berasal dari halaman *list* laporan yang dipilih. Sehingga pada halaman ini hanya

diperlukan proses penampilan data laporan tersebut melalui BLoC ReportDetailBloc seperti kode pada Gambar 5.14.

```
class _ReportDetailScreenState extends State<ReportDetailScreen> {  
  
  @override  
  Widget build(BuildContext context) {  
    final ReportDetailBloc reportDetailBloc = BlocProvider.of<ReportDetailBloc>(context);  
    reportDetailBloc.dispatch(SetData(report: widget.report));  
  
    return Scaffold(  
      appBar: AppBar(  
        title: Text('Detil Laporan'),  
      ), // AppBar  
      body: SafeArea(  
        child: contentBody(),  
      ), // SafeArea  
    ); // Scaffold  
  }  
}
```

Gambar 5.14 Potongan Kode ReportDetailBloc

Pada Gambar 5.14, fungsi ReportDetailBloc berisi data laporan dari BlocProvider yang berisi parameter context. Pada parameter context tersebut, berisi *object* data-data laporan yang sudah dimasukkan oleh *surveyor* sebelumnya.

```
DetailLayout(  
  icon: Icons.account_balance_wallet,  
  title: 'Unit Pekerjaan',  
  value: state.report.jobName  
)
```

Gambar 5.15 Potongan Kode Penampilan Detail Laporan Dalam *TextView*

Pada Gambar 5.15, menjelaskan data laporan diambil dari DetailReportState. Sebagai contoh dalam pengambilan data unit pekerjaan, data didapat dari `state.report.jobName`.

### 5.1.7. Tampil Data Profil *Surveyor*

Agung 9:41 AM 100%

Profile

Nama Pengguna

Nama Lengkap

Unit Kerja

Posisi

Email

KTP

Nomor Telepon

Kata sandi

Konfirmasi Kata sandi

Logout

Gambar 5.16 Halaman Profil Pengguna

Pada halaman profil pengguna, pertama kali program akan melakukan *API Call* pada *endpoint* *whoami* yang berisi data *surveyor* yang sedang *login*. Data-data tersebut diambil dengan mengakses *UserOverviewState* melalui *UserOverviewBloc* seperti pada Gambar 5.17.

```
body: BlocBuilder<UserOverviewBloc, UserOverviewState>(
  builder: (context, state) {
    if (state.user == null || state.isLoading) {
      return Center(
        child: Text('Memuat data'),
      ); // Center
    }
  }
)
```

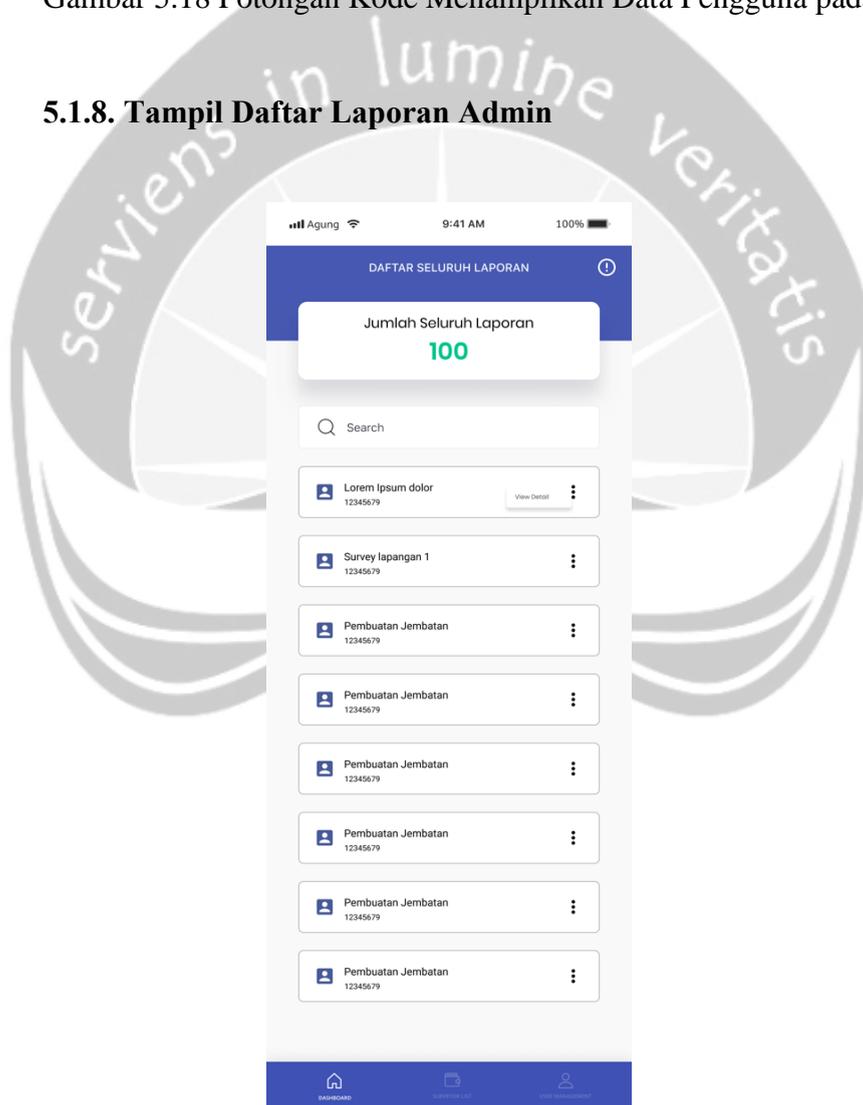
Gambar 5.17 Kode *UserOverViewBloc*

Pada gambar kode 5.17 `UserOverviewState` mengambil informasi detail dari pengguna melalui `UserOverviewBloc`. Data tersebut berisi *object* data pengguna yang sedang *login*, lalu *object* tersebut akan ditampilkan dalam bentuk `TextView` menggunakan kode seperti pada Gambar 5.18.

```
child: ListTile(  
  title: Text('Nama Lengkap'),  
  subtitle: Text(state.user.fullName),  
), // ListTile
```

Gambar 5.18 Potongan Kode Menampilkan Data Pengguna pada `TextView`

### 5.1.8. Tampil Daftar Laporan Admin



Gambar 5.19 Tampil Daftar Laporan Seluruh *Surveyor*

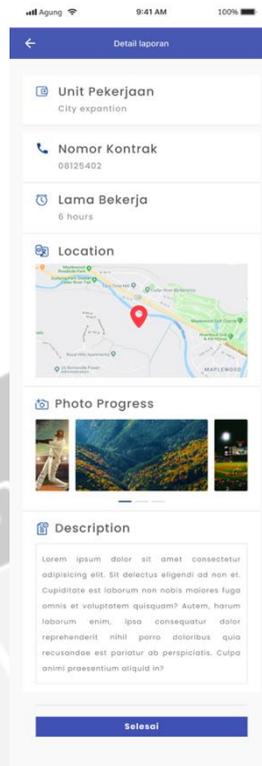
Pada Gambar 5.19, dilakukan pemanggilan pada *endpoint* `getAllReportQueries`. *Endpoint* ini akan mengembalikan semua data laporan *surveyor* yang ada.

```
child: ListView.builder(  
  itemCount: state.filteredReports.length,  
  itemBuilder: (context, position) {  
    return reportCard(context, state.filteredReports[position]);  
  }  
), // ListView.builder
```

Gambar 5.20 Kode Untuk Menghitung Laporan *Surveyor* dan Menampilkan Dalam `CardView`

Pada Gambar 5.20, terdapat fungsi untuk menghitung jumlah laporan dari seluruh laporan *surveyor*. Data jumlah laporan tersebut ditampilkan pada `cardView` jumlah laporan. Data setiap laporan akan ditampilkan menggunakan fungsi `reportCard` dan laporan akan ditampilkan dalam bentuk *list*.

### 5.1.9. Tampil Detail Laporan Admin



Gambar 5.21 Detail Laporan

Pada halaman detail laporan, data laporan tersebut berasal dari halaman *list* laporan yang dipilih. Sehingga pada halaman ini hanya diperlukan proses penampilan data laporan tersebut melalui BLoC `ReportDetailBloc` seperti pada Gambar 5.22.

```
class _ReportDetailScreenState extends State<ReportDetailScreen> {  
  
  @override  
  Widget build(BuildContext context) {  
    final ReportDetailBloc reportDetailBloc = BlocProvider.of<ReportDetailBloc>(context);  
    reportDetailBloc.dispatch(SetData(report: widget.report));  
  
    return Scaffold(  
      appBar: AppBar(  
        title: Text('Detil Laporan'),  
      ), // AppBar  
      body: SafeArea(  
        child: contentBody(),  
      ), // SafeArea  
    ); // Scaffold  
  }  
}
```

Gambar 5.22 Potongan Kode Detail Laporan

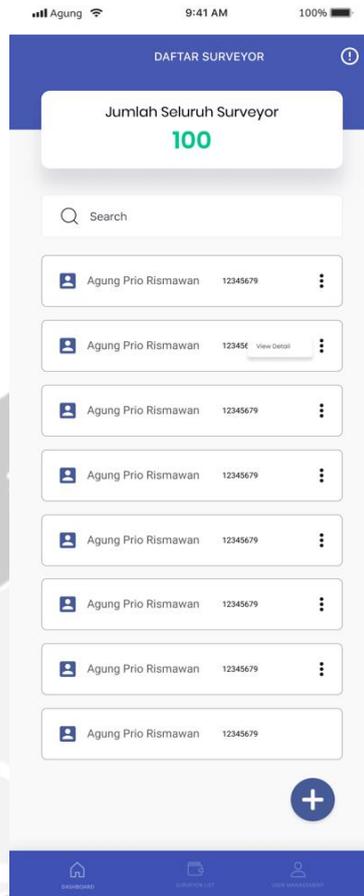
Pada Gambar 5.22, dijelaskan bahwa BLoC `ReportDetailBloc` berisi data laporan dari `BlocProvider` yang berisi parameter `context`. Pada parameter tersebut sebagai parameter penampung data yang berisi *object* data-data laporan yang sudah dimasukkan oleh *surveyor* sebelumnya.

```
DetailLayout(  
  icon: Icons.account_balance_wallet,  
  title: 'Unit Pekerjaan',  
  value: state.report.jobName  
)
```

Gambar 5.23 Potongan Kode Penampilan Detail Laporan Dalam `TextView`

Pada Gambar 5.23, data laporan yang ingin ditampilkan diambil dari `DetailReportState`. Sebagai contoh dalam pengambilan data unit pekerjaan, data didapat dari `state.report.jobName`.

### 5.1.10. Tampil Daftar *Surveyor* Admin



Gambar 5.24 Halaman Tampil Daftar *Surveyor*

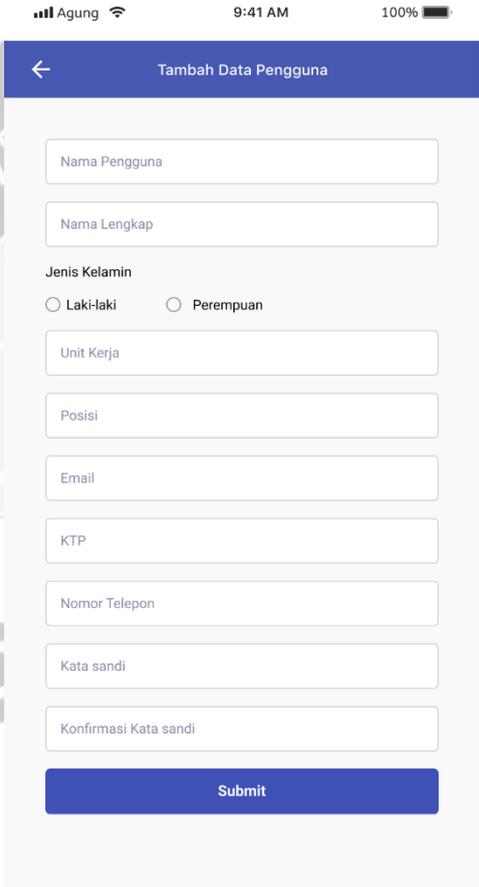
Pada Gambar 5.24, akan dilakukan *API Call* pada *endpoint* `getAllUsersQueries`. Data semua *surveyor* ini kemudian dihitung jumlahnya dan ditampilkan pada *card* jumlah *surveyor*. Setelah itu, data ditampilkan pada *list* menggunakan kode seperti pada Gambar 5.25.

```
child: ListView.builder(  
  itemCount: state.filteredUser.length,  
  itemBuilder: (context, position) {  
    return SurveyorCard(user: state.filteredUser[position]);  
  }  
), // ListView.builder
```

Gambar 5.25 Potongan Kode Untuk Menampilkan Jumlah *Surveyor* dengan *cardView*

Pada potongan kode pada Gambar 5.25, jumlah *surveyor* yang terdaftar dalam aplikasi akan dihitung dan data *surveyor* tersebut akan ditampilkan dalam bentuk `cardView`. Kemudian apabila admin menekan tombol *floating button* tambah, maka akan dialihkan ke halaman `addSurveyor`. Apabila admin melakukan klik pada *card* yang dipilih, maka akan dialihkan menuju halaman detail *surveyor* yang dipilih.

### 5.1.11. Tambah Data *Surveyor* Admin



The screenshot shows a mobile application interface for adding a user. The title bar is blue with a white back arrow and the text "Tambah Data Pengguna". The status bar at the top shows "Agung", signal strength, Wi-Fi, 9:41 AM, and 100% battery. The form consists of the following fields and controls:

- Nama Pengguna (text input)
- Nama Lengkap (text input)
- Jenis Kelamin (radio buttons for "Laki-laki" and "Perempuan")
- Unit Kerja (text input)
- Posisi (text input)
- Email (text input)
- KTP (text input)
- Nomor Telepon (text input)
- Kata sandi (text input)
- Konfirmasi Kata sandi (text input)
- Submit (blue button)

Gambar 5.26 Tambah Data *Surveyor*

Pada halaman tambah data *surveyor*, terdapat beberapa masukan yang wajib untuk diisi. Ketika admin melakukan *input* data *surveyor* dan menekan tombol *submit*, maka aplikasi akan mengeksekusi kode seperti pada Gambar 5.27.

```

    if (!state.isRegistering && state.isRegisteringSuccess) {
      Navigator.pushNamed(context, '/');
    }
  }
  _handleRegisterStarted() async {
    try {
      var form = currentState.form;

      final MutationOptions options = MutationOptions(
        document: RegisterMutation,
        variables: <String, dynamic>{
          'username': form.username,
          'password': form.password,
          'confirmPassword': form.confirmPassword,
          'roleId': form.roleId,
          'jobUnit': form.jobUnit,
          'jobPosition': form.jobPosition,
          'ktp': form.ktp,
          'fullName': form.fullName,
          'gender': form.gender,
          'email': form.email,
          'phone': form.phone,
        },
      );

      QueryResult result = await _gqlClient.mutate(options);

      if (result.hasErrors) {
        print('ckck ' + result.errors.first.toString());
        this.dispatch(RegisterError(error: InvalidRegisterDataException()));
        return;
      }

      this.dispatch(RegisterSuccess());
    } catch (e) {
      print('ckck ' + e.toString());
      this.dispatch(RegisterError(error: RegistrationFailedException()));
    }
  }

  RegisterState _mapRegisterSuccess() =>
    currentState.copyWith(
      isRegistering: false,
      isRegisteringSuccess: true,
      form: RegisterForm.initial(),
    );

    if (!state.isRegistering && state.isRegisteringSuccess) {
      Navigator.pushNamed(context, '/');
    }
  }

```

Gambar 5.27 Kode Registrasi BLoC

Pada potongan kode pada Gambar 5.27, data-data *surveyor* akan diinisialisasi pada BLoC `_handleRegisterStarted()`. Selanjutnya pada kelas `_handleRegisterStarted()`, data tersebut diberi nilai sesuai dengan *params* yang ada pada API lalu akan disimpan pada variabel `result`. Variabel `result` diberi kelas `await` yang menunggu proses pengisian *form* pada halaman *input* data *surveyor*. Apabila sudah selesai

maka hasil isian tersebut akan dilakukan mutasi data menggunakan fungsi `_graphqlClient.mutate(options)`. Langkah selanjutnya adalah program akan melakukan pengecekan apabila terdapat data yang kosong atau belum lengkap, maka akan dimasukkan pada *exception* `invalidRegisterException()` dan apabila sukses maka akan memanggil BLoC *event* `registrationSuccess()`. Data dari *surveyor* yang didaftarkan oleh admin akan disimpan pada basis data.

### 5.1.12. Ubah Status *Surveyor*



Gambar 5. 28 Halaman Pengelolaan *Surveyor*

Ketika admin mengakses halaman pengelolaan *surveyor*, aplikasi akan melakukan *API Call* menggunakan *endpoint* `getAllusersQuery`. Pemanggilan *endpoint* dilakukan melalui `dispatch` pada `UserBloc`.

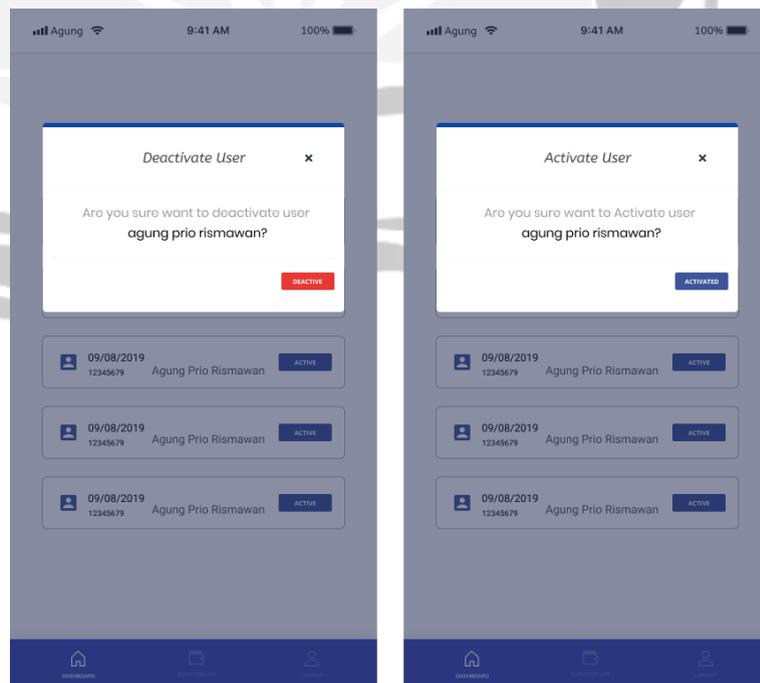
Semua data *surveyor* diambil melalui `UserState` yang berisi seluruh daftar *surveyor* yang sudah terdaftar dalam aplikasi.

```
List<User> get activeUsers {
  return users.where((user) => user.isActive).toList();
}

List<User> get inactiveUsers {
  return users.where((user) => !user.isActive).toList();
}
```

Gambar 5.29 Kode Untuk Mendapatkan Status *Activate/Deactivate Surveyor*

Potongan kode pada Gambar 5.29 menunjukkan 2 fungsi untuk mengambil *user* atau *surveyor* yang sudah terdaftar berdasarkan status *active* maupun *inactive*. *List user* yang telah didapatkan dari kedua fungsi tersebut akan ditampilkan dalam bentuk *list* dengan menggunakan fungsi `userCardview`.



Gambar 5.30 Halaman *Activate dan Deactivate User*

Ketika admin melakukan klik pada tombol *active/inactive* program akan mengeksekusi kode pada Gambar 5.31.

```
_handleActivateUser(User user) async {
  try {
    final MutationOptions options = MutationOptions(
      document: ActivateUserMutation,
      variables: <String, dynamic>{
        "id": user.id,
      }
    );

    QueryResult result = await _graphqlClient.mutate(options);

    if (result.hasErrors) {
      this.dispatch(UserStatusUpdateError(error: UpdateUserDataFailedException()));
      return;
    }

    this.dispatch(UsersFetchStarted());
  } catch(_) {
    this.dispatch(UserStatusUpdateError(error: UpdateUserDataFailedException()));
  }
}

_handleDeactivateUser(User user) async {
  try {
    final MutationOptions options = MutationOptions(
      document: DeactivateUserMutation,
      variables: <String, dynamic>{
        "id": user.id,
      }
    );

    QueryResult result = await _graphqlClient.mutate(options);

    if (result.hasErrors) {
      this.dispatch(UserStatusUpdateError(error: UpdateUserDataFailedException()));
      return;
    }

    this.dispatch(UsersFetchStarted());
  } catch(_) {
    this.dispatch(UserStatusUpdateError(error: UpdateUserDataFailedException()));
  }
}
```

Gambar 5.31 Potongan Kode Untuk *Activated/Deactivated Surveyor*

Pada potongan kode pada Gambar 5.31, digunakan untuk melakukan *activate/deactivate surveyor*. Pertama kali aplikasi akan mengeksekusi fungsi `_handleActivateUser` atau `_handleDeactivateUser`, sesuai dengan *request* dari admin. Pada kedua fungsi tersebut akan melakukan *update* status yang berisi data *Boolean*

berdasarkan *id* dari *surveyor* yang dipilih. Data yang sudah di-*update* tersebut akan ditampung dalam variabel `result` akan melakukan API *Request* pada *endpoint* `activateUserMutasi` dan `DeActivateUserMutasi`. Hasil dari *update* tersebut akan dikembalikan lagi oleh API dan akan mengambil hasil terbaru menggunakan *endpoint* `getAllusersQuery`.



## 5.2. Pengujian Fungsionalitas Perangkat Lunak

Pada tabel 5.1 merupakan tabel pengujian fungsionalitas aplikasi MOTS-PeR. Metode pengujian aplikasi MOTS-PeR *Client* dan MOTS-PeR Admin menggunakan metode pengujian *Black Box*. Metode *Black Box* merupakan metode pengujian yang fokus pada hasil program dan tidak melihat proses yang berjalan. Metode *Black Box* ini menggunakan teknik *Equivalence Partitions*. Teknik ini merupakan teknik yang membagi domain *input* menjadi sebuah kelas data di mana *test case* akan diambil. Proses pembuatan *test case* yaitu berdasarkan *use case* yang terdapat pada poin 4.4 fungsi produk.

Tabel 5.1. Pengujian Fungsionalitas Perangkat Lunak

Identifikasi	Deskripsi	Prosedur Pengujian	Masukan	Keluaran yang Diharapkan	Hasil Yang Didapat	Kesimpulan
P-01-01	Pengujian terhadap menampilkan data daftar laporan pada halaman admin.	- Pilih menu daftar laporan	Tidak ada masukan	Data-data daftar laporan ditampilkan dalam bentuk <i>list</i> . Data yang ditampilkan adalah jumlah seluruh laporan, nama, dan id pengguna.	Data-data <i>layout</i> ditampilkan dalam bentuk <i>list</i> . Data yang ditampilkan adalah jumlah seluruh laporan, nama, dan id pengguna.	Handal

P-01-02	Pengujian terhadap pencarian data pada halaman daftar laporan di halaman admin.	- Pilih menu daftar laporan - Tekan tombol <i>search</i> bar	<i>Search : seller</i>	Sistem menampilkan data <i>seller</i> .	Sistem menampilkan data <i>seller</i> .	Handal
P-01-03	Pengujian terhadap pencarian pada halaman daftar laporan di halaman admin.	- Pilih menu daftar laporan - Tekan tombol <i>search</i> bar	<i>Search : babarsari</i>	Sistem tidak menampilkan data yang dicari.	Sistem tidak menampilkan data yang dicari.	Handal
P-01-04	Pengujian terhadap tampil detail laporan pada halaman admin.	- Pilih menu daftar laporan - Tekan tombol <i>detail report</i>	Tidak ada masukan	Data-data daftar laporan ditampilkan sesuai dengan <i>list</i> data yang dipilih. Data yang ditampilkan adalah map lokasi laporan, pekerjaan yang dikerjakan, nomor kontrak, durasi, deskripsi, dan lampiran berupa <i>image</i> .	Data-data daftar laporan dalam bentuk <i>list</i> . Data yang ditampilkan adalah map lokasi laporan, pekerjaan yang dikerjakan, nomor kontrak, durasi, deskripsi, dan lampiran berupa <i>image</i> .	Handal
P-02-05	Pengujian terhadap menampilkan data daftar	- Pilih menu daftar <i>surveyor</i>	Tidak ada masukan	Data-data daftar laporan ditampilkan	Data-data layout ditampilkan dalam	

	<i>surveyor</i> pada halaman admin.			dalam bentuk <i>list</i> . Data yang ditampilkan adalah nama dan <i>id</i> pengguna.	bentuk <i>list</i> . Data yang ditampilkan adalah nama dan <i>id</i> pengguna.	Handal
P-01-06	Pengujian terhadap pencarian pada halaman daftar <i>surveyor</i> di halaman admin.	- Pilih menu daftar <i>surveyor</i> - Tekan tombol <i>search bar</i>	<i>Search</i> : agungprio	Sistem menampilkan data agungprio.	Sistem menampilkan data agungprio.	Handal
P-01-07	Pengujian terhadap pencarian pada halaman daftar <i>surveyor</i> di halaman admin.	- Pilih menu daftar laporan - Tekan tombol <i>search bar</i>	<i>Search</i> : besty	Sistem tidak menampilkan data yang dicari.	Sistem tidak menampilkan data yang dicari.	Handal
P-01-08	Pengujian terhadap tampil detail <i>surveyor</i> pada halaman admin.	- Pilih menu daftar <i>surveyor</i> - Tekan tombol <i>detail surveyor</i>	Tidak ada masukan	Data–data daftar laporan ditampilkan sesuai dengan <i>list</i> data yang dipilih. Data yang ditampilkan adalah Nama Lengkap, <i>Username</i> , Email, Jenis Kelamin, Posisi Pekerjaan, Unit	Data–data daftar laporan dalam bentuk <i>list</i> . Data yang ditampilkan adalah Nama Lengkap, <i>Username</i> , Email, Jenis Kelamin, Posisi Pekerjaan, Unit Pekerjaan, KTP, dan	Handal

				Pekerjaan, KTP, dan Nomor Telepon.	Nomor Telepon.	
P-01-09	Pengujian terhadap tambah <i>surveyor</i> pada halaman admin.	- Pilih menu daftar <i>surveyor</i> - Tekan <i>icon</i> tambah “+”	- Nama Pengguna : Ray - Nama Lengkap : Hendrikus Adi Purnama - Jenis Kelamin : Laki-laki - Unit Kerja : <i>Software</i> - Posisi : <i>Software Engineer</i> - Email : <u><i>Hendrikusray@gmail.com</i></u> - KTP : 11111111 - No. Telepon : 08121281921 - Kata Sandi : HendrikusRay - Konfirmasi Kata Sandi : HendrikusRay	Data <i>surveyor</i> berhasil ditambahkan ke basis data.	Data <i>surveyor</i> berhasil ditambahkan ke basis data.	Handal
P-01-10	Pengujian terhadap	- Pilih menu daftar	- Nama Pengguna :	Sistem menampilkan	Sistem menampilkan	

	tambah <i>surveyor</i> pada halaman admin.	<i>surveyor</i> - Tekan <i>icon</i> tambah “+”	kosong - Nama Lengkap : Hendrikus Adi Purnama - Jenis Kelamin : kosong - Unit Kerja : <i>Software</i> - Posisi : kosong - Email : <u>Hendrikusray@gmail.com</u> - KTP : 11111111 - No. Telepon : 08121281921 - Kata Sandi : HendrikusRay - Konfirmasi Kata Sandi : HendrikusRay	pesan ‘Data yang diberikan salah/kosong’.	pesan ‘Data yang diberikan salah/kosong’.	Handal
P-01-11	Pengujian terhadap menampilkan daftar data kelola <i>surveyor</i> pada halaman admin.	- Pilih menu daftar kelola <i>surveyor</i>	Tidak ada masukan	Data–data daftar laporan ditampilkan dalam bentuk <i>list</i> . Data yang ditampilkan adalah	Data–data <i>layout</i> ditampilkan dalam bentuk <i>list</i> . Data yang ditampilkan adalah jumlah data yang aktif	Handal

				jumlah data yang aktif atau tidak aktif, nama pengguna, id pengguna, id pengguna, serta <i>button active/inactive</i> .	atau tidak aktif, nama pengguna, id pengguna, serta <i>button active/inactive</i> .	
P-01-12	Pengujian terhadap pencarian pada halaman daftar kelola <i>surveyor</i> di halaman admin.	- Pilih menu daftar kelola <i>surveyor</i> - Tekan tombol <i>search bar</i>	<i>Search : seller</i>	Sistem menampilkan data <i>seller</i> .	Sistem menampilkan data <i>seller</i> .	Handal
P-01-13	Pengujian terhadap pencarian pada halaman daftar kelola <i>surveyor</i> di halaman admin.	- Pilih menu daftar kelola <i>surveyor</i> - Tekan tombol <i>search bar</i>	<i>Search : besty</i>	Sistem tidak menampilkan data yang dicari.	Sistem tidak menampilkan data yang dicari.	Handal
P-01-14	Pengujian terhadap ubah status data <i>surveyor</i> pada halaman admin.	- Pilih menu daftar kelola <i>surveyor</i> - Tekan tombol <i>action</i> pada <i>list</i>	Menekan <i>toggle button</i>	Status <i>list</i> akan berubah.	Status <i>list</i> akan berubah.	Handal
P-02-01	Pengujian terhadap menampilkan data daftar laporan pada halaman <i>surveyor</i> .	- Pilih menu daftar laporan	Tidak ada masukan	Data–data daftar laporan ditampilkan dalam bentuk <i>list</i> . Data yang ditampilkan adalah	Data–data <i>layout</i> ditampilkan dalam bentuk <i>list</i> . Data yang ditampilkan adalah nama pekerjaan dan	Handal

				laporan <i>surveyor</i> yang <i>login</i> dan ditampilkan dalam bentuk <i>list</i> yang berisi nama pekerjaan dan nomor kontrak dari laporan <i>surveyor</i> yang sedang <i>login</i> tersebut.	nomor kontrak.	
P-02-02	Pengujian terhadap menampilkan <i>dashboard surveyor</i>	- Pilih menu beranda	Tidak ada Masukan	Data-data diri dan jumlah laporan <i>surveyor</i> ditampilkan dalam bentuk <i>card view</i> dan terdapat peta yang merepresentasikan lokasi dia saat ini.	Data-data <i>layout</i> ditampilkan dalam bentuk <i>card view</i> . Data yang ditampilkan adalah jumlah seluruh laporan, nama, dan id pengguna.	Handal
P-02-03	Pengujian terhadap tambah data laporan pada halaman <i>surveyor</i>	- Pilih menu daftar laporan - Tekan <i>icon</i> tambah “+”	- Nama pekerjaan : <i>Survey tambak udang</i> - Nomor Kontrak: 89011 - Lama bekerja (jam) : 6 - Deskripsi : “ini adalah <i>survey</i> tambak udang	Data laporan <i>surveyor</i> berhasil ditambahkan ke basis data.	Data laporan <i>surveyor</i> berhasil ditambahkan ke basis data.	Handal

			testing 1” - Lampiran gambar : gambar1.jpg, gambar2.jpg			
P-02-04	Pengujian terhadap tampil profil pada halaman <i>surveyor</i>	- Pilih menu profil	Tidak ada masukan	Data yang ditampilkan adalah Nama Lengkap, <i>Username</i> , Email, Jenis Kelamin, Posisi Pekerjaan, Unit Pekerjaan, KTP, dan No. Telepon.	Data yang ditampilkan adalah Nama Lengkap, <i>Username</i> , Email, Jenis Kelamin, Posisi Pekerjaan, Unit Pekerjaan, KTP, dan No. Telepon.	Handal

### 5.3. Hasil Pengujian Terhadap Pengguna

Pada tabel 5.2 dan 5.3 merupakan pengujian terhadap pengguna aplikasi MOTS-PeR. Pengujian aplikasi MOTS-PeR kepada pengguna menggunakan metode wawancara serta observasi penggunaan aplikasi. Pengujian ini dilakukan untuk menguji resiliensi serta *usability* dari aplikasi secara keseluruhan. Responden yang ikut berpartisipasi dalam pengujian ini melibatkan 15 orang. 15 orang tersebut kebanyakan bekerja sebagai karyawan swasta dari perusahaan yang berbeda-beda. Ada yang bekerja sebagai *QA Engineer* di DC Indonesia, *Backend Developer* di PT Global Urban Esensial, *Front End Developer* di Privy ID, dan lain-lain. 35% *user* yang diuji adalah perempuan dan 65% lainnya adalah laki-laki.

Pengujian yang melibatkan 15 orang ini dibagi menjadi beberapa pengujian. Sepuluh orang menguji halaman *surveyor* dengan kriteria pengguna mempunyai *smartphone* Android dengan SDK minimal Oreo 8.1, dan 5 orang menguji halaman admin. Penilaian dilakukan dengan kriteria poin sebagai berikut:

1. Sangat Tidak Setuju
2. Tidak Setuju
3. Cukup
4. Setuju
5. Sangat Setuju

Tabel 5.2. Hasil Pengujian Pada Halaman Surveyor

No.	Pertanyaan	1	2	3	4	5
1.	Apakah aplikasi bisa di- <i>install</i> oleh pengguna?	0	0	0	0	10
2.	Apakah aplikasi ini berjalan dengan baik dari segi fungsi?	0	0	1	7	2
3.	Apakah terjadi <i>error</i> saat menggunakan aplikasi?	6	0	0	4	0
4.	Apakah <i>error</i> tersebut mengganggu untuk melanjutkan pada proses berikutnya?	10	0	0	0	0
5.	Apakah aplikasi ini <i>user friendly</i> secara <i>user interface</i> ?	0	0	4	4	2
6.	Apakah waktu akses lambat?	8	1	1	0	0

7.	Apakah pada halaman pengelolaan laporan dapat mempermudah <i>surveyor</i> dalam melaporkan kejadian di lapangan?	0	0	0	2	8
----	--	---	---	---	---	---

Tabel 5. 3. Hasil Pengujian Pada Halaman Admin

No.	Pertanyaan	1	2	3	4	5
1.	Apakah aplikasi bisa di- <i>install</i> oleh pengguna?	0	0	0	0	5
2.	Apakah aplikasi ini berjalan dengan baik dari segi fungsi?	0	0	1	1	3
3.	Apakah terjadi <i>error</i> saat menggunakan aplikasi?	0	0	0	1	4
4.	Apakah <i>error</i> tersebut mengganggu untuk melanjutkan pada proses berikutnya?	5	0	0	0	0
5.	Apakah aplikasi ini <i>user friendly</i> secara <i>user interface</i> ?	0	0	0	0	5
6.	Apakah waktu akses lambat?	5	0	0	0	0
7.	Apakah pada halaman admin dapat membantu admin untuk mengelola laporan, dan manajemen data <i>surveyor</i> ?	0	0	0	0	5

#### 5.4. Analisis Kelebihan dan Kekurangan Aplikasi

Berdasarkan pengujian fungsionalitas dan pengujian pengguna, dapat disimpulkan bahwa aplikasi MOTS-PeR *Surveyor* dan Admin mempunyai beberapa kelebihan, antara lain:

1. Dapat mengakomodasi pelaporan secara digital yang dilakukan oleh *surveyor* di lapangan.
2. Dapat melacak data lokasi *surveyor* pada saat di lapangan secara *realtime* menggunakan Google Maps.
3. Desain antarmuka yang mudah dipahami oleh pengguna.
4. Manajemen data laporan menjadi mudah untuk dikelola dan terpusat

Terdapat pula kekurangan dari aplikasi MOTS-PeR ini, antara lain:

1. Belum dapat menyediakan fitur untuk *cache* data pelaporan dan

informasi *user*, sehingga ketika kondisi *offline* maka data tidak akan ditampilkan.

Pada bab implementasi dan pengujian perangkat lunak ini telah dijelaskan mengenai definisi aplikasi, implementasi aplikasi, dan hasil pengujian aplikasi. Pada bab selanjutnya, yaitu penutup, akan diberikan kesimpulan dan saran yang didapatkan selama pembuatan tugas akhir ini.



## BAB VI. PENUTUP

### 6.1. Kesimpulan

Berdasarkan penelitian yang dilakukan penulis dengan menganalisis aplikasi MOTS-PeR berdasarkan teori-teori yang digunakan dalam penelitian, maka dapat ditarik kesimpulan dari tugas akhir ini antara lain:

1. MOTS-PeR dapat menyediakan fitur untuk melakukan pelaporan *surveyor* secara digital dan dapat melacak data lokasi laporan secara *realtime* berdasarkan lokasi *surveyor*. Lokasi tersebut akan ditampilkan dalam bentuk peta digital pada aplikasi. Dengan adanya fitur ini lokasi laporan *surveyor* dapat dilacak secara *realtime* sehingga mengurangi kecurangan *surveyor* apabila melakukan pelaporan tidak sesuai dengan lokasi yang sudah ditentukan.
2. Saat berada di lapangan, *surveyor* akan melakukan beberapa masukan data yang terdiri dari nama pekerjaan yang dilakukan *surveyor*, deskripsi singkat, lampiran gambar, serta lokasi pelaporan saat di lapangan. Hasil dari inputan *surveyor* kemudian dijadikan sebagai laporan untuk diserahkan kepada kementerian kelautan dan perikanan republik indonesia. Hasil output yang berupa laporan dari aplikasi ini, membantu pihak Kementerian Kelautan dan Perikanan Republik Indonesia untuk melakukan pertimbangan pembangunan infrastruktur atau perbaikan komponen-komponen infrastruktur yang sudah ada.
3. Penggunaan arsitektur BLoC pada Flutter mempermudah dalam implementasi dan perancangan perangkat lunak MOTS-PeR. Arsitektur BLoC memiliki peran yang sangat penting pada tahap pengkodean karena BLoC dapat memisahkan komponen *logic* dan *presentation*. Pemisahan komponen *logic* dan *presentation* mempermudah proses *maintenance* kode apabila aplikasi sudah masuk kedalam tahap *production*.

### 6.2. Saran

Dari hasil proses analisis, perancangan, implementasi, hingga pengujian pada penelitian ini, didapatkan saran pengembangan lebih lanjut untuk MOTS-

PeR yaitu kemampuan untuk *cache* data pelaporan dan informasi *user* pada aplikasi MOTS-PeR *surveyor*. Dengan adanya fitur ini ketika kondisi *device surveyor offline* maka data-data *surveyor* masih bisa diakses dan data pelaporan *surveyor* akan disimpan pada penyimpanan lokal.



## DAFTAR PUSTAKA

- [1] Pratama, O., 2020. KKP | Kementerian Kelautan Dan Perikanan. [online] Kkp.go.id. Tersedia di: <<https://kkp.go.id/djprl/artikel/21045-konservasi-perairan-sebagai-upaya-menjaga-potensi-kelautan-dan-perikanan-indonesia>> [Diakses pada 31 July 2020].
- [2] O. E. Mambu, Y. D. Y. Rindengan, and S. D. S. Karouw, “Pengembangan Aplikasi E-Report Layanan Masyarakat untuk Manado Smart City,” *J. Tek. Inform.*, vol. 8, no. 1, 2016, doi: 10.35793/jti.8.1.2016.12233.
- [3] P. S. Haloho and T. Wuruk Pribadi, “Perancangan Aplikasi Komputer Berbasis Android untuk Survei Kondisi Kapal oleh Owner Surveyor,” *J. Tek. ITS*, vol. 5, no. 2, 2017, doi: 10.12962/j23373539.v5i2.20921.
- [4] G. Hati, A. Suprayogi, and B. Sasmito, “Aplikasi Penanda Lokasi Peta Digital Berbasis Mobile Gis Pada Smartphone Android,” *J. Geod. Undip*, vol. 2, no. 4, p. 82406, 2013.
- [5] R. Somya, “Sistem Monitoring Kendaraan Secara Real Time Berbasis Android menggunakan Teknologi CouchDB di PT. Pura Barutama,” *J. Nas. Teknol. dan Sist. Inf.*, vol. 4, no. 2, pp. 53–60, 2018, doi: 10.25077/teknosi.v4i2.2018.53-60.
- [6] Y. A. Nugroho, “Evaluasi Reaksi Peserta Pada Penyelenggaraan Diklat Diklat Aparatur Kementerian Kelautan Dan Perikanan Evaluation of Participant ’ S Reactions on the Implementation of Basic Training of Functional Position of Fishery ’ S Extension Workers in the Training ,” vol. 13, no. 1, pp. 49–60.
- [7] A. Sabana and H. K. Gestsson, “International Review of Management and Marketing Evaluating the Implementation of Marine and Fisheries Technopark in Tegal, Indonesia Using Project Cycle Management,” *Int. Rev. Manag. Mark.*, vol. 7, no. 5, pp. 28–41, 2017.
- [8] F. Enggar Krisnada and R. Tanone, “Aplikasi Penjualan Tiket Kelas Pelatihan Berbasis Mobile menggunakan Flutter,” *J. Tek. Inform. dan Sist.*

- Inf.*, vol. 5, no. 3, pp. 281–295, 2020, doi: 10.28932/jutisi.v5i3.1865.
- [9] I. O. Widyantara, I. G. A. K. Warmayana, and L. Linawati, “Penerapan Teknologi GPS Tracker Untuk Identifikasi Kondisi Traffik Jalan Raya,” *Maj. Ilm. Teknol. Elektro*, vol. 14, no. 1, pp. 31–35, 2015, doi: 10.24843/mite.2015.v14i01p07.
- [10] L. H. Hoang, “State Management Analyses of the Flutter Application,” no. November, 2019.
- [11] R. Kurniawati, “Pengembangan Sistem Informasi Kependudukan Berbasis Mobile Dan Restful Web Service,” *Semin. Nas. Teknol. Inf. dan Komun.*, vol. 2016, no. SENTIKA, pp. 605–609, 2016.

